



DL205 PLC User Manual

Volume 1 of 2

Manual Number: D2-USER-M

Notes

WARNING

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

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

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

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

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

For additional warranty and safety information, see the Terms and Conditions section of our catalog. If you have any questions concerning the installation or operation of this equipment, or if you need additional information, please call us at 1-770-844-4200.

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

Trademarks

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

Copyright 2017, **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.

⚡ ADVERTENCIA ⚡

Gracias por comprar equipo de automatización de **Automationdirect.com®**. Deseamos que su nuevo equipo de automatización opere de manera segura. Cualquier persona que instale o use este equipo debe leer esta publicación (y cualquier otra publicación pertinente) antes de instalar u operar el equipo.

Para reducir al mínimo el riesgo debido a problemas de seguridad, debe seguir todos los códigos de seguridad locales o nacionales aplicables que regulan la instalación y operación de su equipo. Estos códigos varían de área en área y usualmente cambian con el tiempo. Es su responsabilidad determinar cuales códigos deben ser seguidos y verificar que el equipo, instalación y operación estén en cumplimiento con la revisión mas reciente de estos códigos.

Como mínimo, debe seguir las secciones aplicables del Código Nacional de Incendio, Código Nacional Eléctrico, y los códigos de (NEMA) la Asociación Nacional de Fabricantes Eléctricos de USA. Puede haber oficinas de normas locales o del gobierno que pueden ayudar a determinar cuales códigos y normas son necesarios para una instalación y operación segura.

Si no se siguen todos los códigos y normas aplicables, puede resultar en daños al equipo o lesiones serias a personas. No garantizamos los productos descritos en esta publicación para ser adecuados para su aplicación en particular, ni asumimos ninguna responsabilidad por el diseño de su producto, la instalación u operación.

Nuestros productos no son tolerantes a fallas y no han sido diseñados, fabricados o intencionados para uso o reventa como equipo de control en línea en ambientes peligrosos que requieren una ejecución sin fallas, tales como operación en instalaciones nucleares, sistemas de navegación aérea, o de comunicación, control de tráfico aéreo, máquinas de soporte de vida o sistemas de armamentos en las cuales la falla del producto puede resultar directamente en muerte, heridas personales, o daños físicos o ambientales severos (“Actividades de Alto Riesgo”). **Automationdirect.com** específicamente rechaza cualquier garantía ya sea expresada o implicada para actividades de alto riesgo.

Para información adicional acerca de garantía e información de seguridad, vea la sección de Términos y Condiciones de nuestro catálogo. Si tiene alguna pregunta sobre instalación u operación de este equipo, o si necesita información adicional, por favor llámenos al número 1-770-844-4200 en Estados Unidos.

Esta publicación está basada en la información disponible al momento de impresión. En **Automationdirect.com** nos esforzamos constantemente para mejorar nuestros productos y servicios, así que nos reservamos el derecho de hacer cambios al producto y/o a las publicaciones en cualquier momento sin notificación y sin ninguna obligación. Esta publicación también puede discutir características que no estén disponibles en ciertas revisiones del producto.

Marcas Registradas

Esta publicación puede contener referencias a productos producidos y/u ofrecidos por otras compañías. Los nombres de las compañías y productos pueden tener marcas registradas y son propiedad única de sus respectivos dueños. Automationdirect.com, renuncia cualquier interés propietario en las marcas y nombres de otros.

PROPIEDAD LITERARIA 2017, AUTOMATIONDIRECT.COM® INCORPORATED
Todos los derechos reservados

No se permite copiar, reproducir, o transmitir de ninguna forma ninguna parte de este manual sin previo consentimiento por escrito de **Automationdirect.com® Incorporated**. **Automationdirect.com** retiene los derechos exclusivos a toda la información incluida en este documento. Los usuarios de este equipo pueden copiar este documento solamente para instalar, configurar y mantener el equipo correspondiente. También las instituciones de enseñanza pueden usar este manual para propósitos educativos.

⚡ AVERTISSEMENT ⚡

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

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

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

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

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

Pour des renseignements additionnels touchant la garantie et la sécurité, veuillez consulter la section Modalités et conditions de notre documentation. Si vous avez des questions au sujet de l'installation ou du fonctionnement de cet équipement, ou encore si vous avez besoin de renseignements supplémentaires, n'hésitez pas à nous téléphoner au 1-770-844-4200.

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

Marques de commerce

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

Copyright 2017, 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.

Notes:

VOLUME ONE:

TABLE OF CONTENTS



| | |
|---|-------------|
| Volume One: Table of Contents | i |
| Volume Two: Table of Contents | xi |
| Chapter 1: Getting Started | 1-1 |
| 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 |
| DL205 System Components | 1-4 |
| CPUs | 1-4 |
| Bases | 1-4 |
| I/O Configuration | 1-4 |
| I/O Modules | 1-4 |
| DL205 System Diagrams | 1-5 |
| Programming Methods | 1-7 |
| <i>DirectSOFT</i> Programming for Windows. | 1-7 |
| Handheld Programmer | 1-7 |
| <i>DirectLOGIC™</i> Part Numbering System | 1-8 |
| Quick Start for PLC Validation and Programming | 1-10 |
| Steps to Designing a Successful System | 1-13 |

| | |
|---|-------------|
| Chapter 2: Installation, Wiring and Specifications | 2-1 |
| Safety Guidelines | 2-2 |
| Plan for Safety | 2-2 |
| Three Levels of Protection | 2-3 |
| Emergency Stops | 2-3 |
| Emergency Power Disconnect | 2-4 |
| Orderly System Shutdown | 2-4 |
| Class 1, Division 2, Approval | 2-4 |
| Mounting Guidelines | 2-5 |
| Base Dimensions | 2-5 |
| Panel Mounting and Layout | 2-6 |
| Enclosures | 2-7 |
| Environmental Specifications | 2-8 |
| Power | 2-8 |
| Marine Use | 2-9 |
| Agency Approvals | 2-9 |
| 24 VDC Power Bases | 2-9 |
| Installing DL205 Bases | 2-10 |
| Choosing the Base Type | 2-10 |
| Mounting the Base | 2-10 |
| Using Mounting Rails | 2-11 |
| Installing Components in the Base | 2-12 |
| Base Wiring Guidelines | 2-13 |
| Base Wiring | 2-13 |
| I/O 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 |
| Sinking / Sourcing Concepts | 2-17 |
| I/O “Common” Terminal Concepts | 2-18 |
| Connecting DC I/O to “Solid State” Field Devices | 2-19 |
| Solid State Input Sensors | 2-19 |
| Solid State Output Loads | 2-19 |
| Relay Output Guidelines | 2-21 |
| Relay Outputs – Transient Suppression for Inductive Loads in a Control System | 2-21 |
| I/O Modules Position, Wiring, and Specification | 2-26 |

| | |
|--|-------------|
| Slot Numbering | 2-26 |
| Module Placement Restrictions | 2-26 |
| Special Placement Considerations for Analog Modules | 2-27 |
| Discrete Input Module Status Indicators | 2-27 |
| Color Coding of I/O Modules | 2-27 |
| Wiring the Different Module Connectors | 2-28 |
| I/O Wiring Checklist | 2-29 |
| D2-08ND3, DC Input | 2-30 |
| D2-16ND3-2, DC Input | 2-30 |
| D2-32ND3, DC Input | 2-31 |
| D2-32ND3-2, DC Input | 2-32 |
| D2-08NA-1, AC Input | 2-33 |
| D2-08NA-2, AC Input | 2-34 |
| D2-16NA, AC Input | 2-35 |
| F2-08SIM, Input Simulator | 2-35 |
| D2-04TD1, DC Output | 2-36 |
| D2-08TD1, DC Output | 2-37 |
| D2-08TD2, DC Output | 2-37 |
| D2-16TD1-2, DC Output | 2-38 |
| D2-16TD2-2, DC Output | 2-38 |
| F2-16TD1(2)P, DC Output With Fault Protection | 2-39 |
| F2-16TD1P, DC Output With Fault Protection | 2-40 |
| F2-16TD2P, DC Output with Fault Protection | 2-41 |
| D2-32TD1, DC Output | 2-42 |
| D2-32TD2, DC Output | 2-42 |
| F2-08TA, AC Output | 2-43 |
| D2-08TA, AC Output | 2-43 |
| D2-12TA, AC Output | 2-44 |
| D2-04TRS, Relay Output | 2-45 |
| D2-08TR, Relay Output | 2-46 |
| F2-08TR, Relay Output | 2-47 |

| | |
|---|-------------|
| F2-08TRS, Relay Output | 2-48 |
| D2-12TR, Relay Output | 2-49 |
| D2-08CDR 4 pt., DC Input / 4pt., Relay Output | 2-50 |
| Glossary of Specification Terms | 2-51 |
| Chapter 3: CPU Specifications and Operations | 3-1 |
| CPU Overview | 3-2 |
| General CPU Features | 3-2 |
| DL230 CPU Features | 3-2 |
| DL240 CPU Features | 3-2 |
| DL250-1 CPU Features | 3-3 |
| DL260 CPU Features | 3-3 |
| CPU General Specifications | 3-4 |
| CPU Base Electrical Specifications | 3-5 |
| CPU Hardware Setup | 3-6 |
| Communication Port Pinout Diagrams | 3-6 |
| Port 1 Specifications | 3-7 |
| Port 2 Specifications | 3-8 |
| Selecting the Program Storage Media | 3-9 |
| Built-in EEPROM | 3-9 |
| EEPROM Sizes | 3-9 |
| EEPROM Operations | 3-9 |
| Installing the CPU | 3-10 |
| Connecting the Programming Devices | 3-10 |
| CPU Setup Information | 3-11 |
| Status Indicators | 3-12 |
| Mode Switch Functions | 3-12 |
| Changing Modes in the DL205 PLC | 3-13 |
| Mode of Operation at Power-up | 3-13 |
| Using Battery Backup | 3-14 |
| DL230 and DL240 | 3-14 |
| DL250-1 and DL260 | 3-14 |
| Battery Backup | 3-14 |
| Auxiliary Functions | 3-15 |
| Clearing an Existing Program | 3-16 |

| | |
|--|-------------|
| Initializing System Memory | 3-16 |
| Setting the Clock and Calendar | 3-16 |
| Setting the CPU Network Address | 3-17 |
| Setting Retentive Memory Ranges | 3-17 |
| Using a Password | 3-18 |
| Setting the Analog Potentiometer Ranges | 3-19 |
| CPU Operation | 3-21 |
| CPU Operating System | 3-21 |
| Program Mode Operation | 3-22 |
| Run Mode Operation | 3-22 |
| Read Inputs | 3-23 |
| Read Inputs from Specialty and Remote I/O | 3-23 |
| Service Peripherals and Force I/O | 3-23 |
| CPU Bus Communication | 3-24 |
| Update Clock, Special Relays and Special Registers | 3-24 |
| Solve Application Program | 3-25 |
| Solve PID Loop Equations | 3-25 |
| Write Outputs | 3-25 |
| Write Outputs to Specialty and Remote I/O | 3-26 |
| Diagnostics | 3-26 |
| I/O Response Time | 3-27 |
| Is Timing Important for Your Application? | 3-27 |
| Normal Minimum I/O Response | 3-27 |
| Normal Maximum I/O Response | 3-27 |
| Improving Response Time | 3-28 |
| CPU Scan Time Considerations | 3-29 |
| Initialization Process | 3-30 |
| Reading Inputs | 3-30 |
| Reading Inputs from Specialty I/O | 3-31 |
| Service Peripherals | 3-31 |
| CPU Bus Communication | 3-32 |
| Update Clock/Calendar, Special Relays, Special Registers | 3-32 |
| Writing Outputs | 3-32 |
| Writing Outputs to Specialty I/O | 3-33 |
| Diagnostics | 3-33 |
| Application Program Execution | 3-34 |

| | |
|---|-------------|
| PLC Numbering Systems | 3–35 |
| PLC Resources | 3–35 |
| V–Memory | 3–36 |
| Binary-Coded Decimal Numbers | 3–36 |
| Hexadecimal Numbers | 3–36 |
| Memory Map | 3–37 |
| Octal Numbering System | 3–37 |
| Discrete and Word Locations | 3–37 |
| V–Memory Locations for Discrete Memory Areas | 3–37 |
| Input Points (X Data Type) | 3–38 |
| Output Points (Y Data Type) | 3–38 |
| Control Relays (C Data Type) | 3–38 |
| Timers and Timer Status Bits (T Data type) | 3–38 |
| Timer Current Values (V Data Type) | 3–39 |
| Counters and Counter Status Bits (CT Data type) | 3–39 |
| Counter Current Values (V Data Type) | 3–39 |
| Word Memory (V Data Type) | 3–39 |
| Stages (S Data type) | 3–40 |
| Special Relays (SP Data Type) | 3–40 |
| Remote I/O Points (GX Data Type) | 3–40 |
| DL230 System V-memory | 3–41 |
| DL240 System V-memory | 3–43 |
| DL250–1 System V-memory (DL250 also) | 3–46 |
| DL260 System V-memory | 3–49 |
| DL205 Aliases | 3–52 |
| DL230 Memory Map | 3–53 |
| DL240 Memory Map | 3–54 |
| DL250–1 Memory Map (DL250 also) | 3–55 |
| DL260 Memory Map | 3–56 |
| X Input/Y Output Bit Map | 3–57 |
| Control Relay Bit Map | 3–59 |
| Stage Control/Status Bit Map | 3–63 |
| Timer and Counter Status Bit Maps | 3–65 |
| Remote I/O Bit Map | 3–66 |

| | |
|--|-------------|
| Chapter 4: System Design and Configuration | 4-1 |
| DL205 System Design Strategies | 4-2 |
| I/O System Configurations | 4-2 |
| Networking Configurations | 4-2 |
| Module Placement | 4-3 |
| Slot Numbering | 4-3 |
| Module Placement Restrictions | 4-3 |
| Automatic I/O Configuration | 4-4 |
| Manual I/O Configuration | 4-4 |
| Removing a Manual Configuration | 4-5 |
| Power-On I/O Configuration Check | 4-5 |
| I/O Points Required for Each Module | 4-6 |
| Calculating the Power Budget | 4-7 |
| Managing your Power Resource | 4-7 |
| CPU Power Specifications | 4-7 |
| Module Power Requirements | 4-7 |
| Power Budget Calculation Example | 4-9 |
| Power Budget Calculation Worksheet | 4-10 |
| Local Expansion I/O | 4-11 |
| D2-CM Local Expansion Module | 4-11 |
| D2-EM Local Expansion Module | 4-12 |
| D2-EXCBL-1 Local Expansion Cable | 4-12 |
| DL260 Local Expansion System | 4-13 |
| DL250-1 Local Expansion System | 4-14 |
| Expansion Base Output Hold Option | 4-15 |
| Enabling I/O Configuration Check using <i>DirectSOFT</i> | 4-16 |
| Expanding DL205 I/O | 4-17 |
| I/O Expansion Overview | 4-17 |
| Ethernet Remote Master, H2-ERM(100)(-F) | 4-17 |
| Ethernet Remote Master Hardware Configuration | 4-18 |
| Installing the ERM Module | 4-19 |
| Ethernet Base Controller, H2-EBC(100)(-F) | 4-22 |
| Install the EBC Module | 4-23 |
| Set the Module ID | 4-23 |
| Insert the EBC Module | 4-23 |
| Network Cabling | 4-24 |

Table of Contents

| | |
|---|-------------|
| 10BaseFL Network Cabling | 4-25 |
| Maximum Cable Length | 4-25 |
| Add a Serial Remote I/O Master/Slave Module | 4-26 |
| Configuring the CPU's Remote I/O Channel | 4-27 |
| Configure Remote I/O Slaves | 4-29 |
| Configuring the Remote I/O Table | 4-29 |
| Remote I/O Setup Program | 4-30 |
| Remote I/O Test Program | 4-31 |
| Network Connections to Modbus and <i>DirectNet</i> | 4-32 |
| Configuring Port 2 For <i>DirectNet</i> | 4-32 |
| Configuring Port 2 For Modbus RTU | 4-32 |
| Modbus Port Configuration | 4-33 |
| <i>DirectNET</i> Port Configuration | 4-34 |
| Network Slave Operation | 4-35 |
| Modbus Function Codes Supported | 4-35 |
| Determining the Modbus Address | 4-35 |
| If Your Host Software Requires the Data Type and Address | 4-35 |
| If Your Modbus Host Software Requires an Address ONLY | 4-38 |
| Example 1: V2100 584/984 Mode | 4-40 |
| Example 2: Y20 584/984 Mode | 4-40 |
| Example 3: T10 Current Value 484 Mode | 4-40 |
| Example 4: C54 584/984 Mode | 4-40 |
| Determining the <i>DirectNET</i> Address | 4-40 |
| Network Master Operation | 4-41 |
| Communications from a Ladder Program | 4-44 |
| Multiple Read and Write Interlocks | 4-44 |
| Network Modbus RTU Master Operation (DL260 only) | 4-45 |
| Modbus Function Codes Supported | 4-45 |
| Modbus Port Configuration | 4-46 |
| RS-485 Network (Modbus only) | 4-47 |
| RS-232 Network | 4-47 |
| Modbus Read from Network (MRX) | 4-48 |
| MRX Slave Memory Address | 4-49 |
| MRX Master Memory Addresses | 4-49 |
| MRX Number of Elements | 4-49 |
| MRX Exception Response Buffer | 4-49 |
| Modbus Write to Network (MWX) | 4-50 |

| | |
|---|-------------|
| MWX Slave Memory Address | 4-51 |
| MWX Master Memory Addresses | 4-51 |
| MWX Number of Elements | 4-51 |
| MWX Exception Response Buffer | 4-51 |
| MRX/MWX Example in <i>DirectSOFT</i> | 4-52 |
| Multiple Read and Write Interlocks | 4-52 |
| Non-Sequence Protocol (ASCII In/Out and PRINT) | 4-54 |
| Configure the DL260 Port 2 for Non-Sequence | 4-54 |
| RS-485 Network | 4-55 |
| RS-232 Network | 4-55 |
| Configure the DL250-1 Port 2 for Non-Sequence | 4-56 |
| RS-422 Network | 4-57 |
| RS-232 Network | 4-57 |
| Chapter 5: RLL and Intelligent Box (IBOX) Instructions | 5-1 |
| Introduction | 5-2 |
| Using Boolean Instructions | 5-5 |
| END Statement | 5-5 |
| Simple Rungs | 5-5 |
| Normally Closed Contact | 5-6 |
| Contacts in Series | 5-6 |
| Midline Outputs | 5-6 |
| Parallel Elements | 5-7 |
| Joining Series Branches in Parallel | 5-7 |
| Joining Parallel Branches in Series | 5-7 |
| Combination Networks | 5-7 |
| Comparative Boolean | 5-8 |
| Boolean Stack | 5-8 |
| Immediate Boolean | 5-9 |
| Boolean Instructions | 5-10 |
| Comparative Boolean | 5-27 |
| Immediate Instructions | 5-33 |
| Timer, Counter and Shift Register Instructions | 5-41 |
| Using Timers | 5-41 |
| Timer Example Using Discrete Status Bits | 5-43 |

Table of Contents

| | |
|--|--------------|
| Timer Example Using Comparative Contacts | 5-43 |
| Accumulating Timer (TMRA) | 5-44 |
| Accumulating Timer Example using Discrete Status Bits | 5-45 |
| Accumulator Timer Example Using Comparative Contacts | 5-45 |
| Counter Example Using Discrete Status Bits | 5-47 |
| Counter Example Using Comparative Contacts | 5-47 |
| Stage Counter Example Using Discrete Status Bits | 5-49 |
| Stage Counter Example Using Comparative Contacts | 5-49 |
| Up/Down Counter Example Using Discrete Status Bits | 5-51 |
| Up/Down Counter Example Using Comparative Contacts | 5-51 |
| Accumulator/Stack Load and Output Data Instructions | 5-53 |
| Logical Instructions (Accumulator) | 5-71 |
| Math Instructions | 5-88 |
| Transcendental Functions (DL260 only) | 5-121 |
| Bit Operation Instructions | 5-123 |
| Number Conversion Instructions (Accumulator) | 5-130 |
| Table Instructions | 5-144 |
| Clock/Calendar Instructions | 5-175 |
| CPU Control Instructions | 5-177 |
| Program Control Instructions | 5-179 |
| Interrupt Instructions | 5-187 |
| Intelligent I/O Instructions | 5-191 |
| Network Instructions | 5-193 |
| Message Instructions | 5-197 |
| Modbus RTU Instructions (DL260) | 5-205 |
| Modbus Read from Network (MRX) | 5-205 |
| Modbus Write to Network (MWX) | 5-208 |
| ASCII Instructions (DL260) | 5-211 |
| Intelligent Box (IBox) Instructions (DL250-1/DL260) | 5-230 |

VOLUME TWO:

TABLE OF CONTENTS



| | |
|---|-------------|
| Chapter 6: Drum Instruction Programming (DL250-1/DL260 only) | 6-1 |
| Introduction | 6-2 |
| Purpose | 6-2 |
| Drum Terminology | 6-2 |
| Drum Chart Representation | 6-3 |
| Output Sequences | 6-3 |
| Step Transitions | 6-4 |
| Drum Instruction Types | 6-4 |
| Timer-Only Transitions | 6-4 |
| Timer and Event Transitions | 6-5 |
| Event-Only Transitions | 6-6 |
| Counter Assignments | 6-6 |
| Last Step Completion | 6-7 |
| Overview of Drum Operation | 6-8 |
| Drum Instruction Block Diagram | 6-8 |
| Powerup State of Drum Registers | 6-9 |
| Drum Control Techniques | 6-10 |
| Drum Control Inputs | 6-10 |
| Self-Resetting Drum | 6-11 |
| Initializing Drum Outputs | 6-11 |
| Using Complex Event Step Transitions | 6-11 |
| Drum Instruction | 6-12 |
| Timed Drum with Discrete Outputs (DRUM) | 6-12 |
| Event Drum (EDRUM) | 6-14 |
| Handheld Programmer Drum Mnemonics | 6-16 |
| Masked Event Drum with Discrete Outputs (MDRMD) | 6-19 |
| Masked Event Drum with Word Output (MDRMW) | 6-21 |

| | |
|---|-------------|
| Chapter 7: RLL^{PLUS} Stage Programming | 7-1 |
| Introduction to Stage Programming | 7-2 |
| Overcoming “Stage Fright” | 7-2 |
| Learning to Draw State Transition Diagrams | 7-3 |
| Introduction to Process States | 7-3 |
| The Need for State Diagrams | 7-3 |
| A 2-State Process | 7-3 |
| RLL Equivalent | 7-4 |
| Stage Equivalent | 7-4 |
| Let’s Compare | 7-5 |
| Initial Stages | 7-5 |
| What Stage Bits Do | 7-6 |
| Stage Instruction Characteristics | 7-6 |
| Using the Stage Jump Instruction for State Transitions | 7-7 |
| Stage Jump, Set, and Reset Instructions | 7-7 |
| Stage Program Example: Toggle On/Off Lamp Controller | 7-8 |
| A 4-State Process | 7-8 |
| Four Steps to Writing a Stage Program | 7-9 |
| Stage Program Example: A Garage Door Opener | 7-10 |
| Garage Door Opener Example | 7-10 |
| Draw the Block Diagram | 7-10 |
| Draw the State Diagram | 7-11 |
| Add Safety Light Feature | 7-12 |
| Modify the Block Diagram and State Diagram | 7-12 |
| Using a Timer Inside a Stage | 7-13 |
| Add Emergency Stop Feature | 7-14 |
| Exclusive Transitions | 7-14 |
| Stage Program Design Considerations | 7-15 |
| Stage Program Organization | 7-15 |
| How Instructions Work Inside Stages | 7-16 |
| Using a Stage as a Supervisory Process | 7-17 |
| Stage Counter | 7-17 |
| Unconditional Outputs | 7-18 |
| Power Flow Transition Technique | 7-18 |
| Parallel Processing Concepts | 7-19 |

| | |
|---|----------------|
| Parallel Processes | 7–19 |
| Converging Processes | 7–19 |
| Convergence Stages (CV) | 7–19 |
| Convergence Jump (CVJMP) | 7–20 |
| Convergence Stage Guidelines | 7–20 |
| Managing Large Programs | 7–21 |
| Stage Blocks (BLK, BEND) | 7–21 |
| Block Call (BCALL) | 7–22 |
| RLL^{PLUS} (Stage) Instructions | 7–23 |
| Stage (SG) | 7–23 |
| Initial Stage (ISG) | 7–24 |
| Jump (JMP) | 7–24 |
| Not Jump (NJMP) | 7–24 |
| Converge Stage (CV) and Converge Jump (CVJMP) | 7–25 |
| Block Call (BCALL) | 7–27 |
| Block (BLK) | 7–27 |
| Block End (BEND) | 7–27 |
| Stage View in <i>DirectSOFT</i> | 7–28 |
| Questions and Answers about Stage Programming | 7–29 |
| Chapter 8: PID Loop Operation | 8–1 |
| DL250-1 and DL260 PID Loop Features | 8–2 |
| Main Features | 8–2 |
| Introduction to PID Control | 8–4 |
| Why use PID Control? | 8–4 |
| Introducing DL205 PID Control | 8–6 |
| Process Control Definitions | 8–8 |
| PID Loop Operation | 8–9 |
| Position Form of the PID Equation | 8–9 |
| Reset Windup Protection | 8–10 |
| Freeze Bias | 8–11 |
| Adjusting the Bias | 8–11 |
| Step Bias Proportional to Step Change in SP | 8–12 |
| Eliminating Proportional, Integral or Derivative Action | 8–12 |
| Velocity Form of the PID Equation | 8–12 |

| | |
|---|-------------|
| Bumpless Transfer | 8-13 |
| Loop Alarms | 8-13 |
| Loop Operating Modes | 8-14 |
| Special Loop Calculations | 8-14 |
| Ten Steps to Successful Process Control | 8-16 |
| PID Loop Setup | 8-18 |
| Some Things to Do and Know Before Starting | 8-18 |
| PID Error Flags | 8-18 |
| Establishing the Loop Table Size and Location | 8-18 |
| Loop Table Word Definitions | 8-20 |
| PID Mode Setting 1 Bit Descriptions (Addr + 00) | 8-21 |
| PID Mode Setting 2 Bit Descriptions (Addr + 01) | 8-22 |
| Mode/Alarm Monitoring Word (Addr + 06) | 8-23 |
| Ramp/Soak Table Flags (Addr + 33) | 8-23 |
| Ramp/Soak Table Location (Addr + 34) | 8-24 |
| Ramp/Soak Table Programming Error Flags (Addr + 35) | 8-24 |
| PV Auto Transfer (Addr + 36) from I/O Module Base/Slot/Channel Option | 8-25 |
| PV Auto Transfer (Addr + 36) from V-memory Option | 8-25 |
| Control Output Auto Transfer (Addr + 37) | 8-25 |
| Configure the PID Loop | 8-26 |
| PID Loop Tuning | 8-41 |
| Open-Loop Test | 8-41 |
| Manual Tuning Procedure | 8-42 |
| Alternative Manual Tuning Procedures by Others | 8-45 |
| Tuning PID Controllers | 8-45 |
| Auto Tuning Procedure | 8-46 |
| Use <i>DirectSOFT</i> Data View with PID View | 8-50 |
| Open a New Data View Window | 8-50 |
| Open PID View | 8-51 |
| Using the Special PID Features | 8-54 |
| How to Change Loop Modes | 8-54 |
| Operator Panel Control of PID Modes | 8-55 |
| PLC Modes Effect on Loop Modes | 8-55 |
| Loop Mode Override | 8-55 |
| PV Analog Filter | 8-56 |
| Creating an Analog Filter in Ladder Logic | 8-57 |

| | |
|---|----------------|
| Use the <i>DirectSOFT</i> 5 Filter Intelligent Box (IBOX) Instruction | 8–58 |
| FilterB Example | 8–58 |
| Ramp/Soak Generator | 8–59 |
| Introduction | 8–59 |
| Ramp/Soak Table | 8–60 |
| Ramp/Soak Table Flags | 8–62 |
| Ramp/Soak Generator Enable | 8–62 |
| Ramp/Soak Controls | 8–62 |
| Ramp/Soak Profile Monitoring | 8–63 |
| Ramp/Soak Programming Errors | 8–63 |
| Testing Your Ramp/Soak Profile | 8–63 |
| <i>DirectSOFT</i> Ramp/Soak Example | 8–64 |
| Setup the Profile in PID Setup | 8–64 |
| Program the Ramp/Soak Control in Relay Ladder | 8–64 |
| Test the Profile | 8–65 |
| Cascade Control | 8–66 |
| Introduction | 8–66 |
| Cascaded Loops in the DL205 CPU | 8–67 |
| Tuning Cascaded Loops | 8–68 |
| Time-Proportioning Control | 8–69 |
| On/Off Control Program Example | 8–70 |
| Feedforward Control | 8–71 |
| Feedforward Example | 8–72 |
| PID Example Program | 8–73 |
| Program Setup for the PID Loop | 8–73 |
| Troubleshooting Tips | 8–76 |
| Glossary of PID Loop Terminology | 8–78 |
| Bibliography | 8–80 |
| Chapter 9: Maintenance and Troubleshooting | 9–1 |
| Hardware Maintenance | 9–2 |
| Standard Maintenance | 9–2 |
| Air Quality Maintenance | 9–2 |
| Low Battery Indicator | 9–2 |
| CPU Battery Replacement | 9–2 |

| | |
|---|-------------|
| Diagnostics | 9-3 |
| Diagnostics | 9-3 |
| Fatal Errors | 9-3 |
| Non-fatal Errors | 9-3 |
| Finding Diagnostic Information | 9-4 |
| V-memory Locations Corresponding to Error Codes | 9-4 |
| Special Relays (SP) Corresponding to Error Codes | 9-5 |
| I/O Module Codes | 9-6 |
| Error Message Tables | 9-7 |
| System Error Codes | 9-8 |
| Program Error Codes | 9-9 |
| CPU Error Indicators | 9-10 |
| PWR Indicator | 9-11 |
| Incorrect Base Power | 9-11 |
| Faulty CPU | 9-11 |
| Device or Module causing the Power Supply to Shutdown | 9-12 |
| Power Budget Exceeded | 9-12 |
| Run Indicator | 9-13 |
| CPU Indicator | 9-13 |
| BATT Indicator | 9-13 |
| Communications Problems | 9-13 |
| I/O Module Troubleshooting | 9-14 |
| Things to Check | 9-14 |
| I/O Diagnostics | 9-14 |
| Some Quick Steps | 9-15 |
| Testing Output Points | 9-16 |
| Handheld Programmer Keystrokes Used to Test an Output Point | 9-16 |
| Noise Troubleshooting | 9-17 |
| Electrical Noise Problems | 9-17 |
| Reducing Electrical Noise | 9-17 |
| Machine Startup and Program Troubleshooting | 9-18 |
| Syntax Check | 9-18 |
| Duplicate Reference Check | 9-19 |
| TEST-PGM and TEST-RUN Modes | 9-20 |
| Special Instructions | 9-22 |
| Run Time Edits | 9-24 |

| | |
|------------------------------------|------|
| Forcing I/O Points | 9-26 |
| Regular Forcing with Direct Access | 9-28 |
| Bit Override Forcing | 9-29 |
| Bit Override Indicators | 9-29 |

Appendix A: Auxiliary Functions **A-1**

Introduction **A-2**

| | |
|---|-----|
| What are Auxiliary Functions? | A-2 |
| Accessing AUX Functions via <i>DirectSOFT</i> | A-3 |
| Accessing AUX Functions via the Handheld Programmer | A-3 |

AUX 2* — RLL Operations **A-4**

| | |
|---------------------------|-----|
| AUX 21-24 | 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-5**

| | |
|-------------------------------------|-----|
| AUX 31 | A-5 |
| AUX 31 Clear V-Memory | A-5 |
| AUX 4* — I/O Configuration | A-5 |
| AUX 41-46 | A-5 |
| AUX 41 Show I/O Configuration | A-5 |
| AUX 42 I/O Diagnostics | A-5 |
| AUX 44 Power-up Configuration Check | A-5 |
| AUX 45 Select Configuration | A-6 |
| AUX 46 to I/O Configuration | A-6 |

AUX 5* — CPU Configuration **A-7**

| | |
|--------------------------------|-----|
| AUX 51-5C | A-7 |
| AUX 51 Modify Program Name | A-7 |
| AUX 52 Display/Change Calendar | A-7 |
| AUX 53 Display Scan Time | A-8 |
| AUX 54 Initialize Scratchpad | A-8 |
| AUX 55 Set Watchdog Timer | A-8 |
| AUX 56 CPU Network Address | A-8 |
| AUX 57 Set Retentive Ranges | A-9 |
| AUX 58 Test Operations | A-9 |

Table of Contents

| | |
|--|----------------|
| AUX 59 Bit Override | A-10 |
| AUX 5B Counter Interface Configuration | A-10 |
| AUX 5C Display Error History | A-11 |
| AUX 6* — Handheld Programmer Configuration | A-12 |
| AUX 61, 62 and 65 | A-12 |
| AUX 61 Show Revision Numbers | A-12 |
| AUX 62 Beeper On/Off | A-12 |
| AUX 65 Run Self Diagnostics | A-12 |
| AUX 7* - EEPROM Operations | A-12 |
| AUX 71 - 76 | A-12 |
| Transferable Memory Areas | A-13 |
| AUX 71 CPU to HPP EEPROM | A-13 |
| AUX 72 HPP EEPROM to CPU | A-13 |
| AUX 73 Compare HPP EEPROM to CPU | A-13 |
| AUX 74 HPP EEPROM Blank Check | A-13 |
| AUX 75 Erase HPP EEPROM | A-13 |
| AUX 76 Show EEPROM Type | A-13 |
| AUX 8* — Password Operations | A-14 |
| AUX 81 - 83 | A-14 |
| AUX 81 Modify Password | A-14 |
| AUX 82 Unlock CPU | A-14 |
| AUX 83 Lock CPU | A-14 |
| Appendix B: DL205 Error Codes | B-1 |
| Appendix C: Instruction Execution Times | C-1 |
| Introduction | C-2 |
| V-Memory Data Registers | C-2 |
| V-Memory Bit Registers | C-2 |
| How to Read the Tables | C-2 |
| Boolean Instructions | C-3 |
| Comparative Boolean Instructions | C-4 |
| Bit of Word Boolean Instructions | C-13 |
| Immediate Instructions | C-14 |
| Timer, Counter and Shift Register Instructions | C-15 |

| | |
|--|------------|
| Accumulator Data Instructions | C-16 |
| Logical Instructions | C-18 |
| Math Instructions | C-20 |
| Differential Instructions | C-23 |
| Bit Instructions | C-24 |
| Number Conversion Instructions | C-25 |
| Table Instructions | C-25 |
| CPU Control Instructions | C-27 |
| Program Control Instructions | C-27 |
| Interrupt Instructions | C-28 |
| Network Instructions | C-28 |
| Intelligent I/O Instructions | C-28 |
| Message Instructions | C-29 |
| RLL ^{PLUS} Instructions | C-29 |
| DRUM Instructions | C-29 |
| Clock / Calender Instructions | C-30 |
| Modbus Instructions | C-30 |
| ASCII Instructions | C-30 |
| Appendix D: Special Relays | D-1 |
| DL230 CPU Special Relays | D-2 |
| Startup and Real-Time Relays | D-2 |
| CPU Status Relays | D-2 |
| System Monitoring | D-2 |
| Accumulator Status | D-3 |
| Counter Interface Module Relays | D-3 |
| Equal Relays for Multi-step Presets with Up/Down Counter #1 / DL230 (for use with a Counter Interface Module) | D-4 |
| DL240/DL250-1/DL260 CPU Special Relays | D-5 |
| Startup and Real-Time Relays | D-5 |
| CPU Status Relays | D-5 |
| System Monitoring Relays | D-6 |
| Accumulator Status Relays | D-6 |

Table of Contents

| | |
|--|------------|
| Counter Interface Module Relays | D-7 |
| Communications Monitoring Relays | D-8 |
| Equal Relays for Multi-step Presets with Up/Down Counter #1 (for use with a Counter Interface Module) | D-9 |
| Equal Relays for Multi-step Presets with Up/Down Counter #2 (for use with a Counter Interface Module) | D-10 |
| Appendix E: PLC Memory | E-1 |
| DL205 PLC Memory | E-2 |
| Non-volatile V-memory in the DL205 | E-3 |
| Appendix F: DL205 Product Weight Table | F-1 |
| DL205 Product Weight Table | F-2 |
| Appendix G: ASCII Table | G-1 |
| ASCII Conversion Table | G-2 |
| Appendix H: Numbering Systems | H-1 |
| Introduction | H-2 |
| Binary Numbering System | H-2 |
| Hexadecimal Numbering System | H-3 |
| Octal Numbering System | H-4 |
| Binary Coded Decimal (BCD) Numbering System | H-5 |
| Real (Floating Point) Numbering System | H-5 |
| BCD/Binary/Decimal/Hex/Octal -What is the Difference? | H-6 |
| Data Type Mismatch | H-7 |
| Signed vs. Unsigned Integers | H-8 |
| AutomationDirect.com Products and Data Types | H-9 |
| DirectLOGIC PLCs | H-9 |
| C-more/C-more Micro-Graphic Panels | H-9 |

| | |
|---|------------|
| Appendix I: European Union Directives (CE) | I-1 |
| European Union (EU) Directives | I-2 |
| Member Countries | I-2 |
| Applicable Directives | I-2 |
| Compliance | I-2 |
| General Safety | I-3 |
| Special Installation Manual | I-4 |
| Other Sources of Information | I-4 |
| Basic EMC Installation Guidelines | I-5 |
| Enclosures | I-5 |
| Electrostatic Discharge (ESD) | I-5 |
| AC Mains Filters | I-6 |
| Suppression and Fusing | I-6 |
| Internal Enclosure Grounding | I-6 |
| Equi-potential Grounding | I-7 |
| Communications and Shielded Cables | I-7 |
| Analog and RS232 Cables | I-8 |
| Shielded Cables within Enclosures | I-8 |
| Analog Modules and RF Interference | I-9 |
| Network Isolation | I-9 |
| DC Powered Versions | I-9 |
| Items Specific to the DL205 | I-10 |

Index

Notes

DL205 PLC USER MANUAL



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

Manual Number: D2-USER-M
Issue: 4th Edition, Rev. C
Issue Date: 4/17

| Publication History | | |
|---------------------|-------|---|
| Issue | Date | Description of Changes |
| 1st Edition | 1/94 | Original edition |
| Rev. A | 9/95 | Minor corrections |
| 2nd Edition | 6/97 | Added DL250, downsized manual |
| Rev. A | 5/98 | Minor corrections |
| Rev. B | 7/99 | Added torque specs for base and I/O |
| Rev. C | 11/99 | Minor corrections |
| Rev. D | 3/00 | Added new PID features, minor corrections |
| Rev. E | 11/00 | Added CE information, minor corrections |
| Rev. F | 11/01 | Added surge protection info, corrected RLL and DRUM instructions, minor corrections |
| 3rd Edition | 6/02 | Added DL250–1 and DL260 CPUs, local expansion I/O, ASCII and MODBUS instructions, split manual into two volumes |
| Rev A | 8/03 | Extensive corrections and additions |
| 4th Edition | 11/08 | Changed publishing software resulting in change of appearance, addition of IBox instructions, changes to PID chapter, added info for ERM and EBC modules, other changes as necessary |
| Rev A | 4/10 | Extensive corrections and additions |
| Rev B | 2/13 | Corrected number of memory registers needed in the print message instruction. Added new transient suppression for inductive loads to Chapter 2. Added H2-CTRIO2 and H2-ERM100 references. |
| Rev C | 4/17 | Minor corrections with general updates. ECEMAIL Decimal Status Codes added, Chapter 5 |

Notes

GETTING STARTED



CHAPTER 1

In This Chapter...

| | |
|---|------|
| Introduction | 1-2 |
| Conventions Used | 1-3 |
| DL205 System Components..... | 1-4 |
| Programming Methods | 1-7 |
| <i>Direct</i> LOGIC™ Part Numbering System | 1-8 |
| Quick Start for PLC Validation and Programming..... | 1-10 |
| Steps to Designing a Successful System | 1-13 |

Introduction

The Purpose of this Manual

Thank you for purchasing our DL205 family of products. This manual shows you how to install, program, and maintain the equipment. It also helps you understand how to interface them to other devices in a control system.

This manual contains important information for personnel who will install DL205 PLCs and components and for the PLC programmer. If you understand PLC systems, our manuals will provide all the information you need to start and keep your system up and running.

Where to Begin

If you already understand PLCs please read Chapter 2, “Installation, Wiring, and Specifications,” and proceed on to other chapters as needed. Keep this manual handy for reference when you have questions. If you are a new DL205 customer, we suggest you read this manual completely to understand the wide variety of features in the DL205 family of products. We believe you will be pleasantly surprised with how much you can accomplish with our products.

Supplemental Manuals

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

Technical Support

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

770-844-4200

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

<http://www.automationdirect.com>

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

Conventions Used



When you see the “notepad” icon in the left–hand margin, the paragraph to its immediate right will be a special note.

The word **NOTE** in boldface will mark the beginning of the text.

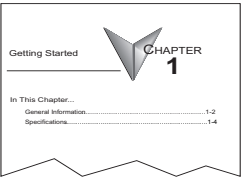


When you see the “exclamation mark” icon in the left–hand margin, the paragraph to its immediate right will be a warning. This information could prevent injury, loss of property, or even death (in extreme cases).

The word **WARNING** in boldface will mark the beginning of the text.

Key Topics for Each Chapter

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



DL205 System Components

The DL205 family is a versatile product line that provides a wide variety of features in an extremely compact package. The CPUs are small, but offer many instructions normally only found in larger, more expensive systems. The modular design also offers more flexibility in the fast moving industry of control systems. The following is a summary of the major DL205 system components.

CPU's

This product line includes four feature-enhanced CPUs: the DL230, DL240, DL250-1 and DL260. All CPUs include built-in communication ports. Each CPU offers a large amount of program memory, a substantial instruction set and advanced diagnostics. The DL250-1 features drum timers, floating-point math, 4 built-in PID loops with automatic tuning and 2 bases of local expansion capability.

The DL260 features ASCII IN/OUT and extended MODBUS communications, table and trigonometric instructions, 16 PID loops with autotuning and up to 4 bases of local expansion. Details of these CPU features and more are covered in Chapter 3, CPU Specifications and Operation.

Bases

Four base sizes are available: 3, 4, 6 and 9 slot. The DL205 PLCs use bases that can be expanded. The part numbers for these bases end with -1. These bases have a connector for local expansion located on the right end of the base. They can serve in local, local expansion and remote I/O configurations. All bases include a built-in power supply. The bases with the -1 suffix can replace existing bases without a suffix if expansion is required.

I/O Configuration

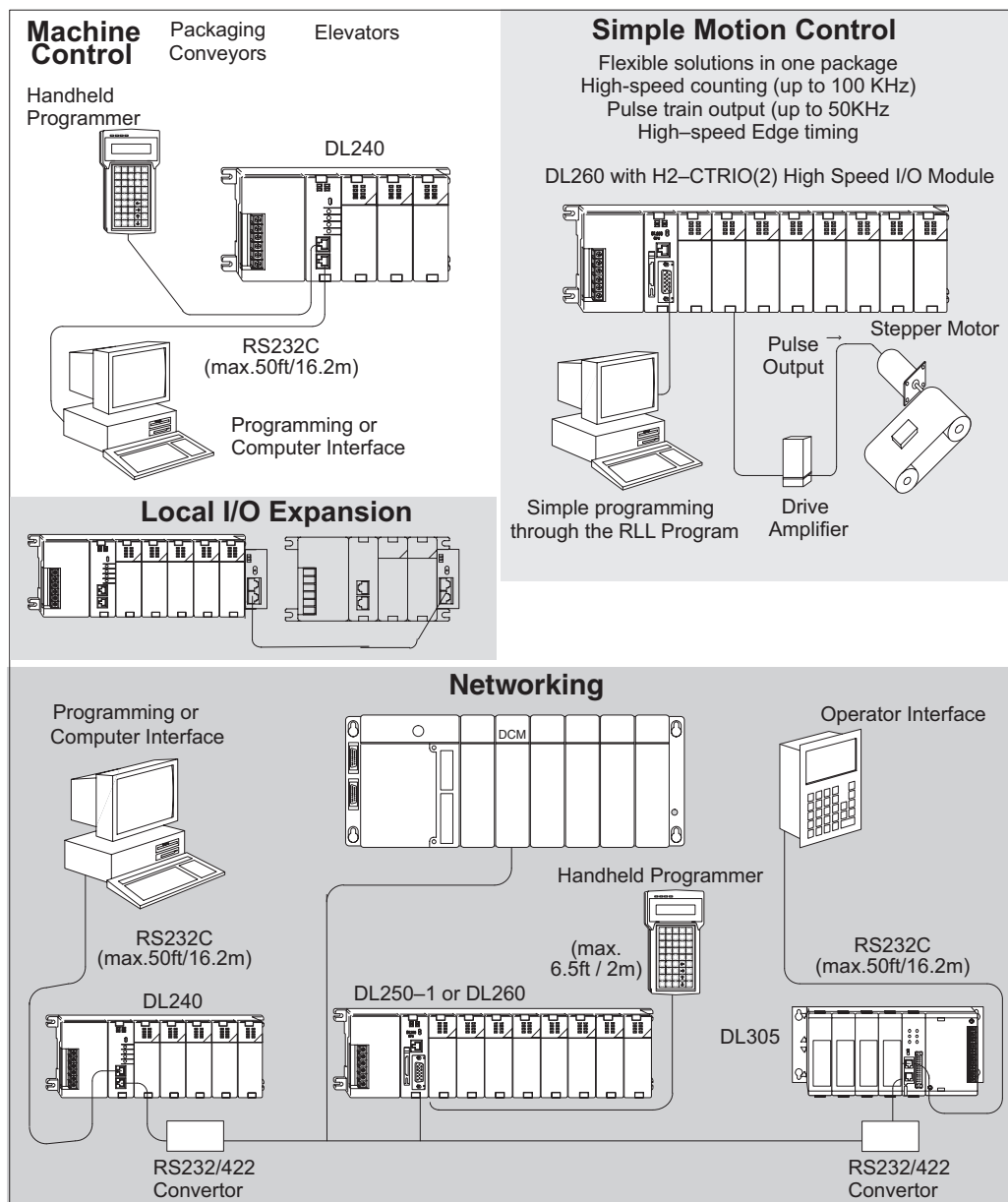
The DL230 and DL240 CPUs can support up to 256 local I/O points. The DL250-1 can support up to 768 local I/O points with up to two expansion bases. The DL260 can support up to 1280 local I/O points with up to four expansion bases. These points can be assigned as input or output points. The DL240, DL250-1 and DL260 systems can also be expanded by adding remote I/O points. The DL250-1 and DL260 provide a built-in master for remote I/O networks. The I/O configurations are explained in Chapter 4, System Design and Configuration. I/O Modules

I/O Modules

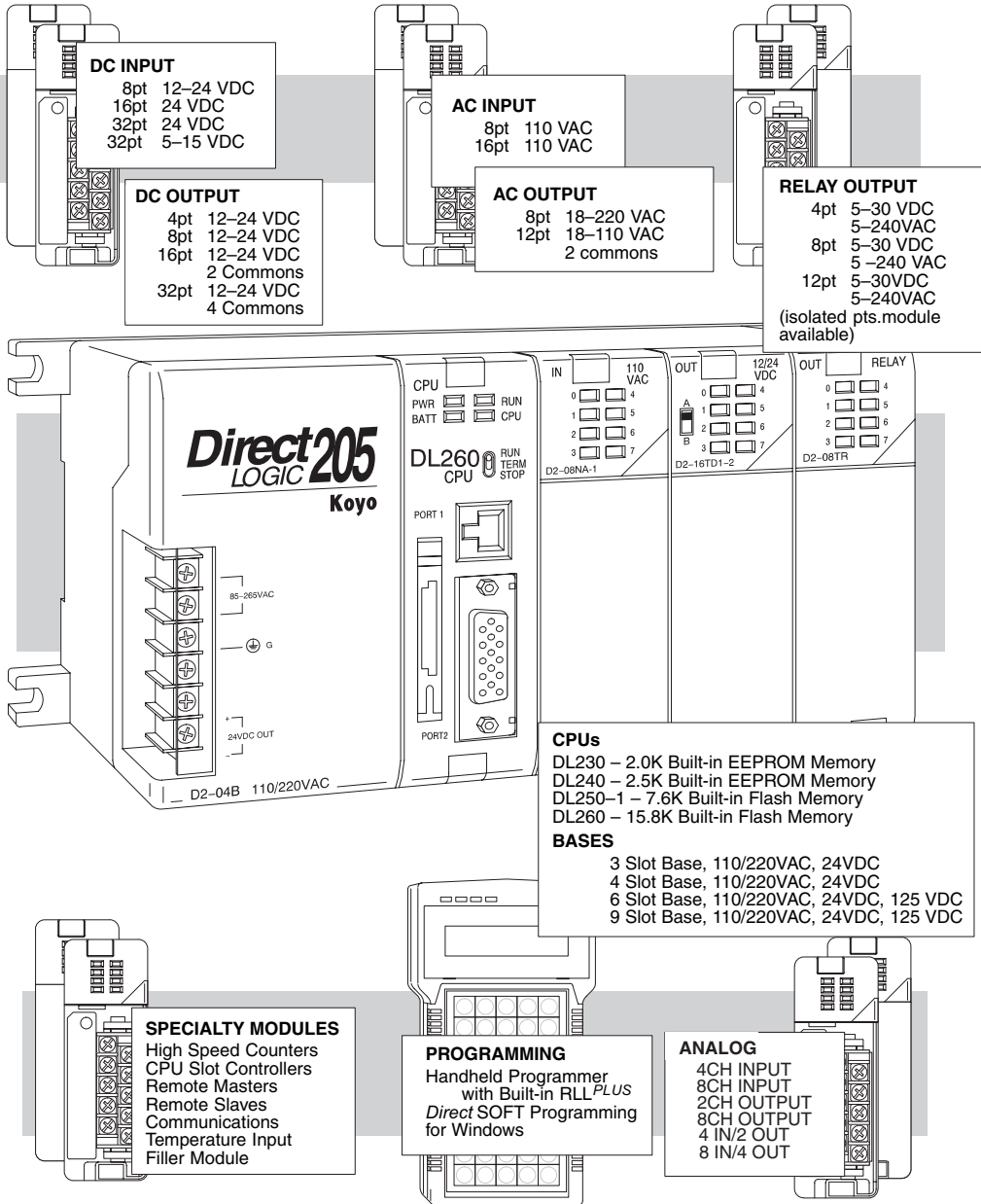
The DL205 has some of the most powerful modules in the industry. A complete range of discrete modules which support 24 VDC, 110/220 VAC and up to 10A relay outputs (subject to derating) are offered. The analog modules provide 12- and 16-bit resolution and several selections of input and output signal ranges (including bipolar). Several specialty and communications modules are also available.

DL205 System Diagrams

The diagram below shows the major components and configurations of the DL205 system. The next two pages show specific components for building your system.



DirectLOGIC DL205 Family



Programming Methods

Two programming methods are available for the DL205 CPUs: Relay Ladder Logic (RLL) and RLL^{PLUS} (Stage Programming). Both the *DirectSOFT5* programming package and the handheld programmer support RLL and Stage.

***DirectSOFT* Programming for Windows**

The DL205 can be programmed with one of the most advanced programming packages in the industry —*DirectSOFT5*. *DirectSOFT5* is a Windows-based software package that supports many Windows features you already know, such as cut and paste between applications, point and click editing, viewing and editing multiple application programs at the same time, etc. *DirectSOFT5* universally supports the *DirectLOGIC* CPU families. This means you can use the same *DirectSOFT5* package to program DL05, DL06, DL105, DL205, DL305, DL405 or any new CPUs we may add to our product line. A separate manual discusses the *DirectSOFT5* programming software which is included with your software package.

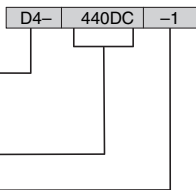
Handheld Programmer

All DL205 CPUs have a built-in programming port for use with the handheld programmer (D2–HPP). The handheld programmer can be used to create, modify and debug your application program. A separate manual that discusses the DL205 Handheld Programmer is available.

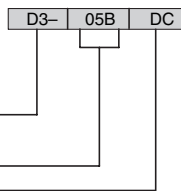
DirectLOGIC™ Part Numbering System

As you examine this manual, you will notice there are many different products available. Sometimes it is difficult to remember the specifications for any given product. However, if you take a few minutes to understand the numbering system, it may save you some time and confusion. The charts below show how the part numbering systems work for each product category. Part numbers for accessory items such as cables, batteries, memory cartridges, etc, are typically an abbreviation of the description for the item.

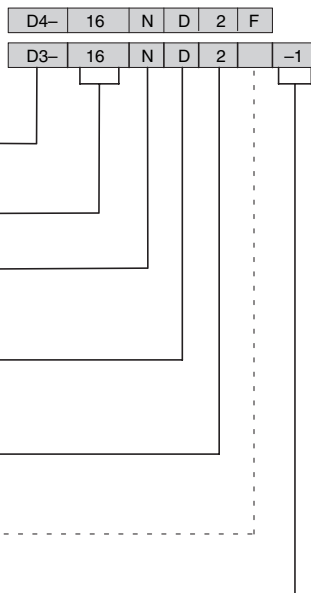
| CPUs | |
|---|----------------------|
| Specialty CPUs | |
| DL05/06 Product family | D0/F0 |
| DL105 Product family | D1/F1 |
| DL205 Product family | D2/F2 |
| DL305 Product family | D3/F3 |
| DL405 Product family | D4/F4 |
| Class of CPU / Abbreviation | 230...,330...,430... |
| Denotes a differentiation between similar modules | -1, -2, -3, -4 |



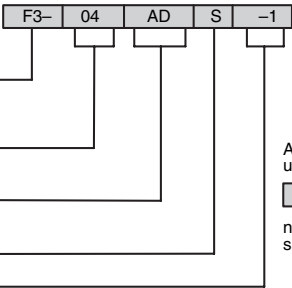
| Bases | |
|----------------------|-------------|
| DL205 Product family | D2/F2 |
| DL305 Product family | D3/F3 |
| DL405 Product family | D4/F4 |
| Number of slots | ##B |
| Type of Base | DC or empty |



| Discrete I/O | |
|---|-------------------|
| DL05/06 Product family | D0/F0 |
| DL205 Product family | D2/F2 |
| DL305 Product family | D3/F3 |
| DL405 Product family | D4/F4 |
| Number of points | 04/08/12/16/32/64 |
| Input | N |
| Output | T |
| Combination | C |
| AC | A |
| DC | D |
| Either | E |
| Relay | R |
| Current Sinking | 1 |
| Current Sourcing | 2 |
| Current Sinking/Sourcing | 3 |
| High Current | H |
| Isolation | S |
| Fast I/O | F |
| Denotes a differentiation between similar modules | -1, -2, -3, -4 |



| Analog I/O | |
|---|----------------|
| DL05/06 Product family | D0/F0 |
| DL205 Product family | D2/F2 |
| DL305 Product family | D3/F3 |
| DL405 Product family | D4/F4 |
| Number of channels | 02/04/08/16 |
| Input (Analog to Digital) | AD |
| Output (Digital to Analog) | DA |
| Combination | AND |
| Isolated | S |
| Denotes a differentiation between Similar modules | -1, -2, -3, -4 |

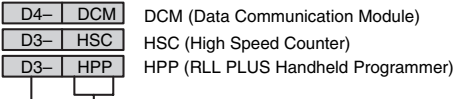


Alternate example of Analog I/O using abbreviations

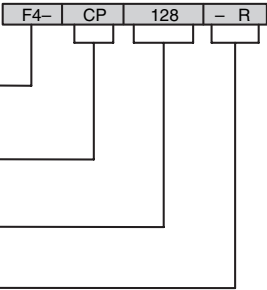


note: -n indicates thermocouple type such as: J, K, T, R, S or E

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



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



Quick Start for PLC Validation and Programming

If you have experience using PLCs, or want to set up a quick example, this section is what you want to use. This example is not intended to explain everything needed to start up your system. It is only intended to provide a general picture of what is needed to get your system powered up.

Step 1: Unpack the DL205 Equipment

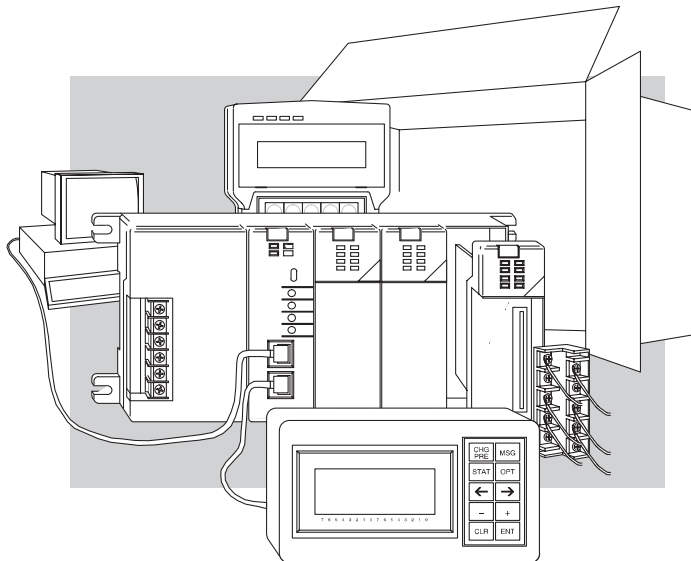
Unpack the DL205 equipment and verify you have the parts necessary to build this demonstration system. The minimum parts needed are as follows:

- Base
- CPU
- A discrete input module such as a D2-16ND3-2 DC or a F2-08SIM input simulator module
- A discrete output module such as a D2-16TD1-2 DC
- *Power cord
- *Hook up wire
- *One or more toggle switches (if not using the input simulator module)
- *A screwdriver, blade or Phillips type

*These items are not supplied with your PLC.

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

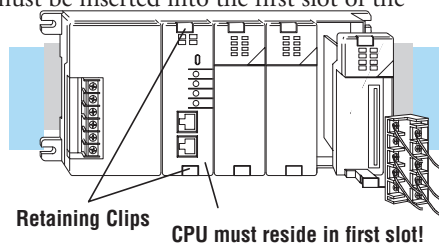
- *DirectSOFT5* Programming Software, *DirectSOFT5* Manual, and a programming cable (connects the CPU to a personal computer), or
- D2-HPP Handheld Programmer and the Handheld Programmer Manual.



Step 2: Install the CPU and I/O Modules

Insert the CPU and I/O into the base. The CPU must be inserted into the first slot of the base (next to the power supply).

- Each unit has a plastic retaining clip at the top and bottom. Slide the retainer clips to the out position before installing the module.
- With the unit square to the base, slide it in using the upper and lower guides.
- Gently push the unit back until it is firmly seated in the backplane.
- Secure the unit to the base by pushing in the retainer clips.

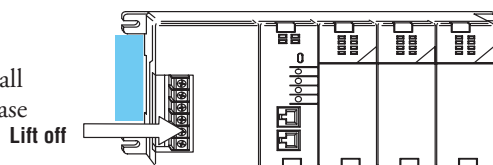


Placement of discrete, analog and relay modules is not critical and may go in any slot in any base; however, for this example, install the output module in the slot next to the CPU and the input module in the next slot. Limiting factors for other types of modules are discussed in Chapter 4, System Design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

Step 3: Remove Terminal Strip Access Cover

Cover

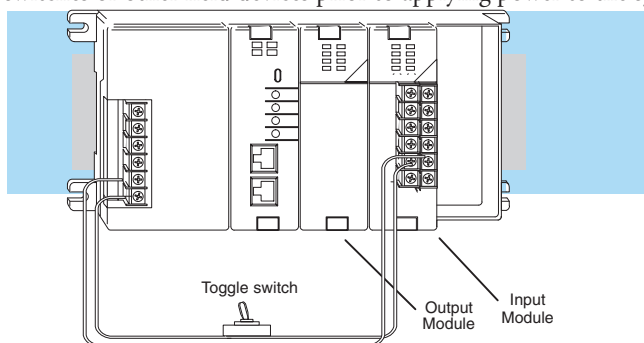
Remove the terminal strip cover. It is a small strip of clear plastic that is located on the base power supply.



Step 4: Add I/O Simulation

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

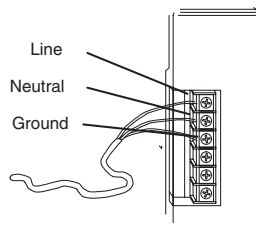
Wire the switches or other field devices prior to applying power to the system to ensure a



point is not accidentally turned on during the wiring operation. This example uses DC input and output modules. Wire the input module, X0, to the toggle switch and 24VDC auxiliary power supply on the CPU terminal strip as shown. Chapter 2, Installation, Wiring, and Specifications, provides a list of I/O wiring guidelines.

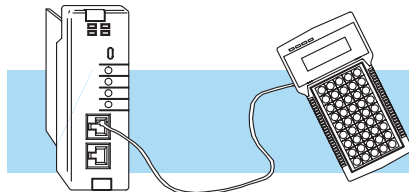
Step 5: Connect the Power Wiring

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



Step 6: Connect the Programmer

Either connect the programming cable connected to a computer loaded with *DirectSOFT* Programming Software or a D2-HPP Handheld Programmer (comes with programming cable) to the top port of the CPU.



Step 7: Switch On the System Power

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

Step 8: Enter the Program

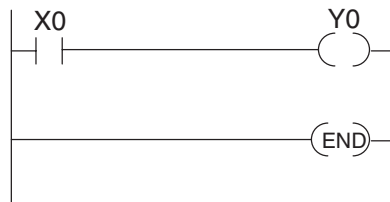
Slide the switch on the CPU to the STOP position (250–1/260 only) and then back to the TERM position. This puts the CPU in the program mode and allows access to the CPU program. Edit a *DirectSOFT* program using the relay ladder diagram below and load it into the PLC. If using an HPP, the PGM indicator should be illuminated on the HPP. Enter the following keystrokes on the HPP:



NOTE: It is not necessary for you to configure the I/O for this system since the DL205 CPUs automatically examine any installed modules and establish the correct configuration.

Handheld Program Keystrokes

| | | | |
|-----|---|---|-----|
| \$ | → | B | ENT |
| STR | | 1 | |
| GX | → | C | ENT |
| OUT | | 2 | |



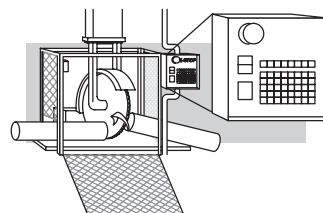
After entering the example program put the CPU in the RUN mode with *DirectSOFT* or after entering the program using the HPP, slide the switch from the TERM position to the RUN position and back to TERM. The RUN indicator on the CPU will come on indicating the CPU has entered the run mode. If not, repeat Step 8 ensuring the program is entered properly or refer to the troubleshooting guide in Chapter 9.

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

Steps to Designing a Successful System

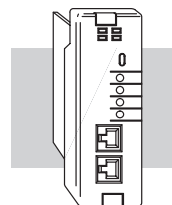
Step 1: Review the Installation Guidelines

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



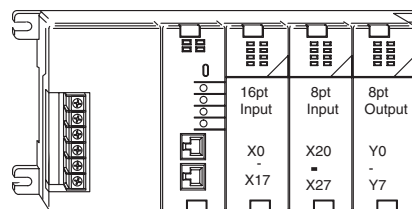
Step 2: Understand the CPU Set-up Procedures

The CPU is the heart of your automation system and is explained in Chapter 3. Make sure you take time to understand the various features and set-up requirements.



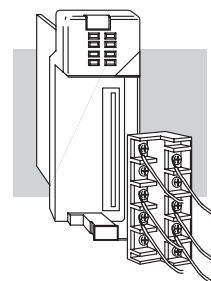
Step 3: Understand the I/O System Configurations

It is important to understand how your local I/O system can be configured. It is also important to understand how the system Power Budget is calculated. This can affect your I/O placement and/or configuration options. See Chapter 4 for more information.



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

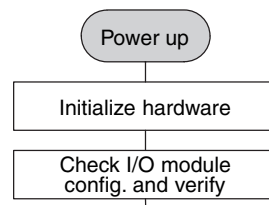
Many different I/O modules are available with the DL205 system. Chapter 2 provides the specifications and wiring diagrams for the discrete I/O modules.



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

Step 5: Understand the System Operation

Before you begin to enter a program, it is very helpful to understand how the DL205 system processes information. This involves not only program execution steps, but also involves the various modes of operation and memory layout characteristics. See Chapter 3 for more information.

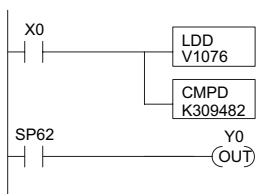


Step 6: Review the Programming Concepts

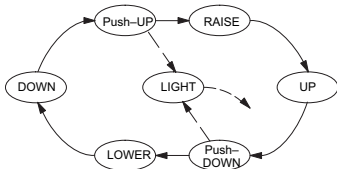
The DL205 provides four main approaches to solving the application program, including the PID loop task depicted in the next figure.

- RLL diagram style programming is the best tool for solving boolean logic and general CPU register/accumulator manipulation. It includes dozens of instructions, which will augment drums, stages and loops.
- The DL250-1 and DL260 have four timer/event drum types, each with up to 16 steps. They offer both time and/or event-based step transitions. Drums are best for a repetitive process based on a single series of steps.
- Stage programming, 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 of your process.
- The DL260 PID loop operation uses set-up tables to configure 16 loops. The DL250-1 PID loop operation uses setup to configure 4 loops. Features include: auto tuning, alarms, SP ramp/soak generation and more.

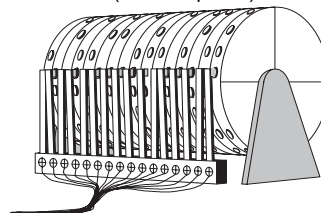
Standard RLL Programming
(see Chapter 5)



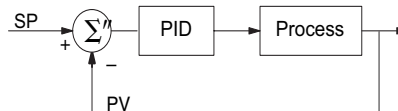
Stage Programming
(see Chapter 7)



Timer/Event Drum Sequencer
(see Chapter 6)



PID Loop Operation
(see Chapter 8)

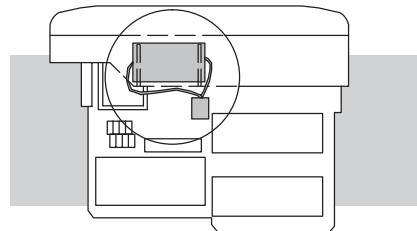
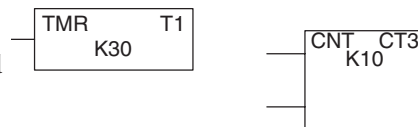


Step 7: Choose the Instructions

Once you have installed the system and understand the theory of operation, you can choose from one of the most powerful instruction sets available.

Step 8: Understand the Maintenance and Troubleshooting Procedures

Equipment failures can occur at any time. Switches fail, batteries need to be replaced, etc. In most cases, the majority of the troubleshooting and maintenance time is spent trying to locate the problem. The DL205 system has many built-in features that help you quickly identify problems. Refer to Chapter 9 for diagnostics.



INSTALLATION, WIRING AND SPECIFICATIONS



CHAPTER 2

In This Chapter:

| | |
|--|------|
| Safety Guidelines..... | 2-2 |
| Mounting Guidelines..... | 2-5 |
| Installing DL205 Bases..... | 2-10 |
| Installing Components in the Base..... | 2-12 |
| Base Wiring Guidelines..... | 2-13 |
| I/O Wiring Strategies..... | 2-14 |
| I/O Modules Position, Wiring, and Specification..... | 2-26 |
| Glossary of Specification Terms..... | 2-51 |

Safety Guidelines

2



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



WARNING: Providing a safe operating environment for personnel and equipment is your responsibility and should be your primary goal during system planning and installation. Automation systems can fail and may result in situations that can cause serious injury to personnel and/or damage equipment. Do not rely on the automation system alone to provide a safe operating environment. Sufficient emergency circuits should be provided to stop either partially or totally the operation of the PLC or the controlled machine or process. These circuits should be routed outside the PLC in the event of controller failure, so that independent and rapid shutdown is available. Devices, such as “mushroom” switches or end of travel limit switches, should operate motor starter, solenoids, or other devices without being processed by the PLC. These emergency circuits should be designed using simple logic with a minimum number of highly reliable electromechanical components. Every automation application is different, so there may be special requirements for your particular application. Make sure to follow all national, state, and local government requirements for the proper installation and use of your equipment.

Plan for Safety

The best way to provide a safe operating environment is to make personnel and equipment safety part of the planning process. You should examine every aspect of the system to determine which areas are critical to operator or machine safety.

If you are not familiar with PLC system installation practices, or your company does not have established installation guidelines, you should obtain additional information from the following sources.

- NEMA — The National Electrical Manufacturers Association, located in Washington, D.C., publishes many different documents that discuss standards for industrial control systems. You can order these publications directly from NEMA. Some of these include:
 - ICS 1, General Standards for Industrial Control and Systems
 - ICS 3, Industrial Systems
 - ICS 6, Enclosures for Industrial Control Systems
- NEC — The National Electrical Code provides regulations concerning the installation and use of various types of electrical equipment. Copies of the NEC Handbook can often be obtained from your local electrical equipment distributor or your local library.
- Local and State Agencies — many local and state governments have additional requirements above and beyond those described in the NEC Handbook. Check with your local Electrical Inspector or Fire Marshall office for information.

Three Levels of Protection

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

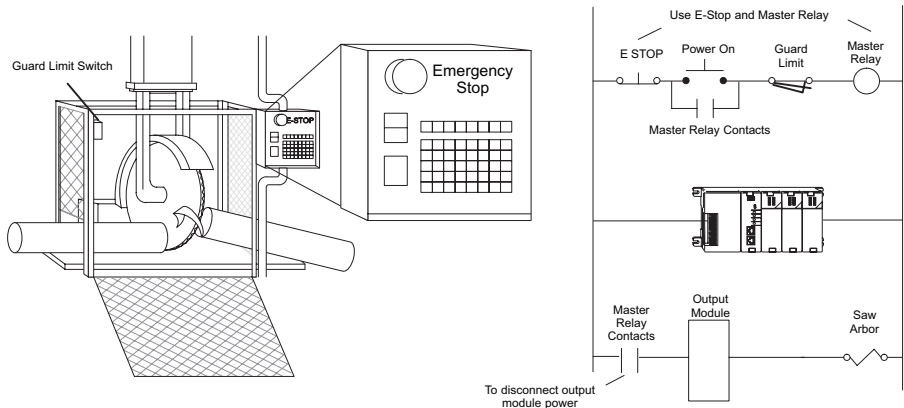
- Emergency stop switch for disconnecting system power
- Mechanical disconnect for output module power
- Orderly system shutdown sequence in the PLC control program

Emergency Stops

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

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

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



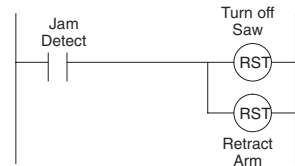
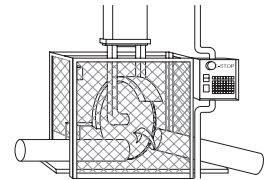
Emergency Power Disconnect

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

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

Orderly System Shutdown

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



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

Class 1, Division 2, Approval

This equipment is suitable for use in Class 1, Zone 2, 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.

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



WARNING: All models used with connector accessories must use R/C (ECBT2) mating plug for all applicable models. All mating plugs shall have suitable ratings for device.

WARNING: This equipment is designed for use in Pollution Degree 2 environments (installed within an enclosure rated at least IP54).

WARNING: Transient suppression must be provided to prevent the rated voltage from being exceeded by 140%.

Mounting Guidelines

Before installing the PLC system, you will need to know the dimensions of the components considered. The diagrams on the following pages provide the component dimensions to use in defining your enclosure specifications. Remember to leave room for potential expansion.

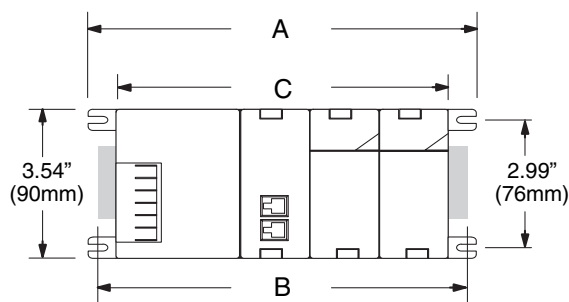
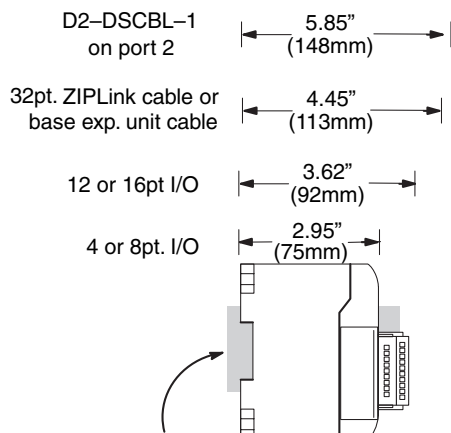


NOTE: If you are using other components in your system, refer to the appropriate manual to determine how those units can affect mounting dimensions.

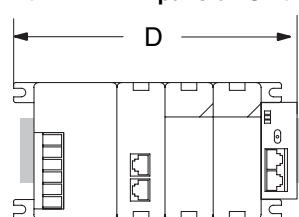
Base Dimensions

The following information shows the proper mounting dimensions. The height dimension is the same for all bases. The depth varies depending on your choice of I/O module. The length varies as the number of slots increase. Make sure you have followed the installation guidelines for proper spacing.

Mounting depths with:



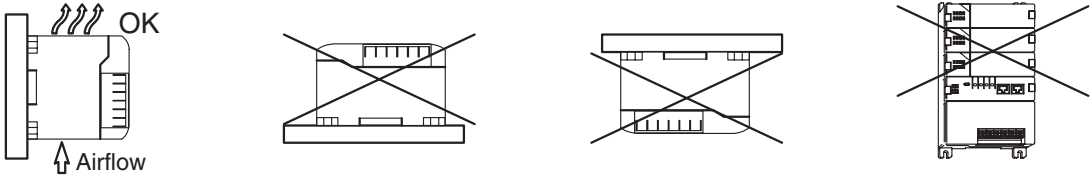
with D2-EM Expansion Unit



| Base | A (Base Total Width) | | B (Mounting Hole) | | C (Component Width) | | D (Width with Exp. Unit) | |
|--------|-------------------------|-------------|----------------------|-------------|------------------------|-------------|-----------------------------|-------------|
| | Inches | Millimeters | Inches | Millimeters | Inches | Millimeters | Inches | Millimeters |
| 3-slot | 6.77 | 172 | 6.41 | 163 | 5.8 | 148 | 7.24 | 184 |
| 4-slot | 7.99 | 203 | 7.63 | 194 | 7.04 | 179 | 8.46 | 215 |
| 6-slot | 10.43 | 265 | 10.07 | 256 | 9.48 | 241 | 10.90 | 277 |
| 9-slot | 14.09 | 358 | 13.74 | 349 | 13.14 | 334 | 14.56 | 370 |

Panel Mounting and Layout

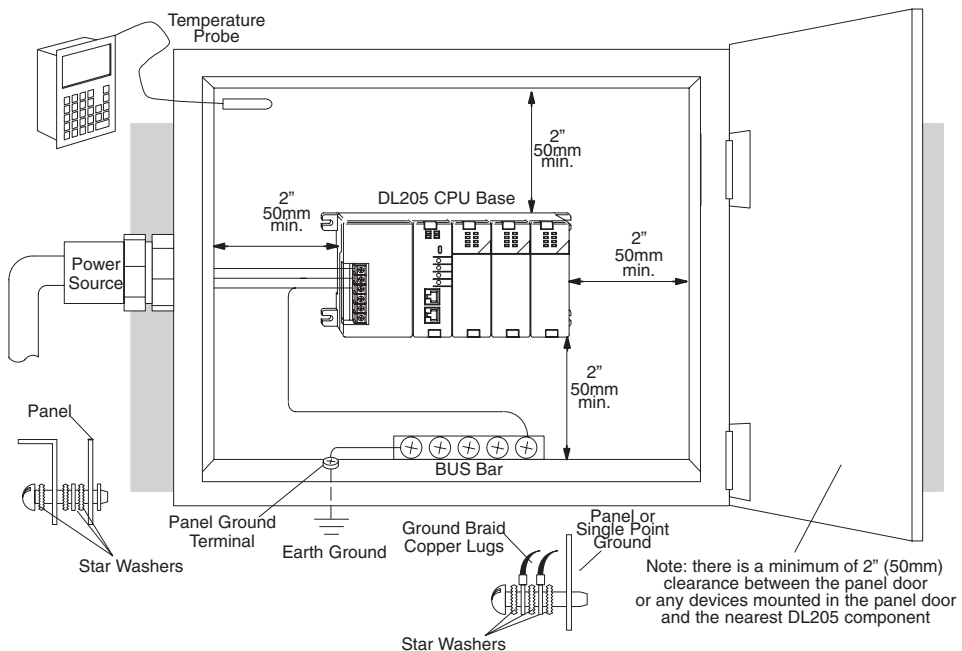
It is important to design your panel properly to help ensure the DL205 products operate within their environmental and electrical limits. The system installation should comply with all appropriate electrical codes and standards. It is important the system also conforms to the operating standards for the application to ensure proper performance. The diagrams below reference the items in the following list.



1. Mount the bases horizontally to provide proper ventilation.
2. If you place more than one base in a cabinet, there should be a minimum of 7.2" (183mm) between bases.
3. Provide a minimum clearance of 2" (50mm) between the base and all sides of the cabinet. There should also be at least 1.2" (30mm) of clearance between the base and any wiring ducts.
4. There must be a minimum of 2" (50mm) clearance between the panel door and the nearest DL205 component.



NOTE: The cabinet configuration below is not suitable for EU installations. Refer to Appendix I, European Union Directives.



5. The ground terminal on the DL205 base must be connected to a single point ground. Use copper stranded wire to achieve a low impedance. Copper eye lugs should be crimped and soldered to the ends of the stranded wire to ensure good surface contact. Remove anodized finishes and use copper lugs and star washers at termination points. A general rule is to achieve a 0.1 ohm of DC resistance between the DL205 base and the single point ground.
6. There must be a single point ground (i.e., copper bus bar) for all devices in the panel requiring an earth ground return. The single point of ground must be connected to the panel ground termination. The panel ground termination must be connected to earth ground. For this connection you should use #12 AWG stranded copper wire at a minimum. Minimum wire sizes, color coding, and general safety practices should comply with appropriate electrical codes and standards for your region. A good common ground reference (Earth ground) is essential for proper operation of the DL205. Methods of providing an adequate common ground reference include:
 - Installing a ground rod as close to the panel as possible.
 - Connection to incoming power system ground.
7. Properly evaluate any installations where the ambient temperature may approach the lower or upper limits of the specifications. Place a temperature probe in the panel, close the door and operate the system until the ambient temperature has stabilized. If the ambient temperature is not within the operating specification for the DL205 system, measures such as installing a cooling/heating source must be taken to get the ambient temperature within the DL205 operating specifications.
8. Device mounting bolts and ground braid termination bolts should be #10 copper bolts or equivalent. Tapped holes instead of nut-bolt arrangements should be used whenever possible. To ensure good contact on termination areas, impediments such as paint, coating or corrosion should be removed in the area of contact.
9. The DL205 system is designed to be powered by 110/220 VAC, 24VDC, or 125VDC normally available throughout an industrial environment. Electrical power in some areas where the PLCs are installed is not always stable and storms can cause power surges. Due to this, powerline filters are recommended for protecting the DL205 PLCs from power surges and EMI/RFI noise. The Automation Powerline Filter, for use with 120VAC and 240VAC, 1–5 Amps, is an excellent choice (can be located at www.automationdirect.com); however, you can use a filter of your choice. These units install easily between the power source and the PLC.

Enclosures

Your selection of a proper enclosure is important to ensure safe and proper operation of your DL205 system. Applications of DL205 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

Environmental Specifications

The following table lists the environmental specifications that generally apply to the DL205 system (CPU, Bases, I/O Modules). The ranges that vary for the Handheld Programmer are noted at the bottom of this chart. I/O module operation may fluctuate depending on the ambient temperature and your application. Refer to the appropriate I/O module specifications for the temperature derating curves applying to specific modules.

| Specification | Rating |
|--------------------------------|--|
| Storage Temperature | -4°F to 158°F (-20°C to 70°C) |
| Ambient Operating Temperature* | 32°F to 131°F (0°C to 55°C) |
| Ambient Humidity** | 30% – 95% relative humidity (non-condensing) |
| Vibration Resistance | MIL STD 810C, Method 514.2 |
| Shock Resistance | MIL STD 810C, Method 516.2 |
| Noise Immunity | NEMA (ICS3-304) |
| Atmosphere | No corrosive gases |

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

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

Power

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

| Specification | AC Powered Bases | 24VDC Powered Bases | 125VDC Powered Bases |
|--|---|---|---|
| Part Numbers | D2-03B-1, D2-04B-1, D2-06B-1 D2-09B-1 | D2-03BDC1-1, D2-04BDC1-1, D2-06BDC1-1, D2-09BDC1-1 | D2-06BDC2-1, D2-09BDC2-1 |
| Input Voltage Range | 100–240 VAC (+10%/ –15%) 50/60Hz | 10.2 – 28.8 VDC (24VDC) with less than 10% ripple | 104–240 VDC +10% –15% |
| Maximum Inrush Current | 30A | 10A | 20A |
| Maximum Power | 80VA | 25W | 30W |
| Voltage Withstand (dielectric) | 1 minute @ 1500VAC between primary, secondary, and field ground | | |
| Insulation Resistance | > 10 MΩ at 500VDC | | |
| Auxiliary 24 VDC Output | 20–28 VDC, less than 1V p-p 300mA max. | None | 20–28 VDC, less than 1V p-p 300mA max. |
| Fusing (internal to base power supply) | Non-replaceable 2A @ 250V slow blow fuse | Non-replaceable 3.15A @ 250V slow blow fuse | Non-replaceable 2A @ 250V slow blow fuse |

Marine Use

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



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

Agency Approvals

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

- UL (Underwriters Laboratories, LLC)
- CSA (Canadian Standards Association)
- FM (Factory Mutual Research Corporation)
- CUL (Underwriters Laboratories of Canada)

24 VDC Power Bases

Follow these additional installation guidelines when installing D2-03BDC1-1, D2-04BDC1-1, D2-06BDC1-1 and D2-09BDC1-1 bases:

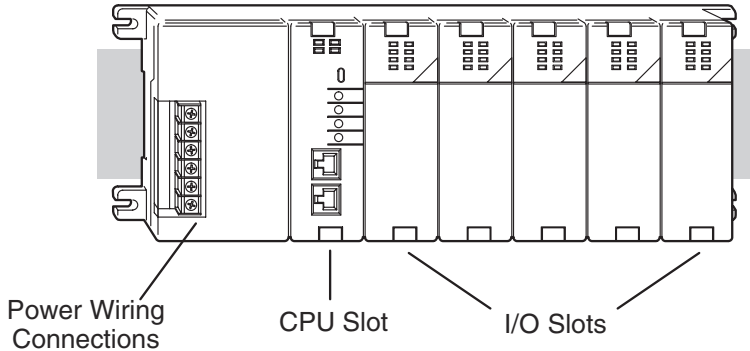
- Install these bases in compliance with the enclosure, mounting, spacing, and segregation requirements of the ultimate application.
- These bases must be used within their marked ratings.
- These bases are intended to be installed within an enclosure rated at least IP54.
- Provisions should be made to prevent the rated voltage being exceeded by transient disturbances of more than 40%.

Installing DL205 Bases

Choosing the Base Type

The DL205 system offers four different sizes of bases and three different power supply options. The following diagram shows an example of a 6-slot base.

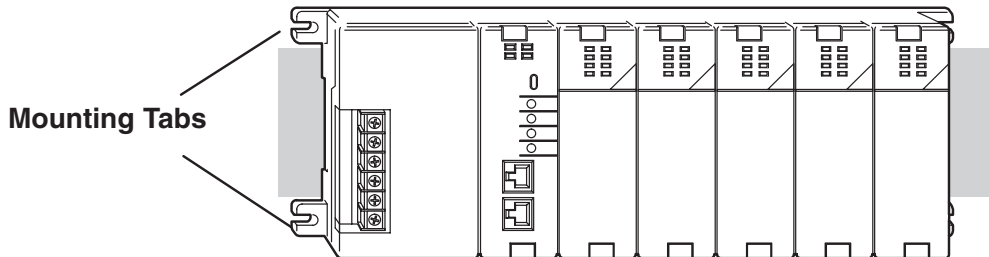
Your choice of base depends on three things:



- Number of I/O modules required
- Input power requirement (AC or DC power)
- Available power budget

Mounting the Base

All I/O configurations of the DL205 may use any of the base configurations. The bases are secured to the equipment panel or mounting location using four M4 screws in the corner tabs of the base. The full mounting dimensions are given in the previous section on Mounting Guidelines.



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

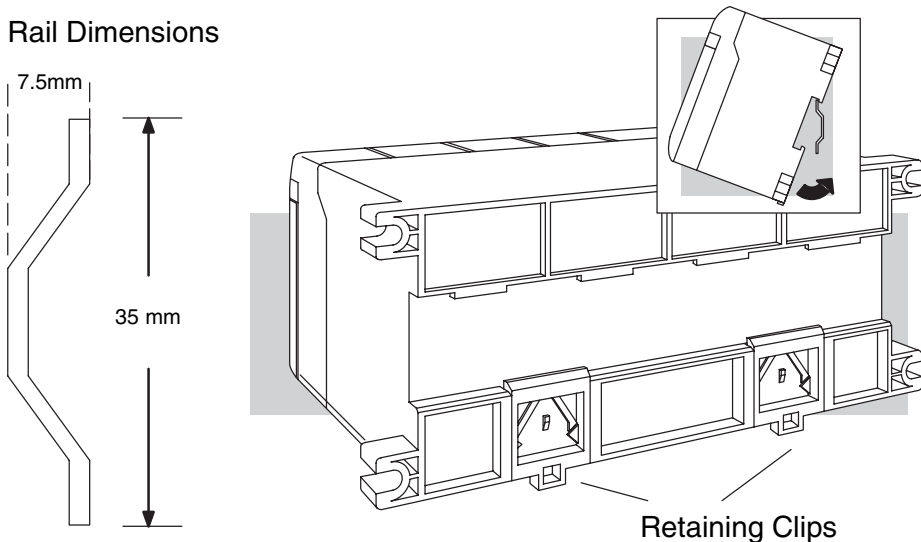
Using Mounting Rails

The DL205 bases can also be secured to the cabinet using mounting rails. You should use rails that conform to DIN EN standard 50 022. Refer to our catalog for a complete line of DIN-rail, DINnectors and DIN-rail mounted apparatus. These rails are approximately 35mm high, with a depth of 7.5 mm. If you mount the base on a rail, you should also consider using end brackets on each end of the rail. The end brackets help keep the base from sliding horizontally along the rail. This helps minimize the possibility of accidentally pulling the wiring loose.

If you examine the bottom of the base, you'll notice small retaining clips. To secure the base to a DIN-rail, place the base onto the rail and gently push up on the retaining clips. The clips lock the base onto the rail.

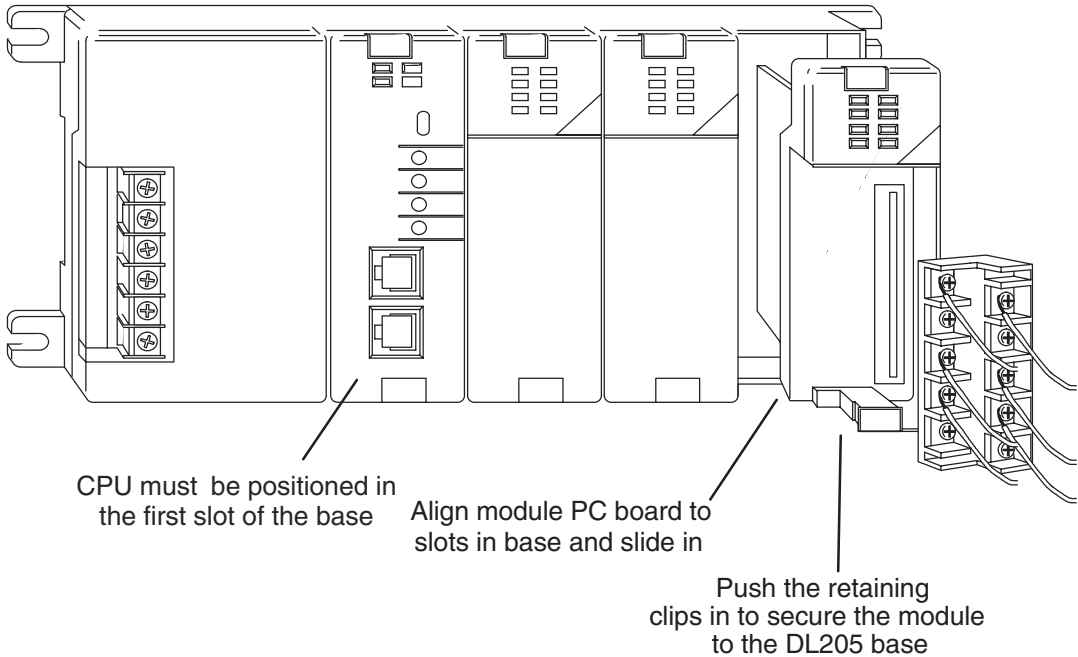
To remove the base, pull down on the retaining clips, lift up on the base slightly, and pull it away from the rail.

DIN Rail Dimensions



Installing Components in the Base

To insert components into the base: first slide the module retaining clips to the out position and align the PC board(s) of the module with the grooves on the top and bottom of the base. Push the module straight into the base until it is firmly seated in the backplane connector. Once the module is inserted into the base, push in the retaining clips to firmly secure the module to the base.



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

Base Wiring Guidelines

Base Wiring

The diagrams show the terminal connections located on the power supply of the DL205 bases. The base terminals can accept up to 16 AWG. You may be able to use larger wiring depending on the type of wire used, but 16 AWG is the recommended size. Do not overtighten the connector screws; the recommended torque value is 7.81 lb·in (0.882 N·m).

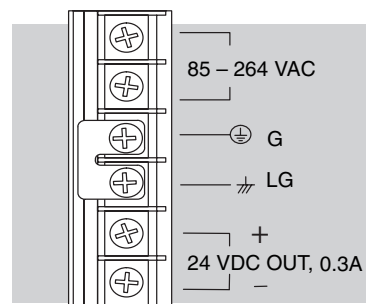


NOTE: You can connect either a 115VAC or 220VAC supply to the AC terminals. Special wiring or jumpers are not required as with some of the other **DirectLOGIC** products.

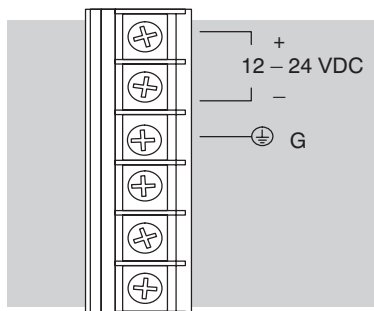


WARNING: Once the power wiring is connected, install the plastic protective cover. When the cover is removed there is a risk of electrical shock if you accidentally touch the wiring or wiring terminals.

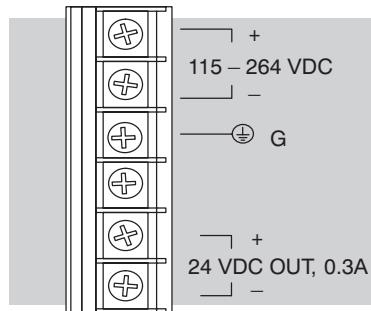
110/220 VAC Base Terminal Strip



12/24 VDC Base Terminal Strip



125 VDC Base Terminal Strip

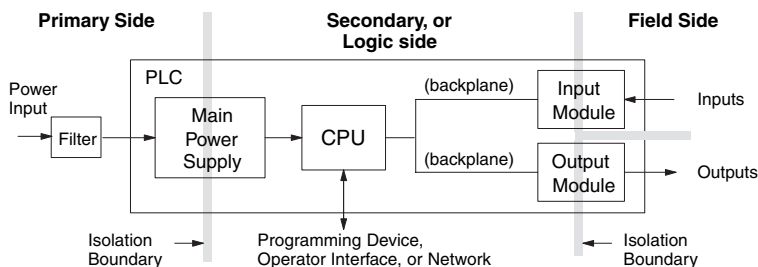


I/O Wiring Strategies

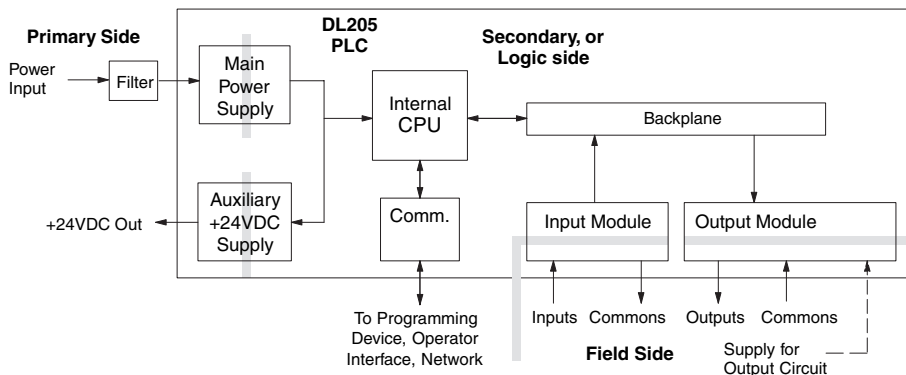
The DL205 PLC system is very flexible and will work in many different wiring configurations. By studying this section before actual installation, you can probably find the best wiring strategy for your application. This will help to lower system cost, wiring errors, and avoid safety problems.

PLC Isolation Boundaries

PLC circuitry is divided into three main regions separated by isolation boundaries, shown in the drawing below. Electrical isolation provides safety, so that a fault in one area does not damage another. A powerline filter will provide isolation between the power source and the power supply. A transformer in the power supply provides magnetic isolation between the primary and secondary sides. Opto-couplers provide optical isolation in Input and Output circuits. This isolates logic circuitry from the field side, where factory machinery connects. Note the discrete inputs are isolated from the discrete outputs, because each is isolated from the logic side. Isolation boundaries protect the operator interface (and the operator) from power input faults or field wiring faults. When wiring a PLC, it is extremely important to avoid making external connections that connect logic side circuits to any other.



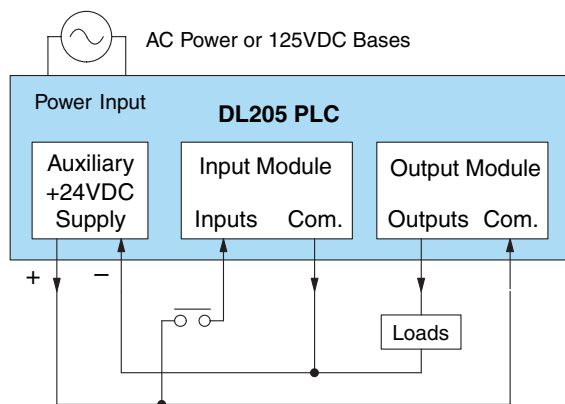
In addition to the basic circuits covered above, AC-powered and 125VDC bases include an auxiliary +24VDC power supply with its own isolation boundary. Since the supply output is isolated from the other three circuits, it can power input and/or output circuits!



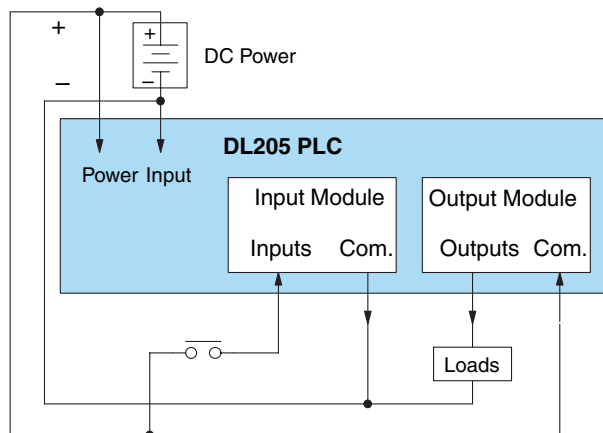
Powering I/O Circuits with the Auxiliary Supply

In some cases, using the built-in auxiliary +24VDC supply can result in a cost savings for your control system. It can power combined loads up to 300mA. Be careful not to exceed the current rating of the supply. If you are the system designer for your application, you may be able to select and design in field devices which can use the +24VDC auxiliary supply.

All AC powered and 125VDC DL205 bases feature the internal auxiliary supply. If input devices AND output loads need +24VDC power, the auxiliary supply may be able to power both circuits as shown in the following diagram.



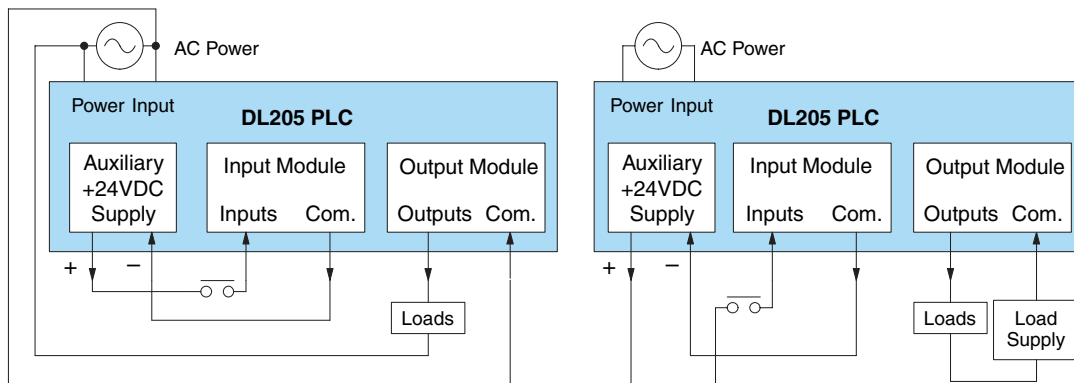
The 12/24 VDC-powered DL205 bases are designed for application environments in which low-voltage DC power is more readily available than AC. These include a wide range of battery-powered applications, such as remotely-located control, in vehicles, portable machines, etc. For this application type, all input devices and output loads typically use the same DC power source. Typical wiring for DC-powered applications is shown in the following diagram.



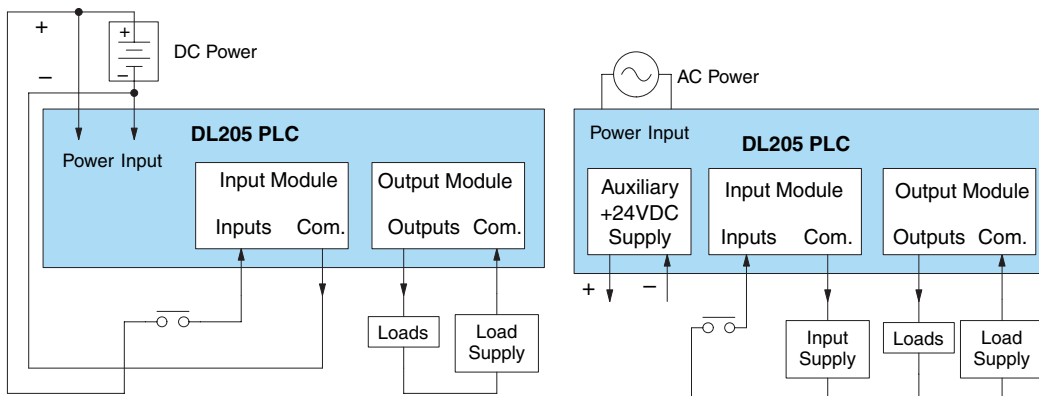
Powering I/O Circuits Using Separate Supplies

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

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



Some applications will use the PLC external power source to also power the input circuit. This typically occurs on DC-powered PLCs, as shown in the drawing below to the left. The inputs share the PLC power source supply, while the outputs have their own separate supply. A worst-case scenario, from a cost and complexity viewpoint, is an application which requires separate power sources for the PLC, input devices, and output loads. The example wiring diagram below on the right shows how this can work, but also the auxiliary supply output is an unused resource. You will want to avoid this situation if possible.



Sinking / Sourcing Concepts

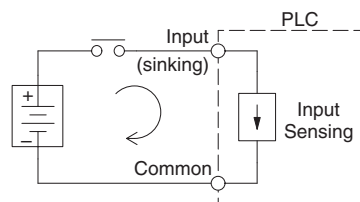
Before going further in the study of wiring strategies, you must have a solid understanding of “sinking” and “sourcing” concepts. Use of these terms occurs frequently in input or output circuit discussions. It is the goal of this section to make these concepts easy to understand, further ensuring your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

Sinking = provides a path to supply ground (–)

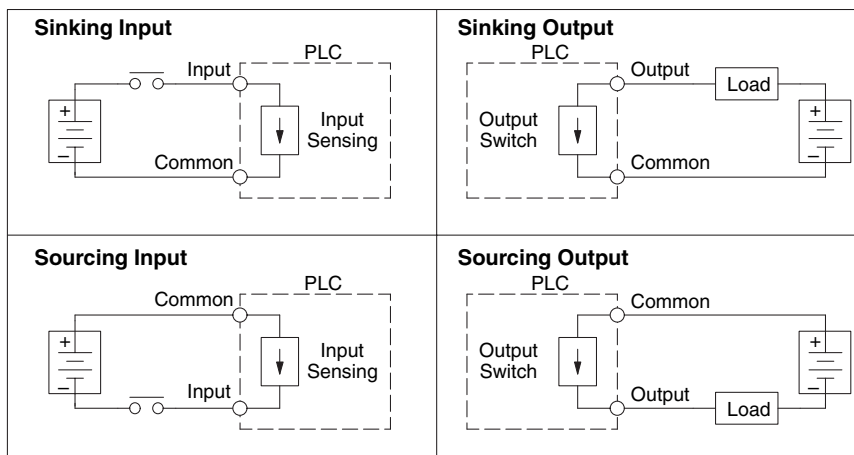
Sourcing = provides a path to supply source (+)

First you will notice these are only associated with DC circuits and not AC, because of the reference to (+) and (–) polarities. Therefore, sinking and sourcing terminology only applies to DC input and output circuits. Input and output points that are sinking only or sourcing only can conduct current in only one direction. This means it is possible to connect the external supply and field device to the I/O point with current trying to flow in the wrong direction, and the circuit will not operate. However, you can successfully connect the supply and field device every time by understanding “sourcing” and “sinking”.

For example, the figure to the right depicts a “sinking” input. To properly connect the external supply, you will have to connect it so the input provides a path to ground (–). Start at the PLC input terminal, follow through the input sensing circuit, exit at the common terminal, and connect the supply (–) to the common terminal. By adding the switch, between the supply (+) and the input, the circuit has been completed. Current flows in the direction of the arrow when the switch is closed.

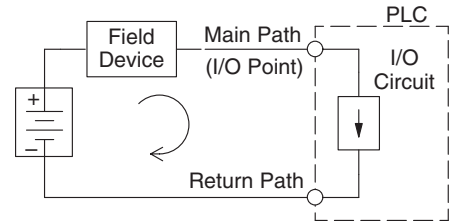


Apply the circuit principle above to the four possible combinations of input/output sinking/sourcing types as shown below. The I/O module specifications at the end of this chapter list the input or output type.

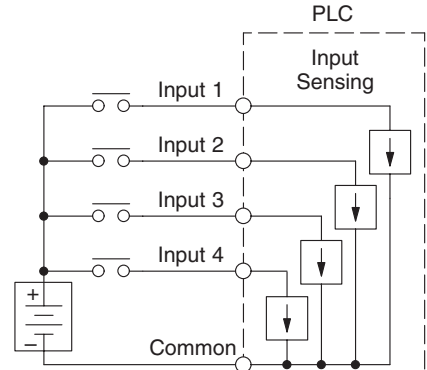


I/O “Common” Terminal Concepts

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



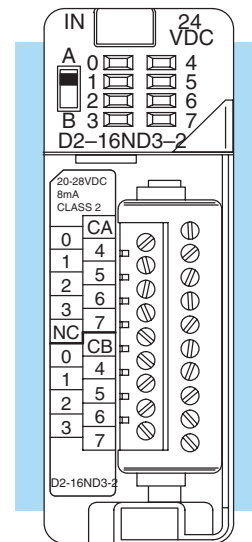
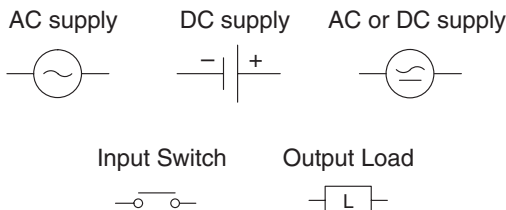
If there was unlimited space and budget for I/O terminals, every I/O point could have two dedicated terminals as the figure above shows. However, providing this level of flexibility is not practical or even necessary for most applications. So, most Input or Output points on PLCs are in groups which share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



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

Most DL205 input and output modules group their I/O points into banks that share a common return path. The best indication of I/O common grouping is on the wiring label, such as the one shown to the right. There are two circuit banks with eight input points in each. The common terminal for each is labeled “CA” and “CB”, respectively.

In the wiring label example, the positive terminal of a DC supply connects to the common terminals. Some symbols you will see on the wiring labels, and their meanings are:

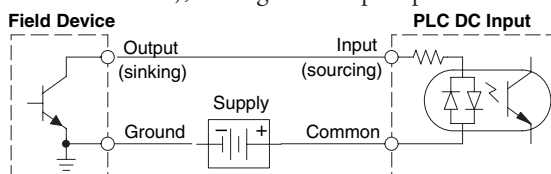


Connecting DC I/O to “Solid State” Field Devices

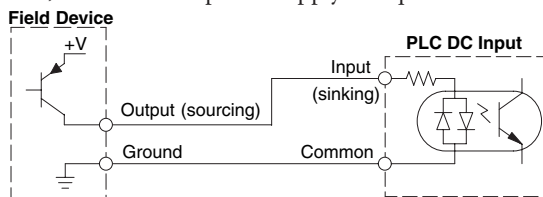
In the previous section on Sourcing and Sinking concepts, the DC I/O circuits were explained to sometimes only allow current to flow one way. This is also true for many of the field devices which have solid-state (transistor) interfaces. In other words, field devices can also be sourcing or sinking.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

Solid State Input Sensors

Several DL205 DC input modules are flexible because they detect current flow in either direction, so they can be wired as either sourcing or sinking. In the following circuit, a field device has an open-collector NPN transistor output. It sinks current from the PLC input point, which sources current. The power supply can be the +24V auxiliary supply or another supply (+12VDC or +24VDC), as long as the input specifications are met.



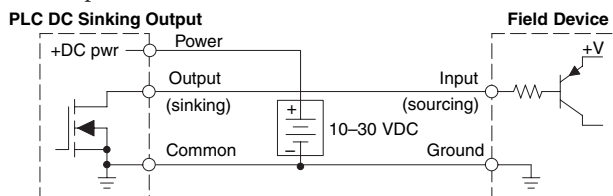
In the next circuit, a field device has an open-collector PNP transistor output. It sources current to the PLC input point, which sinks the current back to ground. Since the field device is sourcing current, no additional power supply is required.



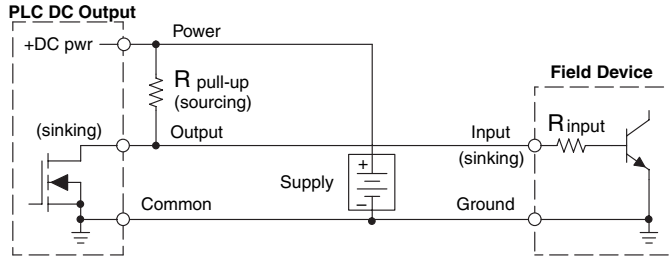
Solid State Output Loads

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

Several of the DL205 DC output modules are the sinking type. This means that each DC output provides a path to ground when it is energized. In the following circuit, the PLC output point sinks current to the output common when energized. It is connected to a sourcing input of a field device input.



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



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

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

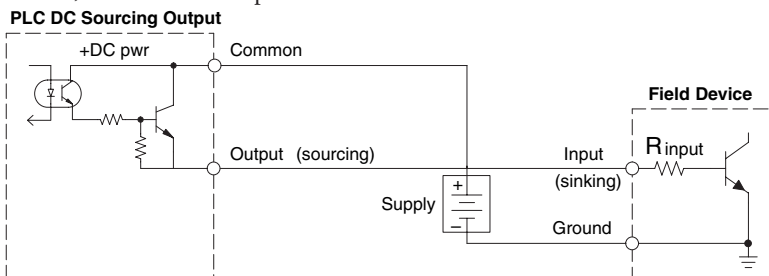
It is important to choose the correct value of $R_{\text{pull-up}}$. In order to do so, you need to know the nominal input current to the field device (I_{input}) when the input is energized. If this value is not known, it can be calculated as shown (a typical value is 15mA). Then use I_{input} and the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

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

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

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

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



Relay Output Guidelines

Several output modules in the DL205 I/O family feature relay outputs: D2–04TRS, D2–08TR, D2–12TR, D2–08CDR, F2–08TR and F2–08TRS. Relays are best for the following applications:

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

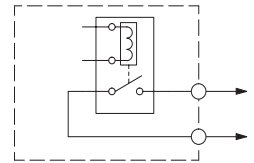
Some applications in which NOT to use relays:

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

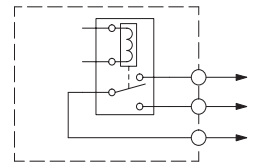
Relay outputs in the DL205 output modules are available in two contact arrangements, shown to the right. The Form A type, or SPST (single pole, single throw) type is normally open and is the simplest to use. The Form C type, or SPDT (single pole, double throw) type has a center contact which moves and a stationary contact on either side. This provides a normally closed contact and a normally open contact.

Some relay output module's relays share common terminals, which connect to the wiper contact in each relay of the bank. Other relay modules have relays which are completely isolated from each other. In all cases, the module drives the relay coil when the corresponding output point is on.

Relay with Form A contacts



Relay with Form C contacts



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

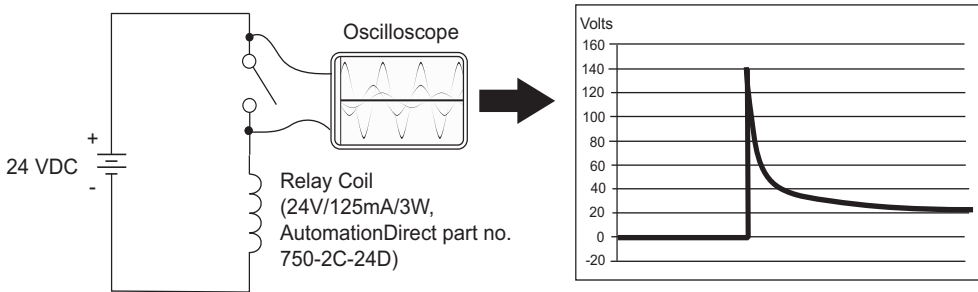
The following pages are intended to give a quick overview of the negative effects of transient voltages on a control system and provide some simple advice on how to effectively minimize them. The need for transient suppression is often not apparent to the newcomers in the automation world. Many mysterious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

What is a Transient Voltage and Why is it Bad?

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

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

Example: Circuit with no Suppression



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

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

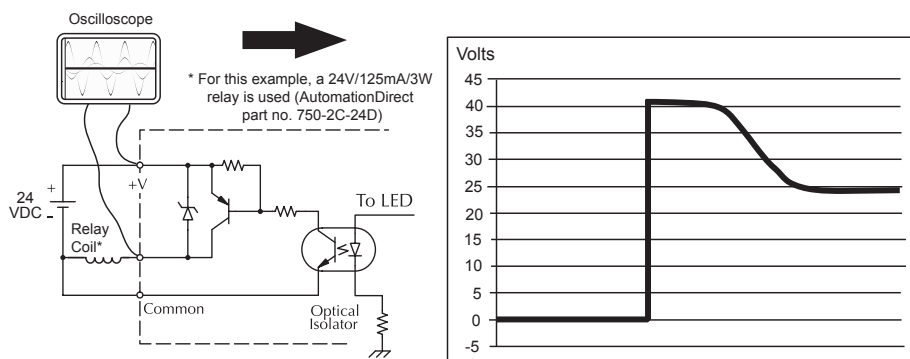
A very destructive side-effect of the arcing across relay contacts is the electromagnetic interference (EMI) it can cause. This occurs because the arcing causes a current surge, which releases RF energy. The entire length of wire between the relay contacts, the coil, and the power source carries the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

PLC's Integrated Transient Suppressors

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

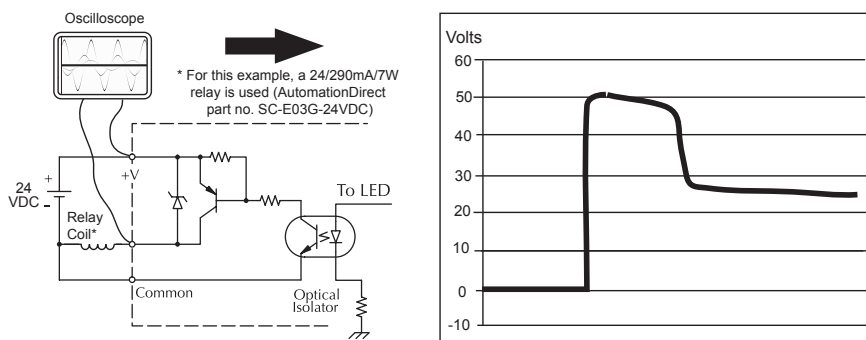
The next example uses the same 24V/125mA/3W relay used earlier. This example measures the PNP transistor output of a D0-06DD2 PLC, which incorporates an integrated Zener diode for transient suppression. Instead of the 140V peak in the first example, the transient voltage here is limited to about 40V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

Example: Small Inductive Load with Only Integrated Suppression



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

Example: Larger Inductive Load with Only Integrated Suppression

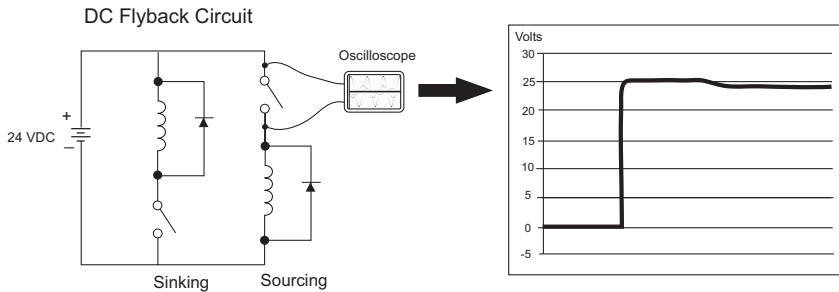


Additional transient suppression should be used in both these examples. If you are unable to measure the transients generated by the connected loads of your control system, using additional transient suppression on all inductive loads would be the safest practice.

Types of Additional Transient Protection

DC Coils:

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



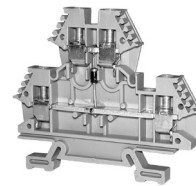
Many AutomationDirect socketed relays and motor starters have add-on flyback diodes that plug or screw into the base, such as the AD-ASMD-250 protection diode module and 784-4C-SKT-1 socket module shown below. If an add-on flyback diode is not available for your inductive load, an easy way to add one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



AD-ASMD-250
Protection Diode Module



784-4C-SKT-1
Relay Socket



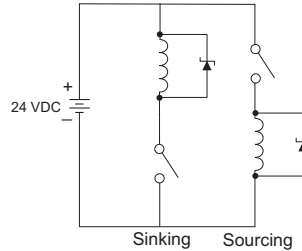
DN-D10DR-A
Diode Terminal Block

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

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



DC MOV or TVS Diode Circuit



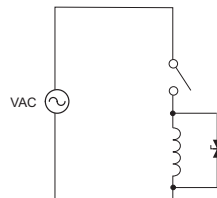
AC Coils:

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

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



AC MOV or Bi-Directional Diode Circuit



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

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

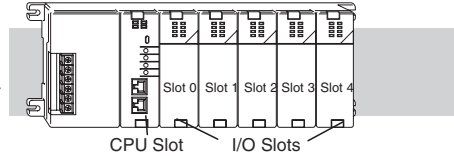
I/O Modules Position, Wiring, and Specification

Slot Numbering

The DL205 bases each provide different numbers of slots for use with the I/O modules. You may notice the bases refer to 3-slot, 4-slot, etc. One of the slots is dedicated to the CPU, so you always have one less I/O slot. For example, you have five I/O slots with a 6-slot base. The I/O slots are numbered 0–4. The CPU slot always contains a PLC CPU or other CPU-slot controller and is not numbered.

Module Placement Restrictions

The following table lists the valid locations for all types of modules in a DL205 system:



| Module/Unit | Local CPU Base | Local Expansion Base | Remote I/O Base |
|--|----------------|----------------------|-----------------|
| CPUs | CPU Slot Only | | |
| DC Input Modules | ✓ | ✓ | ✓ |
| AC Input Modules | ✓ | ✓ | ✓ |
| DC Output Modules | ✓ | ✓ | ✓ |
| AC Output Modules | ✓ | ✓ | ✓ |
| Relay Output Modules | ✓ | ✓ | ✓ |
| Analog Input and Output Modules | ✓ | ✓ | ✓ |
| Local Expansion | | | |
| Base Expansion Module | ✓ | ✓ | |
| Base Controller Module | | CPU Slot Only | |
| Serial Remote I/O | | | |
| Remote Master | ✓ | | |
| Remote Slave Unit | | | CPU Slot Only |
| Ethernet Remote Master | ✓ | | |
| CPU Interface | | | |
| Ethernet Base Controller | Slot 0 Only | | Slot 0 Only* |
| WinPLC | Slot 0 Only | | |
| DeviceNet | Slot 0 Only | | |
| Profibus | Slot 0 Only | | |
| SDS | Slot 0 Only | | |
| Specialty Modules | | | |
| Counter Interface | Slot 0 Only | | |
| Counter I/O | ✓ | | ✓* |
| Data Communications | ✓ | | |
| Ethernet Communications | ✓ | | |
| BASIC CoProcessor | ✓ | | |
| Simulator | ✓ | ✓ | ✓ |
| Filler | ✓ | ✓ | ✓ |

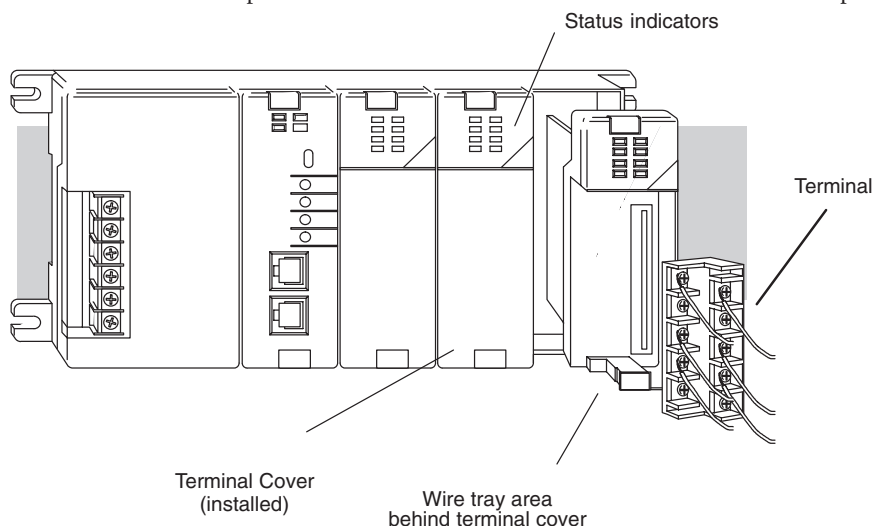
* When used with H2-ERM(100) Ethernet Remote I/O system

Special Placement Considerations for Analog Modules

In most cases, the analog modules can be placed in any slot. However, the placement can also depend on the type of CPU you are using and the other types of modules installed to the left of the analog modules. If you're using a DL230 CPU (or a DL240 CPU with firmware earlier than V1.4) you should check the DL205 Analog I/O Manual for any possible placement restrictions related to your particular module. You can order the DL205 Analog I/O Manual by ordering part number D2-ANLG-M.

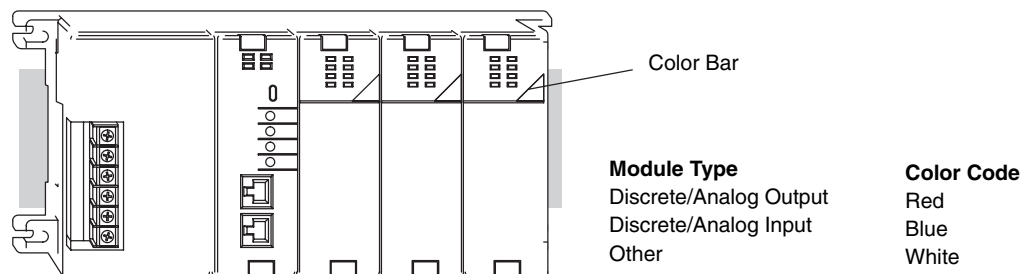
Discrete Input Module Status Indicators

The discrete modules provide LED status indicators to show the status of the input points.



Color Coding of I/O Modules

The DL205 family of I/O modules have a color coding scheme to help you quickly identify if a module is either an input module, output module, or a specialty module. This is done through a color bar indicator located on the front of each module. The color scheme is listed below:



Wiring the Different Module Connectors

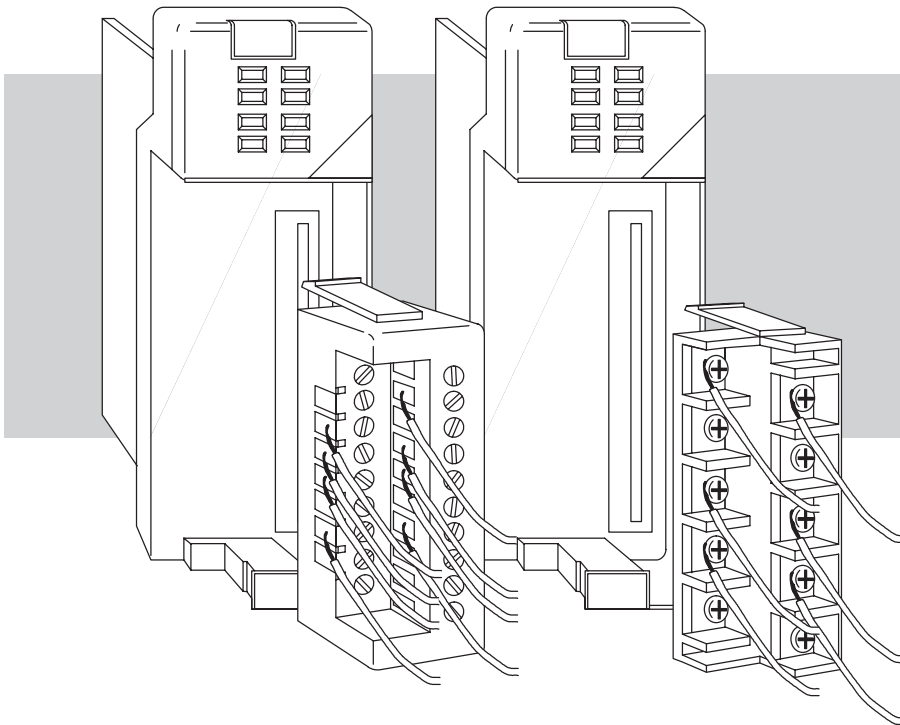
There are two types of module connectors for the DL205 I/O. Some modules have normal screw terminal connectors. Other modules have connectors with recessed screws. The recessed screws help minimize the risk of someone accidentally touching active wiring.

Both types of connectors can be easily removed. If you examine the connectors closely, you'll notice there are squeeze tabs on the top and bottom. To remove the terminal block, press the squeeze tabs and pull the terminal block away from the module.

We also have DIN rail mounted terminal blocks, DINnectors (refer to our catalog for a complete listing of all available products). *ZIPLinks* come with special pre-assembled cables with the I/O connectors installed and wired.



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



I/O Wiring Checklist

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

1. There is a limit to the size of wire the modules can accept. The table below lists the suggested AWG for each module type. When making terminal connections, follow the suggested torque values.

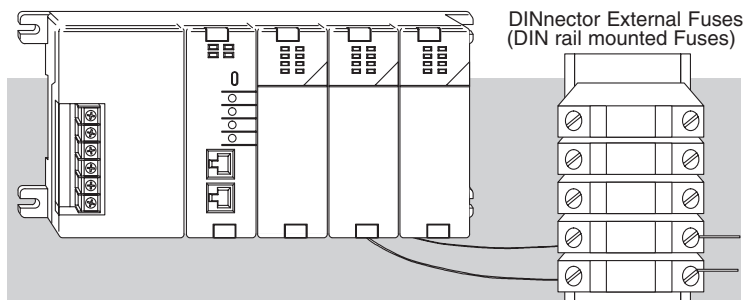
| Terminal type | Suggested AWG Range | Suggested Torque |
|-----------------------|---------------------|-------------------------|
| 10-Terminal Fixed | 14 – 24 AWG | 3.5 lb-inch (0.4 N-m) |
| 10-Terminal Removable | 16* – 24 AWG | 7.81 lb-inch (0.88 N-m) |
| 20-Terminal Removable | 16* – 24 AWG | 2.65 lb-in (0.3 N-m) |

2



***NOTE: 16 AWG Type TFFN or Type MTW is recommended.** Other types of 16 AWG may be acceptable, but it really depends on the thickness and stiffness of the wire insulation. **If the insulation is too thick or stiff and a majority of the module's I/O points are used, then the plastic terminal cover may not close properly or the connector may pull away from the module. This applies especially for high temperature thermoplastics such as THHN.**

2. Always use a continuous length of wire, do not combine 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. Also, avoid running input wiring close to output wiring where possible.
6. To minimize voltage drops when wires must run a long distance, consider using multiple wires for the return line.
7. Avoid running DC wiring in close proximity to AC wiring where possible.
8. Avoid creating sharp bends in the wires.
9. To reduce the risk of having a module with a blown fuse, we suggest you add external fuses to your I/O wiring. A fast blow fuse, with a lower current rating than the I/O module fuse can be added to each common, or a fuse with a rating of slightly less than the maximum current per output point can be added to each output. Refer to our catalog for a complete line of DINnectors, DIN-rail mounted fuse blocks.

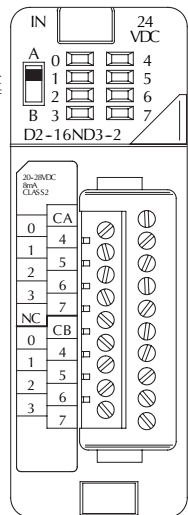
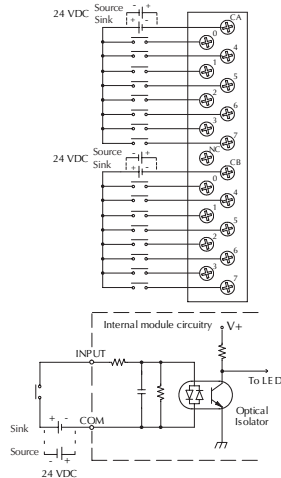
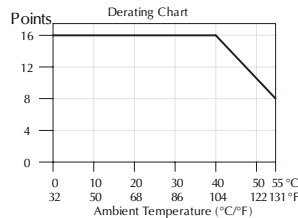
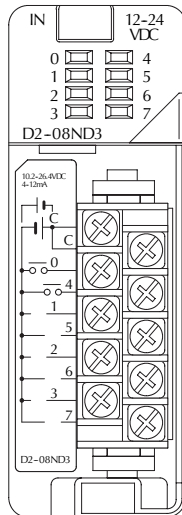
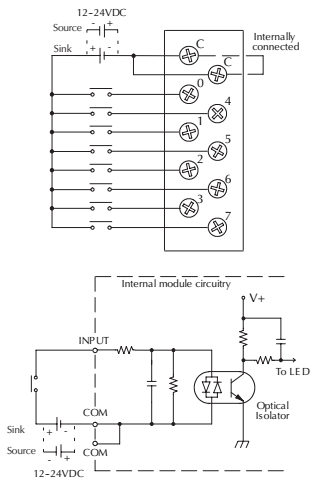
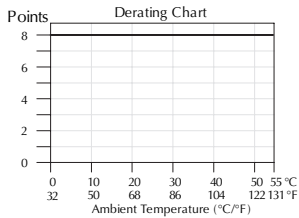


D2-08ND3, DC Input

| D2-08ND3 DC Input | |
|---------------------------------|----------------------------------|
| Inputs per Module | 8 (sink/source) |
| Commons per Module | 1 (2 I/O terminal points) |
| Input Voltage Range | 10.2–26.4 VDC |
| Peak Voltage | 26.4 VDC |
| ON Voltage Level | 9.5 VDC minimum |
| OFF Voltage Level | 3.5 VDC maximum |
| AC Frequency | N/A |
| Input Impedance | 2.7 k Ω |
| Input Current | 4.0 mA @ 12VDC 8.5 mA @ 24VDC |
| Minimum ON Current | 3.5 mA |
| Maximum OFF Current | 1.5 mA |
| Base Power Required 5VDC | 50mA |
| OFF to ON Response | 1 to 8 ms |
| ON to OFF Response | 1 to 8 ms |
| Terminal Type (included) | Removable, D2-8IOCON |
| Status Indicator | Logic side |
| Weight | 2.3 oz. (65g) |

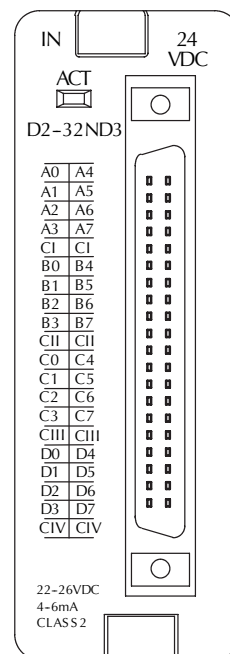
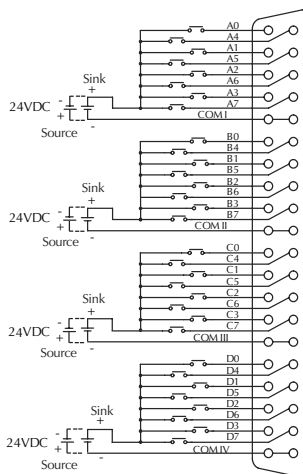
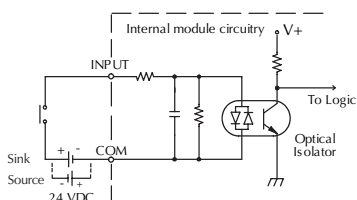
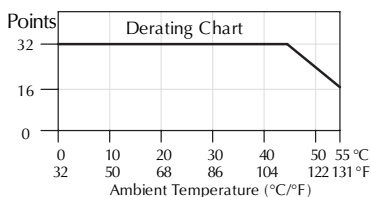
D2-16ND3-2, DC Input

| D2-16ND3-2 DC Input | |
|---------------------------------|---|
| Inputs per Module | 16 (sink/source) |
| Commons per Module | 2 isolated (8 I/O terminal points/com) |
| Input Voltage Range | 20–28 VDC |
| Peak Voltage | 30VDC (10mA) |
| ON Voltage Level | 19 VDC minimum |
| OFF Voltage Level | 7VDC maximum |
| AC Frequency | N/A |
| Input Impedance | 3.9 k Ω |
| Input Current | 6mA @ 24VDC |
| Minimum ON Current | 3.5 mA |
| Maximum OFF Current | 1.5 mA |
| Base Power Required 5VDC | 100mA |
| OFF to ON Response | 3 to 9 ms |
| ON to OFF Response | 3 to 9 ms |
| Terminal Type (included) | Removable, D2-16IOCON |
| Status Indicator | Logic side |
| Weight | 2.3 oz. (65g) |



D2-32ND3, DC Input

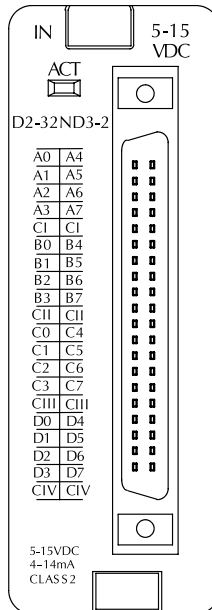
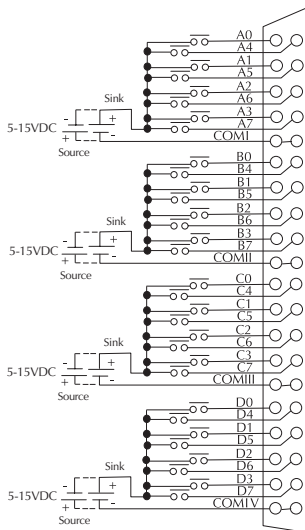
| D2-32ND3 DC Input | |
|--|--|
| Inputs per Module | 32 (sink/source) |
| Commons per Module | 4 isolated (8 I/O terminal points / com) |
| Input Voltage Range | 20-28 VDC |
| Peak Voltage | 30VDC |
| ON Voltage Level | 19VDC minimum |
| OFF Voltage Level | 7VDC maximum |
| AC Frequency | N/A |
| Input Impedance | 4.8 kΩ |
| Input Current | 8.0 mA @ 24 VDC |
| Minimum ON Current | 3.5 mA |
| Maximum OFF Current | 1.5 mA |
| Base Power Required 5VDC | 25mA |
| OFF to ON Response | 3 to 9 ms |
| ON to OFF Response | 3 to 9 ms |
| Terminal Type (not included) | Removable 40-pin Connector ¹ |
| Status Indicator | Module Activity LED |
| Weight | 2.1 oz. (60 g) |
| ¹ Connector sold separately. See Terminal Blocks and Wiring for wiring options. | |



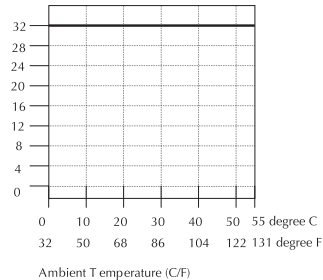
D2-32ND3-2, DC Input

2

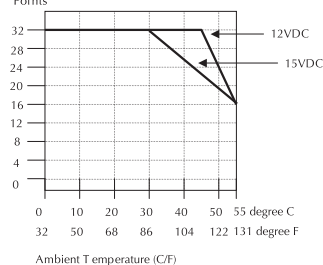
| D2-32ND3-2 DC Input | |
|---|--|
| Inputs per Module | 32 (Sink/Source) |
| Commons per Module | 4 isolated (8 I/O terminal points / com) |
| Input Voltage Range | 4.50 to 15.6 VDC min. to max. |
| Peak Voltage | 16VDC |
| ON Voltage Level | 4VDC minimum |
| OFF Voltage Level | 2VDC maximum |
| AC Frequency | N/A |
| Input Impedance | 1.0 k Ω @ 5-5 VDC |
| Input Current | 4mA @ 5VDC 11mA @ 12VDC 14mA @ 15VDC |
| Maximum Input Current | 16mA @ 15.6 VDC |
| Minimum ON Current | 3mA |
| Maximum OFF Current | 0.5 mA |
| Base Power Required 5VDC | 25mA |
| OFF to ON Response | 3 to 9 ms |
| ON to OFF Response | 3 to 9 ms |
| Terminal Type (not included) | Removable 40-pin connector ¹ |
| Status Indicator | Module activity LED |
| Weight | 2.1 oz. (60g) |
| ¹ Connector sold separately. See Terminal Blocks and Wiring for wiring options. | |



Derating Chart
Input Voltage: 5VDC
Points

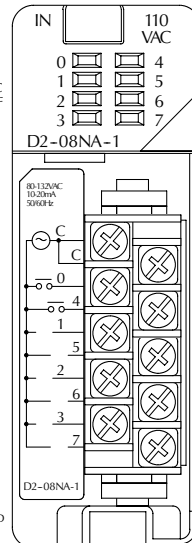
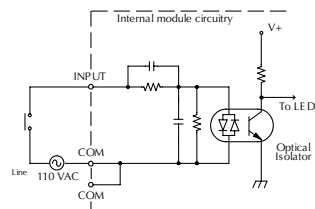
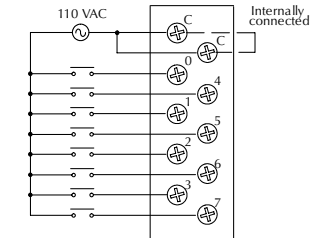
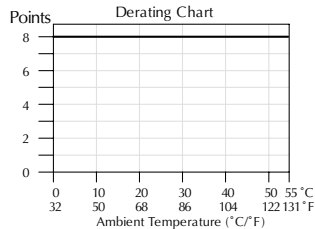


Derating Chart
Input Voltage: 12VDC and 15VDC
Points



D2-08NA-1, AC Input

| D2-08NA-1 AC Input | |
|---------------------------------|---|
| Inputs per Module | 8 |
| Commons per Module | 1 (2 I/O terminal points) |
| Input Voltage Range | 80–132 VAC |
| Peak Voltage | 132VAC |
| ON Voltage Level | 75VAC minimum |
| OFF Voltage Level | 20VAC maximum |
| AC Frequency | 47–63 Hz |
| Input Impedance | 12k Ω @ 60Hz |
| Input Current | 13mA @ 100VAC, 60Hz 11mA @ 100 VAC, 50Hz |
| Minimum ON Current | 5mA |
| Maximum OFF Current | 2mA |
| Base Power Required 5VDC | 50mA |
| OFF to ON Response | 5 to 30 ms |
| ON to OFF Response | 10 to 50 ms |
| Terminal Type (included) | Removable; D2-8IOCON |
| Status Indicator | Logic side |
| Weight | 2.5 oz. (70g) |

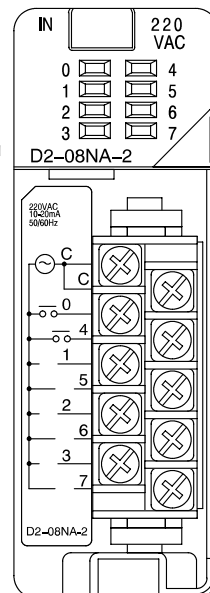
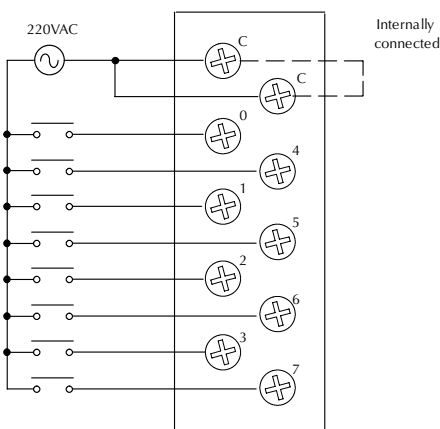
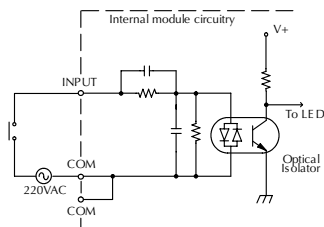
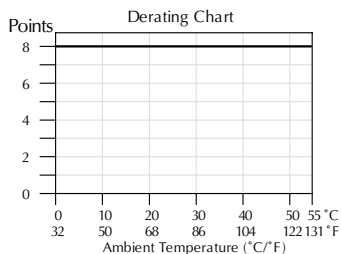


D2-08NA-2, AC Input

D2-08NA-2 AC Input

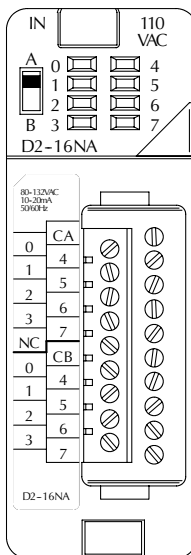
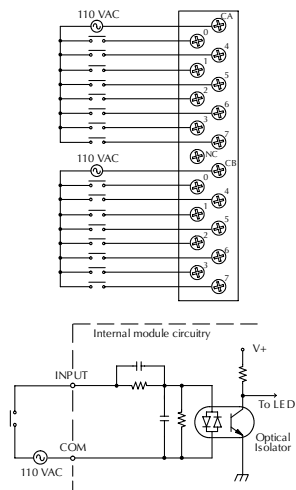
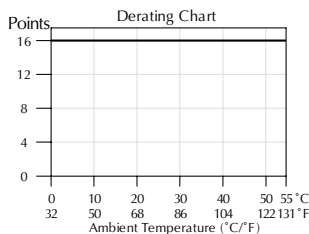
| | |
|---------------------------------|---|
| Inputs per Module | 8 |
| Commons per Module | 1 (2 I/O terminal points) |
| Input Voltage Range | 170–265 VAC |
| Peak Voltage | 265VAC |
| ON Voltage Level | 150VAC minimum |
| OFF Voltage Level | 40VAC maximum |
| AC Frequency | 47–63 Hz |
| Input Impedance | 18k Ω @ 60Hz |
| Input Current | 9mA @ 220VAC, 50Hz 11mA @ 265VAC, 50Hz 10mA @ 220VAC, 60Hz 12mA @ 265VAC, 60Hz |
| Minimum ON Current | 10mA |
| Maximum OFF Current | 2mA |
| Base Power Required 5VDC | 100mA |
| OFF to ON Response | 5 to 30 ms |
| ON to OFF Response | 10 to 50 ms |
| Terminal Type (included) | Removable; D2-8IOCON |
| Status Indicator | Logic side |
| Weight | 2.5 oz. (70g) |

| | |
|-------------------------------------|---|
| Operating Temperature | 32°F to 131°F (0° to 55°C) |
| Storage Temperature | -4°F to 158°F (-20°C to 70°C) |
| Humidity | 35% to 95% (non-condensing) |
| Atmosphere | No corrosive gases permitted |
| Vibration | MIL STD 810C 514.2 |
| Shock | MIL STD 810C 516.2 |
| Insulation Withstand Voltage | 1,500VAC 1 minute (COM-GND) |
| Insulation Resistance | 10M \approx @ 500VDC |
| Noise Immunity | NEMA 1,500V 1 minute SANKI 1,000V 1 minute |
| RFI | 150MHz, 430MHz |



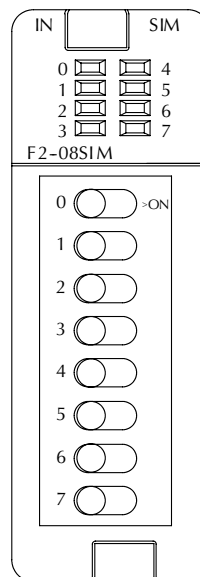
D2-16NA, AC Input

| D2-16NA AC Input | |
|---------------------------------|---|
| Inputs per Module | 16 |
| Commons per Module | 2 (isolated) |
| Input Voltage Range | 80–132 VAC |
| Peak Voltage | 132VAC |
| ON Voltage Level | 70VAC minimum |
| OFF Voltage Level | 20VAC maximum |
| AC Frequency | 47–63 Hz |
| Input Impedance | 12k Ω @ 60Hz |
| Input Current | 11mA @ 100VAC, 50Hz 13mA @ 100VAC, 60Hz 15mA @ 132VAC, 60Hz |
| Minimum ON Current | 5mA |
| Maximum OFF Current | 2mA |
| Base Power Required 5VDC | 100mA |
| OFF to ON Response | 5 to 30 ms |
| ON to OFF Response | 10 to 50 ms |
| Terminal Type (included) | Removable; D2-16IOCON |
| Status Indicator | Logic side |
| Weight | 2.4 oz. (68g) |



F2-08SIM, Input Simulator

| F2-08SIM Input Simulator | |
|---------------------------------|----------------|
| Inputs per Module | 8 |
| Base Power Required 5VDC | 50mA |
| Terminal Type | None |
| Status Indicator | Switch side |
| Weight | 2.65 oz. (75g) |

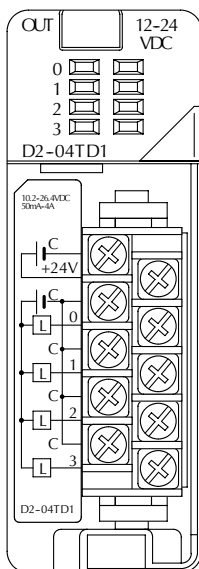
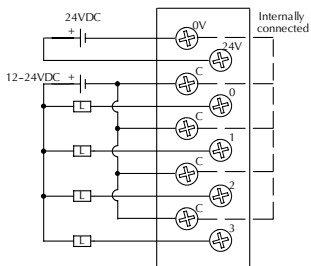
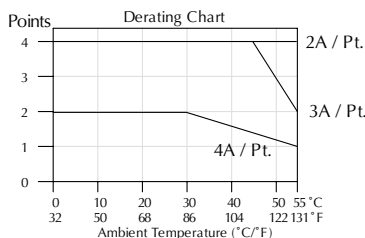


D2-04TD1, DC Output

D2-04TD1 DC Output

| | |
|-------------------------------------|-----------------------------------|
| Outputs per Module | 4 (current sinking) |
| Output Points Consumed | 8 points (only first 4 pts. used) |
| Commons per Module | 1 (4 I/O terminal points) |
| Output Type | NMOS FET (open drain) |
| Operating Voltage | 10.2-26.4 VDC |
| Peak Voltage | 40VDC |
| ON Voltage Drop | 0.72 VDC maximum |
| AC Frequency | N/A |
| Max Load Current (resistive) | 4A/point 8A/common |
| Max Leakage Current | 0.1 mA @ 40 VDC |
| Max Inrush Current | 6A for 100 ms, 15A for 10 ms |
| Minimum Load Current | 50 mA |

| | |
|---------------------------------|---|
| External DC Required | 24VDC @ 20 mA max. |
| Base Power Required 5VDC | 60mA |
| OFF to ON Response | 1ms |
| ON to OFF Response | 1ms |
| Terminal Type (included) | Removable; D2-8IOCON |
| Status Indicator | Logic side |
| Weight | 2.8 oz. (80 g) |
| Fuses | 4 (1 per point) (6.3 A slow blow, non-replaceable) |



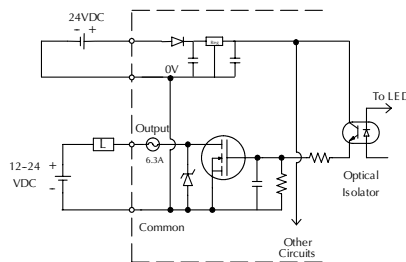
Inductive Load
Maximum Number of Switching Cycles per Minute

| Load Current | Duration of output in ON state 7ms | 40ms | 100ms |
|--------------|---------------------------------------|------|-------|
| 0.1A | 8000 | 1400 | 600 |
| 0.5A | 1600 | 300 | 120 |
| 1.0A | 800 | 140 | 60 |
| 1.5A | 540 | 90 | 35 |
| 2.0A | 400 | 70 | - |
| 3.0A | 270 | - | - |
| 4.0A | 200 | - | - |

At 40 ms duration, loads of 3.0A or greater cannot be used.

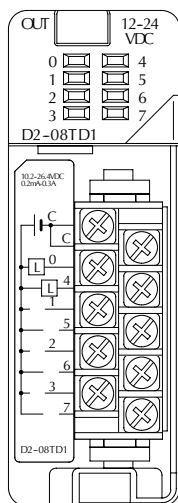
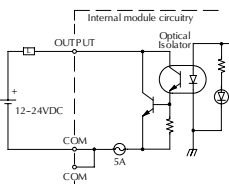
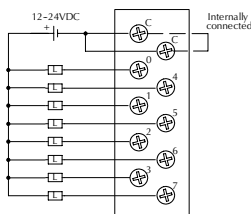
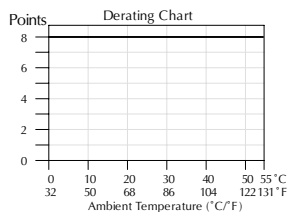
At 100 ms duration, loads of 2.0A or greater cannot be used.

Find the load current you expect to use and the duration that the output is ON. The number at the intersection of the row and column represents the switching cycles per minute. For example, a 1A inductive load that is on for 100 ms can be switched on and off a maximum of 60 times per minute. To convert this to duty cycle percentage use: (duration x cycles)/60. In this example, (60 x .1)/60 = .1, or 10% duty cycle.



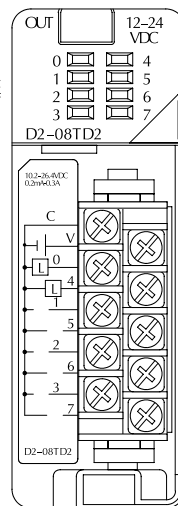
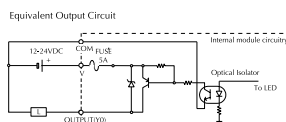
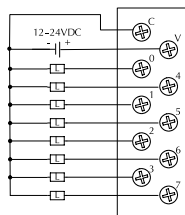
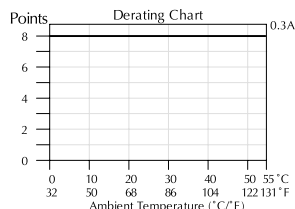
D2-08TD1, DC Output

| D2-08TD1 DC Output | |
|---------------------------------|---|
| Outputs per Module | 8 (current sinking) |
| Commons per Module | 1 (2 I/O terminal points) |
| Output Type | NPN open collector |
| Operating Voltage | 10.2–26.4 VDC |
| Peak Voltage | 40VDC |
| ON Voltage Drop | 1.5 VDC maximum |
| AC Frequency | N/A |
| Minimum Load Current | 0.5 mA |
| Max Load Current | 0.3 A/point; 2.4 A/common |
| Max Leakage Current | 0.1 mA @ 40VDC |
| Max Inrush Current | 1A for 10ms |
| Base Power Required 5VDC | 100mA |
| OFF to ON Response | 1ms |
| ON to OFF Response | 1ms |
| Terminal Type (included) | Removable; D2-8IOCON |
| Status Indicator | Logic side |
| Weight | 2.3 oz. (65g) |
| Fuses | 1 per common 5A fast blow, non-replaceable |



D2-08TD2, DC Output

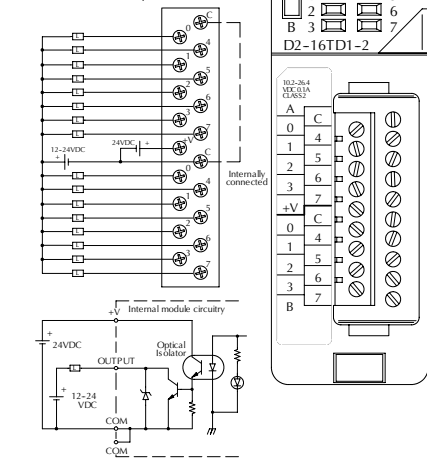
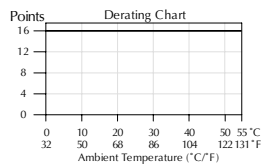
| D2-08TD2 DC Output | |
|---------------------------------|---|
| Outputs per Module | 8 (current sourcing) |
| Commons per Module | 1 |
| Output Type | PNP open collector |
| Operating Voltage | 12–24 VDC |
| Output Voltage | 10.8–26.4 VDC |
| Peak Voltage | 40VDC |
| ON Voltage Drop | 1.5 VDC |
| AC Frequency | N/A |
| Minimum Load Current | N/A |
| Max Load Current | 0.3 A per point; 2.4 A per common |
| Max Leakage Current | 1.0 mA @ 40VDC |
| Max Inrush Current | 1A for 10 ms |
| Base Power Required 5VDC | 100 mA |
| OFF to ON Response | 1ms |
| ON to OFF Response | 1ms |
| Terminal Type (included) | Removable; D2-8IOCON |
| Status Indicator | Logic side |
| Weight | 2.1 oz. (60g) |
| Fuses | 1 per common 5A fast blow, non-replaceable |



D2-16TD1-2, DC Output

D2-16TD1-2 DC Output

| | |
|---------------------------------|----------------------------|
| Outputs per Module | 16 (current sinking) |
| Commons per Module | 1 (2 I/O terminal points) |
| Output Type | NPN open collector |
| External DC required | 24VDC \pm 4V @ 80 mA max |
| Operating Voltage | 10.2-26.4 VDC |
| Peak Voltage | 30VDC |
| ON Voltage Drop | 0.5 VDC maximum |
| AC Frequency | N/A |
| Minimum Load Current | 0.2 mA |
| Max Load Current | 0.1A/point 1.6A/common |
| Max Leakage Current | 0.1 mA @ 30 VDC |
| Max Inrush Current | 150mA for 10 ms |
| Base Power Required 5VDC | 200mA |
| OFF to ON Response | 0.5 ms |
| ON to OFF Response | 0.5 ms |
| Terminal Type (included) | Removable; D2-16IOCON |
| Status Indicator | Logic side |
| Weight | 2.3 oz. (65g) |
| Fuses | None |

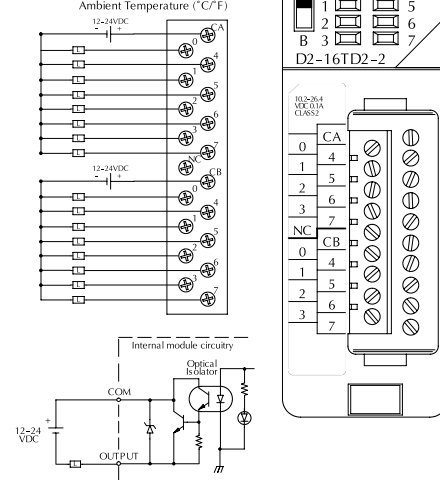
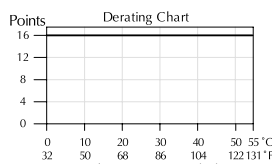


* Can also be used with 5VDC supply

D2-16TD2-2, DC Output

D2-16TD2-2 DC Output

| | |
|---------------------------------|---------------------------|
| Outputs per Module | 16 (current sourcing) |
| Commons per Module | 2 |
| Output Type | NPN open collector |
| Operating Voltage | 10.2-26.4 VDC |
| Peak Voltage | 30VDC |
| ON Voltage Drop | 1.0 VDC maximum |
| AC Frequency | N/A |
| Minimum Load Current | 0.2 mA |
| Max Load Current | 0.1A/point 1.6A/module |
| Max Leakage Current | 0.1 mA @ 30 VDC |
| Max Inrush Current | 150mA for 10 ms |
| Base Power Required 5VDC | 200mA |
| OFF to ON Response | 0.5 ms |
| ON to OFF Response | 0.5 ms |
| Terminal Type (included) | Removable; D2-16IOCON |
| Status Indicator | Logic side |
| Weight | 2.8 oz. (80g) |
| Fuses | None |



F2-16TD1(2)P, DC Output With Fault Protection



NOTE: Not supported in D2-230, D2-240 and D2-250 CPUs.

These modules detect the following fault status and turn the related X bit(s) on.

1. Missing external 24VDC for the module
2. Open load¹
3. Over temperature (the output is shut down)
4. Over load current (the output is shut down)

| Fault Status | X bit Fault Status Indication |
|------------------------|---|
| Missing external 24VDC | All 16 X bits are on. |
| Open load ¹ | Only the X bit assigned to the faulted output is on |
| Over temperature | |
| Over load current | |

When these modules are installed, 16 X bits are automatically assigned as the fault status indicator. Each X bit indicates the fault status of each output.

In this example, X10-X27 are assigned as the fault status indicator.

X10: Fault status indicator for Y0

X11: Fault status indicator for Y1



X26: Fault status indicator for Y16

X27: Fault status indicator for Y17

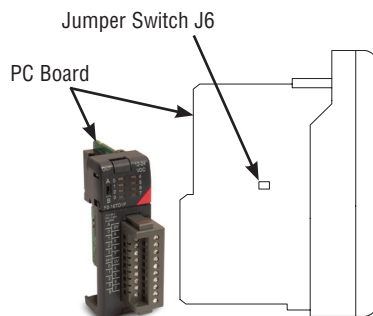
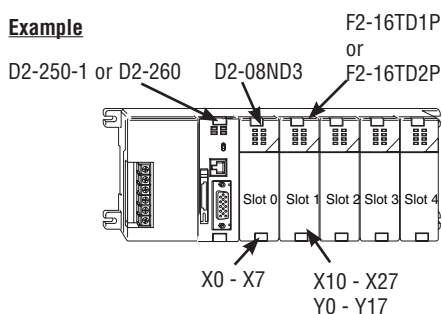
The fault status indicators (X bits) can be reset by performing the indicated operations in the following table:

| Fault Status | Operation |
|------------------------|--|
| Missing external 24VDC | Apply external 24VDC |
| Open load ¹ | Connect the load. |
| Over temperature | Turn the output (Y bit) off or power cycle the PLC |
| Over load current | |



NOTE 1: Open load detection can be disabled by removing the jumper switch J6 on the module PC board.

Example



Continued on next two pages.

F2-16TD1P, DC Output With Fault Protection

2

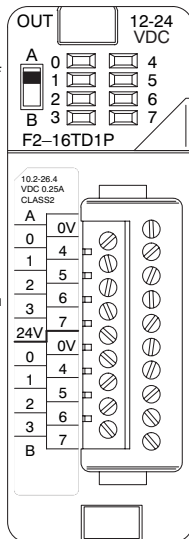
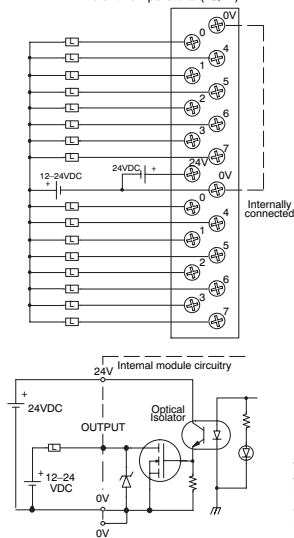
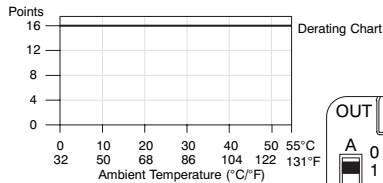


NOTE: Not supported in D2-230, D2-240 and D2-250 CPUs.



NOTE: Supporting Firmware:
D2-250-1 must be V4.80 or later
D2-260 must be V2.60 or later

| F2-16TD1P DC Output with Fault Protection | |
|---|---|
| Inputs per module | 16 (status indication) |
| Outputs per module | 16 (current sinking) |
| Commons per module | 1 (2 I/O terminal points) |
| Output type | NMOS FET (open drain) |
| Operating voltage | 10.2–26.4 VDC, external |
| Peak voltage | 40VDC |
| AC frequency | N/A |
| ON voltage drop | 0.7 V (output current 0.5 A) |
| Overcurrent trip | 0.6A, min., 1.2A, max. |
| Minimum load current | 0.2mA |
| Maximum load current | 0.25A/point; 4A/common |
| Max leakage current | 0.2mA (load detect enabled); 0.3mA disabled |
| Max inrush current | 150 mA for 10ms |
| Base power required 5V | 70 mA |
| OFF to ON response | 0.5 ms |
| ON to OFF response | 0.5 ms |
| Terminal type | Removable (D2-16IOCON) |
| Status indicators | Logic Side |
| Weight | 2.0 oz. (25g) |
| Fuses | None |
| External DC required | 24VDC $\pm 10\%$ @ 50mA |
| External DC overvoltage shutdown | 27V, outputs are restored when voltage is within limits |



When the A/B switch is in the A position, the LEDs display the output status of the module's first 8 output points. Position B displays the output status of the module's second group of 8 output points.

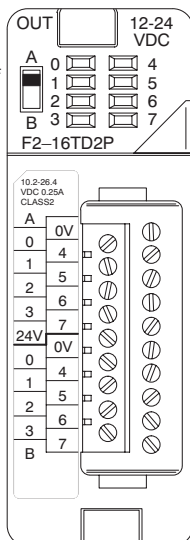
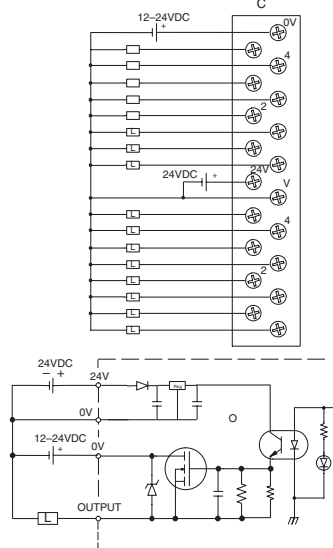
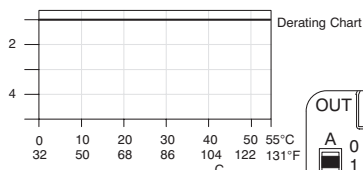
F2-16TD2P, DC Output with Fault Protection



NOTE: Not supported in D2-230, D2-240 and D2-250 CPUs.



NOTE: Supporting Firmware:
D2-250-1 must be V4.80 or later
D2-260 must be V2.60 or later



When the A/B switch is in the A position, the LEDs display the output status of the module's first 8 output points. Position B displays the output status of the module's second group of 8 output points.

| F2-16TD2P DC Output with Fault Protection | |
|---|---|
| Inputs per module | 16 (status indication) |
| Outputs per module | 16 (current sourcing) |
| Commons per module | 1 |
| Output type | NMOS FET (open source) |
| Operating voltage | 10.2–26.4 VDC, external |
| Peak voltage | 40VDC |
| AC frequency | N/A |
| ON voltage drop | 0.7 V (output current 0.5 A) |
| Overcurrent trip | 0.6 A min., 1.2 A max. |
| Minimum load current | 0.2 mA |
| Maximum load current | 0.25 A/point; 4A/common |
| Max leakage current | 0.2 mA (load detect enabled); 0.3 mA disabled |
| Max inrush current | 150mA for 10ms |
| Base power required 5V | 70mA |
| OFF to ON response | 0.5 ms |
| ON to OFF response | 0.5 ms |
| Terminal type | Removable (D2-16IOCON) |
| Status indicators | Logic Side |
| Weight | 2.0 oz. (25g) |
| Fuses | None |
| External DC required | 24VDC $\pm 10\%$ @ 50mA |
| External DC overvoltage shutdown | 27V, outputs are restored when voltage is within limits |

D2-32TD1, DC Output

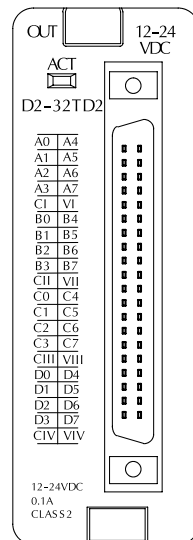
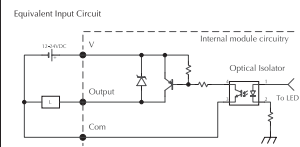
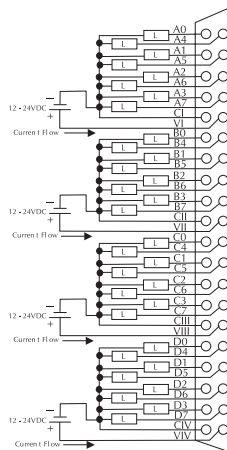
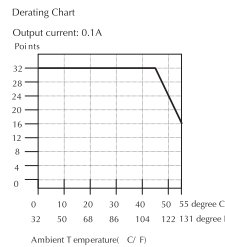
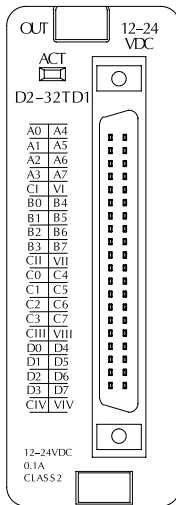
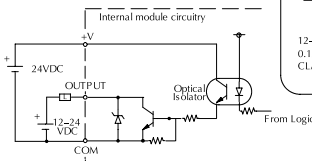
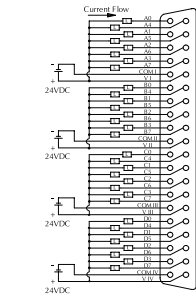
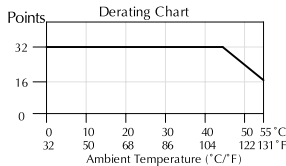
| D2-32TD1 DC Output | |
|-------------------------------------|--|
| Outputs per Module | 32 (current sinking) |
| Commons per Module | 4 (8 I/O terminal points) |
| Output Type | NPN open collector |
| Operating Voltage | 12–24 VDC |
| Peak Voltage | 30VDC |
| ON Voltage Drop | 0.5 VDC maximum |
| Minimum Load Current | 0.2 mA |
| Max Load Current | 0.1 A/point; 3.2 A per module |
| Max Leakage Current | 0.1 mA @ 30VDC |
| Max Inrush Current | 150mA for 10ms |
| Base Power Required 5VDC | 350mA |
| OFF to ON Response | 0.5 ms |
| ON to OFF Response | 0.5 ms |
| Terminal Type (not included) | Removable 40-pin connector ¹ |
| Status Indicator | Module activity (no I/O status indicators) |
| Weight | 2.1 oz. (60g) |
| Fuses | None |
| External DC Power Required | 20–28 VDC max. 120mA (all points on) |

¹ Connector sold separately.
See Terminal Blocks and Wiring for wiring options.

D2-32TD2, DC Output

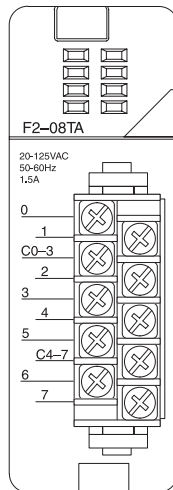
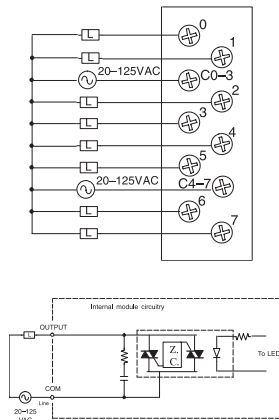
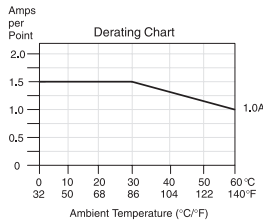
| D2-32TD2 DC Output | |
|-------------------------------------|--|
| Outputs per Module | 32 (current sourcing) |
| Commons per Module | 4 (8 I/O terminal points) |
| Output Type | Transistor |
| Operating Voltage | 12 to 24 VDC |
| Peak Voltage | 30VDC |
| ON Voltage Drop | 0.5 VDC @ 0.1 A |
| Minimum Load Current | 0.2 mA |
| Max Load Current | 0.1 A/point; 0.8 A/common |
| Max Leakage Current | 0.1 mA @ 30VDC |
| Max Inrush Current | 150mA @ 10ms |
| Base Power Required 5VDC | 350mA |
| OFF to ON Response | 0.5 ms |
| ON to OFF Response | 0.5 ms |
| Terminal Type (not included) | Removable 40-pin connector ¹ |
| Status Indicator | Module activity (no I/O status indicators) |
| Weight | 2.1 oz (60g) |
| Fuses | None |

¹ Connector sold separately.
See Terminal Blocks and Wiring for wiring options.



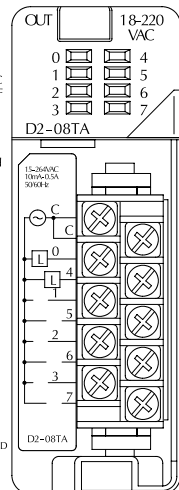
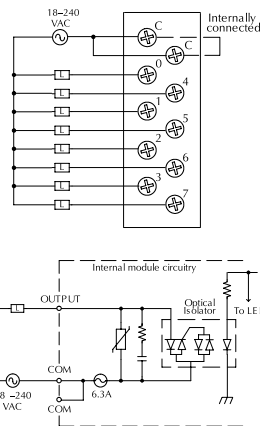
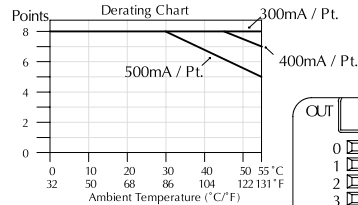
F2-08TA, AC Output

| F2-08TA AC Output | |
|-------------------------------------|---|
| Outputs per Module | 8 |
| Commons per Module | 2 (Isolated) |
| Output Type | SSR (Triac with zero crossover) |
| Operating Voltage | 24–140 VAC |
| Peak Voltage | 140VAC |
| ON Voltage Drop | 1.6 V(rms) @ 1.5 A |
| AC Frequency | 47 to 63 Hz |
| Minimum Load Current | 50 mA |
| Max Load Current | 1.5 A / pt @ 30°C 1.0 A / pt @ 60°C 4.0 A / common; 8.0 A / module @ 60°C |
| Max Leakage Current | 0.7 mA (rms) |
| Peak One Cycle Surge Current | 15A |
| Base Power Required 5VDC | 250mA |
| OFF to ON Response | 0.5 ms - 1/2 cycle |
| ON to OFF Response | 0.5 ms - 1/2 cycle |
| Terminal Type (included) | Removable; D2-8IOCON |
| Status Indicator | Logic side |
| Weight | 3.5 oz. |
| Fuses | None |



D2-08TA, AC Output

| D2-08TA AC Output | |
|---------------------------------|--|
| Outputs per Module | 8 |
| Commons per Module | 1 (2 I/O terminal points) |
| Output Type | SSR (Triac) |
| Operating Voltage | 15–264 VAC |
| Peak Voltage | 264VAC |
| ON Voltage Drop | < 1.5 VAC (>0.1 A) < 3.0 VAC (<0.1 A) |
| AC Frequency | 47 to 63Hz |
| Minimum Load Current | 10mA |
| Max Load Current | 0.5 A/point; 4A/common |
| Max Leakage Current | 4mA (264VAC, 60Hz) 1.2 mA (100VAC, 60Hz) 0.9 mA (100VAC, 50Hz) |
| Max Inrush Current | 10A for 10ms |
| Base Power Required 5VDC | 250mA |
| OFF to ON Response | 1 ms |
| ON to OFF Response | 1 ms + 1/2 cycle |
| Terminal Type (included) | Removable; D2-8IOCON |
| Status Indicator | Logic side |
| Weight | 2.8 oz. (80g) |
| Fuses | 1 per common, 6.3 A slow blow, non-replaceable |

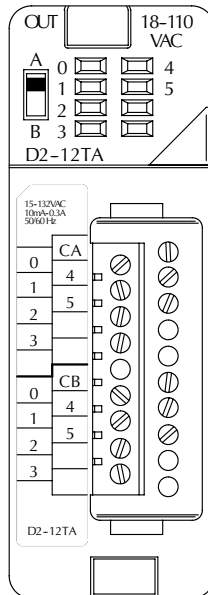
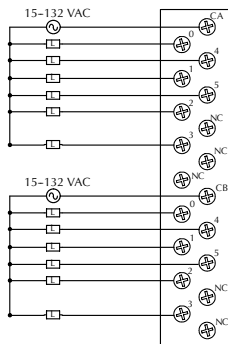
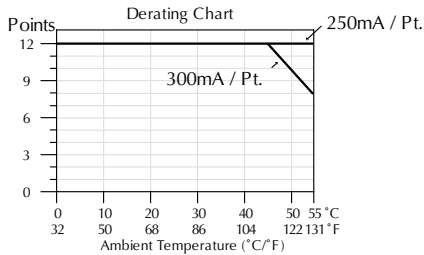


D2-12TA, AC Output

2

| D2-12TA AC Output | |
|--------------------------------|--|
| Outputs per Module | 12 |
| Outputs Points Consumed | 16 (four unused, see chart below) |
| Commons per Module | 2 (isolated) |
| Output Type | SSR (Triac) |
| Operating Voltage | 15–132 VAC |
| Peak Voltage | 132VAC |
| ON Voltage Drop | < 1.5 VAC (>50mA) < 4.0 VAC (<50mA) |
| AC Frequency | 47 to 63 Hz |
| Minimum Load Current | 10mA |
| Max Load Current | 0.3 A/point; 1.8 A/common |

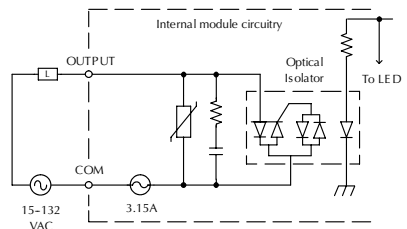
| | |
|---------------------------------|---|
| Max Leakage Current | 2mA (132VAC, 60Hz) |
| Max Inrush Current | 10A for 10ms |
| Base Power Required 5VDC | 350mA |
| OFF to ON Response | 1ms |
| ON to OFF Response | 1ms + 1/2 cycle |
| Terminal Type (included) | Removable; D2-16IOCON |
| Status Indicator | Logic side |
| Weight | 2.8 oz. (80g) |
| Fuses | (2) 1 per common 3.15 A slow blow, replaceable Order D2-FUSE-1 (5 per pack) |



Addresses Used

| Points | Used? | Points | Used? |
|--------|-------|--------|-------|
| Yn+0 | Yes | Yn+10 | Yes |
| Yn+1 | Yes | Yn+11 | Yes |
| Yn+2 | Yes | Yn+12 | Yes |
| Yn+3 | Yes | Yn+13 | Yes |
| Yn+4 | Yes | Yn+14 | Yes |
| Yn+5 | Yes | Yn+15 | Yes |
| Yn+6 | No | Yn+16 | No |
| Yn+7 | No | Yn+17 | No |

n is the starting address



D2-04TRS, Relay Output

2

| D2-04TRS Relay Output | |
|-------------------------------------|---------------------------------|
| Outputs per Module | 4 |
| Outputs Points Consumed | 8 (only 1st 4pts. are used) |
| Commons per Module | 4 (isolated) |
| Output Type | Relay, form A (SPST) |
| Operating Voltage | 5-30 VDC / 5-240 VAC |
| Peak Voltage | 30 VDC, 264 VAC |
| ON Voltage Drop | 0.72 VDC maximum |
| AC Frequency | 47 to 63 Hz |
| Minimum Load Current | 10mA |
| Max Load Current (resistive) | 4A/point; 8A/module (resistive) |

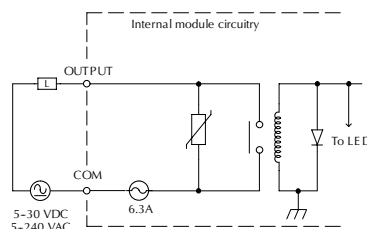
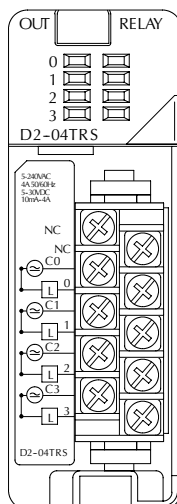
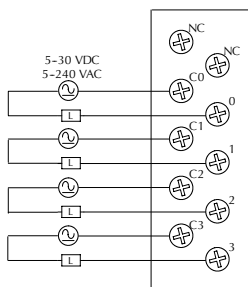
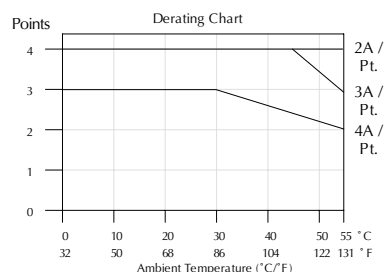
| | |
|---------------------------------|---|
| Max Leakage Current | 0.1 mA @ 264VAC |
| Max Inrush Current | 5A for < 10ms |
| Base Power Required 5VDC | 250mA |
| OFF to ON Response | 10ms |
| ON to OFF Response | 10ms |
| Terminal Type (included) | Removable; D2-810CON |
| Status Indicator | Logic side |
| Weight | 2.8 oz. (80g) |
| Fuses | 1 per point 6.3 A slow blow, replaceable Order D2-FUSE-3 (5 per pack) |

| Typical Relay Life (Operations) | | | | |
|---------------------------------|------|------|------|------|
| Voltage & Load Current | | | | |
| Type of Load | 1A | 2A | 3A | 4A |
| 24VDC Resistive | 500k | 200k | 100k | 50k |
| 24VDC Solenoid | 100k | 40k | — | — |
| 110 VAC Resistive | 500k | 250k | 150k | 100k |
| 110 VAC Solenoid | 200k | 100k | 50k | — |
| 220 VAC Resistive | 350k | 150k | 100k | 50k |
| 220 VAC Solenoid | 100k | 50k | — | — |

At 24 VDC, solenoid (inductive) loads over 2A cannot be used.

At 100 VAC, solenoid (inductive) loads over 3A cannot be used.

At 220 VAC, solenoid (inductive) loads over 2A cannot be used.



D2-08TR, Relay Output

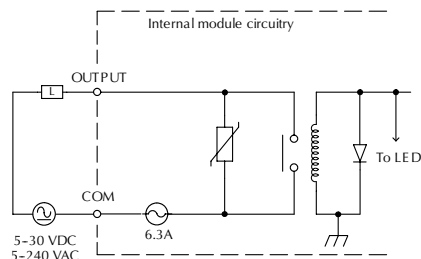
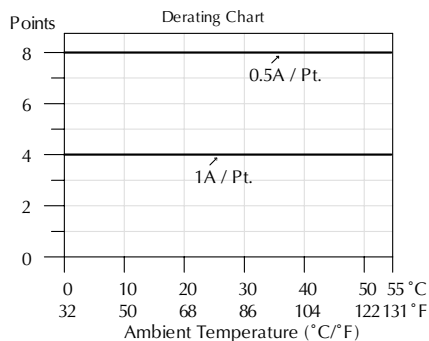
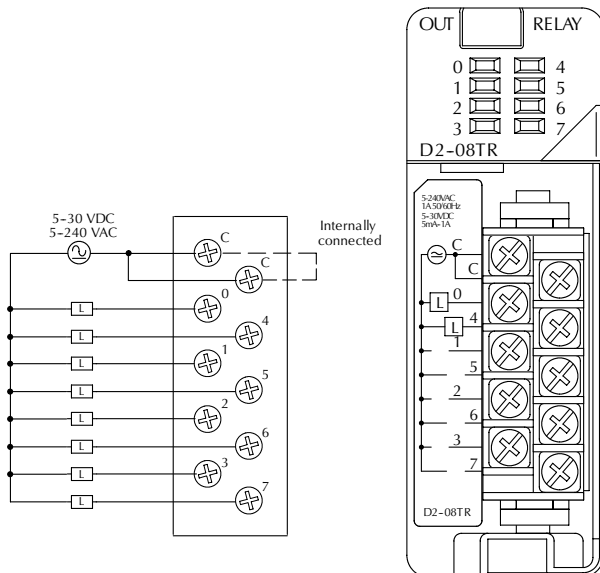
D2-08TR Relay Output

| | |
|-------------------------------------|----------------------|
| Outputs per Module | 8 |
| Outputs Points Consumed | 8 |
| Commons per Module | 1 (2 I/O terminals) |
| Output Type | Relay, form A (SPST) |
| Operating Voltage | 5–30 VDC; 5–240 VAC |
| Peak Voltage | 30VDC, 264VAC |
| ON Voltage Drop | N/A |
| AC Frequency | 47 to 60 Hz |
| Minimum Load Current | 5mA @ 5VDC |
| Max Load Current (resistive) | 1A/point; 4A/common |

| | |
|---------------------------------|---|
| Max Leakage Current | 0.1 mA @265 VAC |
| Max Inrush Current | Output: 3A for 10ms Common: 10A for 10ms |
| Base Power Required 5VDC | 250mA |
| OFF to ON Response | 12ms |
| ON to OFF Response | 10ms |
| Terminal Type (included) | Removable; D2-8IOCON |
| Status Indicator | Logic side |
| Weight | 3.9 oz. (110g) |
| Fuses | One 6.3A slow blow, replaceable Order D2-FUSE-3 (5 per pack) |

Typical Relay Life (Operations)

| Voltage/Load | Current | Closures |
|------------------|---------|----------|
| 24VDC Resistive | 1A | 500k |
| 24VDC Solenoid | 1A | 100k |
| 110VDC Resistive | 1A | 500k |
| 110VDC Solenoid | 1A | 200k |
| 220VAC Resistive | 1A | 350k |
| 220VAC Solenoid | 1A | 100k |



F2-08TR, Relay Output

| F2-08TR Relay Output | |
|------------------------------|---|
| Outputs per Module | 8 |
| Outputs Points Consumed | 8 |
| Commons per Module | 2 (isolated), 4-pts. per common |
| Output Type | 8, Form A (SPST normally open) |
| Operating Voltage | 7A @ 12–28 VDC, 12–250VAC; 0.5 A @ 120VDC |
| Peak Voltage | 150VDC, 265VAC |
| ON Voltage Drop | N/A |
| AC Frequency | 47 to 63 Hz |
| Minimum Load Current | 10 mA @ 12 VDC |
| Max Load Current (resistive) | 10A/point ³ (subject to derating) Max of 10A/common |
| Max Leakage Current | N/A |
| Max Inrush Current | 12A |
| Base Power Required 5VDC | 670mA |
| OFF to ON Response | 15ms (typical) |
| ON to OFF Response | 5ms (typical) |
| Terminal Type (included) | Removable; D2-8IOCON |
| Status Indicator | Logic side |
| Weight | 5.5 oz. (156g) |
| Fuses | None |

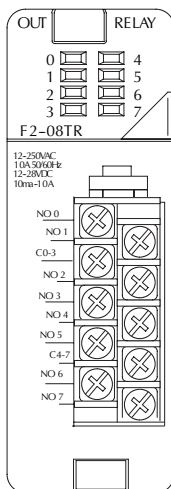
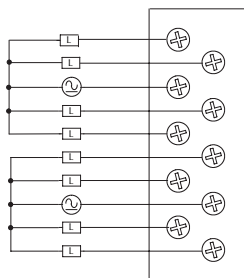
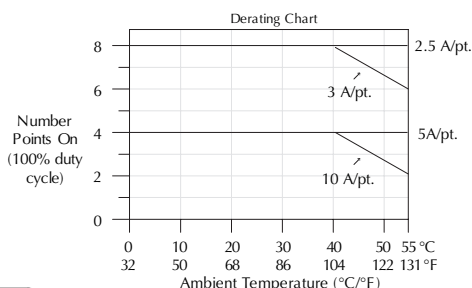
Typical Relay Life¹ (Operations) at Room Temperature

| Voltage & Type of Load ² | 50mA | 5A | 7A |
|-------------------------------------|------|------|------|
| 24 VDC Resistive | 10M | 600k | 300k |
| 24 VDC Solenoid | — | 150k | 75k |
| 110 VDC Resistive | — | 600k | 300k |
| 110 VDC Solenoid | — | 500k | 200k |
| 220 VAC Resistive | — | 300k | 150k |
| 220 VAC Solenoid | — | 250k | 100k |

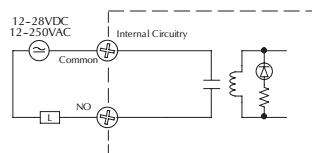
1) Contact life may be extended beyond those values shown with the use of arc suppression techniques described in the DL205 User Manual. Since these modules have no leakage current, they do not have built-in snubber. For example, if you place a diode across a 24VDC inductive load, you can significantly increase the life of the relay.

2) At 120VDC 0.5 A resistive load, contact life cycle is 200k cycles.

3) Normally closed contacts have 1/2 the current handling capability of the normally open contacts.



Typical Circuit



F2-08TRS, Relay Output

F2-08TRS Relay Output

| | |
|-------------------------------------|---|
| Outputs per Module | 8 |
| Outputs Points Consumed | 8 |
| Commons per Module | 8 (isolated) |
| Output Type | 3, Form C (SPDT) |
| Operating Voltage | 5, Form A (SPST normally open) 7A @ 12–28 VDC, 12–250 VAC 0.5A @ 120VDC |
| Peak Voltage | 150VDC, 265VAC |
| ON Voltage Drop | N/A |
| AC Frequency | 47 to 63Hz |
| Minimum Load Current | 10mA @ 12VDC |
| Max Load Current (resistive) | 7A/point ³ (subject to derating) |
| Max Leakage Current | N/A |
| Max Inrush Current | 12A |
| Base Power Required 5VDC | 670mA |
| OFF to ON Response | 15ms (typical) |
| ON to OFF Response | 5ms (typical) |
| Terminal Type (included) | Removable; D2-16IOCON |
| Status Indicator | Logic side |
| Weight | 5.5 oz. (156g) |
| Fuses | None |

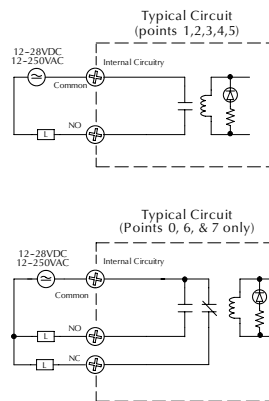
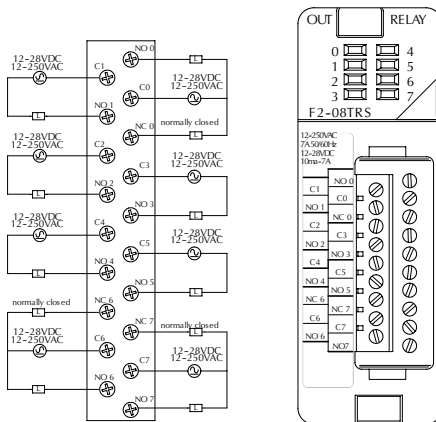
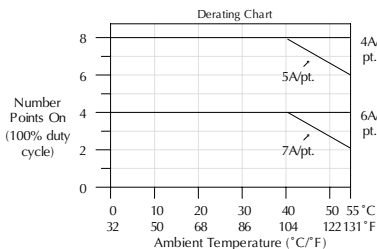
Typical Relay Life¹ (Operations) at Room Temperature

| Voltage & Type of Load ² | Load Current | | |
|--|--------------|------|------|
| | 50mA | 5A | 7A |
| 24VDC Resistive | 10M | 600k | 300k |
| 24VDC Solenoid | – | 150k | 75k |
| 110VDC Resistive | – | 600k | 300k |
| 110VDC Solenoid | – | 500k | 200k |
| 220VAC Resistive | – | 300k | 150k |
| 220VAC Solenoid | – | 250k | 100k |

1) Contact life may be extended beyond those values shown with the use of arc suppression techniques described in the DL205 User Manual. Since these modules have no leakage current, they do not have built-in snubber. For example, if you place a diode across a 24VDC inductive load, you can significantly increase the life of the relay.

2) At 120VDC 0.5 A resistive load, contact life cycle is 200k cycles.

3) Normally, closed contacts have 1/2 the current handling capability of the normally open contacts.



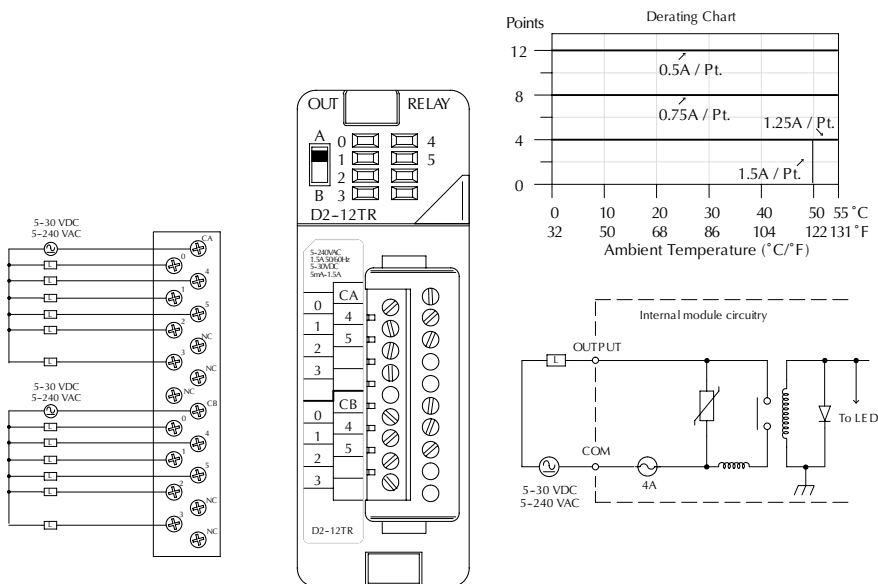
D2-12TR, Relay Output

| D2-12TR Relay Output | |
|-------------------------------------|---|
| Outputs per Module | 12 |
| Outputs Points Consumed | 16 (four unused, see chart below) |
| Commons per Module | 2 (6-pts. per common) |
| Output Type | Relay, form A (SPST) |
| Operating Voltage | 5–30 VDC; 5–240 VAC |
| Peak Voltage | 30VDC; 264VAC |
| ON Voltage Drop | N/A |
| AC Frequency | 47 to 60 Hz |
| Minimum Load Current | 5mA @ 5VDC |
| Max Load Current (resistive) | 1.5 A/point; Max of 3A/common |
| Max Leakage Current | 0.1 mA @ 265VAC |
| Max Inrush Current | Output: 3A for 10ms Common: 10A for 10ms |
| Base Power Required 5VDC | 450mA |
| OFF to ON Response | 10ms |
| ON to OFF Response | 10ms |
| Terminal Type (included) | Removable; D2-16IOCON |
| Status Indicator | Logic side |
| Weight | 4.6 oz. (130g) |
| Fuses | (2) 4A slow blow, replaceable Order D2-FUSE-4 (5 per pack) |

| Typical Relay Life (Operations) | | |
|---------------------------------|---------|----------|
| Voltage/Load | Current | Closures |
| 24VDC Resistive | 1A | 500k |
| 24VDC Solenoid | 1A | 100k |
| 110VDC Resistive | 1A | 500k |
| 110VDC Solenoid | 1A | 200k |
| 220VAC Resistive | 1A | 350k |
| 220VAC Solenoid | 1A | 100k |

| Addresses Used | | | |
|----------------|-------|--------|-------|
| Points | Used? | Points | Used? |
| Yn+0 | Yes | Yn+10 | Yes |
| Yn+1 | Yes | Yn+11 | Yes |
| Yn+2 | Yes | Yn+12 | Yes |
| Yn+3 | Yes | Yn+13 | Yes |
| Yn+4 | Yes | Yn+14 | Yes |
| Yn+5 | Yes | Yn+15 | Yes |
| Yn+6 | No | Yn+16 | No |
| Yn+7 | No | Yn+17 | No |

n is the starting address



D2-08CDR, 4 pt. DC Input / 4 pt. Relay Output

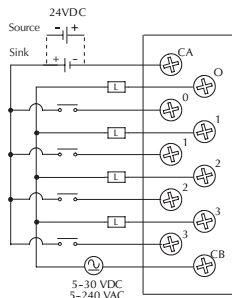
D2-08CDR 4-pt. DC In / 4pt. Relay Out

General Specifications

| | |
|---------------------------------|--------------------------------|
| Base Power Required 5VDC | 200mA |
| Terminal Type (included) | Removable; D2-8IOCON |
| Status Indicator | Logic side |
| Weight | 3.5 oz. (100 g) |
| Input Specifications | |
| Inputs per Module | 4 (sink/source) |
| Input Points Consumed | 8 (only first 4-pts. are used) |
| Commons per Module | 1 |
| Input Voltage Range | 20–28 VDC |
| Peak Voltage | 30VDC |
| ON Voltage Level | 19VDC minimum |
| OFF Voltage Level | 7VDC maximum |
| AC Frequency | N/A |
| Input Impedance | 4.7 kΩ |
| Input Current | 5mA @ 24VDC |
| Maximum Current | 8mA @ 30VDC |
| Minimum ON Current | 4.5 mA |
| Maximum OFF Current | 1.5 mA |
| OFF to ON Response | 1 to 10 ms |
| ON to OFF Response | 1 to 10 ms |
| Fuses (input circuits) | None |

Typical Relay Life (Operations)

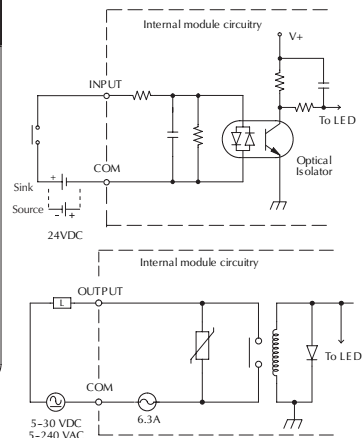
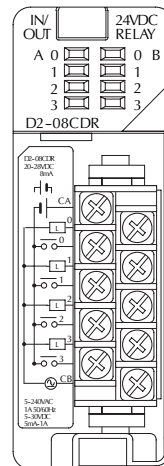
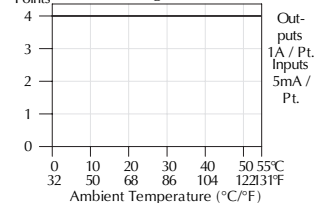
| Voltage/Load | Current | Closures |
|------------------|---------|----------|
| 24VDC Resistive | 1A | 500k |
| 24VDC Solenoid | 1A | 100k |
| 110VAC Resistive | 1A | 500k |
| 110VAC Solenoid | 1A | 200k |
| 220VAC Resistive | 1A | 350k |
| 220VAC Solenoid | 1A | 100k |



Output Specifications

| | |
|-------------------------------------|---|
| Outputs per Module | 4 |
| Outputs Points Consumed | 8 (only first 4-pts. are used) |
| Commons per Module | 1 |
| Output Type | Relay, form A (SPST) |
| Operating Voltage | 5–30 VDC; 5–240 VAC |
| Peak Voltage | 30VDC; 264VAC |
| ON Voltage Drop | N/A |
| AC Frequency | 47 to 63 Hz |
| Minimum Load Current | 5mA @ 5VDC |
| Max Load Current (resistive) | 1A/point; 4A/module |
| Max Leakage Current | 0.1 mA @ 264VAC |
| Max Inrush Current | 3A for < 100ms 10A for < 10ms (common) |
| OFF to ON Response | 12ms |
| ON to OFF Response | 10ms |
| Fuses (output circuits) | 1 (6.3 A slow blow, replaceable); Order D2-FUSE-3 (5 per pack) |

Derating Chart



Glossary of Specification Terms

Inputs or Outputs Per Module

Indicates number of input or output points per module and designates current sinking, current sourcing, or either.

Commons Per Module

Number of commons per module and their electrical characteristics.

Input Voltage Range

The operating voltage range of the input circuit.

Output Voltage Range

The operating voltage range of the output circuit.

Peak Voltage

Maximum voltage allowed for the input circuit.

AC Frequency

AC modules are designed to operate within a specific frequency range.

ON Voltage Level

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

OFF Voltage Level

The 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.

Minimum Load

The minimum load current for the output circuit to operate properly.

External DC Required

Some output modules require external power for the output circuitry.

ON Voltage Drop

Sometimes called “saturation voltage,” it is the voltage measured from an output point to its common terminal when the output is ON at maximum load.

Maximum Leakage Current

The maximum current a connected maximum load will receive when the output point is OFF.

Maximum Inrush Current

The maximum current used by a load for a short duration upon an OFF to ON transition of an output point. It is greater than the normal ON state current and is characteristic of inductive loads in AC circuits.

Base Power Required

Power from the base power supply is used by the DL205 input modules and varies between different modules. The guidelines for using module power are explained in the power budget configuration section in Chapter 4–7.

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 terminal.

Status Indicators

The LEDs that indicate the ON/OFF status of an input point. These LEDs are electrically located on either the logic side or the field device side of the input circuit.

Weight

Indicates the weight of the module. See Appendix F for a list of the weights for the various DL205 components.

Fuses

Protective devices for an output circuit, which stop current flow when current exceeds the fuse rating. They may be replaceable or non-replaceable, or located externally or internally.

CPU SPECIFICATIONS AND OPERATIONS



CHAPTER 3

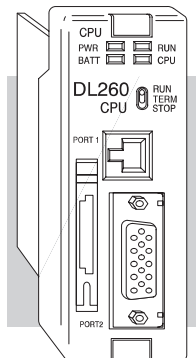
In This Chapter

| | |
|--|------|
| CPU Overview..... | 3-2 |
| CPU General Specifications | 3-4 |
| CPU Base Electrical Specifications..... | 3-5 |
| CPU Hardware Setup | 3-6 |
| Selecting the Program Storage Media..... | 3-9 |
| Using Battery Backup | 3-14 |
| CPU Operation..... | 3-21 |
| I/O Response Time..... | 3-27 |
| CPU Scan Time Considerations | 3-29 |
| PLC Numbering Systems..... | 3-35 |
| Memory Map..... | 3-37 |
| DL230 System V-memory | 3-41 |
| DL240 System V-memory | 3-43 |
| DL250-1 System V-memory (DL250 also) | 3-46 |
| DL260 System V-memory | 3-49 |
| DL205 Aliases..... | 3-52 |
| DL230 Memory Map | 3-53 |
| DL240 Memory Map | 3-54 |
| DL250-1 Memory Map (DL250 also)..... | 3-55 |
| DL260 Memory Map | 3-56 |
| X Input/Y Output Bit Map..... | 3-57 |
| Control Relay Bit Map..... | 3-59 |
| Stage Control/Status Bit Map..... | 3-63 |
| Timer and Counter Status Bit Maps | 3-65 |
| Remote I/O Bit Map..... | 3-66 |

CPU Overview

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

- The differences between the various models of CPUs, and
- The steps required to set up and install the CPU.



General CPU Features

The DL230, DL240, DL250–1 and D2–260 are modular CPUs which can be installed in 3, 4, 6, or 9 slot bases. All I/O modules in the DL205 family will work with any of the CPUs. The DL205 CPUs offer a wide range of processing power and program instructions. All offer RLL and Stage program instructions (See Chapter 5). They also provide extensive internal diagnostics that can be monitored from the application program or from an operator interface.

DL230 CPU Features

The DL230 has 2.4K words of memory comprised of 2.0K of ladder memory and approximately 400 words of V-memory (data registers). It has 92 different instructions available for programming, and supports a maximum of 256 I/O points.

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

The DL230 provides one built-in RS-232 communication port, so you can easily connect a handheld programmer or a personal computer without needing any additional hardware.

DL240 CPU Features

The DL240 has a maximum of 3.8K of memory comprised of 2.5K of ladder memory and approximately 1.3K of V-memory (data registers). There are 129 instructions available for program development and a maximum of 256 points local I/O, and 896 points with remote I/O are supported.

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

The DL240 has two communication ports. The top port is the same port configuration as the DL230. The bottom port also supports the *DirectNET* protocol, so you can use the DL240 in a *DirectNET* network. Since the port is RS-232, you must use an RS-232/RS-422 converter for multi-drop connections.

DL250–1 CPU Features

The DL250–1 replaces the DL250 CPU. It offers all the DL240 features, plus more program instructions and a built-in Remote I/O Master port. It offers all the features of the DL250 CPU with the addition of supporting Local expansion I/O. It has a maximum of 14.8K of program memory comprised of 7.6K of ladder memory and 7.2K of V-memory (data registers). It supports a maximum of 256 points of local I/O and a maximum of 768 I/O points (maximum of two local expansion bases). In addition, port 2 supports up to 2048 points if you use the DL250–1 as a Remote master. It includes an internal RISC-based microprocessor for greater processing power. The DL250–1 has 240 instructions. The instructions are in addition to the DL240 instruction set which includes drum timers, a print function, floating point math, PID loop control for 4 loops and the Intelligent Box (IBox) instructions.

The DL250–1 has a total of two built-in communications ports. The top port is identical to the top port of the DL240, with the exception of the *DirectNet* slave feature. The bottom port is a 15-pin RS-232/RS-422 port. It will interface with *DirectSOFT* and operator interfaces, and provides *DirectNet* and Modbus RTU Master/Slave connections.

DL260 CPU Features

The DL260 offers all the DL250–1 features, plus ASCII IN/OUT and expanded Modbus instructions. It also supports up to 1280 local I/O points by using up to four local expansion bases. It has a maximum of 30.4K of program memory comprised of 15.8K of ladder memory (saved on flash memory) and 14.6K of V-memory (data registers). It also includes an internal RISC-based microprocessor for greater processing power. The DL260 has 297 instructions. In addition to those in the DL250–1 instruction set, the DL260 instruction set includes table instructions, trigonometric instructions and support for 16 PID loops.

The DL260 has a total of two built-in communications ports. The top port is identical to the top port of the DL250–1. The bottom port is a 15-pin RS-232/RS-422/RS-485 port. It will interface with *DirectSOFT* (version 4.0 or later), operator interfaces, and provides *DirectNet*, Modbus RTU Master/Slave connections. Port 2 also supports ASCII IN/OUT instructions.

CPU General Specifications

| Feature | DL230 | DL240 | DL250-1 | DL260 |
|---|-----------------|--------------------------------|--|--|
| Total Program memory (words) | 2.4K | 3.8K | 14.8K | 30.4K |
| Ladder memory (words) | 2048 | 2560 | 7680 (Flash) | 15872 (Flash) |
| V-memory (words) | 256 | 1024 | 7168 | 14592 |
| Non-volatile V Memory (words) | 128 | 256 | No | No |
| Boolean execution /K | 4–6 ms | 10–12 ms | 1.9 ms | 1.9 ms |
| RLL and RLL ^{PLUS} Programming | Yes | Yes | Yes | Yes |
| Handheld programmer | Yes | Yes | Yes | Yes |
| <i>Direct</i> SOFT programming for Windows. | Yes | Yes | Yes | Yes (requires version 4.0 or higher) |
| Built-in communication ports | One RS-232 | Two RS-232 | One RS-232 One RS-232 or RS-422 | One RS-232 One RS-232, RS-422 or RS-485 |
| EEPROM | Standard on CPU | Standard on CPU | Flash | Flash |
| Total CPU memory I/O points available | 256 (X,Y,CR) | 896 (X, Y, CR) | 2048 (X, Y, CR) | 8192 (X, Y, CR, GX, GY) |
| Local I/O points available | 256 | 256 | 256 | 256 |
| Local Expansion I/O points (including local I/O and expansion I/O points) | N/A | N/A | 768 (2 exp. bases max.) | 1280 (4 exp. bases max.) |
| Serial Remote I/O points (including local I/O and expansion I/O points) | N/A | 896 | 2048 | 8192 |
| Serial Remote I/O Channels | N/A | 2 | 8 | 8 |
| Max Number of Serial Remote Slaves | N/A | 7 Remote / 31 Slice | 7 Remote / 31 Slice | 7 Remote / 31 Slice |
| Ethernet Remote I/O Discrete points | N/A | 896 | 2048 | 8192 |
| Ethernet Remote I/O Analog I/O channels | N/A | Map into V-memory | Map into V-memory | Map into V-memory |
| Ethernet Remote I/O channels | N/A | Limited by power budget | Limited by power budget | Limited by power budget |
| Max Number of Ethernet slaves per channel | N/A | 16 | 16 | 16 |
| I/O points per Remote channel | N/A | 16,384 (limited to 896 by CPU) | 16,384 (16 fully expanded H4-EBC slaves using V-memory and bit-of-word instructions) | 16,384 (16 fully expanded H4-EBC slaves using V-memory and bit-of-word instructions) |
| I/O Module Point Density | 4/8/12/16/32 | 4/8/12/16/32 | 4/8/12/16/32 | 4/8/12/16/32 |
| Slots per Base | 3/4/6/9 | 3/4/6/9 | 3/4/6/9 | 3/4/6/9 |

| Feature | DL230 | DL240 | DL250-1 | DL260 |
|--|----------------|----------------|-------------------------|--|
| Number of instructions available (see Chapter 5 for details) | 92 | 129 | 240 | 297 |
| Control relays | 256 | 256 | 1024 | 2048 |
| Special relays (system defined) | 112 | 144 | 144 | 144 |
| Stages in RLL ^{PLUS} | 256 | 512 | 1024 | 1024 |
| Timers | 64 | 128 | 256 | 256 |
| Counters | 64 | 128 | 128 | 256 |
| Immediate I/O | Yes | Yes | Yes | Yes |
| Interrupt input (hardware / timed) | Yes / No | Yes / Yes | Yes / Yes | Yes / Yes |
| Subroutines | No | Yes | Yes | Yes |
| Drum Timers | No | No | Yes | Yes |
| Table Instructions | No | No | No | Yes |
| For/Next Loops | No | Yes | Yes | Yes |
| Math | Integer | Integer | Integer, Floating Point | Integer, Floating Point, Trigonometric |
| ASCII | No | No | Yes, OUT | Yes, IN/OUT |
| PID Loop Control, Built In | No | No | Yes, 4 Loops | Yes, 16 Loops |
| Time of Day Clock/Calendar | No | Yes | Yes | Yes |
| Run Time Edits | Yes | Yes | Yes | Yes |
| Supports Overrides | No | Yes | Yes | Yes |
| Internal diagnostics | Yes | Yes | Yes | Yes |
| Password security | Yes | Yes | Yes | Yes |
| System error log | No | Yes | Yes | Yes |
| User error log | No | Yes | Yes | Yes |
| Battery backup | Yes (optional) | Yes (optional) | Yes (optional) | Yes (optional) |

CPU Base Electrical Specifications

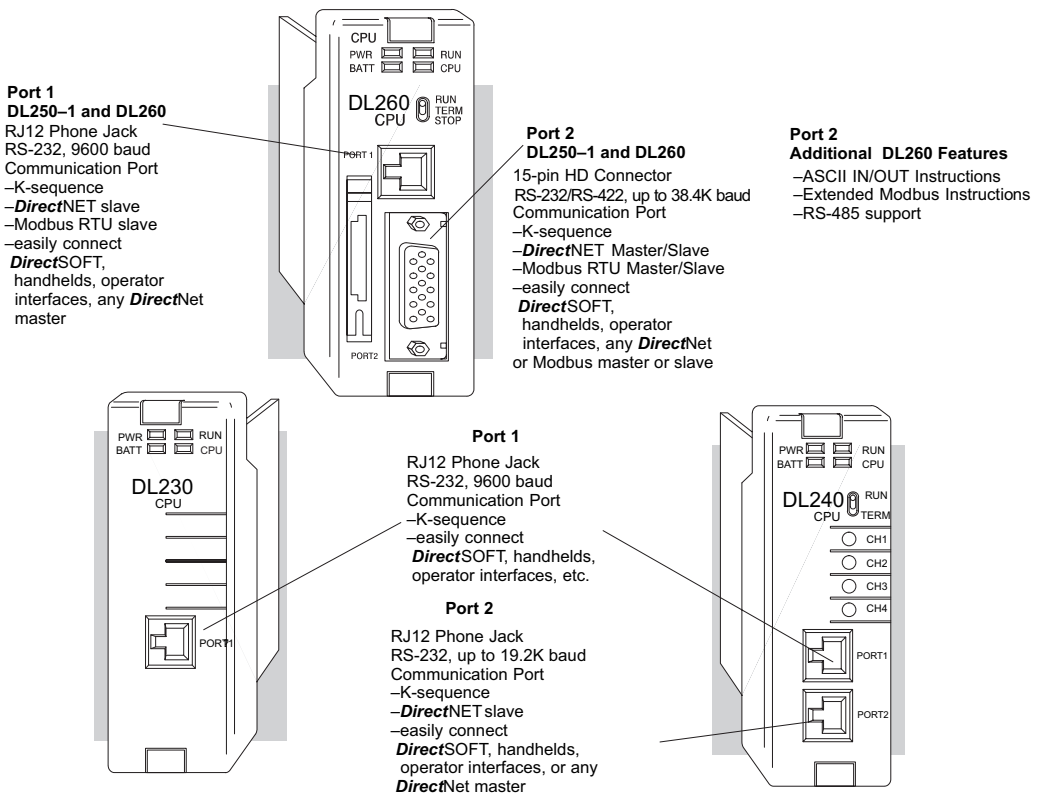
| Specification | AC Powered Bases | 24 VDC Powered Bases | 125 VDC Powered Bases |
|--|--|--|--|
| Part Numbers | D2-03B-1 D2-04B-1 D2-06B-1 D2-09B-1 | D2-03BDC1-1 D2-04BDC1-1 D2-06BDC1-1 D2-09BDC1-1 | D2-06BDC2-1 D2-09BDC2-1 |
| Input Voltage Range | 100-240 VAC +10% -15% | 10.2-28.8 VDC (24VDC) with less than 10% ripple | 104-240 VDC +10% -15% |
| Maximum Inrush Current | 30A | 10A | 20A |
| Maximum Power | 80VA | 25W | 30W |
| Voltage Withstand (dielectric) | 1 minute @ 1500VAC between primary, secondary, field ground, and run relay | | |
| Insulation Resistance | > 10MΩ at 500VDC | | |
| Auxiliary 24 VDC Output | 20-28 VDC, less than 1V p-p 300mA max. | None | 20-28 VDC, less than 1V p-p 300mA max. |
| Fusing (internal to base power supply) | Non-replaceable 2A @ 250V slow blow fuse | Non-replaceable 3.15 A @ 250V slow blow fuse | Non-replaceable 2A @ 250V slow blow fuse |

CPU Hardware Setup

Communication Port Pinout Diagrams

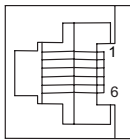
Cables are available that allow you to quickly and easily connect a Handheld Programmer or a personal computer to the DL205 CPUs. However, if you need to build a cable(s), use the pinout descriptions shown on the following pages. You can also use the Tech Support/Cable Wiring diagrams located on our website.

The DL240, DL250-1 and DL260 CPUs have two ports while the DL230 has only one. All of the CPUs require at least one RJ-12 connector. The DL250-1 and DL260 require one 15 pin D-shell connector.



Port 1 Specifications

- ☒ **230** The operating parameters for Port 1 on the DL230 and DL240 CPUs are fixed.
☒ **240**
 - 6-pin female modular (RJ12 phone jack) type connector☒ **250-1**
 - K-sequence protocol (slave only)☒ **260**
 - RS-232, 9600 baud
 - Connect to *DirectSOFT*, D2-HPP, DV-1000, HMI panels
 - Fixed station address of 1
 - 8 data bits, one stop
 - Asynchronous, Half-duplex, DTE
 - Odd parity

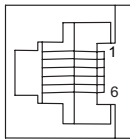


6-pin Female
Modular Connector

| Port 1 Pin Descriptions (DL230 and DL240) | | |
|---|-----|----------------------------|
| 1 | 0V | Power (–) connection (GND) |
| 2 | 5V | Power (+) connection |
| 3 | RXD | Receive Data (RS-232) |
| 4 | TXD | Transmit Data (RS-232) |
| 5 | 5V | Power (+) connection |
| 6 | 0V | Power (–) connection (GND) |

Port 1 Specifications

- ☒ **230** The operating parameters for Port 1 on the DL250–1 and DL260 CPU are fixed. This applies to the DL250 as well.
☒ **240**
☒ **250-1**
 - 6-pin female modular (RJ12 phone jack) type connector☒ **260**
 - K-sequence protocol (slave only)
 - *DirectNET* (slave only)
 - Modbus RTU (slave only) - supported only on D2-250-1 and D2-260
 - RS-232, 9600 baud
 - Connect to *DirectSOFT*, D2-HPP, DV1000 or *DirectNET* master
 - 8 data bits, one start, one stop
 - Asynchronous, Half-duplex, DTE
 - Odd parity



6-pin Female
Modular Connector

| Port 1 Pin Descriptions (DL250-1 and DL260) | | |
|---|-----|----------------------------|
| 1 | 0V | Power (–) connection (GND) |
| 2 | 5V | Power (+) connection |
| 3 | RXD | Receive Data (RS-232C) |
| 4 | TXD | Transmit Data (RS-232C) |
| 5 | 5V | Power (+) connection |
| 6 | 0V | Power (–) connection (GND) |



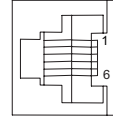
NOTE: The 5V pins are rated at 200mA maximum, primarily for use with some operator interface units.

Port 2 Specifications

- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260

The operating parameters for Port 2 on the DL240 CPU are configurable using Aux functions on a programming device.

- 6-Pin female modular (RJ12 phone jack) type connector
- K-sequence protocol, *DirectNET* (slave),
- RS-232, Up to 19.2K baud
- Address selectable (1–90)
- Connect to *DirectSOFT*, D2–HPP, DV-1000, HMI, or *DirectNET* master
- 8 data bits, one start, one stop
- Asynchronous, Half-duplex, DTE
- Odd or no parity



6-pin Female
Modular Connector

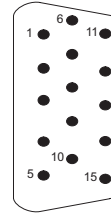
| Port 2 Pin Descriptions (DL240 only) | | |
|--------------------------------------|-----|----------------------------|
| 1 | 0V | Power (–) connection (GND) |
| 2 | 5V | Power (+) connection |
| 3 | RXD | Receive Data (RS-232) |
| 4 | TXD | Transmit Data (RS-232) |
| 5 | RTS | Request to Send |
| 6 | 0V | Power (–) connection (GND) |

Port 2 Specifications

- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260

Port 2 on the DL250-1 and DL260 CPUs is located on the 15-pin D-shell connector. It is configurable using AUX functions on a programming device. This applies to the DL250 as well.

- 15-Pin female D type connector
- Protocol: K-sequence, *DirectNET* Master/Slave, Modbus RTU Master/Slave, Remote I/O, (ASCII IN/OUT DL260 only)
- RS-232, non-isolated, distance within 15m (approximately 50ft)
- RS-422, non-isolated, distance within 1000 m (approximately 3280ft)
- RS-485, non-isolated, distance within 1000m (DL260 only)
- Up to 38.4 K baud
- Address selectable (1–90)
- Connects to *DirectSOFT*, D2–HPP, operator interfaces, any *DirectNET* or Modbus master/slave, (ASCII devices-DL260 only)
- 8 data bits, one start, one stop
- Asynchronous, Half-duplex, DTE Remote I/O
- Odd/even/none parity



15-pin Female
D Connector

| Port 2 Pin Descriptions (DL250–1 / DL260) | | |
|---|--------|---|
| 1 | 5V | 5VDC |
| 2 | TXD2 | Transmit Data (RS-232) |
| 3 | RXD2 | Receive Data (RS-232) |
| 4 | RTS2 | Ready to Send (RS-232) |
| 5 | CTS2 | Clear to Send (RS-232) |
| 6 | RXD2 – | Receive Data – (RS-422) (RS-485 DL260) |
| 7 | 0V | Logic Ground |
| 8 | 0V | Logic Ground |
| 9 | TXD2 + | Transmit Data + (RS-422) (RS-485 DL260) |
| 10 | TXD2 – | Transmit Data – (RS-422) (RS-485 DL260) |
| 11 | RTS2 + | Request to Send + (RS-422) (RS-485 DL260) |
| 12 | RTS2 – | Request to Send – (RS-422)(RS-485 DL260) |
| 13 | RXD2 + | Receive Data + (RS-422) (RS-485 DL260) |
| 14 | CTS2 + | Clear to Send + (RS422) (RS-485 DL260) |
| 15 | CTS2 – | Clear to Send – (RS-422) (RS-485 DL260) |

Selecting the Program Storage Media

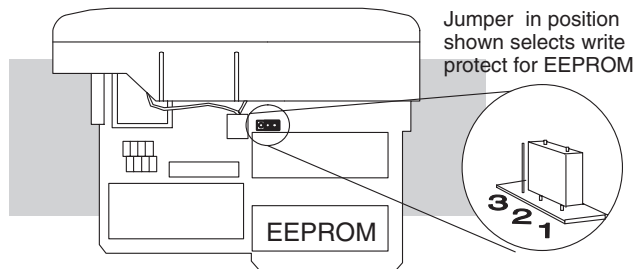
Built-in EEPROM

- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260

The DL230 and DL240 CPUs provide built-in EEPROM storage. This type of memory is non-volatile and is not dependent on battery backup to retain the program. The EEPROM can be electrically reprogrammed without being removed from the CPU. You can also set Jumper 3, which will write protect the EEPROM. The jumper is set at the factory to allow changes to EEPROM. If you select write protection by changing the jumper position, you cannot make changes to the program.



WARNING: Do NOT change Jumper 2. This is for factory test operations. If you change Jumper 2, the CPU will not operate properly.



EEPROM Sizes

The DL230 and DL240 CPUs use different sizes of EEPROMs. The CPUs come from the factory with EEPROMs already installed. However, if you need extra EEPROMs, select one that is compatible with the following part numbers.

| CPU Type | EEPROM Part Number | Capacity |
|----------|----------------------|----------------|
| DL230 | Hitachi HN58C65P-25 | 8K byte (2Kw) |
| DL240 | Hitachi HN58C256P-20 | 32K byte (3Kw) |

EEPROM Operations

Many AUX functions are specifically for use with an EEPROM in the Handheld Programmer. This enables you to quickly and easily copy programs between a program developed offline in the Handheld Programmer and the CPU. Also, you can erase EEPROMs, compare them, etc. See the DL205 Handheld Programmer Manual for details on using these AUX functions with the Handheld Programmer.

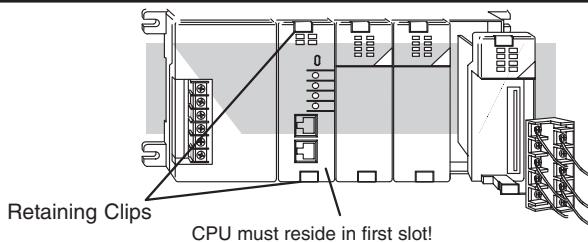
NOTE: If the instructions are supported in both CPUs and the program size is within the limits of the DL230, you can move a program between the two CPUs. However, the EEPROM installed in the Handheld Programmer must be the same size as (or larger than) the CPU being used. For example, you could not install a DL240 EEPROM in the Handheld Programmer and download the program to a DL230. Instead, if the program is within the size limits of the DL230, use a DL230 chip in the Handheld when you obtain the program from the DL240.



Installing the CPU

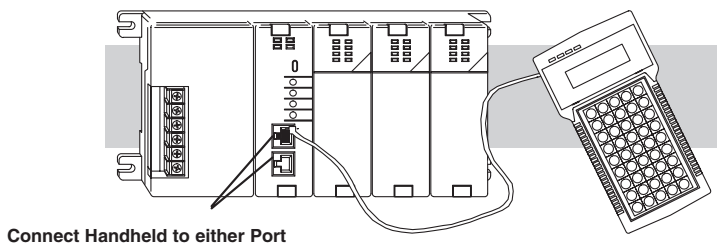
- ✓ 230 The CPU must be installed in the first slot in the base (closest to the power supply). You
 - ✓ 240 cannot install the CPU in any other slot. When inserting the CPU into the base, align the
 - ✓ 250-1 PC board with the grooves on the top and bottom of the base. Push the CPU straight into
 - ✓ 260 the base until it is firmly seated in the backplane connector. Use the retaining clips to secure
- the CPU to the base.

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

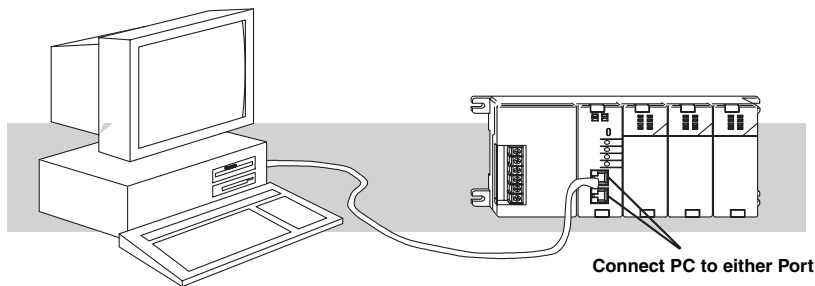


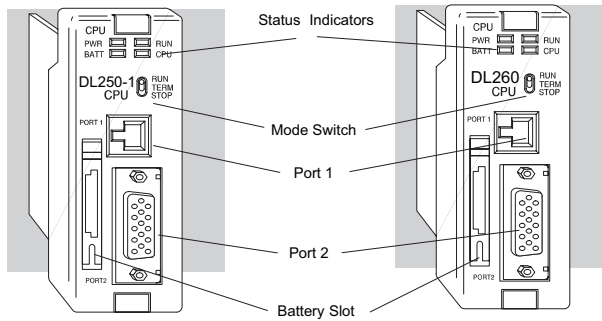
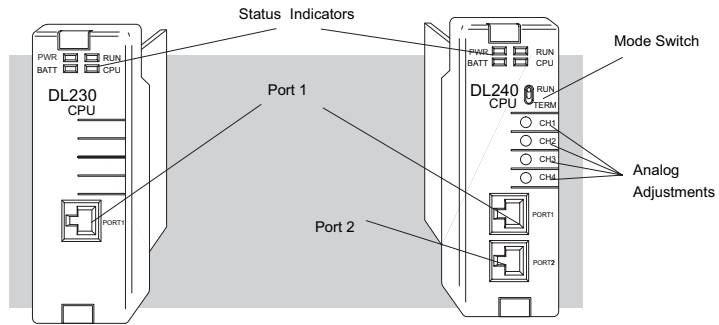
Connecting the Programming Devices

The handheld programmer is connected to the CPU with a Handheld Programmer cable. You can connect the Handheld Programmer to either port on a DL240 CPU. The Handheld Programmer is shipped with a cable. The cable is approximately 6.5 ft (200cm).



If you are using a Personal Computer with the *DirectSOFT* programming package, you can use either the top or bottom port.





CPU Setup Information

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

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

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

Status Indicators

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

| Indicator | Status | Meaning |
|-----------|----------|--|
| PWR | ON | Power good |
| | OFF | Power failure |
| RUN | ON | CPU is in Run Mode |
| | OFF | CPU is in Stop or Program Mode |
| | Blinking | CPU is in Firmware Upgrade Mode |
| CPU | ON | CPU self diagnostics error |
| | OFF | CPU self diagnostics good |
| BATT | ON | Low battery voltage (only with System Memory bit B7633.12 set) |
| | OFF | CPU battery voltage is good or disabled |

Mode Switch Functions

The mode switch on the DL240, DL250–1 and DL260 CPUs provides positions for enabling and disabling program changes in the CPU. Unless the mode switch is in the TERM position, RUN and STOP mode changes will not be allowed by any interface device, (Handheld Programmer, *DirectSOFT* programming package or operator interface). Programs may be viewed or monitored but no changes may be made. If the switch is in the TERM position and no program password is in effect, all operating modes as well as program access will be allowed through the connected programming or monitoring device.

The CPU mode can be changed in two ways:

- Use the CPU mode switch to select the operating mode.
- Place the CPU mode switch in the TERM position and use a programming device to change operating modes. In this position, you can change between Run and Program modes.



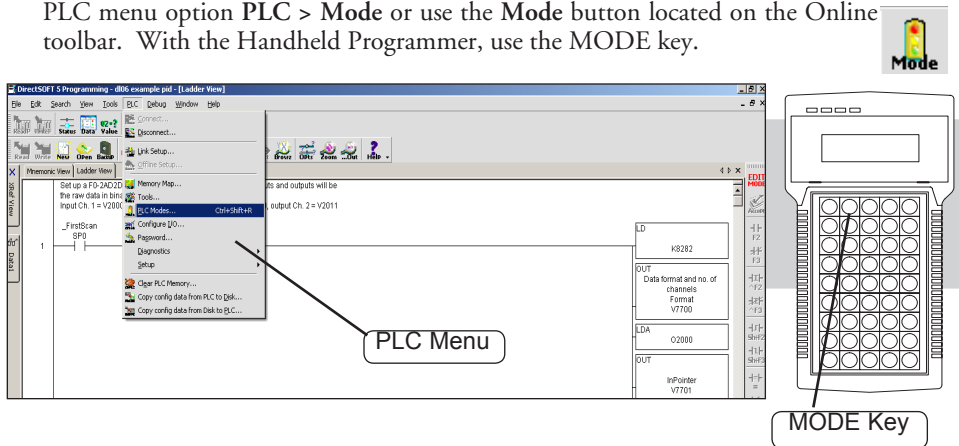
NOTE: If the PLC is switched to the RUN Mode without a program in the CPU, the CPU will produce a FATAL ERROR which can be cleared by cycling the power to the PLC.

| Mode Switch Position | CPU Action |
|--|---|
| RUN (Run Program) | CPU is forced into the RUN mode if no errors are encountered. No changes are allowed by the attached programming/monitoring device. |
| TERM (Terminal) | RUN, PROGRAM and the TEST modes are available. Mode and program changes are allowed by the programming/monitoring device. |
| STOP (DL250–1 and DL260 only Stop Program) | CPU is forced into the STOP mode. No changes are allowed by the programming/monitoring device. |

Changing Modes in the DL205 PLC

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

The CPU mode can be changed in two ways: you can use the CPU mode switch to select the operating mode, or you can place the mode switch in the TERM position and use a programming device to change operating modes. With the switch in this position, the CPU can be changed between Run and Program modes. You can use either *DirectSOFT* or the Handheld Programmer to change the CPU mode of operation. With *DirectSOFT* use the PLC menu option **PLC > Mode** or use the **Mode** button located on the Online toolbar. With the Handheld Programmer, use the **MODE** key.



Mode of Operation at Power Up

The DL205 CPUs will normally power up in the mode that it was in just prior to the power interruption. For example, if the CPU was in Program Mode when the power was disconnected, the CPU will power up in Program Mode (see warning note below).



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

The mode in which the CPU will power up in is also determined by the state of System Memory bit B7633.13. If the bit is set and the Mode Switch is in the TERM position, the CPU will power-up in RUN mode. If B7633.13 is not set with the Mode Switch in TERM position, then the CPU will power up in the state it was in when it was powered down.

Using Battery Backup

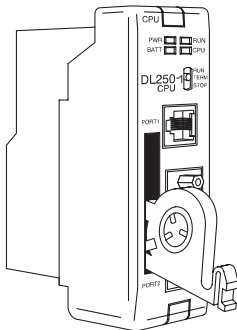
An optional lithium battery is available to maintain the system RAM retentive memory when the DL205 system is without external power. Typical CPU battery life is five years, which includes PLC runtime and normal shut-down periods. However, consider installing a fresh battery if your battery has not been changed recently and the system will be shut down for a period of more than ten days.



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

To install the D2-BAT CPU battery in DL230 or DL240 CPUs:

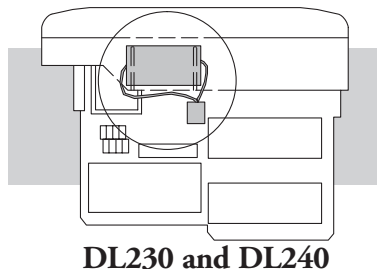
1. Gently push the battery connector onto the circuit board connector.
2. Push the battery into the retaining clip. Don't use excessive force. You may break the retaining clip.
3. Make a note of the date the battery was installed.



DL250-1 and DL260

To install the D2-BAT-1 CPU battery in the DL250-1/DL260 CPUs: (#CR2354)

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



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

Battery Backup

The battery backup is available immediately after the battery has been installed in the DL205 CPUs. *The battery low (BATT) indicator will turn on if the battery is less than 2.5VDC* (refer to the Status Indicator table on page 3-12). Special Relay 43 (SP43) will also be activated. The low battery indication is enabled by setting bit 12 of V7633 (B7633.12). If the low-battery feature is not desired, do not set bit V7633.12.

The super capacitor will retain memory IF it is configured as retentive regardless of the state of B7633.12. The battery will be the same, but for a much longer time.

Auxiliary Functions

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

You can access the AUX Functions from *DirectSOFT* or from the DL205 Handheld Programmer. The manuals for those products provide step-by-step procedures for accessing the AUX Functions. Some of these AUX Functions are designed specifically for the Handheld Programmer setup, so they will not be needed (or available) with the *DirectSOFT* package. The following table shows a list of the Auxiliary functions for the different CPUs and the Handheld Programmer.



NOTE: The Handheld Programmer may have additional AUX functions that are not supported with the DL205 CPUs.

| AUX Function and Description | 230 | 240 | 250-1 | 260 |
|-------------------------------------|-----|-----|-------|-----|
| AUX 2* — RLL Operations | | | | |
| 21 Check Program | ✓ | ✓ | ✓ | ✓ |
| 22 Change Reference | ✓ | ✓ | ✓ | ✓ |
| 23 Clear Ladder Range | ✓ | ✓ | ✓ | ✓ |
| 24 Clear All Ladders | ✓ | ✓ | ✓ | ✓ |
| AUX 3* — V-Memory Operations | | | | |
| 31 Clear V Memory | ✓ | ✓ | ✓ | ✓ |
| AUX 4* — I/O Configuration | | | | |
| 41 Show I/O Configuration | ✓ | ✓ | ✓ | ✓ |
| 42 I/O Diagnostics | ✓ | ✓ | ✓ | ✓ |
| 44 Power-up I/O Configuration Check | ✓ | ✓ | ✓ | ✓ |
| 45 Select Configuration | ✓ | ✓ | ✓ | ✓ |
| 46 Configure I/O | X | X | ✓ | ✓ |
| AUX 5* — CPU Configuration | | | | |
| 51 Modify Program Name | ✓ | ✓ | ✓ | ✓ |
| 52 Display / Change Calendar | X | ✓ | ✓ | ✓ |
| 53 Display Scan Time | ✓ | ✓ | ✓ | ✓ |
| 54 Initialize Scratchpad | ✓ | ✓ | ✓ | ✓ |
| 55 Set Watchdog Timer | ✓ | ✓ | ✓ | ✓ |
| 56 Set CPU Network Address | X | ✓ | ✓ | ✓ |
| 57 Set Retentive Ranges | ✓ | ✓ | ✓ | ✓ |
| 58 Test Operations | ✓ | ✓ | ✓ | ✓ |
| 59 Bit Override | X | ✓ | ✓ | ✓ |
| 5B Counter Interface Config. | ✓ | ✓ | ✓ | ✓ |
| 5C Display Error History | X | ✓ | ✓ | ✓ |

| AUX Function and Description | 230 | 240 | 250-1 | 260 | HPP |
|---|-----|-----|-------|-----|-----|
| AUX 6* — Handheld Programmer Configuration | | | | | |
| 61 Show Revision Numbers | ✓ | ✓ | ✓ | ✓ | — |
| 62 Beeper On / Off | X | X | X | X | ✓ |
| 65 Run Self Diagnostics | X | X | X | X | ✓ |
| AUX 7* — EEPROM Operations | | | | | |
| 71 Copy CPU memory to HPP EEPROM | X | X | X | X | ✓ |
| 72 Write HPP EEPROM to CPU | X | X | X | X | ✓ |
| 73 Compare CPU to HPP EEPROM | X | X | X | X | ✓ |
| 74 Blank Check (HPP EEPROM) | X | X | X | X | ✓ |
| 75 Erase HPP EEPROM | X | X | X | X | ✓ |
| 76 Show EEPROM Type (CPU and HPP) | X | X | X | X | ✓ |
| AUX 8* — Password Operations | | | | | |
| 81 Modify Password | ✓ | ✓ | ✓ | ✓ | — |
| 82 Unlock CPU | ✓ | ✓ | ✓ | ✓ | — |
| 83 Lock CPU | ✓ | ✓ | ✓ | ✓ | — |

✓ Supported
X Not Supported
- Not Applicable

Clearing an Existing Program

Before you enter a new program, you should 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 DL205 CPUs maintain system parameters in a memory area often referred to as the “scratchpad.” In some cases, you may make changes to the system setup that will be stored in system memory. For example, if you specify a range of Control Relays (CRs) as retentive, these changes are stored. AUX 54 resets the system memory to the default values.



WARNING: You may never have to use this feature unless you want to clear any set-up information that is stored in system memory. Usually, you will only need to initialize the system memory if you are changing programs and the old program required a special system setup. You can usually change from program to program without ever initializing system memory. Remember, this AUX function will reset all system memory. If you have set special parameters such as retentive ranges, etc., they will be erased when AUX 54 is used. Make sure that you have considered all ramifications of this operation before you select it.

Setting the Clock and Calendar

- ☐ 230
- ☒ 240
- ☒ 250-1
- ☒ 260

The DL240, DL250–1 and DL260 also have a Clock/Calendar that can be used for many purposes. If you need to use this feature, AUX functions are available that allow you to set the date and time. For example, you would use AUX 52, Display/Change Calendar to set the time and date with the Handheld Programmer. With *DirectSOFT* you would use the PLC set-up menu options using K–Sequence protocol only.

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





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

Handheld Programmer Display

23:08:17 08/02/20

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

Setting the CPU Network Address

-  **230** The DL240, DL250-1 and DL260 CPUs have built in *DirectNet* ports. You can use the Handheld Programmer to set the network address for the port and the port communication parameters. The default settings are:
-  **240**
-  **250-1**
- Station Address 1
-  **260**
- Hex Mode
 - Odd Parity
 - 9600 Baud

The *DirectNet* Manual provides additional information about choosing the communication settings for network operation.

Setting Retentive Memory Ranges

The DL205 CPUs provide certain ranges of retentive memory by default. The default ranges are suitable for many applications, but you can change them if your application requires additional retentive ranges or no retentive ranges at all. The default settings are:

| Memory Area | DL230 | | DL240 | | DL250-1 | | DL260 | |
|-----------------------|-----------------|--------------|-----------------|--------------|-----------------|--------------|-----------------|--------------|
| | Default Range | Avail. Range | Default Range | Avail. Range | Default Range | Avail. Range | Default Range | Avail. Range |
| Control Relays | C300 – C377 | C0 – C377 | C300 – C377 | C0 – C377 | C1000 – C1777 | C0 – C1777 | C1000 – C3777 | C0 – C3777 |
| V-Memory | V2000 – V7777 | V0 – V7777 | V2000 – V7777 | V0 – V7777 | V1400 – V3777 | V0 – V17777 | V400 – V37777 | V0 – V37777 |
| Timers | None by default | T0 – T77 | None by default | T0 – T177 | None by default | T0 – T377 | None by default | T0 – T377 |
| Counters | CT0 – CT77 | CT0 – CT77 | CT0 – CT177 | CT0 – CT177 | CT0 – CT177 | CT0 – CT177 | CT0 – CT377 | CT0 – CT377 |
| Stages | None by default | S0 – S377 | None by default | S0 – S777 | None by default | S0 – S1777 | None by default | S0 – S1777 |

You can use AUX 57 to set the retentive ranges. You can also use *DirectSOFT* menus to select the retentive ranges.



WARNING: The DL205 CPUs do not come with a battery. The super capacitor will retain the values in the event of a power loss, but only for a short period of time, depending on conditions. If the retentive ranges are important for your application, make sure you obtain the optional battery.

Using a Password

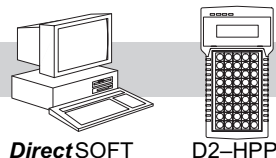
The DL205 CPUs allow you to use a password to help minimize the risk of unauthorized program and/or data changes. Once you enter a password you can “lock” the CPU against access. Once the CPU is locked you must enter the password before you can use a programming device to change any system parameters.

You can select an 8-digit numeric password. The CPUs 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 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. For more information on passwords, see the appropriate appendix on auxiliary functions.



WARNING: Make sure you remember your password. If you forget your password you will not be able to access the CPU. The CPU must be returned to the factory to have the password (along with the ladder project) removed. It is the policy of AutomationDirect to require the memory of the PLC to be cleared along with the password.

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



Select AUX 81



PASSWORD
00000000

Enter the new 8-digit password



PASSWORD
XXXXXXXX

Press CLR to clear the display

The CPU can be locked three ways once the password has been entered.

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

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



NOTE: The DL240, DL250-1 and DL260 CPUs offer multi-level passwords for even more 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).

Setting the Analog Potentiometer Ranges

- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260

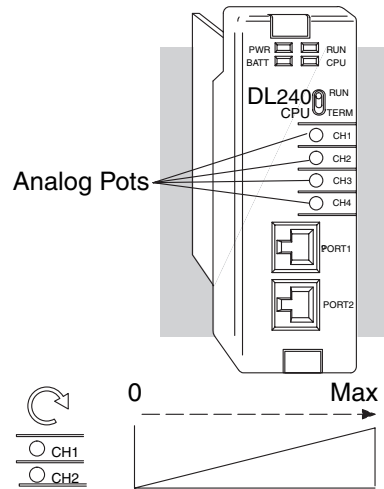
Four analog potentiometers (pots) are on the face plate of the DL240 CPU. These pots can be used to change timer constants, frequency of pulse train output, value for an analog output module, etc.

Each analog channel has corresponding V-memory locations for setting lower and upper limits for each analog channel.

To increase the value associated with the analog pot, turn the pot clockwise. To decrease the value, turn the pot counter clockwise

Turn clockwise to increase value.

The table below shows the V-memory locations used for each analog channel. These are the default locations for the analog pots.



3

| | CH1 | CH2 | CH3 | CH4 |
|-------------------------|-------|-------|-------|-------|
| Analog Data | V3774 | V3775 | V3776 | V3777 |
| Analog Data Lower Limit | V7640 | V7642 | V7644 | V7646 |
| Analog Data Upper Limit | V7641 | V7643 | V7645 | V7647 |

You can use the program logic to load the limits into these locations, or, you can use a programming device to load the values. The range for each limit is 0 – 9999.

These analog pots have a resolution of 256 pieces. Therefore, if the span between the upper and lower limits is less than or equal to 256, then you have better resolution or, more precise control.

Use the formula shown to determine the smallest amount of change that can be detected.

For example, a range of 100 – 600 would result in a resolution of 1.95. Therefore, the smallest increment would be 1.95 units. (The actual result depends on exactly how you are using the values in the control program).

$$\text{Resolution} = \frac{H - L}{256}$$

H = high limit of the range

L = low limit of the range

Example Calculations:

H = 600

L = 100

$$\text{Resolution} = \frac{600 - 100}{256}$$

$$\text{Resolution} = \frac{500}{256}$$

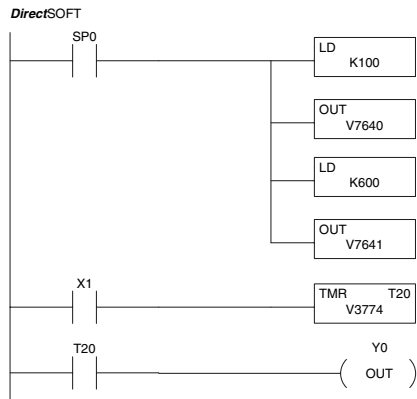
$$\text{Resolution} = 1.95$$

Chapter 3: CPU Specifications and Operations

The following example shows how you could use these analog potentiometers to change the preset value for a timer. See Chapter 5 for details on how these instructions operate.

3

Program loads ranges into V-memory

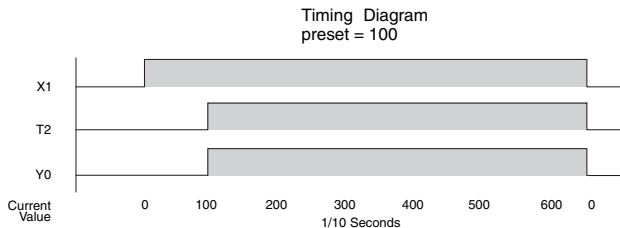
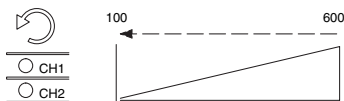


Load the lower limit (100) for the analog range on Ch1 into V7640.

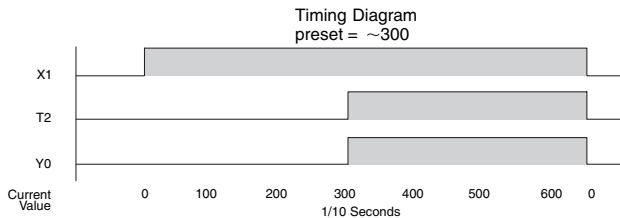
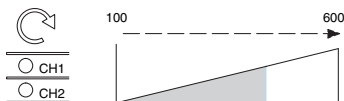
Load the upper limit (600) for the analog range on Ch1 into V7641.

Use V3774 as the preset for the timer. This will allow you to quickly adjust the preset from 100 to 600 with the CH1 analog pot.

Turn all the way counter-clockwise to use lowest value



Turn clockwise to increase the timer preset.



CPU Operation

Achieving the proper control for your equipment or process requires a good understanding of how DL205 CPUs control all aspects of system operation. The flowchart below shows the main tasks of the CPU operating system. In this section, we will investigate four aspects of CPU operation:

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

CPU Operating System

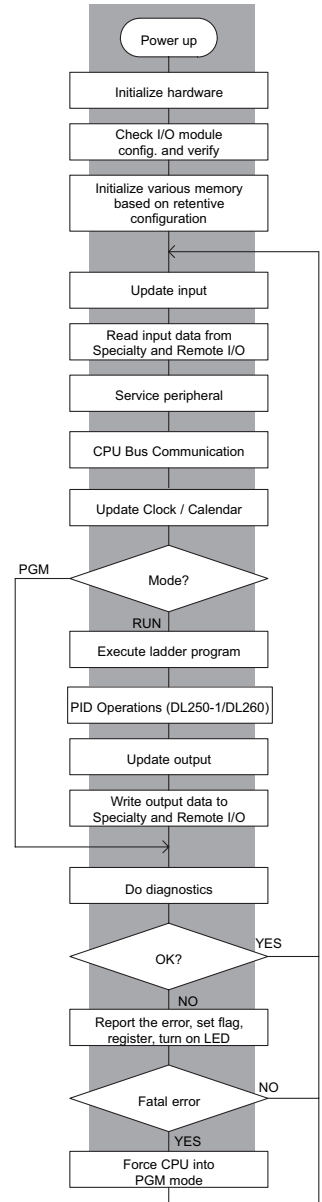
At power up, the CPU initializes the internal electronic hardware. Memory initialization starts with examining the retentive memory settings. In general, the contents of retentive memory are preserved, and non-retentive memory is initialized to zero (unless otherwise specified).

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

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

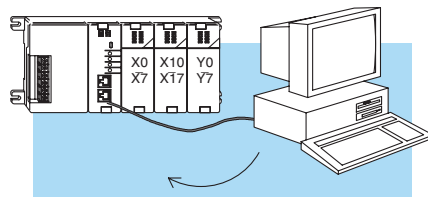
In Run Mode, the CPU executes the user ladder program. Immediately afterwards, any PID loops which are configured are executed (DL250-1 and DL260). Then the CPU writes the output results of these two tasks to the appropriate output points.

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



Program Mode Operation

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



Download Program

You can use the mode switch on the DL250-1 and DL260 CPUs to select Program Mode operation. Or, with the switch in TERM position, you can use a programming device such as the Handheld Programmer to place the CPU in Program Mode.

Run Mode Operation

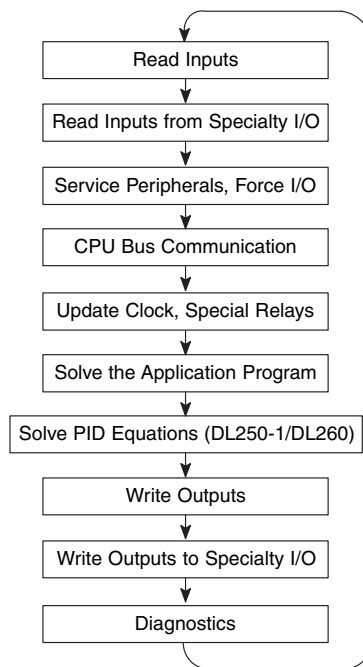
In Run Mode, the CPU executes the application program, does PID calculations for configured PID loops (DL250-1/DL260), and updates the I/O system. You can perform many operations during Run Mode. Some of these include:

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

Run Mode operation can be divided into several key areas. It is very important you understand how each of these areas of execution can affect the results of your application program solutions.

You can use the mode switch to select Run Mode operation (DL240, DL250-1 and DL260). Or, with the mode switch in TERM position, you can use a programming device, such as the Handheld Programmer, to place the CPU in Run Mode.

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



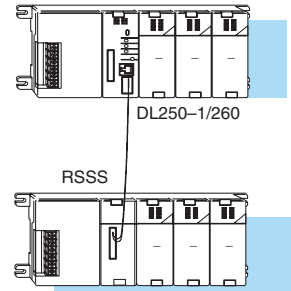
WARNING: Only authorized personnel fully familiar with all aspects of the application should make changes to the program. Changes during Run Mode become effective immediately. Make sure you thoroughly consider the impact of any changes to minimize the risk of personal injury or damage to equipment.

Read Inputs

The CPU reads the status of all inputs, then stores it in the image register. Input image register locations are designated with an X followed by a memory location. Image register data is used by the CPU when it solves the application program. Of course, an input may change after the CPU has read the inputs. Generally, the CPU scan time is measured in milliseconds. If you have an application that cannot wait until the next I/O update, you can use Immediate Instructions. These do not use the status of the input image register to solve the application program. The Immediate instructions immediately read the input status directly from I/O modules. However, this lengthens the program scan since the CPU has to read the I/O point status again. A complete list of the Immediate instructions is included in Chapter 5.

Read Inputs from Specialty and Remote I/O

After the CPU reads the inputs from the input modules, it reads any input point data from any Specialty modules that are installed, such as Counter Interface modules, etc. This is also the portion of the scan that reads the input status from Remote I/O bases.



NOTE: It may appear the Remote I/O point status is updated every scan. This is not quite true. The CPU will receive information from the Remote I/O Master module every scan, but the Remote Master may not have received an update from all the Remote Slaves. Remember, the Remote I/O link is managed by the Remote Master, not the CPU.

Service Peripherals and Force I/O

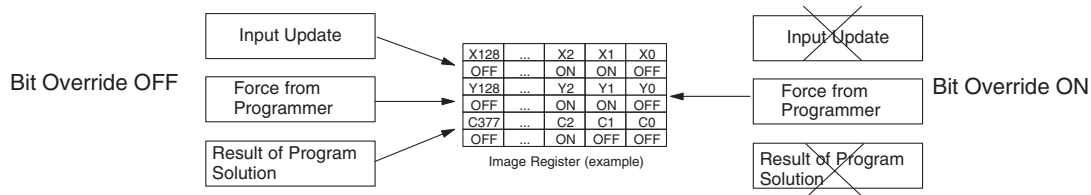
After the CPU reads the inputs from the input modules, it reads any attached peripheral devices. This is primarily a communications service for any attached devices. For example, it would read a programming device to see if any input, output, or other memory type status needs to be modified. Two basic types of forcing are available with the DL205 CPUs.

NOTE: DirectNet protocol does not support bit operations.

- Forcing from a peripheral – not a permanent force, good only for one scan
- Bit Override (DL240, DL250-1 and DL260) – holds the I/O point (or other bit) in the current state. Valid bits are X, Y, C, T, CT, and S. These memory types are discussed in more detail later in this chapter.

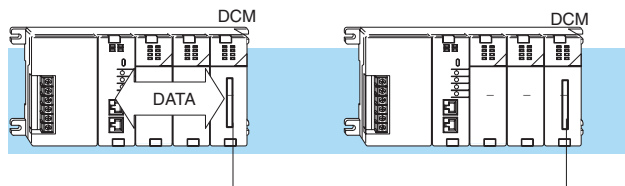
Regular Forcing — This type of forcing can temporarily change the status of a discrete bit. For example, you may want to force an input on, even though it is really off. This allows you to change the point status that was stored in the image register. This value will be valid until the image register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

Bit Override — (DL240, DL250–1 and DL260) Bit override can be enabled on a point-by-point basis by using AUX 59 from the Handheld Programmer or, by a menu option from within *DirectSOFT*. Bit override basically disables any changes to the discrete point by the CPU. For example, if you enable bit override for X1, and X1 is off at the time, then the CPU will not change the state of X1. This means that even if X1 comes on, the CPU will not acknowledge the change. So, if you used X1 in the program, it would always be evaluated as “off” in this case. Of course, if X1 was on when the bit override was enabled, then X1 would always be evaluated as “on.” There is an advantage available when you use the bit override feature. The regular forcing is not disabled because the bit override is enabled. For example, if you enabled the Bit Override for Y0 and it was off at the time, then the CPU would not change the state of Y0. However, you can still use a programming device to change the status. Now, if you use the programming device to force Y0 on, it will remain on and the CPU will not change the state of Y0. If you then force Y0 off, the CPU will maintain Y0 as off. The CPU will never update the point with the results from the application program or from the I/O update until the bit override is removed. The following diagram shows a brief overview of the bit override feature. Notice the CPU does not update the Image Register when bit override is enabled



CPU Bus Communication

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



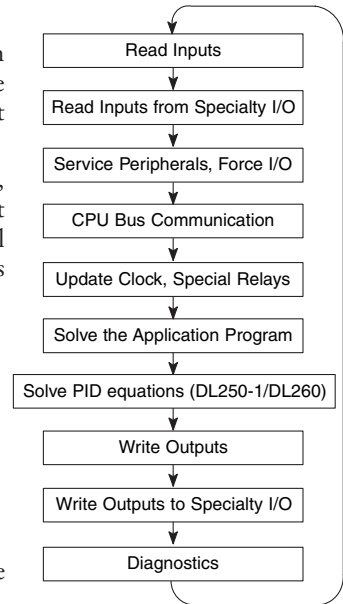
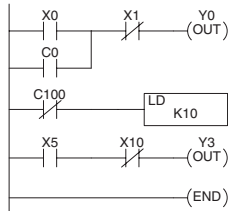
Update Clock, Special Relays and Special Registers

The DL240, DL250–1 and DL260 CPUs have an internal real-time clock and calendar timer which are accessible to the application program. Special V-memory locations hold this information. This portion of the execution cycle makes sure these locations get updated on every scan. Several different Special Relays, such as diagnostic relays, etc., are also updated during this segment.

Solve Application Program

The CPU evaluates each instruction in the application program during this segment of the scan cycle. The instructions define the relationship between input conditions and the system outputs.

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



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

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

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

Solve PID Loop Equations

- ✗ 230
- ✗ 240
- ✓ 250-1
- ✓ 260

The DL260 CPU can process up to 16 PID loops and the DL250-1 can process up to 4 PID loops. The loop calculations are run as a separate task from the ladder program execution, immediately following it. Only loops that have been configured are calculated, and then only according to a built-in loop scheduler. The sample time (calculation interval) of each loop is programmable. Please refer to Chapter 8, PID Loop Operation, for more on the effects of PID loop calculation on the overall CPU scan time.

Write Outputs

Once the application program has solved the instruction logic and constructed the output image register, the CPU writes the contents of the output image register to the corresponding output points located in the local CPU base or the local expansion bases. Remember, the CPU also made sure any forcing operation changes were stored in the output image register, so the forced points get updated with the status specified earlier.

Write Outputs to Specialty and Remote I/O

After the CPU updates the outputs in the local and expansion bases, it sends the output point information that is required by any Specialty modules that are installed. For example, this is the portion of the scan that writes the output status from the image register to the Remote I/O racks.

NOTE: It may appear the Remote I/O point status is updated every scan. This is not quite true. The CPU will send the information to the Remote I/O Master module every scan, but the Remote Master will update the actual remote modules during the next communication sequence between the master and slave modules. Remember, the Remote I/O link communication is managed by the Remote Master, not the CPU.

Diagnostics

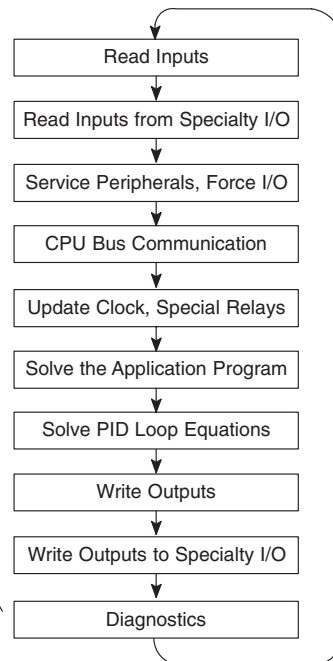
During this part of the scan, the CPU performs all system diagnostics and other tasks, such as:

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

DL205 CPUs automatically detect and report many different error conditions. Appendix B contains a listing of the various error codes available with the DL205 system.

One of the more important diagnostic tasks is the scan time calculation and watchdog timer control. DL205 CPUs have a “watchdog” timer that stores the maximum time allowed for the CPU to complete the solve application segment of the scan cycle. The default value set from the factory is 200ms. If this time is exceeded the CPU will enter the Program Mode, turn off all outputs, and report the error. For example, the Handheld Programmer displays “E003 S/W TIMEOUT” when the scan overrun occurs.

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



I/O Response Time

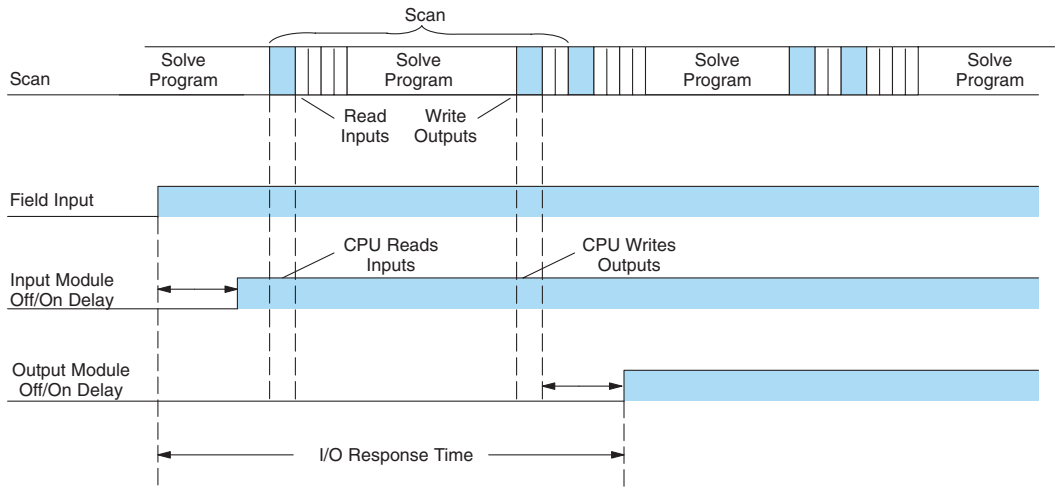
Is Timing Important for Your Application?

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

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

Normal Minimum I/O Response

The I/O response time is shortest when the module senses the input change before the Read Inputs portion of the execution cycle. In this case the input status is read, the application program is solved, and the output point gets updated. The following diagram shows an example of the timing for this situation.



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

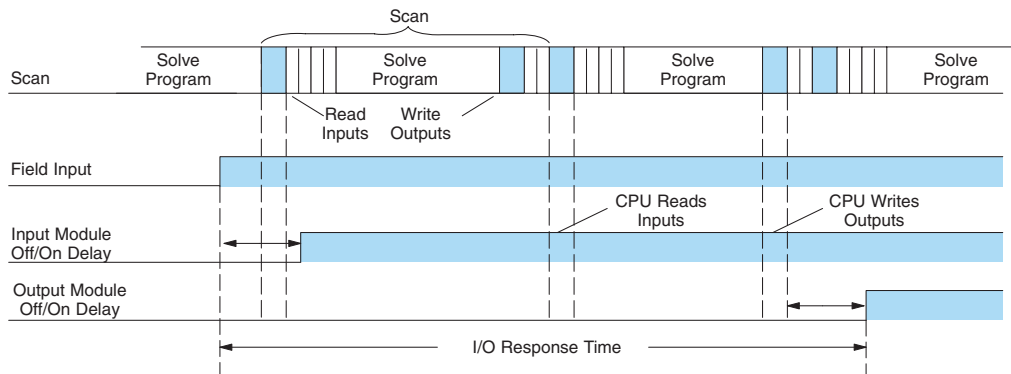
$$\text{Input Delay} + \text{Scan Time} + \text{Output Delay} = \text{Response Time}$$

Normal Maximum I/O Response

The I/O response time is longest when the module senses the input change 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:

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

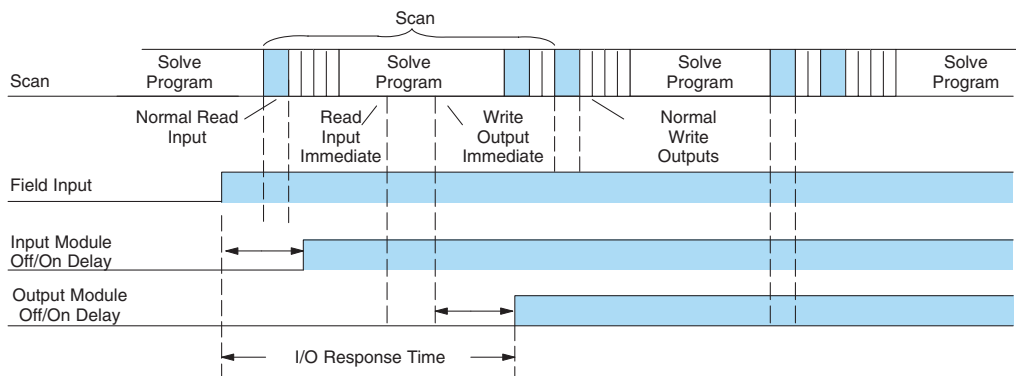


Improving Response Time

You can do a few things to help improve throughput.

- Choose instructions with faster execution times
- Use immediate I/O instructions (which update the I/O points during the ladder program execution segment)
- Choose modules that have faster response times

Immediate I/O instructions are probably the most useful technique. The following example shows immediate input and output instructions and their effect.



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

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

The instruction execution time is calculated by adding the time for the immediate input instruction, the immediate output instruction, and all instructions in between.



NOTE: When the immediate instruction reads the current status from a module, it uses the results to solve that one instruction without updating the image register. Therefore, any regular instructions that follow will still use image register values. Any immediate instructions that follow will access the module again to update the status.

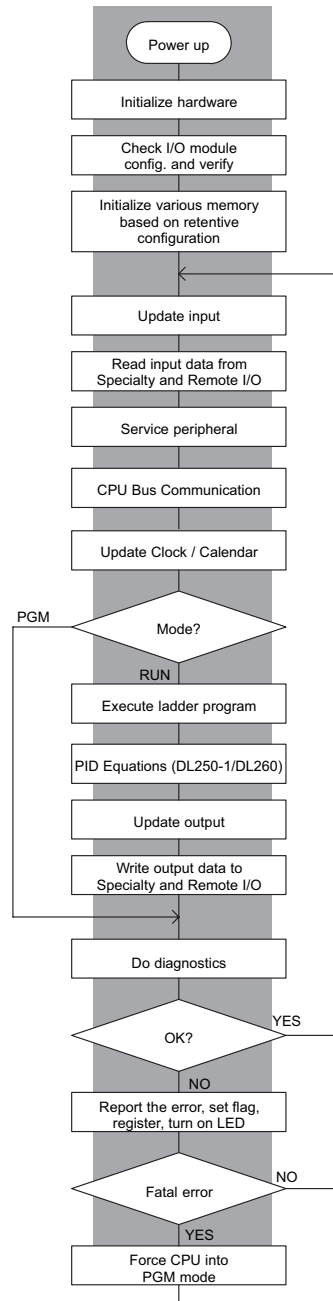
CPU Scan Time Considerations

The scan time covers all the cyclical tasks that the operating system performs. You can use *DirectSOFT* 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 system performance.

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

Your choice of I/O modules and system configuration, such as expansion or remote I/O, can also affect the scan time; however, the application usually dictates them.

For example, if you need to count pulses at high rates of speed, then you will probably have to use a High-Speed Counter module. Also, if you have I/O points that need to be located several hundred feet from the CPU, then you need remote I/O because it is much faster and cheaper to install a single remote I/O cable than it is to run all those signal wires for each individual I/O point. The following paragraphs provide some general information on how much time some of the segments can require.



Initialization Process

The CPU performs an initialization task once the system power is on. The initialization task is performed once at power up, so it does not affect the scan time for the application program.

Reading Inputs

3

| Initialization | DL230 | DL240 | DL250-1 | DL260 |
|----------------|-------------|-------------|------------------------------|-------------------------------|
| Minimum Time | 1.6 Seconds | 1.0 Seconds | 1.2 Seconds | 1.2 Seconds |
| Maximum Time | 3.6 Seconds | 2.0 Seconds | 2.7 Seconds(w/ 2 exp. bases) | 3.7 Seconds (w/ 4 exp. bases) |

The time required to read the input status for the input modules depends on which CPU you are using and the number of input points in the base. The following table shows typical update times required by the CPU.

For example, the time required for a DL240 to read two 8-point input modules would be

| Timing Factors | DL230 | DL240 | DL250-1 | DL260 |
|-----------------|--------------|--------------|--------------|--------------|
| Overhead | 64.0 μ s | 32.0 μ s | 12.6 μ s | 12.6 μ s |
| Per input point | 6.0 μ s | 12.3 μ s | 2.5 μ s | 2.5 μ s |

calculated as follows, where NI is the total number of input points:

Formula

$$\text{Time} = 32\mu\text{s} + (12.3 \times \text{NI})$$

Example

$$\text{Time} = 32\mu\text{s} + (12.3 \times 16)$$

$$\text{Time} = 228.8 \mu\text{s}$$



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

Reading Inputs from Specialty I/O

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

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

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

| Remote Module | DL230 | DL240 | DL250-1 | DL260 |
|--------------------------|-------|--------------|--------------|--------------|
| Overhead | N/A | 6.0 μ s | 1.82 μ s | 1.82 μ s |
| Per module (with inputs) | N/A | 67.0 μ s | 17.9 μ s | 17.9 μ s |
| Per input point | N/A | 40.0 μ s | 2.0 μ s | 2.0 μ s |

For example, the time required for a DL240 to read two 8-point input modules (located in a Remote base) would be calculated as follows, where NM is the number of modules and NI is the total number of input points:

Remote I/O

Formula

$$\text{Time} = 6\mu\text{s} + (67\mu\text{s} \times \text{NM}) + (40\mu\text{s} \times \text{NI})$$

Example

$$\text{Time} = 6\mu\text{s} + (67\mu\text{s} \times 2) + (40\mu\text{s} \times 16)$$

$$\text{Time} = 780\mu\text{s}$$

Service Peripherals

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

| To Log Request (anytime) | | DL230 | DL240 | DL250-1 | DL260 |
|--------------------------|------------------|---------------|---------------|-------------------|-------------------|
| Nothing Connected | Min. & Max. | 0 μ s | 0 μ s | 0 μ s | 0 μ s |
| Port 1 | Send Min. / Max. | 22/28 μ s | 23/26 μ s | 3.2/9.2 μ s | 3.2/9.2 μ s |
| | Rec. Min. / Max. | 24/58 μ s | 52/70 μ s | 25.0/35.0 μ s | 25.0/35.0 μ s |
| Port 2 | Send Min. / Max. | N/A | 26/30 μ s | 3.6/11.5 μ s | 3.6/11.5 μ s |
| | Rec. Min. / Max. | N/A | 60/75 μ s | 35.0/44.0 μ s | 35.0/44.0 μ s |

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

| To Service Request | DL230 | DL240 | DL250-1 | DL260 |
|--------------------|-------------|-----------|-----------|-------------|
| Minimum | 260µs | 250µs | 8µs | 8µs |
| Run Mode Max. | 30ms | 20ms | 410µs | 410µs |
| Program Mode Max. | 3.5 Seconds | 4 Seconds | 2 Seconds | 3.7 Seconds |

CPU Bus Communication

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



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

Update Clock/Calendar, Special Relays, Special Registers

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

| Modes | | DL230 | DL240 | DL250-1 | DL260 |
|--------------|---------|--------------|---------|---------|---------|
| Program Mode | Minimum | 8.0 µs fixed | 35.0 µs | 11.0 µs | 11.0 µs |
| | Maximum | 8.0 µs fixed | 48.0 µs | 11.0 µs | 11.0 µs |
| Run Mode | Minimum | 20.0 µs | 60.0 µs | 19.0 µs | 19.0 µs |
| | Maximum | 26.0 µs | 85.0 µs | 26.0 µs | 26.0 µs |

Writing Outputs

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

| Timing Factors | DL230 | DL240 | DL250-1 | DL260 |
|------------------|---------|---------|---------|---------|
| Overhead | 66.0 µs | 33.0 µs | 28.1 µs | 28.1 µs |
| Per output point | 8.5 µs | 14.6 µs | 3.0 µs | 3.0 µs |

For example, the time required for a DL240 to write data for two 8-point output modules would be calculated as follows (where NO is the total number of output points):

Formula

$$\text{Time} = 33 + (\text{NO} \times 14.6 \mu\text{s})$$

Example

$$\text{Time} = 33 + (16 \times 14.6 \mu\text{s})$$

$$\text{Time} = 266.6 \mu\text{s}$$

Writing Outputs to Specialty I/O

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

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

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

| Remote Module | DL230 | DL240 | DL250-1 | DL260 |
|---------------------------|-------|--------------|--------------|--------------|
| Overhead | N/A | 6.0 μ s | 1.9 μ s | 1.9 μ s |
| Per module (with outputs) | N/A | 67.5 μ s | 17.7 μ s | 17.7 μ s |
| Per output point | N/A | 46.0 μ s | 3.2 μ s | 3.2 μ s |

For example, the time required for a DL240 to write two 8-point output modules (located in a Remote base) would be calculated as follows, where NM is the number of modules and NO is the total number of output points:

Remote I/O

Formula

$$\text{Time} = 6\mu\text{s} + (67.5 \mu\text{s} \times \text{NM}) + (46\mu\text{s} \times \text{NO})$$

Example

$$\text{Time} = 6\mu\text{s} + (67.5 \mu\text{s} \times 2) + (46\mu\text{s} \times 16)$$

$$\text{Time} = 877\mu\text{s}$$



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

Diagnostics

The DL205 CPUs perform many types of system diagnostics. The amount of time required depends on many things, such as the number of I/O modules installed, etc. The following table shows the minimum and maximum times that can be expected.

| Diagnostic Time | DL230 | DL240 | DL250-1 | DL260 |
|-----------------|---------------|---------------|---------------|---------------|
| Minimum | 600.0 μ s | 422.0 μ s | 26.8 μ s | 26.8 μ s |
| Maximum | 900.0 μ s | 855.0 μ s | 103.0 μ s | 103.0 μ s |

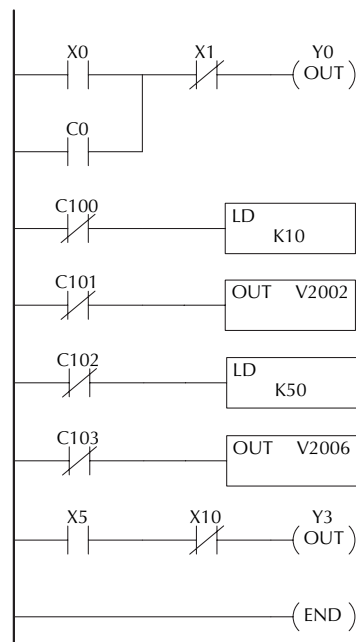
Application Program Execution

The CPU processes the program from the top (address 0) to the END instruction. The CPU executes the program left to right and top to bottom. As each rung is evaluated, the appropriate image register or memory location is updated.

The time required to solve the application program depends on the type and number of instructions used and the amount of execution overhead.

You can add the execution times for all the instructions in your program to find the total program execution time. For example, the execution time for a DL240 running the program shown would be calculated as follows:

| Instruction | Time |
|--------------|----------------|
| STR X0 | 1.4μs |
| OR C0 | 1.0μs |
| ANDN X1 | 1.2μs |
| OUT Y0 | 7.95μs |
| STRN C100 | 1.6μs |
| LD K10 | 62.0μs |
| STRN C101 | 1.6μs |
| OUT V2002 | 21.0μs |
| STRN C102 | 1.6μs |
| LD K50 | 62.0μs |
| STRN C103 | 1.6μs |
| OUT V2006 | 21.0μs |
| STR X5 | 1.4μs |
| ANDN X10 | 1.2μs |
| OUT Y3 | 7.95μs |
| END | 16.0μs |
| TOTAL | 210.5μs |



Appendix C provides a complete list of instruction execution times for DL205 CPUs.

Program Control Instructions — the DL240, DL250–1 and DL260 CPUs offer additional instructions that can change the way the program executes. These instructions include FOR/NEXT loops, Subroutines, and Interrupt Routines. These instructions can interrupt the normal program flow and affect the program execution time. Chapter 5 provides detailed information on how these different types of instructions operate.

PLC Numbering Systems

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

octal 49.832 binary
 ? 1482 BCD ?
 3A9 ? 0402 ?
 7 ? ASCII
 1001011011 ? hexadecimal
 decimal -961428 ? 1011
 -300124 A 72B ?

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

PLC Resources

PLCs offer a fixed number 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 two.

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

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

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

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

After *counting* PLC resources, it'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.

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

V-Memory

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

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

V-memory data is 16-bit binary, but we rarely program the data registers one bit at a time. We use

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

instructions or viewing tools that let us work with binary, decimal, octal, and hexadecimal numbers. All these are converted and stored as binary for us. A frequently-asked question is “How do I tell if a number is binary, octal, BCD, or hex”? The answer is that we usually cannot tell 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 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, we prefer to enter and view PLC data in decimal as well (via operator interfaces). However, computers are more efficient in using pure binary numbers. A compromise solution between the two is Binary-Coded Decimal (BCD) representation. A BCD digit ranges from 0 to 9, and is stored as 4 binary bits (a nibble). This permits each V-memory location to store 4 BCD digits, with a range of decimal numbers from 0000 to 9999.

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

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

Hexadecimal Numbers

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

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

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

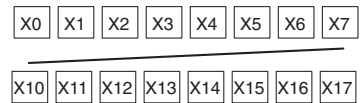
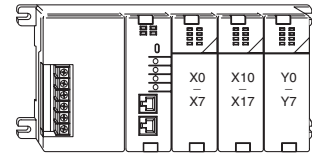
| Hexadecimal number | A | 7 | F | 4 |
|--------------------|---------|---------|---------|---------|
| V-memory storage | 1 0 1 0 | 0 1 1 1 | 1 1 1 1 | 0 1 0 0 |

Memory Map

With any PLC system, you generally have many different types of information to process. This includes input device status, output device status, various timing elements, parts counts, etc. It is important to understand how the system represents and stores the various types of data. For example, you need to know how the system identifies input points, output points, data words, etc. The following paragraphs discuss the various memory types used in the DL205 CPUs. A memory map overview for the DL230, DL240, DL250–1 and DL260 CPUs follows the memory descriptions.

Octal Numbering System

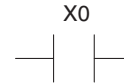
All memory locations or areas are numbered in Octal (base 8). For example, the diagram shows how the octal numbering system works for the discrete input points. Notice the octal system does not contain any numbers with the digits 8 or 9.



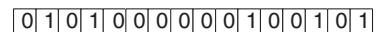
Discrete and Word Locations

As you examine the different memory types, you'll notice two types of memory in the DL205, 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

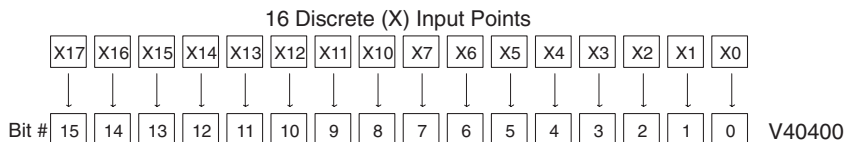


Word Locations – 16 bits



V-Memory Locations for Discrete Memory Areas

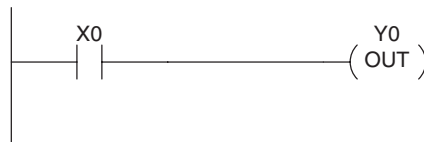
The discrete memory area is for inputs, outputs, control relays, special relays, stages, timer status bits and counter status bits. However, you can also access the bit data types as a V-memory word. Each V-memory location contains 16 consecutive discrete locations. For example, the following diagram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



These discrete memory areas and their corresponding V-memory ranges are listed in the memory area table for the DL230, DL240, DL250–1 and DL260 CPUs in this chapter.

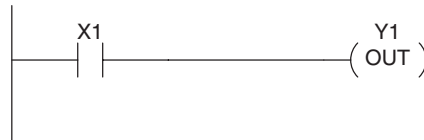
Input Points (X Data Type)

The discrete input points are noted by an X data type. Up to 512 discrete input points are available with the DL205 CPUs. In this example, the output point Y0 will be turned on when input X0 energizes.



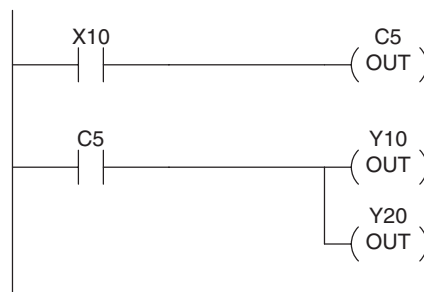
Output Points (Y Data Type)

The discrete output points are noted by a Y data type. Up to 512 discrete output points are available with the DL205 CPUs. In this example, output point Y1 will turn on when input X1 energizes.



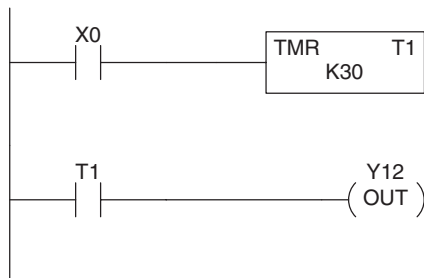
Control Relays (C Data Type)

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. Control relays are internal to the CPU and 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 has 16 consecutive discrete locations. In this example, memory location C5 will energize when input X10 turns on. The second rung shows a simple example of how to use a control relay as an input.



Timers and Timer Status Bits (T Data Type)

The number of timers available depends on the model of CPU you are using. The tables at the end of this section provide the number of timers for the DL230, DL240, D2-250-1 and DL260. Regardless of the number of timers, you have access to timer status bits that reflect the relationship between the current value and the preset value of a specified timer. The timer status bit will be on when the current value is equal to or greater than the preset value of a corresponding timer.

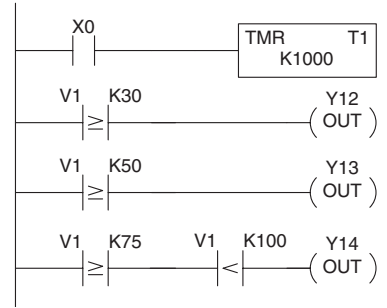


When input X0 turns on, timer T1 will start. When the timer reaches the preset of 3 seconds (K of 30), timer status contact T1 turns on. When T1 turns on, output Y12 turns on.

Timer Current Values (V Data Type)

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

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

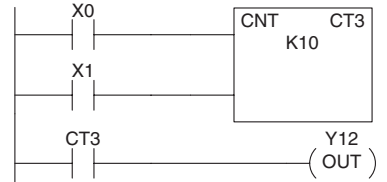


Counters and Counter Status Bits

(CT Data Type)

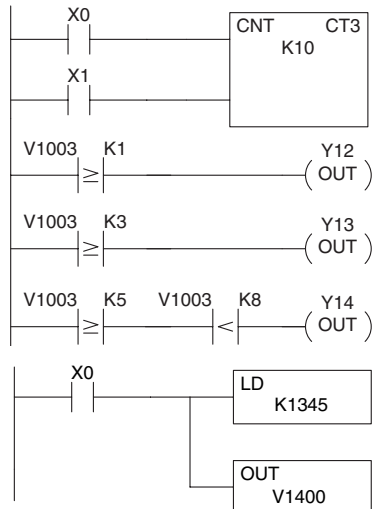
You have access to counter status bits that reflect the relationship between the current value and the preset value of a specified counter. The counter status bit will be on when the current value is equal to or greater than the preset value of a corresponding counter.

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)

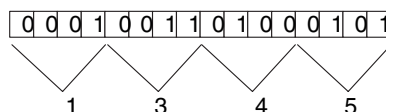
Just like the timers, the counter current values are also automatically stored in V-memory. For example, V1000 holds the current value for Counter CT0, V1001 holds the current value for Counter CT1, etc. These are 4-digit BCD values. The primary reason for this is programming flexibility. The example shows how you can use relational contacts to monitor the counter values.



Word Memory (V Data Type)

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

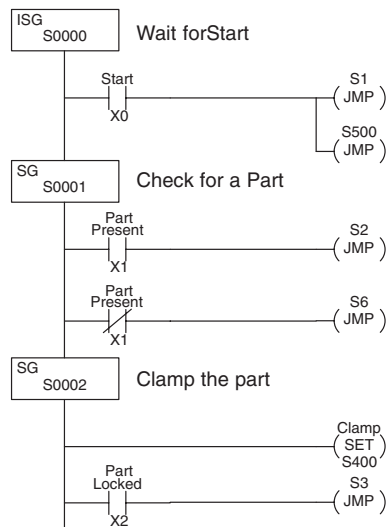
Word Locations – 16 bits



Stages (S Data type)

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

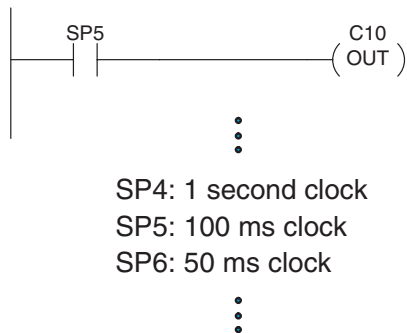
Each stage also has a discrete status bit that can be used as an input to indicate whether the stage is active or inactive. If the stage is active, then the status bit is on. If the stage is inactive, then the status bit is off. This status bit can also be turned on or off by other instructions, such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



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.

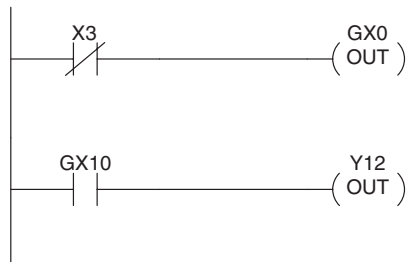
In this example, control relay C10 will energize for 50ms and de-energize for 50 ms because SP5 is a pre-defined relay that will be on for 50ms and off for 50ms.



Remote I/O Points (GX Data Type)

Remote I/O points are represented by global relays. They are generally used only to control remote I/O, but they can be used as normal control relays when remote I/O is not used in the system.

In this example, memory location GX0 represents an output point and memory location GX10 represents an input point.



DL230 System V-memory

| System V-memory | Description of Contents | Default Values/Ranges |
|-----------------|---|--|
| V2320–V2377 | The default location for multiple preset values for the UP counter. | N/A |
| V7620–V7627 | Locations for DV–1000 operator interface parameters V7620 Sets the V-memory location that contains the value. V7621 Sets the V-memory location that contains the message. V7622 Sets the total number (1 - 16) of V-memory locations to be displayed. V7623 Sets the V-memory location that contains the numbers to be displayed. V7624 Sets the V-memory location that contains the character code to be displayed. V7625 Sets the bit control pointer. V7626 Power Up mode change preset value password. V7627 Reserved for future use. | V0–V2377 V0–V2377 1–16 V0–V2377 V0–V2377 V-memory location for X,Y, or C points used. 0,1,2,3,12 Default = 0000 |
| 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 24 presets are 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 mode for the high speed counter, interrupt, pulse catch, pulse train, and input filter (see the D2-CTRINT Manual, D2-CTRIF-M for more information). Location is also used for setting the with/without battery option, enable/disable CPU mode change, and power-up in Run Mode option. | Default: 0000 Lower Byte Range: Range: 0–None 10–Up 40–Interrupt 50–Pulse Catch 60–Filtered discrete In. Upper Byte Range: Bits 8–11, 14,15: Unused Bit 12: With Batt. installed: 0 = disable BATT LED 1 = enable BATT LED Bit 13: Power-up in Run |
| V7634 | Contains set-up information for high-speed counter, interrupt, pulse catch, pulse train output, and input filter for X0 (when D2–CTRINT is installed). | Default: 0000 |
| V7635 | Contains set up-information for high-speed counter, interrupt, pulse catch, pulse train output, and input filter for X1 (when D2–CTRINT is installed). | Default: 0000 |
| V7636 | Contains set-up information for high-speed counter, interrupt, pulse catch, pulse train output, and input filter for X2 (when D2–CTRINT is installed). | Default: 0000 |
| V7637 | Contains set-up information for high-speed counter, interrupt, pulse catch, pulse train output, and input filter for X3 (when D2–CTRINT is installed). | Default: 0000 |
| V7640–V7642 | Additional setup parameters for the DV-1000 V7640 Timer preset value pointer V7641 Counter preset value pointer V7642 Timer preset block size (high byte) / Counter preset block size (low byte) | V2000–V2377 V2000–V2377 1–99 |
| V7643–V7647 | Not used | N/A |
| V7751 | Fault Message Error Code — stores the 4-digit code used with the FAULT instruction when the instruction is executed. | N/A |

| System V-memory | Description of Contents | Default Values/Ranges |
|-----------------|--|-----------------------|
| V7752 | I/O Configuration Error — stores the module ID code for the module that does not match the current configuration. | N/A |
| V7753 | I/O Configuration Error — stores the correct module ID code. | |
| V7754 | I/O Configuration Error — identifies the base and slot number. | |
| V7755 | Error code — stores the fatal error code. | N/A |
| V7756 | Error code — stores the major error code. | N/A |
| V7757 | Error code — stores the minor error code. | |
| V7760–V7764 | Module Error — stores the slot number and error code where an I/O error occurs. | |
| 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 | |
| V7775 | Scan — stores the current scan time (milliseconds). | |
| V7776 | Scan — stores the minimum scan time that has occurred since the last Program Mode to Run Mode transition (milliseconds). | N/A |
| V7777 | Scan — stores the maximum scan time that has occurred since the last Program Mode to Run Mode transition (milliseconds). | N/A |

DL240 System V-memory

| System V-memory | Description of Contents | Default Values/Ranges |
|-----------------|---|--|
| V3630–V3707 | The default location for multiple preset values for UP/DWN and UP counter 1 or pulse output function. | N/A |
| V3710–V3767 | The default location for multiple preset values for UP/DWN and UP counter 2. | N/A |
| V3770–V3773 | Not used | N/A |
| V3774–V3777 | Default locations for analog potentiometer data (channels 1–4, respectively). | Range: 0 – 9999 |
| V7620–V7627 | Locations for DV–1000 operator interface parameters V7620 Sets the V-memory location that contains the value. V7621 Sets the V-memory location that contains the message. V7622 Sets the total number (1 – 16) of V-memory locations to be displayed. V7623 Sets the V-memory location that contains the numbers to be displayed. V7624 Sets the V-memory location that contains the character code to be displayed. V7625 Sets the bit control pointer V7626 Power Up Mode V7627 Change Preset Value Password. | V0 – V3760 V0 – V3760 1 – 16 V0 – V3760 V0 – V3760 V-memory location for X, Y, or C points used. 0,1,2,3,12 Default=0000 |
| V7630 | Starting location for the multi-step presets for channel 1. Since there are 24 presets available, the default range is V3630 – V3707. You can change the starting point if necessary. | Default: V3630 Range: V0 – V3710 |
| V7631 | Starting location for the multi-step presets for channel 2. Since there are 24 presets available, the default range is V3710– V3767. You can change the starting point if necessary. | Default: V3710 Range: V0 – V3710 |
| V7632 | Contains the baud rate setting for Port 2. You can use AUX 56 (from the Handheld Programmer) or, use DirectSOFT to set the port parameters if 9600 baud is unacceptable. Also allows you to set a delay time between the assertion of the RTS signal and the transmission of data. This is useful for radio modems that require a key-up delay before data is transmitted. e.g., a value of 0302 sets 10ms Turnaround Delay (TAD) and 9600 baud. | Default: 2 – 9600 baud Lower Byte = Baud Rate Lower Byte Range: 00 = 300 01 = 1200 02 = 9600 03 = 19.2K Upper Byte = Time Delay Upper Byte Range: 01 = 2ms 02 = 5ms 03 = 10ms 04 = 20ms 05 = 50ms 06 = 100ms 07 = 500ms |

| System V-memory | Description of Contents | Default Values/ Ranges |
|-----------------|--|---|
| V7633 | Sets the desired mode for the high speed counter, interrupt, pulse catch, pulse train, and input filter (see the D2-CTRINT manual, D2-CTRIF-M, for more information). Location is also used for setting the with/without battery option, enable/disable CPU mode change. | Default: 0000 Lower Byte Range: 0 – None 10 – Up 20 – Up/Dwn. 30 – Pulse Out 40 – Interrupt 50 – Pulse Catch 60 – Filtered Dis. Upper Byte Range: Bits 8 – 11, 15 Unused Bit 12: With Batt. installed: 0 = disable BATT LED 1 = enable BATT LED Bit 13: Power-up in Run Bit 14: Mode chg. enable (K-sequence only) |
| V7634 | Contains set-up information for high-speed counter, interrupt, pulse catch, pulse train output, and input filter for X0 (when D2-CTRINT is installed). | Default: 0000 |
| V7635 | Contains set-up information for high-speed counter, interrupt, pulse catch, pulse train output, and input filter for X1 (when D2-CTRINT is installed). | Default: 0000 |
| V7636 | Contains set-up information for high-speed counter, interrupt, pulse catch, pulse train output, and input filter for X2 (when D2-CTRINT is installed). | Default: 0000 |
| V7637 | Contains set-up information for high-speed counter, interrupt, pulse catch, pulse train output, and input filter for X3 (when D2-CTRINT is installed). | Default: 0000 |
| V7640–V7641 | Location for setting the lower and upper limits for the CH1 analog pot. | Default: 0000 Range: 0 – 9999 |
| V7642–V7643 | Location for setting the lower and upper limits for the CH2 analog pot. | Default: 0000 Range: 0 – 9999 |
| V7644–V7645 | Location for setting the lower and upper limits for the CH3 analog pot. | Default: 0000 Range: 0 – 9999 |
| V7646–V7647 | Location for setting the lower and upper limits for the CH4 analog pot. | Default: 0000 Range: 0 – 9999 |
| V7650–V7737 | Locations reserved for set-up information used with future options (remote I/O and data communications). | |
| V7720–V7722 | Locations for DV-1000 operator interface parameters. | |
| V7720 | Timed Timer preset value pointer . | V2000–V2377 |
| V7721 | Timed Counter preset value pointer. | V2000–V2377 |
| V7722 | HiByte-Timed Timer preset block size, LoByte-Timed Counter preset block size. | 1–99 |
| V7746 | Location contains the battery voltage, accurate to 0.1V. For example, a value of 32 indicates 3.2 volts. | |
| V7747 | Location contains a 10ms counter. This location increments once every 10ms. | |
| V7751 | Fault Message Error Code — stores the 4-digit code used with the FAULT instruction when the instruction is executed. If you've used ASCII messages (DL240 only), then the data label (DLBL) reference number for that message is stored here. | |
| V7752 | I/O configuration Error — stores the module ID code for the module that does not match the current configuration. | |

| System V-memory | Description of Contents |
|-----------------|--|
| V7753 | I/O Configuration Error — stores the correct module ID code. |
| V7754 | I/O Configuration Error — identifies the base and slot number. |
| V7755 | Error code — stores the fatal error code. |
| V7756 | Error code — stores the major error code. |
| V7757 | Error code — stores the minor error code. |
| V7760–V7764 | Module Error — stores the slot number and error code where an I/O error occurs. |
| V7765 | Scan—stores the number of scan cycles that have occurred since the last Program to Run Mode transition. |
| V7766 | Contains the number of seconds on the clock. (00 to 59). |
| V7767 | Contains the number of minutes on the clock. (00 to 59). |
| V7770 | Contains the number of hours on the clock. (00 to 23). |
| V7771 | Contains the day of the week. (Mon, Tue, etc.). |
| V7772 | Contains the day of the month (1st, 2nd, etc.). |
| V7773 | Contains the month. (01 to 12) |
| V7774 | Contains the year. (00 to 99) |
| V7775 | Scan — stores the current scan time (milliseconds). |
| V7776 | Scan — stores the minimum scan time that has occurred since the last Program Mode to Run Mode transition (milliseconds). |
| V7777 | Scan — stores the maximum scan time that has occurred since the last Program Mode to Run Mode transition (milliseconds). |

DL250–1 System V-memory (DL250 also)

| System V-memory | Description of Contents | Default Values/Ranges |
|-----------------|--|---|
| V3630–V3707 | The default location for multiple preset values for UP/DWN and UP counter 1 or pulse output function | N/A |
| V3710–V3767 | The default location for multiple preset values for UP/DWN and UP counter 2. | N/A |
| V3770–V3777 | Not used | N/A |
| V7620–V7627 | Locations for DV–1000 operator interface parameters V7620 Sets the V-memory location that contains the value V7621 Sets the V-memory location that contains the message V7622 Sets the total number (1 – 32) of V-memory locations to be displayed V7623 Sets the V-memory location that contains the numbers to be displayed V7624 Sets the V-memory location that contains the character code to be displayed V7625 Sets the bit control pointer V7626 Sets the power up mode V7627 Change Preset Value password | V0 – V3760 V0 – V3760 1 – 32 V0 – V3760 V0 – V3760 V-memory for X, Y, or C 0,1,2,3,12 Default=0000 |
| V7630 | Starting location for the multi-step presets for channel 1. Since there are 24 presets available, the default range is V3630 – V3707. You can change the starting point if necessary. | Default: V3630 Range: V0 – V3710 |
| V7631 | Starting location for the multi-step presets for channel 2. Since there are 24 presets available, the default range is V3710– V3767. You can change the starting point if necessary. | Default: V3710 Range: V0 – V3710 |
| V7632 | Reserved | |
| V7633 | Sets the desired mode for the high-speed counter, interrupt, pulse catch, pulse train, and input filter (see the D2-CTRINT manual, D2-CTRIF-M, for more information). Location is also used for setting the with/without battery option, enable/disable CPU mode change, and power-up in Run Mode option. | Default: 0060 Lower Byte Range: Range: 0 – None 10 – Up 20 – Up/Dwn. 30 – Pulse Out 40 – Interrupt 50 – Pulse Catch 60 – Filtered Dis. Upper Byte Range: Bits 8 – 11, 14–15 Unused Bit 12: With Batt. installed: 0 = disable BATT LED 1 = enable BATT LED Bit 13: Power-up in Run |
| V7634 | Contains set-up information for high-speed counter, interrupt, pulse catch, pulse train output, and input filter for X0 (when D2–CTRINT is installed). | Default: 1006 |
| V7635 | Contains set-up information for high-speed counter, interrupt, pulse catch, pulse train output, and input filter for X1 (when D2–CTRINT is installed). | Default: 1006 |
| V7636 | Contains set-up information for high-speed counter, interrupt, pulse catch, pulse train output, and input filter for X2 (when D2–CTRINT is installed). | Default: 1006 |

| System V-memory | Description of Contents | Default Values/Ranges |
|-----------------|---|---------------------------|
| V7637 | Contains set-up information for high-speed counter, interrupt, pulse catch, pulse train output, and input filter for X3 (when D2-CTRINT is installed). | Default: 1006 |
| V7640 | Loop Table Beginning address. | V1400–V7340 V10000–V17740 |
| V7641 | Number of Loops Enabled | 1–4 |
| V7642 | Error Code – V-memory Error Location for Loop Table. | |
| V7643–V7647 | Reserved. | |
| V7650 | Port 2 End-code setting Setting (A55A), Non-procedure communications start. | |
| V7651 | Port 2 Data format – Non-procedure communications format setting. | |
| V7652 | Port 2 Format Type setting – Non-procedure communications type code setting. | |
| V7653 | Port 2 Terminate-code setting – Non-procedure communications Termination code setting. | |
| V7654 | Port 2 Store v-mem address – Non-procedure communication data store V-Memory address | |
| V7655 | Port 2 Setup area –0–7 Comm protocol (flag 0) 8–15 Comm time out/response delay time (flag 1). | |
| V7656 | Port 2 Setup area – 0–15 Communication (flag 2, flag 3). | |
| V7657 | Port 2: Setup completion code. | |
| V7660–V7717 | Set-up Information – Locations reserved for set-up information used with future options. | |
| V7720–V7722 | Locations for DV-1000 operator interface parameters. | |
| V7720 | Titled Timer preset value pointer. | |
| V7721 | Title Counter preset value pointer. | |
| V7722 | HiByte-Titled Timer preset block size, LoByte-Titled Counter preset block size. | |
| V7740 | Port 2 Communication Auto Reset Timer setup. | |
| V7741 | Output Hold or reset setting: Expansion bases 1 and 2 (DL250–1). | |
| V7747 | Location contains a 10ms counter. This location increments once every 10ms. | |
| V7750 | Reserved. | |
| V7751 | Fault Message Error Code — stores the 4-digit code used with the FAULT instruction when the instruction is executed. If you've used ASCII messages (DL240 only), then the data label (DLBL) reference number for that message is stored here. | |
| V7752 | I/O configuration Error — stores the module ID code for the module that does not match the current configuration. | |
| V7753 | I/O Configuration Error — stores the correct module ID code. | |
| V7754 | I/O Configuration Error — identifies the base and slot number. | |
| V7755 | Error code — stores the fatal error code. | |
| V7756 | Error code — stores the major error code. | |
| V7757 | Error code — stores the minor error code. | |
| V7760–V7764 | Module Error — stores the slot number and error code where an I/O error occurs. | |
| V7765 | Scan — stores the total number of scan cycles that have occurred since the last Program Mode to Run Mode transition. | |

| System V-memory | Description of Contents |
|-----------------|---|
| V7766 | Contains the number of seconds on the clock. (00 to 59) |
| V7767 | Contains the number of minutes on the clock. (00 to 59) |
| V7770 | Contains the number of hours on the clock. (00 to 23) |
| V7771 | Contains the day of the week. (Mon, Tue, etc.) |
| V7772 | Contains the day of the month (1st, 2nd, etc.) |
| V7773 | Contains the month. (01 to 12) |
| V7774 | Contains the year. (00 to 99) |
| V7775 | Scan — stores the current scan time (milliseconds) |
| V7776 | Scan — stores the minimum scan time that has occurred since the last Program Mode to Run Mode transition (milliseconds) |
| V7777 | Scan — stores the maximum scan time that has occurred since the last Program Mode to Run Mode transition (milliseconds) |
| V36000–36057 | Analog pointer method for expansion base 1 (DL250–1) |
| V36100–36157 | Analog pointer method for expansion base 2 (DL250–1) |
| V36400–36427 | Analog pointer method for local base |
| V37700–37737 | Port 2: Setup register for Koyo Remote I/O |

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

DL260 System V-memory

| System V-memory | Description of Contents | Default Values/Ranges |
|-----------------|---|--|
| V3630–V3707 | The default location for multiple preset values for UP/DWN and UP counter 1 or pulse output function | N/A |
| V3710–V3767 | The default location for multiple preset values for UP/DWN and UP counter 2 | N/A |
| V3770–V3777 | Not used | N/A |
| V7620–V7627 | Locations for DV-1000 operator interface parameters | V0 – V3760 |
| V7620 | Sets the V-memory location that contains the value | V0 – V3760 |
| V7621 | Sets the V-memory location that contains the message | 1 – 32 |
| V7622 | Sets the total number (1 – 32) of V-memory locations to be displayed | V0 – V3760 |
| V7623 | Sets the V-memory location that contains the numbers to be displayed | V0 – V3760 |
| V7624 | Sets the V-memory location that contains the character code to be displayed | V-memory for X, Y, or C |
| V7625 | Sets the bit control pointer | 0,1,2,3,12 |
| V7626 | Sets the power up mode | Default=0000 |
| V7627 | Change Preset Value password | |
| V7630 | Starting location for the multi-step presets for channel 1. Since there are 24 presets available, the default range is V3630 – V3707. You can change the starting point if necessary. | Default: V3630 Range: V0 – V3710 |
| V7631 | Starting location for the multi-step presets for channel 2. Since there are 24 presets available, the default range is V3710– V3767. You can change the starting point if necessary. | Default: V3710 Range: V0 – V3710 |
| V7632 | Reserved | |
| V7633 | Sets the desired mode for the high-speed counter, interrupt, pulse catch, pulse train, and input filter (see the D2-CTRINT manual, D2-CTRIF-M, for more information). Location is also used for setting the with/without battery option, enable/disable CPU mode change, and power-up in Run Mode option. | Default: 0060 Lower Byte Range: Range: 0 – None 10 – Up 20 – Up/Dwn 30 – Pulse Ou 40 – Interrupt 50 – Pulse Catch 60 – Filtered Dis. Upper Byte Range Bits 8 – 11, 14–15 Unused Bit 12: With Batt. installed: 0 = disable BATT LED 1 = enable BATT LED Bit 13: Power-up in Run |
| V7634 | Contains set-up information for high-speed counter, interrupt, pulse catch, pulse train output, and input filter for X0 (when D2-CTRINT is installed) | Default: 1006 |
| V7635 | Contains set-up information for high-speed counter, interrupt, pulse catch, pulse train output, and input filter for X1 (when D2-CTRINT is installed) | Default: 1006 |
| V7636 | Contains set-up information for high-speed counter, interrupt, pulse catch, pulse train output, and input filter for X2 (when D2-CTRINT is installed) | Default: 1006 |

| System V-memory | Description of Contents | Default Values/Ranges |
|-----------------|---|--|
| V7637 | Contains set up information for high speed counter, interrupt, pulse catch, pulse train output, and input filter for X3 (when D2-CTRINT is installed). | Default: 1006 |
| V7640 | PID Loop Table Beginning address. | V400-640 V1400-V7340 V10000-V35740 |
| V7641 | Number of Loops Enabled. | 1-16 |
| V7642 | Error Code – V-memory Error Location for Loop Table. | |
| V7643 - V7647 | Reserved. | |
| V7650 | Port 2 End-code Setting (A55A), Non-procedure communications start. | |
| V7651 | Port 2 Data format - Non-procedure communications format setting. | |
| V7652 | Port 2 Format Type setting – Non-procedure communications type code setting. | |
| V7653 | Port 2 Terminate-code setting – Non-procedure communications Termination code setting | |
| V7654 | Port 2 Store v-mem address – Non-procedure communication data store V-Memory address. | |
| V7655 | Port 2 Setup area –0-7 Comm protocol (flag 0) 8-15 Comm time out/response delay time (flag 1) | |
| V7656 | Port 2 Setup area – 0-15 Communication (flag 2, flag 3) | |
| V7657 | Port 2: Setup completion code. | |
| V7660-V7717 | Set-up Information – Locations reserved for set up information used with future options. | |
| V7720-V7722 | Locations for DV-1000 operator interface parameters. | |
| V7720 | Timed Timer preset value pointer. | |
| V7721 | Title Counter preset value pointer. | |
| V7722 | HiByte-Timed Timer preset block size, LoByte-Timed Counter preset block size. | |
| V7740 | Port 2 Communication Auto Reset Timer setup. | |
| V7741 | Output Hold or reset setting: Expansion bases 1 and 2. | |
| V7742 | Output Hold or reset setting: Expansion bases 3 and 4. | |
| V7747 | Location contains a 10ms counter. This location increments once every 10ms. | |
| V7750 | Reserved. | |
| V7751 | Fault Message Error Code — stores the 4-digit code used with the FAULT instruction when the instruction is executed. If you've used ASCII messages (DL240 only), then the data label (DLBL) reference number for that message is stored here. | |
| V7752 | I/O configuration Error — stores the module ID code for the module that does not match the current configuration. | |
| V7753 | I/O Configuration Error — stores the correct module ID code. | |
| V7754 | I/O Configuration Error — identifies the base and slot number. | |
| V7755 | Error code — stores the fatal error code. | |
| V7756 | Error code — stores the major error code. | |
| V7757 | Error code — stores the minor error code. | |
| V7763-V7764 | Module Error — stores the slot number and error code where an I/O error occurs. | |
| V7765 | Scan — stores the total number of scan cycles that have occurred since the last Program Mode to Run Mode transition. | |

| System V-memory | Description of Contents |
|-----------------|--|
| V7766 | Contains the number of seconds on the clock. (00 to 59). |
| V7767 | Contains the number of minutes on the clock. (00 to 59). |
| V7770 | Contains the number of hours on the clock. (00 to 23). |
| V7771 | Contains the day of the week. (Mon, Tue, etc.). |
| V7772 | Contains the day of the month (1st, 2nd, etc.). |
| V7773 | Contains the month. (01 to 12) |
| V7774 | Contains the year. (00 to 99) |
| V7775 | Scan — stores the current scan time (milliseconds). |
| V7776 | Scan — stores the minimum scan time that has occurred since the last Program Mode to Run Mode transition (milliseconds). |
| V7777 | Scan — stores the maximum scan time that has occurred since the last Program Mode to Run Mode transition (milliseconds). |
| V36000–36057 | Analog pointer method for expansion base 1 |
| V36100–36157 | Analog pointer method for expansion base 2 |
| V36200–36257 | Analog pointer method for expansion base 3 |
| V36300–36357 | Analog pointer method for expansion base 4 |
| V36400–36427 | Analog pointer method for local base |
| V37700–37737 | Port 2: Set-up register for Koyo Remote I/O |

The following system control relays are used for Koyo Remote I/O setup on Communications Port 2.

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

DL205 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.

| DL205 Aliases | | |
|---------------|-------------|---|
| Address Start | Alias Start | Example |
| V0 | TA0 | V0 is the timer accumulator value for timer 0, therefore, its alias is TA0. TA1 is the alias for V1, etc. |
| V1000 | CTA0 | V1000 is the counter accumulator value for counter 0, therefore, its alias is CTA0. CTA1 is the alias for V1001, etc. |
| V40000 | VGX | V40000 is the word memory reference for discrete bits GX0 through GX17, therefore, its alias is VGX0. V40001 is the word memory reference for discrete bits GX20 through GX37, therefore, its alias is VGX20. |
| V40200 | VGX | V40200 is the word memory reference for discrete bits GY0 through GY17, therefore, its alias is VGX0. V40201 is the word memory reference for discrete bits GY20 through GY37, therefore, its alias is VGX20. |
| V40400 | VX0 | V40400 is the word memory reference for discrete bits X0 through X17, therefore, its alias is VX0. V40401 is the word memory reference for discrete bits X20 through X37, therefore, its alias is VX20. |
| V40500 | VY0 | V40500 is the word memory reference for discrete bits Y0 through Y17, therefore, its alias is VY0. V40501 is the word memory reference for discrete bits Y20 through Y37, therefore, its alias is VY20. |
| V40600 | VC0 | V40600 is the word memory reference for discrete bits C0 through C17, therefore, its alias is VC0. V40601 is the word memory reference for discrete bits C20 through C37, therefore, its alias is VC20. |
| V41000 | VS0 | V41000 is the word memory reference for discrete bits S0 through S17, therefore, its alias is VS0. V41001 is the word memory reference for discrete bits S20 through S37, therefore, its alias is VS20. |
| V41100 | VT0 | V41100 is the word memory reference for discrete bits T0 through T17, therefore, its alias is VT0. V41101 is the word memory reference for discrete bits T20 through T37, therefore, its alias is VT20. |
| V41140 | VCT0 | V41140 is the word memory reference for discrete bits CT0 through CT17, therefore, its alias is VCT0. V41141 is the word memory reference for discrete bits CT20 through CT37, therefore, its alias is VCT20. |
| V41200 | VSP0 | V41200 is the word memory reference for discrete bits SP0 through SP17, therefore, its alias is VSP0. V41201 is the word memory reference for discrete bits SP20 through SP37, therefore, its alias is VSP20. |

DL230 Memory Map

| Memory Type | Discrete Memory Reference (octal) | Word Memory Reference (octal) | Qty. Decimal | Symbol |
|-------------------------|-----------------------------------|------------------------------------|------------------|--|
| Input Points | X0 – X177 | V40400 – V40407 | 128 ¹ | X0 — — |
| Output Points | Y0 – Y177 | V40500 – V40507 | 128 ¹ | Y0 —() |
| Control Relays | C0 – C377 | V40600 – V40617 | 256 | C0 C0 — — —() |
| Special Relays | SP0 – SP117 SP540 – SP577 | V41200 – V41204 V41226 – V41227 | 112 | SP0 — — |
| Timers | T0 – T77 | | 64 | — TMR TO K100 |
| Timer Current Values | None | V0 – V77 | 64 | V0 K100 — ≥— |
| Timer Status Bits | T0 – T77 | V41100 – V41103 | 64 | T0 — — |
| Counters | CT0 – CT77 | | 64 | — CNT CT0 K10 |
| Counter Current Values | None | V1000 – V1077 | 64 | V1000 K100 — ≥— |
| Counter Status Bits | CT0 – CT77 | V41140 – V41143 | 64 | CT0 — — |
| Data Words | None | V2000 – V2377 | 256 | None specific, used with many instructions |
| Data Words Non-volatile | None | V4000 – V4177 | 128 | None specific, used with many instructions |
| Stages | S0 – S377 | V41000 – V41017 | 256 | — SG S001 S0 — — |
| System parameters | None | V7620 – V7647 V7750–V7777 | 48 | None specific, used for various purposes |



NOTE 1: The DL230 systems are limited to 256 discrete I/O points (total) with the present system hardware available. These can be mixed between inputs and output points as necessary.

DL240 Memory Map

| Memory Type | Discrete Memory Reference (octal) | Word Memory Reference(octal) | Qty. Decimal | Symbol |
|-------------------------|-----------------------------------|------------------------------------|------------------|--|
| Input Points | X0 – X477 | V40400 – V40423 | 320 ¹ | X0 ├─┤ |
| Output Points | Y0 – Y477 | V40500 – V40523 | 320 ¹ | Y0 ├─() |
| Control Relays | C0 – C377 | V40600 – V40617 | 256 | C0 C0 ├─┤ ├─() |
| Special Relays | SP0 – SP137 SP540 – SP617 | V41200 – V41205 V41226 – V41230 | 144 | SP0 ├─┤ |
| Timers | T0 – T177 | | 128 | — TMR T0 K100 |
| Timer Current Values | None | V0 – V177 | 128 | V0 K100 ├─┤ |
| Timer Status Bits | T0 – T177 | V41100 – V41107 | 128 | T0 ├─┤ |
| Counters | CT0 – CT177 | | 128 | — CNT CT0 K10 |
| Counter Current Values | None | V1000 – V1177 | 128 | V1000 K100 ├─┤ |
| Counter Status Bits | CT0 – CT177 | V41140 – V41147 | 128 | CT0 ├─┤ |
| Data Words | None | V2000 – V3777 | 1024 | None specific, used with many instructions |
| Data Words Non-volatile | None | V4000 – V4377 | 256 | None specific, used with many instructions |
| Stages | S0 – S777 | V41000 – V41037 | 512 | — SG S0 S001 ├─┤ |
| System parameters | None | V7620 – V7737 V7746–V7777 | 106 | None specific, used for various purposes |



NOTE 1: The DL240 systems are limited to 256 discrete I/O points (total) with the present system hardware available. These can be mixed between inputs and output points as necessary.

DL250–1 Memory Map (DL250 also)

| Memory Type | Discrete Memory Reference (octal) | Word Memory Reference (octal) | Qty. Decimal | Symbol |
|------------------------|-----------------------------------|-------------------------------|--------------|--|
| Input Points | X0 – X777 | V40400 – V40437 | 512 | X0 ├─┤ |
| Output Points | Y0 – Y777 | V40500 – V40537 | 512 | Y0 ├─┤ |
| Control Relays | C0 – C1777 | V40600 – V40677 | 1024 | C0 C0 ├─┤ └─┘ |
| Special Relays | SP0 – SP777 | V41200 – V41237 | 512 | SP0 ├─┤ |
| Timers | T0 – T377 | | 256 | └─┘ TMR T0 K100 |
| Timer Current Values | None | V0 – V377 | 256 | V0 K100 ├─┤ |
| Timer Status Bits | T0 – T377 | V41100 – V41117 | 256 | T0 ├─┤ |
| Counters | CT0 – CT177 | | 128 | └─┘ CNT CT0 K10 |
| Counter Current Values | None | V1000 – V1177 | 128 | V1000 K100 ├─┤ |
| Counter Status Bits | CT0 – CT177 | V41140 – V41147 | 128 | CT0 ├─┤ |
| Data Words | None | V1400 – V7377 V10000–V17777 | 7168 | None specific, used with many instructions |
| Stages | S0 – S1777 | V41000 – V41077 | 1024 | └─┘ SG S0 S001 └─┤ |
| System parameters | None | V7400–V7777 V36000–V37777 | 768 | None specific, used for various purposes |

DL260 Memory Map

| Memory Type | Discrete Memory Reference (octal) | Word Memory Reference (octal) | Qty. Decimal | Symbol |
|--------------------------------|-----------------------------------|--|--------------|--|
| Input Points | X0 – X1777 | V40400 – V40477 | 1024 | X0 ├─┤ |
| Output Points | Y0 – Y1777 | V40500 – V40577 | 1024 | Y0 ├─() |
| Control Relays | C0 – C3777 | V40600 – V40777 | 2048 | C0 C0 ├─┤ ├─() |
| Special Relays | SP0 – SP777 | V41200 – V41237 | 512 | SP0 ├─┤ |
| Timers | T0 – T377 | | 256 | ├─┤ TMR T0 K100 |
| Timer Current Values | None | V0 – V377 | 256 | V0 K100 ├─┤ |
| Timer Status Bits | T0 – T377 | V41100 – V41117 | 256 | T0 ├─┤ |
| Counters | CT0 – CT377 | | 256 | ├─┤ CNT CT0 K10 |
| Counter Current Values | None | V1000 – V1377 | 256 | V1000 K100 ├─┤ |
| Counter Status Bits | CT0 – CT377 | V41140 – V41157 | 256 | CT0 ├─┤ |
| Data Words | None | V400 – V777 V1400 – V7377 V10000– V35777 | 14.6K | None specific, used with many instructions |
| Stages | S0 – S1777 | V41000 – V41077 | 1024 | SG S0 ├─┤ S001 ┤├─┤ |
| Remote Input and Output Points | GX0 – GX3777 GY0 – GY3777 | V40000 – V40177 V40200–V40377 | 2048 2048 | GX0 GY0 ├─┤ ├─() |
| System parameters | None | V7400–V7777 V36000–V37777 | 1.2K | None specific, used for various purposes |

X Input/Y Output Bit Map

This table provides a listing of the individual Input points associated with each V-memory address bit for the DL230, DL240, and DL250–1 and DL260 CPUs. The DL250–1 ranges apply to the DL250.

| MSB | DL230/DL240/DL250-1/DL260 Input (X) and Output (Y) Points | | | | | | | | | | | | | | | LSB | X Input Address | Y Output Address |
|-----|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----------------|------------------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | |
| 017 | 016 | 015 | 014 | 013 | 012 | 011 | 010 | 007 | 006 | 005 | 004 | 003 | 002 | 001 | 000 | | V40400 | V40500 |
| 037 | 036 | 035 | 034 | 033 | 032 | 031 | 030 | 027 | 026 | 025 | 024 | 023 | 022 | 021 | 020 | | V40401 | V40501 |
| 057 | 056 | 055 | 054 | 053 | 052 | 051 | 050 | 047 | 046 | 045 | 044 | 043 | 042 | 041 | 040 | | V40402 | V40502 |
| 077 | 076 | 075 | 074 | 073 | 072 | 071 | 070 | 067 | 066 | 065 | 064 | 063 | 062 | 061 | 060 | | V40403 | V40503 |
| 117 | 116 | 115 | 114 | 113 | 112 | 111 | 110 | 107 | 106 | 105 | 104 | 103 | 102 | 101 | 100 | | V40404 | V40504 |
| 137 | 136 | 135 | 134 | 133 | 132 | 131 | 130 | 127 | 126 | 125 | 124 | 123 | 122 | 121 | 120 | | V40405 | V40505 |
| 157 | 156 | 155 | 154 | 153 | 152 | 151 | 150 | 147 | 146 | 145 | 144 | 143 | 142 | 141 | 140 | | V40406 | V40506 |
| 177 | 176 | 175 | 174 | 173 | 172 | 171 | 170 | 167 | 166 | 165 | 164 | 163 | 162 | 161 | 160 | | V40407 | V40507 |

| MSB | DL240/DL250-1/DL260 Input (X) and Output (Y) Points | | | | | | | | | | | | | | | LSB | | |
|-----|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|--------|--------|
| 217 | 216 | 215 | 214 | 213 | 212 | 211 | 210 | 207 | 206 | 205 | 204 | 203 | 202 | 201 | 200 | | V40410 | V40510 |
| 237 | 236 | 235 | 234 | 233 | 232 | 231 | 230 | 227 | 226 | 225 | 224 | 223 | 222 | 221 | 220 | | V40411 | V40511 |
| 257 | 256 | 255 | 254 | 253 | 252 | 251 | 250 | 247 | 246 | 245 | 244 | 243 | 242 | 241 | 240 | | V40412 | V40512 |
| 277 | 276 | 275 | 274 | 273 | 272 | 271 | 270 | 267 | 266 | 265 | 264 | 263 | 262 | 261 | 260 | | V40413 | V40513 |
| 317 | 316 | 315 | 314 | 313 | 312 | 311 | 310 | 307 | 306 | 305 | 304 | 303 | 302 | 301 | 300 | | V40414 | V40514 |
| 337 | 336 | 335 | 334 | 333 | 332 | 331 | 330 | 327 | 326 | 325 | 324 | 323 | 322 | 321 | 320 | | V40415 | V40515 |
| 357 | 356 | 355 | 354 | 353 | 352 | 351 | 350 | 347 | 346 | 345 | 344 | 343 | 342 | 341 | 340 | | V40416 | V40516 |
| 377 | 376 | 375 | 374 | 373 | 372 | 371 | 370 | 367 | 366 | 365 | 364 | 363 | 362 | 361 | 360 | | V40417 | V40517 |
| 417 | 416 | 415 | 414 | 413 | 412 | 411 | 410 | 407 | 406 | 405 | 404 | 403 | 402 | 401 | 400 | | V40420 | V40520 |
| 437 | 436 | 435 | 434 | 433 | 432 | 431 | 430 | 427 | 426 | 425 | 424 | 423 | 422 | 421 | 420 | | V40421 | V40521 |
| 457 | 456 | 455 | 454 | 453 | 452 | 451 | 450 | 447 | 446 | 445 | 444 | 443 | 442 | 441 | 440 | | V40422 | V40522 |
| 477 | 476 | 475 | 474 | 473 | 472 | 471 | 470 | 467 | 466 | 465 | 464 | 463 | 462 | 461 | 460 | | V40423 | V40523 |

| MSB | Additional DL250-1/DL260 Input (X) and Output (Y) Points | | | | | | | | | | | | | | | LSB | | |
|-----|--|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|--------|--------|
| 517 | 516 | 515 | 514 | 513 | 512 | 511 | 510 | 507 | 506 | 505 | 504 | 503 | 502 | 501 | 500 | | V40424 | V40524 |
| 537 | 536 | 535 | 534 | 533 | 532 | 531 | 530 | 527 | 526 | 525 | 524 | 523 | 522 | 521 | 520 | | V40425 | V40525 |
| 557 | 556 | 555 | 554 | 553 | 552 | 551 | 550 | 547 | 546 | 545 | 544 | 543 | 542 | 541 | 540 | | V40426 | V40526 |
| 577 | 576 | 575 | 574 | 573 | 572 | 571 | 570 | 567 | 566 | 565 | 564 | 563 | 562 | 561 | 560 | | V40427 | V40527 |
| 617 | 616 | 615 | 614 | 613 | 612 | 611 | 610 | 607 | 606 | 605 | 604 | 603 | 602 | 601 | 600 | | V40430 | V40530 |
| 637 | 636 | 635 | 634 | 633 | 632 | 631 | 630 | 627 | 626 | 625 | 624 | 623 | 622 | 621 | 620 | | V40431 | V40531 |
| 657 | 656 | 655 | 654 | 653 | 652 | 651 | 650 | 647 | 646 | 645 | 644 | 643 | 642 | 641 | 640 | | V40432 | V40532 |
| 677 | 676 | 675 | 674 | 673 | 672 | 671 | 670 | 667 | 666 | 665 | 664 | 663 | 662 | 661 | 660 | | V40433 | V40533 |
| 717 | 716 | 715 | 714 | 713 | 712 | 711 | 710 | 707 | 706 | 705 | 704 | 703 | 702 | 701 | 700 | | V40434 | V40534 |
| 737 | 736 | 735 | 734 | 733 | 732 | 731 | 730 | 727 | 726 | 725 | 724 | 723 | 722 | 721 | 720 | | V40435 | V40535 |
| 757 | 756 | 755 | 754 | 753 | 752 | 751 | 750 | 747 | 746 | 745 | 744 | 743 | 742 | 741 | 740 | | V40436 | V40536 |
| 777 | 776 | 775 | 774 | 773 | 772 | 771 | 770 | 767 | 766 | 765 | 764 | 763 | 762 | 761 | 760 | | V40437 | V40537 |

| MSB | Additional DL260 Input (X) and Output (Y) Points | | | | | | | | | | | | | | | | LSB | X Input | Y Output |
|------|--|------|------|------|------|------|------|------|------|------|------|------|------|------|------|--|---------|---------|----------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | Address | Address | |
| 1017 | 1016 | 1015 | 1014 | 1013 | 1012 | 1011 | 1010 | 1007 | 1006 | 1005 | 1004 | 1003 | 1002 | 1001 | 1000 | | V40440 | V40540 | |
| 1037 | 1036 | 1035 | 1034 | 1033 | 1032 | 1031 | 1030 | 1027 | 1026 | 1025 | 1024 | 1023 | 1022 | 1021 | 1020 | | V40441 | V40541 | |
| 1057 | 1056 | 1055 | 1054 | 1053 | 1052 | 1051 | 1050 | 1047 | 1046 | 1045 | 1044 | 1043 | 1042 | 1041 | 1040 | | V40442 | V40542 | |
| 1077 | 1076 | 1075 | 1074 | 1073 | 1072 | 1071 | 1070 | 1067 | 1066 | 1065 | 1064 | 1063 | 1062 | 1061 | 1060 | | V40443 | V40543 | |
| 1117 | 1116 | 1115 | 1114 | 1113 | 1112 | 1111 | 1110 | 1107 | 1106 | 1105 | 1104 | 1103 | 1102 | 1101 | 1100 | | V40444 | V40544 | |
| 1137 | 1136 | 1135 | 1134 | 1133 | 1132 | 1131 | 1130 | 1127 | 1126 | 1125 | 1124 | 1123 | 1122 | 1121 | 1120 | | V40445 | V40545 | |
| 1157 | 1156 | 1155 | 1154 | 1153 | 1152 | 1151 | 1150 | 1147 | 1146 | 1145 | 1144 | 1143 | 1142 | 1141 | 1140 | | V40446 | V40546 | |
| 1177 | 1176 | 1175 | 1174 | 1173 | 1172 | 1171 | 1170 | 1167 | 1166 | 1165 | 1164 | 1163 | 1162 | 1161 | 1160 | | V40447 | V40547 | |
| 1217 | 1216 | 1215 | 1214 | 1213 | 1212 | 1211 | 1210 | 1207 | 1206 | 1205 | 1204 | 1203 | 1202 | 1201 | 1200 | | V40450 | V40550 | |
| 1237 | 1236 | 1235 | 1234 | 1233 | 1232 | 1231 | 1230 | 1227 | 1226 | 1225 | 1224 | 1223 | 1222 | 1221 | 1220 | | V40451 | V40551 | |
| 1257 | 1256 | 1255 | 1254 | 1253 | 1252 | 1251 | 1250 | 1247 | 1246 | 1245 | 1244 | 1243 | 1242 | 1241 | 1240 | | V40452 | V40552 | |
| 1277 | 1276 | 1275 | 1274 | 1273 | 1272 | 1271 | 1270 | 1267 | 1266 | 1265 | 1264 | 1263 | 1262 | 1261 | 1260 | | V40453 | V40553 | |
| 1317 | 1316 | 1315 | 1314 | 1313 | 1312 | 1311 | 1310 | 1307 | 1306 | 1305 | 1304 | 1303 | 1302 | 1301 | 1300 | | V40454 | V40554 | |
| 1337 | 1336 | 1335 | 1334 | 1333 | 1332 | 1331 | 1330 | 1327 | 1326 | 1325 | 1324 | 1323 | 1322 | 1321 | 1320 | | V40455 | V40555 | |
| 1357 | 1356 | 1355 | 1354 | 1353 | 1352 | 1351 | 1350 | 1347 | 1346 | 1345 | 1344 | 1343 | 1342 | 1341 | 1340 | | V40456 | V40556 | |
| 1377 | 1376 | 1375 | 1374 | 1373 | 1372 | 1371 | 1370 | 1367 | 1366 | 1365 | 1364 | 1363 | 1362 | 1361 | 1360 | | V40457 | V40557 | |
| 1417 | 1416 | 1415 | 1414 | 1413 | 1412 | 1411 | 1410 | 1407 | 1406 | 1405 | 1404 | 1403 | 1402 | 1401 | 1400 | | V40460 | V40560 | |
| 1437 | 1436 | 1435 | 1434 | 1433 | 1432 | 1431 | 1430 | 1427 | 1426 | 1425 | 1424 | 1423 | 1422 | 1421 | 1420 | | V40461 | V40561 | |
| 1457 | 1456 | 1455 | 1454 | 1453 | 1452 | 1451 | 1450 | 1447 | 1446 | 1445 | 1444 | 1443 | 1442 | 1441 | 1440 | | V40462 | V40562 | |
| 1477 | 1476 | 1475 | 1474 | 1473 | 1472 | 1471 | 1470 | 1467 | 1466 | 1465 | 1464 | 1463 | 1462 | 1461 | 1460 | | V40463 | V40563 | |
| 1517 | 1516 | 1515 | 1514 | 1513 | 1512 | 1511 | 1510 | 1507 | 1506 | 1505 | 1504 | 1503 | 1502 | 1501 | 1500 | | V40464 | V40564 | |
| 1537 | 1536 | 1535 | 1534 | 1533 | 1532 | 1531 | 1530 | 1527 | 1526 | 1525 | 1524 | 1523 | 1522 | 1521 | 1520 | | V40465 | V40565 | |
| 1557 | 1556 | 1555 | 1554 | 1553 | 1552 | 1551 | 1550 | 1547 | 1546 | 1545 | 1544 | 1543 | 1542 | 1541 | 1540 | | V40466 | V40566 | |
| 1577 | 1576 | 1575 | 1574 | 1573 | 1572 | 1571 | 1570 | 1567 | 1566 | 1565 | 1564 | 1563 | 1562 | 1561 | 1560 | | V40467 | V40567 | |
| 1617 | 1616 | 1615 | 1614 | 1613 | 1612 | 1611 | 1610 | 1607 | 1606 | 1605 | 1604 | 1603 | 1602 | 1601 | 1600 | | V40470 | V40570 | |
| 1637 | 1636 | 1635 | 1634 | 1633 | 1632 | 1631 | 1630 | 1627 | 1626 | 1625 | 1624 | 1623 | 1622 | 1621 | 1620 | | V40471 | V40571 | |
| 1657 | 1656 | 1655 | 1654 | 1653 | 1652 | 1651 | 1650 | 1647 | 1646 | 1645 | 1644 | 1643 | 1642 | 1641 | 1640 | | V40472 | V40572 | |
| 1677 | 1676 | 1675 | 1674 | 1673 | 1672 | 1671 | 1670 | 1667 | 1666 | 1665 | 1664 | 1663 | 1662 | 1661 | 1660 | | V40473 | V40573 | |
| 1717 | 1716 | 1715 | 1714 | 1713 | 1712 | 1711 | 1710 | 1707 | 1706 | 1705 | 1704 | 1703 | 1702 | 1701 | 1700 | | V40474 | V40574 | |
| 1737 | 1736 | 1735 | 1734 | 1733 | 1732 | 1731 | 1730 | 1727 | 1726 | 1725 | 1724 | 1723 | 1722 | 1721 | 1720 | | V40475 | V40575 | |
| 1757 | 1756 | 1755 | 1754 | 1753 | 1752 | 1751 | 1750 | 1747 | 1746 | 1745 | 1744 | 1743 | 1742 | 1741 | 1740 | | V40476 | V40576 | |
| 1777 | 1776 | 1775 | 1774 | 1773 | 1772 | 1771 | 1770 | 1767 | 1766 | 1765 | 1764 | 1763 | 1762 | 1761 | 1760 | | V40477 | V40577 | |

Control Relay Bit Map

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

| MSB | DL230/DL240/DL250-1/DL260 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 |

| MSB | Additional DL250-1/DL260 Control Relays (C) | | | | | | | | | | | | | | | LSB | Address |
|-----|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|---------|
| 417 | 416 | 415 | 414 | 413 | 412 | 411 | 410 | 407 | 406 | 405 | 404 | 403 | 402 | 401 | 400 | | V40620 |
| 437 | 436 | 435 | 434 | 433 | 432 | 431 | 430 | 427 | 426 | 425 | 424 | 423 | 422 | 421 | 420 | | V40621 |
| 457 | 456 | 455 | 454 | 453 | 452 | 451 | 450 | 447 | 446 | 445 | 444 | 443 | 442 | 441 | 440 | | V40622 |
| 477 | 476 | 475 | 474 | 473 | 472 | 471 | 470 | 467 | 466 | 465 | 464 | 463 | 462 | 461 | 460 | | V40623 |
| 517 | 516 | 515 | 514 | 513 | 512 | 511 | 510 | 507 | 506 | 505 | 504 | 503 | 502 | 501 | 500 | | V40624 |
| 537 | 536 | 535 | 534 | 533 | 532 | 531 | 530 | 527 | 526 | 525 | 524 | 523 | 522 | 521 | 520 | | V40625 |
| 557 | 556 | 555 | 554 | 553 | 552 | 551 | 550 | 547 | 546 | 545 | 544 | 543 | 542 | 541 | 540 | | V40626 |
| 577 | 576 | 575 | 574 | 573 | 572 | 571 | 570 | 567 | 566 | 565 | 564 | 563 | 562 | 561 | 560 | | V40627 |
| 617 | 616 | 615 | 614 | 613 | 612 | 611 | 610 | 607 | 606 | 605 | 604 | 603 | 602 | 601 | 600 | | V40630 |
| 637 | 636 | 635 | 634 | 633 | 632 | 631 | 630 | 627 | 626 | 625 | 624 | 623 | 622 | 621 | 620 | | V40631 |
| 657 | 656 | 655 | 654 | 653 | 652 | 651 | 650 | 647 | 646 | 645 | 644 | 643 | 642 | 641 | 640 | | V40632 |
| 677 | 676 | 675 | 674 | 673 | 672 | 671 | 670 | 667 | 666 | 665 | 664 | 663 | 662 | 661 | 660 | | V40633 |
| 717 | 716 | 715 | 714 | 713 | 712 | 711 | 710 | 707 | 706 | 705 | 704 | 703 | 702 | 701 | 700 | | V40634 |
| 737 | 736 | 735 | 734 | 733 | 732 | 731 | 730 | 727 | 726 | 725 | 724 | 723 | 722 | 721 | 720 | | V40635 |
| 757 | 756 | 755 | 754 | 753 | 752 | 751 | 750 | 747 | 746 | 745 | 744 | 743 | 742 | 741 | 740 | | V40636 |
| 777 | 776 | 775 | 774 | 773 | 772 | 771 | 770 | 767 | 766 | 765 | 764 | 763 | 762 | 761 | 760 | | V40637 |

| MSB | Additional DL250-1/DL260 Control Relays (C) | | | | | | | | | | | | | | | LSB | Address |
|------|---|------|------|------|------|------|------|------|------|------|------|------|------|------|------|--------|---------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |
| 1017 | 1016 | 1015 | 1014 | 1013 | 1012 | 1011 | 1010 | 1007 | 1006 | 1005 | 1004 | 1003 | 1002 | 1001 | 1000 | V40640 | |
| 1037 | 1036 | 1035 | 1034 | 1033 | 1032 | 1031 | 1030 | 1027 | 1026 | 1025 | 1024 | 1023 | 1022 | 1021 | 1020 | V40641 | |
| 1057 | 1056 | 1055 | 1054 | 1053 | 1052 | 1051 | 1050 | 1047 | 1046 | 1045 | 1044 | 1043 | 1042 | 1041 | 1040 | V40642 | |
| 1077 | 1076 | 1075 | 1074 | 1073 | 1072 | 1071 | 1070 | 1067 | 1066 | 1065 | 1064 | 1063 | 1062 | 1061 | 1060 | V40643 | |
| 1117 | 1116 | 1115 | 1114 | 1113 | 1112 | 1111 | 1110 | 1107 | 1106 | 1105 | 1104 | 1103 | 1102 | 1101 | 1100 | V40644 | |
| 1137 | 1136 | 1135 | 1134 | 1133 | 1132 | 1131 | 1130 | 1127 | 1126 | 1125 | 1124 | 1123 | 1122 | 1121 | 1120 | V40645 | |
| 1157 | 1156 | 1155 | 1154 | 1153 | 1152 | 1151 | 1150 | 1147 | 1146 | 1145 | 1144 | 1143 | 1142 | 1141 | 1140 | V40646 | |
| 1177 | 1176 | 1175 | 1174 | 1173 | 1172 | 1171 | 1170 | 1167 | 1166 | 1165 | 1164 | 1163 | 1162 | 1161 | 1160 | V40647 | |
| 1217 | 1216 | 1215 | 1214 | 1213 | 1212 | 1211 | 1210 | 1207 | 1206 | 1205 | 1204 | 1203 | 1202 | 1201 | 1200 | V40650 | |
| 1237 | 1236 | 1235 | 1234 | 1233 | 1232 | 1231 | 1230 | 1227 | 1226 | 1225 | 1224 | 1223 | 1222 | 1221 | 1220 | V40651 | |
| 1257 | 1256 | 1255 | 1254 | 1253 | 1252 | 1251 | 1250 | 1247 | 1246 | 1245 | 1244 | 1243 | 1242 | 1241 | 1240 | V40652 | |
| 1277 | 1276 | 1275 | 1274 | 1273 | 1272 | 1271 | 1270 | 1267 | 1266 | 1265 | 1264 | 1263 | 1262 | 1261 | 1260 | V40653 | |
| 1317 | 1316 | 1315 | 1314 | 1313 | 1312 | 1311 | 1310 | 1307 | 1306 | 1305 | 1304 | 1303 | 1302 | 1301 | 1300 | V40654 | |
| 1337 | 1336 | 1335 | 1334 | 1333 | 1332 | 1331 | 1330 | 1327 | 1326 | 1325 | 1324 | 1323 | 1322 | 1321 | 1320 | V40655 | |
| 1357 | 1356 | 1355 | 1354 | 1353 | 1352 | 1351 | 1350 | 1347 | 1346 | 1345 | 1344 | 1343 | 1342 | 1341 | 1340 | V40656 | |
| 1377 | 1376 | 1375 | 1374 | 1373 | 1372 | 1371 | 1370 | 1367 | 1366 | 1365 | 1364 | 1363 | 1362 | 1361 | 1360 | V40657 | |
| 1417 | 1416 | 1415 | 1414 | 1413 | 1412 | 1411 | 1410 | 1407 | 1406 | 1405 | 1404 | 1403 | 1402 | 1401 | 1400 | V40660 | |
| 1437 | 1436 | 1435 | 1434 | 1433 | 1432 | 1431 | 1430 | 1427 | 1426 | 1425 | 1424 | 1423 | 1422 | 1421 | 1420 | V40661 | |
| 1457 | 1456 | 1455 | 1454 | 1453 | 1452 | 1451 | 1450 | 1447 | 1446 | 1445 | 1444 | 1443 | 1442 | 1441 | 1440 | V40662 | |
| 1477 | 1476 | 1475 | 1474 | 1473 | 1472 | 1471 | 1470 | 1467 | 1466 | 1465 | 1464 | 1463 | 1462 | 1461 | 1460 | V40663 | |
| 1517 | 1516 | 1515 | 1514 | 1513 | 1512 | 1511 | 1510 | 1507 | 1506 | 1505 | 1504 | 1503 | 1502 | 1501 | 1500 | V40664 | |
| 1537 | 1536 | 1535 | 1534 | 1533 | 1532 | 1531 | 1530 | 1527 | 1526 | 1525 | 1524 | 1523 | 1522 | 1521 | 1520 | V40665 | |
| 1557 | 1556 | 1555 | 1554 | 1553 | 1552 | 1551 | 1550 | 1547 | 1546 | 1545 | 1544 | 1543 | 1542 | 1541 | 1540 | V40666 | |
| 1577 | 1576 | 1575 | 1574 | 1573 | 1572 | 1571 | 1570 | 1567 | 1566 | 1565 | 1564 | 1563 | 1562 | 1561 | 1560 | V40667 | |
| 1617 | 1616 | 1615 | 1614 | 1613 | 1612 | 1611 | 1610 | 1607 | 1606 | 1605 | 1604 | 1603 | 1602 | 1601 | 1600 | V40670 | |
| 1637 | 1636 | 1635 | 1634 | 1633 | 1632 | 1631 | 1630 | 1627 | 1626 | 1625 | 1624 | 1623 | 1622 | 1621 | 1620 | V40671 | |
| 1657 | 1656 | 1655 | 1654 | 1653 | 1652 | 1651 | 1650 | 1647 | 1646 | 1645 | 1644 | 1643 | 1642 | 1641 | 1640 | V40672 | |
| 1677 | 1676 | 1675 | 1674 | 1673 | 1672 | 1671 | 1670 | 1667 | 1666 | 1665 | 1664 | 1663 | 1662 | 1661 | 1660 | V40673 | |
| 1717 | 1716 | 1715 | 1714 | 1713 | 1712 | 1711 | 1710 | 1707 | 1706 | 1705 | 1704 | 1703 | 1702 | 1701 | 1700 | V40674 | |
| 1737 | 1736 | 1735 | 1734 | 1733 | 1732 | 1731 | 1730 | 1727 | 1726 | 1725 | 1724 | 1723 | 1722 | 1721 | 1720 | V40675 | |
| 1757 | 1756 | 1755 | 1754 | 1753 | 1752 | 1751 | 1750 | 1747 | 1746 | 1745 | 1744 | 1743 | 1742 | 1741 | 1740 | V40676 | |
| 1777 | 1776 | 1775 | 1774 | 1773 | 1772 | 1771 | 1770 | 1767 | 1766 | 1765 | 1764 | 1763 | 1762 | 1761 | 1760 | V40677 | |

This portion of the table shows additional Control Relays points available with the DL260.

| MSB | Additional DL260 Control Relays (C) | | | | | | | | | | | | | | | LSB | Address |
|------|-------------------------------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|--------|---------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |
| 2017 | 2016 | 2015 | 2014 | 2013 | 2012 | 2011 | 2010 | 2007 | 2006 | 2005 | 2004 | 2003 | 2002 | 2001 | 2000 | V40700 | |
| 2037 | 2036 | 2035 | 2034 | 2033 | 2032 | 2031 | 2030 | 2027 | 2026 | 2025 | 2024 | 2023 | 2022 | 2021 | 2020 | V40701 | |
| 2057 | 2056 | 2055 | 2054 | 2053 | 2052 | 2051 | 2050 | 2047 | 2046 | 2045 | 2044 | 2043 | 2042 | 2041 | 2040 | V40702 | |
| 2077 | 2076 | 2075 | 2074 | 2073 | 2072 | 2071 | 2070 | 2067 | 2066 | 2065 | 2064 | 2063 | 2062 | 2061 | 2060 | V40703 | |
| 2117 | 2116 | 2115 | 2114 | 2113 | 2112 | 2111 | 2110 | 2107 | 2106 | 2105 | 2104 | 2103 | 2102 | 2101 | 2100 | V40704 | |
| 2137 | 2136 | 2135 | 2134 | 2133 | 2132 | 2131 | 2130 | 2127 | 2126 | 2125 | 2124 | 2123 | 2122 | 2121 | 2120 | V40705 | |
| 2157 | 2156 | 2155 | 2154 | 2153 | 2152 | 2151 | 2150 | 2147 | 2146 | 2145 | 2144 | 2143 | 2142 | 2141 | 2140 | V40706 | |
| 2177 | 2176 | 2175 | 2174 | 2173 | 2172 | 2171 | 2170 | 2167 | 2166 | 2165 | 2164 | 2163 | 2162 | 2161 | 2160 | V40707 | |
| 2217 | 2216 | 2215 | 2214 | 2213 | 2212 | 2211 | 2210 | 2207 | 2206 | 2205 | 2204 | 2203 | 2202 | 2201 | 2200 | V40710 | |
| 2237 | 2236 | 2235 | 2234 | 2233 | 2232 | 2231 | 2230 | 2227 | 2226 | 2225 | 2224 | 2223 | 2222 | 2221 | 2220 | V40711 | |
| 2257 | 2256 | 2255 | 2254 | 2253 | 2252 | 2251 | 2250 | 2247 | 2246 | 2245 | 2244 | 2243 | 2242 | 2241 | 2240 | V40712 | |
| 2277 | 2276 | 2275 | 2274 | 2273 | 2272 | 2271 | 2270 | 2267 | 2266 | 2265 | 2264 | 2263 | 2262 | 2261 | 2260 | V40713 | |
| 2317 | 2316 | 2315 | 2314 | 2313 | 2312 | 2311 | 2310 | 2307 | 2306 | 2305 | 2304 | 2303 | 2302 | 2301 | 2300 | V40714 | |
| 2337 | 2336 | 2335 | 2334 | 2333 | 2332 | 2331 | 2330 | 2327 | 2326 | 2325 | 2324 | 2323 | 2322 | 2321 | 2320 | V40715 | |
| 2357 | 2356 | 2355 | 2354 | 2353 | 2352 | 2351 | 2350 | 2347 | 2346 | 2345 | 2344 | 2343 | 2342 | 2341 | 2340 | V40716 | |
| 2377 | 2376 | 2375 | 2374 | 2373 | 2372 | 2371 | 2370 | 2367 | 2366 | 2365 | 2364 | 2363 | 2362 | 2361 | 2360 | V40717 | |
| 2417 | 2416 | 2415 | 2414 | 2413 | 2412 | 2411 | 2410 | 2407 | 2406 | 2405 | 2404 | 2403 | 2402 | 2401 | 2400 | V40720 | |
| 2437 | 2436 | 2435 | 2434 | 2433 | 2432 | 2431 | 2430 | 2427 | 2426 | 2425 | 2424 | 2423 | 2422 | 2421 | 2420 | V40721 | |
| 2457 | 2456 | 2455 | 2454 | 2453 | 2452 | 2451 | 2450 | 2447 | 2446 | 2445 | 2444 | 2443 | 2442 | 2441 | 2440 | V40722 | |
| 2477 | 2476 | 2475 | 2474 | 2473 | 2472 | 2471 | 2470 | 2467 | 2466 | 2465 | 2464 | 2463 | 2462 | 2461 | 2460 | V40723 | |
| 2517 | 2516 | 2515 | 2514 | 2513 | 2512 | 2511 | 2510 | 2507 | 2506 | 2505 | 2504 | 2503 | 2502 | 2501 | 2500 | V40724 | |
| 2537 | 2536 | 2535 | 2534 | 2533 | 2532 | 2531 | 2530 | 2527 | 2526 | 2525 | 2524 | 2523 | 2522 | 2521 | 2520 | V40725 | |
| 2557 | 2556 | 2555 | 2554 | 2553 | 2552 | 2551 | 2550 | 2547 | 2546 | 2545 | 2544 | 2543 | 2542 | 2541 | 2540 | V40726 | |
| 2577 | 2576 | 2575 | 2574 | 2573 | 2572 | 2571 | 2570 | 2567 | 2566 | 2565 | 2564 | 2563 | 2562 | 2561 | 2560 | V40727 | |
| 2617 | 2616 | 2615 | 2614 | 2613 | 2612 | 2611 | 2610 | 2607 | 2606 | 2605 | 2604 | 2603 | 2602 | 2601 | 2600 | V40730 | |
| 2637 | 2636 | 2635 | 2634 | 2633 | 2632 | 2631 | 2630 | 2627 | 2626 | 2625 | 2624 | 2623 | 2622 | 2621 | 2620 | V40731 | |
| 2657 | 2656 | 2655 | 2654 | 2653 | 2652 | 2651 | 2650 | 2647 | 2646 | 2645 | 2644 | 2643 | 2642 | 2641 | 2640 | V40732 | |
| 2677 | 2676 | 2675 | 2674 | 2673 | 2672 | 2671 | 2670 | 2667 | 2666 | 2665 | 2664 | 2663 | 2662 | 2661 | 2660 | V40733 | |
| 2717 | 2716 | 2715 | 2714 | 2713 | 2712 | 2711 | 2710 | 2707 | 2706 | 2705 | 2704 | 2703 | 2702 | 2701 | 2700 | V40734 | |
| 2737 | 2736 | 2735 | 2734 | 2733 | 2732 | 2731 | 2730 | 2727 | 2726 | 2725 | 2724 | 2723 | 2722 | 2721 | 2720 | V40735 | |
| 2757 | 2756 | 2755 | 2754 | 2753 | 2752 | 2751 | 2750 | 2747 | 2746 | 2745 | 2744 | 2743 | 2742 | 2741 | 2740 | V40736 | |
| 2777 | 2776 | 2775 | 2774 | 2773 | 2772 | 2771 | 2770 | 2767 | 2766 | 2765 | 2764 | 2763 | 2762 | 2761 | 2760 | V40737 | |

| MSB | Additional DL260 Control Relays (C) | | | | | | | | | | | | | | | LSB | Address |
|------|-------------------------------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|--------|---------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |
| 3017 | 3016 | 3015 | 3014 | 3013 | 3012 | 3011 | 3010 | 3007 | 3006 | 3005 | 3004 | 3003 | 3002 | 3001 | 3000 | V40740 | |
| 3037 | 3036 | 3035 | 3034 | 3033 | 3032 | 3031 | 3030 | 3027 | 3026 | 3025 | 3024 | 3023 | 3022 | 3021 | 3020 | V40741 | |
| 3057 | 3056 | 3055 | 3054 | 3053 | 3052 | 3051 | 3050 | 3047 | 3046 | 3045 | 3044 | 3043 | 3042 | 3041 | 3040 | V40742 | |
| 3077 | 3076 | 3075 | 3074 | 3073 | 3072 | 3071 | 3070 | 3067 | 3066 | 3065 | 3064 | 3063 | 3062 | 3061 | 3060 | V40743 | |
| 3117 | 3116 | 3115 | 3114 | 3113 | 3112 | 3111 | 3110 | 3107 | 3106 | 3105 | 3104 | 3103 | 3102 | 3101 | 3100 | V40744 | |
| 3137 | 3136 | 3135 | 3134 | 3133 | 3132 | 3131 | 3130 | 3127 | 3126 | 3125 | 3124 | 3123 | 3122 | 3121 | 3120 | V40745 | |
| 3157 | 3156 | 3155 | 3154 | 3153 | 3152 | 3151 | 3150 | 3147 | 3146 | 3145 | 3144 | 3143 | 3142 | 3141 | 3140 | V40746 | |
| 3177 | 3176 | 3175 | 3174 | 3173 | 3172 | 3171 | 3170 | 3167 | 3166 | 3165 | 3164 | 3163 | 3162 | 3161 | 3160 | V40747 | |
| 3217 | 3216 | 3215 | 3214 | 3213 | 3212 | 3211 | 3210 | 3207 | 3206 | 3205 | 3204 | 3203 | 3202 | 3201 | 3200 | V40750 | |
| 3237 | 3236 | 3235 | 3234 | 3233 | 3232 | 3231 | 3230 | 3227 | 3226 | 3225 | 3224 | 3223 | 3222 | 3221 | 3220 | V40751 | |
| 3257 | 3256 | 3255 | 3254 | 3253 | 3252 | 3251 | 3250 | 3247 | 3246 | 3245 | 3244 | 3243 | 3242 | 3241 | 3240 | V40752 | |
| 3277 | 3276 | 3275 | 3274 | 3273 | 3272 | 3271 | 3270 | 3267 | 3266 | 3265 | 3264 | 3263 | 3262 | 3261 | 3260 | V40753 | |
| 3317 | 3316 | 3315 | 3314 | 3313 | 3312 | 3311 | 3310 | 3307 | 3306 | 3305 | 3304 | 3303 | 3302 | 3301 | 3300 | V40754 | |
| 3337 | 3336 | 3335 | 3334 | 3333 | 3332 | 3331 | 3330 | 3327 | 3326 | 3325 | 3324 | 3323 | 3322 | 3321 | 3320 | V40755 | |
| 3357 | 3356 | 3355 | 3354 | 3353 | 3352 | 3351 | 3350 | 3347 | 3346 | 3345 | 3344 | 3343 | 3342 | 3341 | 3340 | V40756 | |
| 3377 | 3376 | 3375 | 3374 | 3373 | 3372 | 3371 | 3370 | 3367 | 3366 | 3365 | 3364 | 3363 | 3362 | 3361 | 3360 | V40757 | |
| 3417 | 3416 | 3415 | 3414 | 3413 | 3412 | 3411 | 3410 | 3407 | 3406 | 3405 | 3404 | 3403 | 3402 | 3401 | 3400 | V40760 | |
| 3437 | 3436 | 3435 | 3434 | 3433 | 3432 | 3431 | 3430 | 3427 | 3426 | 3425 | 3424 | 3423 | 3422 | 3421 | 3420 | V40761 | |
| 3457 | 3456 | 3455 | 3454 | 3453 | 3452 | 3451 | 3450 | 3447 | 3446 | 3445 | 3444 | 3443 | 3442 | 3441 | 3440 | V40762 | |
| 3477 | 3476 | 3475 | 3474 | 3473 | 3472 | 3471 | 3470 | 3467 | 3466 | 3465 | 3464 | 3463 | 3462 | 3461 | 3460 | V40763 | |
| 3517 | 3516 | 3515 | 3514 | 3513 | 3512 | 3511 | 3510 | 3507 | 3506 | 3505 | 3504 | 3503 | 3502 | 3501 | 3500 | V40764 | |
| 3537 | 3536 | 3535 | 3534 | 3533 | 3532 | 3531 | 3530 | 3527 | 3526 | 3525 | 3524 | 3523 | 3522 | 3521 | 3520 | V40765 | |
| 3557 | 3556 | 3555 | 3554 | 3553 | 3552 | 3551 | 3550 | 3547 | 3546 | 3545 | 3544 | 3543 | 3542 | 3541 | 3540 | V40766 | |
| 3577 | 3576 | 3575 | 3574 | 3573 | 3572 | 3571 | 3570 | 3567 | 3566 | 3565 | 3564 | 3563 | 3562 | 3561 | 3560 | V40767 | |
| 3617 | 3616 | 3615 | 3614 | 3613 | 3612 | 3611 | 3610 | 3607 | 3606 | 3605 | 3604 | 3603 | 3602 | 3601 | 3600 | V40770 | |
| 3637 | 3636 | 3635 | 3634 | 3633 | 3632 | 3631 | 3630 | 3627 | 3626 | 3625 | 3624 | 3623 | 3622 | 3621 | 3620 | V40771 | |
| 3657 | 3656 | 3655 | 3654 | 3653 | 3652 | 3651 | 3650 | 3647 | 3646 | 3645 | 3644 | 3643 | 3642 | 3641 | 3640 | V40772 | |
| 3677 | 3676 | 3675 | 3674 | 3673 | 3672 | 3671 | 3670 | 3667 | 3666 | 3665 | 3664 | 3663 | 3662 | 3661 | 3660 | V40773 | |
| 3717 | 3716 | 3715 | 3714 | 3713 | 3712 | 3711 | 3710 | 3707 | 3706 | 3705 | 3704 | 3703 | 3702 | 3701 | 3700 | V40774 | |
| 3737 | 3736 | 3735 | 3734 | 3733 | 3732 | 3731 | 3730 | 3727 | 3726 | 3725 | 3724 | 3723 | 3722 | 3721 | 3720 | V40775 | |
| 3757 | 3756 | 3755 | 3754 | 3753 | 3752 | 3751 | 3750 | 3747 | 3746 | 3745 | 3744 | 3743 | 3742 | 3741 | 3740 | V40776 | |
| 3777 | 3776 | 3775 | 3774 | 3773 | 3772 | 3771 | 3770 | 3767 | 3766 | 3765 | 3764 | 3763 | 3762 | 3761 | 3760 | V40777 | |

Stage Control/Status Bit Map

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

| MSB | DL230/DL240/DL250-1/DL260 Stage (S) Control Bits | | | | | | | | | | | | | | | LSB | Address |
|-----|--|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|--------|---------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |
| 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 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 | |

| MSB | Additional DL240/DL250-1/DL260 Stage (S) Control Bits | | | | | | | | | | | | | | | LSB | Address |
|-----|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|--------|---------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |
| 417 | 416 | 415 | 414 | 413 | 412 | 411 | 410 | 407 | 406 | 405 | 404 | 403 | 402 | 401 | 400 | V41020 | |
| 437 | 436 | 435 | 434 | 433 | 432 | 431 | 430 | 427 | 426 | 425 | 424 | 423 | 422 | 421 | 420 | V41021 | |
| 457 | 456 | 455 | 454 | 453 | 452 | 451 | 450 | 447 | 446 | 445 | 444 | 443 | 442 | 441 | 440 | V41022 | |
| 477 | 476 | 475 | 474 | 473 | 472 | 471 | 470 | 467 | 466 | 465 | 464 | 463 | 462 | 461 | 460 | V41023 | |
| 517 | 516 | 515 | 514 | 513 | 512 | 511 | 510 | 507 | 506 | 505 | 504 | 503 | 502 | 501 | 500 | V41024 | |
| 537 | 536 | 535 | 534 | 533 | 532 | 531 | 530 | 527 | 526 | 525 | 524 | 523 | 522 | 521 | 520 | V41025 | |
| 557 | 556 | 555 | 554 | 553 | 552 | 551 | 550 | 547 | 546 | 545 | 544 | 543 | 542 | 541 | 540 | V41026 | |
| 577 | 576 | 575 | 574 | 573 | 572 | 571 | 570 | 567 | 566 | 565 | 564 | 563 | 562 | 561 | 560 | V41027 | |
| 617 | 616 | 615 | 614 | 613 | 612 | 611 | 610 | 607 | 606 | 605 | 604 | 603 | 602 | 601 | 600 | V41030 | |
| 637 | 636 | 635 | 634 | 633 | 632 | 631 | 630 | 627 | 626 | 625 | 624 | 623 | 622 | 621 | 620 | V41031 | |
| 657 | 656 | 655 | 654 | 653 | 652 | 651 | 650 | 647 | 646 | 645 | 644 | 643 | 642 | 641 | 640 | V41032 | |
| 677 | 676 | 675 | 674 | 673 | 672 | 671 | 670 | 667 | 666 | 665 | 664 | 663 | 662 | 661 | 660 | V41033 | |
| 717 | 716 | 715 | 714 | 713 | 712 | 711 | 710 | 707 | 706 | 705 | 704 | 703 | 702 | 701 | 700 | V41034 | |
| 737 | 736 | 735 | 734 | 733 | 732 | 731 | 730 | 727 | 726 | 725 | 724 | 723 | 722 | 721 | 720 | V41035 | |
| 757 | 756 | 755 | 754 | 753 | 752 | 751 | 750 | 747 | 746 | 745 | 744 | 743 | 742 | 741 | 740 | V41036 | |
| 777 | 776 | 775 | 774 | 773 | 772 | 771 | 770 | 767 | 766 | 765 | 764 | 763 | 762 | 761 | 760 | V41037 | |

| MSB | Additional DL250-1/DL260 Stage (S) Control Bits | | | | | | | | | | | | | | | LSB | Address |
|------|---|------|------|------|------|------|------|------|------|------|------|------|------|------|------|-----|---------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |
| 1017 | 1016 | 1015 | 1014 | 1013 | 1012 | 1011 | 1010 | 1007 | 1006 | 1005 | 1004 | 1003 | 1002 | 1001 | 1000 | | V41040 |
| 1037 | 1036 | 1035 | 1034 | 1033 | 1032 | 1031 | 1030 | 1027 | 1026 | 1025 | 1024 | 1023 | 1022 | 1021 | 1020 | | V41041 |
| 1057 | 1056 | 1055 | 1054 | 1053 | 1052 | 1051 | 1050 | 1047 | 1046 | 1045 | 1044 | 1043 | 1042 | 1041 | 1040 | | V41042 |
| 1077 | 1076 | 1075 | 1074 | 1073 | 1072 | 1071 | 1070 | 1067 | 1066 | 1065 | 1064 | 1063 | 1062 | 1061 | 1060 | | V41043 |
| 1117 | 1116 | 1115 | 1114 | 1113 | 1112 | 1111 | 1110 | 1107 | 1106 | 1105 | 1104 | 1103 | 1102 | 1101 | 1100 | | V41044 |
| 1137 | 1136 | 1135 | 1134 | 1133 | 1132 | 1131 | 1130 | 1127 | 1126 | 1125 | 1124 | 1123 | 1122 | 1121 | 1120 | | V41045 |
| 1157 | 1156 | 1155 | 1154 | 1153 | 1152 | 1151 | 1150 | 1147 | 1146 | 1145 | 1144 | 1143 | 1142 | 1141 | 1140 | | V41046 |
| 1177 | 1176 | 1175 | 1174 | 1173 | 1172 | 1171 | 1170 | 1167 | 1166 | 1165 | 1164 | 1163 | 1162 | 1161 | 1160 | | V41047 |
| 1217 | 1216 | 1215 | 1214 | 1213 | 1212 | 1211 | 1210 | 1207 | 1206 | 1205 | 1204 | 1203 | 1202 | 1201 | 1200 | | V41050 |
| 1237 | 1236 | 1235 | 1234 | 1233 | 1232 | 1231 | 1230 | 1227 | 1226 | 1225 | 1224 | 1223 | 1222 | 1221 | 1220 | | V41051 |
| 1257 | 1256 | 1255 | 1254 | 1253 | 1252 | 1251 | 1250 | 1247 | 1246 | 1245 | 1244 | 1243 | 1242 | 1241 | 1240 | | V41052 |
| 1277 | 1276 | 1275 | 1274 | 1273 | 1272 | 1271 | 1270 | 1267 | 1266 | 1265 | 1264 | 1263 | 1262 | 1261 | 1260 | | V41053 |
| 1317 | 1316 | 1315 | 1314 | 1313 | 1312 | 1311 | 1310 | 1307 | 1306 | 1305 | 1304 | 1303 | 1302 | 1301 | 1300 | | V41054 |
| 1337 | 1336 | 1335 | 1334 | 1333 | 1332 | 1331 | 1330 | 1327 | 1326 | 1325 | 1324 | 1323 | 1322 | 1321 | 1320 | | V41055 |
| 1357 | 1356 | 1355 | 1354 | 1353 | 1352 | 1351 | 1350 | 1347 | 1346 | 1345 | 1344 | 1343 | 1342 | 1341 | 1340 | | V41056 |
| 1377 | 1376 | 1375 | 1374 | 1373 | 1372 | 1371 | 1370 | 1367 | 1366 | 1365 | 1364 | 1363 | 1362 | 1361 | 1360 | | V41057 |
| 1417 | 1416 | 1415 | 1414 | 1413 | 1412 | 1411 | 1410 | 1407 | 1406 | 1405 | 1404 | 1403 | 1402 | 1401 | 1400 | | V41060 |
| 1437 | 1436 | 1435 | 1434 | 1433 | 1432 | 1431 | 1430 | 1427 | 1426 | 1425 | 1424 | 1423 | 1422 | 1421 | 1420 | | V41061 |
| 1457 | 1456 | 1455 | 1454 | 1453 | 1452 | 1451 | 1450 | 1447 | 1446 | 1445 | 1444 | 1443 | 1442 | 1441 | 1440 | | V41062 |
| 1477 | 1476 | 1475 | 1474 | 1473 | 1472 | 1471 | 1470 | 1467 | 1466 | 1465 | 1464 | 1463 | 1462 | 1461 | 1460 | | V41063 |
| 1517 | 1516 | 1515 | 1514 | 1513 | 1512 | 1511 | 1510 | 1507 | 1506 | 1505 | 1504 | 1503 | 1502 | 1501 | 1500 | | V41064 |
| 1537 | 1536 | 1535 | 1534 | 1533 | 1532 | 1531 | 1530 | 1527 | 1526 | 1525 | 1524 | 1523 | 1522 | 1521 | 1520 | | V41065 |
| 1557 | 1556 | 1555 | 1554 | 1553 | 1552 | 1551 | 1550 | 1547 | 1546 | 1545 | 1544 | 1543 | 1542 | 1541 | 1540 | | V41066 |
| 1577 | 1576 | 1575 | 1574 | 1573 | 1572 | 1571 | 1570 | 1567 | 1566 | 1565 | 1564 | 1563 | 1562 | 1561 | 1560 | | V41067 |
| 1617 | 1616 | 1615 | 1614 | 1613 | 1612 | 1611 | 1610 | 1607 | 1606 | 1605 | 1604 | 1603 | 1602 | 1601 | 1600 | | V41070 |
| 1637 | 1636 | 1635 | 1634 | 1633 | 1632 | 1631 | 1630 | 1627 | 1626 | 1625 | 1624 | 1623 | 1622 | 1621 | 1620 | | V41071 |
| 1657 | 1656 | 1655 | 1654 | 1653 | 1652 | 1651 | 1650 | 1647 | 1646 | 1645 | 1644 | 1643 | 1642 | 1641 | 1640 | | V41072 |
| 1677 | 1676 | 1675 | 1674 | 1673 | 1672 | 1671 | 1670 | 1667 | 1666 | 1665 | 1664 | 1663 | 1662 | 1661 | 1660 | | V41073 |
| 1717 | 1716 | 1715 | 1714 | 1713 | 1712 | 1711 | 1710 | 1707 | 1706 | 1705 | 1704 | 1703 | 1702 | 1701 | 1700 | | V41074 |
| 1737 | 1736 | 1735 | 1734 | 1733 | 1732 | 1731 | 1730 | 1727 | 1726 | 1725 | 1724 | 1723 | 1722 | 1721 | 1720 | | V41075 |
| 1757 | 1756 | 1755 | 1754 | 1753 | 1752 | 1751 | 1750 | 1747 | 1746 | 1745 | 1744 | 1743 | 1742 | 1741 | 1740 | | V41076 |
| 1777 | 1776 | 1775 | 1774 | 1773 | 1772 | 1771 | 1770 | 1767 | 1766 | 1765 | 1764 | 1763 | 1762 | 1761 | 1760 | | V41077 |

Timer and Counter Status Bit Maps

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

| MSB | DL230/DL240/DL250-1/DL260 Timer (T) and Counter (CT) Contacts | | | | | | | | | | | | | | | LSB | Timer Address | Counter Address |
|-----|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|---------------|-----------------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | |
| 017 | 016 | 015 | 014 | 013 | 012 | 011 | 010 | 007 | 006 | 005 | 004 | 003 | 002 | 001 | 000 | | V41100 | V41140 |
| 037 | 036 | 035 | 034 | 033 | 032 | 031 | 030 | 027 | 026 | 025 | 024 | 023 | 022 | 021 | 020 | | V41101 | V41141 |
| 057 | 056 | 055 | 054 | 053 | 052 | 051 | 050 | 047 | 046 | 045 | 044 | 043 | 042 | 041 | 040 | | V41102 | V41142 |
| 077 | 076 | 075 | 074 | 073 | 072 | 071 | 070 | 067 | 066 | 065 | 064 | 063 | 062 | 061 | 060 | | V41103 | V41143 |

This portion of the table shows additional Timer and Counter contacts available with the DL240/250–1/260.

| MSB | Additional DL240/DL250-1/DL260 Timer (T) and Counter (CT) Contacts | | | | | | | | | | | | | | | LSB | Timer Address | Counter Address |
|-----|--|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|---------------|-----------------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | |
| 117 | 116 | 115 | 114 | 113 | 112 | 111 | 110 | 107 | 106 | 105 | 104 | 103 | 102 | 101 | 100 | | V41104 | V41144 |
| 137 | 136 | 135 | 134 | 133 | 132 | 131 | 130 | 127 | 126 | 125 | 124 | 123 | 122 | 121 | 120 | | V41105 | V41145 |
| 157 | 156 | 155 | 154 | 153 | 152 | 151 | 150 | 147 | 146 | 145 | 144 | 143 | 142 | 141 | 140 | | V41106 | V41146 |
| 177 | 176 | 175 | 174 | 173 | 172 | 171 | 170 | 167 | 166 | 165 | 164 | 163 | 162 | 161 | 160 | | V41107 | V41147 |

This portion of the table shows additional Timer contacts available with the DL250-1 and DL260.

| MSB | Additional DL250-1/DL260 Timer (T) Contacts | | | | | | | | | | | | | | | LSB | Timer Address |
|-----|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|---------------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |
| 217 | 216 | 215 | 214 | 213 | 212 | 211 | 210 | 207 | 206 | 205 | 204 | 203 | 202 | 201 | 200 | | V41110 |
| 237 | 236 | 235 | 234 | 233 | 232 | 231 | 230 | 227 | 226 | 225 | 224 | 223 | 222 | 221 | 220 | | V41111 |
| 257 | 256 | 255 | 254 | 253 | 252 | 251 | 250 | 247 | 246 | 245 | 244 | 243 | 242 | 241 | 240 | | V41112 |
| 277 | 276 | 275 | 274 | 273 | 272 | 271 | 270 | 267 | 266 | 265 | 264 | 263 | 262 | 261 | 260 | | V41113 |
| 317 | 316 | 315 | 314 | 313 | 312 | 311 | 310 | 307 | 306 | 305 | 304 | 303 | 302 | 301 | 300 | | V41114 |
| 337 | 336 | 335 | 334 | 333 | 332 | 331 | 330 | 327 | 326 | 325 | 324 | 323 | 322 | 321 | 320 | | V41115 |
| 357 | 356 | 355 | 354 | 353 | 352 | 351 | 350 | 347 | 346 | 345 | 344 | 343 | 342 | 341 | 340 | | V41116 |
| 377 | 376 | 375 | 374 | 373 | 372 | 371 | 370 | 367 | 366 | 365 | 364 | 363 | 362 | 361 | 360 | | V41117 |

This portion of the table shows additional Counter contacts available with the DL260.

| MSB | Additional DL260 Counter (CT) Contacts | | | | | | | | | | | | | | | LSB | Counter Address |
|-----|--|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----------------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |
| 217 | 216 | 215 | 214 | 213 | 212 | 211 | 210 | 207 | 206 | 205 | 204 | 203 | 202 | 201 | 200 | | V41150 |
| 237 | 236 | 235 | 234 | 233 | 232 | 231 | 230 | 227 | 226 | 225 | 224 | 223 | 222 | 221 | 220 | | V41151 |
| 257 | 256 | 255 | 254 | 253 | 252 | 251 | 250 | 247 | 246 | 245 | 244 | 243 | 242 | 241 | 240 | | V41152 |
| 277 | 276 | 275 | 274 | 273 | 272 | 271 | 270 | 267 | 266 | 265 | 264 | 263 | 262 | 261 | 260 | | V41153 |
| 317 | 316 | 315 | 314 | 313 | 312 | 311 | 310 | 307 | 306 | 305 | 304 | 303 | 302 | 301 | 300 | | V41154 |
| 337 | 336 | 335 | 334 | 333 | 332 | 331 | 330 | 327 | 326 | 325 | 324 | 323 | 322 | 321 | 320 | | V41155 |
| 357 | 356 | 355 | 354 | 353 | 352 | 351 | 350 | 347 | 346 | 345 | 344 | 343 | 342 | 341 | 340 | | V41156 |
| 377 | 376 | 375 | 374 | 373 | 372 | 371 | 370 | 367 | 366 | 365 | 364 | 363 | 362 | 361 | 360 | | V41157 |

Remote I/O Bit Map

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

| MSB | DL260 Remote I/O (GX) and (GY) Points | | | | | | | | | | | | | | | LSB | GX | GY |
|-----|---------------------------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|---------|---------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | Address | Address |
| 017 | 016 | 015 | 014 | 013 | 012 | 011 | 010 | 007 | 006 | 005 | 004 | 003 | 002 | 001 | 000 | | V40000 | V40200 |
| 037 | 036 | 035 | 034 | 033 | 032 | 031 | 030 | 027 | 026 | 025 | 024 | 023 | 022 | 021 | 020 | | V40001 | V40201 |
| 057 | 056 | 055 | 054 | 053 | 052 | 051 | 050 | 047 | 046 | 045 | 044 | 043 | 042 | 041 | 040 | | V40002 | V40202 |
| 077 | 076 | 075 | 074 | 073 | 072 | 071 | 070 | 067 | 066 | 065 | 064 | 063 | 062 | 061 | 060 | | V40003 | V40203 |
| 117 | 116 | 115 | 114 | 113 | 112 | 111 | 110 | 107 | 106 | 105 | 104 | 103 | 102 | 101 | 100 | | V40004 | V40204 |
| 137 | 136 | 135 | 134 | 133 | 132 | 131 | 130 | 127 | 126 | 125 | 124 | 123 | 122 | 121 | 120 | | V40005 | V40205 |
| 157 | 156 | 155 | 154 | 153 | 152 | 151 | 150 | 147 | 146 | 145 | 144 | 143 | 142 | 141 | 140 | | V40006 | V40206 |
| 177 | 176 | 175 | 174 | 173 | 172 | 171 | 170 | 167 | 166 | 165 | 164 | 163 | 162 | 161 | 160 | | V40007 | V40207 |
| 217 | 216 | 215 | 214 | 213 | 212 | 211 | 210 | 207 | 206 | 205 | 204 | 203 | 202 | 201 | 200 | | V40010 | V40210 |
| 237 | 236 | 235 | 234 | 233 | 232 | 231 | 230 | 227 | 226 | 225 | 224 | 223 | 222 | 221 | 220 | | V40011 | V40211 |
| 257 | 256 | 255 | 254 | 253 | 252 | 251 | 250 | 247 | 246 | 245 | 244 | 243 | 242 | 241 | 240 | | V40012 | V40212 |
| 277 | 276 | 275 | 274 | 273 | 272 | 271 | 270 | 267 | 266 | 265 | 264 | 263 | 262 | 261 | 260 | | V40013 | V40213 |
| 317 | 316 | 315 | 314 | 313 | 312 | 311 | 310 | 307 | 306 | 305 | 304 | 303 | 302 | 301 | 300 | | V40004 | V40214 |
| 337 | 336 | 335 | 334 | 333 | 332 | 331 | 330 | 327 | 326 | 325 | 324 | 323 | 322 | 321 | 320 | | V40015 | V40215 |
| 357 | 356 | 355 | 354 | 353 | 352 | 351 | 350 | 347 | 346 | 345 | 344 | 343 | 342 | 341 | 340 | | V40016 | V40216 |
| 377 | 376 | 375 | 374 | 373 | 372 | 371 | 370 | 367 | 366 | 365 | 364 | 363 | 362 | 361 | 360 | | V40007 | V40217 |
| 417 | 416 | 415 | 414 | 413 | 412 | 411 | 410 | 407 | 406 | 405 | 404 | 403 | 402 | 401 | 400 | | V40020 | V40220 |
| 437 | 436 | 435 | 434 | 433 | 432 | 431 | 430 | 427 | 426 | 425 | 424 | 423 | 422 | 421 | 420 | | V40021 | V40221 |
| 457 | 456 | 455 | 454 | 453 | 452 | 451 | 450 | 447 | 446 | 445 | 444 | 443 | 442 | 441 | 440 | | V40022 | V40222 |
| 477 | 476 | 475 | 474 | 473 | 472 | 471 | 470 | 467 | 466 | 465 | 464 | 463 | 462 | 461 | 460 | | V40023 | V40223 |
| 517 | 516 | 515 | 514 | 513 | 512 | 511 | 510 | 507 | 506 | 505 | 504 | 503 | 502 | 501 | 500 | | V40024 | V40224 |
| 537 | 536 | 535 | 534 | 533 | 532 | 531 | 530 | 527 | 526 | 525 | 524 | 523 | 522 | 521 | 520 | | V40025 | V40225 |
| 557 | 556 | 555 | 554 | 553 | 552 | 551 | 550 | 547 | 546 | 545 | 544 | 543 | 542 | 541 | 540 | | V40026 | V40226 |
| 577 | 576 | 575 | 574 | 573 | 572 | 571 | 570 | 567 | 566 | 565 | 564 | 563 | 562 | 561 | 560 | | V40027 | V40227 |
| 617 | 616 | 615 | 614 | 613 | 612 | 611 | 610 | 607 | 606 | 605 | 604 | 603 | 602 | 601 | 600 | | V40030 | V40230 |
| 637 | 636 | 635 | 634 | 633 | 632 | 631 | 630 | 627 | 626 | 625 | 624 | 623 | 622 | 621 | 620 | | V40031 | V40231 |
| 657 | 656 | 655 | 654 | 653 | 652 | 651 | 650 | 647 | 646 | 645 | 644 | 643 | 642 | 641 | 640 | | V40032 | V40232 |
| 677 | 676 | 675 | 674 | 673 | 672 | 671 | 670 | 667 | 666 | 665 | 664 | 663 | 662 | 661 | 660 | | V40033 | V40233 |
| 717 | 716 | 715 | 714 | 713 | 712 | 711 | 710 | 707 | 706 | 705 | 704 | 703 | 702 | 701 | 700 | | V40034 | V40234 |
| 737 | 736 | 735 | 734 | 733 | 732 | 731 | 730 | 727 | 726 | 725 | 724 | 723 | 722 | 721 | 720 | | V40035 | V40235 |
| 757 | 756 | 755 | 754 | 753 | 752 | 751 | 750 | 747 | 746 | 745 | 744 | 743 | 742 | 741 | 740 | | V40036 | V40236 |
| 777 | 776 | 775 | 774 | 773 | 772 | 771 | 770 | 767 | 766 | 765 | 764 | 763 | 762 | 761 | 760 | | V40037 | V40237 |

| MSB | DL260 Remote I/O (GX) and (GY) Points | | | | | | | | | | | | | | | LSB | GX | GY |
|------|---------------------------------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|-----|---------|---------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | Address | Address |
| 1017 | 1016 | 1015 | 1014 | 1013 | 1012 | 1011 | 1010 | 1007 | 1006 | 1005 | 1004 | 1003 | 1002 | 1001 | 1000 | | V40040 | V40240 |
| 1037 | 1036 | 1035 | 1034 | 1033 | 1032 | 1031 | 1030 | 1027 | 1026 | 1025 | 1024 | 1023 | 1022 | 1021 | 1020 | | V40041 | V40241 |
| 1057 | 1056 | 1055 | 1054 | 1053 | 1052 | 1051 | 1050 | 1047 | 1046 | 1045 | 1044 | 1043 | 1042 | 1041 | 1040 | | V40042 | V40242 |
| 1077 | 1076 | 1075 | 1074 | 1073 | 1072 | 1071 | 1070 | 1067 | 1066 | 1065 | 1064 | 1063 | 1062 | 1061 | 1060 | | V40043 | V40243 |
| 1117 | 1116 | 1115 | 1114 | 1113 | 1112 | 1111 | 1110 | 1107 | 1106 | 1105 | 1104 | 1103 | 1102 | 1101 | 1100 | | V40044 | V40244 |
| 1137 | 1136 | 1135 | 1134 | 1133 | 1132 | 1131 | 1130 | 1127 | 1126 | 1125 | 1124 | 1123 | 1122 | 1121 | 1120 | | V40045 | V40245 |
| 1157 | 1156 | 1155 | 1154 | 1153 | 1152 | 1151 | 1150 | 1147 | 1146 | 1145 | 1144 | 1143 | 1142 | 1141 | 1140 | | V40046 | V40246 |
| 1177 | 1176 | 1175 | 1174 | 1173 | 1172 | 1171 | 1170 | 1167 | 1166 | 1165 | 1164 | 1163 | 1162 | 1161 | 1160 | | V40047 | V40247 |
| 1217 | 1216 | 1215 | 1214 | 1213 | 1212 | 1211 | 1210 | 1207 | 1206 | 1205 | 1204 | 1203 | 1202 | 1201 | 1200 | | V40050 | V40250 |
| 1237 | 1236 | 1235 | 1234 | 1233 | 1232 | 1231 | 1230 | 1227 | 1226 | 1225 | 1224 | 1223 | 1222 | 1221 | 1220 | | V40051 | V40251 |
| 1257 | 1256 | 1255 | 1254 | 1253 | 1252 | 1251 | 1250 | 1247 | 1246 | 1245 | 1244 | 1243 | 1242 | 1241 | 1240 | | V40052 | V40252 |
| 1277 | 1276 | 1275 | 1274 | 1273 | 1272 | 1271 | 1270 | 1267 | 1266 | 1265 | 1264 | 1263 | 1262 | 1261 | 1260 | | V40053 | V40253 |
| 1317 | 1316 | 1315 | 1314 | 1313 | 1312 | 1311 | 1310 | 1307 | 1306 | 1305 | 1304 | 1303 | 1302 | 1301 | 1300 | | V40054 | V40254 |
| 1337 | 1336 | 1335 | 1334 | 1333 | 1332 | 1331 | 1330 | 1327 | 1326 | 1325 | 1324 | 1323 | 1322 | 1321 | 1320 | | V40055 | V40255 |
| 1357 | 1356 | 1355 | 1354 | 1353 | 1352 | 1351 | 1350 | 1347 | 1346 | 1345 | 1344 | 1343 | 1342 | 1341 | 1340 | | V40056 | V40256 |
| 1377 | 1376 | 1375 | 1374 | 1373 | 1372 | 1371 | 1370 | 1367 | 1366 | 1365 | 1364 | 1363 | 1362 | 1361 | 1360 | | V40057 | V40257 |
| 1417 | 1416 | 1415 | 1414 | 1413 | 1412 | 1411 | 1410 | 1407 | 1406 | 1405 | 1404 | 1403 | 1402 | 1401 | 1400 | | V40060 | V40260 |
| 1437 | 1436 | 1435 | 1434 | 1433 | 1432 | 1431 | 1430 | 1427 | 1426 | 1425 | 1424 | 1423 | 1422 | 1421 | 1420 | | V40061 | V40261 |
| 1457 | 1456 | 1455 | 1454 | 1453 | 1452 | 1451 | 1450 | 1447 | 1446 | 1445 | 1444 | 1443 | 1442 | 1441 | 1440 | | V40062 | V40262 |
| 1477 | 1476 | 1475 | 1474 | 1473 | 1472 | 1471 | 1470 | 1467 | 1466 | 1465 | 1464 | 1463 | 1462 | 1461 | 1460 | | V40063 | V40263 |
| 1517 | 1516 | 1515 | 1514 | 1513 | 1512 | 1511 | 1510 | 1507 | 1506 | 1505 | 1504 | 1503 | 1502 | 1501 | 1500 | | V40064 | V40264 |
| 1537 | 1536 | 1535 | 1534 | 1533 | 1532 | 1531 | 1530 | 1527 | 1526 | 1525 | 1524 | 1523 | 1522 | 1521 | 1520 | | V40065 | V40265 |
| 1557 | 1556 | 1555 | 1554 | 1553 | 1552 | 1551 | 1550 | 1547 | 1546 | 1545 | 1544 | 1543 | 1542 | 1541 | 1540 | | V40066 | V40266 |
| 1577 | 1576 | 1575 | 1574 | 1573 | 1572 | 1571 | 1570 | 1567 | 1566 | 1565 | 1564 | 1563 | 1562 | 1561 | 1560 | | V40067 | V40267 |
| 1617 | 1616 | 1615 | 1614 | 1613 | 1612 | 1611 | 1610 | 1607 | 1606 | 1605 | 1604 | 1603 | 1602 | 1601 | 1600 | | V40070 | V40270 |
| 1637 | 1636 | 1635 | 1634 | 1633 | 1632 | 1631 | 1630 | 1627 | 1626 | 1625 | 1624 | 1623 | 1622 | 1621 | 1620 | | V40071 | V40271 |
| 1657 | 1656 | 1655 | 1654 | 1653 | 1652 | 1651 | 1650 | 1647 | 1646 | 1645 | 1644 | 1643 | 1642 | 1641 | 1640 | | V40072 | V40272 |
| 1677 | 1676 | 1675 | 1674 | 1673 | 1672 | 1671 | 1670 | 1667 | 1666 | 1665 | 1664 | 1663 | 1662 | 1661 | 1660 | | V40073 | V40273 |
| 1717 | 1716 | 1715 | 1714 | 1713 | 1712 | 1711 | 1710 | 1707 | 1706 | 1705 | 1704 | 1703 | 1702 | 1701 | 1700 | | V40074 | V40274 |
| 1737 | 1736 | 1735 | 1734 | 1733 | 1732 | 1731 | 1730 | 1727 | 1726 | 1725 | 1724 | 1723 | 1722 | 1721 | 1720 | | V40075 | V40275 |
| 1757 | 1756 | 1755 | 1754 | 1753 | 1752 | 1751 | 1750 | 1747 | 1746 | 1745 | 1744 | 1743 | 1742 | 1741 | 1740 | | V40076 | V40276 |
| 1777 | 1776 | 1775 | 1774 | 1773 | 1772 | 1771 | 1770 | 1767 | 1766 | 1765 | 1764 | 1763 | 1762 | 1761 | 1760 | | V40077 | V40277 |

| MSB | DL260 Remote I/O (GX) and (GY) Points | | | | | | | | | | | | | | LSB | GX | GY |
|------|---------------------------------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|---------|---------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Address | Address |
| 2017 | 2016 | 2015 | 2014 | 2013 | 2012 | 2011 | 2010 | 2007 | 2006 | 2005 | 2004 | 2003 | 2002 | 2001 | 2000 | V40100 | V40300 |
| 2037 | 2036 | 2035 | 2034 | 2033 | 2032 | 2031 | 2030 | 2027 | 2026 | 2025 | 2024 | 2023 | 2022 | 2021 | 2020 | V40101 | V40301 |
| 2057 | 2056 | 2055 | 2054 | 2053 | 2052 | 2051 | 2050 | 2047 | 2046 | 2045 | 2044 | 2043 | 2042 | 2041 | 2040 | V40102 | V40302 |
| 2077 | 2076 | 2075 | 2074 | 2073 | 2072 | 2071 | 2070 | 2067 | 2066 | 2065 | 2064 | 2063 | 2062 | 2061 | 2060 | V40103 | V40303 |
| 2117 | 2116 | 2115 | 2114 | 2113 | 2112 | 2111 | 2110 | 2107 | 2106 | 2105 | 2104 | 2103 | 2102 | 2101 | 2100 | V40104 | V40304 |
| 2137 | 2136 | 2135 | 2134 | 2133 | 2132 | 2131 | 2130 | 2127 | 2126 | 2125 | 2124 | 2123 | 2122 | 2121 | 2120 | V40105 | V40305 |
| 2157 | 2156 | 2155 | 2154 | 2153 | 2152 | 2151 | 2150 | 2147 | 2146 | 2145 | 2144 | 2143 | 2142 | 2141 | 2140 | V40106 | V40306 |
| 2177 | 2176 | 2175 | 2174 | 2173 | 2172 | 2171 | 2170 | 2167 | 2166 | 2165 | 2164 | 2163 | 2162 | 2161 | 2160 | V40107 | V40307 |
| 2217 | 2216 | 2215 | 2214 | 2213 | 2212 | 2211 | 2210 | 2207 | 2206 | 2205 | 2204 | 2203 | 2202 | 2201 | 2200 | V40110 | V40310 |
| 2237 | 2236 | 2235 | 2234 | 2233 | 2232 | 2231 | 2230 | 2227 | 2226 | 2225 | 2224 | 2223 | 2222 | 2221 | 2220 | V40111 | V40311 |
| 2257 | 2256 | 2255 | 2254 | 2253 | 2252 | 2251 | 2250 | 2247 | 2246 | 2245 | 2244 | 2243 | 2242 | 2241 | 2240 | V40112 | V40312 |
| 2277 | 2276 | 2275 | 2274 | 2273 | 2272 | 2271 | 2270 | 2267 | 2266 | 2265 | 2264 | 2263 | 2262 | 2261 | 2260 | V40113 | V40313 |
| 2317 | 2316 | 2315 | 2314 | 2313 | 2312 | 2311 | 2310 | 2307 | 2306 | 2305 | 2304 | 2303 | 2302 | 2301 | 2300 | V40114 | V40314 |
| 2337 | 2336 | 2335 | 2334 | 2333 | 2332 | 2331 | 2330 | 2327 | 2326 | 2325 | 2324 | 2323 | 2322 | 2321 | 2320 | V40115 | V40315 |
| 2357 | 2356 | 2355 | 2354 | 2353 | 2352 | 2351 | 2350 | 2347 | 2346 | 2345 | 2344 | 2343 | 2342 | 2341 | 2340 | V40116 | V40316 |
| 2377 | 2376 | 2375 | 2374 | 2373 | 2372 | 2371 | 2370 | 2367 | 2366 | 2365 | 2364 | 2363 | 2362 | 2361 | 2360 | V40117 | V40317 |
| 2417 | 2416 | 2415 | 2414 | 2413 | 2412 | 2411 | 2410 | 2407 | 2406 | 2405 | 2404 | 2403 | 2402 | 2401 | 2400 | V40120 | V40320 |
| 2437 | 2436 | 2435 | 2434 | 2433 | 2432 | 2431 | 2430 | 2427 | 2426 | 2425 | 2424 | 2423 | 2422 | 2421 | 2420 | V40121 | V40321 |
| 2457 | 2456 | 2455 | 2454 | 2453 | 2452 | 2451 | 2450 | 2447 | 2446 | 2445 | 2444 | 2443 | 2442 | 2441 | 2440 | V40122 | V40322 |
| 2477 | 2476 | 2475 | 2474 | 2473 | 2472 | 2471 | 2470 | 2467 | 2466 | 2465 | 2464 | 2463 | 2462 | 2461 | 2460 | V40123 | V40323 |
| 2517 | 2516 | 2515 | 2514 | 2513 | 2512 | 2511 | 2510 | 2507 | 2506 | 2505 | 2504 | 2503 | 2502 | 2501 | 2500 | V40124 | V40324 |
| 2537 | 2536 | 2535 | 2534 | 2533 | 2532 | 2531 | 2530 | 2527 | 2526 | 2525 | 2524 | 2523 | 2522 | 2521 | 2520 | V40125 | V40325 |
| 2557 | 2556 | 2555 | 2554 | 2553 | 2552 | 2551 | 2550 | 2547 | 2546 | 2545 | 2544 | 2543 | 2542 | 2541 | 2540 | V40126 | V40326 |
| 2577 | 2576 | 2575 | 2574 | 2573 | 2572 | 2571 | 2570 | 2567 | 2566 | 2565 | 2564 | 2563 | 2562 | 2561 | 2560 | V40127 | V40327 |
| 2617 | 2616 | 2615 | 2614 | 2613 | 2612 | 2611 | 2610 | 2607 | 2606 | 2605 | 2604 | 2603 | 2602 | 2601 | 2600 | V40130 | V40330 |
| 2637 | 2636 | 2635 | 2634 | 2633 | 2632 | 2631 | 2630 | 2627 | 2626 | 2625 | 2624 | 2623 | 2622 | 2621 | 2620 | V40131 | V40331 |
| 2657 | 2656 | 2655 | 2654 | 2653 | 2652 | 2651 | 2650 | 2647 | 2646 | 2645 | 2644 | 2643 | 2642 | 2641 | 2640 | V40132 | V40332 |
| 2677 | 2676 | 2675 | 2674 | 2673 | 2672 | 2671 | 2670 | 2667 | 2666 | 2665 | 2664 | 2663 | 2662 | 2661 | 2660 | V40133 | V40333 |
| 2717 | 2716 | 2715 | 2714 | 2713 | 2712 | 2711 | 2710 | 2707 | 2706 | 2705 | 2704 | 2703 | 2702 | 2701 | 2700 | V40134 | V40334 |
| 2737 | 2736 | 2735 | 2734 | 2733 | 2732 | 2731 | 2730 | 2727 | 2726 | 2725 | 2724 | 2723 | 2722 | 2721 | 2720 | V40135 | V40335 |
| 2757 | 2756 | 2755 | 2754 | 2753 | 2752 | 2751 | 2750 | 2747 | 2736 | 2735 | 2734 | 2733 | 2732 | 2731 | 2730 | V40136 | V40336 |
| 2777 | 2776 | 2775 | 2774 | 2773 | 2772 | 2771 | 2770 | 2767 | 2766 | 2765 | 2764 | 2763 | 2762 | 2761 | 2760 | V40137 | V40337 |

| MSB | DL260 Remote I/O (GX) and (GY) Points | | | | | | | | | | | | | | LSB | GX | GY |
|------|---------------------------------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|---------|---------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Address | Address |
| 3017 | 3016 | 3015 | 3014 | 3013 | 3012 | 3011 | 3010 | 3007 | 3006 | 3005 | 3004 | 3003 | 3002 | 3001 | 3000 | V40140 | V40340 |
| 3037 | 3036 | 3035 | 3034 | 3033 | 3032 | 3031 | 3030 | 3027 | 3026 | 3025 | 3024 | 3023 | 3022 | 3021 | 3020 | V40141 | V40341 |
| 3057 | 3056 | 3055 | 3054 | 3053 | 3052 | 3051 | 3050 | 3047 | 3046 | 3045 | 3044 | 3043 | 3042 | 3041 | 3040 | V40142 | V40342 |
| 3077 | 3076 | 3075 | 3074 | 3073 | 3072 | 3071 | 3070 | 3067 | 3066 | 3065 | 3064 | 3063 | 3062 | 3061 | 3060 | V40143 | V40343 |
| 3117 | 3116 | 3115 | 3114 | 3113 | 3112 | 3111 | 3110 | 3107 | 3106 | 3105 | 3104 | 3103 | 3102 | 3101 | 3100 | V40144 | V40344 |
| 3137 | 3136 | 3135 | 3134 | 3133 | 3132 | 3131 | 3130 | 3127 | 3126 | 3125 | 3124 | 3123 | 3122 | 3121 | 3120 | V40145 | V40345 |
| 3157 | 3156 | 3155 | 3154 | 3153 | 3152 | 3151 | 3150 | 3147 | 3146 | 3145 | 3144 | 3143 | 3142 | 3141 | 3140 | V40146 | V40346 |
| 3177 | 3176 | 3175 | 3174 | 3173 | 3172 | 3171 | 3170 | 3167 | 3166 | 3165 | 3164 | 3163 | 3162 | 3161 | 3160 | V40147 | V40347 |
| 3217 | 3216 | 3215 | 3214 | 3213 | 3212 | 3211 | 3210 | 3207 | 3206 | 3205 | 3204 | 3203 | 3202 | 3201 | 3200 | V40150 | V40350 |
| 3237 | 3236 | 3235 | 3234 | 3233 | 3232 | 3231 | 3230 | 3227 | 3226 | 3225 | 3224 | 3223 | 3222 | 3221 | 3220 | V40151 | V40351 |
| 3257 | 3256 | 3255 | 3254 | 3253 | 3252 | 3251 | 3250 | 3247 | 3246 | 3245 | 3244 | 3243 | 3242 | 3241 | 3240 | V40152 | V40352 |
| 3277 | 3276 | 3275 | 3274 | 3273 | 3272 | 3271 | 3270 | 3267 | 3266 | 3265 | 3264 | 3263 | 3262 | 3261 | 3260 | V40153 | V40353 |
| 3317 | 3316 | 3315 | 3314 | 3313 | 3312 | 3311 | 3310 | 3307 | 3306 | 3305 | 3304 | 3303 | 3302 | 3301 | 3300 | V40154 | V40354 |
| 3337 | 3336 | 3335 | 3334 | 3333 | 3332 | 3331 | 3330 | 3327 | 3326 | 3325 | 3324 | 3323 | 3322 | 3321 | 3320 | V40155 | V40355 |
| 3357 | 3356 | 3355 | 3354 | 3353 | 3352 | 3351 | 3350 | 3347 | 3346 | 3345 | 3344 | 3343 | 3342 | 3341 | 3340 | V40156 | V40356 |
| 3377 | 3376 | 3375 | 3374 | 3373 | 3372 | 3371 | 3370 | 3367 | 3366 | 3365 | 3364 | 3363 | 3362 | 3361 | 3360 | V40157 | V40357 |
| 3417 | 3416 | 3415 | 3414 | 3413 | 3412 | 3411 | 3410 | 3407 | 3406 | 3405 | 3404 | 3403 | 3402 | 3401 | 3400 | V40160 | V40360 |
| 3437 | 3436 | 3435 | 3434 | 3433 | 3432 | 3431 | 3430 | 3427 | 3426 | 3425 | 3424 | 3423 | 3422 | 3421 | 3420 | V40161 | V40361 |
| 3457 | 3456 | 3455 | 3454 | 3453 | 3452 | 3451 | 3450 | 3447 | 3446 | 3445 | 3444 | 3443 | 3442 | 3441 | 3440 | V40162 | V40362 |
| 3477 | 3476 | 3475 | 3474 | 3473 | 3472 | 3471 | 3470 | 3467 | 3466 | 3465 | 3464 | 3463 | 3462 | 3461 | 3460 | V40163 | V40363 |
| 3517 | 3516 | 3515 | 3514 | 3513 | 3512 | 3511 | 3510 | 3507 | 3506 | 3505 | 3504 | 3503 | 3502 | 3501 | 3500 | V40164 | V40364 |
| 3537 | 3536 | 3535 | 3534 | 3533 | 3532 | 3531 | 3530 | 3527 | 3526 | 3525 | 3524 | 3523 | 3522 | 3521 | 3520 | V40165 | V40365 |
| 3557 | 3556 | 3555 | 3554 | 3553 | 3552 | 3551 | 3550 | 3547 | 3546 | 3545 | 3544 | 3543 | 3542 | 3541 | 3540 | V40166 | V40366 |
| 3577 | 3576 | 3575 | 3574 | 3573 | 3572 | 3571 | 3570 | 3567 | 3566 | 3565 | 3564 | 3563 | 3562 | 3561 | 3560 | V40167 | V40367 |
| 3617 | 3616 | 3615 | 3614 | 3613 | 3612 | 3611 | 3610 | 3607 | 3606 | 3605 | 3604 | 3603 | 3602 | 3601 | 3600 | V40170 | V40370 |
| 3637 | 3636 | 3635 | 3634 | 3633 | 3632 | 3631 | 3630 | 3627 | 3626 | 3625 | 3624 | 3623 | 3622 | 3621 | 3620 | V40171 | V40371 |
| 3657 | 3656 | 3655 | 3654 | 3653 | 3652 | 3651 | 3650 | 3647 | 3646 | 3645 | 3644 | 3643 | 3642 | 3641 | 3640 | V40172 | V40372 |
| 3677 | 3676 | 3675 | 3674 | 3673 | 3672 | 3671 | 3670 | 3667 | 3666 | 3665 | 3664 | 3663 | 3662 | 3661 | 3660 | V40173 | V40373 |
| 3717 | 3716 | 3715 | 3714 | 3713 | 3712 | 3711 | 3710 | 3707 | 3706 | 3705 | 3704 | 3703 | 3702 | 3701 | 3700 | V40174 | V40374 |
| 3737 | 3736 | 3735 | 3734 | 3733 | 3732 | 3731 | 3730 | 3727 | 3726 | 3725 | 3724 | 3723 | 3722 | 3721 | 3720 | V40175 | V40375 |
| 3757 | 3756 | 3755 | 3754 | 3753 | 3752 | 3751 | 3750 | 3747 | 3746 | 3745 | 3744 | 3743 | 3742 | 3741 | 3740 | V40176 | V40376 |
| 3777 | 3776 | 3775 | 3774 | 3773 | 3772 | 3771 | 3770 | 3767 | 3766 | 3765 | 3764 | 3763 | 3762 | 3761 | 3760 | V40177 | V40377 |

Notes

SYSTEM DESIGN AND CONFIGURATION



CHAPTER 4

In This Chapter:

| | |
|---|------|
| DL205 System Design Strategies | 4-2 |
| Module Placement | 4-3 |
| Calculating the Power Budget..... | 4-7 |
| Local Expansion I/O | 4-11 |
| Expanding DL205 I/O | 4-17 |
| Network Connections to Modbus and DirectNet | 4-32 |
| Network Slave Operation | 4-35 |
| Network Modbus RTU Master Operation (DL260 only)..... | 4-45 |
| Non-Sequence Protocol (ASCII In/Out and PRINT) | 4-54 |

DL205 System Design Strategies

I/O System Configurations

The DL205 PLCs offer the following ways to add I/O to the system:

- **Local I/O** – consists of I/O modules located in the same base as the CPU.
- **Local Expansion I/O** – consists of I/O modules in expansion bases located close to the CPU local base. Expansion cables connect the expansion bases and CPU base in daisy-chain format.
- **Ethernet Remote Master** – provides a low-cost, high-speed Ethernet Remote I/O link to Ethernet Remote Slave I/O.
- **Ethernet Base Controller** – provides a low-cost, high-speed Ethernet link between a network master to AutomationDirect Ethernet Remote Slave I/O.
- **Remote I/O** – consists of I/O modules located in bases which are serially connected to the local CPU base through a Remote Master module, or may connect directly to the bottom port on a DL250–1 or DL260 CPU.

A DL205 system can be developed using many different arrangements of these configurations. All I/O configurations use the standard complement of DL205 I/O modules and bases. Local expansion requires using (–1) bases.

Networking Configurations

The DL205 PLCs offers the following way to add networking to the system:

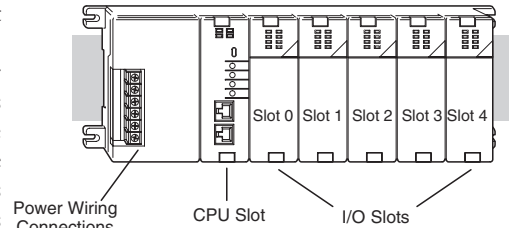
- **Ethernet Communications Module** – connects DL205 systems (DL240, DL250–1 or DL260 CPUs only) and DL405 CPU systems in high-speed, peer-to-peer networks. Any PLC can initiate communications with any other PLC when using either the ECOM or ECOM100 modules.
- **Data Communications Module** – connects a DL205 (DL240, DL250–1 and DL260 only) system to devices using the *DirectNET* protocol, or connects as a slave to a Modbus RTU network.
- **DL250–1 Communications Port** – The DL250–1 CPU has a 15–Pin connector on Port 2 that provides a built-in Modbus RTU or *DirectNET* master/slave connection.
- **DL260 Communications Port** – The DL260 CPU has a 15–Pin connector on Port 2 that provides a built-in *DirectNET* master/slave or Modbus RTU master/slave connection with more Modbus function codes than the DL250–1. (The DL260 MRX and MWX instructions allow you to enter native Modbus addressing in your ladder program with no need to perform octal to decimal conversions.) Port 2 can also be used for ASCII IN or ASCII OUT communications.

| Module/Unit | Master | Slave |
|-------------|--------------------------------------|--|
| DL240 CPU | | <i>DirectNet</i> , K–Sequence |
| DL250–1 CPU | <i>DirectNet</i> , Modbus RTU | <i>DirectNet</i> , K–Sequence, Modbus RTU |
| DL260 CPU | <i>DirectNet</i> , Modbus RTU, ASCII | <i>DirectNet</i> , K–Sequence, Modbus RTU, ASCII |
| ECOM | Ethernet | Ethernet |
| ECOM100 | Ethernet, Modbus TCP | Ethernet, Modbus TCP |
| DCM | <i>DirectNet</i> | <i>DirectNet</i> , K–Sequence, Modbus RTU |

Module Placement

Slot Numbering

The DL205 bases each provide different numbers of slots for use with the I/O modules. You may notice the bases refer to 3-slot, 4-slot, etc. One of the slots is dedicated to the CPU, so you always have one less I/O slot. For example, you have five I/O slots with a 6-slot base. The I/O slots are numbered 0 – 4. The CPU slot always contains a CPU or a base controller (EBC) or Remote Slave and is not numbered.



Module Placement Restrictions

The following table lists the valid locations for all types of modules in a DL205 system.

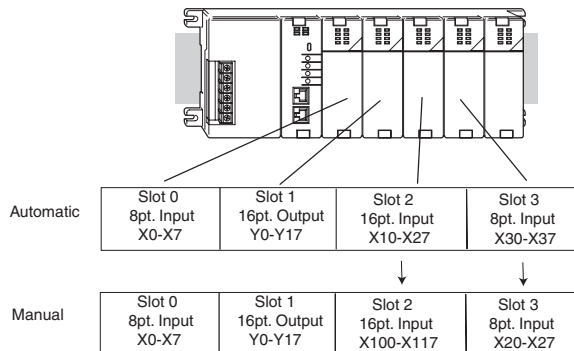
| Module/Unit | Local CPU Base | Local Expansion Base | Remote I/O Base |
|--|----------------|----------------------|-----------------|
| CPUs | CPU Slot Only | | |
| DC Input Modules | √ | √ | √ |
| AC Input Modules | √ | √ | √ |
| DC Output Modules | √ | √ | √ |
| AC Output Modules | √ | √ | √ |
| Relay Output Modules | √ | √ | √ |
| Analog Input and Output Modules | √ | √ | √ |
| Local Expansion | | | |
| Base Expansion Unit | √ | √ | |
| Base Controller Module | | CPU Slot Only | |
| Serial Remote I/O | | | |
| Remote Master | √ (not Slot 0) | | |
| Remote Slave Unit | | | CPU Slot Only |
| Ethernet Remote Master | √ (not Slot 0) | | |
| Ethernet Slave (EBC) | CPU Slot Only | | |
| CPU Interface | | | |
| Ethernet Base Controller | CPU Slot Only | | CPU Slot Only* |
| WinPLC | CPU Slot Only | | |
| DeviceNet | CPU Slot Only | | √ |
| Profibus | CPU Slot Only | | |
| SDS | CPU Slot Only | | |
| Specialty Modules | | | |
| Counter Interface (CTRINT) | Slot 0 Only | | |
| Counter I/O (CTRIO) | √ | | √ * |
| Data Communications | √ (not Slot 0) | | |
| Ethernet Communications | √ (not Slot 0) | | |
| BASIC CoProcessor | √ (not Slot 0) | | |
| Simulator | √ | √ | √ |
| Filler | √ | √ | √ |

*When used in H2-ERM(100) Ethernet Remote I/O systems.

Automatic I/O Configuration

- ✓ 230 The DL205 CPUs automatically detect any installed I/O modules (including specialty modules) at powerup, and establish the correct I/O configuration and addresses. This applies to modules located in local and local expansion I/O bases. For most applications, you will never have to change the configuration.
- ✓ 240
- ✓ 250-1
- ✓ 260 I/O addresses use octal numbering, starting at X0 and Y0 in the slot next to the CPU. The addresses are assigned in groups of 8 or 16, depending on the number of points for the I/O module. The discrete input and output modules can be mixed in any order, but there may be restrictions placed on some specialty modules. The following diagram shows the I/O numbering convention for an example system.

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



Manual I/O Configuration

- ✗ 230 It may never become necessary, but DL250-1 and DL260 CPUs allow manual I/O address assignments for any I/O slot(s) in local or local expansion bases. You can manually modify an auto configuration to match arbitrary I/O numbering. For example, two adjacent input modules can have starting addresses at X20 and X200. Use *DirectSOFT* PLC Configure I/O menu option to assign manual I/O address.
- ✗ 240
- ✓ 250-1
- ✓ 260

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



WARNING: If you manually configure an I/O slot, the I/O addressing for the other modules may change. This is because the DL250-1 and DL260 CPUs do not allow you to assign duplicate I/O addresses. You must always correct any I/O configuration errors before you place the CPU in RUN mode. Uncorrected errors can cause unpredictable machine operation that can result in a risk of personal injury or damage to equipment.

Removing a Manual Configuration

After a manual configuration, the system will automatically retain the new I/O addresses through a power cycle. You can remove (overwrite) any manual configuration changes by changing all of the manually configured addresses back to automatic.

Power-On I/O Configuration Check

The DL205 CPUs can also be set to automatically check the I/O configuration on power-up. By selecting this feature, you can detect any changes that may have occurred while the power was disconnected. For example, if someone places an output module in a slot that previously held an input module, the CPU will not go into RUN mode and the configuration check will detect the change and print a message on the Handheld Programmer or *DirectSOFT* screen (use AUX 44 on the HPP to enable the configuration check).

If the system detects a change in the PLC/Setup/I/O configuration check at power-up, error code E252 will be generated. You can use AUX 42 (HPP) or *DirectSOFT* I/O diagnostics to determine the exact base and slot location where the change occurred. When a configuration error is generated, you may actually want to use the new I/O configuration. For example, you may have intentionally changed an I/O module to use with a program change. You can use PLC/Diagnostics/I/O Diagnostics in *DirectSoft* or AUX 45 to select the new configuration, or, keep the existing configuration stored in memory.



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

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

I/O Points Required for Each Module

Each type of module requires a certain number of I/O points. This is also true for some specialty modules, such as analog, counter interface, etc.

| DC Input Modules | Number of I/O Pts. Required | Specialty Modules, etc. | Number of I/O Pts. Required |
|-----------------------------|---|-------------------------|-----------------------------|
| D2-08ND3 | 8 Input | H2-ECOM(-F) | None |
| D2-16ND3-2 | 16 Input | D2-DCM | None |
| D2-32ND3(-2) | 32 Input | H2-ERM(100,-F) | None |
| AC Input Modules | | H2-EBC(-F) | None |
| D2-08NA-1 | 8 Input | D2-RMSM | None |
| D2-08NA-2 | 8 Input | D2-RSSS | None |
| D2-16NA | 16 Input | F2-CP128 | None |
| DC Output Modules | | H2-CTRIO(2) | None |
| D2-04TD1 | 8 Output (Only the first four points are used) | D2-CTRINT | 8 Input 8 Output |
| D2-08TD1 | 8 Output | F2-DEVNETS-1 | None |
| D2-16TD1-2 (2-2) | 16 Output | H2-PBC | None |
| D2-16TD1(2)P | 16 Output | F2-SDS-1 | None |
| D2-32TD1(-2) | 32 Output | D2-08SIM | 8 Input |
| AC Output Modules | | D2-EM | None |
| D2-08TA | 8 Output | D2-CM | None |
| F2-08TA | 8 Output | H2-ECOM(100) | None |
| D2-12TA | 16 Output (See note 1) | | |
| Relay Output Modules | | | |
| D2-04TRS | 8 Output (Only the first four points are used) | | |
| D2-08TR | 8 Output | | |
| F2-08TRS | 8 Output | | |
| F2-08TR | 8 Output | | |
| D2-12TR | 16 Output (See note 1) | | |
| Combination Modules | | | |
| D2-08CDR | 8 In, 8 Out (Only the first four points are used for each type) | | |
| Analog Modules | | | |
| F2-04AD-1 & 1L | 16 Input | | |
| F2-04AD-2 & 2L | 16 Input | | |
| F2-08AD-1 | 16 Input | | |
| F2-02DA-1 & 1L | 16 Output | | |
| F2-02DA-2 & 2L | 16 Output | | |
| F2-08DA-1 | 16 Output | | |
| F2-08DA-2 | 16 Output | | |
| F2-02DAS-1 | 32 Output | | |
| F2-02DAS-2 | 32 Output | | |
| F2-4AD2DA | 16 Input & 16 Output | | |
| F2-8AD4DA-1 | 32 Input & 32 Output | | |
| F2-8AD4DA-2 | 32 Input & 32 Output | | |
| F2-04RTD | 32 Input | | |
| F2-04THM | 32 Input | | |



NOTE 1: -12pt. modules consume 16 points. The first 6 points are assigned, two are skipped, and then the next 6 points are assigned. For example, a D2-12TA installed in slot 0 would use Y0-Y5, and Y-10-Y15. Y6-Y7 and Y16-Y17 would be unused.

Calculating the Power Budget

Managing Your Power Resource

When you determine the types and quantities of I/O modules you will be using in the DL205 system, it is important to remember there is a limited amount of power available from the power supply. We have provided a chart to help you easily see the amount of power available with each base. The following chart will help you calculate the amount of power you need with your I/O selections. At the end of this section is an example of power budgeting and a worksheet for your own calculations.

If the I/O you choose exceeds the maximum power available from the power supply, you may need to use local expansion bases or remote I/O bases.



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

CPU Power Specifications

The following chart shows the amount of current available for the two voltages supplied from the DL205 base. Use these currents when calculating the power budget for your system. The Auxiliary 24V Power Source mentioned in the table is a connection at the base terminal strip allowing you to connect to devices or DL205 modules that require 24VDC.

| Bases | 5V Current Supplied | Auxiliary 24VDC Current Supplied |
|-------------|---------------------|----------------------------------|
| D2-03B-1 | 2600 mA | 300 mA |
| D2-04B-1 | 2600 mA | 300 mA |
| D2-06B-1 | 2600 mA | 300 mA |
| D2-09B-1 | 2600 mA | 300 mA |
| D2-03BDC1-1 | 2600 mA | None |
| D2-04BDC1-1 | 2600 mA | None |
| D2-06BDC1-1 | 2600 mA | None |
| D2-09BDC1-1 | 2600 mA | None |
| D2-06BDC2-1 | 2600 mA | 300 mA |
| D2-09BDC2-1 | 2600 mA | 300 mA |

Module Power Requirements

Use the power requirements shown on the next page to calculate the power budget for your system. If an External 24VDC power supply is required, the external 24VDC from the base power supply may be used as long as the power budget is not exceeded.

| Power Consumed | | | Power Consumed | | |
|-------------------------------------|---------|---------------------|----------------------------|---------|---------------------|
| Device | 5V (mA) | 24V Auxilliary (mA) | Device | 5V (mA) | 24V Auxilliary (mA) |
| CPUs | | | Combination Modules | | |
| D2-230 | 120 | 0 | D2-08CDR | 200 | 0 |
| D2-240 | 120 | 0 | Specialty Modules | | |
| D2-250-1 | 330 | 0 | H2-PBC | 530 | 0 |
| D2-260 | 330 | 0 | H2-ECOM | 450 | 0 |
| DC Input Modules | | | H2-ECOM100 | 300 | 0 |
| D2-08ND3 | 50 | 0 | H2-ECOM-F | 640 | 0 |
| D2-16ND3-2 | 100 | 0 | H2-ERM(100) | 320 | 0 |
| D2-32ND3(-2) | 25 | 0 | H2-ERM-F | 450 | 0 |
| AC Input Modules | | | H2-EBC | 320 | 0 |
| D2-08NA-1 | 50 | 0 | H2-EBC-F | 450 | 0 |
| D2-08NA-2 | 100 | 0 | H2-CTRIO(2) | 275 | 0 |
| D2-16NA | 100 | 0 | D2-DCM | 300 | 0 |
| DC Output Modules | | | D2-RMSM | 200 | 0 |
| D2-04TD1 | 60 | 20 | D2-RSSS | 150 | 0 |
| D2-08TD1(-2) | 100 | 0 | D2-CTRINT | 50* | 0 |
| D2-16TD1-2 | 200 | 80 | D2-08SIM | 50 | 0 |
| D2-16TD2-2 | 200 | 0 | D2-CM | 100 | 0 |
| D2-32TD1(-2) | 350 | 0 | D2-EM | 130 | 0 |
| AC Output Modules | | | F2-CP128 | 235 | 0 |
| D2-08TA | 250 | 0 | F2-DEVNETS-1 | 160 | 0 |
| F2-08TA | 250 | 0 | F2-SDS-1 | 160 | 0 |
| D2-12TA | 350 | 0 | | | |
| Relay Output Modules | | | | | |
| D2-04TRS | 250 | 0 | | | |
| D2-08TR | 250 | 0 | | | |
| F2-08TRS | 670 | 0 | | | |
| F2-08TR | 670 | 0 | | | |
| D2-12TR | 450 | 0 | | | |
| Analog Modules | | | | | |
| F2-04AD-1 | 50 | 80 | F2-02DAS-1 | 100 | 50mA per channel |
| F2-04AD-1L | 100 | 5mA @ 10-30V | F2-02DAS-2 | 100 | 60mA per channel |
| F2-04AD-2 | 110 | 5mA @ 10-30V | F2-4AD2DA | 90 | 80mA** |
| F2-04AD-2L | 60 | 90mA @ 12V** | F2-8AD4DA-1 | 35 | 100 |
| F2-08AD-1 | 100 | 5mA @ 10-30V | F2-8AD4DA-2 | 35 | 80 |
| F2-08AD-2 | 100 | 5mA @ 10-30V | F2-04RTD | 90 | 0 |
| F2-02DA-1 | 40 | 60** | F2-04THM | 110 | 60 |
| F2-02DA-1L | 40 | 70mA @ 12V** | | | |
| F2-02DA-2 | 40 | 60 | | | |
| F2-02DA-2L | 40 | 70mA @ 12V** | | | |
| F2-08DA-1 | 30 | 50mA** | | | |
| F2-08DA-2 | 60 | 140 | | | |
| *requires external 5VDC for outputs | | | | | |
| **add an additional 20mA per loop | | | | | |

Power Budget Calculation Example

The following example shows how to calculate the power budget for the DL205 system.

| Base # 0 | Module Type | 5 VDC (mA) | Auxiliary Power Source 24 VDC Output (mA) |
|---------------------------|-------------|------------------|---|
| Available Base Power | D2-09B-1 | 2600 | 300 |
| CPU Slot | D2-260 | + 330 | |
| Slot 0 | D2-16ND3-2 | + 100 | + 0 |
| Slot 1 | D2-16NA | + 100 | + 0 |
| Slot 2 | D2-16NA | + 100 | + 0 |
| Slot 3 | F2-04AD-1 | + 50 | + 80 |
| Slot 4 | F2-02DA-1 | + 40 | + 60 |
| Slot 5 | D2-08TA | + 250 | + 0 |
| Slot 6 | D2-08TD1 | + 100 | + 0 |
| Slot 7 | D2-08TR | + 250 | + 0 |
| Other | | | |
| Handheld Programmer | D2-HPP | + 200 | + 0 |
| | | | |
| Total Power Required | | 1520 | 140 |
| Remaining Power Available | | 2600-1520 = 1080 | 300 - 140 = 160 |

1. Use the power budget table to fill in the power requirements for all the system components. First, enter the amount of power supplied by the base. Next, list the requirements for the CPU, any I/O modules, and any other devices, such as the Handheld Programmer, C-more HMI or the DV-1000 operator interface. Remember, even though the Handheld Programmer or the DV-1000 are not installed in the base, they still obtain their power from the system. Also, make sure you obtain any external power requirements, such as the 24VDC power required by the analog modules.
2. Add the current columns starting with CPU slot and put the total in the row labeled "Total Power Required."
3. Subtract the row labeled "Total Power Required" from the row labeled "Available Base Power." Place the difference in the row labeled "Remaining Power Available."
4. If "Total Power Required" is greater than the power available from the base, the power budget will be exceeded. It will be unsafe to use this configuration, and you will need to restructure your I/O configuration.



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

Power Budget Calculation Worksheet

This blank chart is provided for you to copy and use in your power budget calculations.

| Base # 0 | Module Type | 5 VDC (mA) | Auxiliary Power Source 24 VDC Output (mA) |
|---------------------------|-------------|------------|---|
| Available Base Power | | | |
| CPU Slot | | | |
| Slot 0 | | | |
| Slot 1 | | | |
| Slot 2 | | | |
| Slot 3 | | | |
| Slot 4 | | | |
| Slot 5 | | | |
| Slot 6 | | | |
| Slot 7 | | | |
| Other | | | |
| | | | |
| Total Power Required | | | |
| Remaining Power Available | | | |

1. Use the power budget table to fill in the power requirements for all the system components. This includes the CPU, any I/O modules, and any other devices, such as the Handheld Programmer, C-more HMI or the DV-1000 operator interface. Also, make sure you obtain any external power requirements, such as the 24VDC power required by the analog modules.
2. Add the current columns starting with CPU slot and put the total in the row labeled "Total Power Required."
3. Subtract the row labeled "Total Power Required" from the row labeled "Available Base Power." Place the difference in the row labeled "Remaining Power Available."
4. If "Total Power Required" is greater than the power available from the base, the power budget will be exceeded. It will be unsafe to use this configuration, and you will need to restructure your I/O configuration.



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

Local Expansion I/O

Use local expansion when you need more I/O points, a greater power budget than the local CPU base provides or when placing an I/O base at a location away from the CPU base, but within the expansion cable limits. Each local expansion base requires the D2–CM controller module in the CPU slot. The local CPU base requires the D2–EM expansion module, as well as each expansion base. All bases in the system must be the new (–1) bases. These bases have a connector on the right side of the base to which the D2–EM expansion module attaches. All local and local expansion I/O points are updated on every CPU scan.

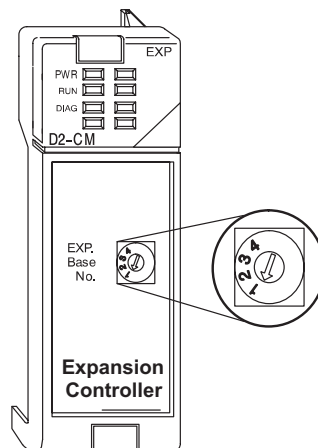
Use the *DirectSOFT* PLC Configure I/O menu option to view the local expansion system automatic I/O addressing configuration. This menu also allows manual addresses to be assigned if necessary.

| | DL230 | DL240 | DL250 | DL250-1 | DL260 |
|--|---|-------|-------|-------------|-------|
| Total number of local / expansion bases per system | These CPUs do not support local expansion systems | | | 3 | 5 |
| Maximum number of expansion bases | | | | 2 | 4 |
| Total I/O (includes CPU base and expansion bases) | | | | 768 | 1280 |
| Maximum inputs | | | | 512 | 1024 |
| Maximum outputs | | | | 512 | 1024 |
| Maximum expansion system cable length | | | | 30m (98ft.) | |

D2–CM Local Expansion Module

The D2–CM module is placed in the CPU slot of each expansion base. The rotary switch is used to select the expansion base number. The expansion base I/O addressing (Xs and Ys) is based on the numerical order of the rotary switch selection and is recognized by the CPU on power-up. Duplicate expansion base numbers will not be recognized by the CPU.

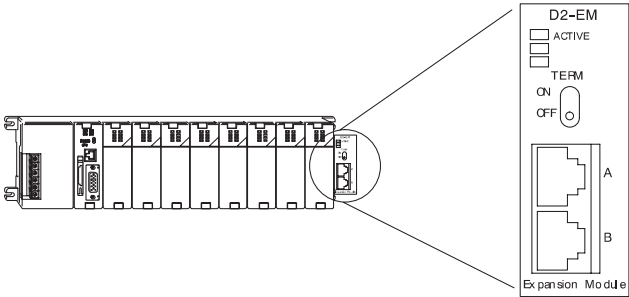
The status indicator LEDs on the D2–CM front panels have specific functions which can help in programming and troubleshooting.



| D2–CM Indicators | Status | Meaning |
|------------------|--------|--|
| PWR (Green) | ON | Power good |
| | OFF | Power failure |
| RUN (Green) | ON | D2–CM has established communication with PLC |
| | OFF | D2–CM has not established communication with PLC |
| DIAG (Red) | ON | Hardware watch-dog failure |
| | ON/OFF | I/O module failure (ON 500ms / OFF 500ms) |
| | OFF | No D2–CM error |

D2-EM Local Expansion Module

The D2-EM expansion unit is attached to the right side of each base in the expansion system, including the local CPU base. (All bases in the local expansion system must be the new (-1) bases). The D2-EMs on each end of the expansion system should have the TERM (termination) switch placed in the ON position. The expansion units between the endmost bases should have the TERM switch placed in the OFF position. The CPU base can be located at any base position in the expansion system. The bases are connected in a daisy-chain fashion using the D2-EXCBL-1 (category 5 straight-through cable with RJ45 connectors). Either of the RJ45 ports (labelled A and B) can be used to connect one expansion base to another.



The status indicator LEDs on the D2-EM front panels have specific functions which can help in programming and troubleshooting.

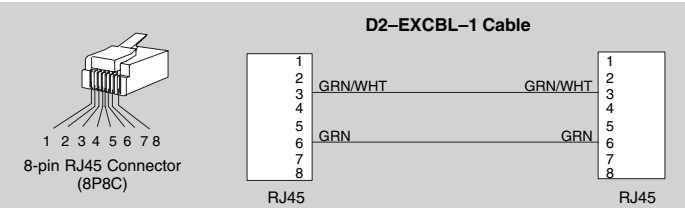
| D2-EM Indicator | Status | Meaning |
|-----------------|--------|---|
| ACTIVE (Green) | ON | D2-EM is communicating with other D2-EM |
| | OFF | D2-EM is not communicating with other D2-EM |



WARNING: Connect/disconnect the expansion cables with the PLC power turned OFF in order for the ACTIVE indicator to function normally.

D2-EXCBL-1 Local Expansion Cable

The category 5 straight-through D2-EXCBL-1 (1m) is used to connect the D2-EM expansion modules together. If longer cable lengths are required, we recommend that you purchase a commercially manufactured cable with RJ45 connectors already attached. The maximum total expansion system cable length is 30m (98ft). **Do not use Ethernet hubs** to connect the local expansion network together.

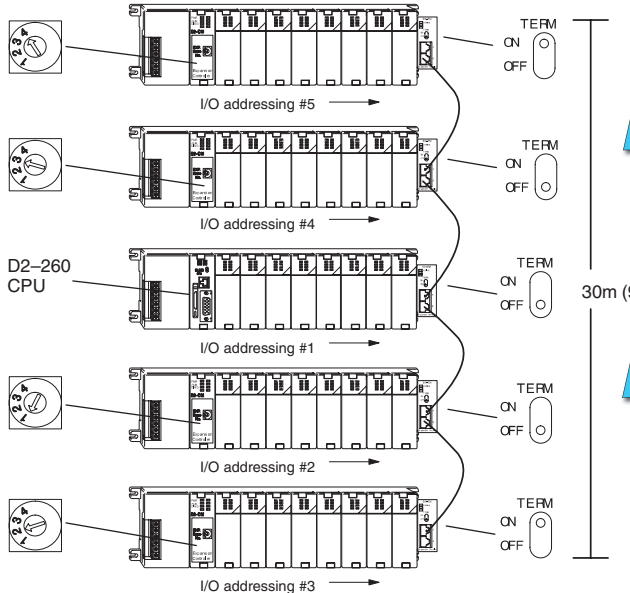


NOTE: Commercially available Patch (Straight-through) Category 5, UTP cables will work in place of the D2-EXCBL-1. The D2-EM modules only use the wires connected to pins 3 and 6 as shown above.

DL260 Local Expansion System

The D2-260 supports local expansion up to five total bases (one CPU base + four local expansion bases) and up to a maximum of 1280 total I/O points. An example local expansion system is shown below. All local and expansion I/O points are updated on every CPU scan. **No specialty modules can be located in the expansion bases** (refer to the Module Placement Table earlier in this chapter for restrictions).

D2-CM Expansion
Base Number Selection



D2-EM Termination
Switch Settings

TERM
ON
OFF

TERM
ON
OFF

TERM
ON
OFF

TERM
ON
OFF

TERM
ON
OFF

30m (98ft.) max. cable length

NOTE: Do not use Ethernet hubs to connect the local expansion system together.

NOTE: Use D2-EXCBL-1 (1m) (Category 5 straight-through cable) to connect the D2-EMs together.

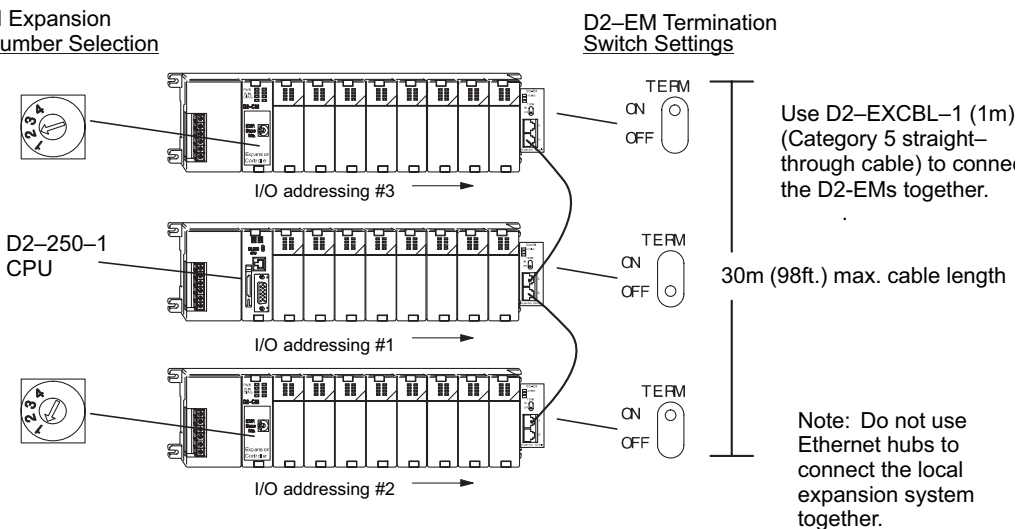
- The CPU base can be located at any base position in the expansion system.
- All discrete and analog modules are supported in the expansion bases. Specialty modules are not supported in the expansion bases.
- The D2-CMs do not have to be in successive numerical order; however, the numerical rotary selection determines the X and Y addressing order. The CPU will recognize the local and expansion I/O on power-up. Do not duplicate numerical selections.
- The TERM (termination) switch on the two endmost D2-EMs must be in the ON position. The other D2-EMs in between should be in the OFF position.
- Use the D2-EXCBL-1 or equivalent cable to connect the D2-EMs together. Either of the RJ45 ports (labeled A and B) on the D2-EM can be used to connect one base to another.



NOTE: When applying power to the CPU (DL250-1/260) and local expansion bases, make sure the expansion bases power up at the same time or before the CPU base. Expansion bases that power up after the CPU base will not be recognized by the CPU. (See chapter 3 Initialization Process timing specifications).

DL250-1 Local Expansion System

The D2-250-1 supports local expansion up to three total bases (one CPU base + two local expansion bases) and up to a maximum of 768 total I/O points. An example local expansion system is shown below. All local and expansion I/O points are updated on every CPU scan. No specialty modules can be located in the expansion bases (refer to the Module Placement Table earlier in this chapter for restrictions).



- The CPU base can be located at any base position in the expansion system.
- All discrete and analog modules are supported in the expansion bases. Specialty modules are not supported in the expansion bases.
- The D2-CMs do not have to be in successive numerical order, however, the numerical rotary selection determines the X and Y addressing order. The CPU will recognize the local and expansion I/O on power-up. Do not duplicate numerical selections.
- The TERM (termination) switch on the two endmost D2-EMs must be in the ON position. The other D2-EMs in between should be in the OFF position.
- Use the D2-EXCBL-1 or equivalent cable to connect the D2-EMs together. Either of the RJ45 ports (labelled A and B) on the D2-EM can be used to connect one base to another.

Expansion Base Output Hold Option

The bit settings in V-memory registers V7741 and V7742 determine the expansion bases' outputs response to a communications failure. The CPU will exit the RUN mode to the STOP mode when an expansion base communications failure occurs. If the Output Hold bit is ON, the outputs on the corresponding module will hold their last state when a communication error occurs. If OFF (default), the outputs on the module unit will turn off in response to an error. The setting does not have to be the same for all the modules on an expansion base.

The selection of the output mode will depend on your application. You must consider the consequences of turning off all the devices in one or all expansion bases at the same time vs. letting the system run "steady state" while unresponsive to input changes. For example, a conveyor system would typically suffer no harm if the system were shut down all at once. In a way, it is the equivalent of an "E-STOP". On the other hand, for a continuous process such as waste water treatment, holding the last state would allow the current state of the process to continue until the operator can intervene manually. V7741 and V7742 are reserved for the expansion base Output Hold option. The bit definitions are as follows:

Bit = 0 Output Off (Default)

Bit = 1 Output Hold

| D2-CM Expansion Base Hold Output | | | | | | | | | | |
|----------------------------------|-------------------|-----|--------|--------|--------|--------|--------|--------|--------|--------|
| Expansion Base No. | V-memory Register | | Slot 0 | Slot 1 | Slot 2 | Slot 3 | Slot 4 | Slot 5 | Slot 6 | Slot 7 |
| Exp. Base 1 | V7741 | Bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Exp. Base 2 | | | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Exp. Base 3 | V7742 | Bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Exp. Base 4 | | | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |



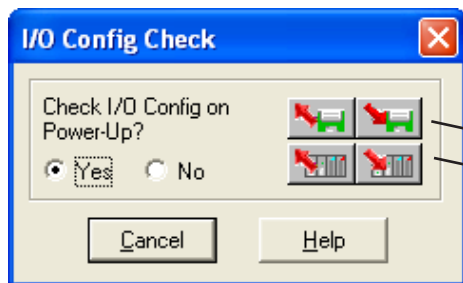
WARNING: Selecting "HOLD LAST STATE" means that outputs on the expansion bases will not be under program control in the event of a communications failure. Consider the consequences to process operation carefully before selecting this mode.

Enabling I/O Configuration Check using *DirectSOFT*

Enabling the I/O Config Check will force the CPU, at power up, to examine the local and expansion I/O configuration before entering the RUN mode. If there is a change in the I/O configuration, the CPU will not enter the RUN mode. For example, if local expansion base #1 does not power up with the CPU and the other expansion bases, the I/O Configuration Check will prevent the CPU from entering the RUN mode. If the I/O Configuration check is disabled and automatic addressing is used, the CPU would assign addresses from expansion base #1 to base #2 and possibly enter the RUN mode. This is not desirable, and can be prevented by enabling the I/O Configuration check.

Manual addressing can be used to manually assign addresses to the I/O modules. This will prevent any automatic addressing re-assignments by the CPU. The I/O Configuration Check can also be used with manual addressing.

To display the I/O Config Check window, use *DirectSOFT*>PLC menu>Setup>I/O Config Check.



Select "Yes," then save to disk or to PLC, if connected to the PLC.

Expanding DL205 I/O

I/O Expansion Overview

Expanding I/O beyond the local chassis is useful for a system which has a sufficient number of sensors and other field devices located a relatively long distance from the CPU. Two forms of communication can be used to add remote I/O to your system: either an Ethernet or a serial communication network. A discussion of each method follows.

Ethernet Remote Master, H2-ERM(100, -F)



230



240



250-1



260

The Ethernet Remote Master, H2-ERM(100, -F), is a module that provides a low-cost, high-speed Ethernet Remote I/O link to connect either a DL240, a DL250-1 or a DL260 CPU to slave I/O over a high-speed Ethernet link.

Each H2-ERM(100) module can support up to 16 additional H2-EBC systems, 16 Terminator I/O EBC systems, or 16 fully expanded H4-EBC systems.

The H2-ERM(100) connects to your control network using Category 5 UTP cables for distances up to 100m (328ft). Repeaters are used to extend the distances and to expand the number of nodes. The fiber optic version, H2-ERM-F, uses industry standard 62.5/125 ST-style fiber optic cables and can be run up to 2,000m (6560ft).

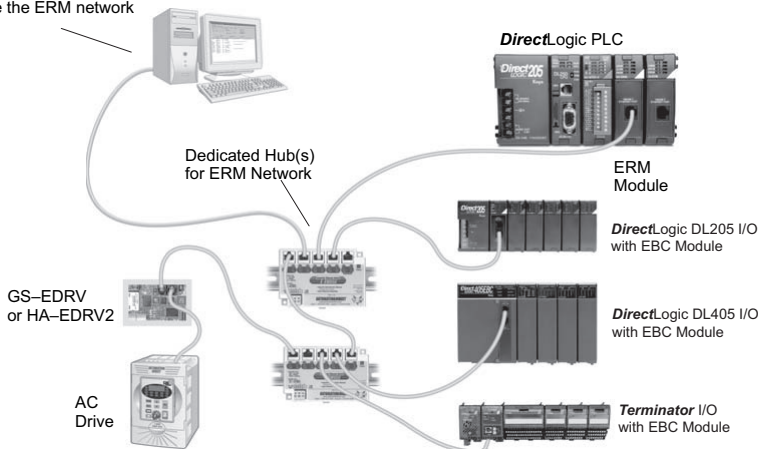
The PLC, ERM and EBC slave modules work together to update the remote I/O points. These three scan cycles are occurring at the same time, but asynchronously. We recommend that critical I/O points that must be monitored every scan be placed in the CPU base.

| Specifications | H2-ERM | H2-ERM100 | H2-ERM-F |
|---------------------------|---------------------|--|-----------------------|
| Communications | 10BaseT Ethernet | 10/100BaseT Ethernet | 10BaseFL Ethernet |
| Data Transfer Rate | 10Mbps | 100Mbps | 10Mbps |
| Link Distance | 100 meters (328 ft) | | 2000 meters (6560 ft) |
| Ethernet Port | RJ45 | | ST-style fiber optic |
| Ethernet Protocols | TCP/IP, IPX | TCP/IP, IPX, Modbus TCP/IP, DHCP, HTML configuration | TCP/IP, IPX |
| Power Consumption | 320mA @ 5VDC | | 450mA @ 5VDC |

Ethernet Remote Master Hardware Configuration

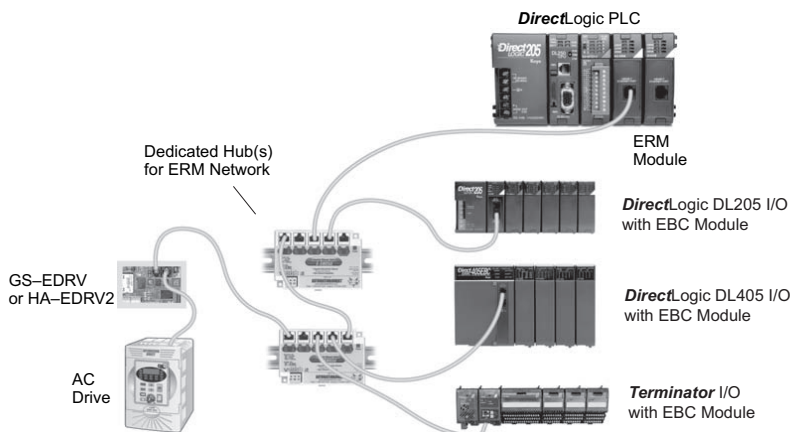
Use a PC equipped with a 10/100BaseT or a 10BaseFL network adapter card and the Ethernet Remote Master (ERM) Workbench software configuration utility (included with the ERM manual, H24-ERM-M) to configure the ERM module and its slaves over the Ethernet remote I/O network.

PC running ERM WorkBench
to configure the ERM network



When networking ERM's with other Ethernet devices, we recommend that a dedicated Ethernet remote I/O network be used for the ERM and its slaves. While Ethernet networks can handle an extremely large number of data transactions, and normally very quickly, heavy Ethernet traffic can adversely affect the reliability of the slave I/O and the speed of the I/O network. Keep ERM networks, multiple ERM networks and ECOM/office networks isolated from one another.

Once the ERM remote I/O network is configured and running, the PC can be removed from the network.



Installing the ERM Module

This section will briefly describe the installation of the ERM module. More detailed information is available in the Ethernet Remote Master Module manual, H24-ERM-M, which will be needed to configure the communication link to the remote I/O.

In addition to the manual, configuration software will be needed. The ERM Workbench software utility must be used to configure the ERM and its slave modules. The utility is provided on a CD which comes with the ERM manual. The ERM module can be identified by two different methods, either by Module ID (dip switch) or by Ethernet address. Whichever method is used, the ERM Workbench is all that is needed to configure the network modules.

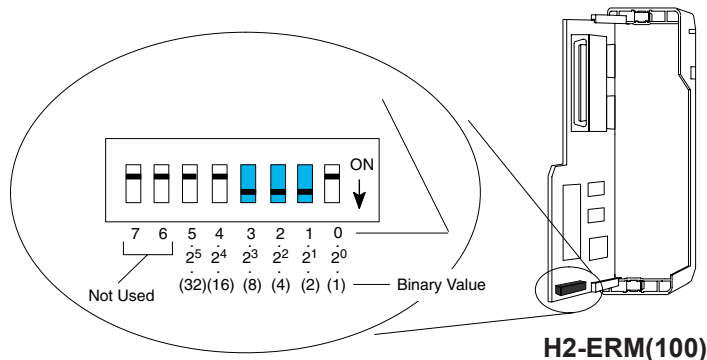
If IP addressing (UDP/IP) is necessary or if the Module ID is set with software, the NetEdit software utility (included with the ERM Workbench utility) will be needed in addition to the ERM Workbench.

ERM Module ID

Set the ERM Module ID before installing the module in the DL205 base. Always set the module ID to 0. A Module ID can be set in one of two ways:

- Use the DIP switches on the module (1-63)
- Use the configuration tools in NetEdit

Use the DIP switch to install and change slave modules without using a PC to set the Module ID. Set the module's DIP switch, insert the module in the base, and connect the network cable. The Module ID is set on power up, and it is ready to communicate on the network.

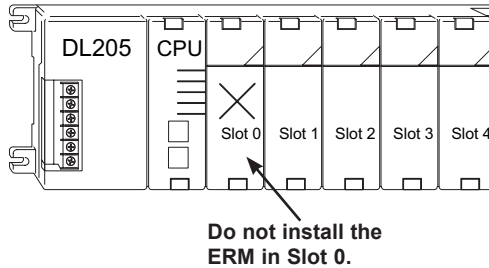


The Module IDs can also be set or changed on the network from a single PC by using the tools in NetEdit.

The Module ID equals the *sum* of the binary values of the slide switches set in the ON position. For example, if slide switches 1, 2 and 3 are set to the ON position, the Module ID will be 14. This is found by adding $8+4+2=14$. The maximum value which can be set on the DIP switch is $32+16+8+4+2=63$. This is achieved by setting switches 0 through 5 to the ON position. The 6 and 7 switch positions are inactive.

Insert the ERM Module

The DL205 system *only* supports the placement of the ERM module in the CPU base. It does not support installation of the ERM module in either local expansion or remote I/O bases. The number of usable slots depends on how many slots the base has. All of the DL205 CPUs support the ERM module, except the D2-230.



NOTE: The module will not work in slot 0 of the DL205 series PLCs, the slot next to the CPU.

Network Cabling

Of the three types of ERM modules available, one supports the 10BaseT standard, another supports 10/100BaseT and the other one supports the 10BaseFL standard. The 10/100BaseT standard uses twisted pairs of copper wire conductors and the 10BaseFL standard is used with fiber optic cabling.

10/100BaseT

Unshielded
Twisted-Pair
cable with RJ45
connectors



10BaseFL

62.5/125 MMF
fiber optics cable
with ST-style
connectors

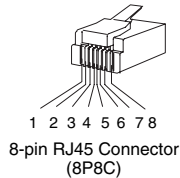


10/100BaseT Networks

A patch (straight-through) cable is used to connect a PLC (or PC) to a hub or to a repeater. Use a crossover cable to connect two Ethernet devices (point-to-point) together. It is recommended that pre-assembled cables be purchased for convenient and reliable networking.

The above diagram illustrates the standard wire positions of the RJ45 connector. It is recommended that Catagory 5, UTP cable be used for all ERM 10/100BaseT cables.

10/100BaseT



Patch (Straight-through) Cable

| | | | | | |
|-----|---|---------|---------|---|-----|
| TD+ | 1 | OR/WHT | OR/WHT | 1 | RD+ |
| TD- | 2 | OR | OR | 2 | RD- |
| RD+ | 3 | GRN/WHT | GRN/WHT | 3 | TD+ |
| | 4 | BLU | BLU | 4 | |
| | 5 | BLU/WHT | BLU/WHT | 5 | |
| RD- | 6 | GRN | GRN | 6 | TD- |
| | 7 | BRN/WHT | BRN/WHT | 7 | |
| | 8 | BRN | BRN | 8 | |

RJ45 RJ45

Crossover Cable

| | | | | | |
|-----|---|---------|---------|---|-----|
| TD+ | 1 | OR/WHT | GRN/WHT | 1 | TD+ |
| TD- | 2 | OR | GRN | 2 | TD- |
| RD+ | 3 | GRN/WHT | OR/WHT | 3 | RD+ |
| | 4 | BLU | BLU | 4 | |
| | 5 | BLU/WHT | BLU/WHT | 5 | |
| RD- | 6 | GRN | OR | 6 | RD- |
| | 7 | BRN/WHT | BRN/WHT | 7 | |
| | 8 | BRN | BRN | 8 | |

RJ45 RJ45

Refer to the ERM manual for using the fiber optic cable with the H2-ERM-F.

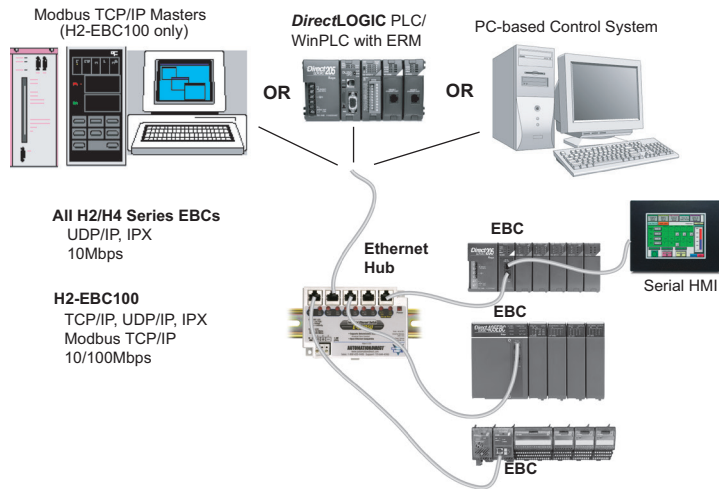
An explanation of the use of the ERM Workbench software is too lengthy for this manual. The full use of the workbench and NetEdit utilities is discussed in the ERM manual.

Ethernet Base Controller, H2-EBC(100)(-F)

The Ethernet Base Controller module H2-EBC(100)(-F) provides a low-cost, high-performance Ethernet link between a network master controller and an *DirectLOGIC* PLC I/O slave system. Also, the H2-EBC100 supports the Modbus TCP/IP client/server protocol.

The Ethernet Base Controller (EBC) serves as an interface between the master control system and the DL205/405 I/O modules. The control function is performed by the master controller, not the EBC slave. The EBC occupies the CPU slot in the base and communicates across the backplane to input and output modules. Various master controllers with EBC slaves are shown in the diagram below.

Example EBC Systems: Various Masters with EBC Slaves



The H2-EBC module supports industry standard 10BaseT Ethernet communications, the H2-EBC100 module supports industry standard 10/100BaseT Ethernet communications and the H2-EBC-F module supports 10BaseFL (fiber optic) Ethernet standards.

| Specifications | H2-EBC | H2-EBC100 | H2-EBC-F |
|--------------------|--------------------------|---|----------------------|
| Communications | 10BaseT Ethernet | 10/100BaseT Ethernet | 10BaseFL Ethernet |
| Data Transfer Rate | 10 Mbps max. | 100 Mbps max. | 10 Mbps max. |
| Link Distance | 100m (328ft) | 100m (328ft) | 2000m (6560ft) |
| Ethernet Port | RJ45 | RJ45 | ST-style fiber optic |
| Ethernet Protocols | TCP/IP, IPX | TCP/IP, IPX/Modbus TCP/IP, DHCP, HTML configuration | TCP/IP, IPX |
| Serial Port | RJ12 | RJ12 | None |
| Serial Protocols | K-Sequence, ASCII IN/OUT | K-Sequence, ASCII IN/OUT, Modbus RTU | None |
| Power Consumption | 450mA @ 5VDC | 300mA @ 5VDC | 640mA @ 5VDC |

Install the EBC Module

Like the ERM module discussed in the previous section, this section will briefly describe the installation of the H2 Series EBCs. More detailed information is available in the Ethernet Base Controller manual, H24-EBC-M, which will be needed to configure the remote I/O.

Each EBC module must be assigned at least one unique identifier to make it possible for master controllers to recognize it on the network. Two methods for identifying the EBC module give it the flexibility to fit most networking schemes. These identifiers are:

- Module ID (IPX protocol only)
- IP Address (for TCP/IP and Modbus TCP/IP protocols)

Set the Module ID

The two methods which can be used to set the EBC module ID are either by DIP switch or by software. One software method is to use the NetEdit3 program which is included with the EBC manual. To keep the set-up discussion simple here, only the DIP switch method will be discussed. Refer to the EBC manual for the complete use of NetEdit3.

It is recommended to use the DIP switch to set the Module ID because the DIP switch is simple to set, and the Module ID can be determined by looking at the physical module, without reference to a software utility.

The DIP switch can be used to set the Module ID to a number from 1-63. Do not use Module ID 0 for communication.

If the DIP switch is set to a number greater than 0, the software utilities are disabled from setting the Module ID. Software utilities will only allow changes to the Module ID if the DIP switch setting is 0 (all switches OFF).

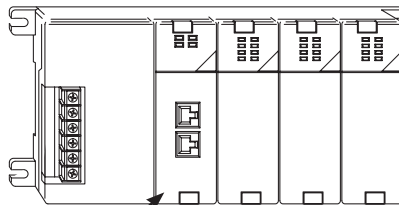


NOTE: The DIP switch settings are read at powerup only. The power must be cycled each time the DIP switches are changed.

Setting the Module ID with the DIP switches is identical to setting the DIP switches on the H2-ERM(100) module. Refer to page 4-19 in this chapter.

Insert the EBC Module

Once the Module ID DIP switches are set, insert the module in the CPU slot of any DL205 base.

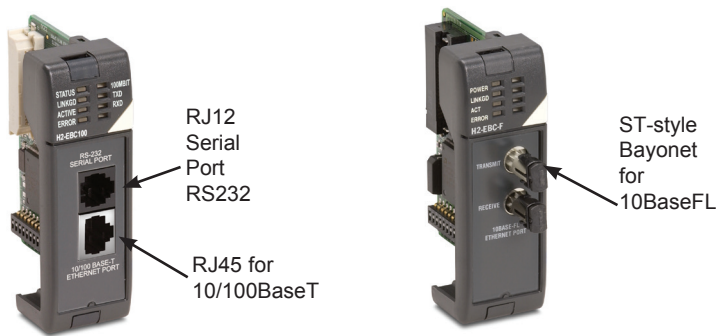


Insert H2-EBC in CPU slot

Network Cabling

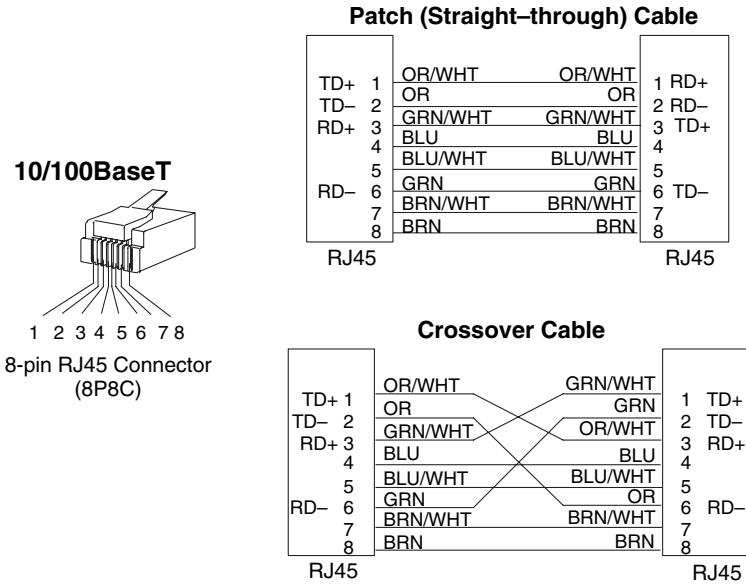
Of the two types of EBC modules available, one supports the 10/100BaseT standard and the other one supports the 10BaseFL standard. The 10/100BaseT standard uses twisted pairs of copper wire conductors and the 10BaseFL standard is used with fiber optic cabling.

10/100BaseT



The 10BaseT and 100BaseT EBCs have an eight-pin modular jack that accepts RJ45 connectors. UTP Category 5 (CAT5) cable is highly recommended for use with all Ethernet 10/100BaseT connections. For convenient and reliable networking, purchase commercially manufactured cables which have the connectors already installed.

To connect an EBC, or a PC, to a hub or repeater, use a patch cable (sometimes called a straight-through cable). The cable used to connect a PC directly to an EBC or to connect two hubs is referred to as a crossover cable.

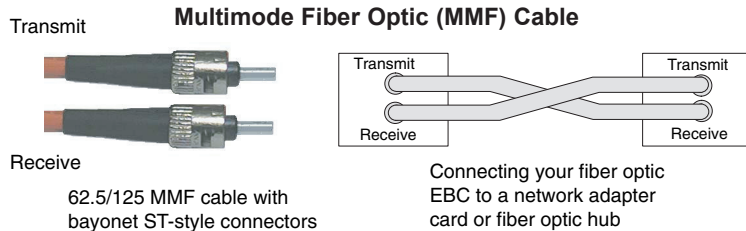


10BaseFL Network Cabling

The H2-EBC-F and the H2-ERM-F modules have two ST-style bayonet connectors. The ST-style connector uses a quick release coupling which requires a quarter turn to engage or disengage. The connectors provide mechanical and optical alignment of fibers.

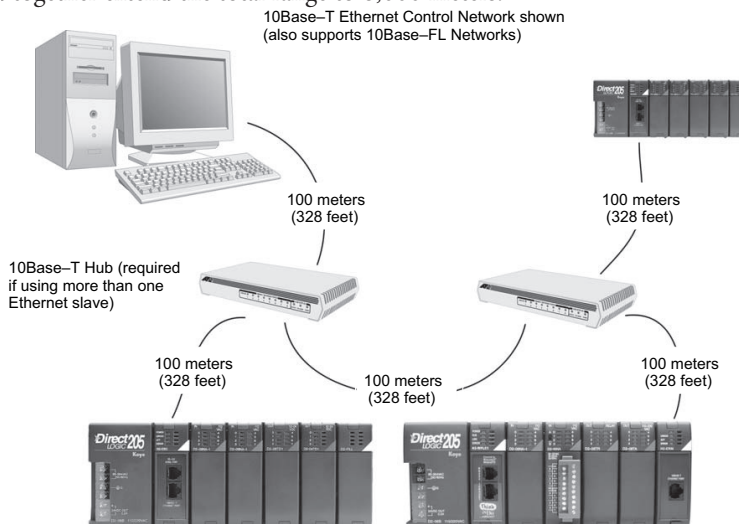
Each cable segment requires two strands of fiber; one to transmit data and one to receive data. The ST-style connectors are used to connect the H2-Exx-F module to a PC or a fiber optic hub or repeater. The modules themselves cannot act as repeaters.

The H2-EBC-F and the H2-ERM-F modules accept 62.5/125 multimode fiber optic (MMF) cable. The glass core diameter is 62.5 micrometers, and the glass cladding is 125 micrometers. The fiber optic cable is highly immune to noise and permits communications over much greater distances than 10/100BaseT.



Maximum Cable Length

The maximum distance per 10/100BaseT cable segment is 100 meters (328 feet). Repeaters extend the distance. Each cable segment attached to a repeater can be 100 meters. Two repeaters connected together extend the total range to 300 meters. The maximum distance per 10BaseFL cable segment is 2,000 meters (6,560 feet or 1.2 miles). Repeaters extend the distance. Each cable segment attached to a repeater can be 2,000 meters. Two repeaters connected together extend the total range to 6,000 meters.



Add a Serial Remote I/O Master/Slave Module

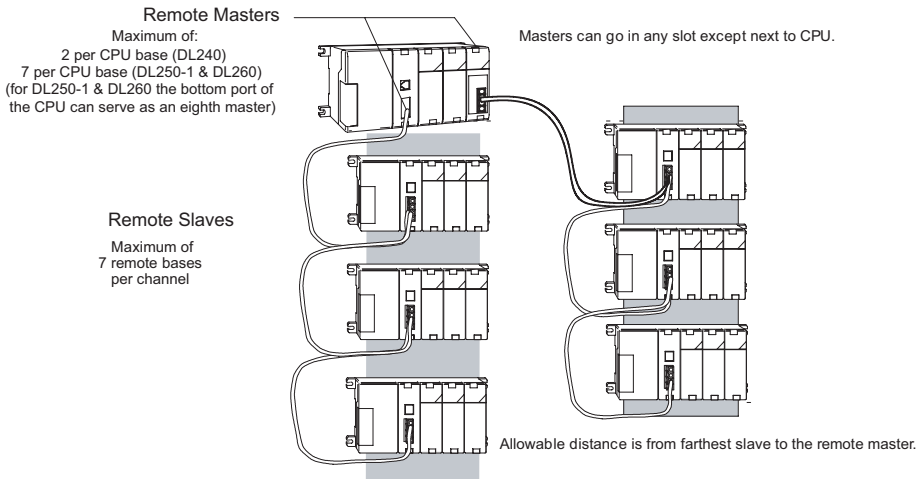
In addition to the I/O located in the local base, adding remote I/O can be accomplished via a shielded twisted-pair cable linking the master CPU to a remote I/O base. The methods of adding serial remote I/O are:

- DL240 CPUs: Remote I/O requires a remote master module (D2–RMSM) to be installed in the local base. The CPU updates the remote master, then the remote master handles all communication to and from the remote I/O base by communicating to a remote slave module (D2–RSSS) installed in each remote base.
- DL250–1 and D2–260 CPU: The CPU's comm port 2 features a built-in Remote I/O channel. You may also use up to seven D2–RMSM remote masters in the local base as described above (you can use either or both methods).

| | DL230 | DL240 | DL250–1 | DL260 |
|--|-------|---------------------------------------|---------|-------|
| Maximum number of Remote Masters supported in the local CPU base (1 channel per Remote Master) | none | 2 | 7 | 7 |
| CPU built-in Remote I/O channels | none | none | 1 | 1 |
| Maximum I/O points supported by each channel | none | 2048 | 2048 | 2048 |
| Maximum Remote I/O points supported | none | Limited by total references available | | |
| Maximum number of Remote I/O bases per channel(RM–NET) | none | 7 | 7 | 7 |
| Maximum number of Remote I/O bases per channel (SM–NET) | none | 31 | 31 | 31 |

Remote I/O points map into different CPU memory locations, therefore it does not reduce the number of local I/O points. Refer to the DL205 Remote I/O manual for details on remote I/O configuration and numbering. Configuring the built-in remote I/O channel is described in the following section.

The figure below shows one CPU base, and one remote I/O channel with six remote bases. If the CPU is a DL250–1 or DL260, adding the first remote I/O channel does not require installing a remote master module (use the CPU's built-in remote I/O channel).



Configuring the CPU's Remote I/O Channel

 230

 240

 250-1

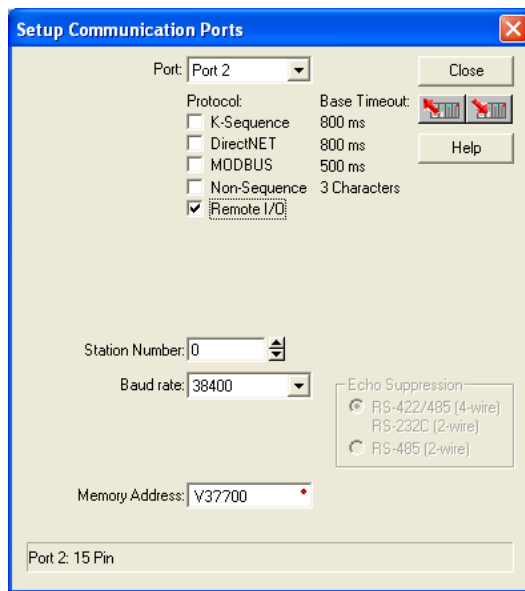
 260

This section describes how to configure the DL250-1 and DL260's built-in remote I/O channel. Additional information is in the Remote I/O manual, D2-REMIO-M, which you will need in configuring the Remote slave units on the network. You can use the D2-REMIO-M manual exclusively when using regular Remote Masters and Remote Slaves for remote I/O in any DL205 system.

The DL250-1 and DL260 CPU's built-in remote I/O channel only supports RM-Net which allows it to communicate with up to seven remote bases containing a maximum of 2048 I/O points per channel, at a maximum distance of 1000 meters. If required, you can still use Remote Master modules in the local CPU base (2048 I/O points on each channel).

You may recall from the CPU specifications in Chapter 3 that the DL250-1 and DL260's Port 2 is capable of several protocols. To configure the port using the Handheld Programmer, use AUX 56 and follow the prompts, making the same choices as indicated below on this page. To configure the port in *DirectSOFT*, choose the PLC menu, then Setup, then Setup Secondary Comm Port.

- **Port:** From the port number list box at the top, choose "Port 2."
- **Protocol:** Click the check box to the left of "Remote I/O" (called "M-NET" on the HPP), and then you'll see the dialog box shown below.
- **Station Number:** Choose "0" as the station number, which makes the DL250-1 or DL260 the master. Station numbers 1-7 are reserved for remote slaves.
- **Baud Rate:** The baud rates 19200 and 38400 are available. Choose 38400 initially as the remote I/O baud rate, and revert to 19200 baud if you experience data errors or noise problems on the link.
- **Memory Address:** Choose a V-memory address to use as the starting location of a Remote I/O configuration table (V37700 is the default). This table is separate and independent from the table for any Remote Master(s) in the system, and it is 32 words in length.




Then click the button indicated to send the Port 2 configuration to the CPU, and click Close.

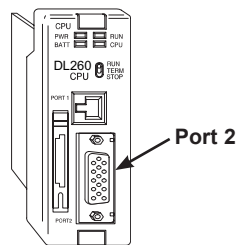


NOTE: You must configure the baud rate on the Remote Slaves with DIP switches to match the baud rate selection for the CPU's Port 2.

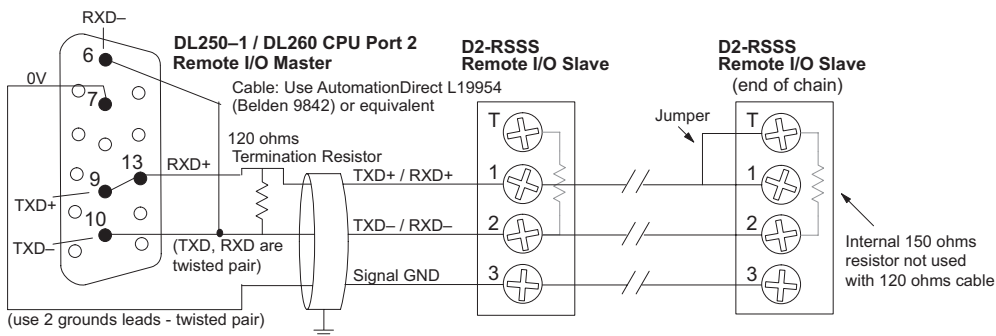
The next step is to make the connections between all devices on the Remote I/O link.

The location of Port 2 on the DL250-1 and DL260 is on the 15-pin connector, as pictured to the right.

- Pin 7 Signal GND
- Pin 9 TXD+
- Pin 10 TXD-
- Pin 13 RXD+
- Pin 6 RXD-

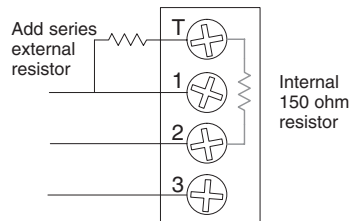


Now we are ready to discuss wiring the DL250-1 or DL260 to the remote slaves on the remote base(s). The remote I/O link is a 3-wire, half-duplex type. Since Port 2 of the DL250-1 and DL260 CPU is a 5-wire full duplex-capable port, we must jumper its transmit and receive lines together as shown below (converts it to 3-wire, half-duplex).



The twisted/shielded pair connects to the DL250-1 or DL260 Port 2 as shown. A termination resistor must be added externally to the CPU, as close as possible to the connector pins. Its purpose is to minimize electrical reflections that occur over long cables. A termination resistor must be present at both physical ends of the network.

Ideally, the two termination resistors at the cable's opposite ends and the cable's rated impedance will all match. For cable impedances greater than 150 ohms, add a series resistor at the last slave as shown to the right. If less than 150 ohms, parallel a matching resistance across the slave's pins 1 and 2 instead. Remember to size the termination resistor at Port 2 to match the cables rated impedance. *The resistance values should be between 100 and 500 ohms.*



NOTE: To match termination resistance to AutomationDirect L19827 (Belden 9841), use a 120 ohm resistor across terminals 1 and 2.

See the transient suppression for inductive loads information in Chapter 2 of this manual for further information on wiring practices.

Configure Remote I/O Slaves

After configuring the DL250–1 or DL260 CPU's Port 2 and wiring it to the remote slave(s), use the following checklist to complete the configuration of the remote slaves. Full instructions for these steps are in the Remote I/O manual.

- Set the baud rate to match CPU's Port 2 setting.
- Select a station address for each slave, from 1 to 7. Each device on the remote link must have a unique station address. There can be only one master (address 0) on the remote link.

Configuring the Remote I/O Table

The beginning of the configuration table for the built-in remote I/O channel is the memory address we selected in the Port 2 setup.

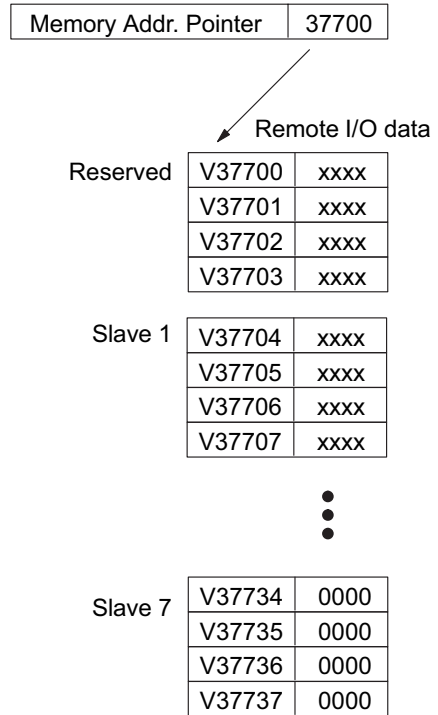
The table consists of blocks of four words which correspond to each slave in the system, as shown to the right. The first four table locations are reserved.

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

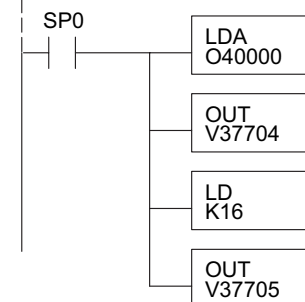
1. Starting address of slave's input data
2. Number of slave's input points
3. Starting address of outputs in slave
4. Number of slave's output points

The table is 32 words long. If your system has fewer than seven remote slave bases, then the remainder of the table must be filled with zeros. For example, a three-slave system will have a remote configuration table containing four reserved words, 12 words of data and 16 words of "0000."

A portion of the ladder program must configure this table (only once) at powerup. Use the LDA instruction as shown to the right, to load an address to place in the table. Use the regular LD constant to load the number of the slave's input or output points. The following page gives a short program example for one slave.

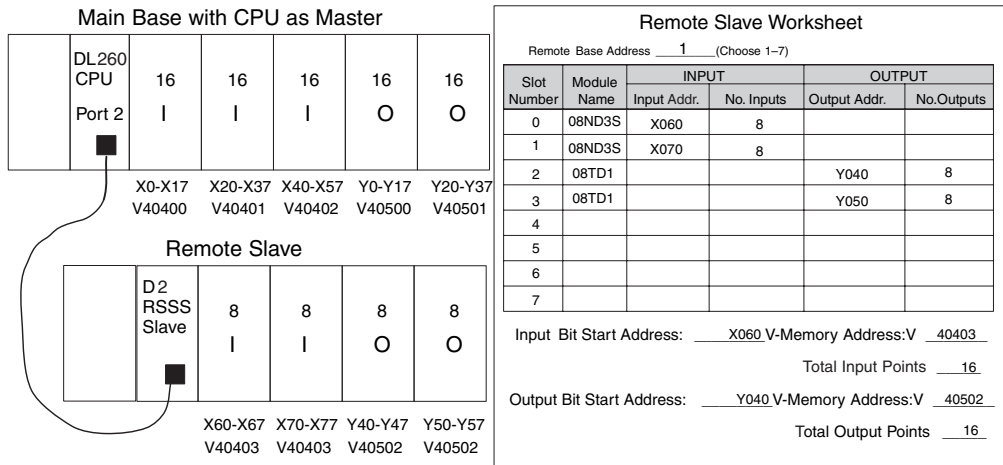


DirectSOFT



Consider the simple system featuring Remote I/O shown below. The DL250-1 or DL260's built-in Remote I/O channel connects to one slave base, which we will assign a station address=1. The baud rates on the master and slave will be 38.4KB.

We can map the remote I/O points as any type of I/O point, simply by choosing the appropriate range of V-memory. Since we have plenty of standard I/O addresses available (X and Y), we will have the remote I/O points start at the next X and Y addresses after the main base points (X60 and Y40, respectively).



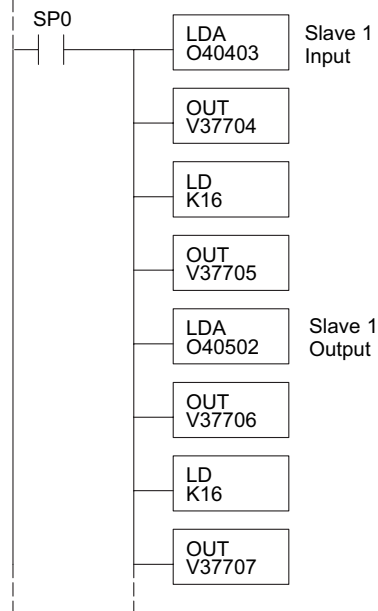
Remote I/O Setup Program

Using the Remote Slave Worksheet shown above can help organize our system data in preparation for writing our ladder program (a blank full-page copy of this worksheet is in the Remote I/O Manual). The four key parameters we need to place in our Remote I/O configuration table are in the lower right corner of the worksheet. You can determine the address values by using the memory map given at the end of Chapter 3, CPU Specifications and Operation.

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

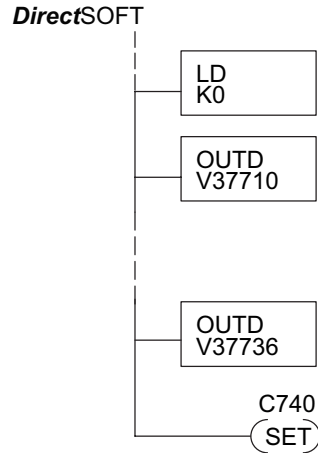
The next page covers the remainder of the required program to get this remote I/O link up and running.

DirectSOFT



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

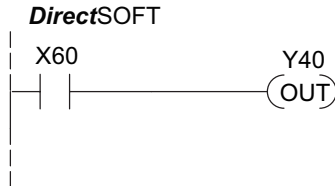
We continue our set-up program from the previous page by adding a segment which fills the remainder of the table with zeros. The example to the right fills zeros for slave numbers 2–7, which do not exist in our example system.



On the last rung in the example program above, we set a special relay contact C740. This particular contact indicates to the CPU the ladder program has finished specifying a remote I/O system. At that moment, the CPU begins remote I/O communications. Be sure to include this contact after any Remote I/O set-up program.

Remote I/O Test Program

Now we can verify the remote I/O link and set-up program operation. A simple quick check can be done with one rung of ladder, shown to the right. It connects the first input of the remote base with the first output. After placing the PLC in RUN mode, we can go to the remote base and activate its first input. Then its first output should turn on.



Network Connections to Modbus and *DirectNET*

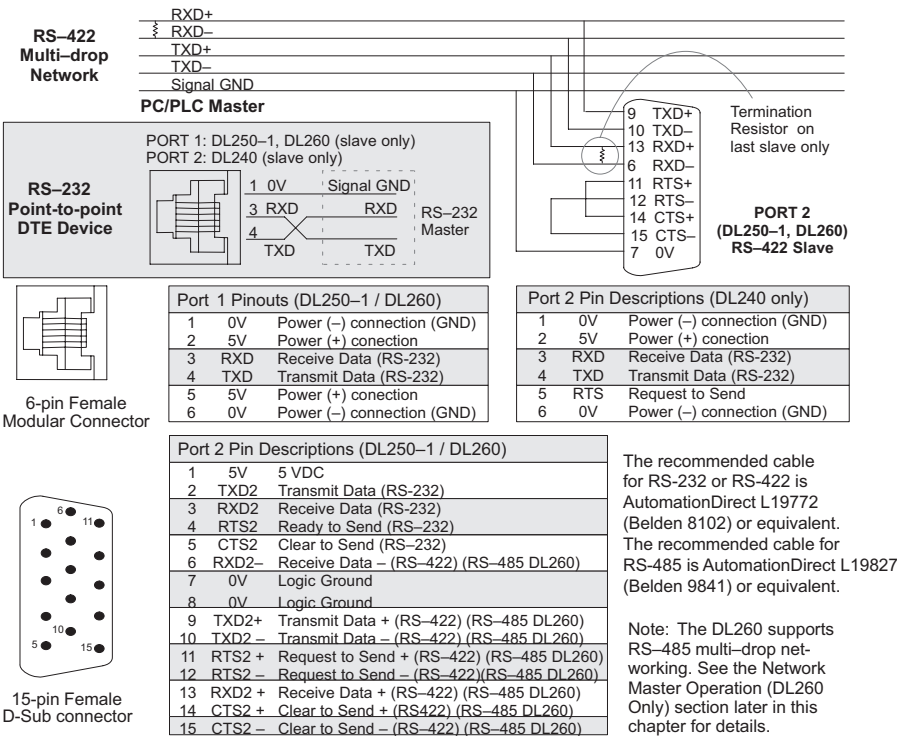
Configuring Port 2 For *DirectNET*

This section describes how to configure the CPU's built-in networking ports for either Modbus or *DirectNET*. This will allow you to connect the DL205 PLC system directly to Modbus networks using the RTU protocol, or to other devices on a *DirectNET* network. For more details on *DirectNET*, order our *DirectNET* manual, part number DA-DNET-M.

Configuring Port 2 For Modbus RTU

Modbus hosts system on the network must be capable of issuing the Modbus commands to read or write the appropriate data. For details on the Modbus protocol, please refer to the Gould Modbus Protocol reference Guide (P1-MBUS-300 Rev. J). In the event a more recent version is available, check with your Modbus supplier before ordering the documentation.

You will need to determine whether the network connection is a 3-wire RS-232 type, or a 5-wire RS-422 type. Normally, the RS-232 signals are used for shorter distance (15 meters (50 feet) maximum) communications between two devices. RS-422 signals are for longer distance (1000 meters (3280ft) maximum) multi-drop networks (from two to 247 devices). Use termination resistors at both ends of RS-422 network wiring, matching the impedance rating of the cable (between 100 and 500 ohms).



Modbus Port Configuration

 230

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

 240

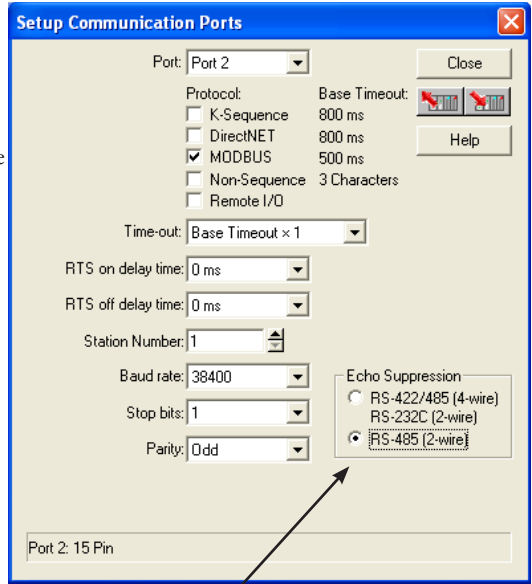
- **Port:** From the port number list box at the top, choose “Port 2.”

 250-1

- **Protocol:** Click the check box to the left of “MODBUS” (use AUX 56 on the HPP, and select “MBUS”), and then you’ll see the dialog box below.

 260

- **Timeout:** The amount of time the port will wait after it sends a message to get a response before logging an error.
- **RTS On Delay Time:** The amount of time between raising the RTS line and sending the data.
- **RTS Off Delay Time:** The amount of time between resetting the RTS line after sending the data.
- **Station Number:** To make the CPU port a Modbus master, choose “1.” The possible range for Modbus slave numbers is from 1 to 247, but the DL250–1 and DL260 WX and RX network instructions used in Master mode will access only slaves 1 to 90. Each slave must have a unique number. At powerup, the port is automatically a slave, unless and until the DL250–1 or DL260 executes ladder logic network instructions which use the port as a master. Thereafter, the port reverts back to slave mode until ladder logic uses the port again.
- **Baud Rate:** The available baud rates include 300, 600, 900, 2400, 4800, 9600, 19200, and 38400 baud. Choose a higher baud rate initially, reverting to lower baud rates if you experience data errors or noise problems on the network. Important: You must configure the baud rates of all devices on the network to the same value. Refer to the appropriate product manual for details.
- **Stop Bits:** Choose 1 or 2 stop bits for use in the protocol.
- **Parity:** Choose none, even, or odd parity for error checking.
- **Echo Suppression:** Select the appropriate radio button based on the wiring configuration used on port 2.



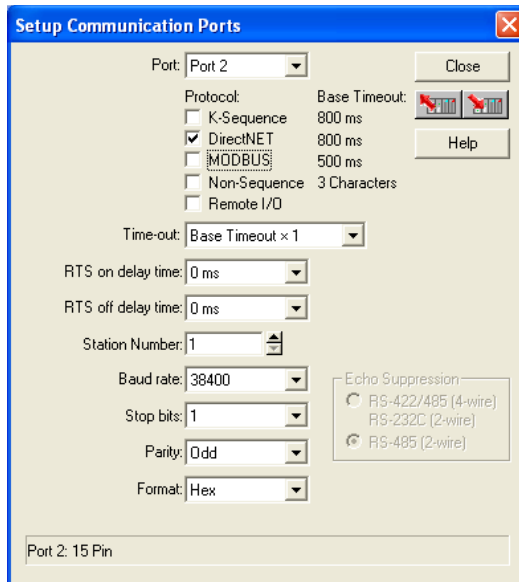
NOTE: The DL250–1 does not support the Echo Suppression feature



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

DirectNET Port Configuration

- 230 In *DirectSOFT*, choose the PLC menu, then Setup, then “Secondary Comm Port.”
- 240 • **Port:** From the port number list box, choose “Port 2.”
- 250-1 • **Protocol:** Click the check box to the left of “*DirectNET*” (use AUX 56 on the HPP, then select
- 260 “DNET”), and then you’ll see the dialog box below.



- **Timeout:** The amount of time the port will wait after it sends a message to get a response before logging an error.
- **RTS On Delay Time:** The amount of time between raising the RTS line and sending the data.
- **RTS Off Delay Time:** The amount of time between resetting the RTS line after sending the data.
- **Station Number:** To make the CPU port a *DirectNET* master, choose “1”. The allowable range for *DirectNET* slaves is from 1 to 90 (each slave must have a unique number). At powerup, the port is automatically a slave, unless and until the DL250–1 or DL260 executes ladder logic instructions which attempt to use the port as a master. Thereafter, the port reverts back to slave mode until ladder logic uses the port again.
- **Baud Rate:** The available baud rates include 300, 600, 900, 2400, 4800, 9600, 19200, and 38400 baud. Choose a higher baud rate initially, reverting to lower baud rates if you experience data errors or noise problems on the network. Important: You must configure the baud rates of all devices on the network to the same value.
- **Stop Bits:** Choose 1 or 2 stop bits for use in the protocol.
- **Parity:** Choose none, even, or odd parity for error checking.
- **Format:** Choose hex or ASCII formats.



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

Network Slave Operation

- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260

This section describes how other devices on a network can communicate with a CPU port that you have configured as a *DirectNET* slave (DL240/250-1/260) or Modbus slave (DL250-1, DL260). A Modbus host must use the Modbus RTU protocol to communicate with the DL250-1 or DL260 as a slave. The host software must send a Modbus function code and Modbus address to specify a PLC memory location the DL250-1 or DL260 comprehends. The *DirectNET* host uses normal I/O addresses to access applicable DL205 CPU and system. No CPU ladder logic is required to support either Modbus slave or *DirectNET* slave operation.

Modbus Function Codes Supported

- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260

The Modbus function code determines whether the access is a read or a write, and whether to access a single data point or a group of them. The DL250-1 and DL260 support the Modbus function codes described below.

| Modbus Function Code | Function | DL205 Data Types Available |
|----------------------|---|----------------------------|
| 01 | Read a group of coils | Y, C, T, CT |
| 02 | Read a group of inputs | X, SP |
| 05 | Set / Reset a single coil (slave only) | Y, C, T, CT |
| 15 | Set / Reset a group of coils | Y, C, T, CT |
| 03, 04 | Read a value from one or more registers | V |
| 06 | Write a value into a single register (slave only) | V |
| 16 | Write a value into a group of registers | V |

Determining the Modbus Address

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

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

If Your Host Software Requires the Data Type and Address

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

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

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

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



NOTE: For information about the Modbus protocol see www.Modbus.org and select Technical Resources. For more information about the *DirectNET* protocol, order our *DirectNET User Manual, DA-DNET-M*, or download the manual free from our website: www.automationdirect.com. Select Manuals/Docs>Online User Manuals>Misc.>DA-DNET-M

| DL250-1 Memory Type | QTY (Dec.) | PLC Range (Octal) | Modbus Address Range (Decimal) | Modbus Data Type |
|---|--------------|----------------------------------|--------------------------------|------------------|
| For Discrete Data Types Convert PLC Addr. to Dec. + Start of Range + Data Type | | | | |
| Inputs (X) | 512 | X0 – X777 | 2048 – 2560 | Input |
| Special Relays (SP) | 512 | SP0 – SP137 SP320 – SP717 | 3072 – 3167 3280 – 3535 | Input |
| Outputs (Y) | 512 | Y0 – Y777 | 2048 – 2560 | Coil |
| Control Relays (C) | 1024 | C0 – C1777 | 3072 – 4095 | Coil |
| Timer Contacts (T) | 256 | T0 – T377 | 6144 – 6399 | Coil |
| Counter Contacts (CT) | 128 | CT0 – CT177 | 6400 – 6527 | Coil |
| Stage Status Bits (S) | 1024 | S0 – S1777 | 5120 – 6143 | Coil |
| For Word Data Types Convert PLC Addr. to Dec. + Data Type | | | | |
| Timer Current Values (V) | 256 | V0 – V377 | 0 – 255 | Input Register |
| Counter Current Values (V) | 128 | V1000 – V1177 | 512 – 639 | Input Register |
| V-Memory, user data (V) | 3072 4096 | V1400 – V7377 V10000 – V17777 | 768 – 3839 4096 – 8191 | Holding Register |
| V-Memory, system (V) | 256 | V7400 – V7777 | 3480 – 3735 | Holding Register |

| DL260 Memory Type | QTY (Dec.) | PLC Range (Octal) | Modbus Address Range (Decimal) | Modbus Data Type |
|---|-------------|---|--------------------------------|------------------|
| For Discrete Data Types Convert PLC Addr. to Dec. + Start of Range + Data Type | | | | |
| Inputs (X) | 1024 | X0 – X1777 | 2048 – 3071 | Input |
| Remote Inputs (GX) | 2048 | GX0 – GX3777 | 3840 – 18431 | Input |
| Special Relays (SP) | 512 | SP0 – SP777 | 3072 – 3583 | Input |
| Outputs (Y) | 1024 | Y0 – Y777 | 2048 – 3071 | Coil |
| Remote Outputs (GY) | 2048 | GY0 – GY3777 | 18432 – 20479 | Coil |
| Control Relays (C) | 2048 | C0 – C377 | 3072 – 5159 | Coil |
| Timer Contacts (T) | 256 | T0 – T177 | 6144 – 6399 | Coil |
| Counter Contacts (CT) | 256 | CT0 – CT177 | 6400 – 6655 | Coil |
| Stage Status Bits (S) | 1024 | S0 – S777 | 5120 – 6143 | Coil |
| For Word Data Types Convert PLC Addr. to Dec. + Data Type | | | | |
| Timer Current Values (V) | 256 | V0 – V177 | 0 – 255 | Input Register |
| Counter Current Values (V) | 256 | V1000 – V1177 | 512 – 767 | Input Register |
| V-Memory, user data (V) | 14.6K | V400 – V777 V1400 – V7377 V10000 – V35777 | 1024 – 2047 | Holding Register |
| V-Memory, system (V) | 256 1024 | V7400 – V7777 V36000 – V37777 | 3480 – 4095 15360 – 16383 | Holding Register |

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

Example 1: V2100

Find the Modbus address for User V location V2100.

1. Find V memory in the table.
2. Convert V2100 into decimal (1089).
3. Use the Modbus data type from the table.

| | | | | |
|----------------------------|------|---------------|-------------|------------------|
| Timer Current Values (V) | 128 | V0 - V177 | 0 - 127 | Input Register |
| Counter Current Values (V) | 128 | V1000 - V1177 | 512 - 639 | Input Register |
| V Memory, user data (V) | 1024 | V2000 - V3777 | 1024 - 2047 | Holding Register |

PLC Address (Dec.) + Data Type

V2100 = 1088 decimal

1088 + Hold. Reg. = **Holding Reg. 1089**

Example 2: Y20

Find the Modbus address for output Y20.

1. Find Y outputs in the table.
2. Convert Y20 into decimal (16).
3. Add the starting address for the range (2049).
4. Use the Modbus data type from the table.

| | | | | |
|---------------------|-----|-----------|-------------|------|
| Outputs (Y) | 320 | Y0 - Y477 | 2049 - 2367 | Coil |
| Control Relays (CR) | 256 | C0 - C377 | 3072 - 3551 | Coil |

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

Y20 = 16 decimal

16 + 2049 + Coil = **Coil 2065**

Example 3: T10 Current Value

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

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

| | | | | |
|----------------------------|-----|---------------|-----------|----------------|
| Timer Current Values (V) | 128 | V0 - V177 | 0 - 128 | Input Register |
| Counter Current Values (V) | 128 | V1000 - V1177 | 512 - 639 | Input Register |

PLC Address (Dec.) + Data Type

T10 = 8 decimal

8 + Input Reg. = **Input Reg. 9**

Example 4: C54

Find the Modbus address for Control Relay C54.

1. Find Control Relays in the table.
2. Convert C54 into decimal (44).
3. Add the starting address for the range (3073).
4. Use the Modbus data type from the table.

| | | | | |
|--------------------|-----|-----------|-------------|------|
| Outputs (Y) | 320 | Y0 - Y477 | 2048 - 2367 | Coil |
| Control Relays (C) | 256 | C0 - C377 | 3073 - 3551 | Coil |

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

C54 = 44 decimal

44 + 3073 + Coil = **Coil 3117**

If Your Modbus Host Software Requires an Address ONLY

Some host software does not allow you to specify the Modbus data type and address. Instead, you specify an address only. This method requires another step to determine the address, but it is not difficult. Basically, Modbus separates the data types by address ranges as well. So this means an address alone can actually describe the type of data and location. This is often referred to as “adding the offset.” One important thing to remember here is that two different addressing modes may be available in your host software package. These are:

- 484 Mode
- 584/984 Mode

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

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

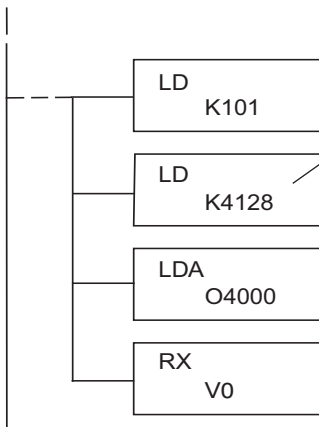
In either case, you basically convert the PLC octal address to decimal and add the appropriate Modbus addresses (as required). The table below shows the exact equation used for each group of data.

| Discrete Data Types | | | | |
|-----------------------|-------------------|--------------------------|------------------------------|------------------|
| DL260 Memory Type | PLC Range (Octal) | Address Range (484 Mode) | Address Range (584/984 Mode) | Modbus Data Type |
| Global Inputs (GX) | GX0 – GX1746 | 1001 – 1999 | 10001 – 10999 | Input |
| | GX1747 – GX3777 | --- | 11000 – 12048 | Input |
| Inputs (X) | X0 – X1777 | --- | 12049 – 13072 | Input |
| Special Relays (SP) | SP0 – SP777 | --- | 13073 – 13584 | Input |
| Global Outputs (GY) | GY0 – GY3777 | 1 – 2048 | 1 – 2048 | Output |
| Outputs (Y) | Y0 – Y1777 | 2049 – 3072 | 2049 – 3072 | Output |
| Control Relays (C) | C0 – C3777 | 3073 – 5120 | 3073 – 5120 | Output |
| Timer Contacts (T) | T0 – T377 | 6145 – 6400 | 6145 – 6400 | Output |
| Counter Contacts (CT) | CT0 – CT377 | 6401 – 6656 | 6401 – 6656 | Output |
| Stage Status Bits (S) | S0 – S1777 | 5121 – 6144 | 5121 – 6144 | Output |

| Word Data Types | | | |
|-----------------------|-------------------|---------------------------|-------------------------------|
| Registers | PLC Range (Octal) | Input/Holding (484 Mode)* | Input/Holding (585/984 Mode)* |
| V-Memory (Timers) | V0 – V377 | 3001/4001 | 30001/40001 |
| V-Memory (Counters) | V1000 – V1177 | 3513/4513 | 30513/40513 |
| V-Memory (Data Words) | V1200 – V1377 | 3641/4641 | 30641/40641 |
| | V1400 – V1746 | 3769/4769 | 30769/40769 |
| | V1747 – V1777 | --- | 31000/41000 |
| | V2000 – V7377 | --- | 41025 |
| | V10000 – V17777 | --- | 44097 |

*Modbus: Function 04

The DL-250 supports **function 04** read input register (**Address 30001**). To use function 04, put the number '4' into the most significant position (4xxx) when defining the number of bytes to read. Four digits must be entered for the instruction to work properly with this mode.



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

Refer to your PLC user manual for the correct memory size of your PLC. Some of the addresses shown above might not pertain to your particular CPU.

For an automated Modbus/Koyo address conversion utility, search and download the file **modbus_conversion.xls** from the www.automationdirect.com website.

Example 1: V2100 584/984 Mode

Find the Modbus address for User V location V2100.

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

PLC Address (Dec.) + Mode Address

V2100 = 1088 decimal

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

| For Word Data Types... | PLC Address (Dec.) | + | Appropriate Mode Address |
|---------------------------|--------------------|-------------|---------------------------|
| Timer Current Value (V) | 128 V0 - V177 | 0 - 127 | 3001 30001 Input Register |
| Counter Current Value (V) | 128 V1000 - V1177 | 512 - 639 | 3001 30001 Input Register |
| V Memory, User Data (V) | 1024 V2000 - V3777 | 1024 - 2047 | 4001 40001 Hold Register |

Example 2: Y20 584/984 Mode

Find the Modbus address for output Y20.

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

PLC Addr. (Dec.) + Start Address + Mode

Y20 = 16 decimal

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

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

Example 3: T10 Current Value 484 Mode

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

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

PLC Address (Dec.) + Mode Address

TA10 = 8 decimal

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

| For Word Data Types... | PLC Address (Dec.) | + | Appropriate Mode Address |
|---------------------------|--------------------|-------------|---------------------------|
| Timer Current Value (V) | 128 V0 - V177 | 0 - 127 | 3001 30001 Input Register |
| Counter Current Value (V) | 128 V1000 - V1177 | 512 - 639 | 3001 30001 Input Register |
| V Memory, User Data (V) | 1024 V2000 - V3777 | 1024 - 2047 | 4001 40001 Hold Register |

Example 4: C54 584/984 Mode

Find the Modbus address for Control Relay C54.

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

PLC Addr. (Dec.) + Start Address + Mode

C54 = 44 decimal

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

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

Determining the DirectNET Address

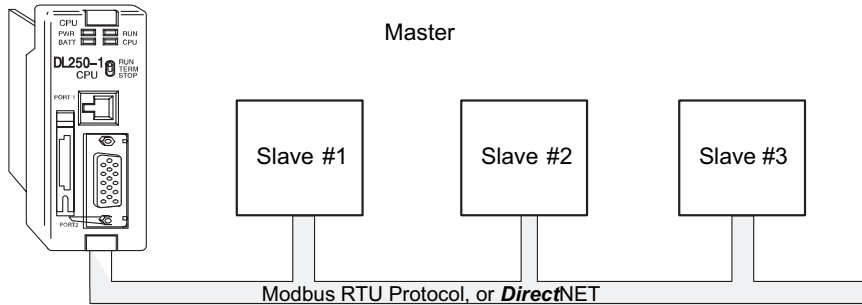
Addressing the memory types for *DirectNET* slaves is very easy. Use the ordinary native address of the slave device itself. To access a slave PLC's memory address V2000 via *DirectNET*, for example, the network master will request V2000 from the slave.

- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260

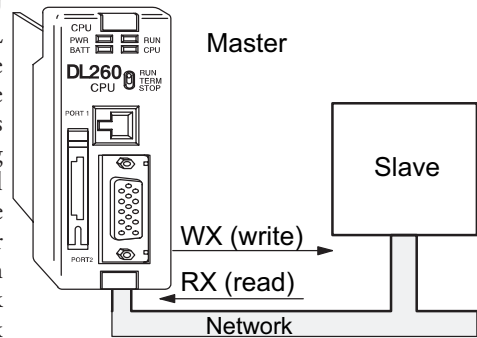
Network Master Operation

- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260

This section describes how the DL250-1 and DL260 can communicate on a Modbus or *DirectNET* network as a master. For Modbus networks, it uses the Modbus RTU protocol, which must be interpreted by all the slaves on the network. Both Modbus and *DirectNET* are single master/multiple slave networks. The master is the only member of the network that can initiate requests on the network. This section teaches you how to design the required ladder logic for network master operation.



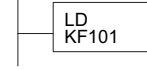
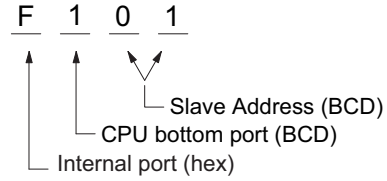
When using the DL250-1 or DL260 CPU as the master station, you use simple RLL instructions to initiate the requests. The WX instruction initiates network write operations, and the RX instruction initiates network read operations. Before executing either the WX or RX commands, we will need to load data related to the read or write operation onto the CPU's accumulator stack. When the WX or RX instruction executes, it uses the information on the stack combined with data in the instruction box to completely define the task, which goes to the port.



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

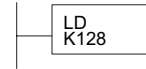
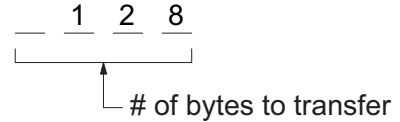
Step 1: Identify Master Port # and Slave

The first Load (LD) instruction identifies the communications port number on the network master (DL250-1/260) and the address of the slave station. This instruction can address up to 99 Modbus slaves, or 90 *DirectNET* slaves. The format of the word is shown to the right. The “F1” in the upper byte indicates the use of the bottom port of the DL250-1/260 PLC, port number 2. The lower byte contains the slave address number in BCD (01 to 99).



Step 2: Load Number of Bytes to Transfer

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



The number of bytes specified also depends on the type of data you want to obtain. For example, the DL205 Input points can be accessed by V-memory locations or as X input locations. However, if you only want X0 – X27, you’ll have to use the X input data type because the V-memory locations can only be accessed in 2-byte increments. The following table shows the byte ranges for the various types of *DirectLOGIC™* products.

| DL205/405 Memory | Bits per unit | Bytes |
|---------------------------------|---------------|-------|
| V-memory | 16 | 2 |
| T / C current value | 16 | 2 |
| Inputs (X, SP) | 8 | 1 |
| Outputs (Y, C, Stage, T/C bits) | 8 | 1 |
| Scratch Pad Memory | 8 | 1 |
| Diagnostic Status | 8 | 1 |

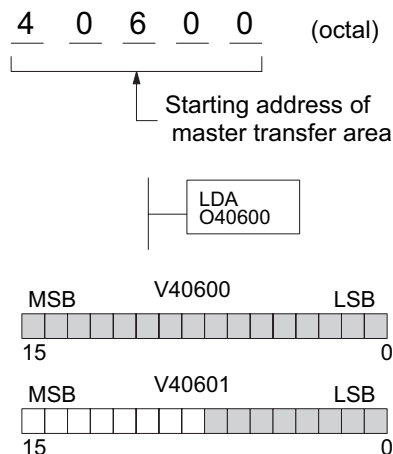
| DL305 Memory | Bits per unit | Bytes |
|---|---------------|-------|
| Data registers | 8 | 1 |
| T / C accumulator | 16 | 2 |
| I/O, internal relays, shift register bits, T/C bits, stage bits | 1 | 1 |
| Scratch Pad Memory | 8 | 2 |
| Diagnostic Status(5 word R/W) | 16 | 10 |

Step 3: Specify Master Memory Area

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

For a WX instruction, the DL250-1/260 CPU sends the number of bytes previously specified from its memory area beginning at the LDA address specified.

For an RX instruction, the DL250-1/260 CPU reads the number of bytes previously specified from the slave, placing the received data into its memory area beginning at the LDA address specified.

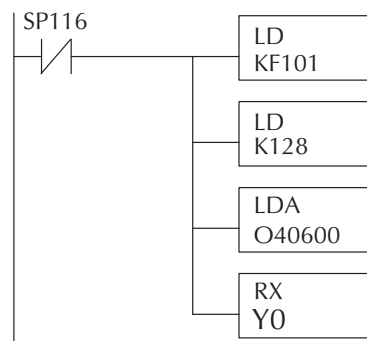


NOTE: Since V-memory words are always 16 bits, you may not always use the whole word. For example, if you only specify 3 bytes and you are reading Y outputs from the slave, you will only get 24 bits of data. In this case, only the 8 least significant bits of the last word location will be modified. The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

Step 4: Specify Slave Memory Area

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

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



DL305 Series CPU Memory Type-to-Modbus Cross Reference

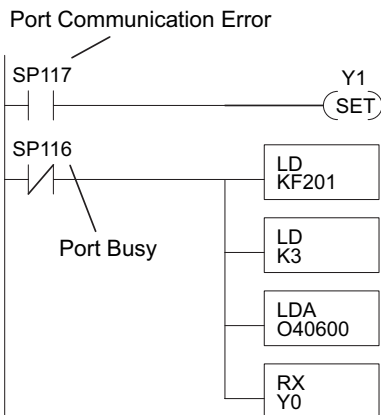
| PLC Memory Type | PLC Base Address | Modbus Base Address | PLC Memory Type | PLC Base Address | Modbus Base Address |
|----------------------------------|------------------|---------------------|---------------------|------------------|---------------------|
| TMR/CNT Current Values | R600 | V0 | TMR/CNT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

Communications from a Ladder Program

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

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

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

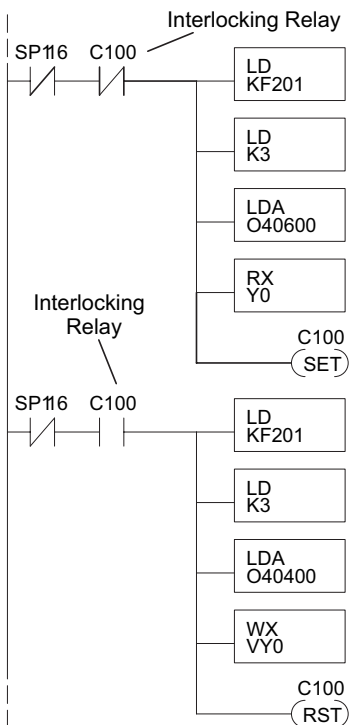


Multiple Read and Write Interlocks

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

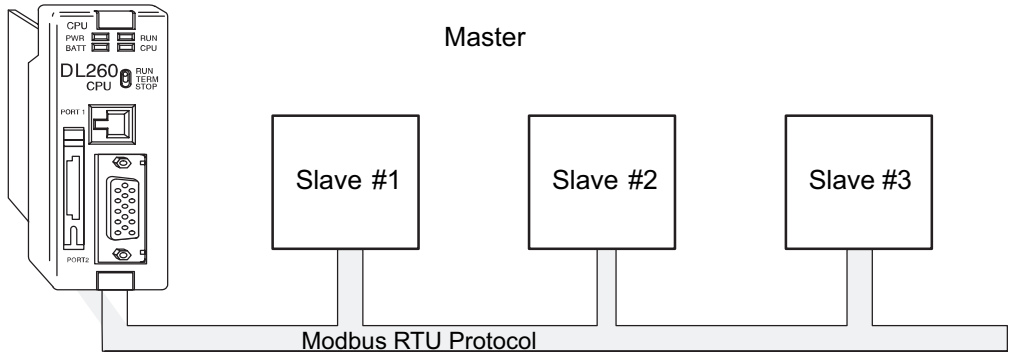
In the example to the right, after the RX instruction is executed, C100 is set. When the port has finished the communication task, the second routine is executed and C100 is reset.

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



Network Modbus RTU Master Operation (DL260 only)

- ☐ 230 This section describes how the DL260 can communicate on a Modbus RTU network as a master using the MRX and MWX read/write instructions. These instructions allow you to enter native Modbus addressing in your ladder logic program with no need to perform octal-to-decimal conversions. Modbus is a single-master, multiple-slave network. The master is the only member of the network that can initiate requests on the network. This section teaches you how to design the required ladder logic for network master operation.
- ☐ 240
- ☐ 250-1
- ☒ 260



Modbus Function Codes Supported

The Modbus function code determines whether the access is a read or a write, and whether to access a single data point or a group of them. The DL260 supports the Modbus function codes described below.

| Modbus Function Code | Function | DL205 Data Types Available |
|----------------------|---|----------------------------|
| 01 | Read a group of coils | Y, C, T, CT |
| 02 | Read a group of inputs | X, SP |
| 05 | Set / Reset a single coil (slave only) | Y, C, T, CT |
| 15 | Set / Reset a group of coils | Y, C, T, CT |
| 03, 04 | Read a value from one or more registers | V |
| 06 | Write a value into a single register (slave only) | V |
| 07 | Read Exception Status | V |
| 08 | Diagnostics | V |
| 16 | Write a value into a group of registers | V |

Modbus Port Configuration

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

✗ 230

- **Port:** From the port number list box at the top, choose “Port 2.”

✗ 240

- **Protocol:** Click the check box to the left of “MODBUS” (use AUX 56 on the HPP, and select “MBUS”), and then you’ll see the dialog box below.

✗ 250-1

✓ 260

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

- **RTS On Delay Time:** The amount of time between raising the RTS line and sending the data.
- **RTS Off Delay Time:** The amount of time between resetting the RTS line after sending the data.
- **Station Number:** For making the CPU port a Modbus master, choose “1.” The possible range for Modbus slave numbers is from 1 to 247. Each slave must have a unique number. At powerup, the port is automatically a slave, unless and until the DL06 executes ladder logic MWX/MRX network instructions which use the port as a master. Thereafter, the port reverts back to slave mode until ladder logic uses the port again.
- **Baud Rate:** The available baud rates include 300, 600, 900, 2400, 4800, 9600, 19200, and 38400 baud. Choose a higher baud rate initially, reverting to lower baud rates if you experience data errors or noise problems on the network. Important: You must configure the baud rates of all devices on the network to the same value. Refer to the appropriate product manual for details.
- **Stop Bits:** Choose 1 or 2 stop bits for use in the protocol.
- **Parity:** Choose none, even, or odd parity for error checking.
- **Echo Suppression:** Select the appropriate radio button based on the wiring configuration used on port 2.

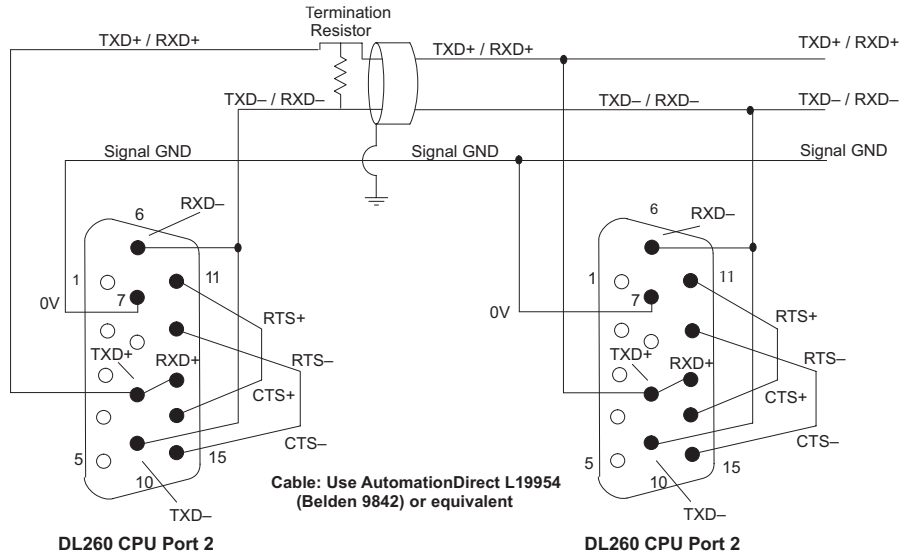


Then click the button indicated to send the Port configuration to the CPU, and click Close.

RS-485 Network (Modbus Only)

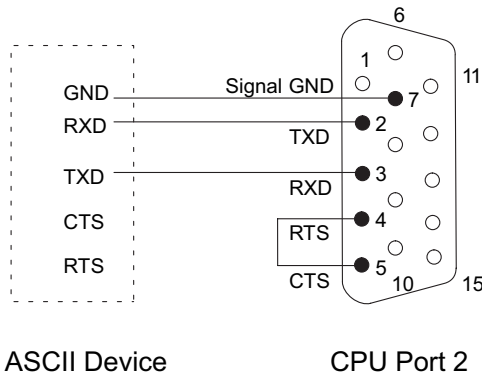
- ☐ 230
- ☐ 240
- ☐ 250-1
- ☒ 260

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



RS-232 Network

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



Port 2 Pin Descriptions (DL260 only)

| | | |
|-----------|--------|-----------------------------------|
| 1 | 5V | 5 VDC |
| 2 | TXD2 | Transmit Data (RS-232) |
| 3 | RXD2 | Receive Data (RS-232) |
| 4 | RTS2 | Ready to Send (RS-232) |
| 5 | CTS2 | Clear to Send (RS-232) |
| 6 | RXD2- | Receive Data - (RS-422/RS-485) |
| 7 | 0V | Logic Ground |
| 8 | 0V | Logic Ground |
| 9 | TXD2+ | Transmit Data + (RS-422/RS-485) |
| 10 | TXD2 - | Transmit Data - (RS-422/RS-485) |
| 11 | RTS2 + | Request to Send + (RS-422/RS-485) |
| 12 | RTS2 - | Request to Send - (RS-422/RS-485) |
| 13 | RXD2 + | Receive Data + (RS-422/RS-485) |
| 14 | CTS2 + | Clear to Send + (RS422/RS-485) |
| 15 | CTS2 - | Clear to Send - (RS-422/RS-485) |

Modbus Read from Network (MRX)

- ☐ 230 The Modbus Read from Network (MRX) instruction is used by the DL260 network master to read a block of data from a connected slave device and to write the data into V-memory addresses within the master. The instruction allows the user to specify the Modbus Function Code, slave station address, starting master and slave memory addresses, number of elements to transfer, Modbus data format and the Exception Response Buffer.
- ☐ 240
- ☐ 250-1
- ☒ 260

- **Port Number:** must be DL260 Port 2 (K2)
- **Slave Address:** specify a slave station address (1–247)
- **Function Code:** the MRX instruction supports the following Modbus function codes:
 - 01 – Read a group of coils
 - 02 – Read a group of inputs
 - 03 – Read holding registers
 - 04 – Read input registers
 - 07 – Read Exception status
- **Start Slave Memory Address:** specifies the starting slave memory address of the data to be read. See the table on the following page.
- **Start Master Memory Address:** specifies the starting memory address in the master where the data will be placed. See the table on the following page.
- **Number of Elements:** specifies how many coils, input, holding registers or input registers will be read. See the table on the following page.
- **Modbus Data Format:** specifies Modbus 584/984 or 484 data format to be used.
- **Exception Response Buffer:** specifies the master memory address where the Exception Response will be placed. See the table on the following page.

MRX Slave Memory Address

| MRX Slave Address Ranges | | |
|----------------------------|----------------------|---|
| Function Code | Modbus Data Format | Slave Address Range(s) |
| 01 – Read Coil | 484 Mode | 1–999 |
| 01 – Read Coil | 584/984 Mode | 1–65535 |
| 02 – Read Input Status | 484 Mode | 1001–1999 |
| 02 – Read Input Status | 584/984 Mode | 10001–19999 (5 digit) or 100001–165535 (6 digit) |
| 03 – Read Holding Register | 484 Mode | 4001–4999 |
| 03 – Read Holding Register | 584/984 | 40001–49999 (5 digit) or 4000001–465535 (6 digit) |
| 04 – Read Input Register | 484 Mode | 3001–3999 |
| 04 – Read Input Register | 584/984 Mode | 300001–39999 (5 digit) or 3000001–365535 (6 digit) |
| 07 – Read Exception Status | 484 and 584/984 Mode | N/A |

MRX Master Memory Addresses

| MRX Master Memory Address Ranges | | |
|----------------------------------|----|-------------|
| Operand Data Type | | DL260 Range |
| Inputs | X | 0–1777 |
| Outputs | Y | 0–1777 |
| Control Relays | C | 0–3777 |
| Stage Bits | S | 0–1777 |
| Timer Bits | T | 0–377 |
| Counter Bits | CT | 0–377 |
| Special Relays | SP | 0–777 |
| V-memory | V | All |
| Global Inputs | GX | 0–3777 |
| Global Outputs | GY | 0–3777 |

MRX Number of Elements

| Number of Elements | | |
|--------------------|---|------------------------------|
| Operand Data Type | | DL260 Range |
| V-memory | V | All (see page 3-56) |
| Constant | K | Bits:1–2000 Registers: 1-125 |

MRX Exception Response Buffer

| Exception Response Buffer | | |
|---------------------------|---|---------------------|
| Operand Data Type | | DL260 Range |
| V-memory | V | All (see page 3-56) |

Modbus Write to Network (MWX)

- ☐ 230 The Modbus Write to Network (MWX) instruction is used to write a block of data from the network master (DL260) memory to Modbus memory addresses within a slave device on the network. The instruction allows the user to specify the Modbus Function Code, slave station address, starting master and slave memory addresses, number of elements to transfer, Modbus data format and the Exception Response Buffer.
- ☐ 240
- ☐ 250-1
- ☒ 260

Port Number: must be DL260 Port 2 (K2).

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

MWX Slave Memory Address

4

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

MWX Master Memory Addresses

| MRX Master Memory Address Ranges | | |
|----------------------------------|----|---------------------|
| Operand Data Type | | DL260 Range |
| Inputs | X | 0–1777 |
| Outputs | Y | 0–1777 |
| Control Relays | C | 0–3777 |
| Stage Bits | S | 0–1777 |
| Timer Bits | T | 0–377 |
| Counter Bits | CT | 0–377 |
| Special Relays | SP | 0–777 |
| V–memory | V | All (see page 3-56) |
| Global Inputs | GX | 0–3777 |
| Global Outputs | GY | 0–3777 |

MWX Number of Elements

| Number of Elements | | |
|--------------------|---|-------------------------------|
| Operand Data Type | | DL260 Range |
| V–memory | V | All (see page 3-56) |
| Constant | K | Bits: 1–2000 Registers: 1-125 |

MWX Exception Response Buffer

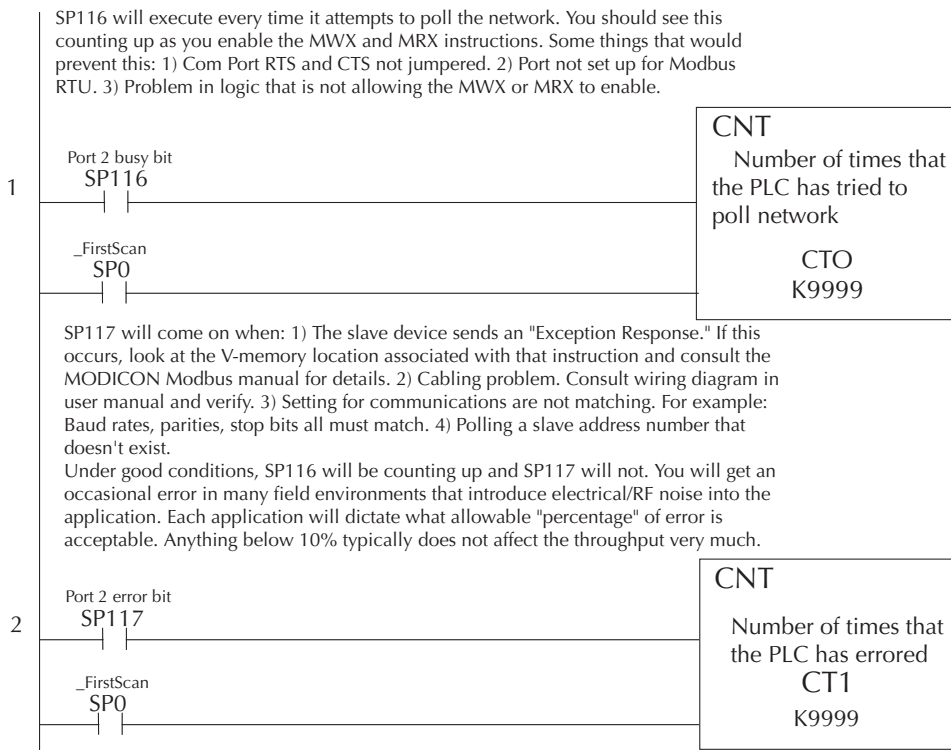
| Exception Response Buffer | | |
|---------------------------|---|---------------------|
| Operand Data Type | | DL260 Range |
| V–memory | V | All (see page 3-56) |

MRX/MWX Example in *DirectSOFT*

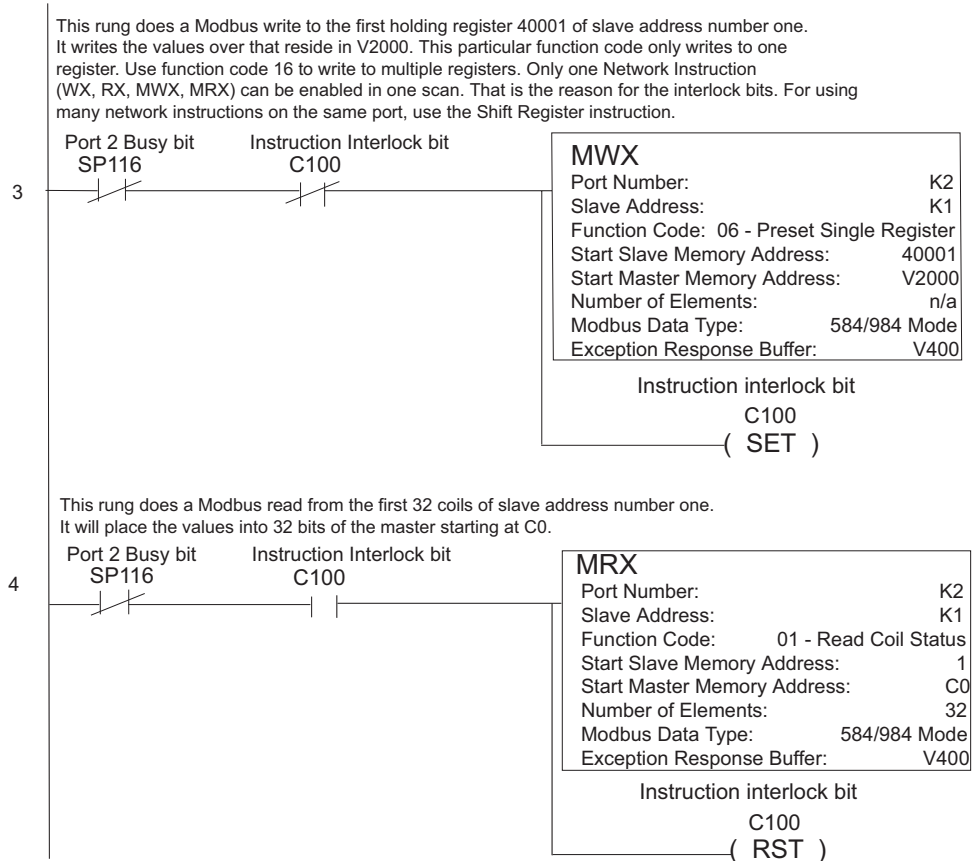
DL260 port 2 has two Special Relay contacts associated with it (see Appendix D for comm port special relays). One indicates "Port busy"(SP116), and the other indicates "Port Communication Error"(SP117). The "Port Busy" bit is on while the PLC communicates with the slave. When the bit is off, the program can initiate the next network request. The "Port Communication Error" bit turns on when the PLC has detected an error and use of this bit is optional. When used, it should be ahead of any network instruction boxes since the error bit is reset when an MRX or MWX instruction is executed. Typically, network communications will last longer than one CPU scan. The program must wait for the communications to finish before starting the next transaction.

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

Multiple Read and Write Interlocks



If you are using multiple reads and writes in the RLL program, you need to interlock the routines to make sure all the routines are executed. If you don't use the interlocks, then the CPU will only execute the first routine. This is because each port can only handle one transaction at a time. In the example, rungs 3 and 4 show that C100 will get set after the RX instruction has been executed. When the port has finished the communication task, the second routine is executed and C100 is reset. If you're using RLL^{PLUS} Stage Programming, you can put each routine in a separate program stage to ensure proper execution and switch from stage to stage allowing only one of them to be active at a time.



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

Configure the DL260 Port 2 for Non-Sequence

Configuring port 2 on the DL260 for Non-Sequence allows the CPU to use port 2 to either read or write raw ASCII strings using the ASCII instructions. See the ASCII In/Out instructions and the PRINT instruction in chapter 5.

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

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

- **Timeout:** Amount of time the port will wait after it sends a message to get a response before logging an error.
- **RTS On Delay Time:** The amount of time between raising the RTS line and sending the data.
- **RTS Off Delay Time:** The amount of time between resetting the RTS line after sending the data.
- **Data Bits:** Select either 7-bits or 8-bits to match the number of data bits specified for the connected devices.
- **Baud Rate:** The available baud rates include 300, 600, 900, 2400, 4800, 9600, 19200, and 38400 baud. Choose a higher baud rate initially, reverting to lower baud rates if you experience data errors or noise problems on the network. Important: You must configure the baud rates of all devices on the network to the same value. Refer to the appropriate product manual for details.
- **Stop Bits:** Choose 1 or 2 stop bits to match the number of stop bits specified for the connected devices.
- **Parity:** Choose none, even, or odd parity for error checking. Be sure to match the parity specified for the connected devices.
- **Memory Address:** Starting V-memory address for ASCII In data storage.

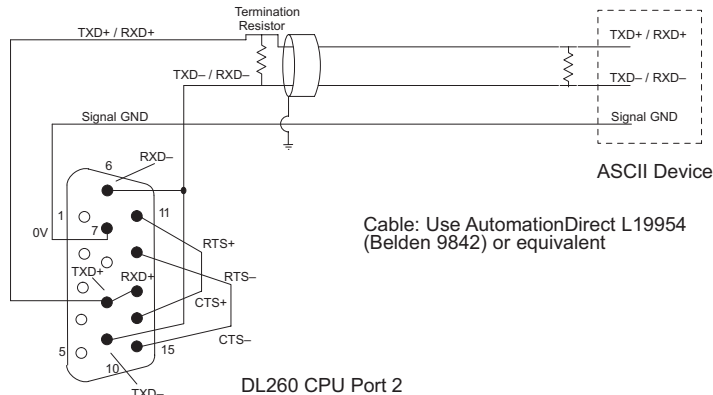
- **XON/XOFF Flow Control:** When this function is enabled, the PLC will send data (PRINT command) until it receives a XOFF (0x13) Pause transmission command. It will continue to wait until it then sees a XON (0x11) Resume transmission command. This selection is only available when the “Non-Sequence(ASCII)” option has been selected and only functions when the PLC is sending data (not receiving with AIN command).
- **RTS Flow Control:** When this function is enabled, the PLC will assert the RTS signal(s) of the port and wait to see the CTS signal(s) go true before sending data (PRINT command). This selection is only available when the “Non-Sequence(ASCII)” option has been selected and only functions when the PLC is sending data (not receiving with AIN command).
- **Echo Suppression:** Select the appropriate radio button based on the wiring configuration used on port 2.



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

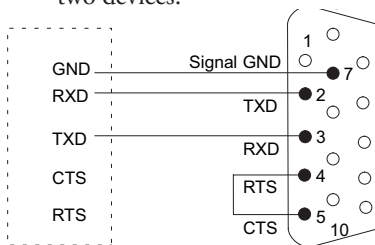
RS-485 Network

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



RS-232 Network

RS-232 signals are used for shorter distances (15 meters maximum) and limited to communications between two devices.



ASCII Device

CPU Port

Port 2 Pin Descriptions (DL260 only)

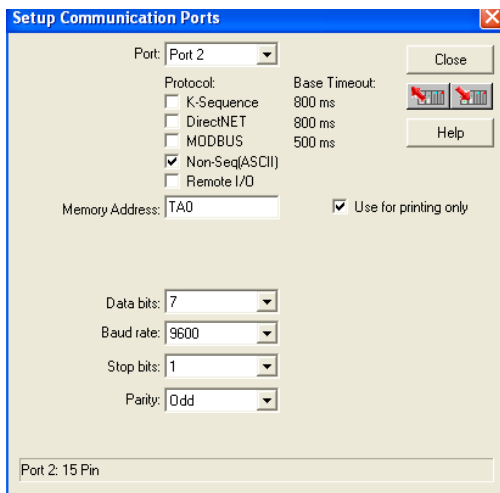
| Pin | Signal | Description |
|-----|--------|-----------------------------------|
| 1 | 5V | 5 VDC |
| 2 | TXD2 | Transmit Data (RS-232) |
| 3 | RXD2 | Receive Data (RS-232) |
| 4 | RTS2 | Ready to Send (RS-232) |
| 5 | CTS2 | Clear to Send (RS-232) |
| 6 | RXD2- | Receive Data - (RS-422/RS-485) |
| 7 | 0V | Logic Ground |
| 8 | 0V | Logic Ground |
| 9 | TXD2+ | Transmit Data + (RS-422/RS-485) |
| 10 | TXD2 - | Transmit Data - (RS-422/RS-485) |
| 11 | RTS2 + | Request to Send + (RS-422/RS-485) |
| 12 | RTS2 - | Request to Send - (RS-422/RS-485) |
| 13 | RXD2 + | Receive Data + (RS-422/RS-485) |
| 14 | CTS2 + | Clear to Send + (RS-422/RS-485) |
| 15 | CTS2 - | Clear to Send - (RS-422/RS-485) |

Configure the DL250-1 Port 2 for Non-Sequence

Configuring port 2 on the DL250-1 for Non-Sequence enables the CPU to use the PRINT instruction to print embedded text or text/data variable message from port 2. See the PRINT instruction in chapter 5.

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

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



- **Memory Address:** Choose a V-memory address to use as the starting location for the port set-up parameters listed below. This location is the start of protocol memory buffer. It should not be used for other purposes. Buffer size = 2 + (Max receiving data size) / 2 or to allocate the maximum allowable space buffer size = 66 Words (for example V2000-V2102).
- **Use For Printing Only:** Check the box to enable the port settings described below. Match the settings to the connected device.
- **Data Bits:** Select either 7-bits or 8-bits to match the number of data bits specified for the connected device.
- **Baud Rate:** The available baud rates include 300, 600, 900, 2400, 4800, 9600, 19200, and 38400 baud. Choose a higher baud rate initially, reverting to lower baud rates if you experience data errors or noise problems on the network. Important: You must configure the baud rates of all devices on the network to the same value. Refer to the appropriate product manual for details.
- **Stop Bits:** Choose 1 or 2 stop bits to match the number of stop bits specified for the connected device.
- **Parity:** Choose none, even, or odd parity for error checking. Be sure to match the parity specified for the connected device.



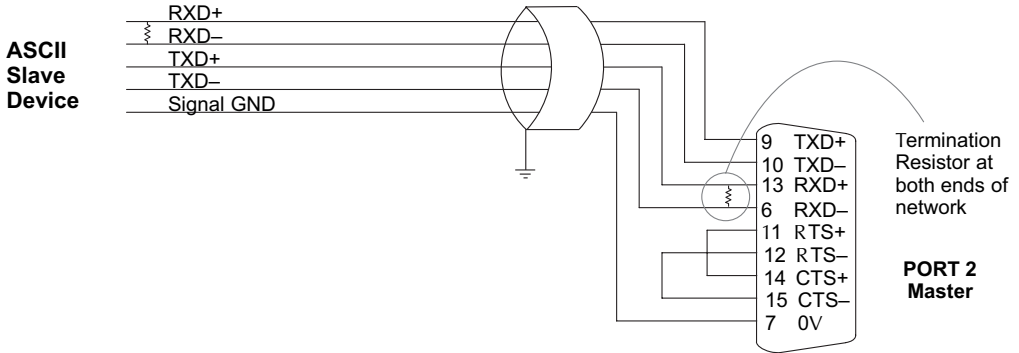
Then click the button indicated to send the Port configuration to the CPU, and click Close.

RS-422 Network

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



NOTE: For RS-422 cabling, we recommend AutomationDirect L19853 (Belden 8103) or equivalent.

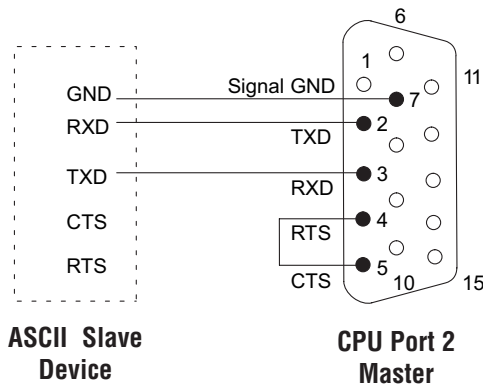


RS-232 Network

RS-232 signals are used for shorter distances (15 meters maximum) and limited to communications between two devices.



NOTE: For RS-232 cabling, we recommend AutomationDirect L19772 (Belden 8102) or equivalent.



| Port 2 Pin Descriptions (DL250-1) | | |
|-----------------------------------|--------|----------------------------|
| 1 | 5V | 5 VDC |
| 2 | TXD2 | Transmit Data (RS-232) |
| 3 | RXD2 | Receive Data (RS-232) |
| 4 | RTS2 | Ready to Send (RS-232) |
| 5 | CTS2 | Clear to Send (RS-232) |
| 6 | RXD2- | Receive Data - (RS-422) |
| 7 | 0V | Logic Ground |
| 8 | 0V | Logic Ground |
| 9 | TXD2+ | Transmit Data + (RS-422) |
| 10 | TXD2 - | Transmit Data - (RS-422) |
| 11 | RTS2 + | Request to Send + (RS-422) |
| 12 | RTS2 - | Request to Send - (RS-422) |
| 13 | RXD2 + | Receive Data + (RS-422) |
| 14 | CTS2 + | Clear to Send + (RS422) |
| 15 | CTS2 - | Clear to Send - (RS-422) |

Notes

RLL AND INTELLIGENT BOX INSTRUCTIONS



In This Chapter:

| | |
|--|-------|
| Introduction..... | 5-2 |
| Using Boolean Instructions..... | 5-5 |
| Boolean Instructions | 5-10 |
| Comparative Boolean..... | 5-27 |
| Immediate Instructions..... | 5-33 |
| Timer, Counter and Shift Register Instructions | 5-41 |
| Accumulator/Stack Load and Output Data Instructions..... | 5-53 |
| Logical Instructions (Accumulator) | 5-71 |
| Math Instructions..... | 5-88 |
| Transcendental Functions (DL260 only) | 5-121 |
| Bit Operation Instructions | 5-123 |
| Number Conversion Instructions (Accumulator)..... | 5-130 |
| Table Instructions..... | 5-144 |
| Clock/Calendar Instructions | 5-175 |
| CPU Control Instructions..... | 5-177 |
| Program Control Instructions | 5-179 |
| Interrupt Instructions | 5-187 |
| Intelligent I/O Instructions | 5-191 |
| Network Instructions..... | 5-193 |
| Message Instructions..... | 5-197 |
| Modbus RTU Instructions (DL260) | 5-205 |
| ASCII Instructions (DL260) | 5-211 |
| Intelligent Box (IBox) Instructions (DL250-1/DL260 Only) | 5-230 |

Introduction

The DL205 CPUs offer a wide variety of instructions to perform many different types of operations. Several instructions are not available in all of the CPUs. This chapter shows you how to use these individual instructions. There are two ways to quickly find the instruction you need:

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

5

| Instruction | | Page |
|-------------|---------------------------|------------------|
| ACON | ASCII Constant | 5-199 |
| ACOSR | Arc Cosine Real | 5-122 |
| ACRB | ASCII Clear Buffer | 5-229 |
| ADD | Add BCD | 5-88 |
| ADDB | Add Binary | 5-101 |
| ADDBD | Add Binary Double | 5-102 |
| ADDBS | Add Binary Top of Stack | 5-117 |
| ADD | Add Double BCD | 5-89 |
| ADDF | Add Formatted | 5-109 |
| ADDR | Add Real | 5-90 |
| ADD | Add Top of Stack | 5-113 |
| AEX | ASCII Extract | 5-220 |
| AFIND | ASCII Find | 5-217 |
| AIN | ASCII IN | 5-212 |
| AND | And for contacts or boxes | 5-14, 5-32, 5-71 |
| AND STR | And Store | 5-16 |
| ANDB | And Bit-of-Word | 5-15 |
| ANDD | And Double | 5-72 |
| ANDE | And if Equal | 5-29 |
| ANDF | And Formatted | 5-73 |
| ANDI | And Immediate | 5-35 |
| ANDMOV | And Move | 5-171 |
| ANDN | And Not | 5-14, 5-32 |
| ANDNB | And Not Bit-of-Word | 5-15 |
| ANDND | And Negative Differential | 5-23 |
| ANDNE | And if Not Equal | 5-29 |
| ANDNI | And Not Immediate | 5-35 |
| ANDPD | And Positive Differential | 5-23 |
| ANDS | And Stack | 5-74 |
| ASINR | Arc Sine Real | 5-121 |
| ATANR | Arc Tangent Real | 5-122 |
| ATH | ASCII to Hex | 5-137 |
| ATT | Add to Top of Table | 5-166 |
| BCD | Binary Coded Decimal | 5-131 |
| BCDCPL | Tens Complement | 5-133 |

| Instruction | | Page |
|-------------|----------------------------|-------|
| BIN | Binary | 5-130 |
| BCALL | Block Call (Stage) | 7-27 |
| BEND | Block End (Stage) | 7-27 |
| BLK | Block (Stage) | 7-27 |
| BTOR | Binary to Real | 5-134 |
| CMP | Compare | 5-83 |
| CMPD | Compare Double | 5-84 |
| CMPF | Compare Formatted | 5-85 |
| CMR | Compare Real Number | 5-87 |
| CMPS | Compare Stack | 5-86 |
| CMPV | ASCII Compare | 5-221 |
| CNT | Counter | 5-46 |
| COSR | Cosine Real | 5-121 |
| CV | Converge (Stage) | 7-25 |
| CVJMP | Converge Jump (Stage) | 7-25 |
| DATE | Date | 5-175 |
| DEC | Decrement | 5-100 |
| DECB | Decrement Binary | 5-108 |
| DECO | Decode | 5-129 |
| DEGR | Degree Real Conversion | 5-136 |
| DISI | Disable Interrupts | 5-188 |
| DIV | Divide | 5-97 |
| DIVB | Divide Binary | 5-106 |
| DIVBS | Divide Binary Top of Stack | 5-120 |
| DIVD | Divide Double | 5-98 |
| DIVF | Divide Formatted | 5-112 |
| DIVR | Divide Real Number | 5-99 |
| DIVS | Divide Top of Stack | 5-116 |
| DLBL | Data Label | 5-199 |
| DRUM | Timed Drum | 6-12 |
| EDRUM | Event Drum | 6-14 |
| ENCO | Encode | 5-128 |
| END | End | 5-177 |
| ENI | Enable Interrupts | 5-188 |

| Instruction | | Page |
|-------------|------------------------------|-------|
| FAULT | Fault | 5-197 |
| FDGT | Find Greater Than | 5-152 |
| FILL | Fill | 5-150 |
| FIND | Find | 5-151 |
| FINDB | Find Block | 5-173 |
| FOR | For/Next | 5-180 |
| GOTO | Goto/Label | 5-179 |
| GRAY | Gray Code | 5-141 |
| GTS | Goto Subroutine | 5-182 |
| HTA | Hex to ASCII | 5-138 |
| INC | Increment | 5-100 |
| INCB | Increment Binary | 5-107 |
| INT | Interrupt | 5-187 |
| INV | Invert | 5-132 |
| IRT | Interrupt Return | 5-188 |
| IRTC | Interrupt Return Conditional | 7-188 |
| ISG | Initial Stage | 7-24 |
| JMP | Jump | 5-24 |
| LBL | Label | 5-179 |
| LD | Load | 5-58 |
| LDI | Load Immediate | 5-39 |
| LDIF | Load Immediate Formatted | 5-40 |
| LDA | Load Address | 5-61 |
| LDD | Load Double | 5-59 |
| LDF | Load Formatted | 5-60 |
| LDR | Load Real Number | 5-64 |
| LDX | Load Indexed | 5-62 |
| LDLBL | Load Label | 5-145 |
| LDSX | Load Indexed from Constant | 5-63 |
| MDRMD | Masked Drum Event Discrete | 6-19 |
| MDRMW | Masked Drum Event Word | 6-21 |
| MLR | Master Line Reset | 5-185 |
| MLS | Master Line Set | 5-185 |
| MOV | Move | 5-144 |
| MOVMC | Move Memory Cartridge | 5-145 |
| MRX | Read from MODBUS Network | 5-205 |
| MWX | Write to MODBUS | 5-208 |
| MUL | Multiply | 5-94 |
| MULB | Multiply Binary | 5-105 |
| MULBS | Multiply Binary top of stack | 5-119 |
| MULD | Multiply Double | 5-95 |
| MULF | Multiply Formatted | 5-111 |
| MULR | Multiply Real | 5-96 |
| MULS | Multiply Top of Stack | 5-115 |
| NCON | Numeric Constand | 5-199 |
| NEXT | Next (For/Next) | 5-180 |

| Instruction | | Page |
|-------------|------------------------------|------------------|
| NJMP | Not Jump (Stage) | 7-24 |
| NOP | No Operation | 5-177 |
| NOT | Not | 5-19 |
| OR | Or | 5-12, 5-31, 5-75 |
| OR OUT | Or Out | 5-19 |
| OR OUTI | Or Out Immediate | 5-36 |
| OR STR | Or Store | 5-16 |
| ORB | Or Bit-of-Word | 5-13 |
| ORD | Or Double | 5-76 |
| ORE | Or if Equal | 5-28 |
| ORF | Or Formatted | 5-77 |
| ORI | Or Immediate | 5-34 |
| ORMOV | Or Move | 5-171 |
| ORN | Or Not | 5-12, 5-31 |
| ORNB | Or Not Bit-of-Word | 5-13 |
| ORND | Or Negative Differential | 5-22 |
| ORNE | Or if Not Equal | 5-28 |
| ORNI | Or Not Immediate | 5-34 |
| ORPD | Or Positive Differential | 5-22 |
| ORS | Or Stack | 5-78 |
| OUT | Out | 5-17, 5-65 |
| OUTB | Out Bit-of-Word | 5-18 |
| OUTD | Out Double | 5-66 |
| OUTF | Out Formatted | 5-67 |
| OUTI | Out Immediate | 5-36 |
| OUTIF | Out Immediate Formatted | 5-37 |
| OUTL | Out Least | 5-69 |
| OUTM | Out Most | 5-69 |
| OUTX | Out Indexed | 5-68 |
| PAUSE | Pause | 5-26 |
| PD | Positive Differential | 5-20 |
| POP | Pop | 5-70 |
| PRINT | Print | 5-201 |
| PRINTV | ASCII Print from V-Memory | 5-227 |
| RADR | Radian Real Conversion | 5-136 |
| RD | Read from Intelligent Module | 5-191 |
| RFB | Remove from Bottom of Table | 5-157 |
| RFT | Remove from Top of Table | 5-163 |
| ROTL | Rotate Left | 5-126 |
| ROTR | Rotate Right | 5-127 |
| RST | Reset | 5-24 |
| RSTB | Reset Bit-of-Word | 5-25 |
| RSTBIT | Reset Bit | 5-148 |
| RSTI | Reset Immediate | 5-38 |
| RSTWT | Reset Watch Dog Timer | 5-178 |

| Instruction | | Page |
|-------------|-------------------------------|------------|
| RT | Subroutine Return | 5-182 |
| RTC | Subroutine Return Conditional | 5-182 |
| RTOB | Real to Binary | 5-135 |
| RX | Read from Network | 5-193 |
| SBR | Subroutine (Goto Subroutine) | 5-182 |
| SEG | Segment | 5-140 |
| SET | Set | 5-24 |
| SETB | Set Bit-of-Word | 5-25 |
| SETBIT | Set Bit | 5-148 |
| SETI | Set Immediate | 5-38 |
| SFLDGT | Shuffle Digits | 5-142 |
| SG | Stage | 7-23 |
| SGCNT | Stage Counter | 5-48 |
| SHFL | Shift Left | 5-124 |
| SHFR | Shift Right | 5-125 |
| SINR | Sine Real | 5-121 |
| SQRTR | Square Root Real | 5-122 |
| SR | Shift Register | 5-52 |
| STOP | Stop | 5-177 |
| STR | Store | 5-10, 5-30 |
| STRB | Store Bit-of-Word | 5-11 |
| STRE | Store if Equal | 5-27 |
| STRI | Store Immediate | 5-33 |
| STRN | Store Not | 5-10, 5-30 |
| STRNB | Store Not Bit-of-Word | 5-11 |
| STRND | Store Negative Differential | 5-21 |
| STRNE | Store if Not Equal | 5-27 |
| STRNI | Store Not Immediate | 5-33 |
| STRPD | Store Positive Differential | 5-21 |
| STT | Source to Table | 5-160 |

| Instruction | | Page |
|-------------|------------------------------|-------|
| SUB | Subtract | 5-91 |
| SUBB | Subtract Binary | 5-103 |
| SUBBD | Subtract Binary Double | 5-104 |
| SUBBS | Subtract Binary Top of Stack | 5-118 |
| SUBD | Subtract Double | 5-92 |
| SUBF | Subtract Formatted | 5-110 |
| SUBS | Subtract Top of Stack | 5-114 |
| SUBR | Subtract Real Number | 5-93 |
| SUM | Sum | 5-123 |
| SWAP | Swap Table Data | 5-174 |
| SWAPB | ASCII Swap Bytes | 5-228 |
| TANR | Tangent Real | 5-121 |
| TIME | Time | 5-176 |
| TMR | Timer | 5-42 |
| TMRF | Fast Timer | 5-42 |
| TMRA | Accumulating Timer | 5-44 |
| TMRAF | Fast Accumulating Timer | 5-44 |
| TSHFL | Table Shift Left | 5-169 |
| TSHFR | Table Shift Right | 5-169 |
| TTD | Table to Destination | 5-154 |
| UDC | Up Down Counter | 5-50 |
| VPRINT | ASCII Print to V-Memory | 5-222 |
| WT | Write to Intelligent Module | 5-192 |
| WX | Write to Network | 5-195 |
| XOR | Exclusive Or | 5-79 |
| XORD | Exclusive Or Double | 5-80 |
| XORF | Exclusive Or Formatted | 5-81 |
| XORMOV | Exclusive Or Move | 5-171 |
| XORS | Exclusive Or Stack | 5-82 |

Using Boolean Instructions

Do you ever wonder why so many PLC manufacturers always quote the scan time for a 1K boolean program? Simple, most programs utilize many boolean instructions. These are typically very simple instructions designed to join input and output contacts in various series and parallel combinations. Our *DirectSOFT* programming package is a similar program. It uses graphic symbols to develop a program; therefore, you don't necessarily have to know the instruction mnemonics in order to develop your program.

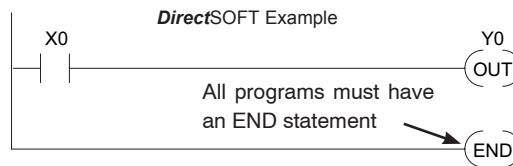
Many of the instructions in this chapter are not program instructions used in *DirectSOFT*, but are implied. In other words, they are not actually keyboard commands but they can be seen in a Mnemonic View of the program once the *DirectSOFT* program has been developed and accepted (compiled). Each instruction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

| | |
|-----|---------|
| DS | Implied |
| HPP | Used |

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

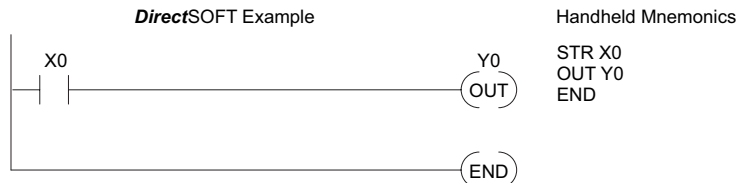
END Statement

All DL205 programs require an END statement as the last instruction. This tells the CPU that this is the end of the program. Normally, any instructions placed after the END statement will not be executed. There are exceptions to this such as interrupt routines, etc. Chapter 5 discusses the instruction set in detail.



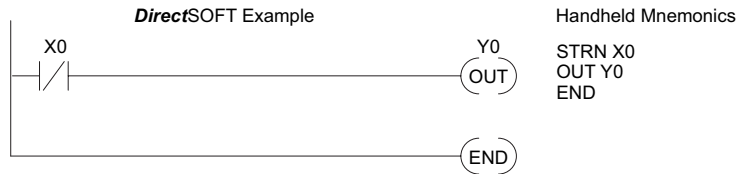
Simple Rungs

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



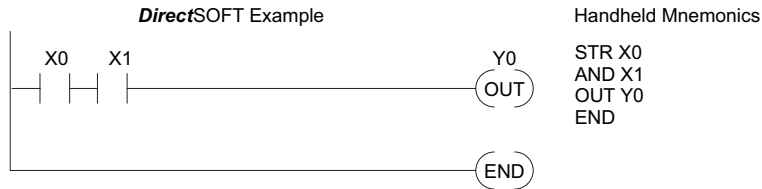
Normally Closed Contact

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



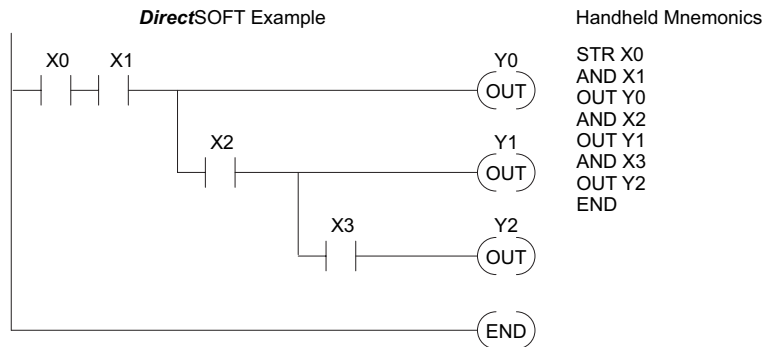
Contacts in Series

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



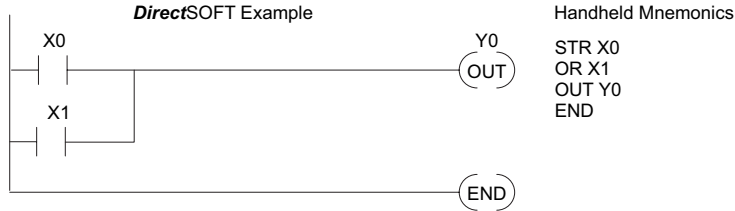
Midline Outputs

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



Parallel Elements

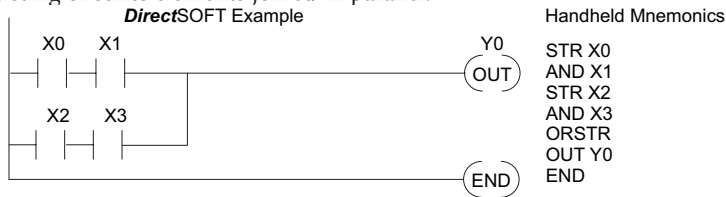
You may also have to join contacts in parallel. The OR instruction allows you to do this. The following example shows two contacts in parallel and a single output coil. The instructions would be STR X0, OR X1, followed by OUT Y0.



5

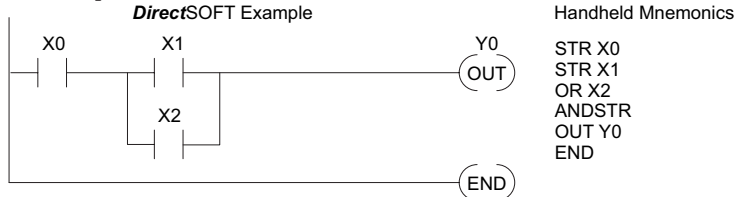
Joining Series Branches in Parallel

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



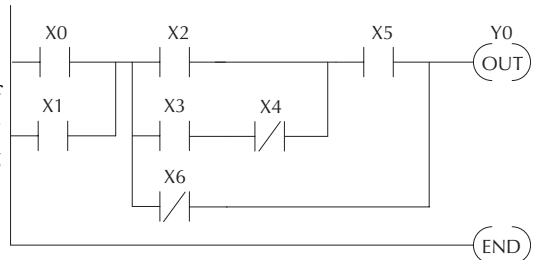
Joining Parallel Branches in Series

You can also join one or more parallel branches in series. The And Store (ANDSTR) instruction allows this operation. The following example shows a simple network with contact branches in series with parallel contacts.



Combination Networks

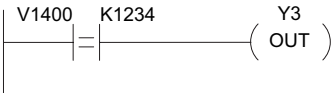
You can combine the various types of series and parallel branches to solve most any application problem. The following example shows a simple combination network.



Comparative Boolean

The DL205 Micro PLCs provide Comparative Boolean instructions that allow you to quickly and easily solve compare two numbers. 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.

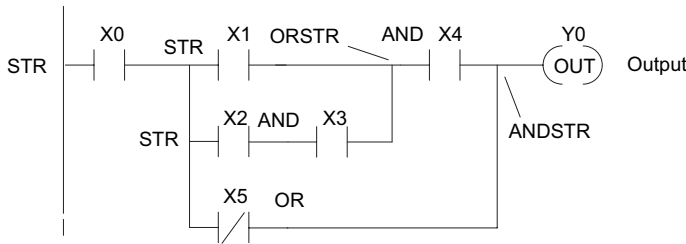
In the example ,when the BCD value in V-memory location V1400 is equal to the constant value 1234, Y3 will energize.



Boolean Stack

There are limits to how many elements you can include in a rung. This is because the DL205 CPUs use an 8-level boolean stack to evaluate the various logic elements. The boolean stack is a temporary storage area that solves the logic for the rung. Each time you enter a STR instruction, the instruction is placed on the top of the boolean stack. Any other STR instructions on the boolean stack are pushed down a level. The ANDSTR, and ORSTR instructions combine levels of the boolean stack when they are encountered. Since the boolean stack is only eight levels, an error will occur if the CPU encounters a rung that uses more than the eight levels of the boolean stack.

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



STR X0

| | |
|---|--------|
| 1 | STR X0 |
| 2 | |
| 3 | |
| 4 | |

STR X1

| | |
|---|--------|
| 1 | STR X1 |
| 2 | STR X0 |
| 3 | |
| 4 | |

STR X2

| | |
|---|--------|
| 1 | STR X2 |
| 2 | STR X1 |
| 3 | STR X0 |
| 4 | |

AND X3

| | |
|---|--------|
| 1 | STR X2 |
| 2 | STR X1 |
| 3 | STR X0 |
| 4 | |

ORSTR

| | |
|---|-------------------|
| 1 | X1 or (X2 AND X3) |
| 2 | STR X0 |
| 3 | |

AND X4

| | |
|---|----------------------------|
| 1 | X4 AND {X1 or (X2 AND X3)} |
| 2 | STR X0 |
| 3 | |

ORNOT X5

| | |
|---|--------------------------------------|
| 1 | NOT X5 OR X4 AND {X1 OR (X2 AND X3)} |
| 2 | STR X0 |
| 3 | |

ANDSTR

| | |
|---|---|
| 1 | X0 AND (NOT X5 or X4) AND {X1 or (X2 AND X3)} |
| 2 | |
| 3 | |

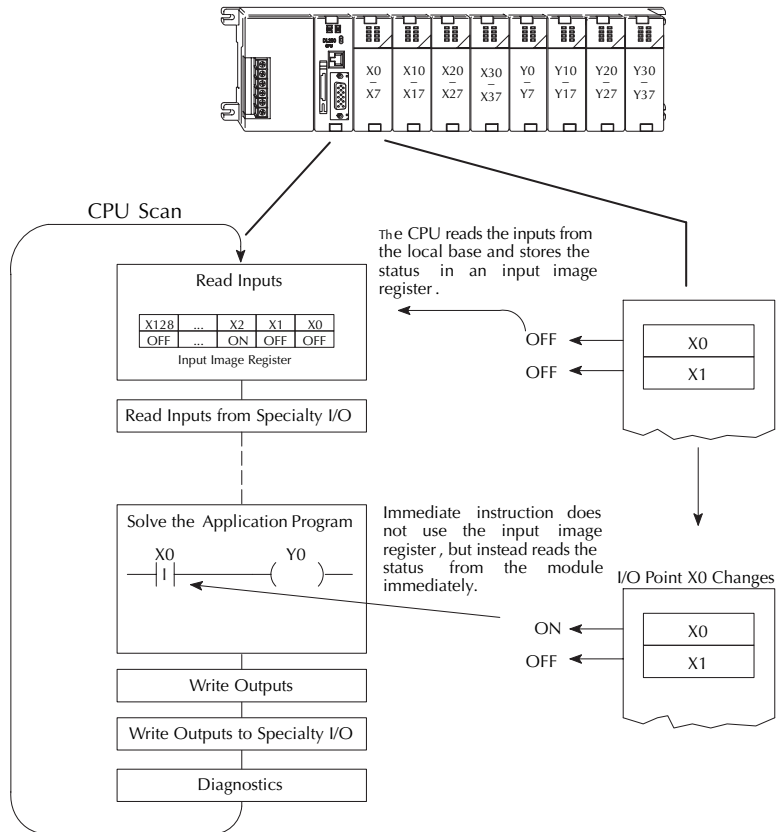
Immediate Boolean

The DL205 Micro PLCs can usually complete an operation cycle in a matter of milliseconds. However, in some applications you may not be able to wait a few milliseconds until the next I/O update occurs. The DL205 PLCs offer immediate input and outputs which are special boolean instructions that allow reading directly from inputs and writing directly to outputs during the program execution portion of the CPU cycle. You may recall that this is normally done during the input or output update portion of the CPU cycle. The immediate instructions take longer to execute because the program execution is interrupted while the CPU reads or writes the I/O point. This function is not normally done until the read inputs or the write outputs portion of the CPU cycle.



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

5



Boolean Instructions

Store (STR)

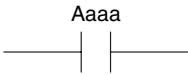
- ✓

230
- ✓

240
- ✓

250-1
- ✓

260
- 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)

- ✓

230
- ✓

240
- ✓

250-1
- ✓

260
- 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.



In the following

| Operand Data Type | | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|----|--------------------|-------------------|---------------|-------------|
| A | | aaa | aaa | aaa | aaa |
| Inputs | X | 0 – 177 | 0 – 477 | 0 – 777 | 0 – 1777 |
| Outputs | Y | 0 – 177 | 0 – 477 | 0 – 777 | 0 – 1777 |
| Control Relays | C | 0 – 377 | 0 – 377 | 0 – 1777 | 0 – 3777 |
| Stage | S | 0 – 377 | 0 – 777 | 0 – 1777 | 0 – 1777 |
| Timer | T | 0 – 77 | 0 – 177 | 0 – 377 | 0 – 377 |
| Counter | CT | 0 – 77 | 0 – 177 | 0 – 177 | 0 – 377 |
| Special Relay | SP | 0 – 117, 540 – 577 | 0 – 137 540 – 617 | 0 – 777 | 0 – 777 |
| Global | GX | – | – | – | 0 –3777 |
| Global | GY | – | – | – | 0 –3777 |

| | |
|-----|------|
| DS | Used |
| HPP | Used |

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



Handheld Programmer Keystrokes

| | | | |
|-----------|---|--------|-----|
| \$ STR | → | B 1 | ENT |
| GX OUT | → | C 2 | ENT |

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



Handheld Programmer Keystrokes

| | | | |
|------------|---|--------|-----|
| SP STRN | → | B 1 | ENT |
| GX OUT | → | C 2 | ENT |

Store Bit-of-Word (STRB)

- ✗

230

✗

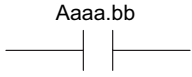
240

✓

250-1

✓

260
- The Store Bit-of-Word instruction begins a new rung or an additional branch in a rung with a normally open contact. Status of the contact will be the same state as the bit referenced in the associated memory location.



Store Not Bit-of-Word (STRNB)

- ✗

230

✗

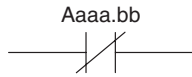
240

✓

250-1

✓

260
- The Store Not instruction begins a new rung or an additional branch in a rung with a normally closed contact. Status of the contact will be opposite the state of the bit referenced in the associated memory location.



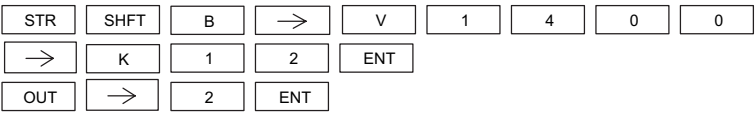
| Operand Data Type | | DL250-1 Range | | DL260 Range | |
|-------------------|----|--------------------------|--------------|--------------------------|--------------|
| | A | aaa | bb | aaa | bb |
| V-memory | B | See memory map page 3-55 | BCD, 0 to 15 | See memory map page 3-56 | BCD, 0 to 15 |
| Pointer | PB | See memory map page 3-55 | BCD, 0 to 15 | See memory map page 3-56 | BCD, 0 to 15 |

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

| | |
|-----|------|
| DS | Used |
| HPP | Used |



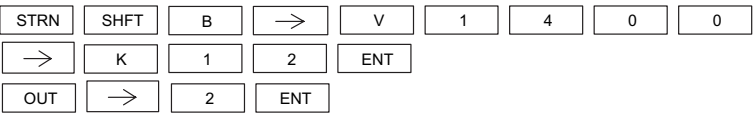
Handheld Programmer Keystrokes



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



Handheld Programmer Keystrokes



Or (OR)

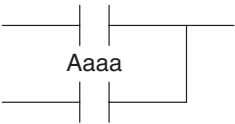
- ✓

230
- ✓

240
- ✓

250-1
- ✓

260
- 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)

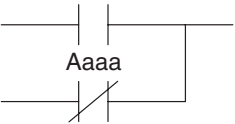
- ✓

230
- ✓

240
- ✓

250-1
- ✓

260
- 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.



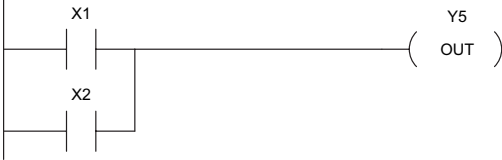
5

| Operand Data Type | | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|----|----------------|----------------|----------------|----------------|
| A | | aaa | aaa | aaa | aaa |
| Inputs | X | 0-177 | 0-477 | 0-777 | 0-1777 |
| Outputs | Y | 0-177 | 0-477 | 0-777 | 0-1777 |
| Control Relays | C | 0-377 | 0-377 | 0-1777 | 0-3777 |
| Stage | S | 0-377 | 0-777 | 0-1777 | 0-1777 |
| Timer | T | 0-77 | 0-177 | 0-377 | 0-377 |
| Counter | CT | 0-77 | 0-177 | 0-177 | 0-377 |
| Special Relay | SP | 0-117, 540-577 | 0-137, 540-617 | 0-137, 540-717 | 0-137, 540-717 |
| Global | GX | - | - | - | 0-3777 |
| Global | GY | - | - | - | 0-3777 |

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

| | |
|-----|---------|
| DS | Implied |
| HPP | Used |

DirectSOFT



Handheld Programmer Keystrokes

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

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

DirectSOFT

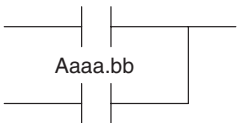


Handheld Programmer Keystrokes

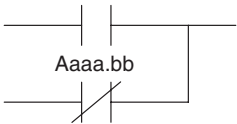
| | | | |
|-----|---|---|-----|
| \$ | → | B | ENT |
| STR | | 1 | |
| R | → | C | ENT |
| ORN | | 2 | |
| GX | → | F | ENT |
| OUT | | 5 | |

Or Bit-of-Word (ORB)

- ☒ 230 The Or Bit-of-Word instruction logically ors a normally open Bit-of-Word contact in parallel with another
- ☒ 240 contact in a rung. Status of the contact will be the same
- ☒ 250-1 state as the bit referenced in the associated memory
- ☒ 260 location.



- ☒ 230 **Or Not Bit-of-Word (ORNB)**
 - ☒ 240 The Or Not Bit-of-Word instruction logically ors a
 - ☒ 250-1 normally closed Bit-of-Word contact in parallel with
 - ☒ 260 another contact in a rung. Status of the contact will be
- opposite the state of the bit referenced in the associated memory location.



| Operand Data Type | | DL250-1 Range | | DL260 Range | |
|-------------------|----|--------------------------|--------------|--------------------------|--------------|
| | A | aaa | bb | aaa | bb |
| V-memory | B | See memory map page 3-55 | BCD, 0 to 15 | See memory map page 3-56 | BCD, 0 to 15 |
| Pointer | PB | See memory map page 3-55 | BCD, 0 to 15 | See memory map page 3-56 | BCD, 0 to 15 |

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

| | |
|-----|---------|
| DS | Implied |
| HPP | Used |

Handheld Programmer Keystrokes

| | | | | | | | | | |
|-----|------|---|-----|---|---|---|---|---|--|
| STR | → | 1 | ENT | | | | | | |
| OR | SHFT | B | → | V | 1 | 4 | 0 | 0 | |
| → | K | 7 | ENT | | | | | | |
| OUT | → | 7 | ENT | | | | | | |

In the following Or Not Bit-of-Word example, when input X1 is on or bit 7 of V1400 is off, output Y7 will energize.

DirectSOFT

Handheld Programmer Keystrokes

| | | | | | | | | | |
|-----|------|---|-----|---|---|---|---|---|--|
| STR | → | 1 | ENT | | | | | | |
| ORN | SHFT | B | → | V | 1 | 4 | 0 | 0 | |
| → | K | 7 | ENT | | | | | | |
| OUT | → | 7 | ENT | | | | | | |


And (AND)

- ✓ 230

✓ 240

✓ 250-1

✓ 260
- 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)

- ✓ 230

✓ 240

✓ 250-1

✓ 260
- 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.



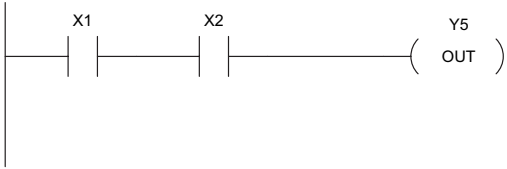
5

| Operand Data Type | | DL230 Range | DL240 Range | DL250-1 | DL260 Range |
|-------------------|----|----------------|----------------|----------------|----------------|
| A | | aaa | aaa | aaa | aaa |
| Inputs | X | 0-177 | 0-477 | 0-777 | 0-1777 |
| Outputs | Y | 0-177 | 0-477 | 0-777 | 0-1777 |
| Control Relays | C | 0-377 | 0-377 | 0-1777 | 0-3777 |
| Stage | S | 0-377 | 0-777 | 0-1777 | 0-1777 |
| Timer | T | 0-77 | 0-177 | 0-377 | 0-377 |
| Counter | CT | 0-77 | 0-177 | 0-177 | 0-377 |
| Special Relay | SP | 0-117, 540-577 | 0-137, 540-617 | 0-137, 540-717 | 0-137, 540-717 |
| Global | GX | - | - | - | 0-3777 |
| Global | GY | - | - | - | 0-3777 |

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

| | |
|-----|---------|
| DS | Implied |
| HPP | Used |

DirectSOFT

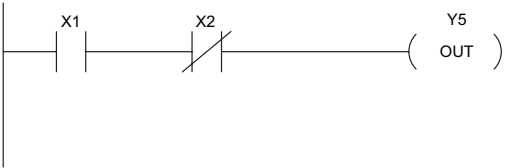


Handheld Programmer Keystrokes

| | | | |
|-----|---|---|-----|
| \$ | → | B | ENT |
| STR | | 1 | |
| V | → | C | ENT |
| AND | | 2 | |
| GX | → | F | ENT |
| OUT | | 5 | |

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

DirectSOFT



Handheld Programmer Keystrokes

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

AND Bit-of-Word (ANDB)

- 230 The And Bit-of-Word instruction logically ands a normally open contact in series with another
- 240 contact in a rung. The status of the contact will be the same state as the bit referenced in the
- 250-1 associated memory location.
- 260

And Not Bit-of-Word (ANDNB)

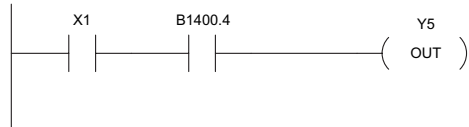
- 230 The And Not Bit-of-Word instruction logically ands a normally closed contact in series with
- 240 another contact in a rung. The status of the contact will be opposite the state of the bit
- 250-1 referenced in the associated memory location.
- 260

| Operand Data Type | | DL250-1 Range | | DL260 Range | |
|-------------------|----|--------------------------|--------------|--------------------------|--------------|
| | A | aaa | bb | aaa | bb |
| V-memory | B | See memory map page 3-55 | BCD, 0 to 15 | See memory map page 3-56 | BCD, 0 to 15 |
| Pointer | PB | See memory map page 3-55 | BCD | See memory map page 3-56 | BCD |

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

| | |
|-----|---------|
| DS | Implied |
| HPP | Used |

DirectSOFT

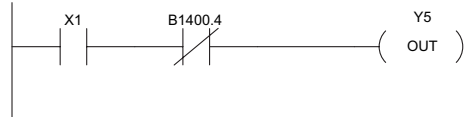


Handheld Programmer Keystrokes

| | | | | | | | | | |
|-----|------|---|-----|---|---|---|---|---|--|
| STR | → | 1 | ENT | | | | | | |
| AND | SHFT | B | → | V | 1 | 4 | 0 | 0 | |
| → | K | 4 | ENT | | | | | | |
| OUT | → | 5 | ENT | | | | | | |

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

DirectSOFT

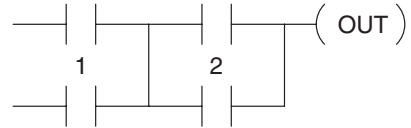


Handheld Programmer Keystrokes

| | | | | | | | | | |
|------|------|---|-----|---|---|---|---|---|--|
| STR | → | 1 | ENT | | | | | | |
| ANDN | SHFT | B | → | V | 1 | 4 | 0 | 0 | |
| → | K | 4 | ENT | | | | | | |
| OUT | → | 5 | ENT | | | | | | |

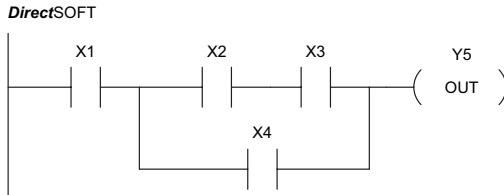
And Store (ANDSTR)

- ✓ 230 The And Store instruction logically ands two branches of a rung in series. Both branches must begin with the Store instruction.
- ✓ 240
- ✓ 250-1
- ✓ 260



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.

| | |
|-----|---------|
| DS | Implied |
| HPP | Used |

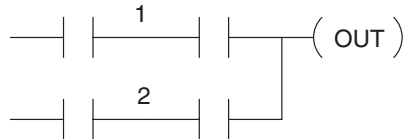


Handheld Programmer Keystrokes

| | | | |
|---------|-----|-----|-----|
| \$ 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 |

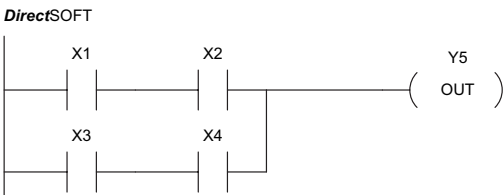
Or Store (ORSTR)

- ✓ 230 The Or Store instruction logically ors two branches of a rung in parallel. Both branches must begin with the Store instruction.
- ✓ 240
- ✓ 250-1
- ✓ 260



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

| | |
|-----|---------|
| DS | Implied |
| HPP | Used |



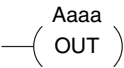
Handheld Programmer Keystrokes

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

Out (OUT)

- 230
- 240
- 250-1
- 260

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.

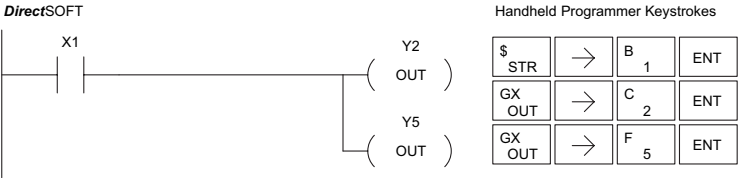


| Operand Data Type | | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|----|-------------|-------------|---------------|-------------|
| | A | aaa | aaa | aaa | aaa |
| Inputs | X | 0-177 | 0-477 | 0-777 | 0-1777 |
| Outputs | Y | 0-177 | 0-477 | 0-777 | 0-1777 |
| Control Relays | C | 0-377 | 0-377 | 0-1777 | 0-3777 |
| Global | GX | - | - | - | 0-3777 |
| Global | GY | - | - | - | 0-3777 |

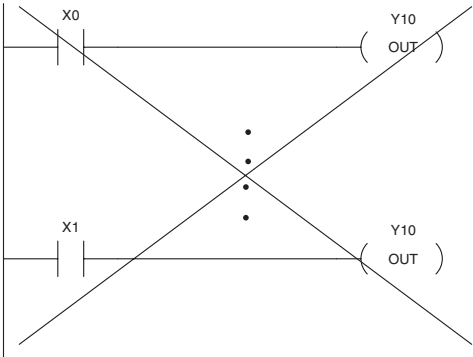
5

| | |
|-----|------|
| DS | Used |
| HPP | Used |

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



In the following Out example, the program contains two Out instructions using the same location (Y10). The physical output of Y10 is ultimately controlled by the last rung of logic referencing Y10. X1 will override the Y10 output being controlled by X0. To avoid this situation, multiple outputs using the same location should not be used in programming. If you need to have an output controlled by multiple inputs, see the OROUT instruction on page 5-19.



Out Bit-of-Word (OUTB)

- 230
- 240
- 250-1
- 260

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

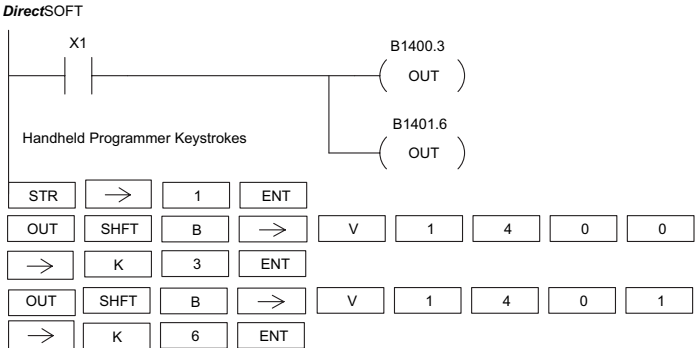
Aaaa.bb
—(OUT)

| Operand Data Type | | DL250-1 Range | | DL260 Range | |
|-------------------|----|--------------------------|--------------|--------------------------|--------------|
| A | | aaa | bb | aaa | bb |
| V-memory | B | See memory map page 3-55 | BCD, 0 to 15 | See memory map page 3-56 | BCD, 0 to 15 |
| Pointer | PB | See memory map page 3-55 | BCD | See memory map page 3-56 | BCD |

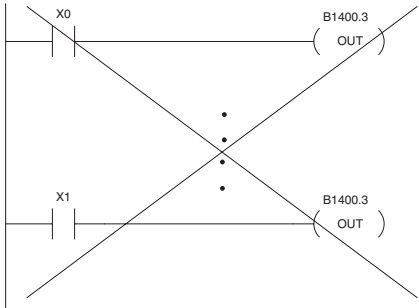
5

| | |
|-----|------|
| DS | Used |
| HPP | Used |

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

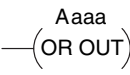


The following Out Bit-of-Word example contains two Out Bit-of-Word instructions using the same bit in the same memory word. The final state bit 3 of V1400 is ultimately controlled by the last rung of logic referencing it. X1 will override the logic state controlled by X0. To avoid this situation, multiple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



Or Out (OROUT)

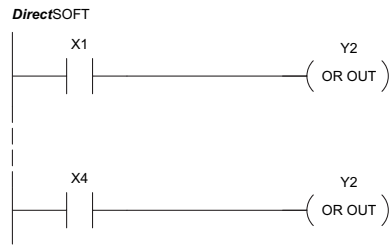
- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260
- The Or Out instruction allows more than one rung of discrete logic to control a single output. Multiple Or Out instructions referencing the same output coil may be used, since *all* contacts controlling the output are logically OR'd together. If the status of *any* rung is on, the output will also be on.



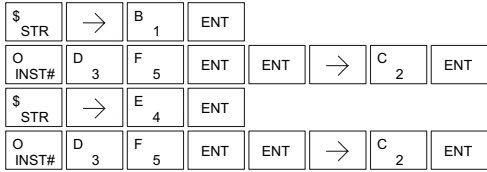
| Operand Data Type | | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|----|-------------|-------------|---------------|-------------|
| | A | aaa | aaa | aaa | aaa |
| Inputs | X | 0-177 | 0-477 | 0-777 | 0-1777 |
| Outputs | Y | 0-177 | 0-477 | 0-777 | 0-1777 |
| Control Relays | C | 0-377 | 0-377 | 0-1777 | 0-3777 |
| Global | GX | - | - | - | 0-3777 |
| Global | GY | - | - | - | 0-3777 |

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

| | |
|-----|------|
| DS | Used |
| HPP | Used |



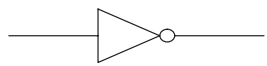
Handheld Programmer Keystrokes



- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260

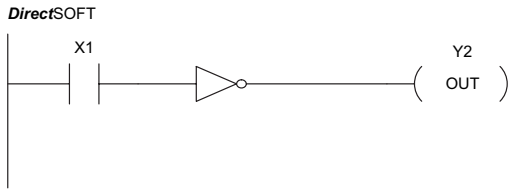
Not (NOT)

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

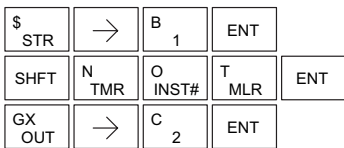


In the following example, when X1 is off, Y2 will energize. This is because the Not instruction inverts the status of the rung at the Not instruction.

| | |
|-----|------|
| DS | Used |
| HPP | Used |

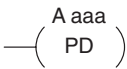


Handheld Programmer Keystrokes



Positive Differential (PD)

- 230 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.
- 240
- 250-1
- 260



5

| Operand Data Type | | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|---|-------------|-------------|---------------|-------------|
| | A | aaa | aaa | aaa | aaa |
| Inputs | X | 0-177 | 0-477 | 0-777 | 0-1777 |
| Outputs | Y | 0-177 | 0-477 | 0-777 | 0-1777 |
| Control Relays | C | 0-377 | 0-377 | 0-1777 | 0-3777 |

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

| | |
|-----|------|
| DS | Used |
| HPP | Used |

DirectSOFT



Handheld Programmer Keystrokes

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



NOTE: To generate a “one-shot” pulse on an on-to-off transition, place a NOT instruction immediately before the PD instruction. The DL250-1 and DL260 CPUs support the STRND instruction.

Store Positive Differential (STRPD)



230



240



250-1



260

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



230



240



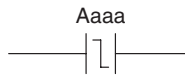
250-1



260

Store Negative Differential (STRND)

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



| Operand Data Type | | DL250-1 Range | DL260 Range |
|-------------------|----|---------------|-------------|
| A | | aaa | aaa |
| Inputs | X | 0-777 | 0-1777 |
| Outputs | Y | 0-777 | 0-1777 |
| Control Relays | C | 0-1777 | 0-3777 |
| Stage | S | 0-1777 | 0-1777 |
| Timer | T | 0-377 | 0-377 |
| Counter | CT | 0-177 | 0-377 |
| Global | GX | - | 0-3777 |
| Global | GY | - | 0-3777 |

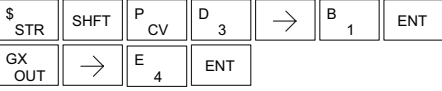
In the following example, each time X1 makes an off-to-on transition, Y4 will energize for one scan.

| | |
|-----|------|
| DS | Used |
| HPP | Used |

DirectSOFT



Handheld Programmer Keystrokes

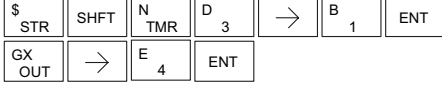


In the following example, each time X1 makes an on-to-off transition, Y4 will energize for one scan.

DirectSOFT



Handheld Programmer Keystrokes



Or Positive Differential (ORPD)

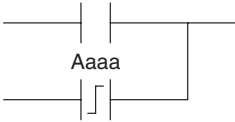
- ☒

230
- ☒

240
- ☒

250-1
- ☒

260
- The Or Positive Differential instruction logically ORs a contact in parallel with another contact in a rung. The status of the contact will be open until the associated image register point makes an off-to-on transition, closing it for one CPU scan. Thereafter, it remains open until another off-to-on transition.



Or Negative Differential (ORND)

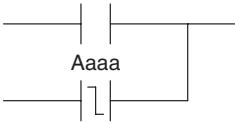
- ☒

230
- ☒

240
- ☒

250-1
- ☒

260
- The Or Negative Differential instruction logically ORs a contact in parallel with another contact in a rung. The status of the contact will be open until the associated image register point makes an on-to-off transition, closing it for one CPU scan. Thereafter, it remains open until another on-to-off transition.



| Operand Data Type | | DL250-1 Range | DL260 Range |
|-------------------|----|---------------|-------------|
| A | | aaa | aaa |
| Inputs | X | 0-777 | 0-1777 |
| Outputs | Y | 0-777 | 0-1777 |
| Control Relays | C | 0-1777 | 0-3777 |
| Stage | S | 0-1777 | 0-1777 |
| Timer | T | 0-377 | 0-377 |
| Counter | CT | 0-177 | 0-377 |
| Global | GX | - | 0-3777 |
| Global | GY | - | 0-3777 |

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

| | |
|-----|---------|
| DS | Implied |
| HPP | Used |

DirectSOFT

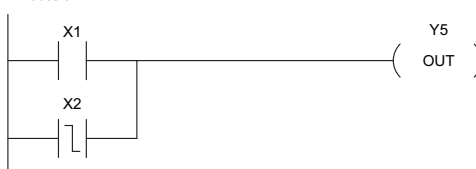


Handheld Programmer Keystrokes

| | | | |
|--------|------|------|-----|
| \$ STR | → | B 1 | ENT |
| Q OR | SHFT | P CV | D 3 |
| GX OUT | → | F 5 | ENT |

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

DirectSOFT



Handheld Programmer Keystrokes

| | | | |
|--------|------|-------|-----|
| \$ STR | → | B 1 | ENT |
| Q OR | SHFT | N TMR | D 3 |
| GX OUT | → | F 5 | ENT |

And Positive Differential (ANDPD)

☒ 230

☒ 240

☒ 250-1

☒ 260

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

And Negative Differential (ANDND)

☒ 230

☒ 240

☒ 250-1

☒ 260

The And Negative Differential instruction logically ANDs a normally open contact in series with another contact 5-23 in a rung. The status of the contact will be open until the associated image register point makes an on-to-off transition, closing it for one CPU scan. Thereafter, it remains open until another on-to-off transition.

5

| Operand Data Type | | DL250-1 Range | DL260 Range |
|-------------------|----|---------------|-------------|
| | A | aaa | aaa |
| Inputs | X | 0-777 | 0-1777 |
| Outputs | Y | 0-777 | 0-1777 |
| Control Relays | C | 0-1777 | 0-3777 |
| Stage | S | 0-1777 | 0-1777 |
| Timer | T | 0-377 | 0-377 |
| Counter | CT | 0-177 | 0-377 |
| Global | GX | — | 0-3777 |
| Global | GY | — | 0-3777 |

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

| | |
|-----|---------|
| DS | Implied |
| HPP | Used |

DirectSOFT

Handheld Programmer Keystrokes

| | | | |
|-----------|------|---------|--------|
| \$ STR | → | B 1 | ENT |
| V AND | SHFT | P CV | D 3 |
| GX OUT | → | F 5 | ENT |

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

DirectSOFT

Handheld Programmer Keystrokes

| | | | |
|-----------|------|----------|--------|
| \$ STR | → | B 1 | ENT |
| V AND | SHFT | N TMR | D 3 |
| GX OUT | → | F 5 | ENT |

Set (SET)

- ✓ 230 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.
- ✓ 240
- ✓ 250-1
- ✓ 260

Optional
memory range

A aaa aaa
— (SET)

Reset (RST)

- ✓ 230 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.
- ✓ 240
- ✓ 250-1
- ✓ 260

Optional
Memory range

A aaa aaa
— (RST)

5

| Operand Data Type | | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|----|-------------|-------------|---------------|-------------|
| | A | aaa | aaa | aaa | aaa |
| Inputs | X | 0-177 | 0-477 | 0-777 | 0-1777 |
| Outputs | Y | 0-177 | 0-477 | 0-777 | 0-1777 |
| Control Relays | C | 0-377 | 0-377 | 0-1777 | 0-3777 |
| Stage | S | 0-377 | 0-777 | 0-1777 | 0-1777 |
| Timer* | T | 0-77 | 0-177 | 0-377 | 0-377 |
| Counter * | CT | 0-77 | 0-177 | 0-177 | 0-377 |
| Global | GX | - | - | - | 0-3777 |
| Global | GY | - | - | - | 0-3777 |

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



NOTE: You cannot set inputs (Xs) that are assigned to input modules

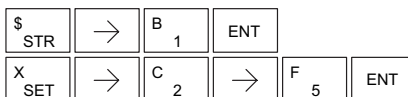
In the following example, when X1 is on, Y2 through Y5 will energize.

| | |
|-----|------|
| DS | Used |
| HPP | Used |

DirectSOFT



Handheld Programmer Keystrokes

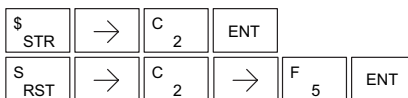


In the following example, when X2 is on, Y2 through Y5 will be reset or de-energized.

DirectSOFT



Handheld Programmer Keystrokes



Set Bit-of-Word (SETB)

- ✗

230

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

240
- ✓

250-1
- ✓

260

Aaaa.bb
(SET)

Reset Bit-of-Word (RSTB)

- ✗

230

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

A aaa.bb
(RST)

✗ 240

✓ 250-1

✓ 260

| Operand Data Type | | DL250-1 Range | | DL260 Range | |
|-------------------|----|----------------|--------------|----------------|--------------|
| | A | aaa | bb | aaa | bb |
| V-memory | B | See memory map | BCD, 0 to 15 | See memory map | BCD, 0 to 15 |
| Pointer | PB | See memory map | BCD | See memory map | BCD |

| | |
|-----|------|
| DS | Used |
| HPP | Used |

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

DirectSOFT



Handheld Programmer Keystrokes

| | | | | | | | | |
|-----|------|---|-----|---|---|---|---|---|
| STR | → | 1 | ENT | | | | | |
| SET | SHFT | B | → | V | 1 | 4 | 0 | 0 |
| → | K | 1 | ENT | | | | | |

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

DirectSOFT

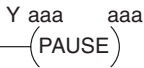


Handheld Programmer Keystrokes

| | | | | | | | | |
|-----|------|---|-----|---|---|---|---|---|
| STR | → | 2 | ENT | | | | | |
| RST | SHFT | B | → | V | 1 | 4 | 0 | 0 |
| → | K | 1 | ENT | | | | | |

Pause (PAUSE)

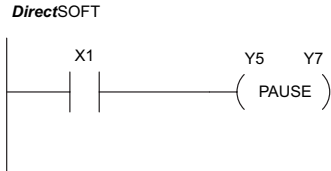
- 230 The Pause instruction disables the output update on a range of outputs. The ladder program will continue to run and update the image register; however, the outputs in the range specified in the Pause instruction will be turned off at the output points.
- 240
- 250-1
- 260



| Operand Data Type | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|-------------|-------------|---------------|-------------|
| | aaa | aaa | aaa | aaa |
| Outputs | Y | 0-177 | 0-777 | 0-1777 |

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

| | |
|-----|------|
| DS | Used |
| HPP | Used |



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

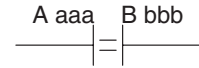
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-----------|---|--------|-----|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|----|
| \$ STR | → | B 1 | ENT | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | </ |
|-----------|---|--------|-----|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|----|

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

Comparative Boolean

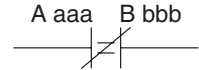
Store If Equal (STRE)

- ✓ 230 The Store If Equal instruction begins a new rung or
- ✓ 240 additional branch in a rung with a normally open
- ✓ 250-1 comparative contact. The contact will be on when Vaaa
- ✓ 260 equals Bbbb .



Store If Not Equal (STRNE)

- ✓ 230 The Store If Not Equal instruction begins a new rung or
- ✓ 240 additional branch in a rung with a normally closed
- ✓ 250-1 comparative contact. The contact will be on when Vaaa
- ✓ 260 does not equal Bbbb.



| Operand Data Type | | DL230 Range | | DL240 Range | | DL250-1 Range | | DL260 Range | |
|-------------------|---|---------------------------------|---------------------------------|---------------------------------|---------------------------------|---------------------------------|---------------------------------|---------------------------------|---------------------------------|
| A/B | | aaa | bbb | aaa | bbb | aaa | bbb | aaa | bbb |
| V-memory | V | All. (See memory map page 3-53) | All. (See memory map page 3-53) | All. (See memory map page 3-54) | All. (See memory map page 3-54) | All. (See memory map page 3-54) | All. (See memory map page 3-55) | All. (See memory map page 3-56) | All. (See memory map page 3-56) |
| Pointer | P | — | — | — | All. (See memory map page 3-54) | — | All. (See memory map page 3-55) | — | All. (See memory map page 3-56) |
| Constant | K | — | 0-FFFF | — | 0-FFFF | — | 0-FFFF | — | 0-FFFF |

| | |
|-----|---------|
| DS | Implied |
| HPP | Used |

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)

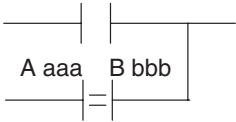
- ✓

230
- ✓

240
- ✓

250-1
- ✓

260
- The Or If Equal instruction connects a normally open comparative contact in parallel with another contact. The contact will be on when Vaaa equals Bbbb.



Or If Not Equal (ORNE)

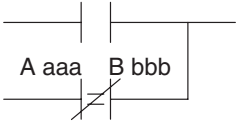
- ✓

230
- ✓

240
- ✓

250-1
- ✓

260
- The Or If Not Equal instruction connects a normally closed comparative contact in parallel with another contact. The contact will be on when Vaaa does not equal Bbbb.



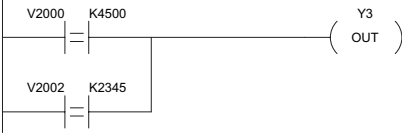
5

| Operand Data Type | | DL230 Range | | DL240 Range | | DL250-1 Range | | DL260 Range | |
|-------------------|---|---------------------------------|---------------------------------|---------------------------------|---------------------------------|---------------------------------|---------------------------------|---------------------------------|---------------------------------|
| A/B | | aaa | bbb | aaa | bbb | aaa | bbb | aaa | bbb |
| V-memory | V | All. (See memory map page 3-53) | All. (See memory map page 3-53) | All. (See memory map page 3-54) | All. (See memory map page 3-54) | All. (See memory map page 3-55) | All. (See memory map page 3-55) | All. (See memory map page 3-56) | All. (See memory map page 3-56) |
| Pointer | P | — | — | — | All. (See memory map page 3-54) | — | All. (See memory map page 3-55) | — | All. (See memory map page 3-56) |
| Constant | K | — | 0-FFFF | — | 0-FFFF | — | 0-FFFF | — | 0-FFFF |

In the following example, when the value in V-memory location V2000 = 4500 or V2202 = 2345, Y3 will energize.

| | |
|-----|---------|
| DS | Implied |
| HPP | Used |

DirectSOFT

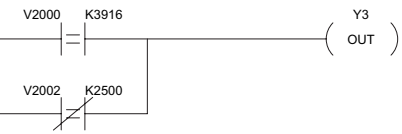


Handheld Programmer Keystrokes

| | | | | | | | | |
|--------|------|-----|-----|-----|-----|-----|-----|---|
| \$ 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

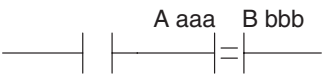


Handheld Programmer Keystrokes

| | | | | | | | | |
|--------|------|-----|-----|-----|-----|-----|-----|---|
| \$ 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 | | | | | |

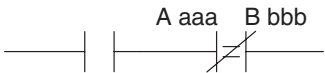
And If Equal (ANDE)

- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260
- The And If Equal instruction connects a normally open comparative contact in series with another contact. The contact will be on when Vaaa equals Bbbb.



And If Not Equal (ANDNE)

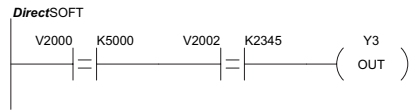
- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260
- The And If Not Equal instruction connects a normally closed comparative contact in series with another contact. The contact will be on when Vaaa does not equal Bbbb
- In the following example, when the value in V-memory location V2000 = 5000 and V2002 = 2345, Y3 will energize.



| Operand Data Type | | DL230 Range | | DL240 Range | | DL250-1 Range | | DL260 Range | |
|-------------------|---|---------------------------------|---------------------------------|---------------------------------|--|---------------------------------|--|---------------------------------|--|
| A/B | | aaa | bbb | aaa | bbb | aaa | bbb | aaa | bbb |
| V-memory | V | All. (See memory map page 3-53) | All. (See memory map page 3-53) | All. (See memory map page 3-54) | All. (See memory map page 3-54) | All. (See memory map page 3-55) | All. (See memory map page 3-55) | All. (See memory map page 3-56) | All. (See memory map page 3-56) |
| Pointer | P | — | — | — | All V-memory. (See memory map page 3-54) | — | All V-memory. (See memory map page 3-55) | — | All V-memory. (See memory map page 3-56) |
| Constant | K | — | 0-FFFF | — | 0-FFFF | — | 0-FFFF | — | 0-FFFF |

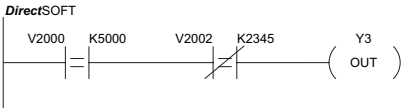
| | |
|-----|---------|
| DS | Implied |
| HPP | Used |

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



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 | | | | | |



Handheld Programmer Keystrokes

| | | | | | | | | |
|--------|------|-----|-----|-----|-----|-----|-----|---|
| \$ STR | SHFT | E 4 | → | C 2 | A 0 | A 0 | A 0 | → |
| F 5 | A 0 | A 0 | A 0 | ENT | | | | |
| W ANDN | SHFT | E 4 | → | C 2 | A 0 | A 0 | C 2 | → |
| C 2 | D 3 | E 4 | F 5 | ENT | | | | |
| GX OUT | → | D 3 | ENT | | | | | |

Store (STR)

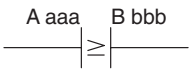
- ✓

230
- ✓

240
- ✓

250-1
- ✓

260
- The Comparative Store instruction begins a new rung or additional branch in a rung with a normally open comparative contact. The contact will be on when Aaaa is equal to or greater than Bbbb.



Store Not (STRN)

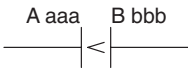
- ✓

230
- ✓

240
- ✓

250-1
- ✓

260
- The Comparative Store Not instruction begins a new rung or additional branch in a rung with a normally open comparative contact. The contact will be on when Aaaa is less than Bbbb.



5

| Operand Data Type | | DL230 Range | | DL240 Range | | DL250-1 Range | | DL260 Range | |
|-------------------|---|---------------------------------|---------------------------------|---------------------------------|--|---------------------------------|--|---------------------------------|--|
| A/B | | aaa | bbb | aaa | bbb | aaa | bbb | aaa | bbb |
| V-memory | V | All. (See memory map page 3-53) | All. (See memory map page 3-53) | All. (See memory map page 3-54) | All. (See memory map page 3-54) | All. (See memory map page 3-55) | All. (See memory map page 3-55) | All. (See memory map page 3-56) | All. (See memory map page 3-56) |
| Pointer | P | — | — | — | All V-memory. (See memory map page 3-54) | — | All V-memory. (See memory map page 3-55) | — | All V-memory. (See memory map page 3-56) |
| Constant | K | — | 0-FFFF | — | 0-FFFF | — | 0-FFFF | — | 0-FFFF |

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

| | |
|-----|---------|
| DS | Implied |
| HPP | Used |

DirectSOFT



Handheld Programmer Keystrokes

| | | | | | | | | | | | | | |
|----|-----|---|------|---|-----|---|---|---|-----|---|---|---|---|
| \$ | STR | → | SHFT | V | AND | C | 2 | A | 0 | A | 0 | A | 0 |
| → | B | 1 | A | 0 | A | 0 | A | 0 | ENT | | | | |
| GX | OUT | → | D | 3 | ENT | | | | | | | | |

In the following example, when the value in V-memory location V2000 $<$ 4050, Y3 will energize.

DirectSOFT

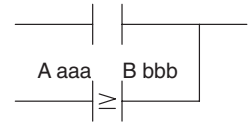


Handheld Programmer Keystrokes

| | | | | | | | | | | | | | |
|----|------|---|------|---|-----|---|---|---|-----|---|---|---|---|
| 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 | | | | | | | | |

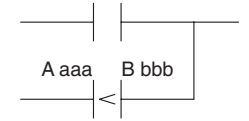
Or (OR)

- ✓ 230 The Comparative Or instruction connects a normally open comparative contact in parallel with another contact. The
- ✓ 240 contact will be on when Aaaa is equal to or greater than
- ✓ 250-1 Bbbb.
- ✓ 260



Or Not (ORN)

- ✓ 230 The Comparative Or Not instruction connects a normally open comparative contact in parallel with another contact.
- ✓ 240 The contact will be on when Aaaa is less than Bbbb.
- ✓ 250-1
- ✓ 260

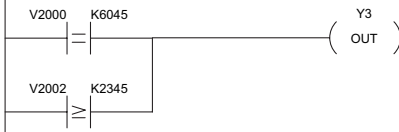


| Operand Data Type | | DL230 Range | | DL240 Range | | DL250-1 Range | | DL260 Range | |
|-------------------|---|---------------------------------|---------------------------------|---------------------------------|--|---------------------------------|--|---------------------------------|--|
| A/B | | aaa | bbb | aaa | bbb | aaa | bbb | aaa | bbb |
| V-memory | V | All. (See memory map page 3-53) | All. (See memory map page 3-53) | All. (See memory map page 3-54) | All. (See memory map page 3-54) | All. (See memory map page 3-55) | All. (See memory map page 3-55) | All. (See memory map page 3-56) | All. (See memory map page 3-56) |
| Pointer | P | — | — | — | All V-memory. (See memory map page 3-54) | — | All V-memory. (See memory map page 3-55) | — | All V-memory. (See memory map page 3-56) |
| Constant | K | — | 0-FFFF | — | 0-FFFF | — | 0-FFFF | — | 0-FFFF |

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

| | |
|-----|---------|
| DS | Implied |
| HPP | Used |

DirectSOFT

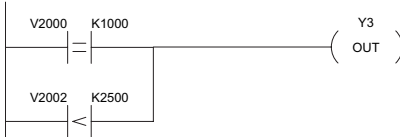


Handheld Programmer Keystrokes

| | | | | | | | | | |
|-----|-----|------|------|-------|-----|-----|-----|-----|---|
| \$ | 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.

DirectSOFT

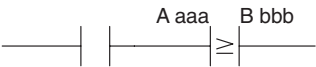


Handheld Programmer Keystrokes

| | | | | | | | | | |
|-----|-----|------|------|-------|-----|-----|-----|-----|---|
| \$ | 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 | | | | | |

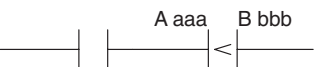
And (AND)

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



And Not (ANDN)

- 230 The Comparative And Not instruction connects a normally open comparative contact in series with another contact.
- 240 The contact will be on when Aaaa < Bbbb.

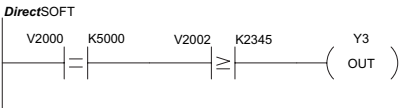


5

| Operand Data Type | | DL230 Range | | DL240 Range | | DL250-1 Range | | DL260 Range | |
|-------------------|---|--------------------------------|--------------------------------|--------------------------------|---|--------------------------------|---|--------------------------------|---|
| A/B | | aaa | bbb | aaa | bbb | aaa | bbb | aaa | bbb |
| V-memory | V | All (See memory map page 3-53) | All (See memory map page 3-53) | All (See memory map page 3-54) | All (See memory map page 3-54) | All (See memory map page 3-55) | All (See memory map page 3-55) | All (See memory map page 3-56) | All (See memory map page 3-56) |
| Pointer | P | - | - | - | All V-memory (See memory map page 3-54) | - | All V-memory (See memory map page 3-55) | - | All V-memory (See memory map page 3-56) |
| Constant | K | - | 0-FFFF | - | 0-FFFF | - | 0-FFFF | - | 0-FFFF |

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

| | |
|-----|---------|
| DS | Implied |
| HPP | Used |



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.



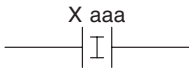
Handheld Programmer Keystrokes

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

Immediate Instructions

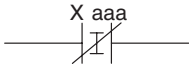
Store Immediate (STRI)

- ✓ 230 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.
- ✓ 240
- ✓ 250-1
- ✓ 260



Store Not Immediate (STRNI)

- ✓ 230 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.
- ✓ 240
- ✓ 250-1
- ✓ 260

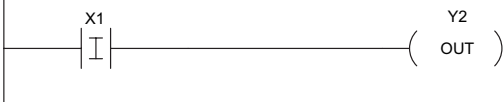


| Operand Data Type | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|-------------|-------------|---------------|-------------|
| | aaa | aaa | aaa | aaa |
| Inputs | X 0-177 | 0-477 | 0-777 | 0-1777 |

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

| | |
|-----|---------|
| DS | Implied |
| HPP | Used |

DirectSOFT

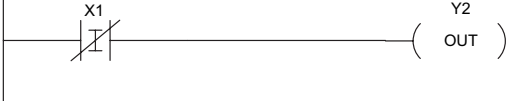


Handheld Programmer Keystrokes

| | | | | | |
|--------|------|-----|-----|-----|-----|
| \$ STR | SHFT | I 8 | → | B 1 | ENT |
| GX OUT | → | C 2 | ENT | | |

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

DirectSOFT

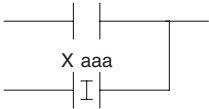


Handheld Programmer Keystrokes

| | | | | | |
|---------|------|-----|-----|-----|-----|
| SP STRN | SHFT | I 8 | → | B 1 | ENT |
| GX OUT | → | C 2 | ENT | | |

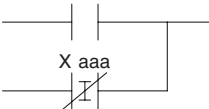
Or Immediate (ORI)

- 230 The Or Immediate connects two contacts in parallel. The
- 240 status of the contact will be the same as the status of the
- 250-1 associated input point *at the time the instruction is executed.*
- 260 The image register is not updated.



Or Not Immediate (ORNI)

- 230 The Or Not Immediate connects two contacts in parallel.
- 240 The status of the contact will be opposite the status of the
- 250-1 associated input point *at the time the instruction is executed.*
- 260 The image register is not updated.



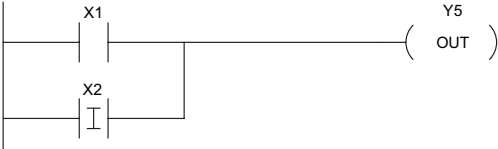
5

| Operand Data Type | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|-------------|-------------|---------------|-------------|
| | aaa | aaa | aaa | aaa |
| Inputs | X 0-177 | 0-477 | 0-777 | 0-1777 |

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

| | |
|-----|---------|
| DS | Implied |
| HPP | Used |

DirectSOFT

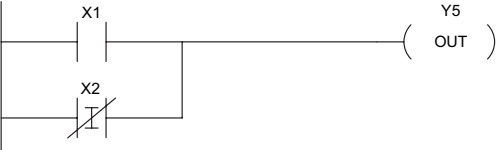


Handheld Programmer Keystrokes

| | | | | | |
|--------|------|-----|-----|-----|-----|
| \$ STR | → | B 1 | ENT | | |
| Q OR | SHFT | I 8 | → | C 2 | ENT |
| GX OUT | → | F 5 | ENT | | |

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

DirectSOFT



Handheld Programmer Keystrokes

| | | | | | |
|--------|------|-----|-----|-----|-----|
| \$ STR | → | B 1 | ENT | | |
| R ORN | SHFT | I 8 | → | C 2 | ENT |
| GX OUT | → | F 5 | ENT | | |

And Immediate (ANDI)

- 230 The And Immediate connects two contacts in series. The
- 240 status of the contact will be the same as the status of the
- 250-1 associated input point *at the time the instruction is executed*.
- 260 The image register is not updated.



And Not Immediate (ANDNI)

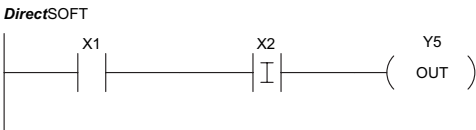
- 230 The And Not Immediate connects two contacts in series.
- 240 The status of the contact will be opposite the status of the
- 250-1 associated input point *at the time the instruction is executed*.
- 260 The image register is not updated.



| Operand Data Type | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|-------------|-------------|---------------|-------------|
| | aaa | aaa | aaa | aaa |
| Inputs | X | 0-177 | 0-777 | 0-1777 |

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

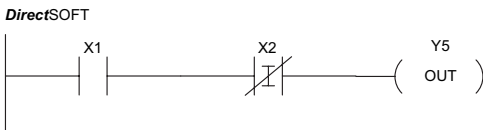
| | |
|-----|---------|
| DS | Implied |
| HPP | Used |



Handheld Programmer Keystrokes

| | | | | | |
|-----------|------|--------|-----|--------|-----|
| \$ STR | → | B 1 | ENT | | |
| V AND | SHFT | I 8 | → | C 2 | ENT |
| GX OUT | → | F 5 | ENT | | |

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



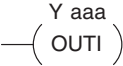
Handheld Programmer Keystrokes

| | | | | | |
|--------|------|----------------|-----|----------------|-----|
| \$ STR | → | B ₁ | ENT | | |
| W ANDN | SHFT | I ₈ | → | C ₂ | ENT |
| GX OUT | → | F ₅ | ENT | | |

Out Immediate (OUTI)

- 230
- 240
- 250-1
- 260

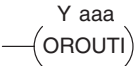
The Out Immediate instruction reflects the status of the rung (on/off) and outputs the discrete (on/off) status to the specified module output point and the image register *at the time the instruction is executed*. If multiple Out Immediate instructions referencing the same discrete point are used, it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



Or Out Immediate (OROUTI)

- 230
- 240
- 250-1
- 260

The Or Out Immediate instruction has been designed to use more than one 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 | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|-------------|-------------|---------------|-------------|
| | aaa | aaa | aaa | aaa |
| Outputs | Y 0-177 | 0-477 | 0-777 | 0-1777 |

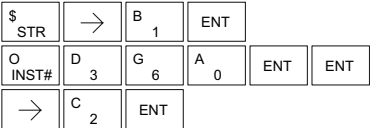
In the following example, when X1 is on, output point Y2 on the output module will turn on. For instruction entry on the Handheld Programmer, you can use the instruction number (#350) as shown, or type each letter of the command.

| | |
|-----|------|
| DS | Used |
| HPP | Used |

DirectSOFT

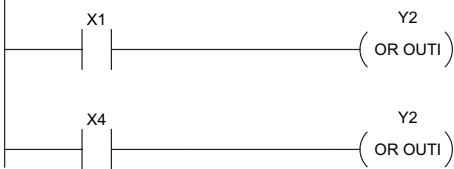


Handheld Programmer Keystrokes

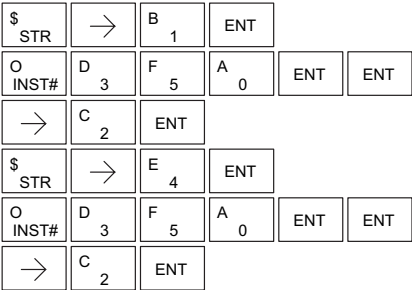


In the following example, when X1 or X4 is on, Y2 will energize.

DirectSOFT



Handheld Programmer Keystrokes



Out Immediate Formatted (OUTIF)

- ☐ 230
- ☐ 240
- ☐ 250-1
- ☒ 260

The Out Immediate Formatted instruction outputs a 1 to 32 bit binary value from the accumulator to specified output points *at the time the instruction is executed*. Accumulator bits that are not used by the instruction are set to zero.



| Operand Data Type | | DL260 Range | |
|-------------------|---|-------------|------|
| | | aaa | bbb |
| Outputs | Y | 0-1777 | — |
| Constant | K | — | 1-32 |

In the following example, when C0 is on, the binary pattern for X10–X17 is loaded into the accumulator using the Load Immediate Formatted instruction. The binary pattern in the accumulator is written to Y30–Y37 using the Out Immediate Formatted instruction. This technique is useful to quickly copy an input pattern to outputs (without waiting for the CPU scan).

5

| | |
|-----|------|
| DS | Used |
| HPP | Used |

DirectSOFT

CO



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



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

Unused accumulator bits are set to zero

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

Acc.

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



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

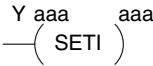
Handheld Programmer Keystrokes

| | | | | | | | | | | | |
|-----------|------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|-----|--|
| \$ STR | → | NEXT | NEXT | NEXT | NEXT | A ₀ | ENT | | | | |
| SHFT | L ANDST | D ₃ | I ₈ | F ₅ | → | B ₁ | A ₀ | → | I ₈ | ENT | |
| GX OUT | SHFT | I ₈ | F ₅ | → | D ₃ | A ₀ | → | I ₈ | ENT | | |

Set Immediate (SETI)

- 230
- 240
- 250-1
- 260

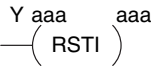
The Set Immediate instruction immediately sets or turns on an output or a range of outputs in the image register and the corresponding output point(s) *at the time the instruction is executed*. Once the outputs are set, it is not necessary for the input to remain on. The Reset Immediate instruction can be used to reset the outputs.



Reset Immediate (RSTI)

- 230
- 240
- 250-1
- 260

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 | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|-------------|-------------|---------------|-------------|
| | aaa | aaa | aaa | aaa |
| Outputs | Y 0-177 | 0-477 | 0-777 | 0-1777 |

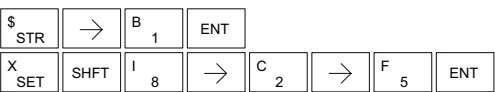
In the following example, when X1 is on, Y2 through Y5 will be set on in the image register and on the corresponding output points.

| | |
|-----|------|
| DS | Used |
| HPP | Used |

DirectSOFT



Handheld Programmer Keystrokes



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

DirectSOFT



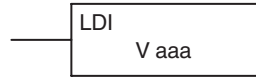
Handheld Programmer Keystrokes



Load Immediate (LDI)

- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260

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

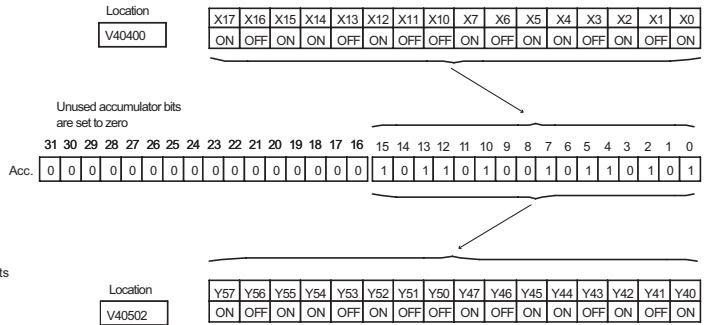
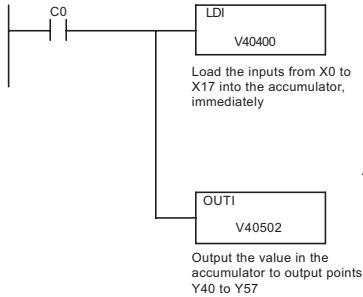


| Operand Data Type | DL260 Range |
|-------------------|---------------|
| | aaaaa |
| Inputs V-memory | V 40400–40477 |

In the following example, when C0 is on, the binary pattern of X0–X17 will be loaded into the accumulator using the Load Immediate instruction. The Out Immediate instruction could be used to copy the 16 bits in the accumulator to output points, such as Y40–Y57. This technique is useful to quickly copy an input pattern to output points (without waiting for a full CPU scan to occur).

| | |
|-----|------|
| DS | Used |
| HPP | Used |

DirectSOFT



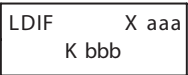
Handheld Programmer Keystrokes

| | | | | | | | | | | | | | | | |
|-----------|------------|--------|--------|--------|--------|--------|--------|--------|--------|-----|--|--|--|--|--|
| \$ STR | → | SHFT | C 2 | A 0 | ENT | | | | | | | | | | |
| SHFT | L ANDST | D 3 | I 8 | → | E 4 | A 0 | E 4 | A 0 | A 0 | ENT | | | | | |
| GX OUT | SHFT | I 8 | → | NEXT | E 4 | A 0 | F 5 | A 0 | C 2 | ENT | | | | | |

Load Immediate Formatted (LDIF)

- 230
- 240
- 250-1
- 260

The Load Immediate Formatted instruction loads a 1–32 bit binary value into the accumulator. The value reflects the current status of the input module(s) *at the time the instruction is executed*. Accumulator bits that are not used by the instruction are set to zero.

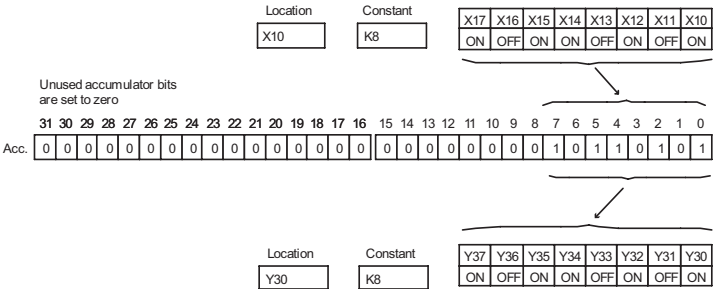
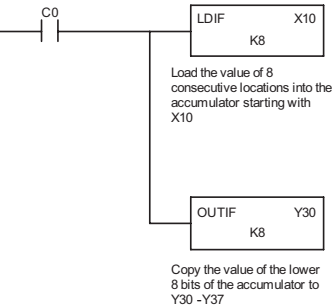


| Operand Data Type | | DL260 Range | |
|-------------------|---|-------------|------|
| | | aaa | bbb |
| Inputs | X | 0–1777 | – |
| Constant | K | – | 1–32 |

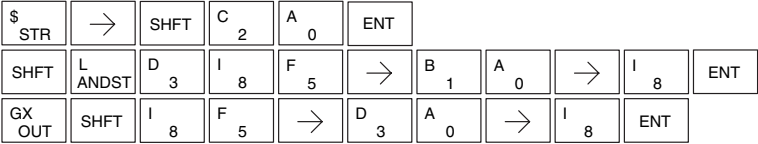
In the following example, when C0 is on, the binary pattern of X10–X17 will be loaded into the accumulator using the Load Immediate Formatted instruction. The Out Immediate Formatted instruction could be used to copy the specified number of bits in the accumulator to the specified outputs on the output module, such as Y30–Y37. This technique is useful to quickly copy an input pattern to outputs (without waiting for the CPU scan).

| | |
|-----|------|
| DS | Used |
| HPP | Used |

DirectSOF



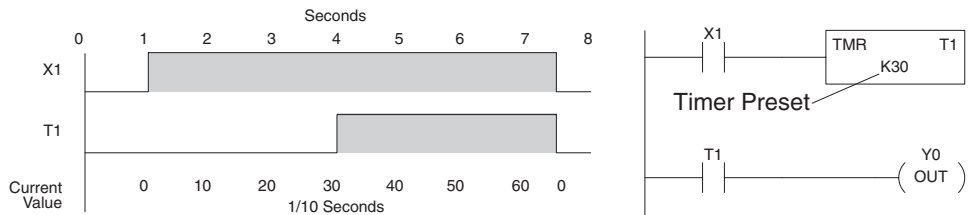
Handheld Programmer Keystrokes



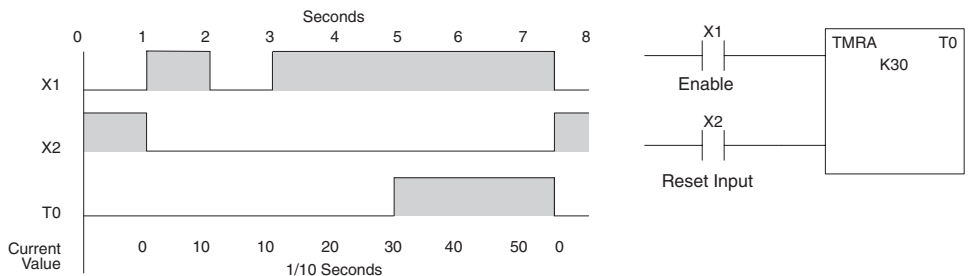
Timer, Counter and Shift Register Instructions

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. A discrete bit is 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.

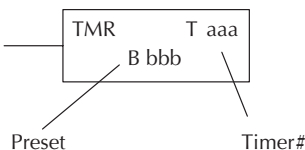


Some applications that need an accumulating timer, meaning it has the ability to time, stop, and then resume from where it previously stopped. The accumulating timer works similarly to the regular timer, but two inputs are required. The enable input starts and stops the timer. When the timer stops, the elapsed time is maintained. When the timer starts again, the timing continues from the elapsed time. When the reset input is turned on, the elapsed time is cleared and the timer will start at 0 when it is restarted. A tenth of a second and a hundredth of a second timers are available with a maximum time of 999999.9 and 99999.99 seconds respectively. The timing diagram below shows the relationship between the timer input, timer reset, associated discrete bit, current value, and timer preset.



Timer (TMR) and Timer Fast (TMRF)

- ✓ **230** The Timer instruction is a 0.1 second single-input timer that times to a maximum of 999.9 seconds. The Timer Fast
- ✓ **240** instruction is a 0.01 second single input timer that times up to a maximum of 99.99 seconds. These timers will be enabled
- ✓ **250-1** if the input logic is true (on) and will be reset to 0 if the input
- ✓ **260** logic is false (off).



Instruction Specifications

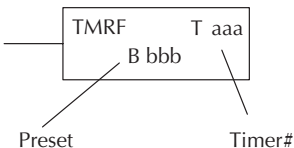
| | |
|-----|------|
| DS | Used |
| HPP | Used |

Timer Reference (Taaa): Specifies the timer number.

Preset Value (Bbbb): Constant value (K) or a V-memory location. (Pointer (P) for DL240, DL250-1 and DL260).

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. It will be on if the current value is equal to or greater than the preset value. For example, the discrete status bit for Timer 2 would be T2.



NOTE: A V-memory preset is required only if the ladder program or an Operator Interface unit must change the preset.

| Operand Data Type | | DL230 Range | | DL240 Range | | DL250-1 Range | | DL260 Range | |
|----------------------------|------|----------------------|-----------|-----------------------|-----------|-----------------------|--------------------------|-----------------------|--------------------------|
| | B | aaa | bbb | aaa | bbb | aaa | bbb | aaa | bbb |
| Timers | T | 0-77 | | 0-177 | | 0-377 | | 0-377 | |
| V-memory for preset values | V | — | 2000-2377 | — | 2000-3777 | — | 1400-7377 10000-17777 | — | 1400-7377 10000-37777 |
| Pointers (presets only) | P | | | | 2000-3777 | | 1400-7377 10000-17777 | | 1400-7377 10000-37777 |
| Constants (presets only) | K | — | 0-9999 | — | 0-9999 | — | 0-9999 | — | 0-9999 |
| Timer discrete status bits | T/V* | 0-77 or V41100-41103 | | 0-177 or V41100-41107 | | 0-377 or V41100-41117 | | 0-377 or V41100-41117 | |
| Timer current values | V/T* | 0-77 | | 0-177 | | 0-377 | | 0-377 | |

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

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

Handheld Programmer Keystrokes

| | | | | | |
|----|-----|---|------|-----|-----|
| \$ | STR | → | B | 1 | ENT |
| N | TMR | → | C | 2 | → |
| | | | 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 |

Timing Diagram

DL205 User Manual, 4th Edition, Rev. C

5-43

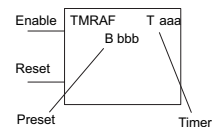
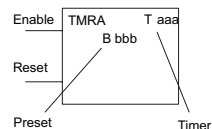
Accumulating Timer (TMRA)

- ✓ 230 The Accumulating Timer is a 0.1 second two-input timer that will time to a maximum of 9999999.9. *The TMRA uses two timer registers in V-memory.*
- ✓ 240
- ✓ 250-1

Accumulating Fast Timer (TMRAF)

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

- ✓ 230 These timers have two inputs: an enable and a reset. The timer will start timing when the enable is on and stop timing when the enable is off without resetting the value to 0. The reset will reset the timer when on and allow the timer to time when off.
- ✓ 240
- ✓ 250-1
- ✓ 260



Instruction Specifications

| | |
|-----|------|
| DS | Used |
| HPP | Used |

Timer Reference (Taaa): Specifies the timer number.

Preset Value (Bbbb): Constant value (K) or two consecutive V-memory locations. (Pointer (P) for DL240, DL250-1 and DL260).

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. It will be on if the current value is equal to or greater than the preset value. For example, the discrete status bit for Timer 2 would be T2.



NOTE: The accumulating timer uses two consecutive V-memory locations for the 8-digit value; therefore, two consecutive timer locations. For example, if TMRA T1 is used, the next available timer number is T3.



NOTE: A V-memory preset is required only if the ladder program or an OIT must be used to change the preset.

| Operand Data Type | | DL230 Range | | DL240 Range | | DL250-1 Range | | DL260 Range | |
|----------------------------|------|----------------------|------------|-----------------------|------------|-----------------------|--------------------------|-----------------------|--------------------------|
| B | | aaa | bbb | aaa | bbb | aaa | bbb | aaa | bbb |
| Timers | T | 0-76 | | 0-176 | | 0-376 | | 0-376 | |
| V-memory for preset values | V | — | 2000-2377 | — | 2000-3777 | — | 1400-7377 10000-17777 | — | 1400-7377 10000-37777 |
| Pointers (presets only) | P | | | | 2000-3777 | | 1400-7377 10000-17777 | | 1400-7377 10000-37777 |
| Constants (presets only) | K | — | 0-99999999 | — | 0-99999999 | — | 0-99999999 | — | 0-99999999 |
| Timer discrete status bits | T/V* | 0-76 or V41100-41103 | | 0-176 or V41100-41107 | | 0-376 or V41100-41117 | | 0-376 or V41100-41117 | |
| Timer current values | V/T* | 0-76 | | 0-176 | | 0-376 | | 0-376 | |

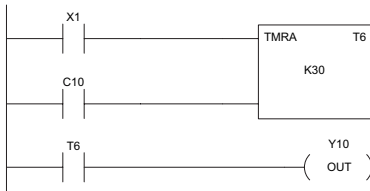


NOTE: * Both the Timer discrete status bits and the current value are accessed with the same data reference with the HPP. **DirectSOFT** uses separate references, such as "T2" for discrete status bit for Timer T2, and "TA2" for the current value of Timer T2.

Accumulating Timer Example using Discrete Status Bits

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

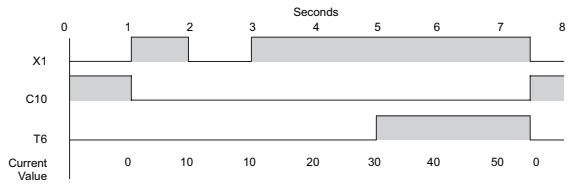
DirectSOFT



Handheld Programmer Keystrokes

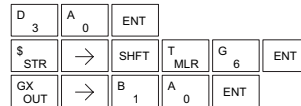


Timing Diagram



1/10th Seconds

Handheld Programmer Keystrokes (cont'd)

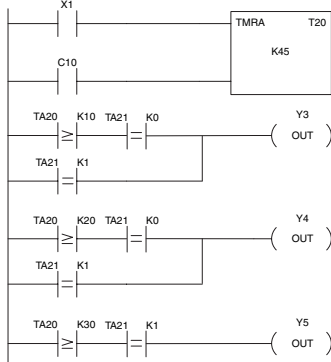


Accumulator Timer Example Using Comparative Contacts

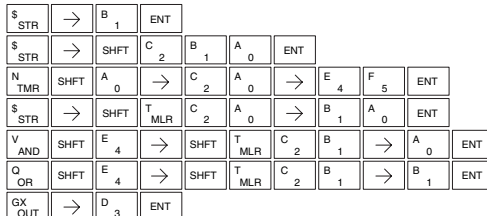
In the following example, a two-input timer is used with a preset of 4.5 seconds. Comparative contacts are used to energize Y3, Y4, and Y5 at one-second intervals respectively. The comparative contacts will turn off when the timer is reset.

Contacts

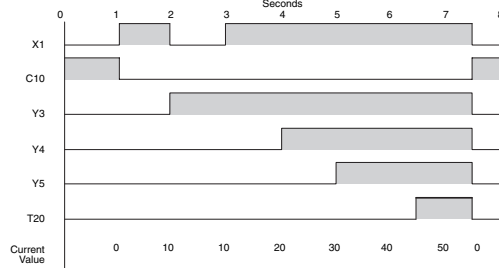
DirectSOFT



Handheld Programmer Keystrokes

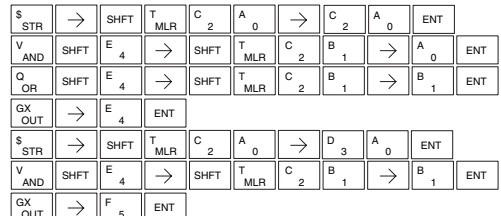


Timing Diagram



1/10th Seconds

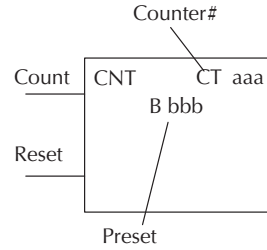
Handheld Programmer Keystrokes (cont'd)



Counter (CNT)

- ✓ 230
- ✓ 240
- ✓ 250-1
- ✓ 260

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 zero. 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.



| | |
|-----|------|
| DS | Used |
| HPP | Used |

Instruction Specifications

Counter Reference (CTaaa): Specifies the counter number.

Preset Value (Bbbb): Constant value (K) or a V-memory location. (Pointer (P) for DL240, DL250-1 and DL260.)

Current Values: Counter current values are accessed by referencing the associated V or CT memory locations. The V-memory location is the counter location + 1000. For example, the counter current value for CT3 resides in V-memory location V1003.

Discrete Status Bit: The discrete status bit is accessed by referencing the associated CT memory location. It will be on if the value is equal to or greater than the preset value. For example, the discrete status bit for Counter 2 would be CT2.



NOTE: A V-memory preset is required only if the ladder program or an OIT must be used to change the preset.

| Operand Data Type | | DL230 Range | | DL240 Range | | DL250-1 Range | | DL260 Range | |
|------------------------------|-------|----------------------|-----------|-----------------------|-----------|-----------------------|--------------------------|-----------------------|--------------------------|
| | B | aaa | bbb | aaa | bbb | aaa | bbb | aaa | bbb |
| Counters | CT | 0-77 | | 0-177 | | 0-177 | | 0-377 | |
| V-memory for preset values | V | — | 2000-2377 | — | 2000-3777 | — | 1400-7377 10000-17777 | — | 1400-7377 10000-37777 |
| Pointers (presets only) | P | | | | 2000-3777 | | 1400-7377 10000-17777 | | 1400-7377 10000-37777 |
| Constants (presets only) | K | — | 0-9999 | — | 0-9999 | — | 0-9999 | — | 0-9999 |
| Counter discrete status bits | CT/V* | 0-77 or V41140-41143 | | 0-177 or V41140-41147 | | 0-177 or V41140-41147 | | 0-377 or V41100-41157 | |
| Counter current values | V/CT* | 1000-1077 | | 1000-1177 | | 1000-1177 | | 1000-1377 | |

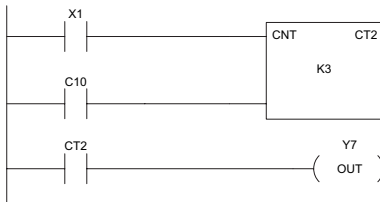


NOTE: * Both the Counter discrete status bits and the current value are accessed with the same data reference with the HPP. **DirectSOFT** uses separate references, such as “CT2” for discrete status bit for Counter CT2, and “CTA2” for the current value of Counter CT2.

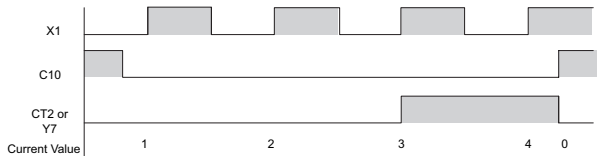
Counter Example Using Discrete Status Bits

In the following example, when X1 makes an off-to-on transition, counter CT2 will increment by one. When the current value reaches the preset value of 3, the counter status bit CT2 will turn on and energize Y7. When the reset C10 turns on, the counter status bit will turn off and the current value will be 0. The current value for counter CT2 will be held in V-memory location V1002.

DirectSOFT



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 |

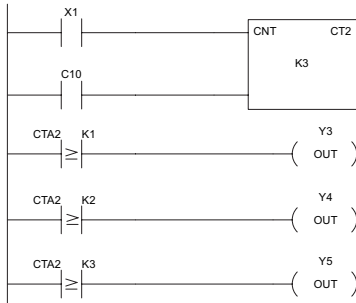
Handheld Programmer Keystrokes (cont)

| | | | |
|--------|---|----------|--------------------|
| \$ STR | → | SHFT C 2 | SHFT T MLR C 2 ENT |
| GX OUT | → | H 7 | ENT |

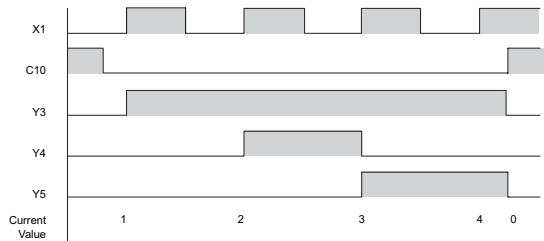
Counter Example Using Comparative Contacts

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

DirectSOFT



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)

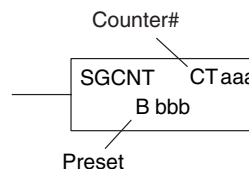
| | | | |
|--------|-----|----------|----------------|
| \$ STR | → | SHFT C 2 | SHFT T MLR C 2 |
| → | C 2 | ENT | |
| GX OUT | → | E 4 | ENT |
| \$ STR | → | SHFT C 2 | SHFT T MLR C 2 |
| → | D 3 | ENT | |
| GX OUT | → | F 5 | ENT |

Stage Counter (SGCNT)

- ✓ 230
- ✓ 240
- ✓ 250-1
- ✓ 260

| | |
|-----|------|
| DS | Used |
| HPP | Used |

The Stage Counter is a single-input counter that increments when the input logic transitions from off to on. This counter differs from other counters since it will hold its current value until reset using the RST instruction. The Stage Counter is designed for use in RLL^{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.(Pointer (P) for DL240, DL250-1 and DL260.)

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: When using a counter inside a stage, the stage must be active for one scan before the input to the counter makes a 0-1 transition. Otherwise, there is no real transition and the counter will not count.



NOTE: A V-memory preset is required only if the ladder program or an OIT must be used to change the preset.

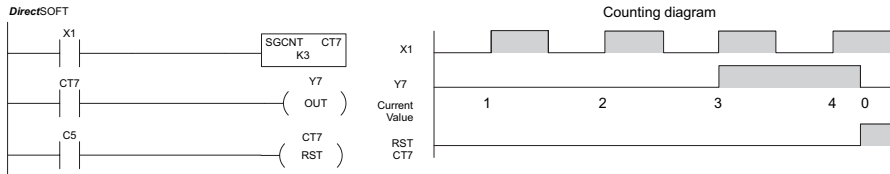
| Operand Data Type | | DL230 Range | | DL240 Range | | DL250-1 Range | | DL260 Range | |
|------------------------------|------|----------------------|-----------|-----------------------|-----------|-----------------------|--------------------------|-----------------------|--------------------------|
| | B | aaa | bbb | aaa | bbb | aaa | bbb | aaa | bbb |
| Counters | CT | 0-77 | | 0-177 | | 0-177 | | 0-377 | |
| V-memory for preset values | V | – | 2000-2377 | – | 2000-3777 | – | 1400-7377 10000-17777 | – | 1400-7377 10000-37777 |
| Pointers (presets only) | P | | | | 2000-3777 | | 1400-7377 10000-17777 | | 1400-7377 10000-37777 |
| Constants (presets only) | K | – | 0-9999 | – | 0-9999 | – | 0-9999 | – | 0-9999 |
| Counter discrete status bits | CT/V | 0-77 or V41140-41143 | | 0-177 or V41140-41147 | | 0-177 or V41140-41147 | | 0-377 or V41140-41157 | |
| Counter current values | V/CT | 1000-1077 | | 1000-1177 | | 1000-1177 | | 1000-1377 | |



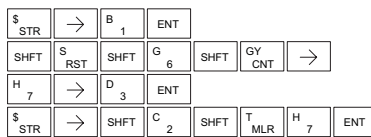
NOTE: * Both the Counter discrete status bits and the current value are accessed with the same data reference with the HPP. **DirectSOFT** uses separate references, such as “CT2” for discrete status bit for Counter CT2, and “CTA2” for the current value of Counter CT2.

Stage Counter Example Using Discrete Status Bits

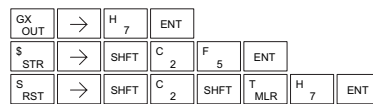
In the following example, when X1 makes an off-to-on transition, stage counter CT7 will increment by one. When the current value reaches 3, the counter status bit CT7 will turn on and energize Y7. The counter status bit CT7 will remain on until the counter is reset using the RST instruction. When the counter is reset, the counter status bit will turn off and the counter current value will be 0. The current value for counter CT7 will be held in V-memory location V1007.



Handheld Programmer Keystrokes

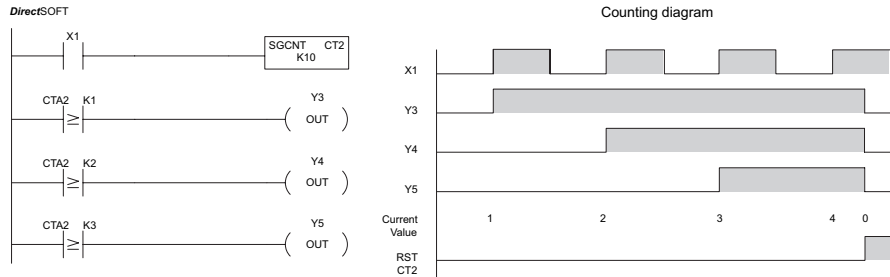


Handheld Programmer Keystrokes (cont)

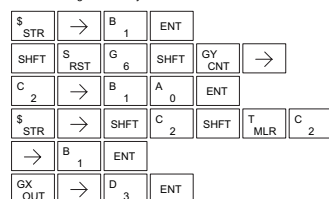


Stage Counter Example Using Comparative Contacts

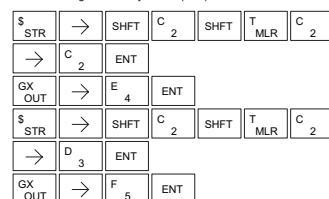
In the following example, when X1 makes an off-to-on transition, counter CT2 will increment by one. Comparative contacts are used to energize Y3, Y4, and Y5 at different counts. Although this is not shown in the example, when the counter is reset using the Reset instruction, the counter status bit will turn off and the current value will be 0. The current value for counter CT2 will be held in V-memory location V1002.



Handheld Programmer Keystrokes



Handheld Programmer Keystrokes (cont)



Up Down Counter (UDC)

- ✓ 230
- ✓ 240
- ✓ 250-1
- ✓ 260

| | |
|-----|------|
| DS | Used |
| HPP | Used |

This Up/Down Counter counts up on each off-to-on transition of the Up input and counts down on each off to on transition of the Down input. The counter is reset to 0 when the Reset input is on. The count range is 0 to 99999999. The count input not being used must be off in order for the active count input to function.

Instruction Specification

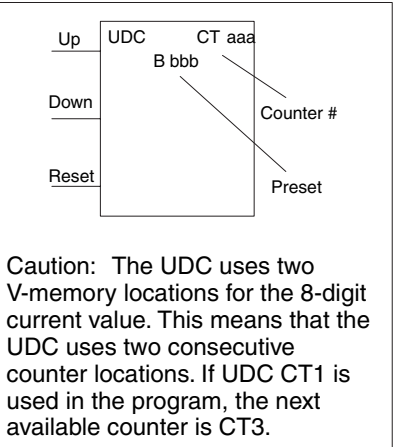
Counter Reference (CTaaa): Specifies the counter number.

Preset Value (Bbbb): Constant value (K) or two consecutive V-memory locations. (Pointer (P) for DL240, DL250-1 and DL260).

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. It will be on if the value is equal to or greater than the preset value. For example, the discrete status bit for Counter 2 would be CT2.

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



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

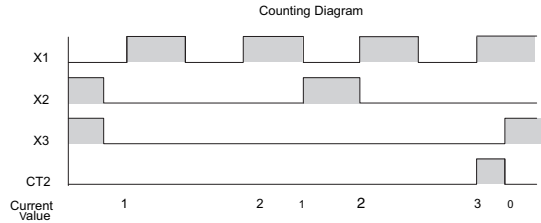
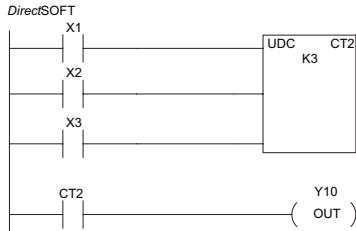
NOTE: A V-memory preset is required only if the ladder program or an OIT must be used to change the preset.

| Operand Data Type | DL230 Range | | DL240 Range | | DL250-1 Range | | DL260 Range | |
|------------------------------------|-------------|----------------------|-----------------------|-----------------------|----------------------------|----------------------------|----------------------------|----------------------------|
| | B | aaa bbb | aaa bbb | aaa bbb | aaa bbb | aaa bbb | aaa bbb | aaa bbb |
| Counters CT | | 0-76 | 0-176 | 0-176 | 0-176 | 0-376 | 0-376 | |
| V-memory for preset values V | | – 2000-2377 | – 2000-3777 | – 2000-3777 | – 1400-7377 10000-17777 | – 1400-7377 10000-17777 | – 1400-7377 10000-17777 | – 1400-7377 10000-17777 |
| Pointers (presets only) P | | | | 2000-3777 | | 1400-7377 10000-17777 | | 1400-7377 10000-17777 |
| Constants (presets only) K | | – 0-99999999 | – 0-99999999 | – 0-99999999 | – 0-99999999 | – 0-99999999 | – 0-99999999 | – 0-99999999 |
| Counter discrete status bits CT/V* | | 0-76 or V41140-41143 | 0-176 or V41140-41147 | 0-176 or V41140-41147 | 0-176 or V41140-41147 | 0-376 or V41100-41157 | 0-376 or V41100-41157 | |
| Counter current values V/CT* | | 1000-1076 | 1000-1176 | 1000-1176 | 1000-1176 | 1000-1376 | 1000-1376 | |

NOTE: * Both the Counter discrete status bits and the current value are accessed with the same data reference with the HPP. **DirectSOFT** 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

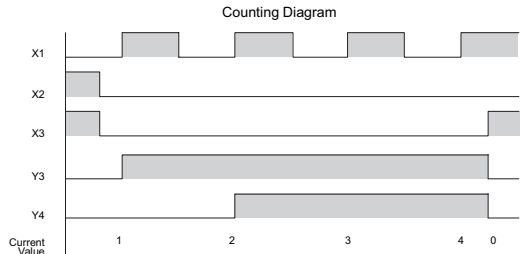
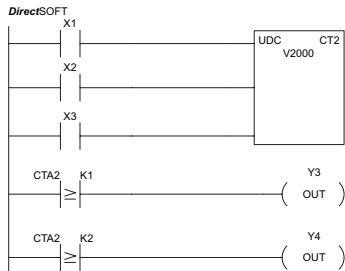
| | | | | | |
|-----------|------------------|----------------|----------------|---|----------------|
| \$ STR | → | B ₁ | ENT | | |
| \$ STR | → | C ₂ | ENT | | |
| \$ STR | → | D ₃ | ENT | | |
| SHFT | U _{ISG} | D ₃ | C ₂ | → | C ₂ |

Handheld Programmer Keystrokes (cont)

| | | | | | | | | | | |
|-------|----------------|----------------|----------------|------|------------------|----------------|-----|--|--|--|
| → | D ₃ | ENT | | | | | | | | |
| \$STR | → | SHFT | C ₂ | SHFT | T _{MLR} | C ₂ | ENT | | | |
| GXOUT | → | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | </ |
|--------|---|----------------|-----|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|----|

Handheld Programmer Keystrokes (cont)

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|----------------|-----|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
| → | B ₁ | ENT | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|----------------|-----|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|

Shift Register (SR)

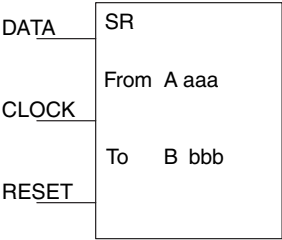
- ✓

230
- ✓

240
- ✓

250-1
- ✓

260
- The Shift Register instruction shifts data through a predefined number of control relays. The control ranges in the shift register block must start at the beginning of an 8-bit boundary and use 8-bit blocks.
- The Shift Register has three contacts.

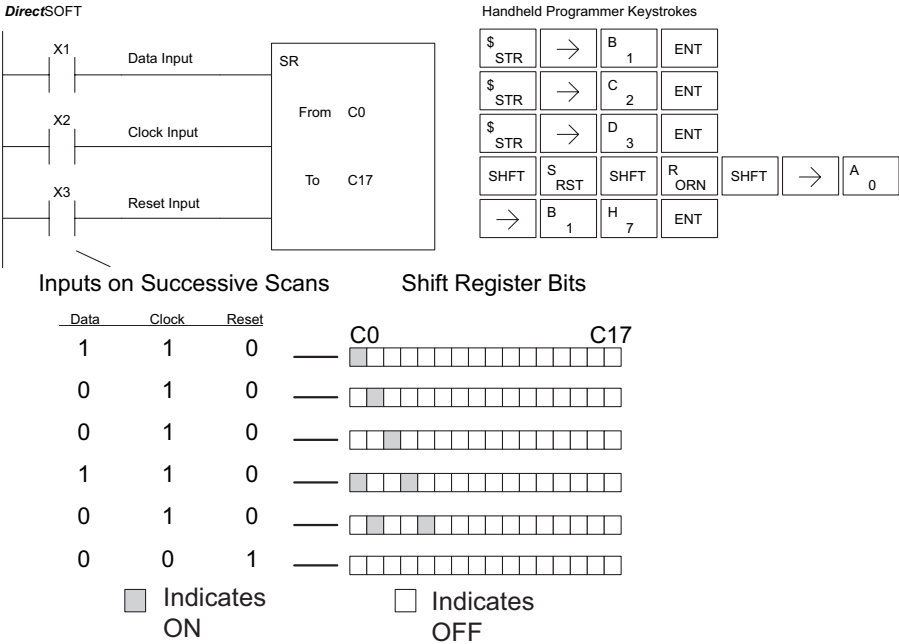


| | |
|-----|------|
| DS | Used |
| HPP | Used |

- 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 16 bits to be shifted from left to right. From C17 to C0 would define a block of 16 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 | | DL230 Range | | DL240 Range | | DL250-1 Range | | DL260 Range | |
|-------------------|---|-------------|-------|-------------|-------|---------------|--------|-------------|--------|
| A/B | | aaa | bbb | aaa | bbb | aaa | bbb | aaa | bbb |
| Control Relay | C | 0-377 | 0-377 | 0-377 | 0-377 | 0-1777 | 0-1777 | 0-3777 | 0-3777 |



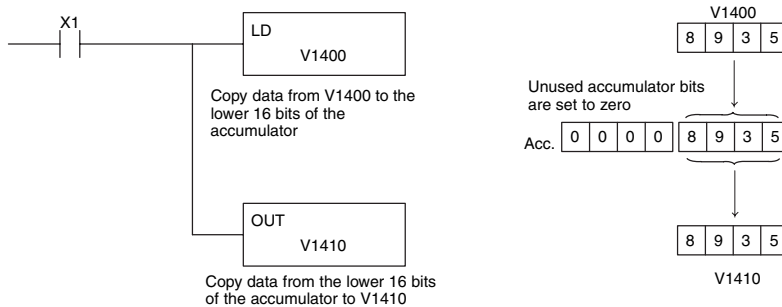
Accumulator/Stack Load and Output Data Instructions

Using the Accumulator

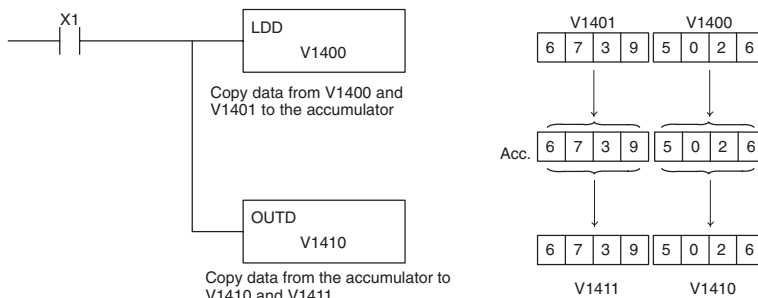
The accumulator in the DL205 series CPUs is a 32-bit register that is used as a temporary storage location for data that is being copied or manipulated in some manner. For example, you have to use the accumulator to perform math operations, such as, add, subtract, multiply, etc. Since there are 32 bits, you can use up to an 8-digit BCD number or a 32-bit 2's complement number. The accumulator is reset to 0 at the end of every CPU scan.

Copying Data to the Accumulator

The Load and Out instructions and their variations are used to copy data from a V-memory location to the accumulator, or to copy data from the accumulator to V-memory. The following example copies data from V-memory location V1400 to V-memory location V1410.

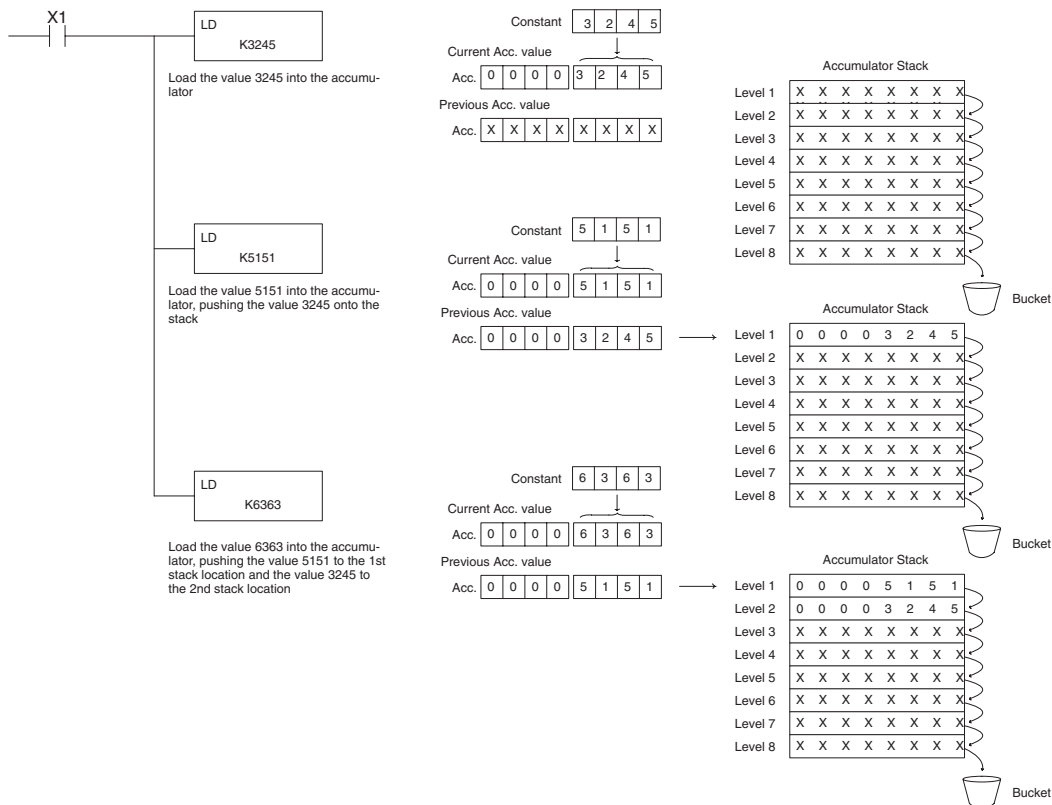


Since the accumulator is 32 bits and V-memory locations are 16 bits, the Load Double and Out Double (or variations thereof) use two consecutive V-memory locations or 8-digit BCD constants to copy data either to the accumulator from a V-memory address or from a V-memory address to the accumulator. For example, if you wanted to copy data from V1400 and V1401 to V1410 and V1411, the most efficient way to perform this function would be as follows:

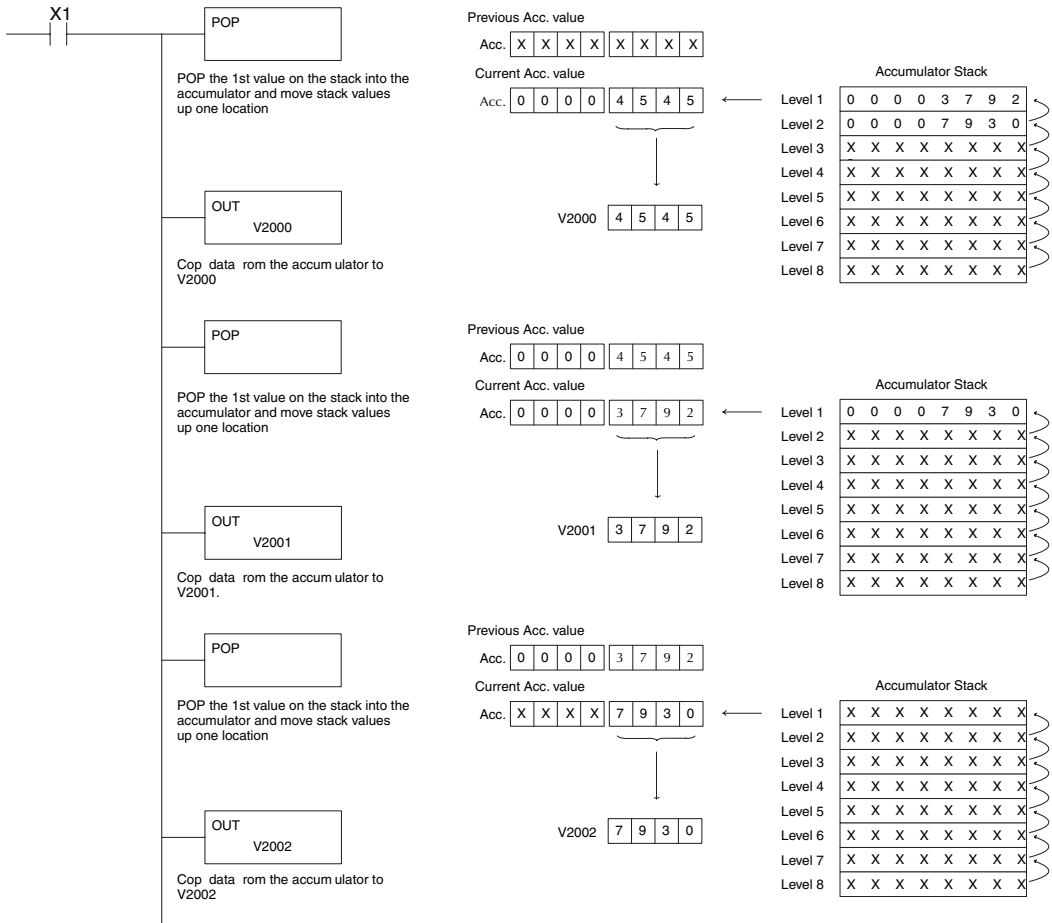


Using the Accumulator Stack

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.

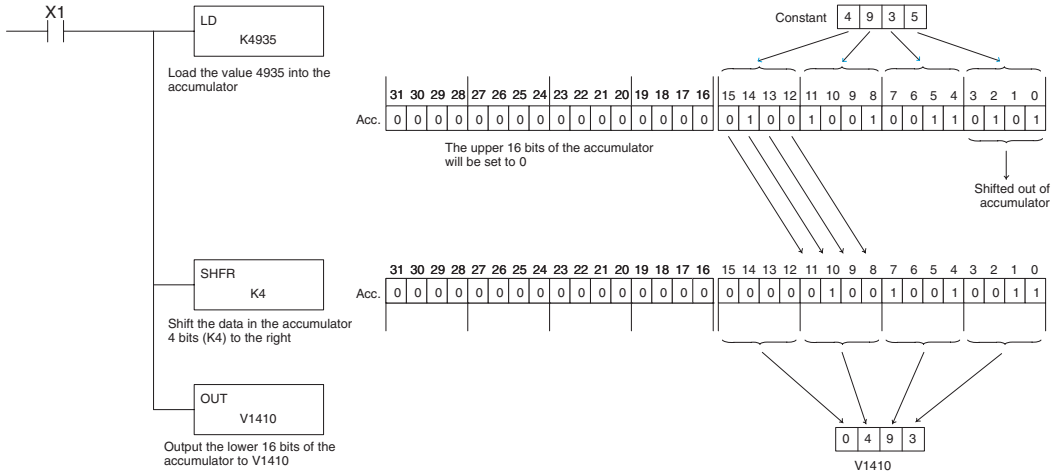


The POP instruction rotates values upward through the stack into the accumulator. When a POP is executed, the value that was in the accumulator is cleared and the value that was on top of the stack is in the accumulator. The values in the stack are shifted up one position in the stack.



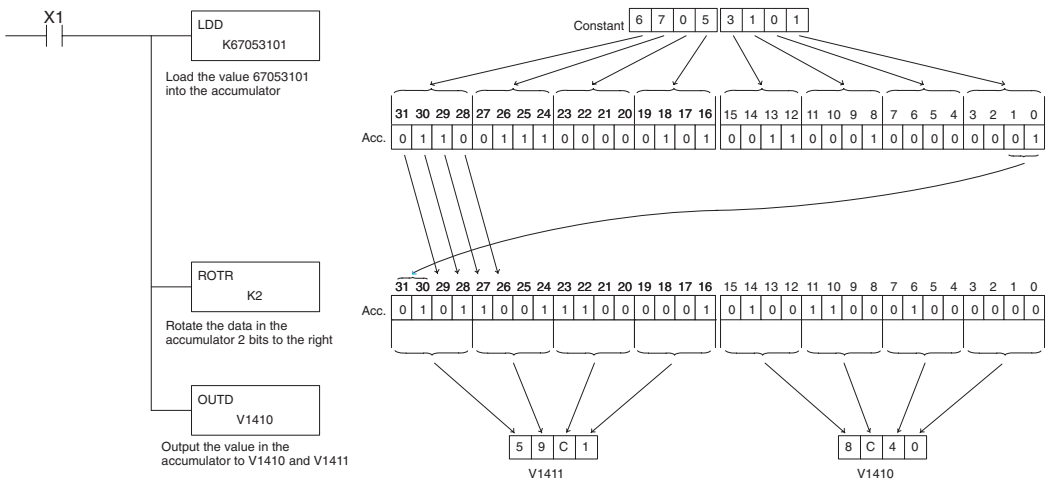
Changing the Accumulator Data

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 V1410.



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.

The following example rotates the value 67053101 two bits to the right and outputs the value to V1410 and V1411.



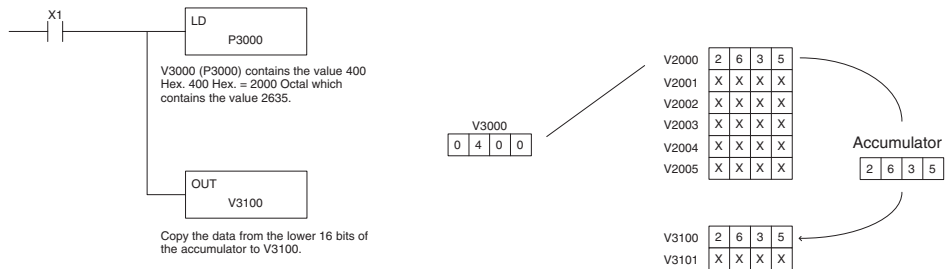
Using Pointers

Many of the DL205 series instructions will allow V-memory pointers as an operand. Pointers can be useful in ladder logic programming, but can be difficult to understand or implement in your application if you do not have prior experience with pointers (commonly known as indirect addressing). Pointers allow instructions to obtain data from V-memory locations referenced by the pointer value.

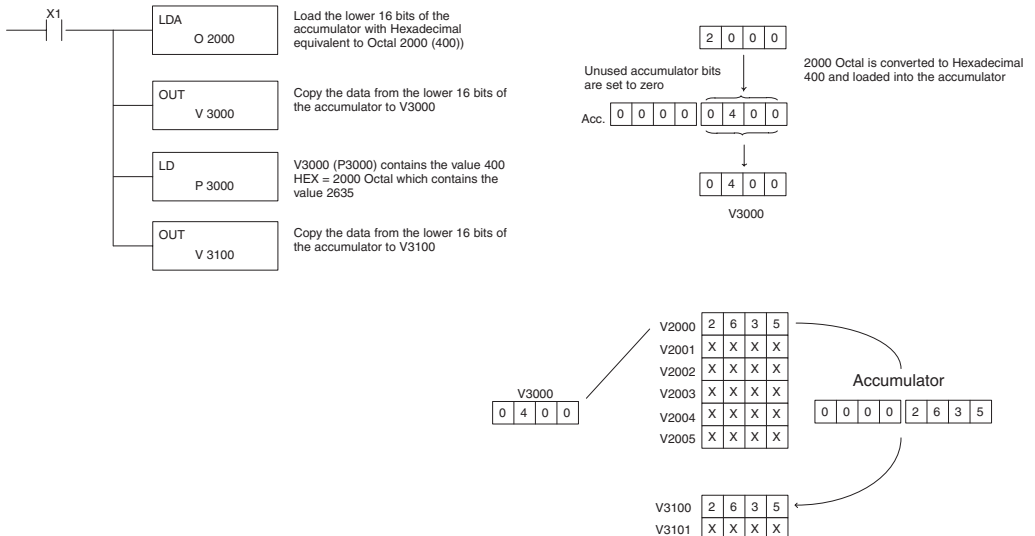


NOTE: In the DL205, V-memory addressing is in octal. However the value in the pointer location which will reference a V-memory location is viewed as HEX. Use the Load Address instruction to move an address into the pointer location. This instruction performs the Octal to Hexadecimal conversion for you.

The following example uses a pointer operand in a Load instruction. V-memory location 3000 is the pointer location. V3000 contains the value 400, which is the HEX equivalent of the Octal address V-memory location V2000. The CPU copies the data from V2000 into the lower word of the accumulator.



The following example is similar to the one above, except for the LDA (load address) instruction, which automatically converts the Octal address to the Hex equivalent.



Load (LD)

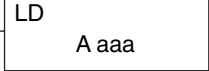
- ✓

230
- ✓

240
- ✓

250-1
- ✓

260
- 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 | | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range | |
|-------------------|---|---------------------|--------------------------------|--------------------------------|--------------------------------|-----|
| A | | aaa | aaa | aaa | aaa | bbb |
| V-memory | V | All. See Memory map | All. See Memory map | All See Memory map | All See Memory map | |
| Pointer | P | — | All V-memory See Memory map | All V-memory See Memory map | All V-memory See Memory map | |
| Constant | K | 0-FFFF | 0-FFFF | 0-FFFF | 0-FFFF | |

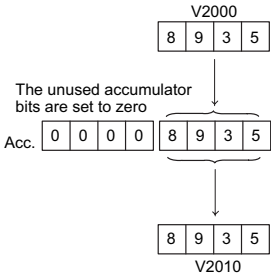
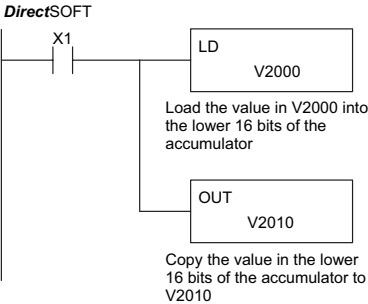
| Discrete Bit Flags | Description |
|--------------------|---|
| SP76 | On when the value loaded into the accumulator by any instruction is zero. |



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

| | |
|-----|------|
| DS | Used |
| HPP | Used |

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

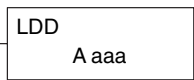


Handheld Programmer Keystrokes

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

Load Double (LDD)

- 230 The Load Double instruction is a 32-bit instruction that loads
- 240 the value (Aaaa), which is either two consecutive V-memory
- 250-1 locations or an 8-digit constant value, into the accumulator.
- 260



5

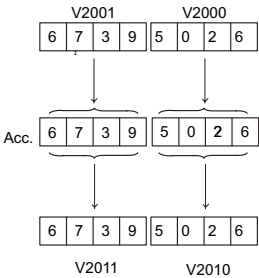
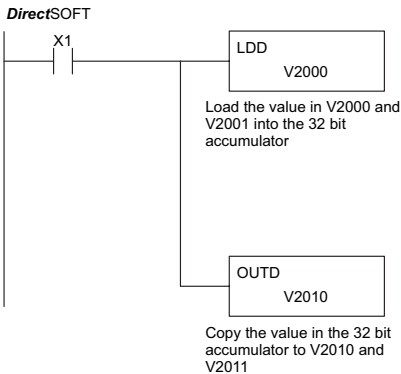
| Operand Data Type | | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|---|---------------------|--------------------------------|--------------------------------|--------------------------------|
| | A | aaa | aaa | aaa | aaa bbb |
| V-memory | V | All. See Memory map | All. See Memory map | All See Memory map | All See Memory map |
| Pointer | P | – | All V-memory See Memory map | All V-memory See Memory map | All V-memory See Memory map |
| Constant | K | 0-FFFFFFF | 0-FFFFFFF | 0-FFFFFFF | 0-FFFFFFF |

| 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.

| | | |
|-----|------|---|
| DS | Used | 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. |
| HPP | Used | |



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)

- ✗

230

✓

240

✓

250-1

✓

260
- The Load Formatted instruction loads 1 to 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.

LDF

A aaa

K bbb

| Operand Data Type | DL240 Range | | | DL250-1 Range | | DL260 Range | |
|-------------------|-------------|----------------|------|---------------|------|-------------|------|
| | A | aaa | bbb | aaa | bbb | aaa | bbb |
| Inputs | X | 0-477 | – | 0-777 | – | 0-1777 | – |
| Outputs | Y | 0-477 | – | 0-777 | – | 0-1777 | – |
| Control Relays | C | 0-377 | – | 0-1777 | – | 0-3777 | – |
| Stage bits | S | 0-777 | – | 0-1777 | – | 0-1777 | – |
| Timer bits | T | 0-177 | – | 0-377 | – | 0-377 | – |
| Counter bits | CT | 0-177 | – | 0-177 | – | 0-377 | – |
| Special Relays | SP | 0-137, 540-617 | – | 0-777 | – | 0-777 | – |
| Global I/O | GX/GY | – | – | – | – | 0-3777 | – |
| Constant | K | – | 1-32 | – | 1-32 | – | 1-32 |

| Discrete Bit Flags | Description |
|--------------------|---|
| SP76 | On when the value loaded into the accumulator by any instruction is zero. |

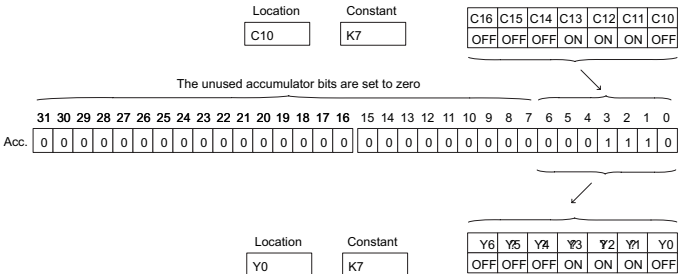
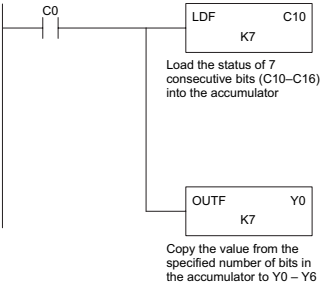


NOTE: Two consecutive Load instructions will place the value of the first Load instruction onto the accumulator stack.

| | |
|-----|------|
| DS | Used |
| HPP | Used |

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

DirectSOFT

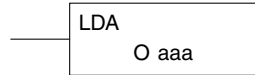


Handheld Programmer Keystrokes

| | | | | | | | | |
|------|-----|-------|------|---|-----|---|---|---------|
| \$ | 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)

- ✓ 230 The Load Address instruction is a 16-bit instruction. It
 - ✓ 240 converts any octal value or address to the HEX equivalent
 - ✓ 250-1 value and loads the HEX value into the accumulator. This
 - ✓ 260 instruction is useful when an address parameter is required
- since all addresses for the DL205 system are in octal.



| Operand Data Type | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|-------------|--------------------------------|--------------------------------|--------------------------------|
| | aaa | aaa | aaa | aaa |
| Octal Address | 0 | All V-memory See Memory map | All V-memory See Memory map | All V-memory See Memory map |

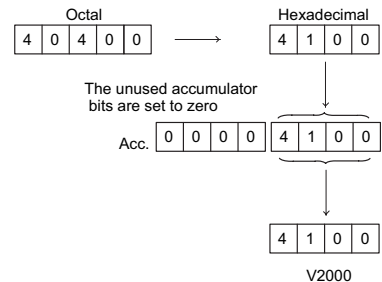
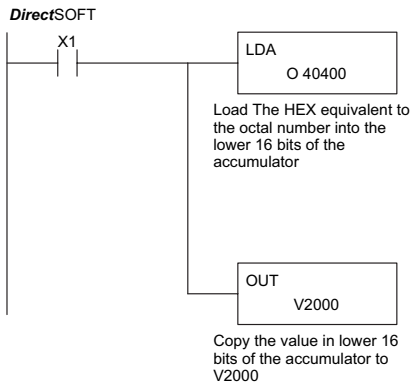
| Discrete Bit Flags | Description |
|--------------------|---|
| SP76 | On when the value loaded into the accumulator by any instruction is zero. |



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.

| | |
|-----|------|
| DS | Used |
| HPP | Used |



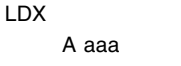
Handheld Programmer Keystrokes

| | | | | | | | | | |
|------|-----|-------|------|---|-----|---|---|-----|-----|
| \$ | 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 |

Load Accumulator Indexed (LDX)

- 230
- 240
- 250-1
- 260

Load Accumulator Indexed is a 16-bit instruction that specifies a source address (V-memory) which will be offset by the value in the first stack location. This instruction interprets the value in the first stack location as HEX. The value in the offset address (source address + offset) is loaded into the lower 16 bits of the accumulator. The upper 16 bits of the accumulator are set to 0.



| | |
|-----|------|
| DS | Used |
| HPP | Used |

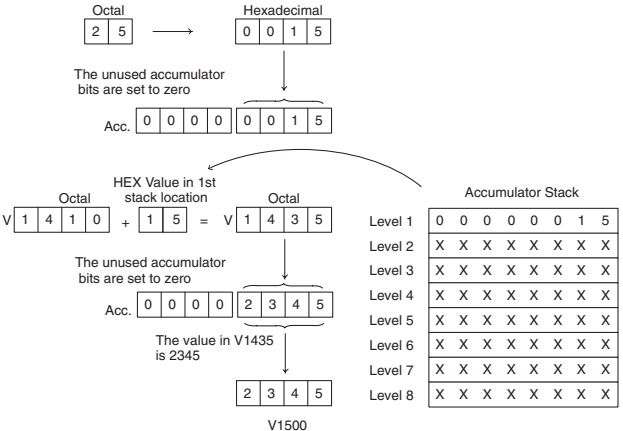
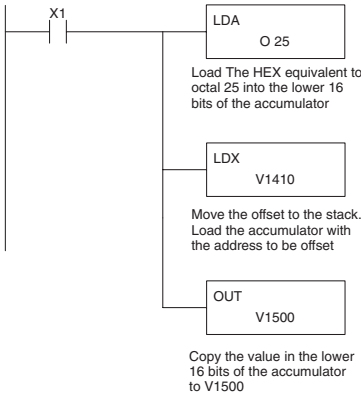
Helpful hint: — The Load Address instruction can be used to convert an octal address to a HEX address and load the value into the accumulator.

| Operand Data Type | | DL250-1 Range | DL260 Range |
|-------------------|---|------------------------------|------------------------------|
| | A | aaa | aaa |
| V-memory | V | All. See memory map | All. See memory map |
| Pointer | P | All V-memory. See memory map | All V-memory. See memory map |



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

In the following example, when X1 is on, the HEX equivalent for octal 25 will be loaded into the accumulator (this value will be placed on the stack when the Load Accumulator Indexed instruction is executed). V-memory location V1410 will be added to the value in the first level of the stack and the value in this location (V1435 = 2345) is loaded into the lower 16 bits of the accumulator using the Load Accumulator Indexed instruction. The value in the lower 16 bits of the accumulator is output to V1500 using the Out instruction.



| | |
|-----|------|
| DS | Used |
| HPP | Used |

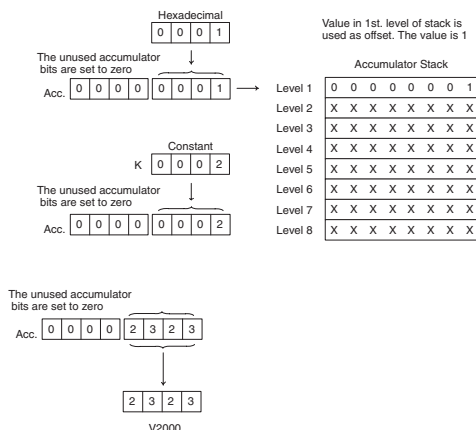
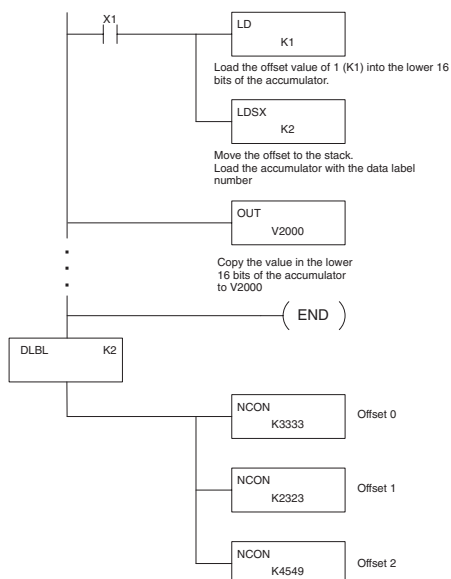
| | |
|------|-------|
| LDSX | K aaa |
|------|-------|

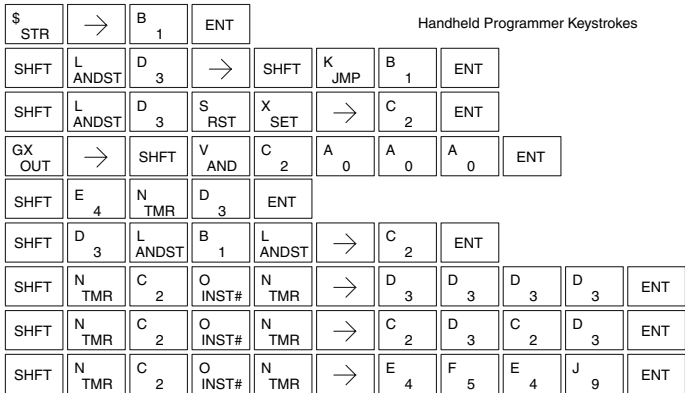
Helpful hint: — The Load Address instruction can be used to convert octal to HEX and load the value into the accumulator.

| Operand Data Type | | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|---|-------------|---------------|-------------|
| | | aaa | aaa | aaa |
| Constant | K | 1-FFFF | 1-FFFF | 1-FFFF |



In the following example, when X1 is on, the offset of 1 is loaded into the accumulator. This value will be placed into the first level of the accumulator stack when the LDSX instruction is executed. The LDSX instruction specifies the Data Label (DLBL K2) where the numerical constant(s) are located in the program and loads the constant value, indicated by the offset in the stack, into the lower 16 bits of the accumulator.

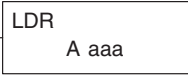




Load Real Number (LDR)

- 230
- 240
- 250-1
- 260

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



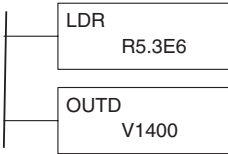
| | |
|-----|------|
| DS | Used |
| HPP | N/A |

| Operand Data Type | DL250-1 Range | DL260 Range |
|-------------------|----------------------------------|----------------------------------|
| | aaa | aaa |
| V-memory | All. See memory map | All. See memory map |
| Pointer | All V-memory. See memory map | All V-memory. See memory map |
| Real Constants | -3.402823E+038 to +3.402823E+038 | -3.402823E+038 to +3.402823E+038 |

DirectSOFT allows you to enter real numbers directly, by using the leading “R” to indicate a *real number* entry. You can enter a constant such as Pi, shown in the example to the right. To enter negative numbers, use a minus (–) after the “R”.

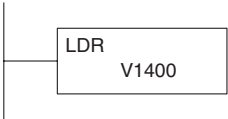


For very large numbers or very small numbers, you can use exponential notation. The number to the right is 5.3 million. The OUTD instruction stores it in V1400 and V1401.



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

The previous example above stored a real number in V1400 and V1401. Suppose that now we want to retrieve that number. Just use the Load Real with the V data type, as shown to the right. Next we could perform real math on it, or convert it to a binary number.



Out (OUT)

- ✓ 230
- ✓ 240
- ✓ 250-1
- ✓ 260

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

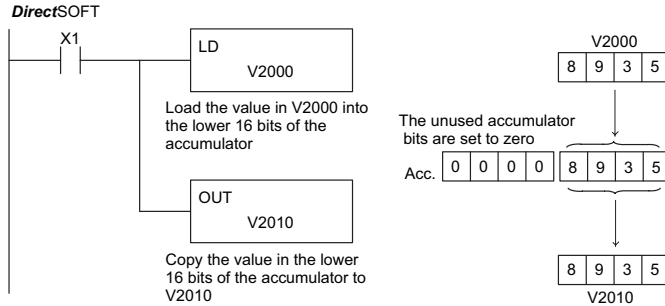
OUT
A aaa

| Operand Data Type | | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|---|---------------------|------------------------------|------------------------------|------------------------------|
| | A | aaa | aaa | aaa | aaa |
| V-memory | V | All. See memory map | All. See memory map | All. See memory map | All. See memory map |
| Pointer | P | - | All V-memory. See memory map | All V-memory. See memory map | All V-memory. See memory map |

| Discrete Bit Flags | Description |
|--------------------|---|
| SP76 | On when the value loaded into the accumulator by any instruction is zero. |

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.

| | |
|-----|------|
| DS | Used |
| HPP | Used |



Handheld Programmer Keystrokes

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

Out Double (OUTD)

- ✓

230
- ✓

240
- ✓

250-1
- ✓

260
- The Out Double instruction is a 32-bit instruction that copies the value in the accumulator to two consecutive V-memory locations at a specified starting location (Aaaa).

OUTD

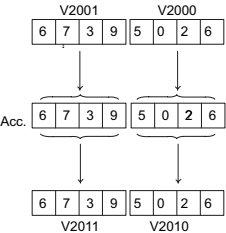
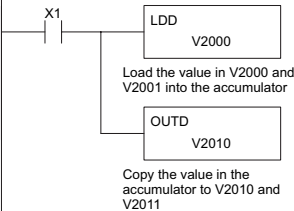
A aaa

| Operand Data Type | | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|---|---------------------|------------------------------|------------------------------|------------------------------|
| A | | aaa | aaa | aaa | aaa |
| V-memory | V | All. See memory map | All. See memory map | All. See memory map | All. See memory map |
| Pointer | P | - | All V-memory. See memory map | All V-memory. See memory map | All V-memory. See memory map |

In the following example, when X1 is on, the 32-bit value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The value in the accumulator is output to V2010 and V2011 using the Out Double instruction.

| | |
|-----|------|
| DS | Used |
| HPP | Used |

DirectSOFT



Handheld Programmer Keystrokes

| | | | | | |
|------|-----|-------|---|---|---------|
| \$ | STR | → | B | 1 | ENT |
| SHFT | L | ANDST | D | 3 | D 3 → |
| C | A | 0 | A | 0 | A 0 ENT |
| GX | OUT | SHFT | D | 3 | → |
| C | A | 0 | B | 1 | A 0 ENT |

Out Formatted (OUTF)

- ☒ 230 The Out Formatted instruction outputs 1 to 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.
- ☒ 240
- ☒ 250-1
- ☒ 260

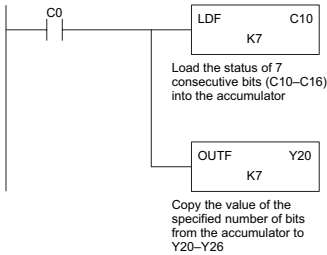
OUTF A aaa
 K bbb

| | |
|-----|------|
| DS | Used |
| HPP | Used |

| Operand Data Type | DL240 Range | | DL250-1 Range | | DL260 Range | |
|-------------------|-------------|------|---------------|------|-------------|------|
| A | aaa | bbb | aaa | bbb | aaa | bbb |
| Constant | K | 1-32 | | 1-32 | | 1-32 |

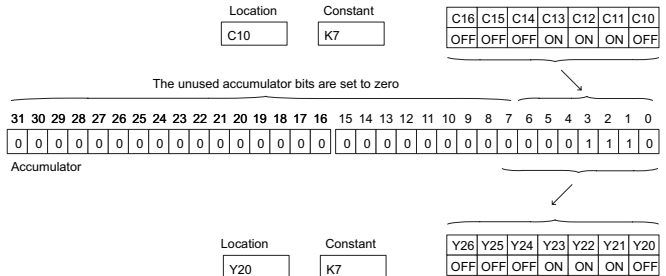
In the following example, when C0 is on, the binary pattern of C10–C16 (7 bits) will be loaded into the accumulator using the Load Formatted instruction. The lower 7 bits of the accumulator are output to Y20–Y26 using the Out Formatted instruction.

DirectSOFT



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 | → | | |
| C 2 | A 0 | → | H 7 | ENT | | |



Out Indexed (OUTX)

- 230
- 240
- 250-1
- 260

| | |
|-----|------|
| DS | Used |
| HPP | Used |

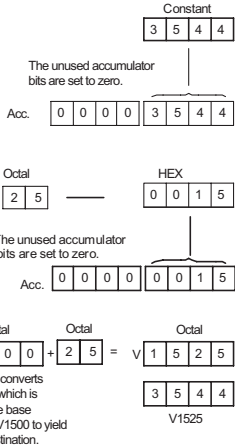
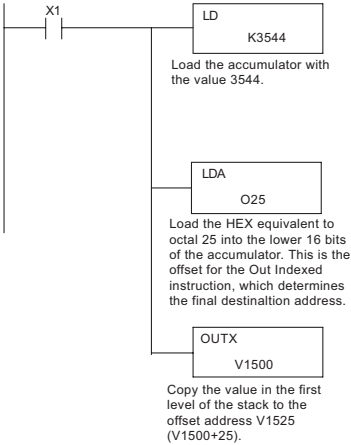
The Out Indexed instruction is a 16-bit instruction. It copies a 16-bit or 4-digit value from the first level of the accumulator stack to a source address offset by the value in the accumulator (V-memory + offset). This instruction interprets the offset value as a HEX number. The upper 16 bits of the accumulator are set to zero.

| |
|-------|
| OUTX |
| A aaa |

| Operand Data Type | | DL250-1 Range | DL260 Range |
|-------------------|---|------------------------------|------------------------------|
| | A | aaa | aaa |
| V-memory | V | All. See memory map | All. See memory map |
| Pointer | P | All V-memory. See memory map | All V-memory. See memory map |

In the following example, when X1 is on, the constant value 3544 is loaded into the accumulator. This is the value that will be output to the specified offset V-memory location (V1525). The value 3544 will be placed onto the stack when the Load Address instruction is executed. Remember, two consecutive Load instructions places the value of the first load instruction onto the stack. The Load Address instruction converts octal 25 to HEX 15 and places the value in the accumulator. The Out Indexed instruction outputs the value 3544, which resides in the first level of the accumulator stack to V1525.

DirectSOFT



| | |
|-------------------|-----------------|
| Accumulator Stack | |
| Level 1 | 0 0 0 0 3 5 4 4 |
| Level 2 | X X X X X X X X |
| Level 3 | X X X X X X X X |
| Level 4 | X X X X X X X X |
| Level 5 | X X X X X X X X |
| Level 6 | X X X X X X X X |
| Level 7 | X X X X X X X X |
| Level 8 | X X X X X X X X |

Handheld Programmer Keystrokes

| | | | | | | | | | | | | |
|--------|---------|------------------|----------------|----------------|----------------|----------------|----------------|----------------|-----|--|--|--|
| \$ | STR | → | B ₁ | ENT | | | | | | | | |
| SHFT | L ANDST | D ₃ | → | PREV | D ₃ | F ₅ | E ₄ | E ₄ | ENT | | | |
| SHFT | L ANDST | D ₃ | A ₀ | → | C ₂ | F ₅ | ENT | | | | | |
| GX OUT | SHFT | X _{SET} | → | B ₁ | F ₅ | A ₀ | A ₀ | ENT | | | | |

Out Least (OUTL)

- 230
- 240
- 250-1
- 260

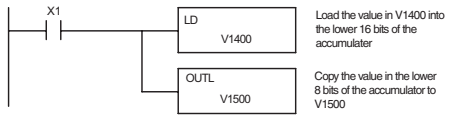
The Out Least instruction copies the value in the lower eight bits of the accumulator to the lower eight bits of the specified V-memory location (i.e., it copies the low byte of the low word of the accumulator).

In the following example, when X1 is on, the value in V1400 will be loaded into the lower 16 bits of the accumulator using the Load instruction. The value in the lower 8 bits of the accumulator are copied to V1500 using the Out Least instruction.



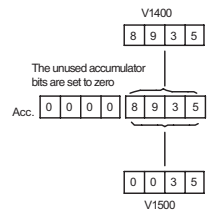
| | |
|-----|------|
| DS | Used |
| HPP | Used |

| Operand Data Type | | DL260 Range |
|-------------------|---|------------------------------|
| A | | aaa |
| V-memory | V | All V-memory. See memory map |
| Pointer | P | All V-memory. See memory map |



Handheld Programmer Keystrokes

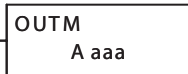
| | | | | | | | | | | | | | | | |
|-----------|------------|----------------|-----|----------------|----------------|----------------|----------------|-----|--|--|--|--|--|--|--|
| \$ | → | B ₁ | ENT | | | | | | | | | | | | |
| SHFT | L ANDST | D ₃ | → | B ₁ | E ₄ | A ₀ | A ₀ | ENT | | | | | | | |
| GX OUT | SHFT | L ANDST | → | B ₁ | F ₅ | A ₀ | A ₀ | ENT | | | | | | | |



Out Most (OUTM)

- 230
- 240
- 250-1
- 260

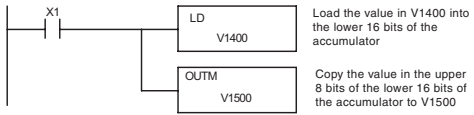
The Out Most instruction copies the value in the upper eight bits of the lower 16 bits of the accumulator to the upper eight bits of the specified V-memory location (i.e., it copies the high byte of the low word of the accumulator).



| | |
|-----|------|
| DS | Used |
| HPP | Used |

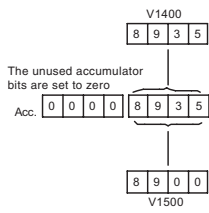
| Operand Data Type | | DL260 Range |
|-------------------|---|------------------------------|
| A | | aaa |
| V-memory | V | All V-memory. See memory map |
| Pointer | P | All V-memory. See memory map |

In the following example, when X1 is on, the value in V1400 will be loaded into the lower 16 bits of the accumulator using the Load instruction. The value in the upper 8 bits of the lower 16 bits of the accumulator are copied to V1500 using the Out Most instruction.



Handheld Programmer Keystrokes

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----|---|----------------|-----|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
| \$ | → | B ₁ | ENT | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----|---|----------------|-----|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|



Pop (POP)

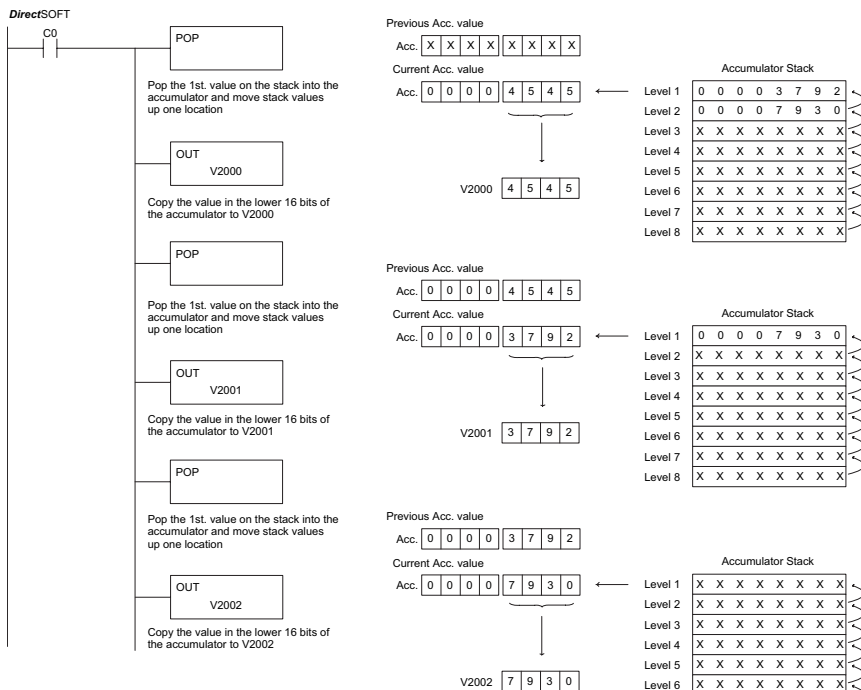
- ✓ 230
- ✓ 240
- ✓ 250-1
- ✓ 260

| | |
|-----|------|
| DS | Used |
| HPP | Used |

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

POP

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



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)

- 230
- 240
- 250-1
- 260

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.

AND
A aaa

| Operand Data Type | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|-------------|---------------------|------------------------------|------------------------------|
| | A | aaa | aaa | aaa |
| V-memory | V | All. See memory map | All. See memory map | All. See memory map |
| Pointer | P | - | All V-memory. See memory map | All V-memory. See memory map |

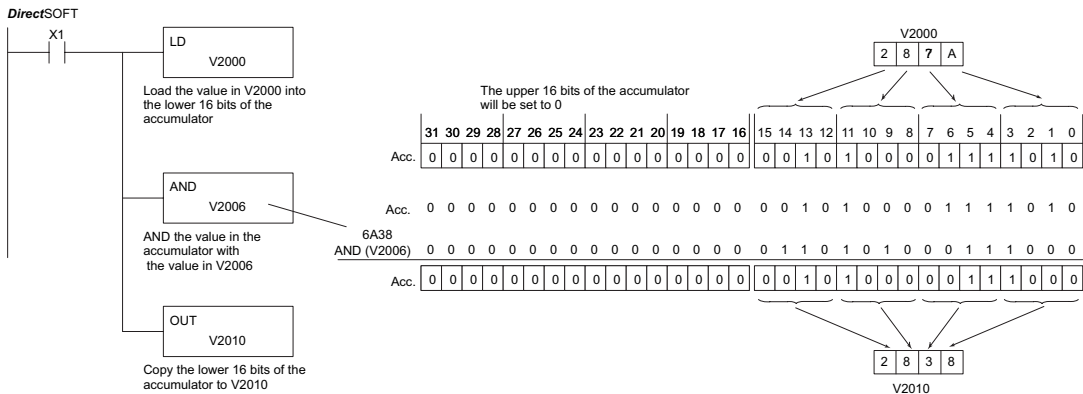
| Discrete Bit Flags | Description |
|--------------------|---|
| SP63 | On when the result of the instruction causes the value in the accumulator to be zero. |



NOTE: The status flags are only valid until another instruction that uses the same flags is executed.

In the following example, when 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.

| | |
|-----|------|
| DS | Used |
| HPP | Used |



Handheld Programmer Keystrokes

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

And Double (ANDD)

- ✓

230
- ✓

240
- ✓

250-1
- ✓

260
- DS

Used
- HPP

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

ANDD

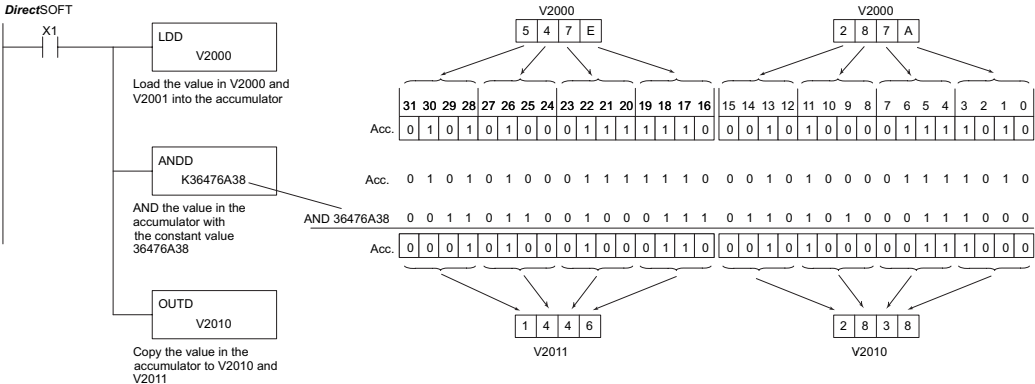
A aaa

| Operand Data Type | | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|---|-------------|-------------|------------------------------|------------------------------|
| A | | aaa | aaa | aaa | aaa |
| V-memory | V | - | - | All. See memory map | All. See memory map |
| Pointer | P | - | - | All V-memory. See memory map | All V-memory. See memory map |
| Constant | K | 0-FFFFFFF | 0-FFFFFFF | 0-FFFFFFF | 0-FFFFFFF |

| Discrete Bit Flags | Description |
|--------------------|---|
| SP63 | Will be on if the result in the accumulator is zero |
| SP70 | Will be on if the result in the accumulator is negative |

NOTE: The status flags are only valid until another instruction that uses the same flags is executed.

In the following example, when X1 is on, the value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The value in the accumulator is anded with 36476A38 using the And Double instruction. The value in the accumulator is output to V2010 and V2011 using the Out Double instruction.



Handheld Programmer Keystrokes

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|------|-----|-------|---|---|-----|------|---|-----|---|---|---|---|---|-----|---|---|---|---|------|---|---|------|---|---|---|---|-----|--|--|--|--|
| \$ | STR | → | B | 1 | ENT | | | | | | | | | | | | | | | | | | | | | | | | | | |
| SHFT | L | ANDST | D | 3 | → | 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 | | | | | | | | | | | | | | | | | |

And Formatted (ANDF)

- 230
- 240
- 250-1
- 260

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



5

| | |
|-----|------|
| DS | Used |
| HPP | Used |

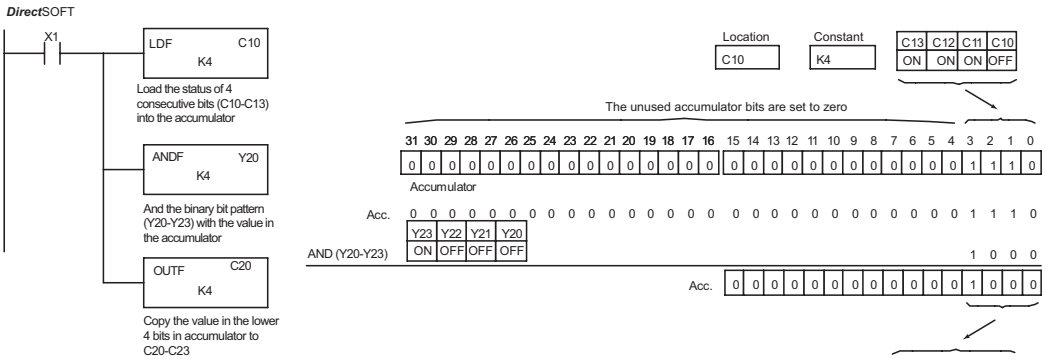
| Operand Data Type | | DL250-1 Range | | DL260 Range | |
|-------------------|-------|---------------|------|-------------|------|
| | A | aaa | bbb | aaa | bbb |
| Inputs | X | 0-777 | - | 0-1777 | - |
| Outputs | Y | 0-777 | - | 0-1777 | - |
| Control Relays | C | 0-1777 | - | 0-3777 | - |
| Stage bits | S | 0-1777 | - | 0-1777 | - |
| Timer bits | T | 0-377 | - | 0-377 | - |
| Counter bits | CT | 0-177 | - | 0-377 | - |
| Special Relays | SP | 0-777 | - | 0-777 | - |
| Global I/O | GX/GY | - | - | 0-3777 | - |
| Constant | K | - | 1-32 | - | 1-32 |

| Discrete Bit Flags | Description |
|--------------------|---|
| SP63 | Will be on if the result in the accumulator is zero |
| SP70 | Will be on if the result in the accumulator is negative |

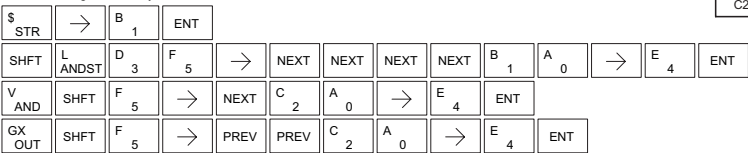


NOTE: Status flags are valid only until another instruction uses the same flag.

In the following example, when X1 is on, the Load Formatted instruction loads C10-C13 (4 binary bits) into the accumulator. The accumulator content is logically ANDed with the bit pattern from Y20-Y23 using the And Formatted instruction. The Out Formatted instruction outputs the accumulator's lower four bits to C20-C23.



Handheld Programmer Keystrokes



And with Stack (ANDS)

- 230
- 240
- 250-1
- 260

The And with Stack instruction is a 32-bit instruction that logically ANDs the value in the accumulator with the first level of the accumulator stack. The result resides in the accumulator. The value in the first level of the accumulator stack is removed from the stack and all values are moved up one level. Discrete status flags indicate if the result of the And with Stack is zero or a negative number (the most significant bit is on).

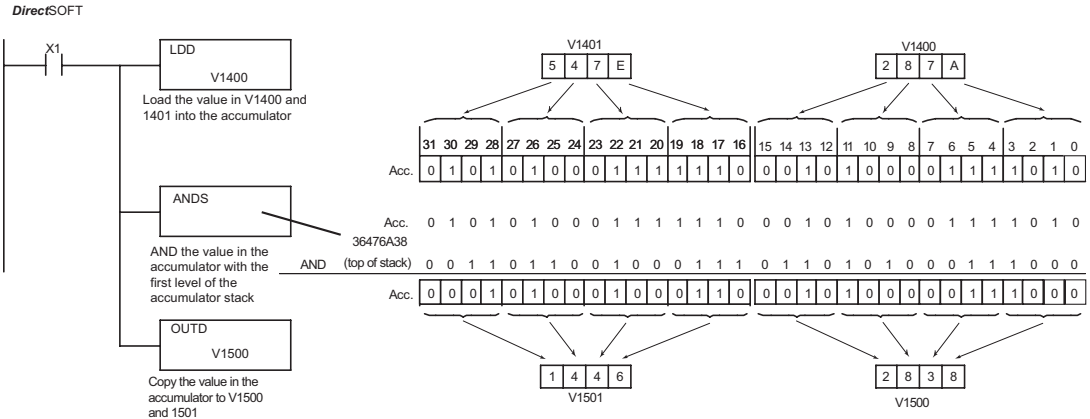
ANDS

| | |
|-----|------|
| DS | Used |
| HPP | Used |

| Discrete Bit Flags | Description |
|--------------------|---|
| SP63 | Will be on if the result in the accumulator is zero |
| SP70 | Will be on if the result in the accumulator is negative |

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

In the following example, when X1 is on, the binary value in the accumulator will be anded with the binary value in the first level of the accumulator stack. The result resides in the accumulator. The 32-bit value is then output to V1500 and V1501.



Handheld Programmer Keystrokes

| | | | |
|--------|---------|-------|-----|
| \$ STR | → | B 1 | ENT |
| SHFT | L ANDST | D 3 | D 3 |
| V AND | SHFT | S RST | ENT |
| GX OUT | SHFT | D 3 | → |
| | | B 1 | F 5 |
| | | A 0 | A 0 |
| | | | ENT |

Or (OR)

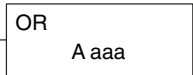
- ✓

230
- ✓

240
- ✓

250-1
- ✓

260
- 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.



| | |
|-----|------|
| DS | Used |
| HPP | Used |

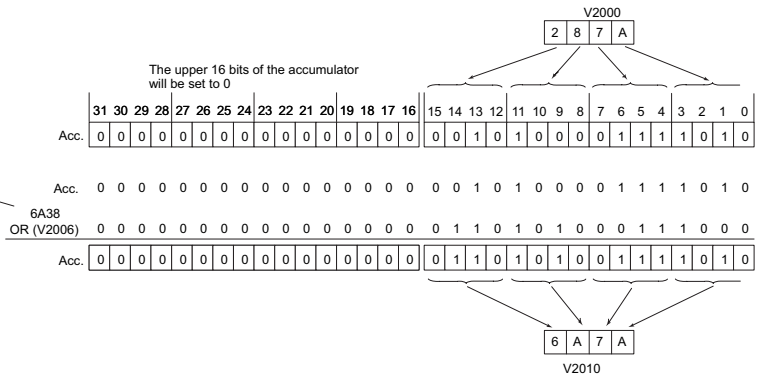
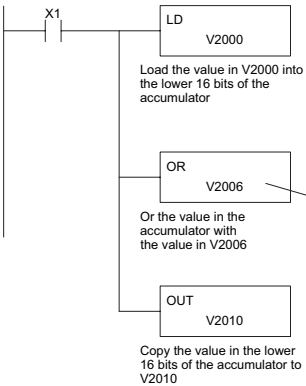
| Operand Data Type | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|--------------------|-------------|---|---------------------------------|---------------------------------|
| A | aaa | aaa | aaa | aaa |
| V-memory | V | All See memory map | All See memory map | All See memory map |
| Pointer | P | - | All V-memory. See memory map | All V-memory. See memory map |
| 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 OR'd with V2006 using the OR instruction. The value in the lower 16 bits of the accumulator are output to V2010 using the Out instruction.

DirectSOFT

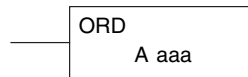


Handheld Programmer Keystrokes

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

Or Double (ORD)

- | | |
|-------|--|
| 230 | The Or Double is a 32-bit instruction that ORs the value in the accumulator with the value (Aaaa) or an 8-digit (max) constant value. The result resides in the accumulator. |
| 240 | |
| 250-1 | Discrete status flags indicate if the result of the Or Double is zero or a negative number (the most significant bit is on). |
| 260 | |



| | |
|-----|------|
| DS | Used |
| HPP | Used |

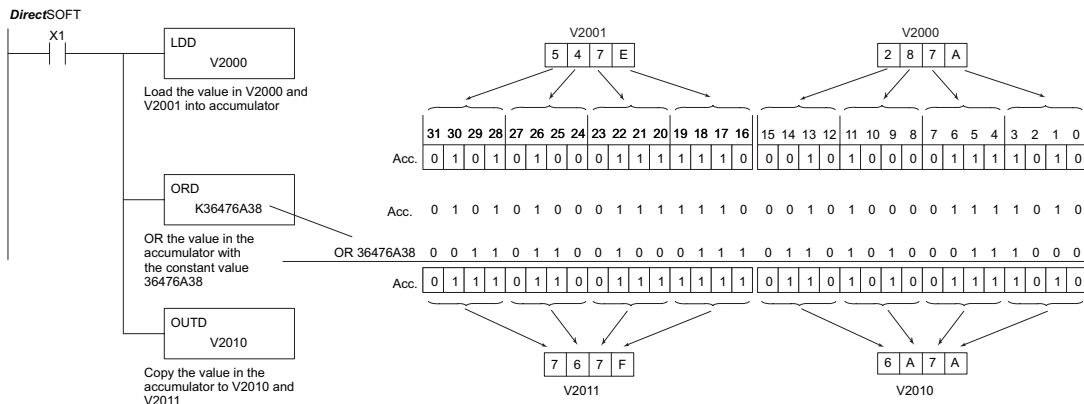
| Operand Data Type | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|-------------|-------------|---------------------------------|---------------------------------|
| A | aaa | aaa | aaa | aaa |
| V-memory | V | - | All. See memory map | All. See memory map |
| Pointer | P | - | All V-memory. See memory map | All V-memory. See memory map |
| Constant | K | 0-FFFFFFFF | 0-FFFFFFFF | 0-FFFFFFFF |

| Discrete Bit Flags | Description |
|--------------------|---|
| SP63 | Will be on if the result in the accumulator is zero |
| SP70 | Will be on if the result in the accumulator is negative |



NOTE: The status flags are only valid until another instruction that uses the same flags is executed.

In the following example, when X1 is on, the value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The value in the accumulator is OR'd with 36476A38 using the Or Double instruction. The value in the accumulator is output to V2010 and V2011 using the Out Double instruction.



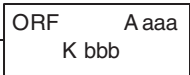
Handheld Programmer Keystrokes

| | | | | | | | | | | | | | | | | | | | |
|---------|--------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|-------|----------------|-------|----------------|----------------|-----|--|--|--|
| \$ STR | → | B ₁ | ENT | | | | | | | | | | | | | | | | |
| SHIFT | LANDST | D ₃ | D ₃ | → | C ₂ | A ₀ | A ₀ | A ₀ | ENT | | | | | | | | | | |
| Q OR | SHIFT | D ₃ | → | SHIFT | K JMP | D ₃ | G ₆ | E ₄ | H ₇ | G ₆ | SHIFT | A ₀ | SHIFT | D ₃ | I ₈ | ENT | | | |
| GX QUIT | SHIFT | D ₃ | → | C ₂ | A ₀ | B ₁ | A ₀ | ENT | | | | | | | | | | | |

Or Formatted (ORF)

- 230
- 240
- 250-1
- 260

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



| | |
|-----|------|
| DS | Used |
| HPP | Used |

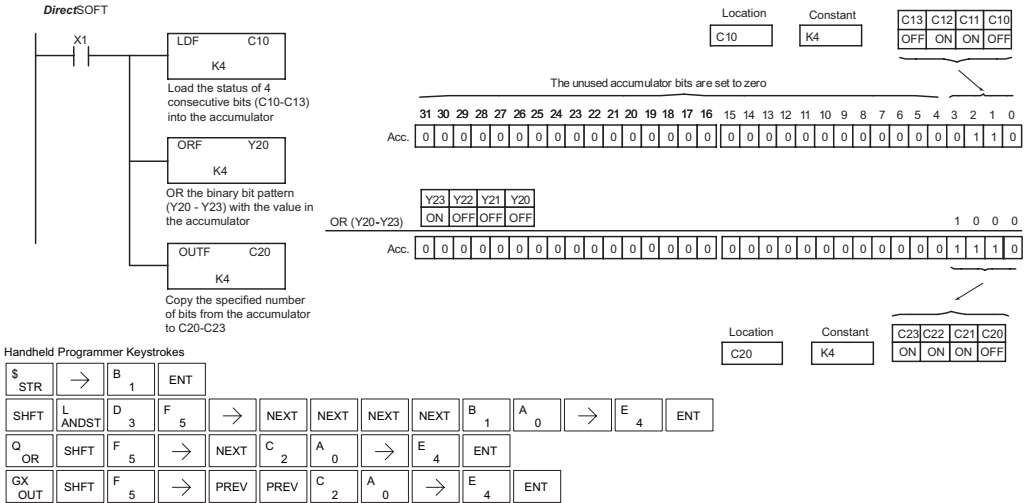
| Operand Data Type | | DL250-1 Range | | DL260 Range | |
|-------------------|-------|---------------|------|-------------|------|
| | A | aaa | bbb | aaa | bbb |
| Inputs | X | 0-777 | - | 0-1777 | - |
| Outputs | Y | 0-777 | - | 0-1777 | - |
| Control Relays | C | 0-1777 | - | 0-3777 | - |
| Stage bits | S | 0-1777 | - | 0-1777 | - |
| Timer bits | T | 0-377 | - | 0-377 | - |
| Counter bits | CT | 0-177 | - | 0-377 | - |
| Special Relays | SP | 0-777 | - | 0-777 | - |
| Global I/O | GX/GY | - | - | 0-3777 | - |
| Constant | K | - | 1-32 | - | 1-32 |

| Discrete Bit Flags | Description |
|--------------------|---|
| SP63 | Will be on if the result in the accumulator is zero |
| SP70 | Will be on if the result in the accumulator is negative |



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

In the following example, when X1 is on the Load Formatted instruction loads C10-C13 (4 binary bits) into the accumulator. The Or Formatted instruction logically ORs the accumulator contents with Y20-Y23 bit pattern. The Out Formatted instruction outputs the accumulator's lower four bits to C20-C23.



Or with Stack (ORS)

- 230
- 240
- 250-1
- 260

| | |
|-----|------|
| DS | Used |
| HPP | Used |

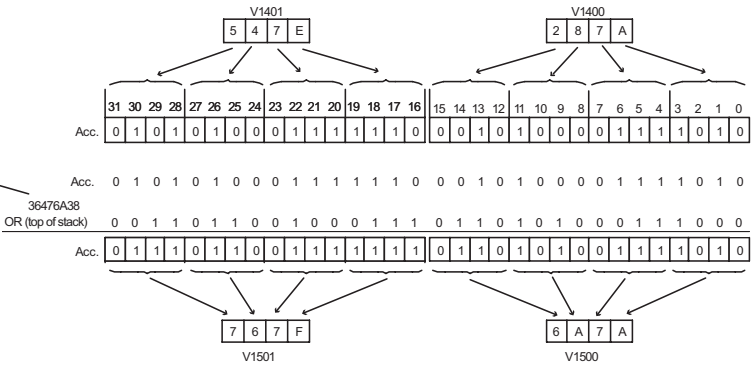
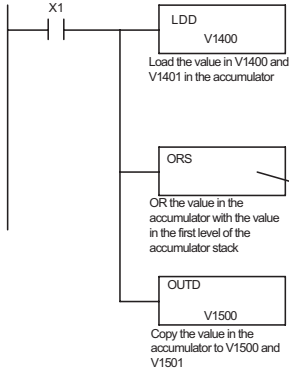
The Or with Stack instruction is a 32-bit instruction that logically ORs the value in the accumulator with the first level of the accumulator stack. The result resides in the accumulator. The value in the first level of the accumulator stack is removed from the stack and all values are moved up one level. Discrete status flags indicate if the result of the Or with Stack is zero or a negative number (the most significant bit is on).

ORS

| Discrete Bit Flags | Description |
|--------------------|---|
| SP63 | Will be on if the result in the accumulator is zero |
| SP70 | Will be on if the result in the accumulator is negative |

In the following example, when X1 is on, the binary value in the accumulator will be ORed with the binary value in the first level of the stack. The result resides in the accumulator.

DirectSOFT



Handheld Programmer Keystrokes

| | | | | | | | | | | | | | | | | | | | |
|--------|---------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|-----|--|--|--|--|--|--|--|--|--|--|
| \$ | STR | → | B ₁ | ENT | | | | | | | | | | | | | | | |
| SHFT | L ANDST | D ₃ | D ₃ | → | B ₁ | E ₄ | A ₀ | A ₀ | ENT | | | | | | | | | | |
| Q OR | SHFT | S RST | ENT | | | | | | | | | | | | | | | | |
| GX OUT | SHFT | D ₃ | → | B ₁ | F ₅ | A ₀ | A ₀ | ENT | | | | | | | | | | | |

Exclusive Or (XOR)

- ☒ **230** 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 accumulator. The discrete status flag indicates if the result of the XOR is zero.
- ☒ **240**
- ☒ **250-1**
- ☒ **260**

XOR
A aaa

| | |
|-----|------|
| DS | Used |
| HPP | Used |

| Operand Data Type | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|-------------|-----------------------|---------------------------------|---------------------------------|
| | A | aaa | aaa | aaa |
| V-memory | V | All See memory map | All See memory map | All See memory map |
| Pointer | P | - | All V-memory. See memory map | All V-memory. See memory map |

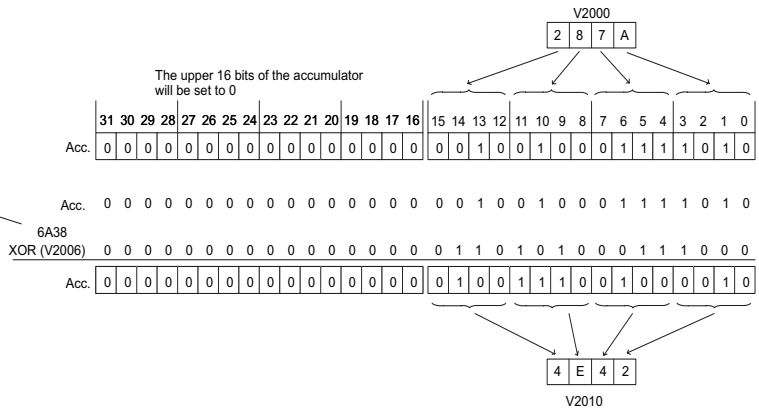
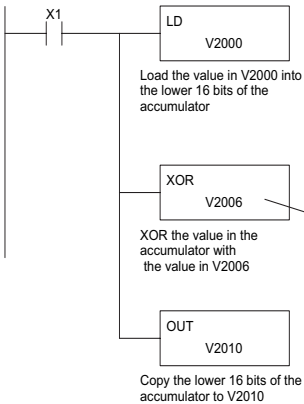
| Discrete Bit Flags | Description |
|--------------------|---|
| SP63 | Will be on if the result in the accumulator is zero |
| SP70 | Will be on if the result in the accumulator is negative |



NOTE: The status flags are only valid until another instruction that uses the same flags is executed.

In the following example, when X1 is on, the value in V2000 will be loaded into the accumulator using the Load instruction. The value in the accumulator is exclusive OR'd with V2006 using the Exclusive Or instruction. The value in the lower 16 bits of the accumulator are output to V2010 using the Out instruction.

DirectSOFT

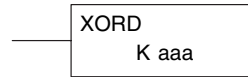


Handheld Programmer Keystrokes

| | | | | | | | | | | | | | | | | | | |
|------|---|----------------|-----|----------------|----------------|----------------|----------------|----------------|----------------|----------------|-----|--|--|--|--|--|--|--|
| \$ | → | B ₁ | ENT | | | | | | | | | | | | | | | |
| SHFT | L | D ₃ | → | SHFT | V | C ₂ | A ₀ | A ₀ | A ₀ | ENT | | | | | | | | |
| SHFT | X | SHFT | Q | → | SHFT | V | C ₂ | A ₀ | A ₀ | G ₆ | ENT | | | | | | | |
| GX | → | SHFT | V | C ₂ | A ₀ | B ₁ | A ₀ | ENT | | | | | | | | | | |
| OUT | | | | | | | | | | | | | | | | | | |

Exclusive Or Double (XORD)

- | | |
|---------|--|
| ✓ 230 | <p>The Exclusive Or Double is a 32-bit instruction that performs an exclusive OR of the value in the accumulator and the value (Kaaa), which is 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).</p> |
| ✓ 240 | |
| ✓ 250-1 | |
| ✓ 260 | |



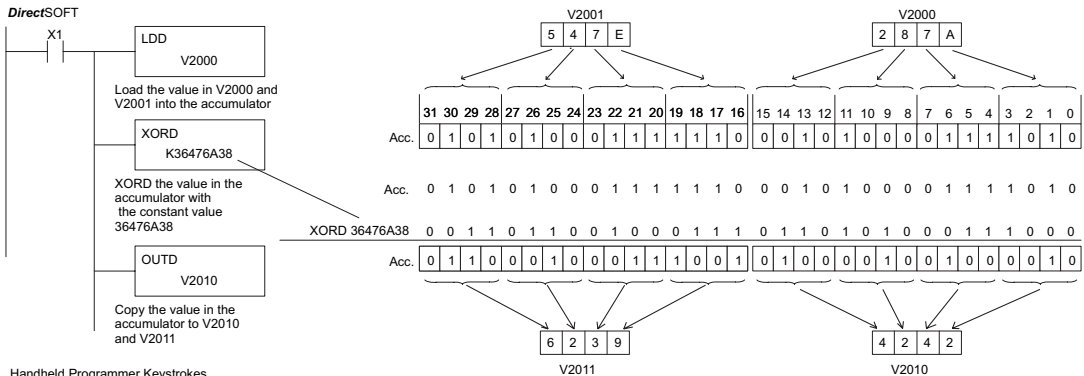
| | |
|-----|------|
| DS | Used |
| HPP | Used |

| Operand Data Type | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|-------------|-------------|---------------|-------------|
| | aaa | aaa | aaa | aaa |
| Constant K | 0-FFFFFFF | 0-FFFFFFF | 0-FFFFFFF | 0-FFFFFFF |

NOTE: The status flags are only valid until another instruction that uses the same flags is executed.

| Discrete Bit Flags | Description |
|--------------------|---|
| SP63 | Will be on if the result in the accumulator is zero |
| SP70 | Will be on if the result in the accumulator is negative |

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 OR'd 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

V2011

Exclusive OR Formatted (XORF)

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

The instruction requires a starting location (Aaaa) and the number of bits (Kbbb) to be exclusive OR'd. Discrete status flags indicate if the result of the Exclusive Or Formatted is zero or negative (the most significant bit is on).

| | |
|------|-------|
| XORF | A aaa |
| | K bbb |

| | |
|-----|------|
| DS | Used |
| HPP | Used |

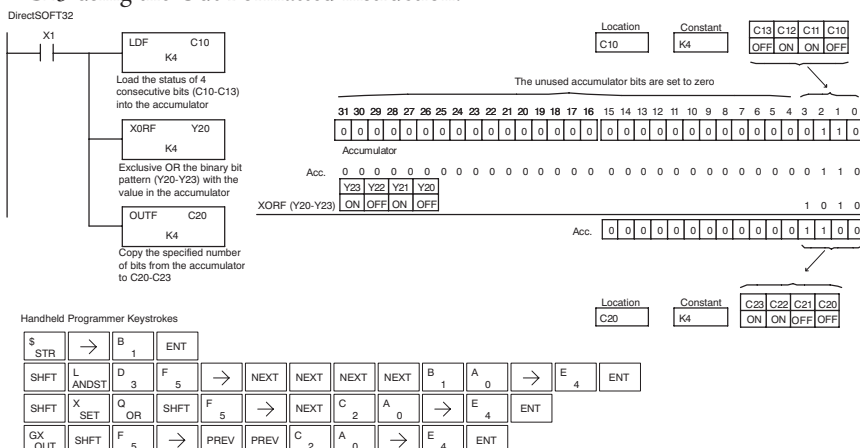
| Operand Data Type | | DL250-1 Range | | DL260 Range | |
|-------------------|-------|---------------|------|-------------|------|
| | A | aaa | bbb | aaa | bbb |
| Inputs | X | 0-777 | – | 0-1777 | – |
| Outputs | Y | 0-777 | – | 0-1777 | – |
| Control Relays | C | 0-1777 | – | 0-3777 | – |
| Stage bits | S | 0-1777 | – | 0-1777 | – |
| Timer bits | T | 0-377 | – | 0-377 | – |
| Counter bits | CT | 0-177 | – | 0-377 | – |
| Special Relays | SP | 0-777 | – | 0-777 | – |
| Global I/O | GX/GY | - | – | 0-3777 | – |
| Constant | K | - | 1-32 | - | 1-32 |

| Discrete Bit Flags | Description |
|--------------------|---|
| SP63 | Will be on if the result in the accumulator is zero |
| SP70 | Will be on if the result in the accumulator is negative |



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

In the following example, when X1 is on, the binary pattern of C10–C13 (4 bits) will be loaded into the accumulator using the Load Formatted instruction. The value in the accumulator will be logically Exclusive OR'd with the bit pattern from Y20–Y23 using the Exclusive Or Formatted instruction. The value in the lower 4 bits of the accumulator are output to C20–C23 using the Out Formatted instruction.



Exclusive Or with Stack (XORS)

- 230
- 240
- 250-1
- 260

| | |
|-----|------|
| DS | Used |
| HPP | Used |

The Exclusive Or with Stack instruction is a 32-bit instruction that performs an Exclusive Or of the value in the accumulator with the first level of the accumulator stack. The result resides in the accumulator. The value in the first level of the accumulator stack is removed from the stack and all values are moved up one level. Discrete status flags indicate if the result of the Exclusive Or with Stack is zero or a negative number (the most significant bit is on).

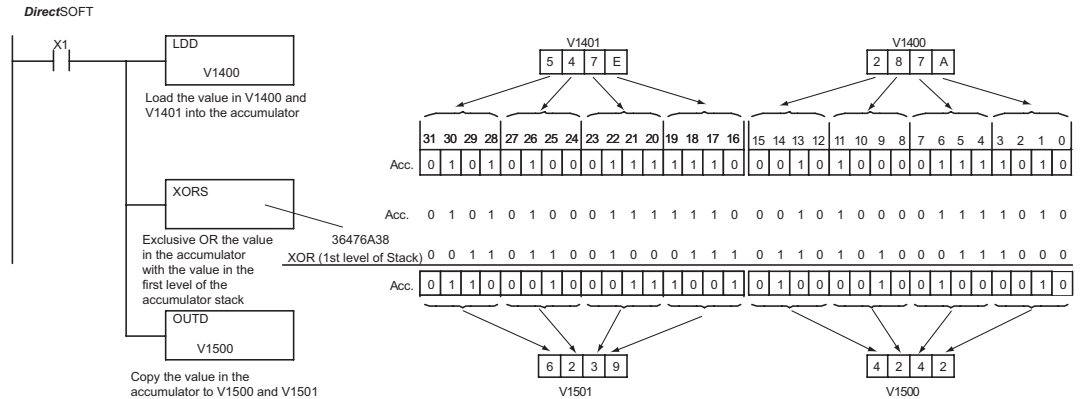
XORS

5

| Discrete Bit Flags | Description |
|--------------------|---|
| SP63 | Will be on if the result in the accumulator is zero |
| SP70 | Will be on if the result in the accumulator is negative |

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

In the following example, when X1 is on, the binary value in the accumulator will be Exclusive OR'd with the binary value in the first level of the accumulator stack. The result will reside in the accumulator.



Handheld Programmer Keystrokes

| | | | |
|--------------|----------|------------|---------------------|
| \$ STR | → | B 1 | ENT |
| SHFT L ANDST | D 3 | → | B 1 E 4 A 0 A 0 ENT |
| SHFT X SET | Q OR | SHFT S RST | ENT |
| GX OUT | SHFT D 3 | → | B 1 F 5 A 0 A 0 ENT |

Compare (CMP)

- ✓ 230
- ✓ 240
- ✓ 250-1
- ✓ 260

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

CMP
A aaa

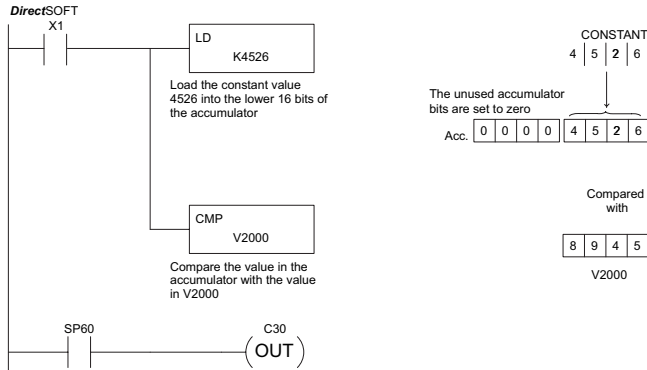
| | |
|-----|------|
| DS | Used |
| HPP | Used |

| Operand Data Type | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|----------------------------|---------------------------------|---------------------------------|---------------------------------|
| A | aaa | aaa | aaa | aaa |
| V-memory | V All See memory map | All See memory map | All See memory map | All See memory map |
| Pointer | P - | All V-memory. See memory map | All V-memory. See memory map | All V-memory. See memory map |

| Discrete Bit Flags | Description |
|--------------------|---|
| SP60 | On when the value in the accumulator is less than the instruction value. |
| SP61 | On when the value in the accumulator is equal to the instruction value. |
| SP62 | On when the value in the accumulator is greater than the instruction value. |

NOTE: The status flags are only valid until another instruction that uses the same flags is executed.

In the following example, when X1 is on, the constant 4526 will be loaded into the lower 16 bits of the accumulator using the Load instruction. The value in the accumulator is compared with the value in V2000 using the Compare instruction. The corresponding discrete status flag will be turned on indicating the result of the comparison. In this example, if the value in the accumulator is less than the value specified in the Compare instruction, SP60 will turn on, energizing contact 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)

- ✓ 230

✓ 240

✓ 250-1

✓ 260
- 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.

CMPD
A aaa

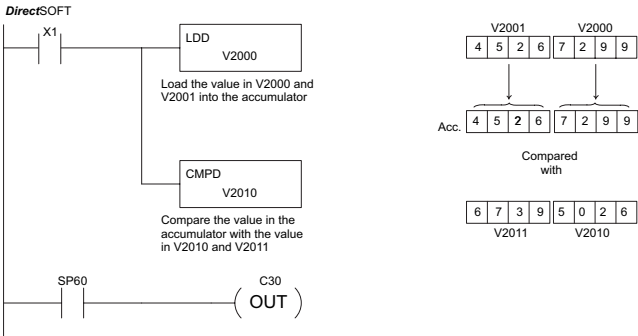
| | |
|-----|------|
| DS | Used |
| HPP | Used |

| Operand Data Type | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|----------------------------|---------------------------------|---------------------------------|---------------------------------|
| A | aaa | aaa | aaa | aaa |
| V-memory | V All See memory map | All See memory map | All See memory map | All See memory map |
| Pointer | P - | All V-memory. See memory map | All V-memory. See memory map | All V-memory. See memory map |
| Constant | K 0-FFFFFFF | 0-FFFFFFF | 0-FFFFFFF | 0-FFFFFFF |

| Discrete Bit Flags | Description |
|--------------------|--|
| SP60 | On when the value in the accumulator is less than the instruction value |
| SP61 | On when the value in the accumulator is equal to the instruction value |
| SP62 | On when the value in the accumulator is greater than the instruction value |

NOTE: The status flags are only valid until another instruction that uses the same flags is executed.

In the following example, when X1 is on, the value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The value in the accumulator is compared with the value in V2010 and V2011 using the CMPD instruction. The corresponding discrete status flag will be turned on indicating the result of the comparison. In this example, if the value in the accumulator is less than the value specified in the Compare instruction, SP60 will turn on, energizing contact C30.



Handheld Programmer Keystrokes

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

Compare Formatted (CMPF)

- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260

The Compare Formatted compares the value in the accumulator with a specified number of discrete locations (1–32). The instruction requires a starting location (Aaaa) and the number of bits (Kbbb) to be compared. The corresponding status flag will be turned on indicating the result of the comparison.

| | |
|------|-------|
| CMPF | A aaa |
| | K bbb |

| | |
|-----|------|
| DS | Used |
| HPP | Used |

| | Operand Data Type | DL250-1 Range | | DL260 Range | |
|----------------|-------------------|---------------|------|-------------|------|
| | A | aaa | bbb | aaa | bbb |
| Inputs | X | 0–777 | – | 0–1777 | – |
| Outputs | Y | 0–777 | – | 0–1777 | – |
| Control Relays | C | 0–1777 | – | 0–3777 | – |
| Stage bits | S | 0–1777 | – | 0–1777 | – |
| Timer bits | T | 0–377 | – | 0–377 | – |
| Counter bits | CT | 0–177 | – | 0–377 | – |
| Special Relays | SP | 0–777 | – | 0–777 | – |
| Global I/O | GX/GY | – | – | 0–3777 | – |
| Constant | K | – | 1–32 | – | 1–32 |

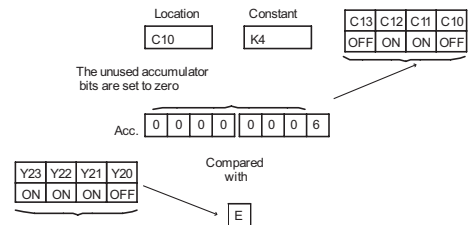
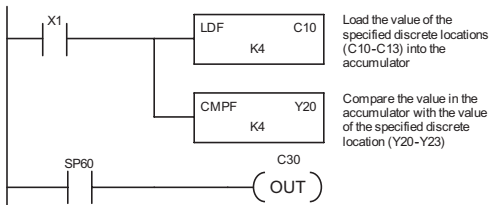
| Discrete Bit Flags | Description |
|--------------------|--|
| SP60 | On when the value in the accumulator is less than the first level value in the Accumulator Stack. |
| SP61 | On when the value in the accumulator is equal to the first level value in the Accumulator Stack. |
| SP62 | On when the value in the accumulator is greater than the first level value in the Accumulator Stack. |



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

In the following example, when X1 is on the Load Formatted instruction loads the binary value (6) from C10–C13 into the accumulator. The CMPF instruction compares the value in the accumulator to the value in Y20–Y23 (E hex). The corresponding discrete status flag will be turned on indicating the result of the comparison. In this example, if the value in the accumulator is less than the value specified in the Compare instruction, SP60 will turn on, energizing C30.

DirectSOFT



Compare with Stack (CMPS)

- 230
- 240
- 250-1
- 260

| | |
|-----|------|
| DS | Used |
| HPP | Used |

The Compare with Stack instruction is a 32-bit instruction that compares the value in the accumulator with the value in the first level of the accumulator stack.

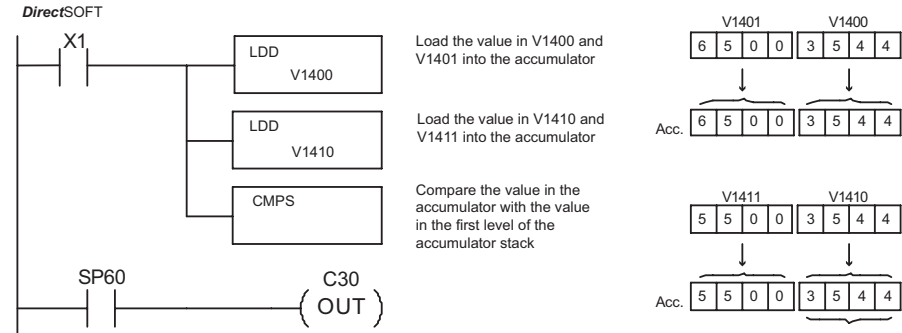
The corresponding status flag will be turned on indicating the result of the comparison. This does not affect the value in the accumulator.

CMPS

| Discrete Bit Flags | Description |
|--------------------|---|
| SP60 | On when the value in the Accumulator is less than the first level value in the Accumulator Stack |
| SP61 | On when the value in the Accumulator is equal to the first level value in the Accumulator Stack |
| SP62 | On when the value in the Accumulator is greater than the first level value in the Accumulator Stack |

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

In the following example when X1 is on, the value in V1400 and V1401 is loaded into the accumulator using the Load Double instruction. The value in V1410 and V1411 is loaded into the accumulator using the Load Double instruction. The value that was loaded into the accumulator from V1400 and V1401 is placed on top of the stack when the second Load instruction is executed. The value in the accumulator is compared with the value in the first level of the accumulator stack using the CMPS instruction. The corresponding discrete status flag will be turned on indicating the result of the comparison. In this example, if the value in the accumulator is less than the value in the stack, SP60 will turn on, energizing C30.



Handheld Programmer Keystrokes

| | | | |
|--------|---------|------|---------|
| \$ STR | → | B 1 | ENT |
| SHFT | L ANDST | D 3 | D 3 |
| SHFT | L ANDST | D 3 | D 3 |
| SHFT | C 2 | SHFT | M ORST |
| \$ STR | → | SHFT | SP STRN |
| GX OUT | → | SHFT | C 2 |

Compared with Top of Stack

Compare Real Number (CMPR)

- 230
- 240
- 250-1
- 260

The Compare Real Number instruction compares a real number value in the accumulator with two consecutive V-memory locations containing a real number. The corresponding status flag will be turned on indicating the result of the comparison. Both numbers being compared are 32 bits long.

CMPR
A aaa

| | |
|-----|------|
| DS | Used |
| HPP | N/A |

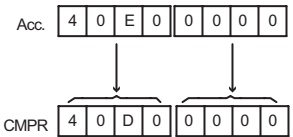
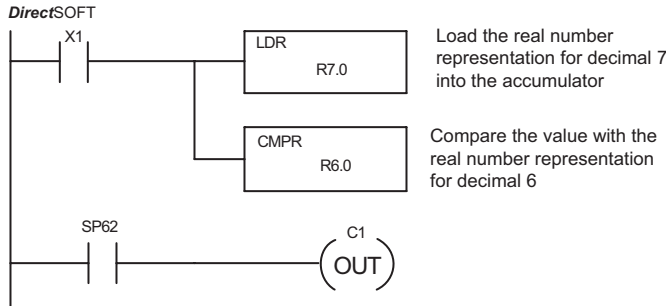
| Operand Data Type | | DL250-1 Range | DL260 Range |
|-------------------|---|----------------------------------|----------------------------------|
| | A | aaa | aaa |
| V-memory | V | All. See memory map | All. See memory map |
| Pointer | P | All V-memory. See memory map | All V-memory. See memory map |
| Constant | R | -3.402823E+038 to +3.402823E+038 | -3.402823E+038 to +3.402823E+038 |

| Discrete Bit Flags | Description |
|--------------------|--|
| SP60 | On when the value in the accumulator is less than the instruction value. |
| SP61 | On when the value in the accumulator is equal to the instruction value. |
| SP62 | On when the value in the accumulator is greater than the instruction value. |
| SP71 | On anytime the V-memory specified by a pointer (P) is not valid |
| SP75 | On when a real number instruction is executed and a non-real number encountered. |



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

In the following example, when X1 is on, the LDR instruction loads the real number representation for 7 decimal into the accumulator. The CMPR instruction compares the accumulator contents with the real representation for decimal 6. Since 7 > 6, the corresponding discrete status flag is turned on (special relay SP62).

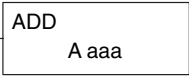


Math Instructions

Add (ADD)

- 230
- 240
- 250-1
- 260

Add is a 16-bit instruction that adds a BCD value in the accumulator with a BCD value in a V-memory location (Aaaa). (You cannot use a constant (K) as the BCD value in the box.) The result resides in the accumulator.



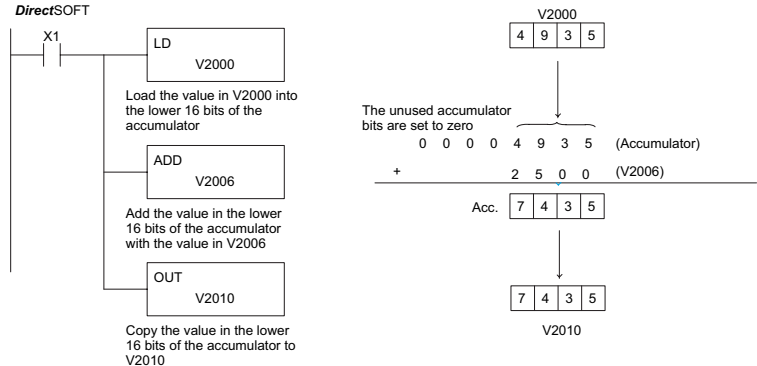
| | |
|-----|------|
| DS | Used |
| HPP | Used |

| Operand Data Type | | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|---|-----------------------|---------------------------------|---------------------------------|---------------------------------|
| | A | aaa | aaa | aaa | aaa |
| V-memory | V | All See memory map | All See memory map | All See memory map | All See memory map |
| Pointer | P | - | All V-memory. See memory map | All V-memory. See memory map | All V-memory. See memory map |

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

- ✓ 230 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.
- ✓ 240
- ✓ 250-1
- ✓ 260

ADDD
A aaa

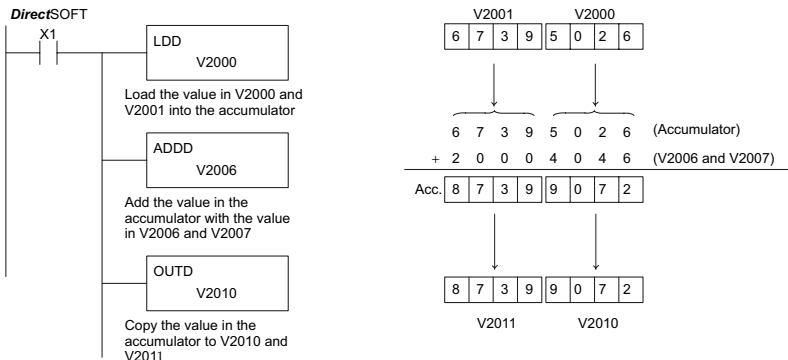
| DS | Used | Operand Data Type | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-----|------|-------------------|-------------|-----------------------|---------------------------------|---------------------------------|
| HPP | Used | A | aaa | aaa | aaa | aaa |
| | | V-memory | V | All See memory map | All See memory map | All See memory map |
| | | Pointer | P | - | All V-memory. See memory map | All V-memory. See memory map |
| | | Constant | K | 0-99999999 | 0-99999999 | 0-99999999 |

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

Add Real (ADDR)

- 230
- 240
- 250-1
- 260

Add Real is a 32-bit instruction that adds a real number, which is either two consecutive V-memory locations or a 32-bit constant, to a real number in the accumulator. Both numbers must conform to the IEEE floating point format. The result is a 32-bit real number that resides in the accumulator.

ADDR
A aaa

| | |
|-----|------|
| DS | Used |
| HPP | N/A |

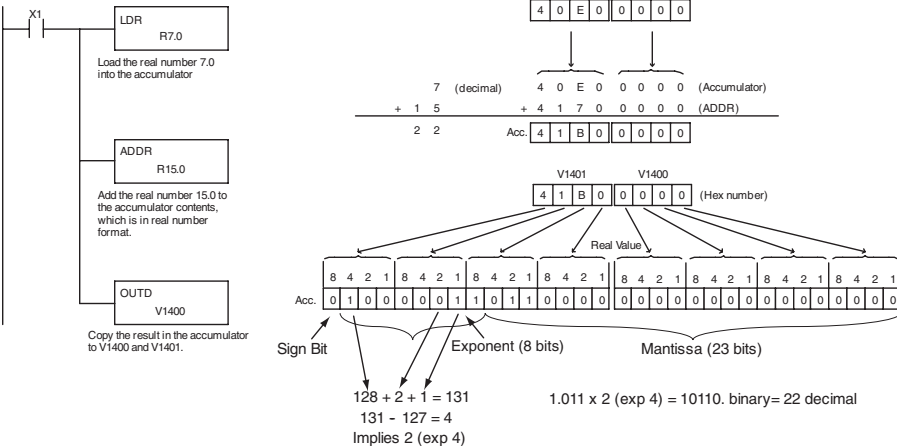
| Operand Data Type | | DL250-1 Range | DL260 Range |
|-------------------|---|-----------------------------------|-----------------------------------|
| A | | aaa | aaa |
| V-memory | V | All. See memory map | All. See memory map |
| Pointer | P | All V-memory. See memory map | All V-memory. See memory map |
| Constant | R | -3.402823E+038 to + 3.402823E+038 | -3.402823E+038 to + 3.402823E+038 |

Discrete Bit Flags Description

| | |
|------|--|
| SP63 | On when the result of the instruction causes the value in the accumulator to be zero |
| SP70 | On anytime the value in the accumulator is negative |
| SP71 | On anytime the V-memory specified by a pointer (P) is not valid |
| SP72 | On anytime the value in the accumulator is an invalid floating point number |
| SP73 | On when a signed addition or subtraction results in a incorrect sign bit |
| SP74 | On anytime a floating point math operation results in an underflow error |
| SP75 | On when a real number instruction is executed and a non-real number was encountered |

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

DirectSOFT

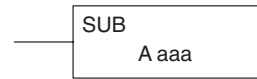


NOTE₁: The current HPP does not support real number entry with automatic conversion to the 32-bit IEEE format. You must use **DirectSOFT** for this feature.

NOTE₂: If the value being added to a real number is 16,777,216 times smaller than the real number, the calculation will not work.

Subtract (SUB)

- ✓ 230 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.
- ✓ 240
- ✓ 250-1
- ✓ 260



| | |
|-----|------|
| DS | Used |
| HPP | Used |

| Operand Data Type | | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|---|-----------------------|---------------------------------|---------------------------------|---------------------------------|
| | A | aaa | aaa | aaa | aaa |
| V-memory | V | All See memory map | All See memory map | All See memory map | All See memory map |
| Pointer | P | - | All V-memory. See memory map | All V-memory. See memory map | All V-memory. See memory map |

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

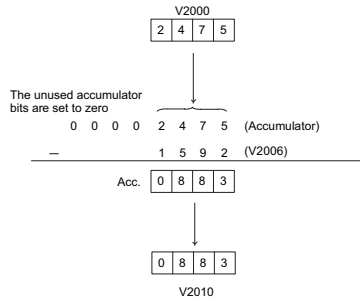
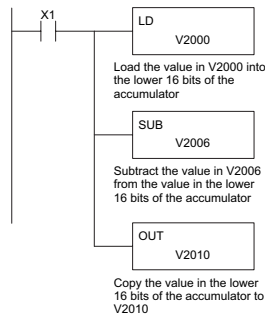


NOTE: A constant (K) cannot be used for the BCD value.

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.

Direct SOFT



Handheld Programmer Keystrokes

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

- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260

SUBD
A aaa

| Operand Data Type | | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|---|-----------------------|---------------------------------|---------------------------------|---------------------------------|
| | A | aaa | aaa | aaa | aaa |
| V-memory | V | All (See page 3 - 53) | All (See page 3-54) | All (See page 3-55) | All (See page 3-56) |
| Pointer | P | - | All V-memory (See page 3-54) | All V-memory (See page 3-55) | All V-memory (See page 3-56) |
| Constant | K | 0-99999999 | 0-99999999 | 0-99999999 | 0-99999999 |

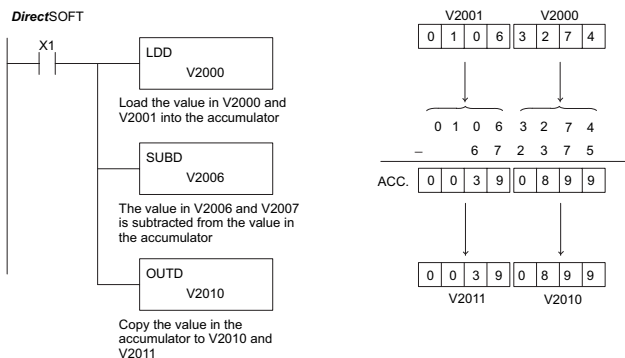
| Discrete Bit Flags | Description |
|--------------------|--|
| SP63 | On when the result of the instruction causes the value in the accumulator to be zero |
| SP64 | On when the 16 bit subtraction instruction results in a borrow |
| SP65 | On when the 32 bit subtraction instruction results in a borrow |
| SP70 | On anytime the value in the accumulator is negative |
| SP75 | On when a BCD instruction is executed and a NON-BCD number was encountered |



NOTE: 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.

| | |
|-----|------|
| DS | Used |
| HPP | Used |



Handheld Programmer Keystrokes

| | | | | | | | | | | | | | |
|--------|---------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|-----|--|--|
| \$ STR | → | B ₁ | ENT | | | | | | | | | | |
| SHFT | L ANDST | D ₃ | D ₃ | → | C ₂ | A ₀ | A ₀ | A ₀ | ENT | | | | |
| SHFT | S RST | SHFT | U ISG | B ₁ | D ₃ | → | C ₂ | A ₀ | A ₀ | G ₆ | ENT | | |
| GX_OUT | SHFT | D ₃ | → | C ₂ | A ₀ | B ₁ | A ₀ | ENT | | | | | |

Subtract Real (SUBR)

-  230
-  240
-  250-1
-  260

The Subtract Real is a 32-bit instruction that subtracts a real number, which is either two consecutive V-memory locations or a 32-bit constant, from a real number in the accumulator. The result is a 32-bit real number that resides in the accumulator. Both numbers must be Real data type (IEEE floating point format).

SUBR
A aaa

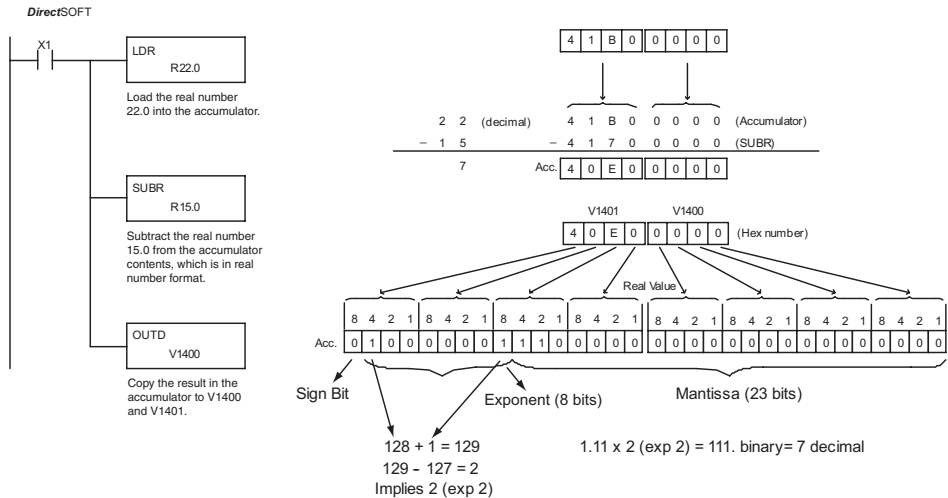
| | |
|-----|------|
| DS | Used |
| HPP | N/A |

| Operand Data Type | | DL250-1 Range | DL260 Range |
|-------------------|----------|----------------------------------|----------------------------------|
| | A | aaa | aaa |
| V-memory | V | All. (See page 3-55) | All. (See page 3-56) |
| Pointer | P | All V-memory (See page 3-55) | All V-memory (See page 3-56) |
| Constant | R | -3.402823E+038 to +3.402823E+038 | -3.402823E+038 to +3.402823E+038 |

| Discrete Bit Flags | Description |
|--------------------|--|
| SP63 | On when the result of the instruction causes the value in the accumulator to be zero |
| SP70 | On anytime the value in the accumulator is negative |
| SP71 | On anytime the V-memory specified by a pointer (P) is not valid |
| SP72 | On anytime the value in the accumulator is an invalid floating point number |
| SP73 | On when a signed addition or subtraction results in an incorrect sign bit |
| SP74 | On anytime a floating point math operation results in an underflow error |
| SP75 | On when a real number instruction is executed and a non-real number was encountered |



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



NOTE: The current HPP does not support real number entry with automatic conversion to the 32-bit IEEE format. You must use **DirectSOFT** for this feature.

Multiply (MUL)

- ✓

230
- ✓

240
- ✓

250-1
- ✓

260
- 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.

MUL

A aaa

| Operand | Data Type | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|----------|-----------|---------------------|---------------------------------|---------------------------------|---------------------------------|
| | A | aaa | aaa | aaa | aaa |
| V-memory | V | All (See page 3-53) | All (See page 3-54) | All (See page 3-55) | All (See page 3-56) |
| Pointer | P | - | All V-memory (See page 3-54) | All V-memory (See page 3-55) | All V-memory (See page 3-56) |
| Constant | K | 0-9999 | 0-9999 | 0-9999 | 0-9999 |

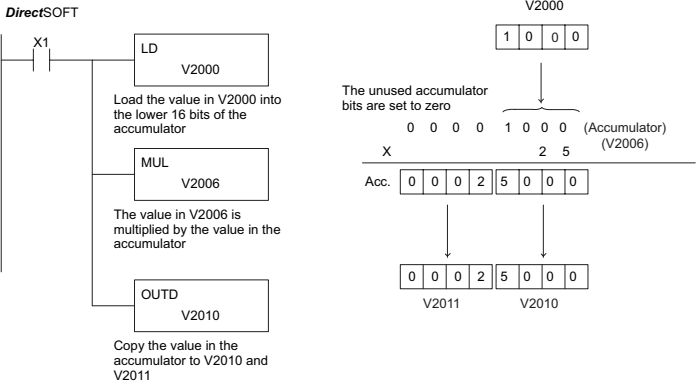
| Discrete Bit Flags | Description |
|--------------------|---|
| SP63 | On when the result of the instruction causes the value in the accumulator to be zero. |
| SP70 | On anytime the value in the accumulator is negative. |
| SP75 | On when a BCD instruction is executed and a NON-BCD number was encountered.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



NOTE: 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.

| | |
|-----|------|
| DS | Used |
| HPP | Used |



Handheld Programmer Keystrokes

| | | | | | | | | | |
|--------|---------|-------|---------|-----|-----|-----|-----|-----|-----|
| \$ STR | → | B 1 | ENT | | | | | | |
| SHFT | L ANDST | D 3 | → | C 2 | A 0 | A 0 | A 0 | ENT | |
| SHFT | M ORST | U ISG | L ANDST | → | C 2 | A 0 | A 0 | G 6 | ENT |
| GX OUT | SHFT | D 3 | → | C 2 | A 0 | B 1 | A 0 | ENT | |

Multiply Double (MULD)

- 230
- 240
- 250-1
- 260

Multiply Double is a 32-bit instruction that multiplies the 8-digit BCD value in the accumulator by the 8-digit BCD value in the two consecutive V-memory locations specified in the instruction. The lower 8 digits of the results reside in the accumulator. Upper digits of the result reside in the accumulator stack.

MULD

A aaa

| Operand Data Type | | DL250-1 Range | DL260 Range |
|-------------------|---|---------------------------|---------------------------|
| | A | aaa | aaa |
| V-memory | V | All V-mem (See page 3-55) | All V-mem (See page 3-56) |
| Pointer | P | All V-mem (See page 3-55) | All V-mem (See page 3-56) |

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



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

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

| | |
|-----|------|
| DS | Used |
| HPP | Used |

DirectSOFT

X1

LDD

Kbc614e

Load the hex equivalent of 12345678 decimal into the accumulator.

BCD

Convert the value to BCD format. It will occupy eight BCD digits (32 bits).

OUTD

V1400

Output the number to V1400 and V1401 using the OUTD instruction.

LD

K2

Load the constant K2 into the accumulator.

MULD

V1400

Multiply the accumulator contents (2) by the 8-digit number in V1400 and V1401.

OUTD

V1402

Move the result in the accumulator to V1402 and V1403 using the OUTD instruction.

1 2 3 4 5 6 7 8 (Accumulator)

V1401

V1400

1 2 3 4 5 6 7 8

X

2 (Accumulator)

Acc. 2 4 6 9 1 3 5 6

V1403

V1402

2 4 6 9 1 3 5 6

Handheld Programmer Keystrokes

S

STR

→

B

1

ENT

SHFT

L

ANDST

D

3

→

PREV

SHFT

B

1

C

2

SHFT

G

6

B

1

E

4

SHFT

E

4

ENT

SHFT

B

1

C

2

D

3

ENT

GX

OUT

SHFT

D

3

→

B

1

E

4

A

0

A

0

ENT

SHFT

L

ANDST

D

3

→

PREV

C

2

ENT

SHFT

M

ORST

U

ISG

L

ANDST

D

3

→

B

1

E

4

A

0

A

0

ENT

GX

OUT

SHFT

D

3

→

B

1

E

4

A

0

C

2

ENT

Multiply Real (MULR)

- ✗

230

✗

240

✓

250-1

✓

260
- The Multiply Real instruction multiplies a real number in the accumulator with either a real constant or a real number occupying two consecutive V-memory locations. The result resides in the accumulator. Both numbers must be Real data type (IEEE floating point format).

MULR

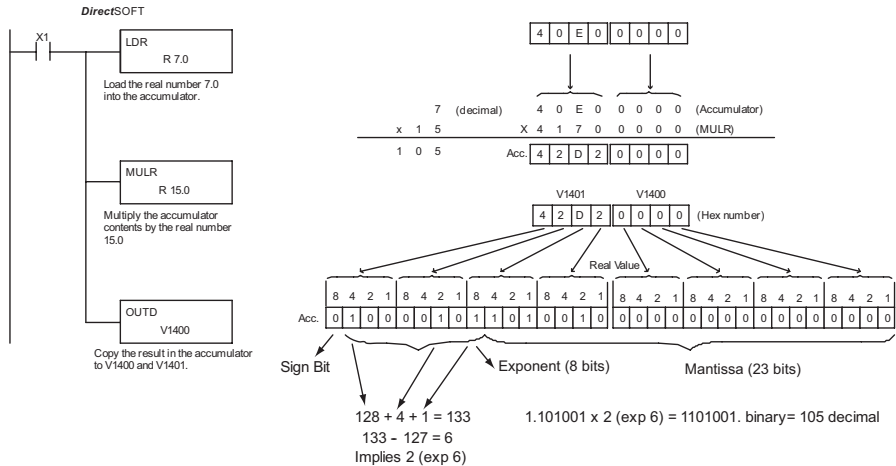
A aaa

| | |
|-----|------|
| DS | Used |
| HPP | N/A |

| Operand Data Type | | DL250-1 Range | DL260 Range |
|-------------------|---|-----------------------------------|-----------------------------------|
| | A | aaa | aaa |
| V-memory | V | All. (See page 3-55) | All. (See page 3-56) |
| Pointer | P | All V-memory (See page 3-55) | All V-memory (See page 3-56) |
| Constant | R | -3.402823E+038 to + 3.402823E+038 | -3.402823E+038 to + 3.402823E+038 |

| Discrete Bit Flags | Description |
|--------------------|--|
| SP63 | On when the result of the instruction causes the value in the accumulator to be zero |
| SP70 | On anytime the value in the accumulator is negative |
| SP71 | On anytime the V-memory specified by a pointer (P) is not valid |
| SP72 | On anytime the value in the accumulator is an invalid floating point number |
| SP73 | On when a signed addition or subtraction results in an incorrect sign bit |
| SP74 | On anytime a floating point math operation results in an underflow error |
| SP75 | On when a real number instruction is executed and a non-real number was encountered |

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



NOTE: The current HPP does not support real number entry with automatic conversion to the 32-bit IEEE format. You must use **DirectSOFT** for this feature.

Divide (DIV)

- ✓ 230 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.
- ✓ 240
- ✓ 250-1
- ✓ 260

DIV
A aaa

| Operand Data Type | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|-----------------------|------------------------------|------------------------------|------------------------------|
| A | aaa | aaa | aaa | aaa |
| V-memory | V All (See page 3-53) | All (See page 3-54) | All (See page 3-55) | All (See page 3-56) |
| Pointer | P - | All V-memory (See page 3-54) | All V-memory (See page 3-55) | All V-memory (See page 3-56) |
| Constant | K 1-9999 | 1-9999 | 1-9999 | 1-9999 |

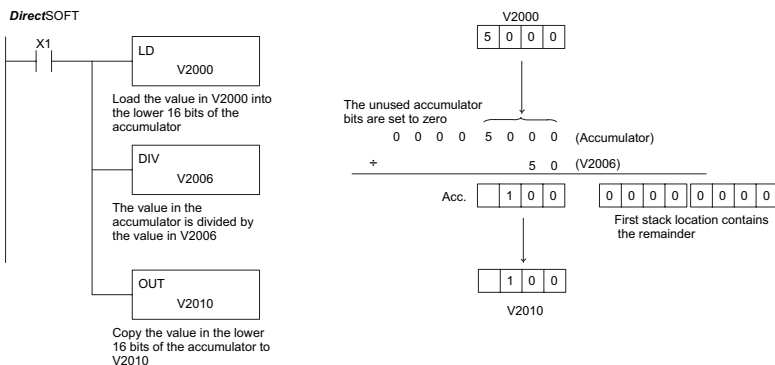
| Discrete Bit Flags | Description |
|--------------------|--|
| SP53 | On when the value of the operand is larger than the accumulator can work with |
| SP63 | On when the result of the instruction causes the value in the accumulator to be zero |
| SP70 | On anytime the value in the accumulator is negative |
| 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.

| | |
|-----|------|
| DS | Used |
| HPP | Used |



Handheld Programmer Keystrokes

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

Divide Double (DIVD)

- 230
- 240
- 250-1
- 260

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

DIVD
A aaa

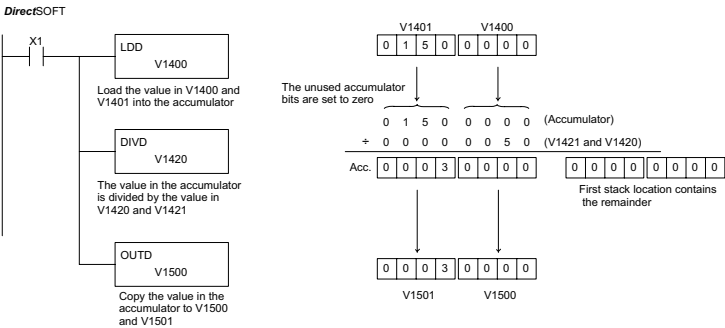
| | |
|-----|------|
| DS | Used |
| HPP | Used |

| Operand Data Type | | DL250-1 Range | DL260 Range |
|-------------------|---|------------------------------|------------------------------|
| A | | aaa | aaa |
| V-memory | V | All V-memory (See page 3-55) | All V-memory (See page 3-56) |
| Pointer | P | All V-memory (See page 3-55) | All V-memory (See page 3-56) |

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

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

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



Handheld Programmer Keystrokes

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

Divide Real (DIVR)

- 230
- 240
- 250-1
- 260

The Divide Real instruction divides a real number in the accumulator by either a real constant or a real number occupying two consecutive V-memory locations. The result resides in the accumulator. Both numbers must conform to the IEEE floating point format.

DIVR
A aaa

| | |
|-----|------|
| DS | Used |
| HPP | N/A |

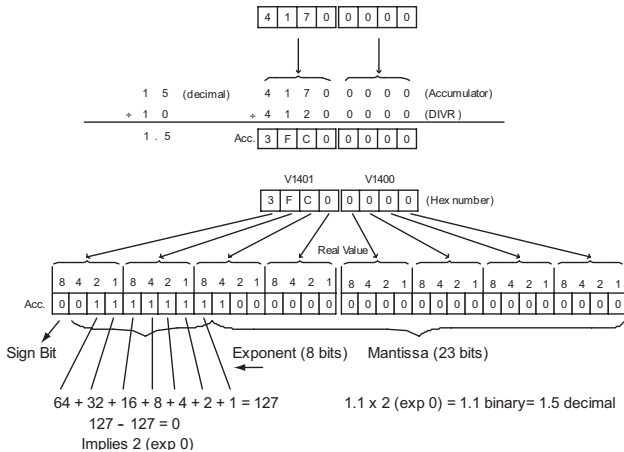
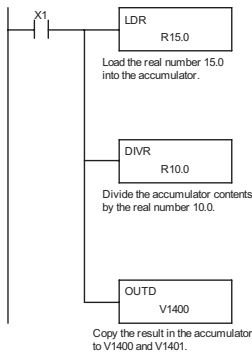
| Operand Data Type | | DL250-1 Range | DL260 Range |
|-------------------|---|--|--|
| A | | aaa | aaa |
| V-memory | V | All (See page 3-55) | All (See page 3-56) |
| Pointer | P | All V-mem (See page 3-55) | All V-mem (See page 3-56) |
| Constant | R | -3.402823E + 038 to + 3.402823E+038 | -3.402823E + 038 to + 3.402823E+038 |

| Discrete Bit Flags | Description |
|--------------------|---|
| SP63 | On when the result of the instruction causes the value in the accumulator to be zero. |
| SP70 | On anytime the value in the accumulator is negative. |
| SP71 | On anytime the V-memory specified by a pointer (P) is not valid. |
| SP72 | On anytime the value in the accumulator is a valid floating point number. |
| SP73 | On when a signed addition or subtraction results in a incorrect sign bit. |
| SP74 | On anytime a floating point math operation results in an underflow error. |
| SP75 | On when a real number instruction is executed and a non-real number was encountered. |

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



DirectSOFT

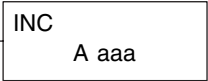


NOTE: The current HPP does not support real number entry with automatic conversion to the 32-bit IEEE format. You must use DirectSOFT for this feature.

Increment (INC)

- 230
- 240
- 250-1
- 260

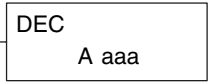
The Increment instruction increments a BCD value in a specified V-memory location by “1” each time the instruction is executed.



Decrement (DEC)

- 230
- 240
- 250-1
- 260

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



| Operand Data Type | | DL250-1 Range | DL260 Range |
|-------------------|---|---------------------------|---------------------------|
| | A | aaa | aaa |
| V-memory | V | All V mem (See page 3-55) | All V mem (See page 3-56) |
| Pointer | P | All V mem (See page 3-55) | All V mem (See page 3-56) |

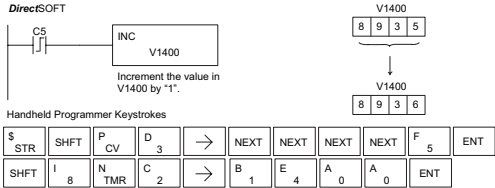
| | |
|-----|------|
| DS | Used |
| HPP | Used |

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

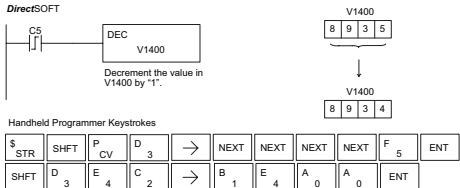


NOTE: Status flags are valid only until another instruction uses the same flag.

In the following increment example, the value in V1400 increases by one each time that C5 is closed (true).



In the following decrement example, the value in V1400 is decreased by one each time that C5 is closed (true).



NOTE: Use a pulsed contact closure to INC/DEC the value in V-memory once per closure.

Add Binary (ADDB)

- 230
- 240
- 250-1
- 260

| | |
|-----|------|
| DS | Used |
| HPP | Used |

The Add Binary instruction adds a 16-bit number (Aaaa) to the value stored in the accumulator. The number in the accumulator can be up to 32 bits long. The source of the 16-bit operand can be a constant or a data value located in V-memory. Add Binary performs the addition operation on the full binary representation of the operands, which distinguishes it from the Add instruction (see page 5-88), which treats the operands as BCD numbers. Although the addition operation is performed on the underlying binary values, the native display format is hexadecimal. For that reason you will need to load constants in hex.

ADDB
A aaa

The sum of the Add Binary operation occupies the full 32-bit accumulator and requires an Out Double to move the sum to V-memory. If the value in the accumulator occupies fewer than 32 bits, leading zeros are loaded in the left-most empty bit positions.

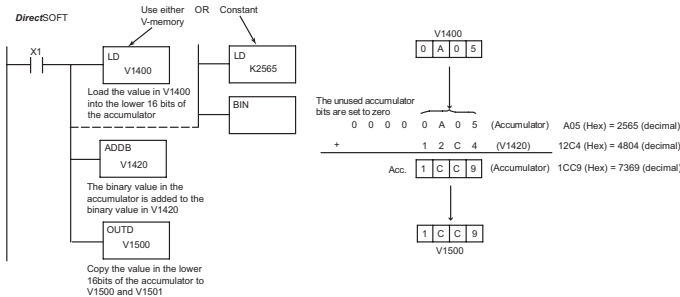
5

| Operand Data Type | | DL250-1 Range | DL260 Range |
|-------------------|---|---------------------------|---------------------------|
| A | | aaa | aaa |
| V-memory | V | All (See page 3-55) | All (See page 3-56) |
| Pointer | P | All V mem (See page 3-55) | All V mem (See page 3-56) |
| Constant. | K | 0-FFFF | 0-FFFF |

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

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

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



Handheld Programmer Keystrokes

| | | | |
|------|------|-----|-----|
| STR | → | 1 | ENT |
| SHFT | L | D | 1 |
| SHFT | A | D | D |
| OUT | SHFT | D | → |
| | | 1 | 5 |
| | | 0 | 0 |
| | | 0 | 0 |
| | | ENT | |

Add Binary Double (ADDBD)

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

ADDBD
A aaa

- 230
- 240
- 250-1
- 260

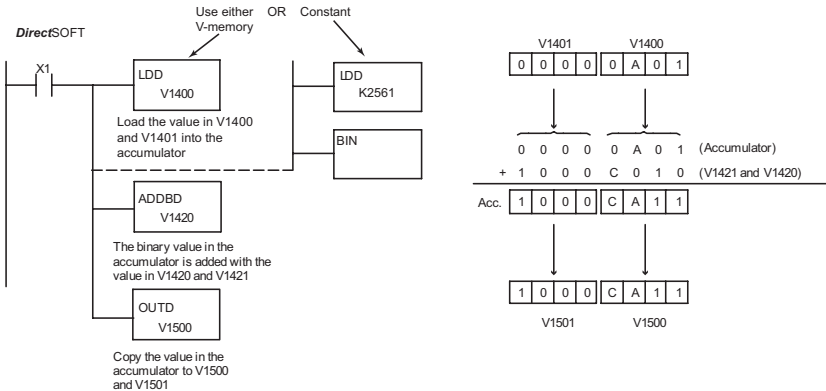
| | |
|-----|------|
| DS | Used |
| HPP | Used |

| Operand Data Type | | DL260 Range |
|-------------------|---|---------------------------|
| A | | aaa |
| V-memory | V | All (See page 3-56) |
| Pointer | P | All V mem (See page 3-56) |
| Constant | K | 0-FFFFFFF |

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

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

In the following example, when X1 is on, the value in V1400 and V1401 will be loaded into the accumulator using the Load Double instruction. The binary value in the accumulator is added with the binary value in V1420 and V1421 using the Add Binary Double instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.



Handheld Programmer Keystrokes

| | | |
|-----|------|---|
| STR | 1 | ← |
| LD | SHFT | D |
| ADD | SHFT | B |
| OUT | SHFT | D |

Subtract Binary (SUBB)

 230

 240

 250-1

 260

| | |
|-----|------|
| DS | Used |
| HPP | Used |

The Subtract Binary instruction subtracts a 16-bit number (Aaaa) from the value stored in the accumulator. The number in the accumulator can be up to 32 bits long. The source of the 16-bit operand can be a constant or a data value located in V-memory. Subtract Binary performs the subtraction operation on the full binary representation of the operands, which distinguishes it from the Subtract instruction (see page 5-91), which treats the operands as BCD numbers. Although the subtraction operation is performed on the underlying binary values, the native display format is hexadecimal. For that reason, you will need to load constants in hex.

The difference (result) of the Subtract Binary operation occupies the full 32 bits of the accumulator and requires an Out Double to move the value to V-memory. If the value in the accumulator occupies fewer than 32 bits, leading zeros are loaded in the left-most empty bit positions of the accumulator.

5

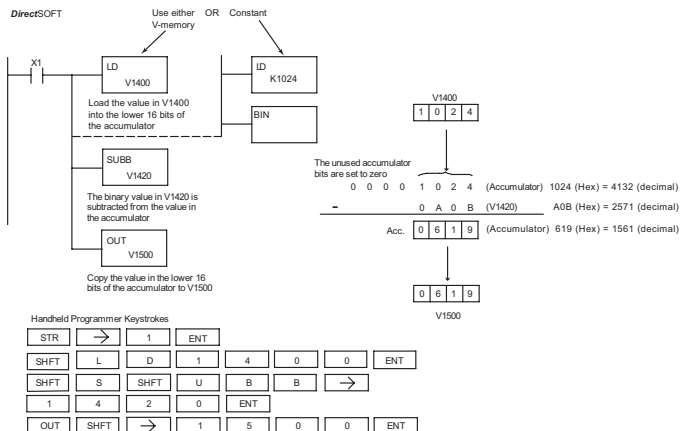
| Operand Data Type | | DL250-1 Range | DL260 Range |
|-------------------|----------|---------------------------|---------------------------|
| | A | aaa | aaa |
| V-memory | V | All (See page 3-55) | All (See page 3-56) |
| Pointer | P | All V-mem (See page 3-55) | All V-mem (See page 3-56) |
| Constant | K | 0-FFFF | 0-FFFF |

| Discrete Bit Flags | Description |
|--------------------|--|
| SP63 | On when the result of the instruction causes the value in the accumulator to be zero |
| SP64 | On when the 16-bit subtraction instruction results in a borrow |
| SP65 | On when the 32-bit subtraction instruction results in a borrow |
| SP70 | On anytime the value in the accumulator is negative |



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

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



Subtract Binary Double (SUBBD)

- 230
- 240
- 250-1
- 260

| | |
|-----|------|
| DS | Used |
| HPP | Used |

Subtract Binary Double is a 32-bit instruction that subtracts the binary value (Aaaa), which is either two consecutive V-memory locations or an 8-digit (max) binary constant, from the binary value in the accumulator. The result resides in the accumulator.

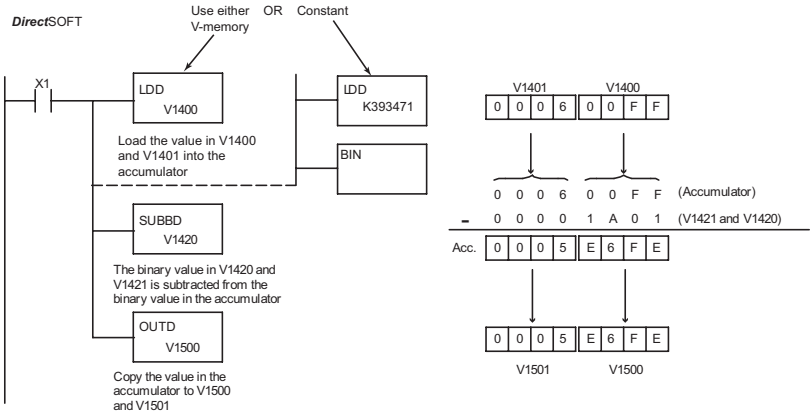
SUBBD
A aaa

| Operand Data Type | DL260 Range |
|-------------------|-----------------------------|
| | A |
| | aaa |
| V-memory | V All (See page 3-56) |
| Pointer | P All V mem (See page 3-56) |
| Constant | K 0-FFFFFFF |

| Discrete Bit Flags | Description |
|--------------------|--|
| SP63 | On when the result of the instruction causes the value in the accumulator to be zero |
| SP64 | On when the 16-bit subtraction instruction results in a borrow |
| SP65 | On when the 32-bit subtraction instruction results in a borrow |
| SP70 | On anytime the value in the accumulator is negative |

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

In the following example, when X1 is on, the value in V1400 and V1401 will be loaded into the accumulator using the Load Double instruction. The binary value in V1420 and V1421 is subtracted from the binary value in the accumulator using the Subtract Binary Double instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.



Handheld Programmer Keystrokes

| | | | |
|------|------|------|-----|
| STR | → | 1 | ENT |
| SHFT | L | D | D |
| SHFT | S | SHFT | U |
| 1 | 4 | 2 | 0 |
| OUT | SHFT | D | → |

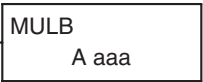
1 5 0 0 ENT

Multiply Binary (MULB)

- 230
- 240
- 250-1
- 260

| | |
|-----|------|
| DS | Used |
| HPP | Used |

The Multiply Binary instruction multiplies a 16-bit number A(aaa) by the value stored in the accumulator. The number in the accumulator can be up to 32 bits long. The source of the 16-bit operand can be a constant or a data value located in V-memory. Multiply Binary performs the multiplication operation on the full binary representation of the operands, which distinguishes it from the Multiply instruction (see page 5-94), which treats the operands as BCD numbers. Although the multiplication operation is performed on the underlying binary values, the native display format is hexadecimal. For that reason, you will need to load constants in hex.



The product of the Multiply Binary operation occupies the full 32-bit accumulator and requires an Out Double to move the product to V-memory. If the value in the accumulator occupies fewer than 32 bits, leading zeros are loaded in the left-most empty bit positions.

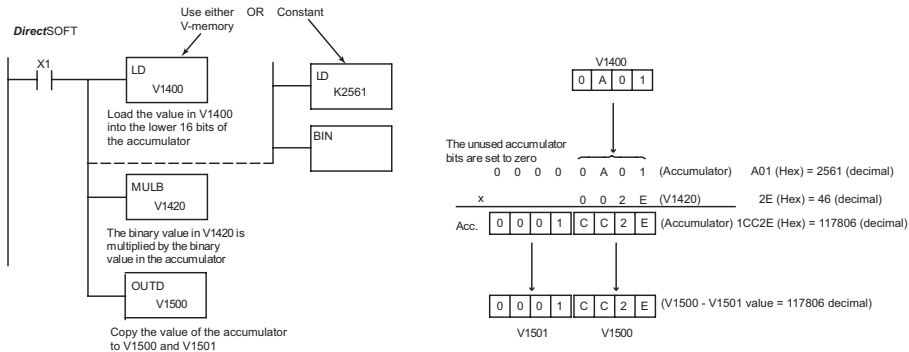
| Operand Data Type | | DL250-1 Range | DL260 Range |
|-------------------|---|---------------------------|---------------------------|
| | A | aaa | aaa |
| V-memory | V | All (See page 3-55) | All (See page 3-56) |
| Pointer | P | All V mem (See page 3-55) | All V mem (See page 3-56) |
| Constant | K | 0-FFFF | 0-FFFF |

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



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

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



Handheld Programmer Keystrokes

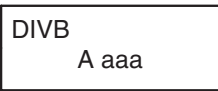
| | | | |
|------|------|---|-------------------|
| STR | → | 1 | ENT |
| SHFT | L | D | 1 4 0 0 ENT |
| SHFT | M | U | L B → 1 4 2 0 ENT |
| OUT | SHFT | D | → 1 5 0 0 ENT |

Divide Binary (DIVB)

- 230
- 240
- 250-1
- 260

| | |
|-----|------|
| DS | Used |
| HPP | Used |

The Divide Binary instruction divides a 16-bit number (Aaaa) into the value stored in the accumulator. The number in the accumulator can be up to 32 bits long. The source of the 16-bit divisor can be a constant or a data value located in V-memory. Divide Binary performs the division operation on the full binary representation of the operands, which distinguishes it from the Divide instruction (see page 5-97), which treats the operands as BCD numbers. Although the division operation is performed on the underlying binary values, the native display format is hexadecimal. For that reason you will need to load constants in hex.



At the completion of the division operation, the quotient resides in the accumulator and the remainder resides in the first stack location.

The quotient occupies the full 32-bit accumulator and requires an Out Double to move the quotient to V-memory. If the value in the accumulator occupies fewer than 32 bits, leading zeros are loaded in the left-most empty bit positions.

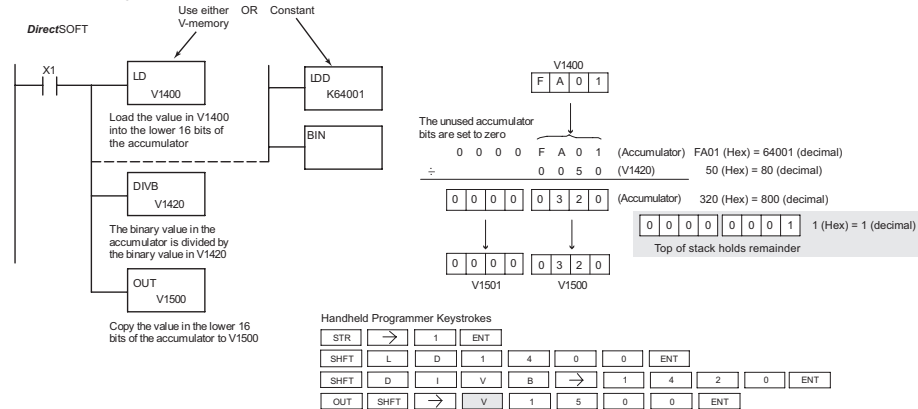
| Operand Data Type | | DL250-1 Range | DL260 Range |
|-------------------|---|---------------------------|---------------------------|
| | A | aaa | aaa |
| V-memory | V | All (See page 3-55) | All (See page 3-56) |
| Pointer | P | All V mem (See page 3-55) | All V mem (See page 3-56) |
| Constant | K | 0-FFFF | 0-FFFF |

| Discrete Bit Flags | Description |
|--------------------|--|
| SP53 | On when the value of the operand is larger than the accumulator can work with |
| SP63 | On when the result of the instruction causes the value in the accumulator to be zero |
| SP70 | On anytime the value in the accumulator is negative |



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

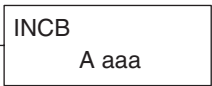
In the following example, when X1 is on, the value in V1400 will be loaded into the accumulator using the Load instruction. The binary value in the accumulator is divided by the binary value in V1420 using the Divide Binary instruction. The value in the accumulator is copied to V1500 using the Out Double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



Increment Binary (INCB)

- ✓ 230
- ✓ 240
- ✓ 250-1
- ✓ 260

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



| | |
|-----|------|
| DS | Used |
| HPP | Used |

| Operand Data Type | | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|---|-----------------------|------------------------------|------------------------------|------------------------------|
| | A | aaa | aaa | aaa | aaa |
| V-memory | V | All (See page 3 - 53) | All (See page 3-54) | All (See page 3-55) | All (See page 3-56) |
| Pointer | P | - | All V-memory (See page 3-54) | All V-memory (See page 3-55) | All V-memory (See page 3-56) |

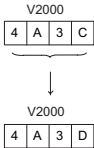
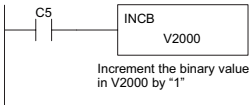
| Discrete Bit Flags | Description |
|--------------------|--|
| SP63 | On when the result of the instruction causes the value in the accumulator to be zero |



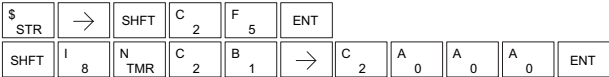
NOTE: The status flags are only valid until another instruction that uses the same flags is executed.

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

DirectSOFT



Handheld Programmer Keystrokes



Decrement Binary (DECB)

- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260
- The Decrement Binary instruction decrements a binary value in a specified V-memory location by “1” each time the instruction is executed.

DECB


A aaa

| | |
|-----|------|
| DS | Used |
| HPP | Used |

5

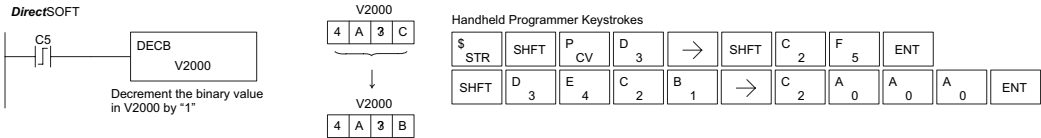
| Operand Data Type | | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|---|-----------------------|-----------------------------------|-----------------------------------|-----------------------------------|
| | A | aaa | aaa | aaa | aaa |
| V-memory | V | All (See page 3 - 53) | All (See page 3 - 54) | All (See page 3 - 55) | All (See page 3 - 56) |
| Pointer | P | - | All V-memory (See page 3 - 54) | All V-memory (See page 3 - 55) | All V-memory (See page 3 - 56) |

| 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 flag is executed.

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



Add Formatted (ADDF)

- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260

Add Formatted is a 32-bit instruction that adds the BCD value in the accumulator with the BCD value (Aaaa), which is a range of discrete bits. The specified range (Kbbb) can be 1 to 32 consecutive bits. The result resides in the accumulator.

ADDF A aaa
 K bbb

| | |
|-----|------|
| DS | Used |
| HPP | Used |

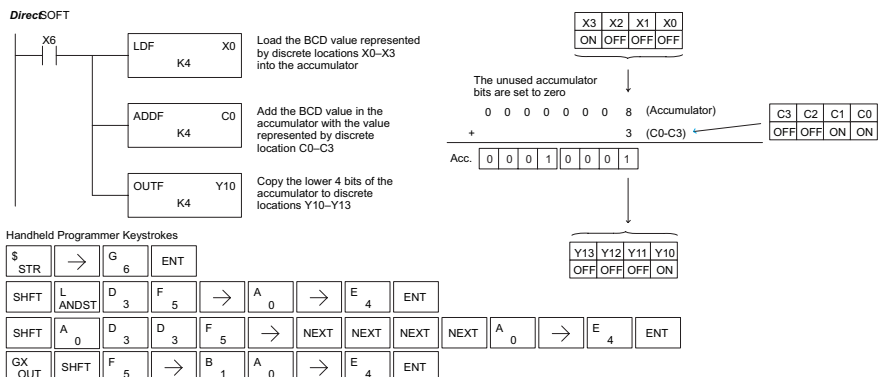
| Operand Data Type | DL260 Range | |
|-------------------|-------------|--------|
| | A | |
| Inputs | X | 0-1777 |
| Outputs | Y | 0-1777 |
| Control Relays | C | 0-3777 |
| Stage Bits | S | 0-1777 |
| Timer Bits | T | 0-377 |
| Counter Bits | CT | 0-377 |
| Special Relays | SP | 0-777 |
| Global I/O | GX/GY | 0-3777 |
| Constant | K | - |
| | | 1-32 |

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





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


In the following example, when X6 is on, the value formed by discrete locations X0–X3 is loaded into the accumulator using the Load Formatted instruction. The value formed by discrete locations C0–C3 is added to the value in the accumulator using the Add Formatted instruction. The value in the lower four bits of the accumulator is copied to Y10–Y13 using the Out Formatted instruction.




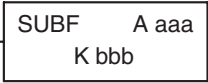
Subtract Formatted (SUBF)

 230 Subtract Formatted is a 32-bit instruction that subtracts the BCD value (Aaaa), which is a range of discrete bits, from the BCD value in the accumulator. The specified range (Kbbb) can be 1 to 32 consecutive bits. The result resides in the accumulator.

 240

 250-1

 260



| | |
|-----|------|
| DS | Used |
| HPP | Used |

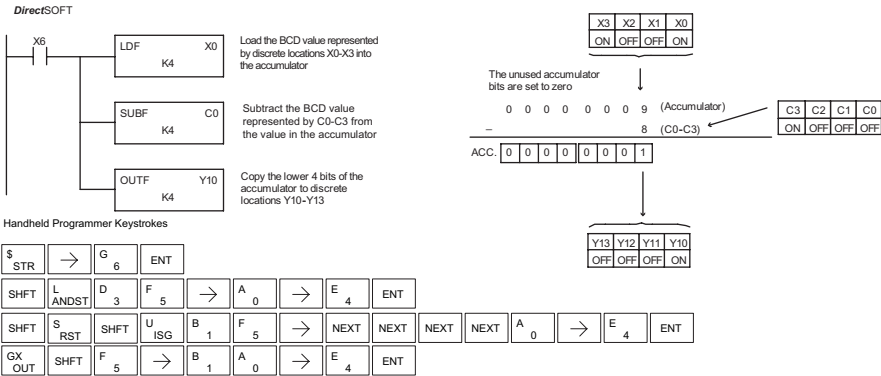
| Operand Data Type | | DL260 Range | |
|-------------------|-------|-------------|------|
| | A | aaa | bbb |
| Inputs | X | 0-1777 | - |
| Outputs | Y | 0-1777 | - |
| Control Relays | C | 0-3777 | - |
| Stage Bits | S | 0-1777 | - |
| Timer Bits | T | 0-377 | - |
| Counter Bits | CT | 0-377 | - |
| Special Relays | SP | 0-777 | - |
| Global I/O | GX/GY | 0-3777 | - |
| Constant | K | - | 1-32 |

| Discrete Bit Flags | Description |
|--------------------|--|
| SP63 | On when the result of the instruction causes the value in the accumulator to be zero |
| SP64 | On when the 16-bit subtraction instruction results in a borrow |
| SP65 | On when the 32-bit subtraction instruction results in a borrow |
| SP70 | On anytime the value in the accumulator is negative |
| SP75 | On when a BCD instruction is executed and a NON-BCD number was encountered |



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

In the following example, when X6 is on, the value formed by discrete locations X0–X3 is loaded into the accumulator using the Load Formatted instruction. The value formed by discrete location C0–C3 is subtracted from the value in the accumulator using the Subtract Formatted instruction. The value in the lower four bits of the accumulator is copied to Y10–Y13 using the Out Formatted instruction.



Multiply Formatted (MULF)

- 230
- 240
- 250-1
- 260

Multiply Formatted is a 16-bit instruction that multiplies the BCD value in the accumulator by the BCD value (Aaaa) which is a range of discrete bits. The specified range (Kbbb) can be 1 to 16 consecutive bits. The result resides in the accumulator.



| | |
|-----|------|
| DS | Used |
| HPP | Used |

| Operand Data Type | | DL260 Range | |
|-------------------|-------|-------------|------|
| | A | aaa | bbb |
| Inputs | X | 0-1777 | - |
| Outputs | Y | 0-1777 | - |
| Control Relays | C | 0-3777 | - |
| Stage Bits | S | 0-1777 | - |
| Timer Bits | T | 0-377 | - |
| Counter Bits | CT | 0-377 | - |
| Special Relays | SP | 0-777 | - |
| Global I/O | GX/GY | 0-3777 | - |
| Constant | K | - | 1-16 |

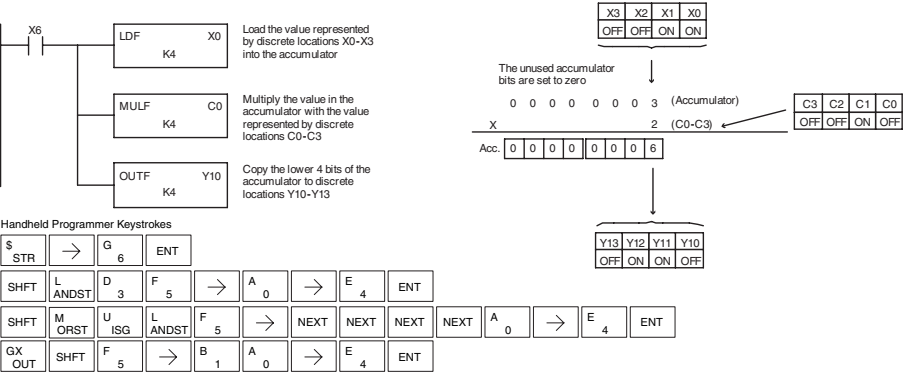
| Discrete Bit Flags | Description |
|--------------------|--|
| SP63 | On when the result of the instruction causes the value in the accumulator to be zero |
| SP70 | On anytime the value in the accumulator is negative |
| SP75 | On when a BCD instruction is executed and a NON-BCD number was encountered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



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

In the following example, when X6 is on, the value formed by discrete locations X0–X3 is loaded into the accumulator using the Load Formatted instruction. The value formed by discrete locations C0–C3 is multiplied by the value in the accumulator using the Multiply Formatted instruction. The value in the lower four bits of the accumulator is copied to Y10–Y13 using the Out Formatted instruction.

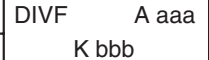
DirectSOFT



Divide Formatted (DIVF)

- 230
- 240
- 250-1
- 260

Divide Formatted is a 16-bit instruction that divides the BCD value in the accumulator by the BCD value (Aaaa), a range of discrete bits. The specified range (Kbbb) can be 1 to 16 consecutive bits. The first part of the quotient resides in the accumulator and the remainder resides in the first stack location.



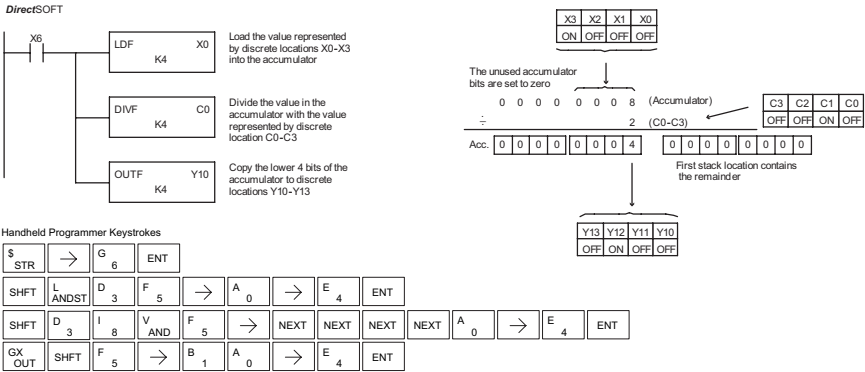
| | |
|-----|------|
| DS | Used |
| HPP | Used |

| Operand Data Type | | DL260 Range | |
|-------------------|-------|-------------|------|
| | A | aaa | bbb |
| Inputs | X | 0-1777 | - |
| Outputs | Y | 0-1777 | - |
| Control Relays | C | 0-3777 | - |
| Stage Bits | S | 0-1777 | - |
| Timer Bits | T | 0-377 | - |
| Counter Bits | CT | 0-377 | - |
| Special Relays | SP | 0-777 | - |
| Global I/O | GX/GY | 0-3777 | - |
| Constant | K | - | 1-16 |

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

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

In the following example, when X6 is on, the value formed by discrete locations X0–X3 is loaded into the accumulator using the Load Formatted instruction. The value in the accumulator is divided by the value formed by discrete location C0–C3 using the Divide Formatted instruction. The value in the lower four bits of the accumulator is copied to Y10–Y13 using the Out Formatted instruction.



Add Top of Stack (ADDS)

-  230
-  240
-  250-1
-  260

Add Top of Stack is a 32-bit instruction that adds the BCD value in the accumulator with the BCD value in the first level of the accumulator stack. The result resides in the accumulator. The value in the first level of the accumulator stack is removed and all stack values are moved up one level.

ADDS

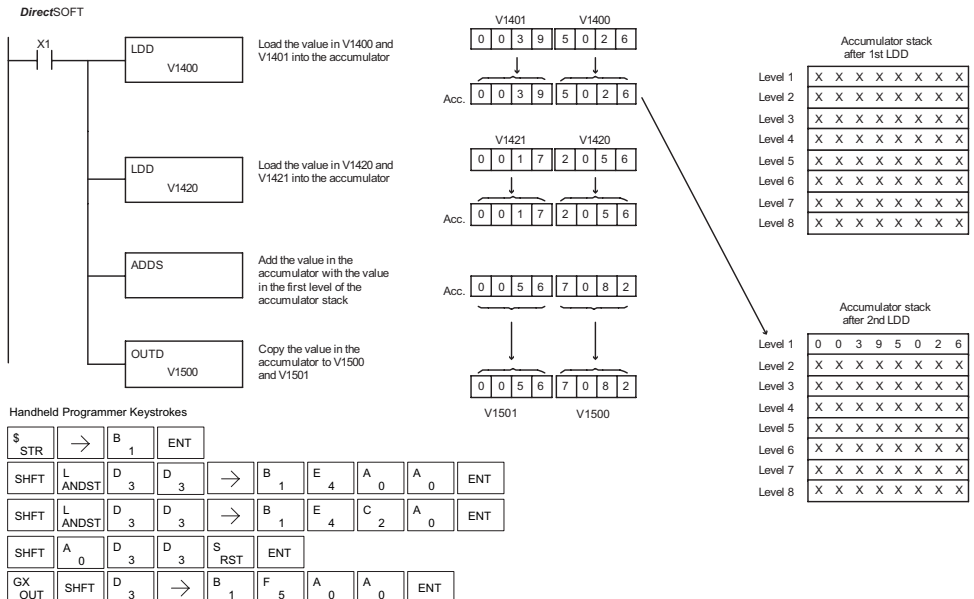
| Discrete Bit Flags | Description |
|--------------------|--|
| SP63 | On when the result of the instruction causes the value in the accumulator to be zero |
| SP66 | On when the 16-bit addition instruction results in a carr |
| SP67 | On when the 32-bit addition instruction results in a carry. |
|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



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

| | |
|-----|------|
| DS | Used |
| HPP | Used |

In the following example, when X1 is on, the value in V1400 and V1401 will be loaded into the accumulator using the Load Double instruction. The value in V1420 and V1421 is loaded into the accumulator using the Load Double instruction, pushing the value previously loaded in the accumulator onto the accumulator stack. The value in the first level of the accumulator stack is added with the value in the accumulator using the Add Stack instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.



Subtract Top of Stack (SUBS)

- 230
- 240
- 250-1
- 260

Subtract Top of Stack is a 32-bit instruction that subtracts the BCD value in the first level of the accumulator stack from the BCD value in the accumulator. The result resides in the accumulator. The value in the first level of the accumulator stack is removed and all stack values are moved up one level.

SUBS

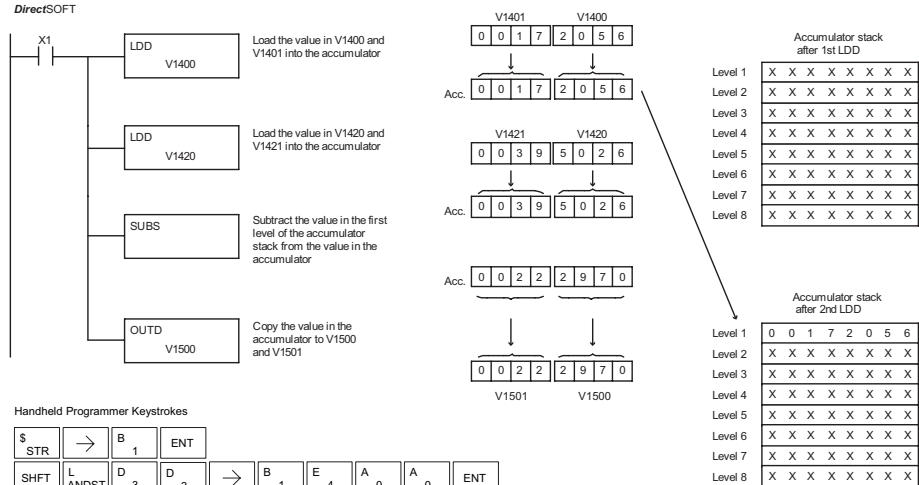
5

| Discrete Bit Flags | Description |
|--------------------|--|
| SP63 | On when the result of the instruction causes the value in the accumulator to be zero |
| SP64 | On when the 16-bit subtraction instruction results in a borrow |
| SP65 | On when the 32-bit subtraction instruction results in a borrow |
| SP70 | On anytime the value in the accumulator is negative |
| SP75 | On when a BCD instruction is executed and a NON-BCD number was encountered |

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

| | |
|-----|------|
| DS | Used |
| HPP | Used |

In the following example, when X1 is on, the value in V1400 and V1401 will be loaded into the accumulator using the Load Double instruction. The value in V1420 and V1421 is loaded into the accumulator using the Load Double instruction, pushing the value previously loaded into the accumulator onto the accumulator stack. The BCD value in the first level of the accumulator stack is subtracted from the BCD value in the accumulator using the Subtract Stack instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.



Handheld Programmer Keystrokes

| | | | |
|--------|---------|------------|-----------------------|
| \$ STR | → | B 1 | ENT |
| SHFT | L ANDST | D 3 | → B 1 E 4 A 0 A 0 ENT |
| SHFT | L ANDST | D 3 | → B 1 E 4 C 2 A 0 ENT |
| SHFT | S RST | SHFT U ISG | B 1 S RST ENT |
| GX OUT | SHFT | D 3 | → B 1 F 5 A 0 A 0 ENT |

Multiply Top of Stack (MULS)

-  230
-  240
-  250-1
-  260

Multiply Top of Stack is a 16-bit instruction that multiplies a 4-digit BCD value in the first level of the accumulator stack by a 4-digit BCD value in the accumulator. The result resides in the accumulator. The value in the first level of the accumulator stack is removed, and all stack values are moved up one level.

MULS

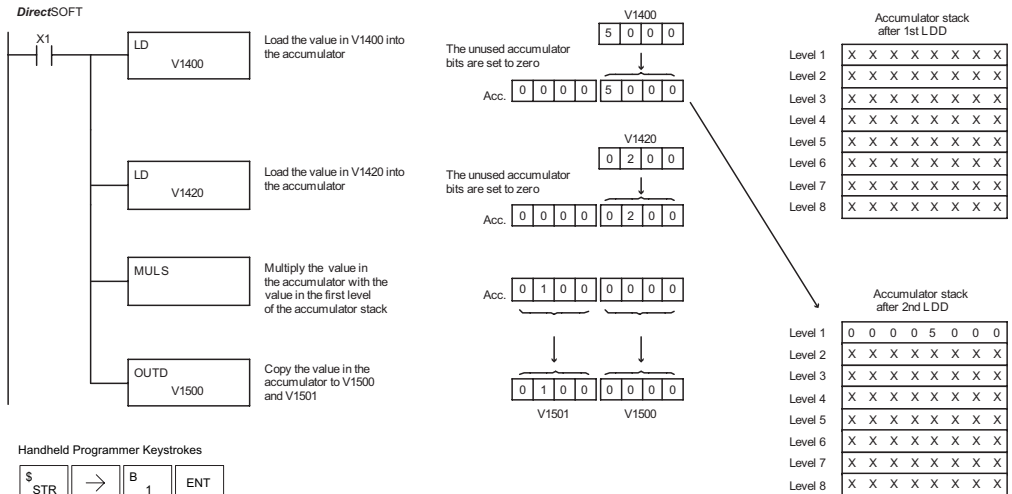
| Discrete Bit Flags | Description |
|--------------------|--|
| SP63 | On when the result of the instruction causes the value in the accumulator to be zero |
| SP70 | On anytime the value in the accumulator is negative |
| SP75 | On when a BCD instruction is executed and a NON-BCD number was encountered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



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

In the following example, when X1 is on, the value in V1400 will be loaded into the accumulator using the Load instruction. The value in V1420 is loaded into the accumulator using the Load instruction, pushing the value previously loaded in the accumulator onto the accumulator stack. The BCD value in the first level of the accumulator stack is multiplied by the BCD value in the accumulator using the Multiply Stack instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.

| | |
|-----|------|
| DS | Used |
| HPP | Used |



Handheld Programmer Keystrokes

| | | | | | | | | | | | | | | |
|------|-----|-------|---|-----|-----|-------|---|-----|-----|---|---|---|---|-----|
| \$ | STR | → | B | 1 | ENT | | | | | | | | | |
| SHFT | L | ANDST | D | 3 | → | B | 1 | E | 4 | A | 0 | A | 0 | ENT |
| SHFT | L | ANDST | D | 3 | → | B | 1 | E | 4 | C | 2 | A | 0 | ENT |
| SHFT | M | ORST | U | ISG | L | ANDST | S | RST | ENT | | | | | |
| GX | OUT | SHFT | D | 3 | → | B | 1 | F | 5 | A | 0 | A | 0 | ENT |

Divide by Top of Stack (DIVS)

- 230

240

250-1

260
- Divide Top of Stack is a 32-bit instruction that divides the 8-digit BCD value in the accumulator by a 4-digit BCD value in the first level of the accumulator stack. The result resides in the accumulator and the remainder resides in the first level of the accumulator stack.

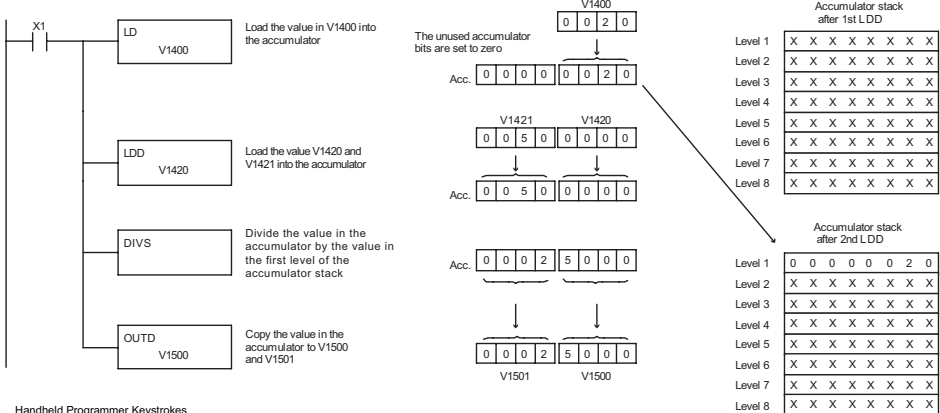
DIVS

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

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

In the following example, when X1 is on, the Load instruction loads the value in V1400 into the accumulator. The value in V1420 is loaded into the accumulator using the Load Double instruction, pushing the value previously loaded in the accumulator onto the accumulator stack. The BCD value in the accumulator is divided by the BCD value in the first level of the accumulator stack using the Divide Stack instruction. The Out Double instruction copies the value in the accumulator to V1500 and V1501.

DirectSOFT



Handheld Programmer Keystrokes

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

Add Binary Top of Stack (ADDBS)

- 230
- 240
- 250-1
- 260

Add Binary Top of Stack instruction is a 32-bit instruction that adds the binary value in the accumulator with the binary value in the first level of the accumulator stack. The result resides in the accumulator. The value in the first level of the accumulator stack is removed, and all stack values are moved up one level.

ADDBS

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

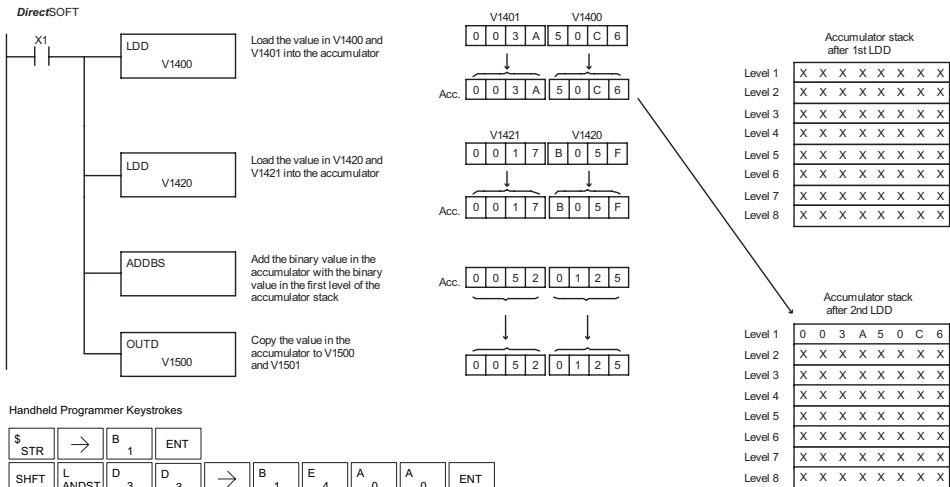


NOTE: Status flags are valid only until another instruction uses the same flag.

5

| | |
|-----|------|
| DS | Used |
| HPP | Used |

In the following example, when X1 is on, the value in V1400 and V1401 will be loaded into the accumulator using the Load Double instruction. The value in V1420 and V1421 is loaded into the accumulator using the Load Double instruction, pushing the value previously loaded in the accumulator onto the accumulator stack. The binary value in the first level of the accumulator stack is added with the binary value in the accumulator using the Add Stack instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.



Subtract Binary Top of Stack (SUBBS)

- ✗

230

✗

240

✗

250-1

✓

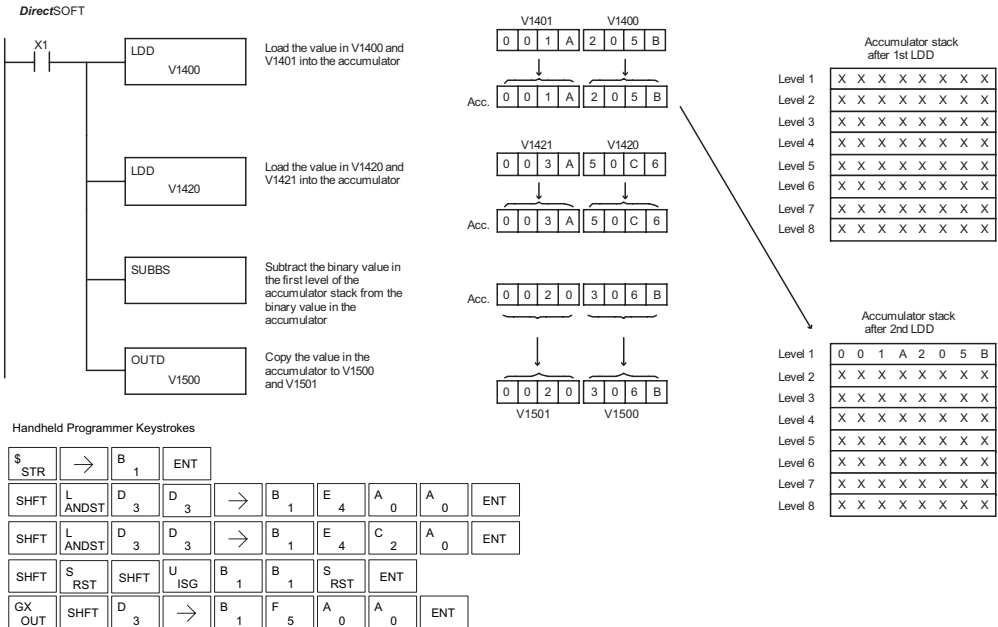
260
- Subtract Binary Top of Stack is a 32-bit instruction that subtracts the binary value in the first level of the accumulator stack from the binary value in the accumulator. The result resides in the accumulator. The value in the first level of the accumulator stack is removed, and all stack locations are moved up one level.

SUBBS

| Discrete Bit Flags | Description |
|--------------------|--|
| SP63 | On when the result of the instruction causes the value in the accumulator to be zero |
| SP64 | On when the 16-bit subtraction instruction results in a borrow |
| SP65 | On when the 32-bit subtraction instruction results in a borrow |
| SP70 | On anytime the value in the accumulator is negative |

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

In the following example, when X1 is on, the value in V1400 and V1401 will be loaded into the accumulator using the Load Double instruction. The value in V1420 and V1421 is loaded into the accumulator using the Load Double instruction, pushing the value previously loaded in the accumulator onto the accumulator stack. The binary value in the first level of the accumulator stack is subtracted from the binary value in the accumulator using the Subtract Stack instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.



Multiply Binary Top of Stack (MULBS)

-  230
-  240
-  250-1
-  260

Multiply Binary Top of Stack is a 16-bit instruction that multiplies the 16-bit binary value in the first level of the accumulator stack by the 16-bit binary value in the accumulator and can be 32 bits (8 digits maximum). The value in the first level of the accumulator stack is removed, and all stack locations are moved up one level.

MULBS

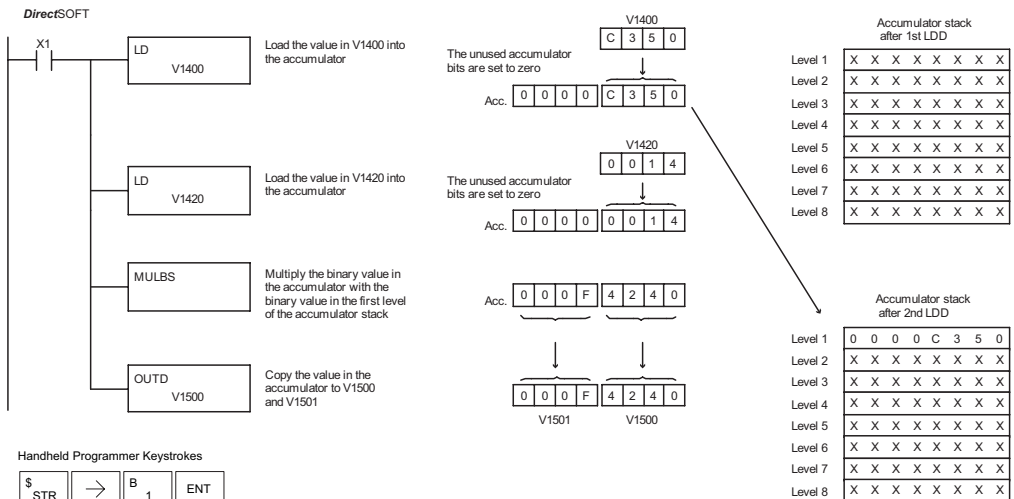
| Discrete Bit Flags | Description |
|--------------------|--|
| SP63 | On when the result of the instruction causes the value in the accumulator to be zero |
| SP70 | On anytime the value in the accumulator is negative |



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

| | |
|-----|------|
| DS | Used |
| HPP | Used |

In the following example, when X1 is on, the Load instruction moves the value in V1400 into the accumulator. The value in V1420 is loaded into the accumulator using the Load instruction, pushing the value previously loaded in the accumulator onto the stack. The binary value in the accumulator stack's first level is multiplied by the binary value in the accumulator using the Multiply Binary Stack instruction. The Out Double instruction copies the value in the accumulator to V1500 and V1501.



Handheld Programmer Keystrokes

| | | | | | | | | | | | | | | |
|------|-----|-------|---|-----|-----|-------|---|---|---|-----|-----|---|---|-----|
| \$ | STR | → | B | 1 | ENT | | | | | | | | | |
| SHFT | L | ANDST | D | 3 | → | B | 1 | E | 4 | A | 0 | A | 0 | ENT |
| SHFT | L | ANDST | D | 3 | → | B | 1 | E | 4 | C | 2 | A | 0 | ENT |
| SHFT | M | ORST | U | ISG | L | ANDST | B | 1 | S | RST | ENT | | | |
| GX | OUT | SHFT | D | 3 | → | B | 1 | F | 5 | A | 0 | A | 0 | ENT |

Divide Binary by Top of Stack (DIVBS)

-  230
-  240
-  250-1
-  260

Divide Binary Top of Stack is a 32-bit instruction that divides the 32-bit binary value in the accumulator by the 16-bit binary value in the first level of the accumulator stack. The result resides in the accumulator, and the remainder resides in the first level of the accumulator stack.

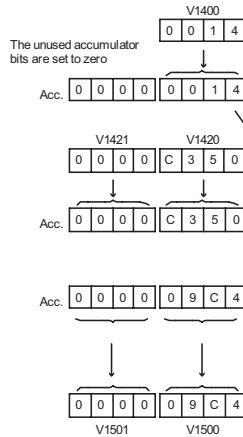
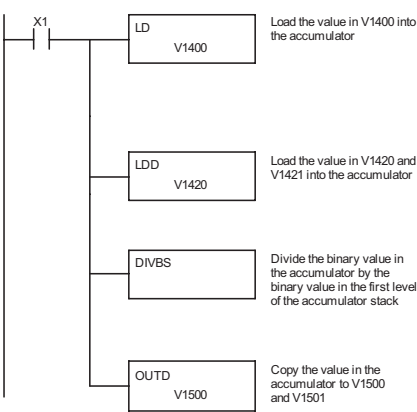
DIVBS

| Discrete Bit Flags | Description |
|--------------------|--|
| SP53 | On when the value of the operand is larger than the accumulator can work with |
| SP63 | On when the result of the instruction causes the value in the accumulator to be zero |
| SP70 | On anytime the value in the accumulator is negative |

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

In the following example, when X1 is on, the value in V1400 will be loaded into the accumulator using the Load instruction. The value in V1420 and V1421 is loaded into the accumulator using the Load Double instruction, pushing the value previously loaded in the accumulator onto the accumulator stack. The binary value in the accumulator is divided by the binary value in the first level of the accumulator stack using the Divide Binary Stack instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.

DirectSOFT



| Accumulator stack after 1st LDD | | | | | | | | | | | | | | | |
|---------------------------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Level 1 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| Level 2 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| Level 3 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| Level 4 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| Level 5 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| Level 6 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| Level 7 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| Level 8 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |

| Accumulator stack after 2nd LDD | | | | | | | | | | | | | | | |
|---------------------------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Level 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 4 | | | | | | | |
| Level 2 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| Level 3 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| Level 4 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| Level 5 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| Level 6 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| Level 7 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| Level 8 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |

Handheld Programmer Keystrokes

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

| The remainder resides in the first stack location | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Level 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 4 | | | | | | |
| Level 2 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| Level 3 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| Level 4 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| Level 5 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| Level 6 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| Level 7 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| Level 8 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |

Transcendental Functions (DL260 only)

- ☐ 230
- ☐ 240
- ☐ 250-1
- ☒ 260

| | |
|-----|------|
| DS | Used |
| HPP | N/A |

The DL260 CPU features special numerical functions to complement its real number capability. The transcendental functions include the trigonometric sine, cosine, and tangent, and also their inverses (arc sine, arc cosine, and arc tangent). The square root function is also grouped with these other functions.

The transcendental math instructions operate on a real number in the accumulator (it cannot be BCD or binary). The real number result resides in the accumulator. The square root function operates on the full range of positive real numbers. The sine, cosine and tangent functions require numbers expressed in radians. You can work with angles expressed in degrees by first converting them to radians with the Radian (RADR) instruction, then performing the trig function. All transcendental functions utilize the following flag bits.

| Discrete Bit Flags | Description |
|--------------------|--|
| SP63 | On when the result of the instruction causes the value in the accumulator to be zero |
| SP70 | On anytime the value in the accumulator is negative |
| SP72 | On anytime the value in the accumulator is a valid floating point number |
| SP73 | On when a signed addition or subtraction results in a incorrect sign bit |
| SP75 | On when a real number instruction is executed and a non-real number was encountered |

| Math Function | Range of Argument |
|---------------|---|
| SP53 | On when the value of the operand is larger than the accumulator can work with |

Sine Real (SINR)

The Sine Real instruction takes the sine of the real number stored in the accumulator. The result resides in the accumulator. Both the original number and the result are in IEEE 32-bit format.

SINR

Cosine Real (COSR)

The Cosine Real instruction takes the cosine of the real number stored in the accumulator. The result resides in the accumulator. Both the original number and the result are in IEEE 32-bit format.

COSR

Tangent Real (TANR)

The Tangent Real instruction takes the tangent of the real number stored in the accumulator. The result resides in the accumulator. Both the original number and the result are in IEEE 32-bit format.

TANR

Arc Sine Real (ASINR)

The Arc Sine Real instruction takes the inverse sine of the real number stored in the accumulator. The result resides in the accumulator. Both the original number and the result are in IEEE 32-bit format.

ASINR

Arc Cosine Real (ACOSR)

The Arc Cosine Real instruction takes the inverse cosine of the real number stored in the accumulator. The result resides in the accumulator. Both the original number and the result are in IEEE 32-bit format.

ACOSR

Arc Tangent Real (ATANR)

The Arc Tangent Real instruction takes the inverse tangent of the real number stored in the accumulator. The result resides in the accumulator. Both the original number and the result are in IEEE 32-bit format.

ATANR

Square Root Real (SQRTR)

The Square Root Real instruction takes the square root of the real number stored in the accumulator. The result resides in the accumulator. Both the original number and the result are in IEEE 32-bit format.

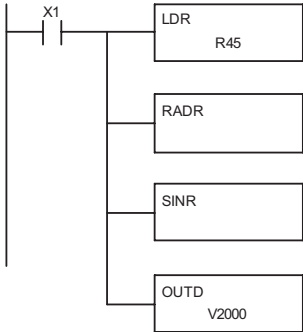
SQRTR

NOTE: The square root function can be useful in several situations. However, if you are trying to do the square-root extract function for an orifice flow meter measurement as the PV to a PID loop, note that the PID loop already has the square-root extract function built in.

| | |
|-----|------|
| DS | Used |
| HPP | N/A |

The following example takes the **sine** of 45 degrees. Since these transcendental functions operate only on real numbers, we do a LDR (Load Real) 45. The trig functions operate only in radians, so we must convert the degrees to radians by using the RADR command. After using the SINR (Sine Real) instruction, we use an OUTD (Out Double) instruction to move the result from the accumulator to V-memory. The result is 32-bits wide, requiring the Out Double to move it.

DirectSOFT



Load the real number 45 into the accumulator.

Convert the degrees into radians, leaving the result in the accumulator.

Take the sine of the number in the accumulator, which is in radians.

Copy the value in the accumulator to V2000 and V2001.

Accumulator contents (viewed as real number)

45.000000

0.7853981

0.7071067

0.7071067

NOTE: The current HPP does not support real number entry with automatic conversion to the 32-bit IEEE format. You must use **DirectSOFT** for entering real numbers, using the LDR (Load Real) instruction.

Shift Left (SHFL)

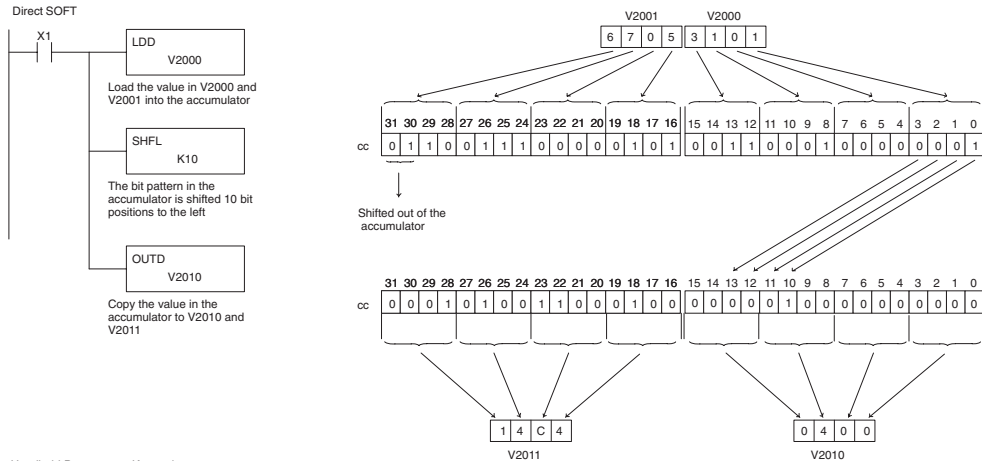
- 230 Shift Left is a 32-bit instruction that shifts the bits in the accumulator a specified number (Aaaa) of places to the left. The vacant positions are filled with zeros, and the bits shifted out of the accumulator are lost.
- 240
- 250-1
- 260

SHFL
A aaa

| Operand Data Type | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|-------------|---------------------|---------------------|---------------------|
| | A | aaa | aaa | aaa |
| V-memory | V | All (See page 3-53) | All (See page 3-54) | All (See page 3-56) |
| Constant | K | 1-32 | 1-32 | 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 10 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.

| | |
|-----|------|
| DS | Used |
| HPP | Used |



Handheld Programmer Keystrokes

| | | | |
|-------|--------|----------|----------------------|
| STR | → | 1 | ENT |
| SHFT | L NDST | D 3 | D 3 → C 2 0 0 0 ENT |
| SHFT | S RST | SHFT H 7 | F 5 L NDST → 1 0 ENT |
| X OUT | SHFT | D 3 | → C 2 0 1 0 ENT |

Shift Right (SHFR)

- 230
- 240
- 250-1
- 260

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

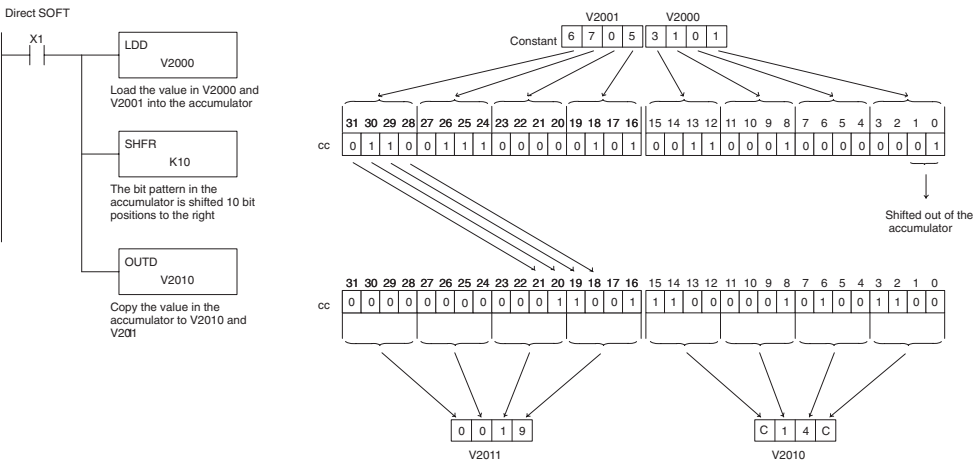
SHFR

A aaa

| Operand Data Type | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|-------------|---------------------|---------------------|---------------------|
| | A | aaa | aaa | aaa |
| V-memory | V | All (See page 3-53) | All (See page 3-54) | All (See page 3-55) |
| Constant | K | 1-32 | 1-32 | 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 10 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.

| | |
|-----|------|
| DS | Used |
| HPP | Used |



Handheld Programmer Keystrokes

| | | | |
|-------|-------|----------|-------------------|
| STR | → | 1 | T |
| SHFT | L DST | D 3 | D 3 → C 2 0 0 0 T |
| SHFT | S RST | SHFT H 7 | F 5 R OR → 1 0 T |
| X OUT | SHFT | D 3 | → C 2 0 1 0 T |

Rotate Left (ROTL)

- 230
- 240
- 250-1
- 260

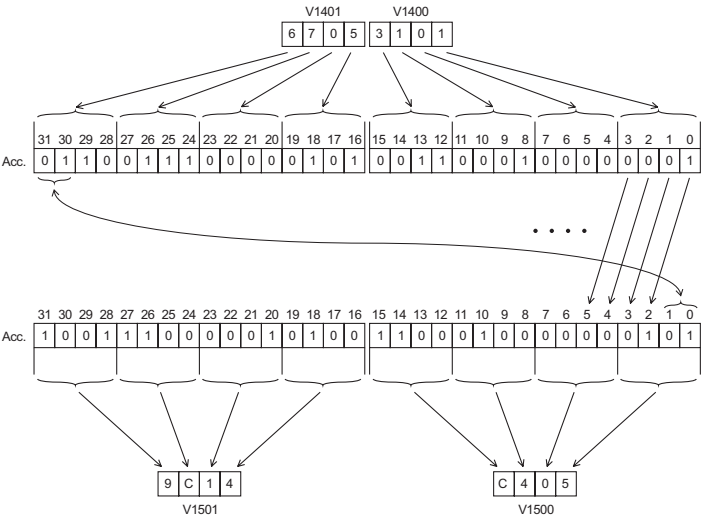
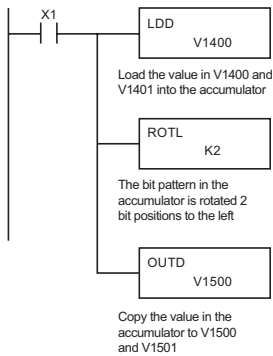
Rotate Left is a 32-bit instruction that rotates the bits in the accumulator a specified number (Aaaa) of places to the left.

ROTL
A aaa

| Operand | Data Type | DL250-1 Range | DL260 Range |
|----------|-----------|---------------------|---------------------|
| | A | aaa | aaa |
| V-memory | V | All (See page 3-55) | All (See page 3-56) |
| Constant | K | 1-32 | 1-32 |

In the following example, when X1 is on, the value in V1400 and V1401 will be loaded into the accumulator using the Load Double instruction. The bit pattern in the accumulator is rotated 2 bit positions to the left using the Rotate Left instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.

DirectSOFT



Handheld Programmer Keystrokes

| | | | | | | | | | | | | | | |
|------|-----|-------|---|-------|-----|-----|---|-------|---|---|---|-----|---|-----|
| \$ | STR | → | B | 1 | ENT | | | | | | | | | |
| SHFT | L | ANDST | D | 3 | → | B | 1 | E | 4 | A | 0 | A | 0 | ENT |
| SHFT | R | ORN | O | INST# | T | MLR | L | ANDST | → | C | 2 | ENT | | |
| GX | OUT | SHFT | D | 3 | → | B | 1 | F | 5 | A | 0 | A | 0 | ENT |

Rotate Right (ROTR)

- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260

Rotate Right is a 32-bit instruction that rotates the bits in the accumulator a specified number (Aaaa) of places to the right.

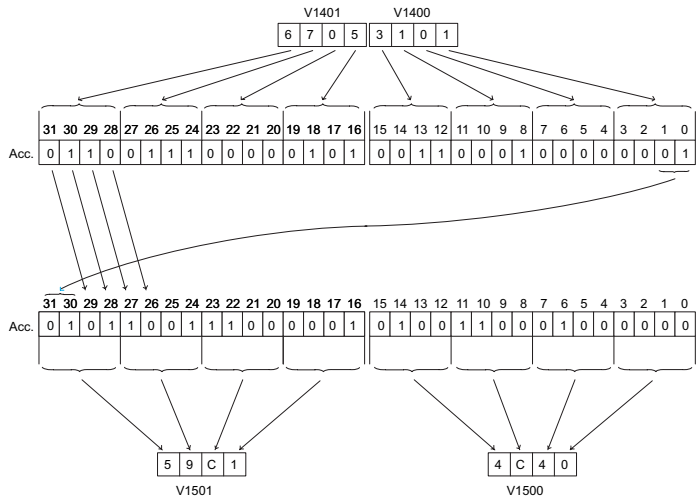
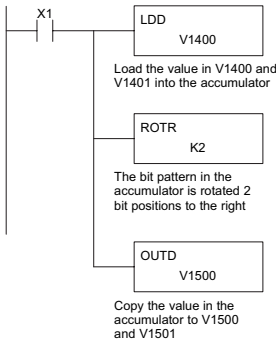
ROTR
A aaa

| Operand Data Type | DL250-1 Range | DL260 Range |
|-------------------|---------------|---------------------|
| A | aaa | aaa |
| V-memory | V | All (See page 3-55) |
| Constant | K | 1-32 |

In the following example, when X1 is on, the value in V1400 and V1401 will be loaded into the accumulator using the Load Double instruction. The bit pattern in the accumulator is rotated 2 bit positions to the right using the Rotate Right instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.

5

DirectSOFT



Handheld Programmer Keystrokes

| | | | | | |
|------|-----|-------|---|-----|-----|
| \$ | STR | → | B | 1 | ENT |
| SHFT | L | ANDST | D | 3 | → |
| SHFT | R | ORN | T | MLR | → |
| GX | OUT | SHFT | D | 3 | → |

Encode (ENCO)

✓ 230

✓ 240

✓ 250-1

✓ 260

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.

DS Used

HPP Used

ENCO

Discrete Bit Flags

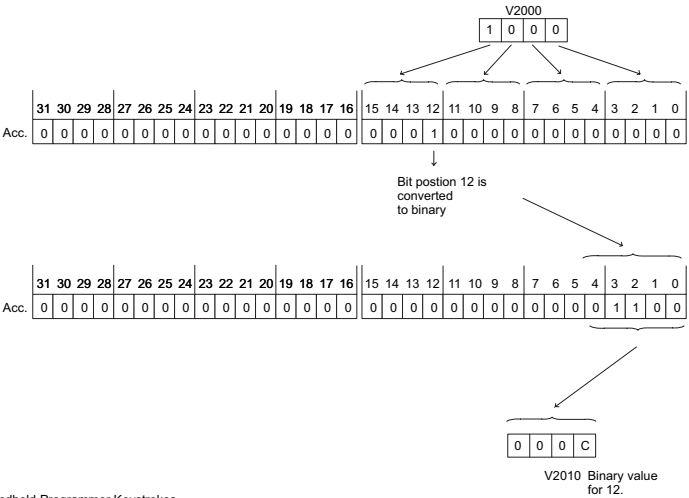
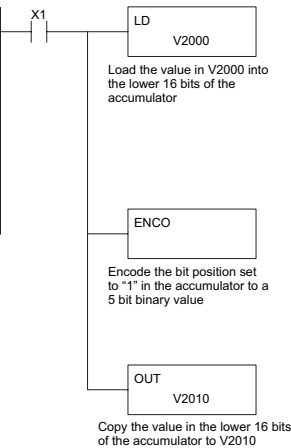
Description

| | |
|------|---|
| 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.

DirectSOFT



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)

- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260

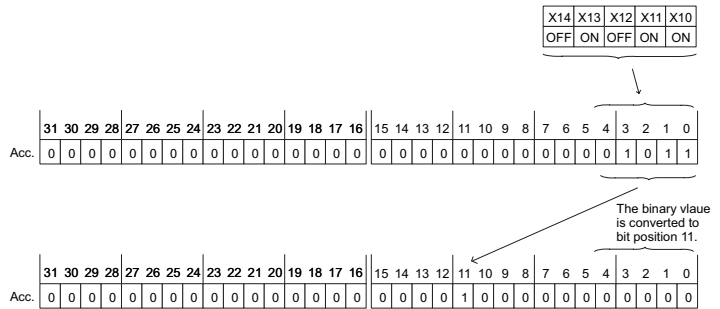
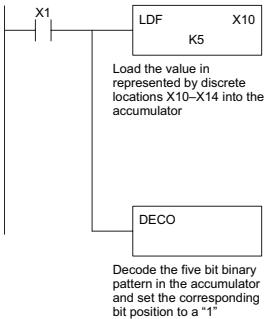
The Decode instruction decodes a 5-bit binary value of 0 to 31 (0 to 1F HEX) in the accumulator by setting the appropriate bit position to a 1. If the accumulator contains the value F (HEX), bit 15 will be set in the accumulator. If the value to be decoded is greater than 31, the number is divided by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

DECO

| | |
|-----|------|
| DS | Used |
| HPP | Used |

In the following example, when X1 is on, the value formed by discrete locations X10–X14 is loaded into the accumulator using the Load Formatted instruction. The 5-bit binary pattern in the accumulator is decoded by setting the corresponding bit position to a “1” using the Decode instruction.

DirectSOFT



Handheld Programmer Keystrokes

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

Number Conversion Instructions (Accumulator)

Binary (BIN)

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

☒ **240**

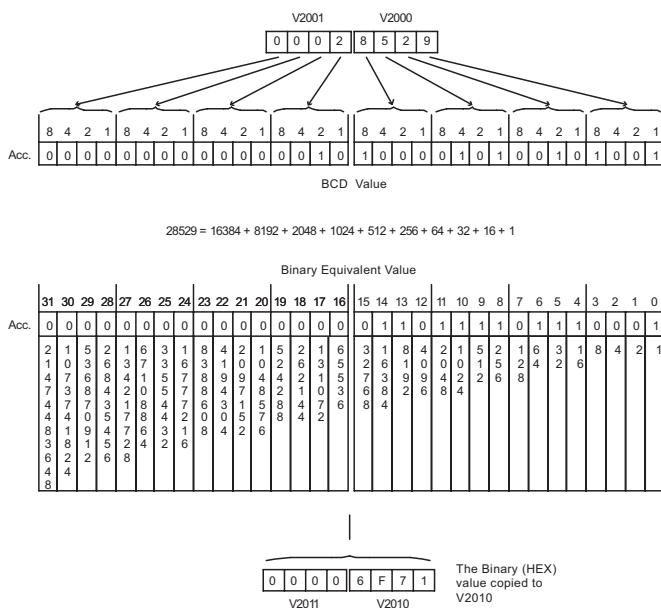
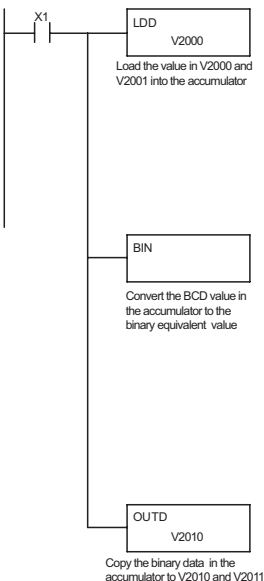
☒ **250-1**

☒ **260**

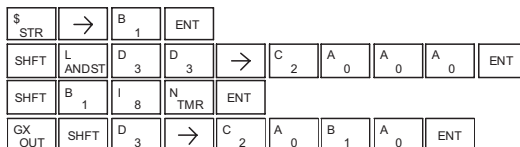
In the following example, when X1 is on, the value in V2000 and V2001 is loaded into the accumulator using the Load Double instruction. The BCD value in the accumulator is converted to the binary (HEX) equivalent using the BIN instruction. The binary value in the accumulator is copied to V2010 and V2011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

BIN

DirectSOFT



Handheld Programmer Keystrokes



Binary Coded Decimal (BCD)

✓ 230

✓ 240

✓ 250-1

✓ 260

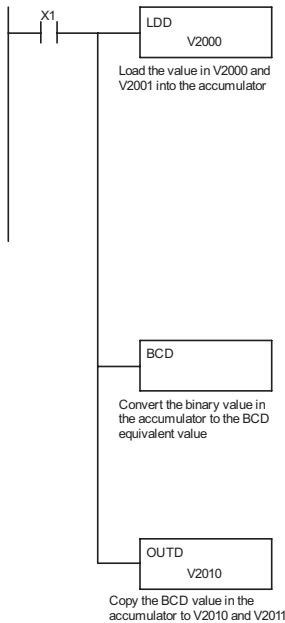
The Binary Coded Decimal instruction converts a binary value in the accumulator to the equivalent BCD value. The result resides in the accumulator.

BCD

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.

| | |
|-----|------|
| DS | Used |
| HPP | Used |

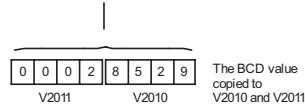
DirectSOFT



| V2001 | | | | | | | | | | | | | | | | V2000 | | | | | | | | | | | | | | | | | | |
|--------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 0 0 0 | | | | | | | | | | | | | | | | 6 F 7 1 | | | | | | | | | | | | | | | | | | |
| Binary Value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |
| Acc. | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 2 | 1 | 0 | 5 | 3 | 2 | 6 | 6 | 1 | 6 | 8 | 4 | 1 | 5 | 2 | 2 | 1 | 6 | 3 | 2 | 2 | 2 | 2 | 0 | 5 | 1 | 2 | 5 | 1 | 6 | 8 | 4 | 2 | 1 | |
| 1 | 0 | 3 | 6 | 8 | 4 | 1 | 6 | 7 | 3 | 9 | 9 | 4 | 2 | 0 | 4 | 2 | 1 | 3 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 4 | 2 | 3 | 2 | 1 | 6 | 3 | 1 | |
| 4 | 7 | 3 | 8 | 4 | 2 | 0 | 5 | 7 | 7 | 8 | 4 | 7 | 8 | 2 | 1 | 0 | 3 | 5 | 5 | 3 | 9 | 9 | 4 | 2 | 2 | 4 | 2 | 1 | 6 | 3 | 1 | 8 | 4 | |
| 7 | 3 | 8 | 4 | 2 | 0 | 5 | 7 | 7 | 7 | 8 | 4 | 7 | 8 | 2 | 1 | 0 | 3 | 5 | 5 | 3 | 9 | 9 | 4 | 2 | 2 | 4 | 2 | 1 | 6 | 3 | 1 | 8 | 4 | |
| 4 | 4 | 3 | 7 | 0 | 5 | 7 | 7 | 7 | 7 | 8 | 4 | 7 | 8 | 2 | 1 | 0 | 3 | 5 | 5 | 3 | 9 | 9 | 4 | 2 | 2 | 4 | 2 | 1 | 6 | 3 | 1 | 8 | 4 | |
| 4 | 4 | 3 | 7 | 0 | 5 | 7 | 7 | 7 | 7 | 8 | 4 | 7 | 8 | 2 | 1 | 0 | 3 | 5 | 5 | 3 | 9 | 9 | 4 | 2 | 2 | 4 | 2 | 1 | 6 | 3 | 1 | 8 | 4 | |
| 8 | 1 | 9 | 4 | 2 | 0 | 5 | 7 | 7 | 7 | 8 | 4 | 7 | 8 | 2 | 1 | 0 | 3 | 5 | 5 | 3 | 9 | 9 | 4 | 2 | 2 | 4 | 2 | 1 | 6 | 3 | 1 | 8 | 4 | |
| 3 | 8 | 2 | 4 | 2 | 0 | 5 | 7 | 7 | 7 | 8 | 4 | 7 | 8 | 2 | 1 | 0 | 3 | 5 | 5 | 3 | 9 | 9 | 4 | 2 | 2 | 4 | 2 | 1 | 6 | 3 | 1 | 8 | 4 | |
| 6 | 2 | 4 | 2 | 0 | 5 | 7 | 7 | 7 | 7 | 8 | 4 | 7 | 8 | 2 | 1 | 0 | 3 | 5 | 5 | 3 | 9 | 9 | 4 | 2 | 2 | 4 | 2 | 1 | 6 | 3 | 1 | 8 | 4 | |
| 4 | 8 | 4 | 2 | 0 | 5 | 7 | 7 | 7 | 7 | 8 | 4 | 7 | 8 | 2 | 1 | 0 | 3 | 5 | 5 | 3 | 9 | 9 | 4 | 2 | 2 | 4 | 2 | 1 | 6 | 3 | 1 | 8 | 4 | |

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

| BCD Equivalent Value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----------------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 | 4 | 2 | 1 | 8 | 4 | 2 | 1 | 8 | 4 | 2 | 1 | 8 | 4 | 2 | 1 | 8 | 4 | 2 | 1 | 8 | 4 | 2 | 1 | 8 | 4 | 2 | 1 | 8 | 4 | 2 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |



Handheld Programmer Keystrokes

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

Invert (INV)

- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260
- The Invert instruction inverts or takes the one's complement of the 32-bit value in the accumulator. The result resides in the accumulator.

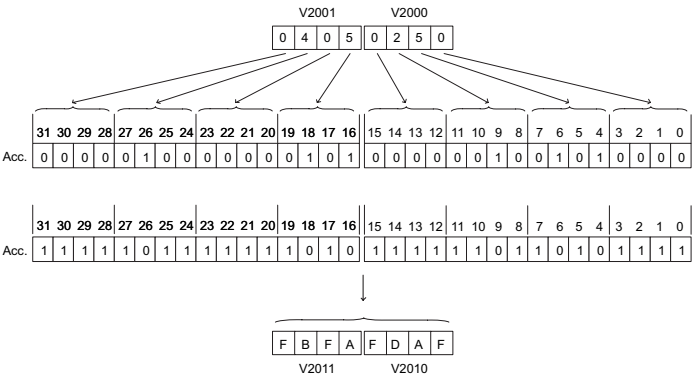
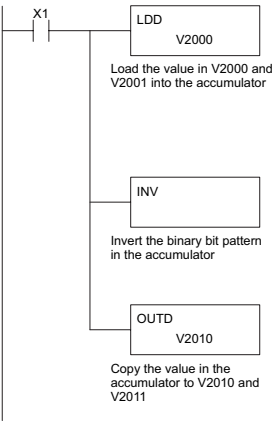
INV

| | |
|-----|------|
| DS | Used |
| HPP | Used |

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.

5

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 | I | 8 | N | TMR | V | AND | ENT | | | | | | | | | |
| GX | OUT | SHFT | D | 3 | → | C | 2 | A | 0 | B | 1 | A | 0 | ENT | | |

Ten's Complement (BCDCPL)

- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260

The Ten's Complement instruction takes the 10's complement (BCD) of the 8-digit accumulator. The result resides in the accumulator. The calculation for this instruction is :

BCDCPL

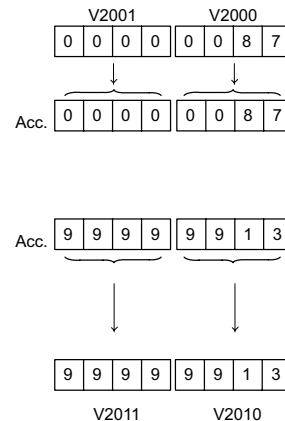
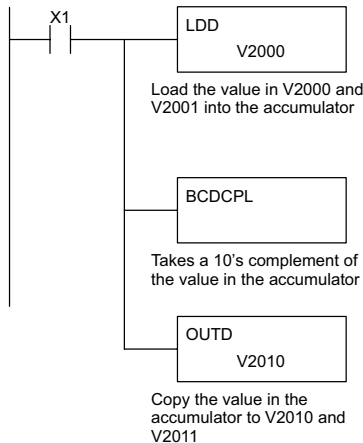
$$\begin{array}{r} 10000000 \\ - \text{accumulator value} \\ \hline 10's \text{ complement value} \end{array}$$

| | |
|-----|------|
| DS | Used |
| HPP | Used |

In the following example when X1 is on, the value in V2000 and V2001 is loaded into the accumulator. The 10's complement is taken for the 8-digit accumulator using the Ten's Complement instruction. The value in the accumulator is copied to V2010 and V2011 using the Out Double instruction.

5

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 | C 2 | D 3 | C 2 | P CV | L ANDST | ENT | | | | | | | | |
| GX OUT | SHFT | D 3 | → | C 2 | A 0 | B 1 | A 0 | ENT | | | | | | | |

Binary to Real Conversion (BTOR)

- 230
- 240
- 250-1
- 260

The Binary-to-Real instruction converts a binary value in the accumulator to its equivalent real number (floating point) format. The result resides in the accumulator. Both the binary and the real number may use all 32 bits of the accumulator.

BTOR



NOTE: This instruction only works with unsigned **binary**, or **decimal** values. It will not work with signed decimal values.

Discrete Bit Flags

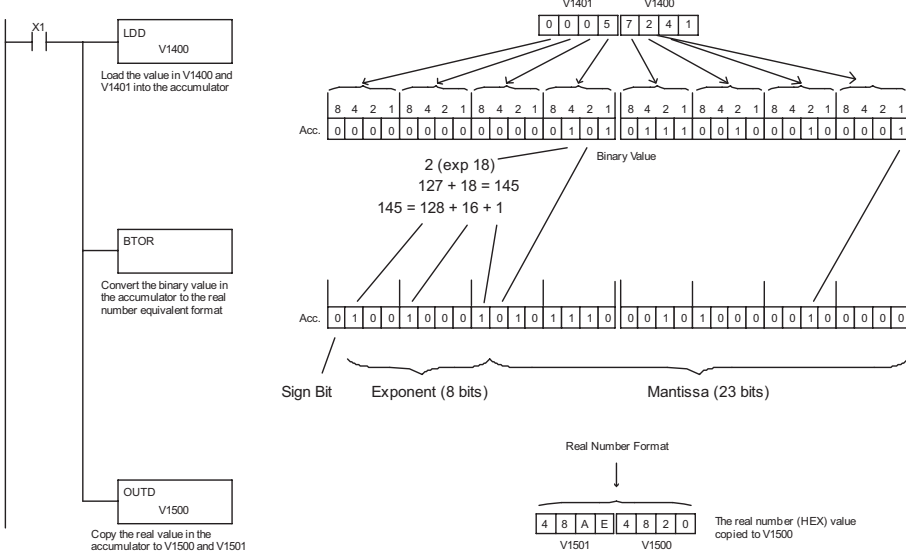
Description

| | |
|------|--|
| SP63 | On when the result of the instruction causes the value in the accumulator to be zero |
| SP70 | On anytime the value in the accumulator is negative |

In the following example, when X1 is on, the value in V1400 and V1401 is loaded into the accumulator using the Load Double instruction. The BTOR instruction converts the binary value in the accumulator the equivalent real number format. The binary weight of the MSB is converted to the real number exponent by adding it to 127 (decimal). Then the remaining bits are copied to the mantissa as shown. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction. The Handheld Programmer would display the binary value in V1500 and V1501 as a HEX value.

| | |
|-----|------|
| DS | Used |
| HPP | Used |

DirectSOFT



Handheld Programmer Keystrokes

| | | | | | | | | | | | | | | | |
|------|-----|-------|---|-----|-----|-------|---|-----|-----|---|---|---|---|-----|--|
| \$ | STR | → | B | 1 | ENT | | | | | | | | | | |
| SHFT | L | ANDST | D | 3 | → | B | 1 | E | 4 | A | 0 | A | 0 | ENT | |
| SHFT | B | 1 | T | MLR | O | INST# | R | ORN | ENT | | | | | | |
| GX | OUT | SHFT | D | 3 | → | B | 1 | F | 5 | A | 0 | A | 0 | ENT | |

Real to Binary Conversion (RTOB)



230



240



250-1



260

The Real-to-Binary instruction converts the real number in the accumulator to a binary value. The result resides in the accumulator. Both the binary and the real number may use all 32 bits of the accumulator.

RTOB



NOTE₁: The decimal portion of the result will be rounded down (14.1 to 14 or - 14.1 to -15).

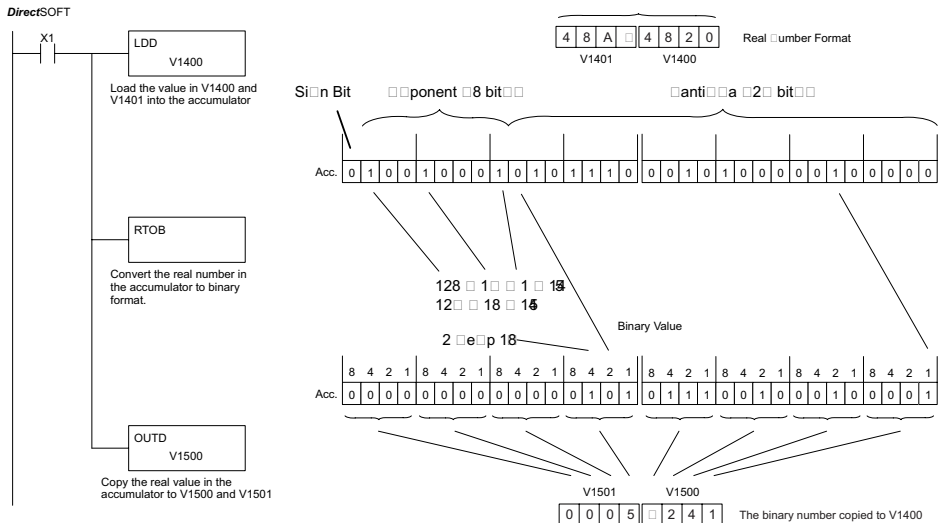
NOTE₂: If the real number is negative, it becomes a signed decimal value.

| Discrete Bit Flags | Description |
|--------------------|--|
| SP63 | On when the result of the instruction causes the value in the accumulator to be zero |
| SP70 | On anytime the value in the accumulator is negative |
| SP72 | On anytime the value in the accumulator is a valid floating point number |
| SP73 | On when a signed addition or subtraction results in an incorrect sign bit. |
| SP75 | On when a number cannot be converted to binary |

5

In the following example, when X1 is on, the value in V1400 and V1401 is loaded into the accumulator using the Load Double instruction. The RTOB instruction converts the real value in the accumulator the equivalent binary number format. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction. The Handheld Programmer would display the binary value in V1500 and V1501 as a HEX value.

| | |
|-----|------|
| DS | Used |
| HPP | Used |



Handheld Programmer Display

| | | | |
|--------|---------|------|-----------------------|
| STR | → | B 1 | OUT |
| SIFT | L A DST | D | → B 1 4 A 0 A 0 OUT |
| SIFT | R OR | T LR | OUT |
| OX OUT | SIFT | D | → B 1 F 5 A 0 A 0 OUT |

Radian Real Conversion (RADR)

- ☒ 230 The Radian Real Conversion instruction converts the real
- ☒ 240 degree value stored in the accumulator to the equivalent real
- ☒ 250-1 number in radians. The result resides in the accumulator.
- ☒ 260

RADR

Degree Real Conversion (DEGR)

- ☒ 230 The Degree Real instruction converts the degree real radian
- ☒ 240 value stored in the accumulator to the equivalent real number
- ☒ 250-1 in degrees. The result resides in the accumulator.
- ☒ 260

DEGR

The two instructions described above convert real numbers in the accumulator from degree format to radian format, and visa-versa. In degree format, a circle contains 360 degrees. In radian format, a circle contains 2 π . These convert between both positive and negative real numbers, and for angles greater than a full circle. These functions are very useful when combined with the transcendental trigonometric functions (see the section on math instructions).

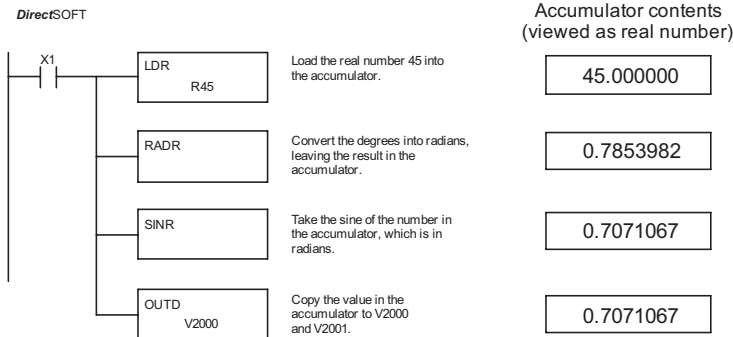
| | |
|-----|------|
| DS | Used |
| HPP | N/A |

| Discrete Bit Flags | Description |
|--------------------|--|
| SP63 | On when the result of the instruction causes the value in the accumulator to be zero |
| SP70 | On anytime the value in the accumulator is negative |
| SP71 | On anytime the V-memory specified by a pointer (P) is not valid |
| SP72 | On anytime the value in the accumulator is a valid floating point number |
| SP74 | On anytime a floating point math operation results in an underflow error |
| SP75 | On when a BCD instruction is executed and a NON-BCD number was encountered |



NOTE: The current HPP does not support real number entry with automatic conversion to the 32-bit IEEE format. You must use **DirectSOFT** for entering real numbers, using the **LDR (Load Real)** instruction.

The following example takes the sine of 45 degrees. Since transcendental functions operate only on real numbers, we do a LDR (Load Real) 45. The trig functions operate only in radians, so we must convert the degrees to radians by using the RADR command. After using the SINR (Sine Real) instruction, we use an OUTD (Out Double) instruction to move the result from the accumulator to V-memory. The result is 32-bits wide, requiring the Out Double to move it.



ASCII to HEX (ATH)

- ☐ 230
☐ 240
☒ 250-1
☒ 260

The ASCII TO HEX instruction converts a table of ASCII values to a specified table of HEX values. ASCII values are two digits and their HEX equivalents are one digit.

ATH
Vaaa

This means an ASCII table of four V-memory locations would only require two V-memory locations for the equivalent HEX table. The function parameters are loaded into the accumulator stack and the accumulator by two additional instructions. Listed below are the steps necessary to program an ASCII to HEX table function. The example on the following page shows a program for the ASCII to HEX table function.

Step 1: Load the number of V-memory locations for the ASCII table into the first level of the accumulator stack.

Step 2: Load the starting V-memory location for the ASCII table into the accumulator. This parameter must be a HEX value.

Step 3: Specify the starting V-memory location (Vaaa) for the HEX table in the ATH instruction.

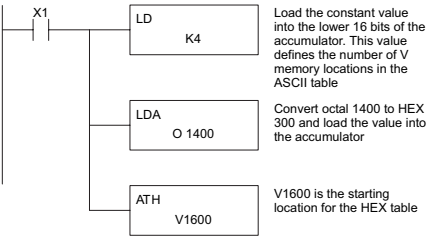
Helpful hint: — For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

| Operand Data Type | | DL250-1 Range | DL260 Range |
|-------------------|---|---------------------|---------------------|
| | | aaa | aaa |
| V-memory | V | All (See page 3-55) | All (See page 3-56) |

In the example on the following page, when X1 is ON, the constant (K4) is loaded into the accumulator using the Load instruction and will be placed in the first level of the accumulator stack when the next Load instruction is executed. The starting location for the ASCII table (V1400) is loaded into the accumulator using the Load Address instruction. The starting location for the HEX table (V1600) is specified in the ASCII to HEX instruction. The table below lists valid ASCII values for ATH conversion.

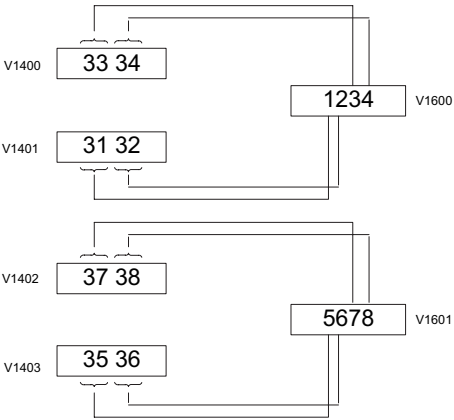
| ASCII Values Valid for ATH Conversion | | | |
|---------------------------------------|-----------|-------------|-----------|
| ASCII | Hex Value | ASCII Value | Hex Value |
| 30 | 0 | 38 | 8 |
| 31 | 1 | 39 | 9 |
| 32 | 2 | 41 | A |
| 33 | 3 | 42 | B |
| 34 | 4 | 43 | C |
| 35 | 5 | 44 | D |
| 36 | 6 | 45 | E |
| 37 | 7 | 46 | F |

DirectSOFT

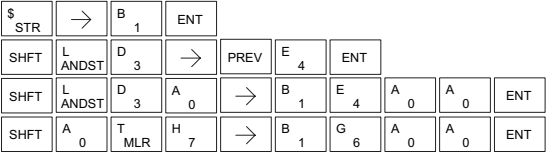


ASCII TABLE

Hexadecimal Equivalents



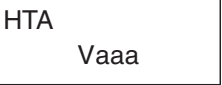
Handheld Programmer Keystrokes



HEX to ASCII (HTA)

- 230
- 240
- 250-1
- 260

The HEX to ASCII instruction converts a table of HEX values to a specified table of ASCII values. HEX values are one digit and their ASCII equivalents are two digits.



This means a HEX table of two V-memory locations would require four V-memory locations for the equivalent ASCII table. The function parameters are loaded into the accumulator stack and the accumulator by two additional instructions. Listed below are the steps necessary to program a HEX to ASCII table function. The example on the following page shows a program for the HEX to ASCII table function.

| | |
|-----|------|
| DS | Used |
| HPP | N/A |

Step 1: Load the number of V-memory locations in the HEX table into the first level of the accumulator stack.

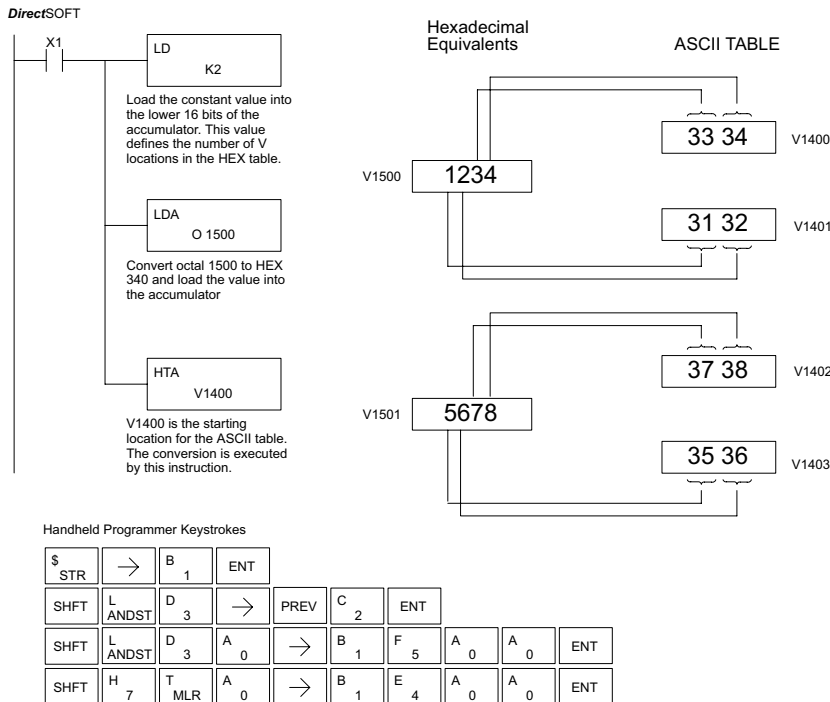
Step 2: Load the starting V-memory location for the HEX table into the accumulator. This parameter must be a HEX value.

Step 3: Specify the starting V-memory location (Vaaa) for the ASCII table in the HTA instruction.

Helpful hint: — For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

| Operand Data Type | DL250-1 Range | DL260 Range |
|-------------------|---------------|---------------------|
| | aaa | aaa |
| V-memory | V | All (See page 3-55) |

In the following example, when X1 is ON, the constant (K2) is loaded into the accumulator using the Load instruction. The starting location for the HEX table (V1500) is loaded into the accumulator using the Load Address instruction. The starting location for the ASCII table (V1400) is specified in the HEX to ASCII instruction.



The table below lists valid ASCII values for HTA conversion.

| ASCII Values Valid for HTA Conversion | | | |
|---------------------------------------|-------------|-----------|-------------|
| Hex Value | ASCII Value | Hex Value | ASCII Value |
| 0 | 30 | 8 | 38 |
| 1 | 31 | 9 | 39 |
| 2 | 32 | A | 41 |
| 3 | 33 | B | 42 |
| 4 | 34 | C | 43 |
| 5 | 35 | D | 44 |
| 6 | 36 | E | 45 |
| 7 | 37 | F | 46 |

Segment (SEG)

- ☒

230

The BCD / Segment instruction converts a 4digit HEX value
- ☒

240

in the accumulator to a 7-segment display format. The result
- ☒

250-1

resides in the accumulator.
- ☒

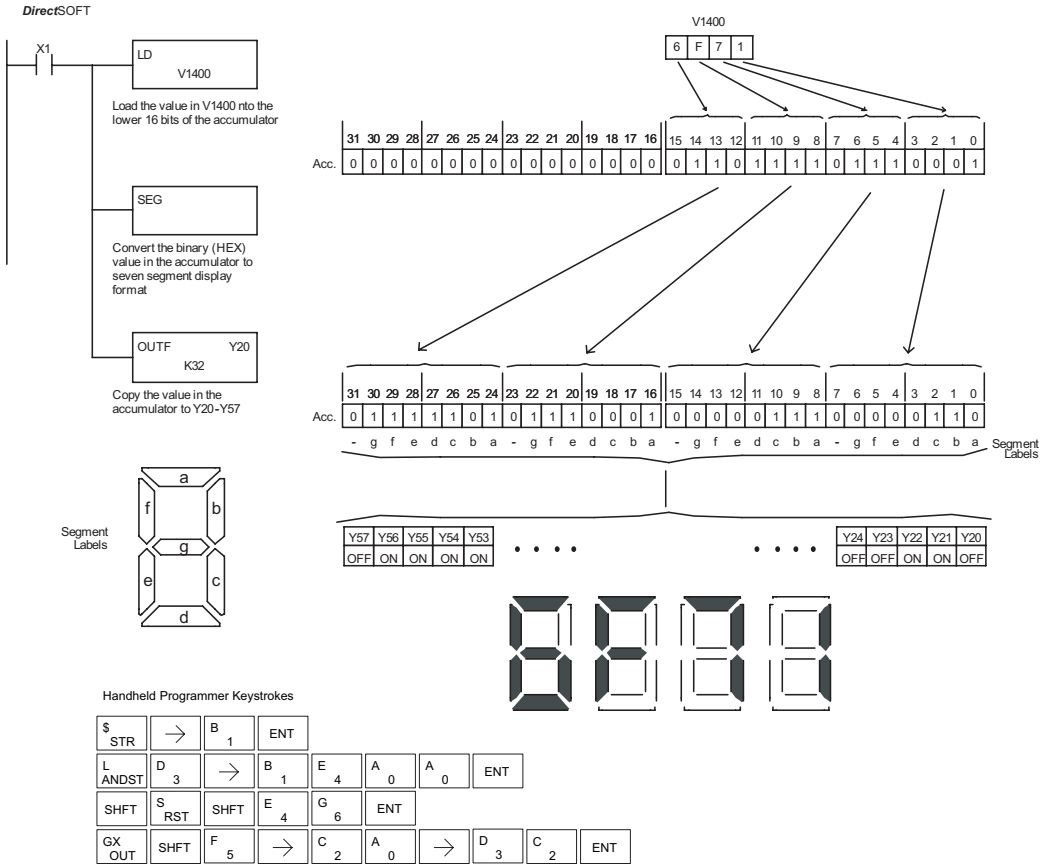
260

SEG

In the following example, when X1 is on, the value in V1400 is loaded into the lower 16 bits of the accumulator using the Load instruction. The binary (HEX) value in the accumulator is converted to 7-segment format using the Segment instruction. The bit pattern in the accumulator is copied to Y20–Y57 using the Out Formatted instruction.

| | |
|-----|------|
| DS | Used |
| HPP | Used |

5



Gray Code (GRAY)

- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260

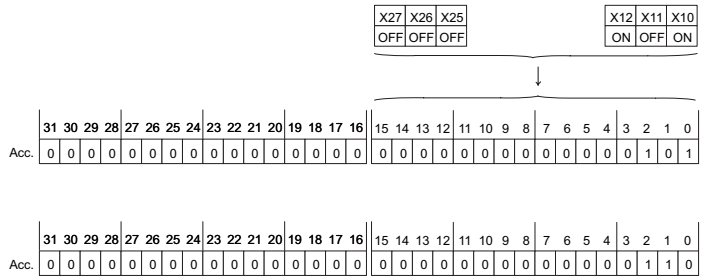
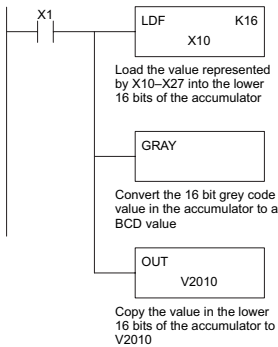
| | |
|-----|------|
| DS | Used |
| HPP | Used |

The Gray code instruction converts a 16-bit gray code value to a BCD value. The BCD conversion requires 10 bits of the accumulator. The upper 22 bits are set to "0." This instruction is designed for use with devices (typically encoders) that use the gray code numbering scheme. The Gray Code instruction will directly convert a gray code number to a BCD number for devices having a resolution of 512 or 1024 counts per revolution. If a device having a resolution of 360 counts per revolution is to be used, you must subtract a BCD value of 76 from the converted value to obtain the proper result. For a device having a resolution of 720 counts per revolution, you must subtract a BCD value of 152.

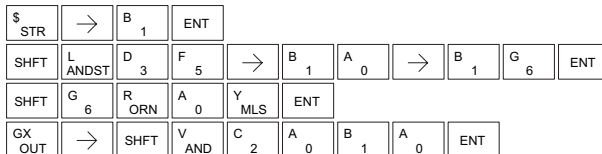
GRAY

In the following example, when X1 is ON the binary value represented by X10–X27 is loaded into the accumulator using the Load Formatted instruction. The gray code value in the accumulator is converted to BCD using the Gray Code instruction. The value in the lower 16 bits of the accumulator is copied to V2010.

DirectSOFT



Handheld Programmer Keystrokes



Gray Code

| | |
|------------|------|
| 0000000000 | 0000 |
| 0000000001 | 0001 |
| 0000000011 | 0002 |
| 0000000010 | 0003 |
| 0000000110 | 0004 |
| 0000000111 | 0005 |
| 0000000101 | 0006 |
| 0000000100 | 0007 |
| . | . |
| . | . |
| . | . |
| 1000000001 | 1022 |
| 1000000000 | 1023 |



Shuffle Digits (SFLDGT)

- 230
- 240
- 250-1
- 260

The Shuffle Digits instruction shuffles a maximum of 8 digits rearranging them in a specified order. This function requires parameters to be loaded into the first level of the accumulator stack and the accumulator with two additional instructions. Listed below are the steps necessary to use the Shuffle Digit function. The example on the following page shows a program for the Shuffle Digits function.

Step 1: Load the value (digits) to be shuffled into the first level of the accumulator stack.

Step 2: Load the order that the digits will be shuffled to into the accumulator.

Step 3: Insert the SFLDGT instruction.

SFLDGT

| | |
|-----|------|
| DS | Used |
| HPP | Used |

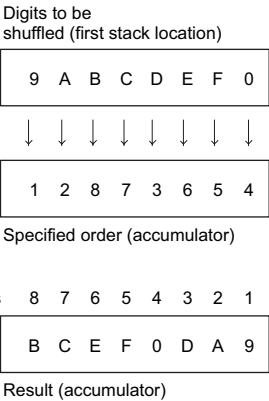
NOTE: If the number used to specify the order contains a 0 or 9–F, the corresponding position will be set to 0.

See example on the next page.

NOTE: If the number used to specify the order contains duplicate numbers, the most significant duplicate number is valid. The result resides in the accumulator.

Shuffle Digits Block Diagram

A maximum of 8 digits can be shuffled. The bit positions in the first level of the accumulator stack define the digits to be shuffled. They correspond to the bit positions in the accumulator that define the order in which the digits will be shuffled. The digits are shuffled and the result resides in the accumulator.



Example C shows how the shuffle digits works when duplicate numbers are used specifying the order the digits are to be shuffled. Notice when the Shuffle Digits instruction is executed, the most significant duplicate number in the order specified is used in the result.

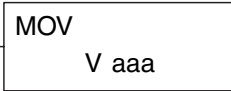
[illegible]

Table Instructions

Move (MOV)

- 230
- 240
- 250-1
- 260

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.

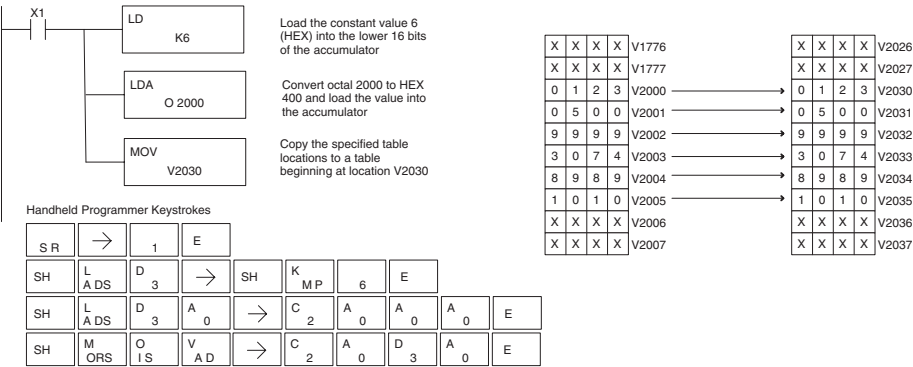


- Step 1: Load the number of V-memory locations to be moved into the first level of the accumulator stack. This parameter is a HEX value (KFFF max, 7777 octal).
- Step 2: Load the starting V-memory location for the locations to be moved into the accumulator. This parameter must be a HEX value.
- Step 3: Insert the MOVE instruction which specifies starting V-memory location (Vaaa) for the destination table.

Helpful hint: — For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

| Operand Data Type | | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|---|---------------------|---------------------|---------------------|---------------------|
| | | aaa | aaa | aaa | aaa |
| V-memory | V | All (See page 3-53) | All (See page 3-54) | All (See page 3-55) | All (See page 3-56) |
| Pointer | P | All (See page 3-53) | All (See page 3-54) | All (See page 3-55) | All (See page 3-56) |

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.



Move Memory Cartridge (MOVMC)

Load Label (LDLBL)

- ✓ 230
- ✓ 240
- ✓ 250-1
- ✓ 260

| | |
|-----|------|
| DS | Used |
| HPP | Used |

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.

MOVMC
V aaa

LDLBL
Kaaa

- 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 the program ladder memory and the beginning of the V-memory block into the first level of the accumulator stack.
- Step 3: Load the *source data label* (LDLBL Kaaa) into the accumulator when copying data from ladder memory to V-memory. Load the source address into the accumulator when copying data from V-memory to ladder memory. This is where the value will be copied from. If the source address is a V-memory location, the value must be entered in HEX.
- Step 4: Insert the MOVMC instruction which specifies destination (Aaaa). This is where the value will be copied to.

| Operand Data Type | | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|---|---------------------|---------------------|---------------------|---------------------|
| | | aaa | aaa | aaa | aaa |
| V-memory | V | All (See page 3-53) | All (See page 3-54) | All (See page 3-55) | All (See page 3-56) |
| Constant | K | K1-KFFFF | K1-KFFFF | K1-KFFFF | K1-KFFFF |



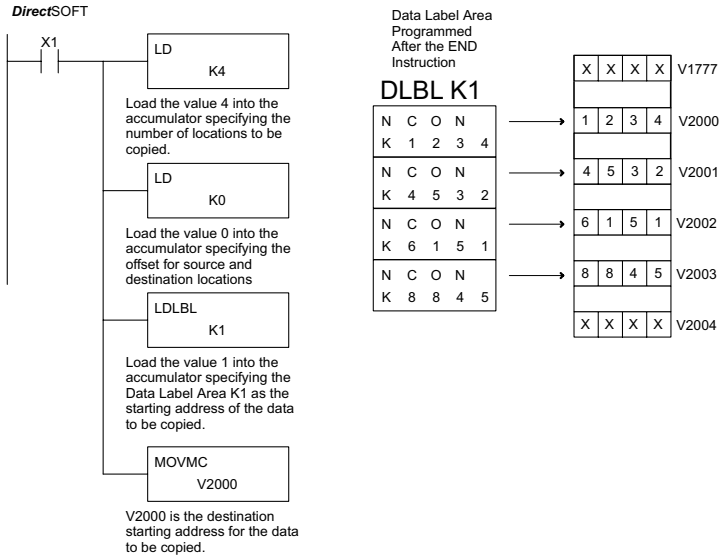
WARNING: The offset for this usage of the instruction starts at 0, but may be any number that *does not* result in data outside of the source data area being copied into the destination table. When an offset is outside of the source information boundaries, then unknown data values will be transferred into the destination table.

Copy Data From a Data Label Area to V-Memory

In the following example, data is copied from a Data Label Area to V-memory. When X1 is on, the constant value (K4) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed in the second stack location after the next Load and Load Label (LDLBL) instructions are executed. The constant value (K0) is loaded into the accumulator using the Load instruction. This value specifies the offset for the source and destination data, and is placed in the first stack location after the LDLBL instruction is executed. The source address where data is being copied from is loaded into the accumulator using the LDLBL instruction. The MOVMC instruction specifies the destination starting location and executes the copying of data from the Data Label Area to V-memory.

- 230
- 240
- 250-1
- 260

| | |
|-----|------|
| DS | Used |
| HPP | Used |



Handheld Programmer Keystrokes

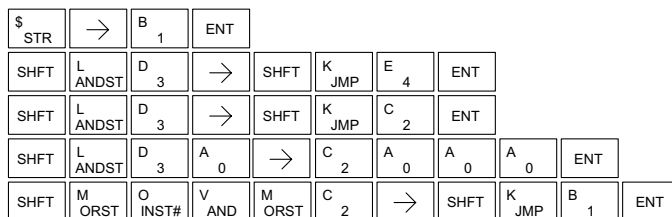
| | | | | | | | | | | | | | | | | | | | |
|------|-----|-------|----------------|-------|-------|----------------|-----|----------------|----------------|----------------|----------------|----------------|----------------|----------------|-----|--|--|--|--|
| \$ | STR | → | B ₁ | ENT | | | | | | | | | | | | | | | |
| 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 | | | | |



WARNING: The offset for this usage of the instruction starts at 0, but may be any number that does not result in data outside of the source data area being copied into the destination table. When an offset is outside of the source information boundaries, then unknown data values will be transferred into the destination table.

☒ 230
☒ 240
☐ 250-1
☐ 260

| | |
|-----|------|
| DS | Used |
| HPP | Used |



WARNING: The offset for this usage of the instruction starts at 0. If the offset (or the specified data table range) is large enough to cause data to be copied from V-memory to beyond the end of the DLBL area, then anything after the specified DLBL area will be replaced with invalid instructions.

Set Bit (SETBIT)

The Set Bit instruction sets a single bit to one within a range of V-memory locations.

SETBIT
V aaa

- ☐ 230
- ☐ 240
- ☐ 250-1
- ☒ 260

Reset Bit (RSTBIT)

The Reset Bit instruction resets a single bit to zero within a range of V-memory locations.

RSTBIT
V aaa

- ☐ 230
- ☐ 240
- ☐ 250-1
- ☒ 260

The following description applies to both the Set Bit and Reset Bit table instructions.

| | |
|-----|------|
| DS | Used |
| HPP | Used |

Step 1: Load the length of the table (number of V-memory locations) into the first level of the accumulator stack. This parameter must be a HEX value, 0 to FF.

Step 2: Load the starting V-memory location for the table into the accumulator. This parameter must be a HEX value. You can use the LDA instruction to convert an octal address to hex.

Step 3: Insert the Set Bit or Reset Bit instruction. This specifies the reference for the bit number of the bit you want to set or reset. The bit number is in octal, and the first bit in the table is number “0.”

Helpful hint: — Remember that each V-memory location contains 16 bits. So, the bits of the first word of the table are numbered from 0 to 17 octal. For example, if the table length is 6 words, then 6 words = (6 x 16) bits, = 96 bits (decimal), or 140 octal. The permissible range of bit reference numbers would be 0 to 137 octal. Flag 53 will be set if the bit specified is outside the range of the table.

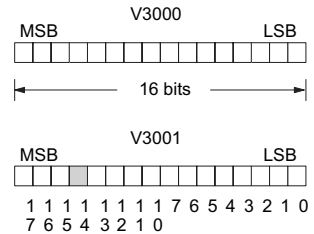
| Operand Data Type | DL260 Range |
|-------------------|---------------------|
| | aaa |
| V-memory V | All (See page 3-56) |

| Discrete Bit Flags | Description |
|--------------------|---|
| SP53 | On when the bit number which is referred in the Set Bit or Reset Bit exceeds the range of the table |



NOTE: Status flags are only valid until the end of the scan or another instruction that uses the same flag is executed.

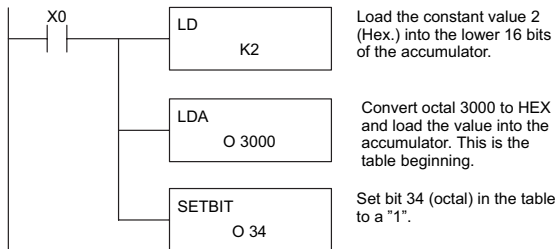
For example, suppose we have a table starting at V3000 that is two words long, as shown to the right. Each word in the table contains 16 bits, or 0 to 17 in octal. To set bit 12 in the second word, we use its octal reference (bit 14). Then we compute the bit's octal address from the start of the table, so $17 + 14 = 34$ octal. The following program shows how to set the bit as shown to a "1."



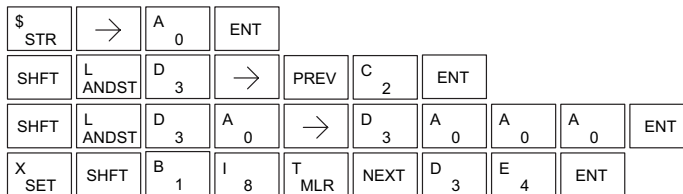
In this ladder example, we will use input X0 to trigger the Set Bit operation. First, we will load the table length (two words) into the accumulator stack. Next, we load the starting address into the accumulator. Since V3000 is an octal number we have to convert it to hex by using the LDA command. Finally, we use the Set Bit (or Reset Bit) instruction and specify the octal address of the bit (bit 34), referenced from the table beginning.

5

DirectSOFT



Handheld Programmer Keystrokes



Fill (FILL)

- 230
- 240
- 250-1
- 260

The Fill instruction fills a table of up to 255 V-memory locations with a value (Aaaa), which is either a V-memory location or a 4-digit constant. The function parameters are loaded into the first level of the accumulator stack and the accumulator by two additional instructions. Listed below are the steps necessary to program the Fill function.



| | |
|-----|------|
| DS | Used |
| HPP | Used |

Step 1: Load the number of V-memory locations to be filled into the first level of the accumulator stack. This parameter must be a HEX value, 0 to FFFF.

Step 2: Load the starting V-memory location for the table into the accumulator. This parameter must be a HEX value.

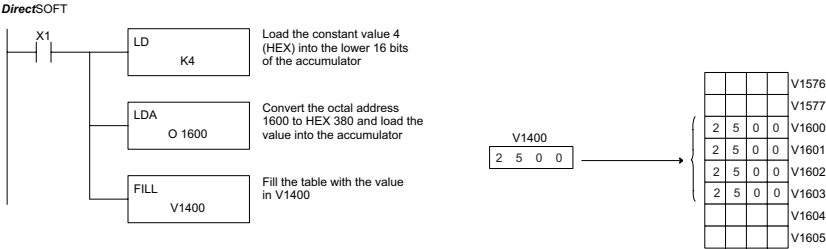
Step 3: Insert the Fill instructions which specifies the value to fill the table with.

Helpful hint: For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

| Operand Data Type | DL260 Range |
|-------------------|-----------------------------|
| | Aaaa |
| V-memory | V All (See page 3-56) |
| Pointer | P All V mem (See page 3-56) |
| Constant | K 0-FFFF |

| Discrete Bit Flag | Description |
|-------------------|--|
| SP53 | On if V-memory address is out of range |

In the following example, when X1 is on, the constant value (K4) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed on the first level of the accumulator stack when the Load Address instruction is executed. The octal address 1600 (V1600) is the starting location for the table and is loaded into the accumulator using the Load Address instruction. The value to fill the table with (V1400) is specified in the Fill instruction.



Handheld Programmer Keystrokes

| | | | | | | | | | | |
|--------|---------|-----|---------|---------|-----|-----|-----|-----|-----|-----|
| \$ STR | → | B 1 | ENT | | | | | | | |
| SHFT | L ANDST | D 3 | → | PREV | E 4 | ENT | | | | |
| SHFT | L ANDST | D 3 | A 0 | → | B 1 | G 6 | A 0 | A 0 | ENT | |
| SHFT | F 5 | I 8 | L ANDST | L ANDST | → | B 1 | E 4 | A 0 | A 0 | ENT |

Find (FIND)

-  230
-  240
-  250-1
-  **260**

The Find instruction is used to search for a specified value in a V-memory table of up to 255 locations. The function parameters are loaded into the first and second levels of the accumulator stack and the accumulator by three additional instructions. Listed below are the steps necessary to program the Find function.

FIND
A aaa

| | |
|-----|------|
| DS | Used |
| HPP | Used |

Step 1: Load the length of the table (number of V-memory locations) into the second level of the accumulator stack. This parameter must be a HEX value, 0 to FFFF.

Step 2: Load the starting V-memory location for the table into the first level of the accumulator stack. This parameter must be a HEX value.

Step 3: Load the offset from the starting location to begin the search. This parameter must be a HEX value.

Step 4: Insert the Find instruction which specifies the first value to be found in the table.

Results: The offset from the starting address to the first V-memory location which contains the search value is returned to the accumulator as a HEX value. SP53 will be set on if an address outside the table is specified in the offset or the value is not found. If the value is not found, 0 will be returned in the accumulator.

Helpful hint: For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

| Operand Data Type | | DL260 Range |
|-------------------|---|---------------------|
| A | | aaa |
| V-memory | V | All (See page 3-56) |
| Constant | K | 0-FFFF |

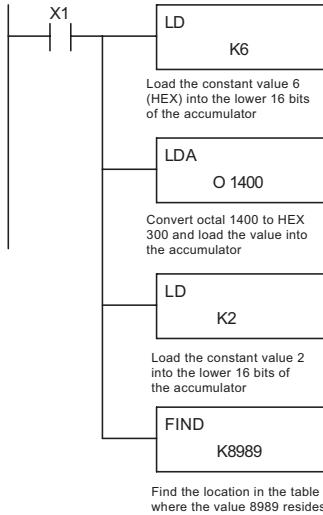
| Discrete Bit Flag | Description |
|-------------------|---|
| SP53 | On if there is no value in the table that is equal to the search value. |



NOTE: Status flags are only valid until another instruction that uses the same flags is executed. The pointer for this instruction starts at 0 and resides in the accumulator.

In the example on the following page, when X1 is on, the constant value (K6) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed in the second stack location when the following Load Address and Load instruction is executed. The octal address 1400 (V1400) is the starting location for the table and is loaded into the accumulator. This value is placed in the first level of the accumulator stack when the following Load instruction is executed. The offset (K2) is loaded into the lower 16 bits of the accumulator using the Load instruction. The value to be found in the table is specified in the Find instruction. If a value is found equal to the search value, the offset (from the starting location of the table) where the value is located will reside in the accumulator.

DirectSOFT



Offset

Begin here →

| | | | | | |
|---|---|---|---|-------|---|
| 0 | 1 | 2 | 3 | V1400 | 0 |
| 0 | 5 | 0 | 0 | V1401 | 1 |
| 9 | 9 | 9 | 9 | V1402 | 2 |
| 3 | 0 | 7 | 4 | V1403 | 3 |
| 8 | 9 | 8 | 9 | V1404 | 4 |
| 1 | 0 | 1 | 0 | V1405 | 5 |
| X | X | X | X | V1406 | |
| X | X | X | X | V1407 | |

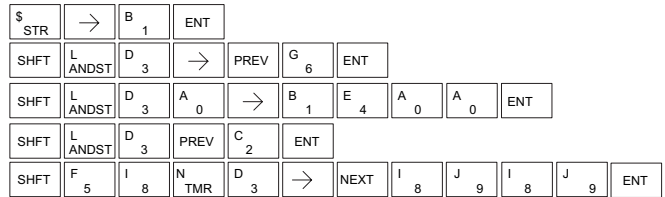
Table length

Accumulator

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 4 |
|---|---|---|---|---|---|---|

V1404 contains the location where the match was found. The value 8989 was the 4th location after the start of the specified table.

Handheld Programmer Keystrokes



Find Greater Than (FDGT)

- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260

The Find Greater Than instruction is used to search for the first occurrence of a value in a V-memory table that is greater than the specified value (Aaaa), which can be either a V-memory location or a 4-digit constant. The function parameters are loaded into the first level of the accumulator stack and the accumulator by two additional instructions. Listed below are the steps necessary to program the Find Greater Than function.

FDGT
A aaa

NOTE: This instruction does not have an offset, such as the one required for the FIND instruction.

Step 1: Load the length of the table (up to 255 locations) into the first level of the accumulator stack. This parameter must be a HEX value, 0 to FFFF.

Step 2: Load the starting V-memory location for the table into the accumulator. This parameter must be a HEX value.

Step 3: Insert the FDGT instruction which specifies the greater than search value.

Results: The offset from the starting address to the first V-memory location which contains the greater than search value is returned to the accumulator as a HEX value. SP53 will be set on if the value is not found and 0 will be returned in the accumulator.

Helpful hint: For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

| Operand Data Type | DL260 Range |
|-------------------|-----------------------|
| A | aaa |
| V-memory | V All (See page 3-56) |
| Constant | K 0-FFFF |

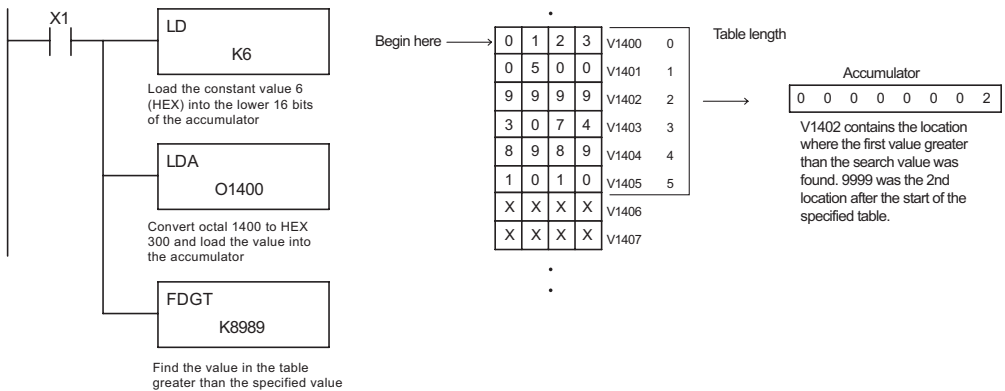
| Discrete Bit Flags | Description |
|--------------------|---|
| SP53 | On if there is no value in the table that is equal to the search value. |



NOTE: Status flags are only valid until another instruction that uses the same flags is executed. The pointer for this instruction starts at 0 and resides in the accumulator.

In the following example, when X1 is on, the constant value (K6) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed in the first stack location after the Load Address instruction is executed. The octal address 1400 (V1400) is the starting location for the table and is loaded into the accumulator. The greater than search value is specified in the Find Greater Than instruction. If a value is found greater than the search value, the offset (from the starting location of the table) where the value is located will reside in the accumulator. If there is no value in the table that is greater than the search value, a zero is stored in the accumulator and SP53 will come ON.

DirectSOFT



Handheld Programmer Keystrokes

| | | | | | |
|------|-----|-------|---|---|------|
| \$ | STR | → | B | 1 | ENT |
| SHFT | L | ANDST | D | 3 | → |
| SHFT | L | ANDST | D | 3 | A |
| SHFT | F | 5 | D | 3 | G |
| | | | | | T |
| | | | | | MLR |
| | | | | | → |
| | | | | | NEXT |
| | | | | | I |
| | | | | | 8 |
| | | | | | J |
| | | | | | 9 |
| | | | | | I |
| | | | | | 8 |
| | | | | | J |
| | | | | | 9 |
| | | | | | ENT |

Table to Destination (TTD)

-  230
-  240
-  250-1
-  260

| | |
|-----|------|
| DS | Used |
| HPP | Used |

The Table To Destination instruction moves a value from a V-memory table to a V-memory location and increments the table pointer by 1. The first V-memory location in the table contains the table pointer which indicates the next location in the table to be moved. The instruction will be executed once per scan provided the input remains on. The table pointer will reset to 1 when the value equals the last location in the table. The function parameters are loaded into the first level of the accumulator stack and the accumulator by two additional instructions. Listed below are the steps necessary to program the Table To Destination function.

TTD
Vaaa

Step 1: Load the length of the data table (number of V-memory locations) into the first level of the accumulator stack. This parameter must be a HEX value, 0 to FF.

Step 2: Load the starting V-memory location for the table into the accumulator. (Remember, the starting location of the table is used as the table pointer.) This parameter must be a HEX value.

Step 3: Insert the TTD instruction that specifies the destination V-memory location (Vaaa).

Helpful hint: For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Helpful hint: The instruction will be executed every scan if the input logic is on. If you do not want the instruction to execute for more than one scan, a one shot (PD) should be used in the input logic.

Helpful hint: The pointer location should be set to the value where the table operation will begin. The special relay SP0 or a one shot (PD) should be used so the value will only be set in one scan and will not affect the instruction operation.

| Operand Data Type | | DL260 Range |
|-------------------|---|-----------------------|
| | | aaa |
| V-memory | V | All (See page 3 - 56) |

| Discrete Bit Flags | Description |
|--------------------|---|
| SP53 | On if there is no value in the table that is equal to the search value. |

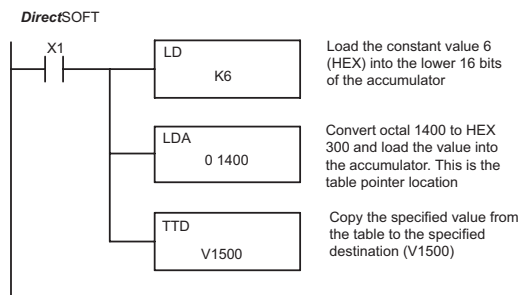
NOTE: Status flags (SPs) are only valid until:

- another instruction that uses the same flag is executed, or
- the end of the scan.



The pointer for this instruction starts at 0 and resets when the table length is reached. At first glance it may appear that the pointer should reset to 0. However, it resets to 1, not 0.

In the following example, when X1 is on, the constant value (K6) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed in the first stack location after the Load Address instruction is executed. The octal address 1400 (V1400) is the starting location for the source table and is loaded into the accumulator. Remember, V1400 is used as the pointer location, and is not actually part of the table data source. The destination location (V1500) is specified in the Table to Destination instruction. The table pointer (V1400 in this case) will be increased by “1” after each execution of the TTD instruction.



Handheld Programmer Keystrokes

| | | | | | | | | | | | |
|--------|---------|----------------|----------------|------|----------------|----------------|----------------|----------------|-----|--|--|
| \$ STR | → | B ₁ | ENT | | | | | | | | |
| SHIFT | L ANDST | D ₃ | → | PREV | G ₆ | ENT | | | | | |
| SHIFT | L ANDST | D ₃ | A ₀ | → | B ₁ | E ₄ | A ₀ | A ₀ | ENT | | |
| SHIFT | T MLR | T MLR | D ₃ | → | B ₁ | F ₅ | A ₀ | A ₀ | ENT | | |

It is important to understand how the table locations are numbered. If you examine the example table, you'll notice that the first data location, V1401, will be used when the pointer is equal to 0, and again when the pointer is equal to 6. Why? Because the pointer is only equal to 0 before the very first execution. From then on, it increments from 1 to 6, and then resets to 1.

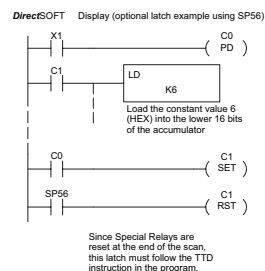
| Table | | | | | | Table Pointer | | | | | |
|-------|---|---|---|---|---|---------------|---|---|---|---|-------|
| V1401 | 0 | 5 | 0 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | V1400 |
| V1402 | 9 | 9 | 9 | 9 | | 1 | | | | | |
| V1403 | 3 | 0 | 7 | 4 | | 2 | | | | | |
| V1404 | 8 | 9 | 8 | 9 | | 3 | | | | | |
| V1405 | 1 | 0 | 1 | 0 | | 4 | | | | | |
| V1406 | 2 | 0 | 4 | 6 | | 5 | | | | | |
| V1407 | X | X | X | X | | | | | | | |

Destination

| | | | | |
|---|---|---|---|---|
| X | X | X | X | X |
|---|---|---|---|---|

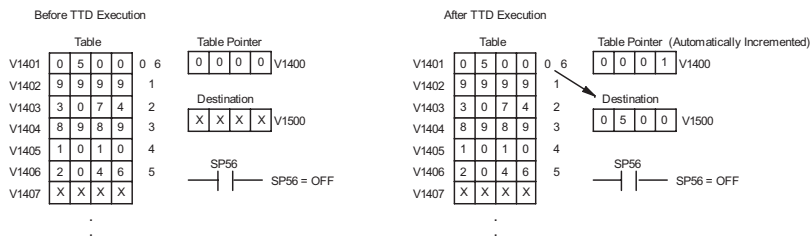
V1500

Also, our example uses a normal input contact (X1) to control the execution. Since the CPU scan is extremely fast, and the pointer increments automatically, the table would cycle through the locations very quickly. If this is a problem, you have an option of using SP56 in conjunction with a one-shot (PD) and a latch (C1 for example) to allow the table to cycle through all locations one time and then stop. The logic shown here is not required, it's just an optional method.

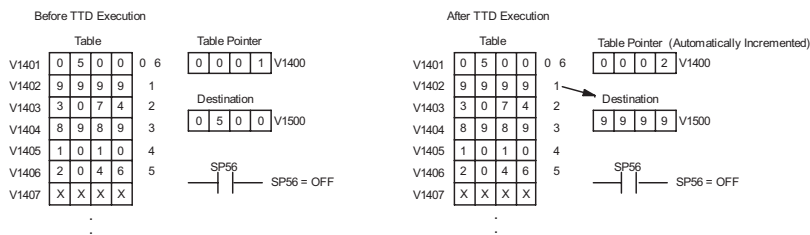


The following diagram shows the scan-by-scan results of the execution for our example program. Notice how the pointer automatically cycles from 0 to 6, and then starts over at 1 instead of 0. Also, notice how SP56 is only on until the end of the scan.

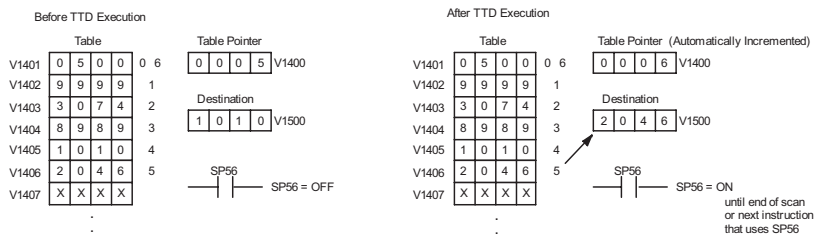
Scan N



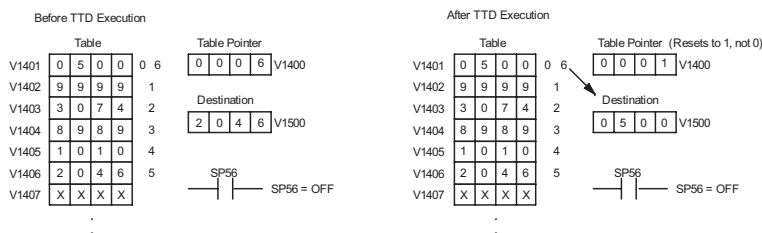
Scan N+1



Scan N+5



Scan N+6



Remove from Bottom (RFB)

| | |
|---|-------|
|  | 230 |
|  | 240 |
|  | 250-1 |
|  | 260 |

| | |
|-----|------|
| DS | Used |
| HPP | Used |

The Remove From Bottom instruction moves a value from the bottom of a V-memory table to a V-memory location and decrements a table pointer by 1. The first V-memory location in the table contains the table pointer which indicates the next location in the table to be moved. The instruction will be executed once per scan provided the input remains on. The instruction will stop operation when the pointer equals 0. The function parameters are loaded into the first level of the accumulator stack and the accumulator by 2 additional instructions. Listed below are the steps necessary to program the Remove From Bottom function.

RFB
Vaaa

Step 1: Load the length of the table (number of V-memory locations) into the first level of the accumulator stack. This parameter must be a HEX value, 0 to FF.

Step 2: Load the starting V-memory location for the table into the accumulator. (Remember, the starting location of the table blank is used as the table pointer.) This parameter must be a HEX value.

Step 3: Insert the RFB instructions which specifies destination V-memory location (Vaaa).

Helpful hint: For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Helpful hint: The instruction will be executed every scan if the input logic is on. If you do not want the instruction to execute for more than one scan, a one shot (PD) should be used in the input logic.

Helpful hint: The pointer location should be set to the value where the table operation will begin. The special relay SP0 or a one shot (PD) should be used so the value will only be set in one scan and will not affect the instruction operation.

| Operand Data Type | DL260 Range |
|-------------------|---------------------|
| | aaa |
| V-memory V | All (See page 3-56) |

| Discrete Bit Flags | Description |
|--------------------|------------------------------------|
| SP56 | On when the table pointer equals 0 |

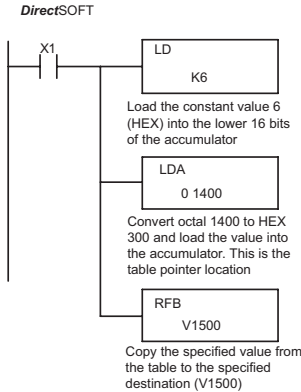
NOTE: Status flags (SPs) are only valid until:

- another instruction that uses the same flag is executed, or
- the end of the scan.



The pointer for this instruction can be set to start anywhere in the table. It is not set automatically. You have to load a value into the pointer somewhere in your program.

In the following example, when X1 is on, the constant value (K6) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed in the first stack location after the Load Address instruction is executed. The octal address 1400 (V1400) is the starting location for the source table and is loaded into the accumulator. Remember, V1400 is used as the pointer location, and is not actually part of the table data source. The destination location (V1500) is specified in the Remove From Bottom. The table pointer (V1400 in this case) will be decremented by “1” after each execution of the RFB instruction.



Handheld Programmer Keystrokes

| | | | | | | | | | |
|--------|---------|----------------|----------------|------|----------------|----------------|----------------|----------------|-----|
| \$ STR | → | B ₁ | ENT | | | | | | |
| SHFT | L ANDST | D ₃ | → | PREV | G ₆ | ENT | | | |
| SHFT | L ANDST | D ₃ | A ₀ | → | B ₁ | E ₄ | A ₀ | A ₀ | ENT |
| SHFT | R ORN | F ₅ | B ₁ | → | B ₁ | F ₅ | A ₀ | A ₀ | ENT |

It is important to understand how the table locations are numbered. If you examine the example table, you’ll notice that the first data location, V1401, will be used when the pointer is equal to one. The second data location, V1402, will be used when the pointer is equal to two, etc.

| Table | | | | | |
|-------|---|---|---|---|---|
| V1401 | 0 | 5 | 0 | 0 | 1 |
| V1402 | 9 | 9 | 9 | 9 | 2 |
| V1403 | 3 | 0 | 7 | 4 | 3 |
| V1404 | 8 | 9 | 8 | 9 | 4 |
| V1405 | 1 | 0 | 1 | 0 | 5 |
| V1406 | 2 | 0 | 4 | 6 | 6 |
| V1407 | X | X | X | X | |

Table Pointer

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
|---|---|---|---|

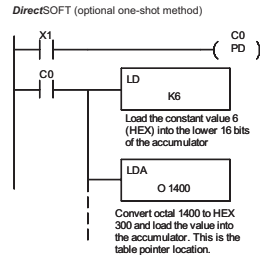
V1400

Destination

| | | | |
|---|---|---|---|
| X | X | X | X |
|---|---|---|---|

V1500

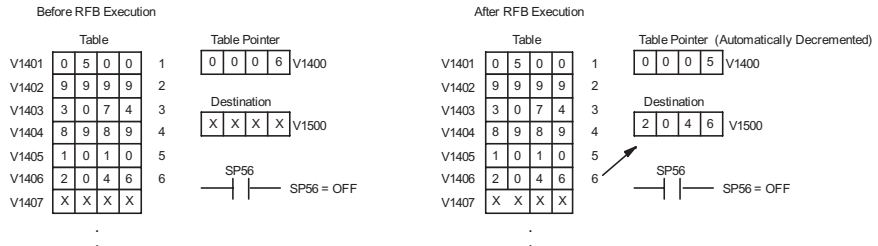
Also, our example uses a normal input contact (X1) to control the execution. Since the CPU scan is extremely fast, and the pointer decrements automatically, the table would cycle through the locations very quickly. If this is a problem for your application, you have an option of using a one-shot (PD) to remove one value each time the input contact transitions from low to high.



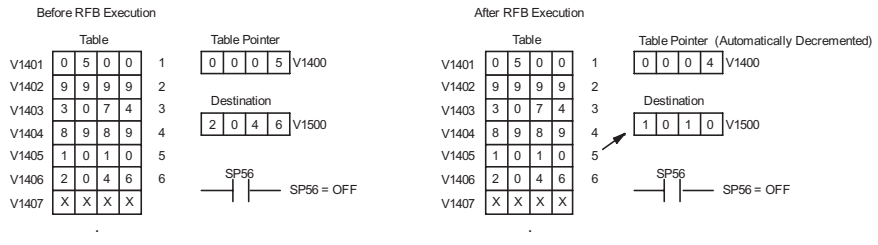
The following diagram shows the scan-by-scan results of the execution for our example program. Notice how the pointer automatically decrements from 6 to 0. Also, notice how SP56 is only on until the end of the scan.

Example of Execution

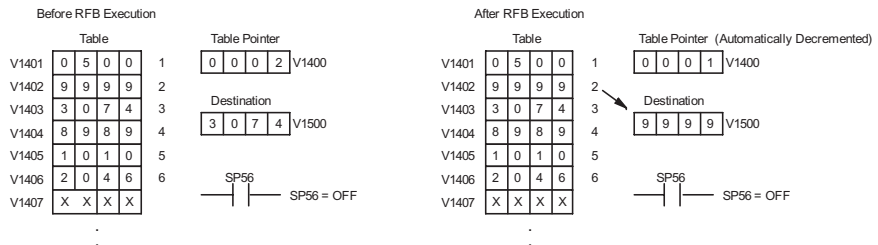
Scan N



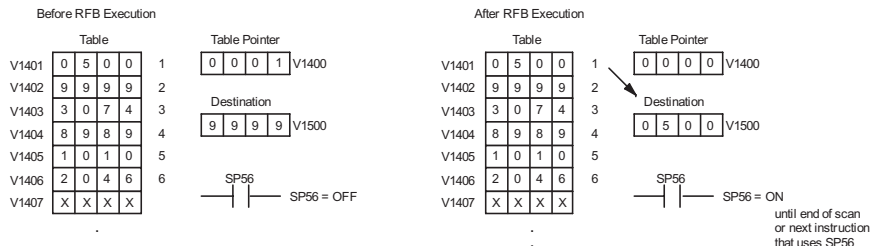
Scan N+1



Scan N+4



Scan N+5



Source to Table (STT)



| | |
|-----|------|
| DS | Used |
| HPP | Used |

The Source To Table instruction moves a value from a V-memory location into a V-memory table and increments a table pointer by 1. When the table pointer reaches the end of the table, it resets to 1. The first V-memory location in the table contains the table pointer which indicates the next location in the table to store a value. The instruction will be executed once per scan provided the input remains on. The function parameters are loaded into the first level of the accumulator stack and the accumulator with two additional instructions. Listed below are the steps necessary to program the Source To Table function.

STT
Vaaa

Step 1: Load the length of the table (number of V-memory locations) into the first level of the accumulator stack. This parameter must be a HEX value, 0 to FF.

Step 2: Load the starting V-memory location for the table into the accumulator. (Remember, the starting location of the table is used as the table pointer.) This parameter must be a HEX value.

Step 3: Insert the STT instruction which specifies the source V-memory location (Vaaa). This is where the value will be moved from.

Helpful hint: For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Helpful hint: The instruction will be executed every scan if the input logic is on. If you do not want the instruction to execute for more than one scan, a one shot (PD) should be used in the input logic.

Helpful hint: The table counter value should be set to indicate the starting point for the operation. Also, it must be set to a value that is within the length of the table. For example, if the table is 6 words long, then the allowable range of values that could be in the pointer should be between 0 and 6. If the value is outside of this range, the data will not be moved. Also, a one shot (PD) should be used so the value will only be set in one scan and will not affect the instruction operation.

| Operand Data Type | DL260 Range |
|-------------------|---------------------|
| | aaa |
| V-memory V | All (See page 3-56) |

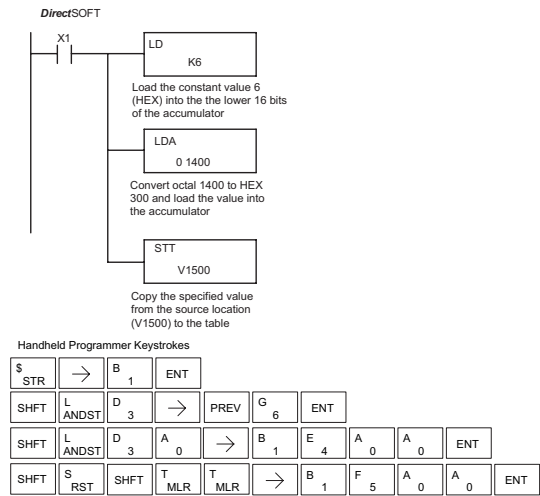
| Discrete Bit Flags | Description |
|--------------------|--|
| SP56 | On when the table pointer equals the table length. |

NOTE: Status flags (SPs) are only valid until:

- another instruction that uses the same flag is executed, or
- the end of the scan

The pointer for this instruction starts at 0 and resets to 1 automatically when the table length is reached.

In the following example, when X1 is on, the constant value (K6) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed in the first stack location after the Load Address instruction is executed. The octal address 1400 (V1400), which is the starting location for the destination table and table pointer, is loaded into the accumulator. The data source location (V1500) is specified in the Source to Table instruction. The table pointer will be increased by “1” after each time the instruction is executed.

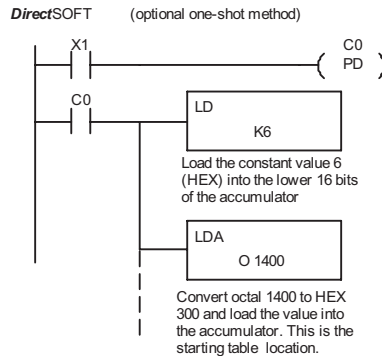


It is important to understand how the table locations are numbered. If you examine the example table, you'll notice that the first data storage location, V1401, will be used when the pointer is equal to 0, and again when the pointer is equal to 6. Why? Because the pointer is only equal to 0 before the very first execution. From then on, it increments from 1 to 6, and then resets to 1.

| Table | | | | | Table Pointer | | | | | |
|-------|---|---|---|---|---------------|---|---|---|---|-------|
| V1401 | X | X | X | X | 0 | 6 | 0 | 0 | 0 | V1400 |
| V1402 | X | X | X | X | 1 | | | | | |
| V1403 | X | X | X | X | 2 | | | | | |
| V1404 | X | X | X | X | 3 | | | | | |
| V1405 | X | X | X | X | 4 | | | | | |
| V1406 | X | X | X | X | 5 | | | | | |
| V1407 | X | X | X | X | | | | | | |
| ⋮ | | | | | | | | | | |

| Data Source | | | | |
|-------------|---|---|---|-------|
| 0 | 5 | 0 | 0 | V1500 |

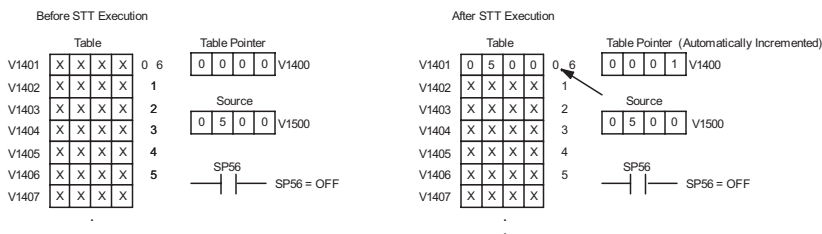
Also, our example uses a normal input contact (X1) to control the execution. Since the CPU scan is extremely fast, and the pointer increments automatically, the source data would be moved into all the table locations very quickly. If this is a problem for your application, you have an option of using a one-shot (PD) to move 1 value each time the input contact transitions from low to high.



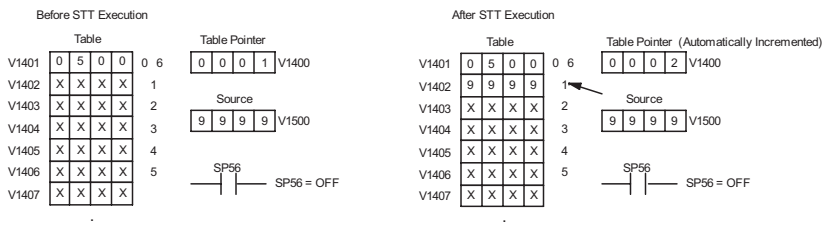
The following diagram shows the scan-by-scan results of the execution for our example program. Notice how the pointer automatically cycles from 0 to 6, and then starts over at 1 instead of 0. Also, notice how SP56 is affected by the execution. Although our example does not show it, we are assuming that there is another part of the program that changes the value in V1500 (data source) prior to the execution of the STT instruction. This is not required, but it makes it easier to see how the data source is copied into the table.

Example of Execution

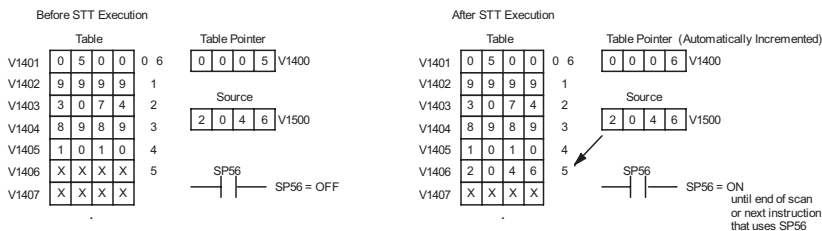
Scan N



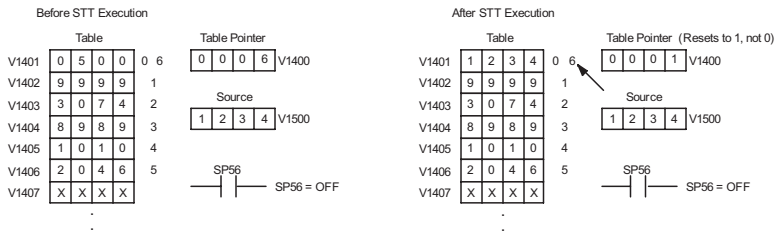
Scan N+1



Scan N+5



Scan N+6



Remove from Table (RFT)

 230
 240
 250-1
 260

| | |
|-----|------|
| DS | Used |
| HPP | Used |

The Remove From Table instruction pops a value off of a table and stores it in a V-memory location. When a value is removed from the table all other values are shifted up 1 location. The first V-memory location in the table contains the table length counter. The table counter decrements by 1 each time the instruction is executed. If the length counter is 0 or greater than the maximum table length (specified in the first level of the accumulator stack), the instruction will not execute and SP56 will be on.

RFT
Vaaa

The instruction will be executed once per scan provided the input remains on. The function parameters are loaded into the first level of the accumulator stack and the accumulator by 2 additional instructions. Listed below are the steps necessary to program the Remove From Table function.

- Step 1: Load the length of the table (number of V-memory locations) into the first level of the accumulator stack. This parameter must be a HEX value, 0 to FF.
- Step 2: Load the starting V-memory location for the table into the accumulator. (Remember, the starting location of the table is used as the table length counter.) This parameter must be a HEX value.
- Step 3: Insert the RFT instructions which specifies destination V-memory location (Vaaa). This is where the value will be moved to.

Helpful hint: For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Helpful hint: The instruction will be executed every scan if the input logic is on. If you do not want the instruction to execute for more than one scan, a one shot (PD) should be used in the input logic.

Helpful hint: The table counter value should be set to indicate the starting point for the operation. Also, it must be set to a value that is within the length of the table. For example, if the table is 6 words long, then the allowable range of values that could be in the table counter should be between 1 and 6. If the value is outside of this range or 0, the data will not be moved from the table. Also, a one shot (PD) should be used so the value will only be set in one scan and will not affect the instruction operation.

| Operand Data Type | | DL260 Range |
|-------------------|---|---------------------|
| | | aaa |
| V-memory | V | All (See page 3-56) |

| Discrete Bit Flags | Description |
|--------------------|-------------------------------------|
| SP56 | On when the table counter equals 0. |

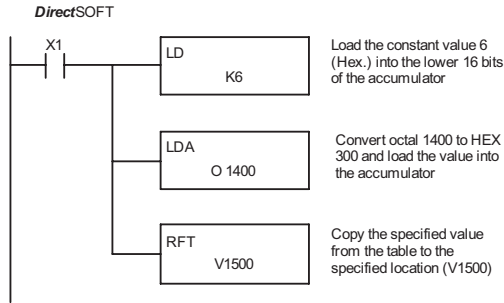
NOTE: Status flags (SPs) are only valid until:

- another instruction that uses the same flag is executed, or
- the end of the scan

The pointer for this instruction can be set to start anywhere in the table. It is not set automatically. You have to load a value into the pointer somewhere in your program.



In the following example, when X1 is on, the constant value (K6) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed in the first stack location after the Load Address instruction is executed. The octal address 1400 (V1400) is the starting location for the source table and is loaded into the accumulator. The destination location (V1500) is specified in the Remove from Table instruction. The table counter will be decreased by “1” after the instruction is executed.



Handheld Programmer Keystrokes

| | | | | | | | | | | |
|-----------|------------|--------|----------|------|--------|--------|--------|--------|-----|--|
| \$ STR | → | B 1 | ENT | | | | | | | |
| SHFT | L ANDST | D 3 | → | PREV | G 6 | ENT | | | | |
| SHFT | L ANDST | D 3 | A 0 | → | B 1 | E 4 | A 0 | A 0 | ENT | |
| SHFT | R ORN | F 5 | T MLR | → | B 1 | F 5 | A 0 | A 0 | ENT | |

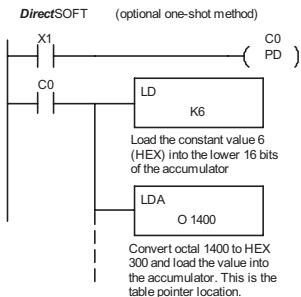
Since the table counter specifies the range of data that will be removed from the table, it is important to understand how the table locations are numbered. If you examine the example table, you'll notice that the data locations are numbered from the top of the table. For example, if the table counter started at 6, then all 6 of the locations would be affected during the instruction execution.

| Table | | | | |
|-------|---|---|---|---|
| V1401 | 0 | 5 | 0 | 1 |
| V1402 | 9 | 9 | 9 | 2 |
| V1403 | 3 | 0 | 7 | 3 |
| V1404 | 8 | 9 | 8 | 4 |
| V1405 | 1 | 0 | 1 | 5 |
| V1406 | 2 | 0 | 4 | 6 |
| V1407 | X | X | X | |
| ⋮ | | | | |

| Table Counter | | | | |
|---------------|---|---|---|-------|
| 0 | 0 | 0 | 6 | V1400 |

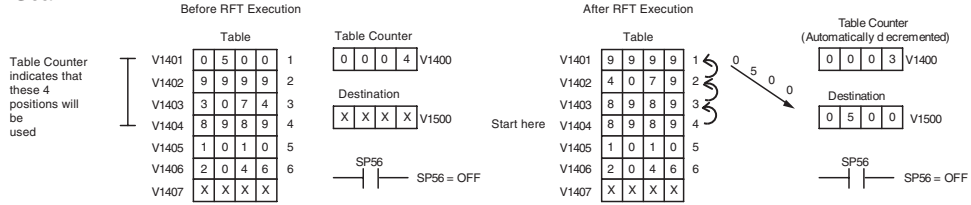
| Destination | | | | |
|-------------|---|---|---|-------|
| X | X | X | X | V1500 |

Also, our example uses a normal input contact (X1) to control the execution. Since the CPU scan is extremely fast, and the pointer decrements automatically, the data would be removed from the table very quickly. If this is a problem for your application, you have the option of using a one-shot (PD) to remove one value each time the input contact transitions from low to high.

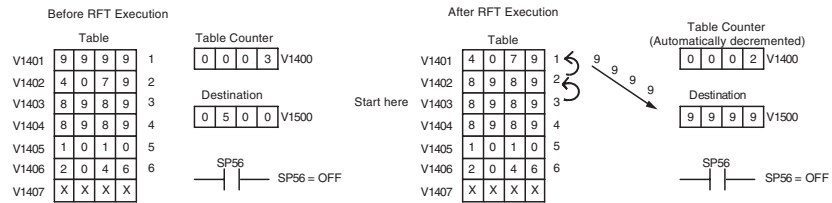


The following diagram shows the scan-by-scan results of the execution for our example program. In our example we're showing the table counter set to 4 initially (Remember, you can set the table counter to any value that is within the range of the table). The table counter automatically decrements from 4 to 0 as the instruction is executed. Notice how the last 2 table positions, 5 and 6, are not moved up through the table. Also, notice how SP56, which comes on when the table counter is 0, is only on until the end of the scan.

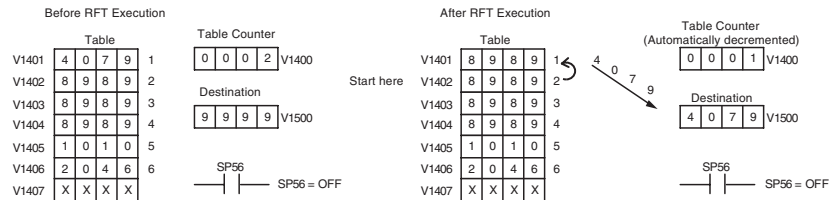
Scan N



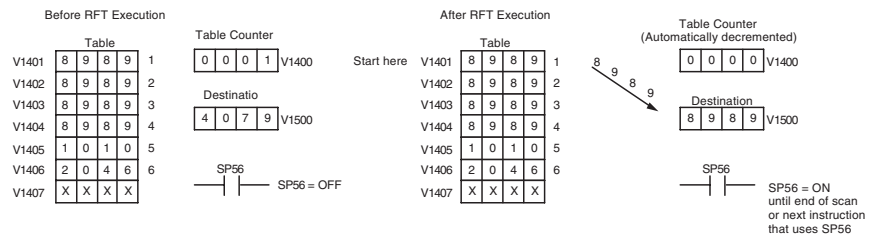
Scan N+1



Scan N+2



Scan N+3



Add to Top (ATT)

✗ 230

✗ 240

✗ 250-1

✓ 260

The Add To Top instruction pushes a value onto a V-memory table from a V-memory location. When the value is added to the table, all other values are pushed down 1 location.

ATT

V aaa

The instruction will be executed once per scan provided the input remains on. The function parameters are loaded into the first level of the accumulator stack and the accumulator by 2 additional instructions. Listed below are the steps necessary to program the Add To Top function.

| | |
|-----|------|
| DS | Used |
| HPP | Used |

Step 1: Load the length of the table (number of V-memory locations) into the first level of the accumulator stack. This parameter must be a HEX value, 0 to FF.

Step 2: Load the starting V-memory location for the table into the accumulator. (Remember, the starting location of the table is used as the table length counter.) This parameter must be a HEX value.

Step 3: Insert the ATT instruction that specifies the source V-memory location (Vaaa). This is where the value will be moved from.

Helpful hint: For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Helpful hint: The instruction will be executed every scan if the input logic is on. If you do not want the instruction to execute for more than one scan, a one shot (PD) should be used in the input logic.

Helpful hint: The table counter value should be set to indicate the starting point for the operation. Also, it must be set to a value that is within the length of the table. For example, if the table is 6 words long, then the allowable range of values that could be in the table counter should be between 1 and 6. If the value is outside of this range or zero, the data will not be moved into the table. Also, a one shot (PD) should be used so the value will only be set in one scan and will not affect the instruction operation.

| Operand Data Type | DL260 Range |
|-------------------|---------------------|
| | aaa |
| V-memory V | All (See page 3-56) |

| Discrete Bit Flags | Description |
|--------------------|-------------------------------------|
| SP56 | On when the table counter equals 0. |

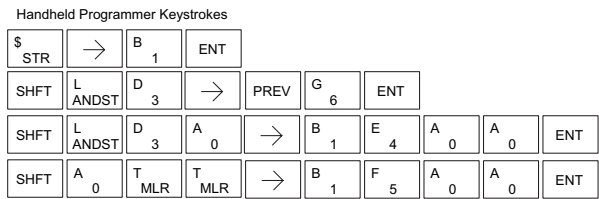
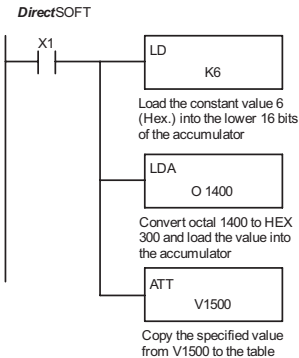
NOTE: Status flags (SPs) are only valid until:

- another instruction that uses the same flag is executed, or
- the end of the scan



The pointer for this instruction can be set to start anywhere in the table. It is not set automatically. You have to load a value into the pointer somewhere in your program.

In the following example, when X1 is on, the constant value (K6) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed in the first stack location after the Load Address instruction is executed. The octal address 1400 (V1400), which is the starting location for the destination table and table counter, is loaded into the accumulator. The source location (V1500) is specified in the Add to Top instruction. The table counter will be increased by “1” after the instruction is executed.



For the ATT instruction, the table counter determines the number of additions that can be made before the instruction will stop executing. So, it is helpful to understand how the system uses this counter to control the execution.

For example, if the table counter was set to 2, and the table length was 6 words, then there could only be 4 additions of data before the execution was stopped. This can be calculated easily by:

$$\text{Table length} - \text{table counter} = \text{number of executions}$$

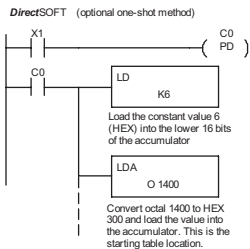
Also, our example uses a normal input contact (X1) to control the execution. Since the CPU scan is extremely fast, and the table counter increments automatically, the data would be moved into the table very quickly. If this is a problem for your application, you have an option of using a one-shot (PD) to add one value each time the input contact transitions from low to high.

| Table | | | | | |
|-------|---|---|---|---|---|
| V1401 | 0 | 5 | 0 | 0 | 1 |
| V1402 | 9 | 9 | 9 | 9 | 2 |
| V1403 | 3 | 0 | 7 | 4 | 3 |
| V1404 | 8 | 9 | 8 | 9 | 4 |
| V1405 | 1 | 0 | 1 | 0 | 5 |
| V1406 | 2 | 0 | 4 | 6 | 6 |
| V1407 | X | X | X | X | |

| Table Counter | | | | | |
|---------------|---|---|---|--|-------|
| 0 | 0 | 0 | 2 | | V1400 |

| Data Source | | | | | |
|-------------|---|---|---|--|-------|
| X | X | X | X | | V1500 |

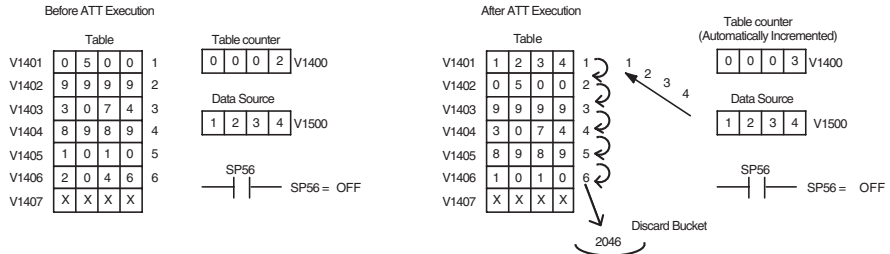
(e.g.: 6 - 2 = 4)



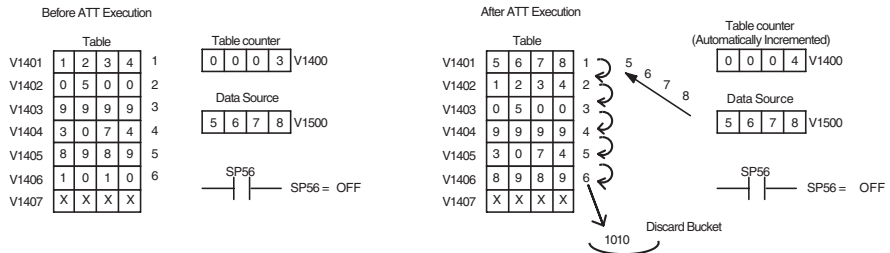
The following diagram shows the scan-by-scan results of the execution for our example program. The table counter is set to 2 initially, and it will automatically increment from 2 to 6 as the instruction is executed. Notice how SP56 comes on when the table counter is 6, which is equal to the table length. Plus, although our example does not show it, we are assuming that there is another part of the program that changes the value in V1500 (data source) prior to the execution of the ATT instruction.

Example of Execution

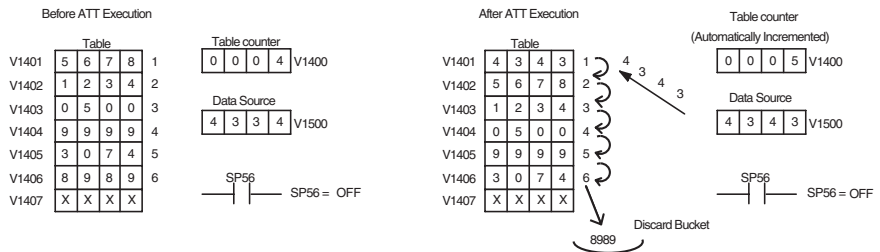
Scan N



Scan N+1



Scan N+2



Scan N+3

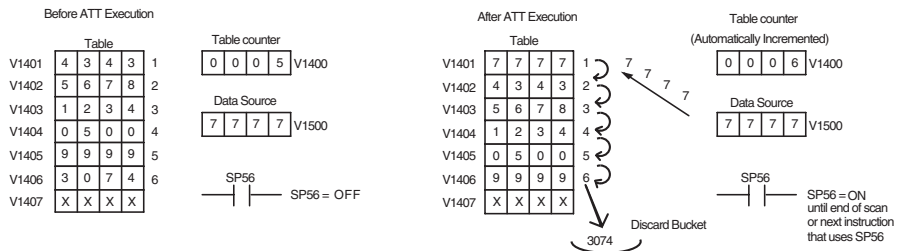


Table Shift Left (TSHFL)

230
240
250-1

The Table Shift Left instruction shifts all the bits in a V-memory table to the left a specified number of bit positions.

TSHFL
Vaaa

260

Table Shift Right (TSHFR)

230
240
250-1

The Table Shift Right instruction shifts all the bits in a V-memory table to the right a specified number of bit positions.

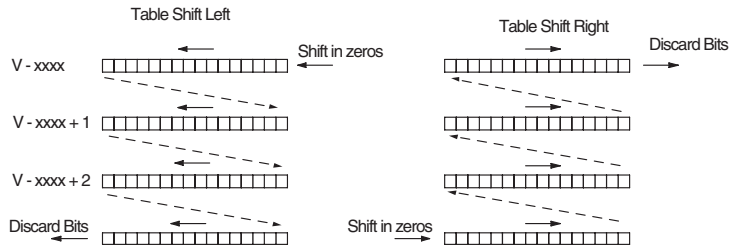
TSHFR
Vaaa

260

The following description applies to both the Table Shift Left and Table Shift Right instructions. A table is a range of V-memory locations. The Table Shift Left and Table Shift Right instructions shift bits serially throughout the entire table. Bits are shifted out the end of one word and into the opposite end of an adjacent word. At the ends of the table, bits are either discarded, or zeros are shifted into the table. The example tables below are arbitrarily four words long.

5

| | |
|-----|------|
| DS | Used |
| HPP | Used |



- Step 1: Load the length of the table (number of V-memory locations) into the first level of the accumulator stack. This parameter must be a HEX value, 0 to FF.
- Step 2: Load the starting V-memory location for the table into the accumulator. This parameter must be a HEX value. You can use the LDA instruction to convert an octal address to hex.
- Step 3: Insert the Table Shift Left or Table Shift Right instruction. This specifies the number of bit positions you wish to shift the entire table. The number of bit positions must be in octal.

Helpful hint: Remember that each V-memory location contains 16 bits. The bits of the first word of the table are numbered from 0 to 17 octal. If you want to shift the entire table by 20 bits, that is 24 octal. Flag 53 will be set if the number of bits to be shifted is larger than the total bits contained within the table. Flag 67 will be set if the last bit shifted (just before it is discarded) is a "1".

| Operand Data Type | | DL260 Range |
|-------------------|---|---------------------|
| | | aaa |
| V-memory | V | All (See page 3-56) |

| Discrete Bit Flags | Description |
|--------------------|---|
| SP53 | On when the number of bits to be shifted is larger than the total bits contained within the table |
| SP67 | On when the last bit shifted (just before it is discarded) is a "1" |



NOTE: Status flags are only valid until:

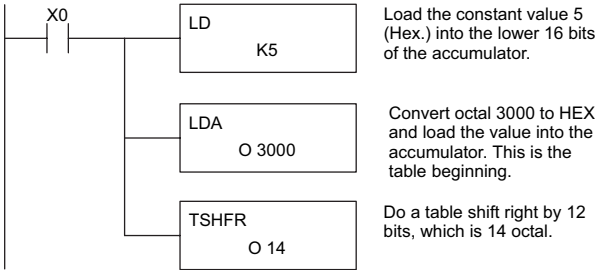
- the end of the scan
- or another instruction that uses the same flag is executed.

The example table to the right contains BCD data as shown (for demonstration purposes). Suppose we want to do a table shift right by 3 BCD digits (12 bits). Converting to octal, 12 bits is 14 octal. Using the Table Shift Right instruction and specifying a shift by octal 14, we have the resulting table shown at the far right. Notice that the 2–3–4 sequence has been discarded, and the 0–0–0 sequence has been shifted in at the bottom.

| V 3000 | V 3000 |
|---------|---------|
| 1 2 3 4 | 6 7 8 1 |
| 5 6 7 8 | 1 2 2 5 |
| 1 1 2 2 | 3 4 4 1 |
| 3 3 4 4 | 5 6 6 3 |
| 5 5 6 6 | 0 0 0 5 |

The following ladder example assumes the data at V3000 to V3004 already exists as shown above. We will use input X0 to trigger the Table Shift Right operation. First, we will load the table length (5 words) into the accumulator stack. Next, we load the starting address into the accumulator. Since V3000 is an octal number we have to convert it to hex by using the LDA command. Finally, we use the Table Shift Right instruction and specify the number of bits to be shifted (12 decimal), which is 14 octal.

DirectSOFT



Handheld Programmer Keystrokes

| | | | | | | | | | | | | | |
|-----------|------------|--------|----------|--------|--------|----------|--------|--------|--------|--------|-----|--|--|
| \$ STR | → | A 0 | ENT | | | | | | | | | | |
| SHFT | L ANDST | D 3 | → | PREV | F 5 | ENT | | | | | | | |
| SHFT | L ANDST | D 3 | A 0 | → | D 3 | A 0 | A 0 | A 0 | ENT | | | | |
| SHFT | T MLR | SHFT | S RST | H 7 | F 5 | R ORN | → | NEXT | B 1 | E 4 | ENT | | |

AND Move (ANDMOV)

- ☐ 230
- ☐ 240
- ☐ 250-1
- ☒ 260

The AND Move instruction copies data from a table to the specified memory location, ANDing each word with the accumulator data as it is written.

ANDMOV
Vaaa

OR Move (ORMOV)

The Or Move instruction copies data from a table to the specified memory location, ORing each word with the accumulator contents as it is written.

ORMOV
Vaaa

Exclusive OR Move (XORMOV)

- ☐ 230
- ☐ 240
- ☐ 250-1
- ☒ 260

The Exclusive OR Move instruction copies data from a table to the specified memory location, XORing each word with the accumulator value as it is written.

XORMOV
Vaaa

The following description applies to the AND Move, OR Move, and Exclusive OR Move instructions. A table is just a range of V-memory locations. These instructions copy the data of a table to another specified location, performing a logical operation on each word with the accumulator contents as the new table is written.

| | |
|-----|------|
| DS | Used |
| HPP | Used |

Step 1: Load the length of the table (number of V-memory locations) into the first level of the accumulator stack. This parameter must be a HEX value, 0 to FF.

Step 2: Load the starting V-memory location for the table into the accumulator. This parameter must be a HEX value. You can use the LDA instruction to convert an octal address to hex.

Step 3: Load the BCD/hex bit pattern into the accumulator which will be logically combined with the table contents as they are copied.

Step 4: Insert the AND Move, OR Move, or XOR Move instruction. This specifies the starting location of the copy of the original table. This new table will automatically be the same length as the original table.

| Operand Data Type | DL260 Range |
|-------------------|-----------------------|
| | aaa |
| V-memory | V All (See page 3-56) |

The example table to the right contains BCD data as shown (for demonstration purposes). Suppose we want to move a table of two words at V3000 and AND it with K6666. The copy of the table at V3100 shows the result of the AND operation for each word.

| | |
|---------|---------|
| V3000 | V3100 |
| 3 3 3 3 | 2 2 2 2 |
| F F F F | 6 6 6 6 |

ANDMOV
K6666
→

The program on the next page performs the ANDMOV operation example above. It assumes that the data in the table at V3000 – V3001 already exists. First we load the table length (two words) into the accumulator. Next we load the starting address of the source table, using the LDA instruction. Then we load the data into the accumulator to be ANDed with the table. In the ANDMOV command, we specify the table destination, V3100.

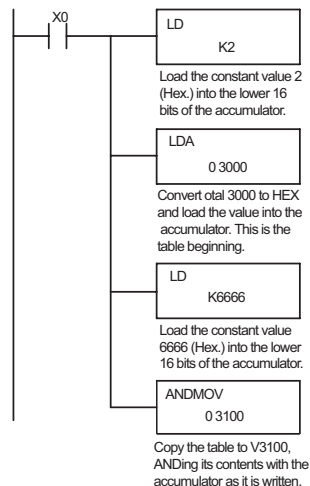
Handheld Programmer Keystrokes

| | | | | | | | | | | | | | | | | | | |
|-------|-----|-------|---|------|-----|-------|---|-----|-----|---|---|---|---|---|-----|---|-----|-----|
| \$ | STR | → | A | 0 | ENT | | | | | | | | | | | | | |
| SHIFT | L | ANDST | D | 3 | → | PREV | C | 2 | ENT | | | | | | | | | |
| SHIFT | L | ANDST | D | 3 | → | A | 0 | → | D | 3 | A | 0 | A | 0 | A | 0 | ENT | |
| SHIFT | L | ANDST | D | 3 | → | PREV | G | 6 | G | 6 | G | 6 | G | 6 | ENT | | | |
| V | AND | SHIFT | M | ORST | O | INST# | V | AND | → | D | 3 | B | 1 | A | 0 | A | 0 | ENT |

The example to the right shows a table of two words at V3000 and logically ORs it with K8888. The copy of the table at V3100 shows the result of the OR operation for each word.

The program to the right performs the ORMOV example above. It assumes that the data in the table at V3000 – V3001 already exists. First we load the table length (two words) into the accumulator. Next we load the starting address of the source table, using the LDA instruction. Then we load the data into the accumulator to be ORed with the table. In the ORMOV command, we specify the table destination, V3100.

DirectSOFT



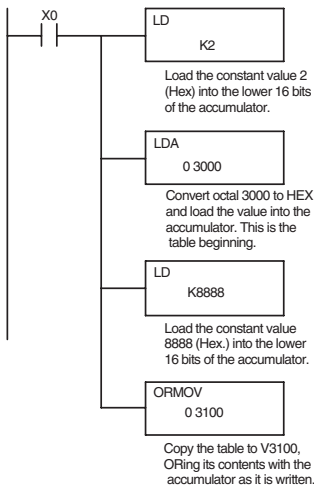
Handheld Programmer Keystrokes

| | | | | | | | | | | | | | | | | | | |
|-------|-----|-------|---|------|-----|-------|---|-----|-----|---|---|---|---|---|-----|---|-----|-----|
| \$ | STR | → | A | 0 | ENT | | | | | | | | | | | | | |
| SHIFT | L | ANDST | D | 3 | → | PREV | C | 2 | ENT | | | | | | | | | |
| SHIFT | L | ANDST | D | 3 | → | A | 0 | → | D | 3 | A | 0 | A | 0 | A | 0 | ENT | |
| SHIFT | L | ANDST | D | 3 | → | PREV | I | 8 | I | 8 | I | 8 | I | 8 | ENT | | | |
| Q | OR | SHIFT | M | ORST | O | INST# | V | AND | → | D | 3 | B | 1 | A | 0 | A | 0 | ENT |

The example to the right shows a table of two words at V3000 and logical XORs it with K3333. The copy of the table at V3100 shows the result of the XOR operation for each word.

The ladder program example for the XORMOV is similar to the one above for the ORMOV. Just use the XORMOV instruction. On the Handheld Programmer, you must use the SHIFT key and spell “XORMOV” explicitly.

DirectSOFT 32



Find Block (FINDB)

- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260

The Find Block instruction searches for an occurrence of a specified block of values in a V-memory table. The function parameters are loaded into the first and second levels of the accumulator stack and the accumulator by three additional instructions. If the block is found, its starting address will be stored in the accumulator. If the block is not found, flag SP53 will be set.

FINDB
Aaaa

| | |
|-----|------|
| DS | Used |
| HPP | N/A |

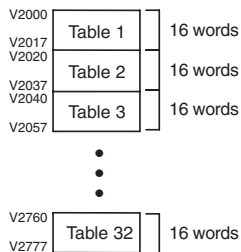
| Operand Data Type | | DL260 Range |
|-------------------|---|-----------------------|
| A | | aaa |
| V-memory | V | All (See page 3 - 56) |
| V-memory | P | All (See page 3 - 56) |

| Discrete Bit Flags | Description |
|--------------------|--|
| SP53 | On when the Find Block instruction was executed but did not find the block of data in table specified. |

The steps below are necessary to program the Find Block function.

- Step 1: Load the number of bytes in the block to be located. This parameter must be a decimal value from 1 to 256.
- Step 2: Load the length of a table (number of words) to be searched. The Find Block will search multiple tables that are adjacent in V-memory. This parameter must be a decimal value from 1 to 128.
- Step 3: Load the ending location for all the tables into the accumulator. This parameter must be a HEX value. You can use the LDA instruction to convert an octal address to hex.
- Step 4: Load the table starting location for all the tables into the accumulator. This parameter must be a HEX value. You can use the LDA instruction to convert an octal address to hex.
- Step 5: Insert the Find Block instruction. This specifies the starting location of the block of data you are trying to locate.

Start Addr.

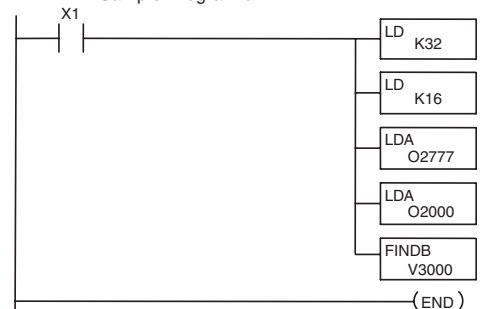


End Addr.

Start Addr.



Sample Program of FINDB

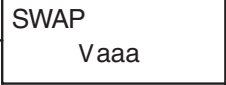


Swap (SWAP)

- 230
- 240
- 250-1
- 260

The Swap instruction exchanges the data in two tables of equal length.

The following steps apply to both the Set Bit and Reset Bit table instructions.



| | |
|-----|------|
| DS | Used |
| HPP | Used |

- Step 1: Load the length of the tables (number of V-memory locations) into the first level of the accumulator stack. This parameter must be a HEX value, 0 to FF. Remember that the tables must be of equal length.
- Step 2: Load the starting V-memory location for the first table into the accumulator. This parameter must be a HEX value. You can use the LDA instruction to convert an octal address to hex.
- Step 3: Insert the Swap instruction. This specifies the starting address of the second table.

Helpful hint: The data swap occurs within a single scan. If the instruction executes on multiple consecutive scans, it will be difficult to know the actual contents of either table at any particular time. So, remember to swap just on a single scan.

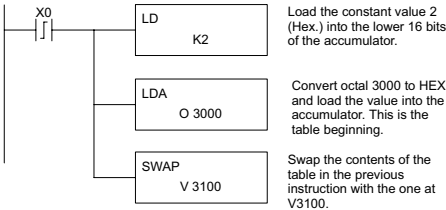
| Operand Data Type | DL260 Range |
|-------------------|-----------------------|
| | aaa |
| V-memory | V All (See page 3-56) |

The example to the right shows a table of two words at V3000. We will swap its contents with another table of two words at V3100 by using the Swap instruction.



The example program below uses a PD contact (triggers for one scan for off-to-on transition). First, we load the length of the tables (two words) into the accumulator. Then we load the address of the first table (V3000) into the accumulator using the LDA instruction, converting the octal address to hex. Note that it does not matter which table we declare “first,” because the swap results will be the same.

DirectSOFT



Handheld Programmer Keystrokes

| | | | | | | |
|------|-------|------|---|------|---|-----|
| \$ | SHFT | P | D | → | A | ENT |
| STR | | CV | 3 | | 0 | |
| SHFT | L | D | → | PREV | C | ENT |
| | ANDST | 3 | | | 2 | |
| SHFT | L | D | A | → | D | A |
| | ANDST | 3 | 0 | | 3 | 0 |
| | | | | | 0 | 0 |
| SHFT | S | SHFT | W | A | P | → |
| | RST | ANDN | 0 | CV | 3 | B |
| | | | | | 1 | A |
| | | | | | | 0 |
| | | | | | | 0 |
| | | | | | | ENT |

Clock/Calendar Instructions

Date (DATE)

- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260

The Date instruction can be used to set the date in the CPU. The instruction requires two consecutive V-memory locations (Vaaa) to set the date. If the values in the specified locations are not valid, the date will not be set. The current date can be read from 4 consecutive V-memory locations (V7771-V7774).

DATE
V aaa

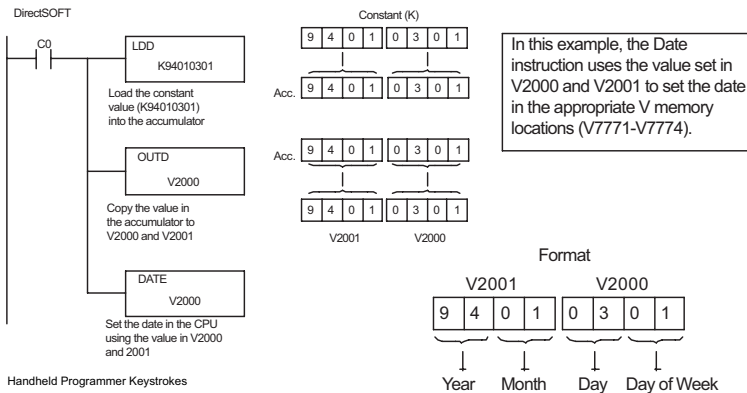
| | |
|-----|------|
| DS | Used |
| HPP | Used |

| Date | Range | V-memory Location (BCD) (READ Only) |
|-------------|-------|-------------------------------------|
| Year | 0-99 | V7774 |
| Month | 1-12 | V7773 |
| Day | 1-31 | V7772 |
| Day of Week | 0-06 | V7771 |

The values entered for the day of week are:
0=Sunday, 1=Monday, 2=Tuesday, 3=Wednesday, 4=Thursday, 5=Friday, 6=Saturday.

| Operand Data Type | DL250-1 Range | DL260 Range |
|-------------------|---------------------|---------------------|
| | aaa | aaa |
| V-memory V | All (See page 3-55) | All (See page 3-56) |

In the following example, when C0 is on, the constant value (K94010301) is loaded into the accumulator using the Load Double instruction (C0 should be a contact from a one shot (PD) instruction). The value in the accumulator is output to V2000 using the Out Double instruction. The Date instruction uses the value in V2000 to set the date in the CPU.



Handheld Programmer Keystrokes

| | | | | | | | | | | | |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|-----|
| \$ | STR | → | NEXT | NEXT | NEXT | NEXT | A ₀ | ENT | | | |
| SHFT | L | ANDST | D ₃ | D ₃ | → | PREV | J ₉ | E ₄ | A ₀ | B ₁ | ENT |
| A ₀ | D ₃ | A ₀ | B ₁ | ENT | | | | | | | |
| GX | OUT | SHFT | D ₃ | → | C ₂ | A ₀ | A ₀ | A ₀ | ENT | | |
| SHFT | D ₃ | A ₀ | T | MIR | E ₄ | → | C ₂ | A ₀ | A ₀ | A ₀ | ENT |

Time (TIME)

- 230
- 240
- 250-1
- 260

The Time instruction can be used to set the time (24-hour clock) in the CPU. The instruction requires two consecutive V-memory locations (Vaaa) which are used to set the time. If the values in the specified locations are not valid, the time will not be set. The current time can be read from memory locations V7747 and V7766–V7770.

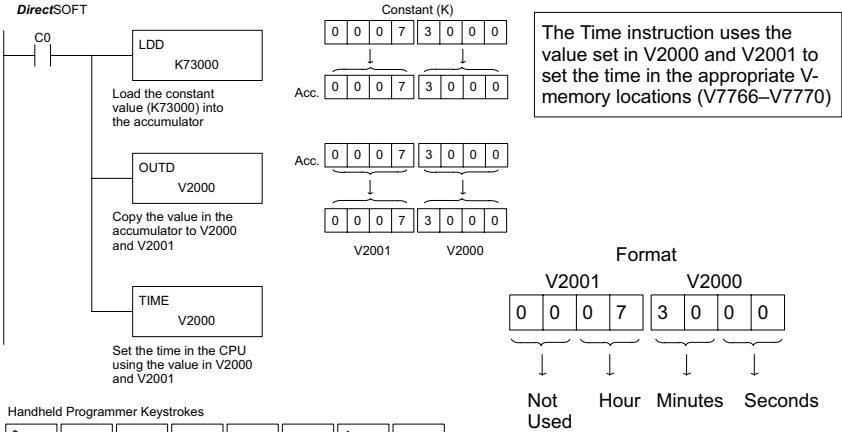
TIME
V aaa

| | |
|-----|------|
| DS | Used |
| HPP | Used |

| Date | Range | V-memory Location (BCD) (READ Only) |
|----------------------|-------|-------------------------------------|
| 1/100 seconds (10ms) | 0-99 | V7747 |
| Seconds | 0-59 | V7766 |
| Minutes | 0-59 | V7767 |
| Hour | 0-23 | V7770 |

| Operand Data Type | DL250-1 Range | DL260 Range |
|-------------------|---------------------|---------------------|
| | aaa | aaa |
| V-memory V | All (See page 3-55) | All (See page 3-56) |

In the following example, when C0 is on, the constant value (K73000) is loaded into the accumulator using the Load Double instruction (C0 should be a contact from a one shot (PD) instruction). The value in the accumulator is output to V2000 using the Out Double instruction. The Time instruction uses the value in V2000 to set the time in the CPU.



Handheld Programmer Keystrokes

| | | | | | | | | | | | | | | | | |
|------|-----|-------|------|------|------|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|--|
| \$ | STR | → | NEXT | NEXT | NEXT | NEXT | A 0 | ENT | | | | | | | | |
| SHFT | L | ANDST | D 3 | D 3 | → | PREV | H 7 | D 3 | A 0 | A 0 | A 0 | A 0 | A 0 | A 0 | ENT | |
| GX | OUT | SHFT | D 3 | → | C 2 | A 0 | A 0 | A 0 | ENT | | | | | | | |
| SHFT | T | MLR | SHFT | I 8 | M | ORST | E 4 | → | C 2 | A 0 | A 0 | A 0 | A 0 | A 0 | ENT | |

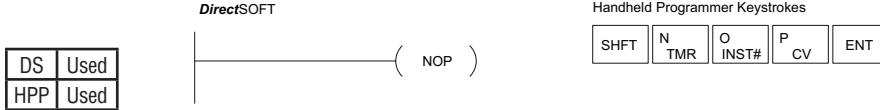
CPU Control Instructions

No Operation (NOP)

The No Operation is an empty (not programmed) memory location.

—(NOP)

- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260

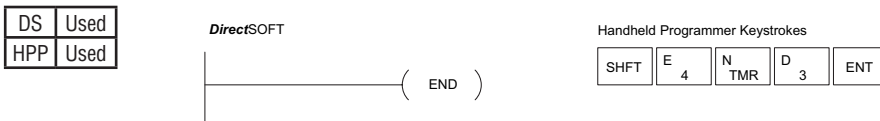


End (END)

The End instruction marks the termination point of the normal program scan. An End instruction is required at the end of the main program body. If the End instruction is omitted, an error will occur and the CPU will not enter the Run Mode. Data labels, subroutines and interrupt routines are placed after the End instruction. The End instruction is not conditional; therefore, no input contact is allowed.

—(END)

- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260



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 a shutdown condition such as an I/O module failure.

—(STOP)

- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260

In the following example, when SP45 comes on indicating an I/O module failure, the CPU will stop operation and switch to the program mode.



Reset Watch Dog Timer (RSTWT)

- ☐ 230
- ☒ 240
- ☒ 250-1
- ☒ 260

The Reset Watch Dog Timer instruction resets the CPU scan timer. The default setting for the watch dog timer is 200ms. Scan times very seldom exceed 200ms, but it is possible. For/next loops, subroutines, interrupt routines, and table instructions can be programmed such that the scan becomes longer than 200ms. When instructions are used in a manner that could exceed the watch dog timer setting, this instruction can be used to reset the timer.

—(RSTWT)

| | |
|-----|------|
| DS | Used |
| HPP | Used |

A software timeout error (E003) will occur and the CPU will enter the program mode if the scan time exceeds the watch dog timer setting. Placement of the RSTWT instruction in the program is very important. The instruction has to be executed before the scan time exceeds the watch dog timer's setting.

If the scan time is consistently longer than the watch dog timer's setting, the timeout value may be permanently increased from the default value of 200ms by AUX 55 on the HPP or the appropriate auxiliary function in your programming package. This eliminates the need for the RSTWT instruction.

In the following example, the CPU scan timer will be reset to 0 when the RSTWT instruction is executed. See the For/Next instruction for a detailed example.

DirectSOFT



Handheld Programmer Keystrokes

| | | | | | | |
|------|----------|----------|----------|-----------|----------|-----|
| SHFT | R ORN | S RST | T MLR | W ANDN | T MLR | ENT |
|------|----------|----------|----------|-----------|----------|-----|

Program Control Instructions

Goto Label (GOTO) (LBL)

☒ 230 The Goto / Label skips all instructions between the Goto and the corresponding LBL instruction. The operand value for the Goto and the corresponding LBL instruction is the same. The logic between Goto and LBL instruction is not executed when the Goto instruction is enabled. Up to 128 Goto instructions and 64 LBL instructions can be used in the program.

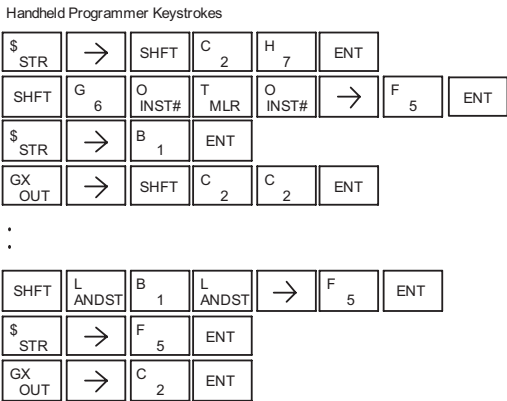
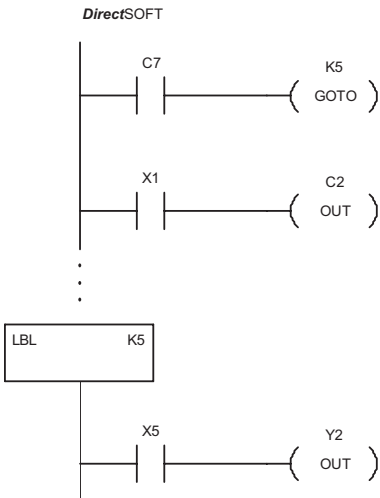
—(K aaa
GOTO)

LBL K aaa

| | |
|-----|------|
| DS | Used |
| HPP | Used |

| Operand Data Type | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|-------------|---------------|-------------|
| | aaa | aaa | aaa |
| Constant K | 1-FFFF | 1-FFFF | 1-FFFF |

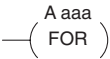
In the following example, when C7 is on, all the program logic between the GOTO and the corresponding LBL instruction (designated with the same constant Kaaa value) will be skipped. The instructions being skipped will not be executed by the CPU.



For/Next (FOR) (NEXT)

- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260

The For and Next instructions are used to execute a section of ladder logic between the For and Next instruction a specified numbers of times. When the For instruction is enabled, the program will loop the specified number of times. If the For instruction is not energized, the section of ladder logic between the For and Next instructions is not executed.



| | |
|-----|------|
| DS | Used |
| HPP | Used |

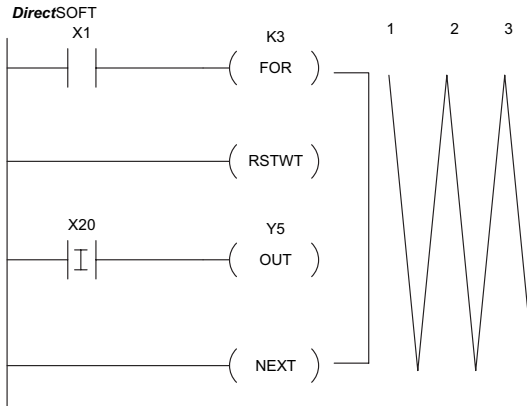
For/Next instructions cannot be nested. Up to 64 For/Next loops may be used in a program. If the maximum number of For/Next loops is exceeded, error E413 will occur.



The normal I/O update and CPU housekeeping is suspended while executing the For/Next loop. The program scan time can increase significantly, depending on the number of times the logic between the For and Next instruction is executed. With the exception of immediate I/O instructions, I/O will not be updated until the program execution is completed for that scan. Depending on the length of time required to complete the program execution, it may be necessary to reset the watchdog timer inside of the For/Next loop using the RSTWT instruction.

| Operand Data Type | | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|---|-----------------------|-----------------------|-----------------------|
| A | | aaa | aaa | aaa |
| V-memory | V | All (See page 3 - 54) | All (See page 3 - 55) | All (See page 3 - 56) |
| Constant | K | 1-9999 | 1-9999 | 1-9999 |

In the following example, when X1 is on, the application program inside the For/Next loop will be executed three times. If X1 is off, the program inside the loop will not be executed. The immediate instructions may or may not be necessary depending on your application. Also, The RSTWT instruction is not necessary if the For/Next loop does not extend the scan time larger the Watchdog Timer setting. For more information on the Watchdog Timer, refer to the RSTWT instruction.



Handheld Programmer Keystrokes

| | | | | | | |
|--------|-------|---------|-------|--------|-------|-----|
| \$ STR | → | B 1 | ENT | | | |
| SHFT | F 5 | O INST# | R ORN | → | D 3 | ENT |
| SHFT | R ORN | S RST | T MLR | W ANDN | T MLR | ENT |
| \$ STR | SHFT | I 8 | → | C 2 | A 0 | ENT |
| GX OUT | → | F 5 | ENT | | | |
| SHFT | N TMR | E 4 | X SET | T MLR | ENT | |

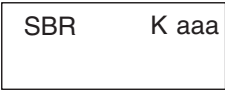
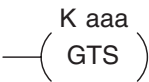
Goto Subroutine (GTS) (SBR)

- 230
- 240
- 250-1
- 260

| | |
|-----|------|
| DS | Used |
| HPP | Used |

The Goto Subroutine instruction allows a section of ladder logic to be placed outside the main body of the program and execute only when needed. There can be a maximum of 128 GTS instructions and 64 SBR instructions used in a program. The GTS instructions can be nested up to 8 levels. An error E412 will occur if the maximum limits are exceeded. Typically this will be used in an application where a block of program logic may be slow to execute and is not required to execute every scan. The subroutine label and all associated logic is placed after the End statement in the program. When the subroutine is called from the main program, the CPU will execute the subroutine (SBR) with the same constant number (K) as the GTS instruction that called the subroutine.

By placing code in a subroutine, it is only scanned and executed when needed since it resides after the End instruction. Code which is not scanned does not impact the overall scan time of the program.



| Operand Data Type | | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|---|-------------|---------------|-------------|
| | | aaa | aaa | aaa |
| Constant | K | 1-FFFF | 1-FFFF | 1-FFFF |

Subroutine Return (RT)

- 230
- 240
- 250-1
- 260

When a Subroutine Return is executed in the subroutine, the CPU will return to the point in the main body of the program from which it was called. The Subroutine Return is used as termination of the subroutine, which must be the last instruction in the subroutine and is a stand-alone instruction (no input contact on the rung).



Subroutine Return Conditional (RTC)

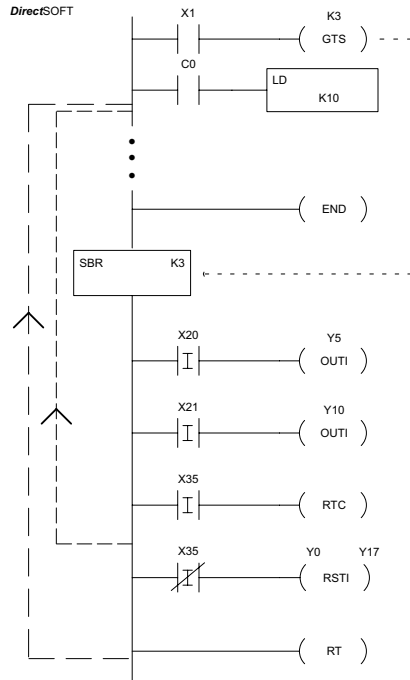
- 230
- 240
- 250-1
- 260

The Subroutine Return Conditional instruction is an optional instruction used with an input contact to implement a conditional return from the subroutine. The Subroutine Return (RT) is still required for termination of the Subroutine.



| | |
|-----|------|
| DS | Used |
| HPP | Used |

In the following example, when X1 is on, Subroutine K3 will be called. The CPU will jump to the Subroutine Label K3, and the ladder logic in the subroutine will be executed. If X35 is on, the CPU will return to the main program at the RTC instruction. If X35 is not on, Y0–Y17 will be reset to off and then the CPU will return to the main body of the program.

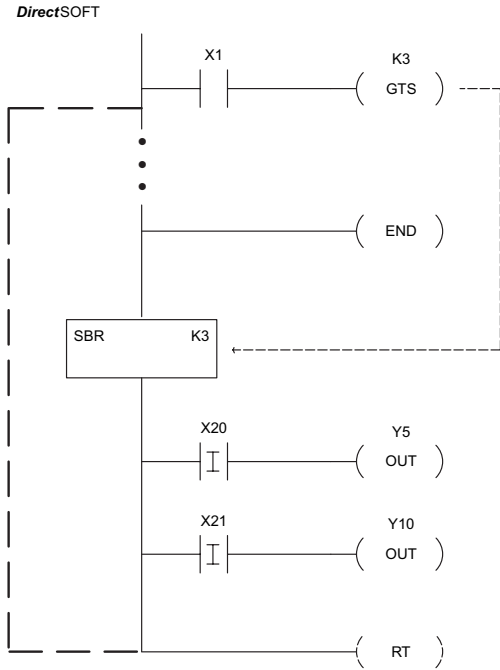


Handheld Programmer Keystrokes

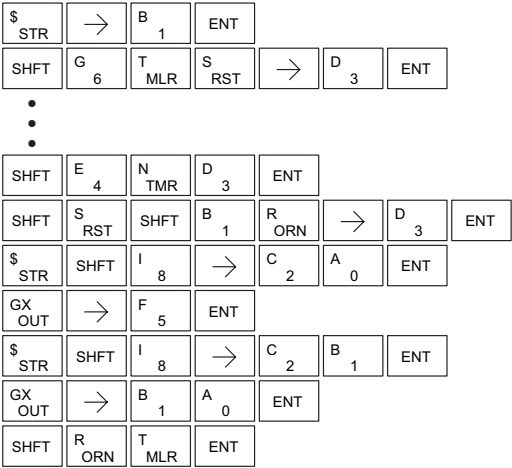
| | | | |
|------|------|------|-----|
| STR | → | 1 | ENT |
| SHFT | G | T | S |
| → | K | 3 | ENT |
| ⋮ | | | |
| SHFT | E | N | D |
| SHFT | S | SHFT | B |
| → | K | 3 | ENT |
| STR | SHFT | I | → |
| OUT | SHFT | I | → |
| STR | SHFT | I | → |
| OUT | SHFT | I | → |
| STR | SHFT | I | → |
| SHFT | R | T | C |
| STRN | SHFT | I | → |
| RST | SHFT | I | → |
| SHFT | R | T | ENT |

In the following example, when X1 is on, Subroutine K3 will be called. The CPU will jump to the Subroutine Label K3 and the ladder logic in the subroutine will be executed. The CPU will return to the main body of the program after the RT instruction is executed.

5



Handheld Programmer Keystrokes



Master Line Set (MLS)

- ✓ 230 The Master Line Set instruction allows the program to control sections of ladder logic by forming a new power rail controlled by the main left power rail. The main left rail is always master line 0. When an MLS K1 instruction is used, a new power rail is created at level 1. Master
- ✓ 240
- ✓ 250-1
- ✓ 260 Line Sets and Master Line Resets can be used to nest power rails up to seven levels deep. Note that unlike stages in RLL^{PLUS}, the logic within the master control relays is still scanned and updated even though it will not function if the MLS is off.

(K aaa
MLS)

| Operand Data Type | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|-------------|-------------|---------------|-------------|
| | aaa | aaa | aaa | aaa |
| Constant K | 1-7 | 1-7 | 1-7 | 1-7 |

- ✓ 230 **Master Line Reset (MLR)**
- ✓ 240 The Master Line Reset instruction marks the end of control for the corresponding MLS instruction. The MLR reference is one less than
- ✓ 250-1 the corresponding MLS.
- ✓ 260

(K aaa
MLR)

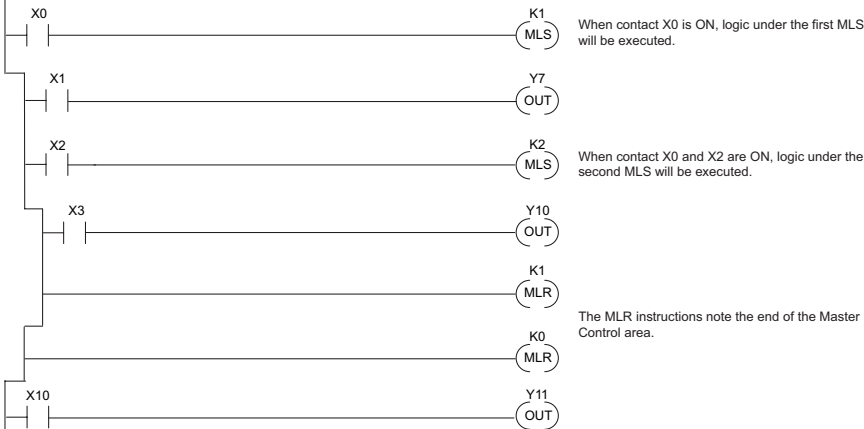
| Operand Data Type | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|-------------|-------------|---------------|-------------|
| | aaa | aaa | aaa | aaa |
| Constant K | 0-6 | 0-6 | 0-6 | 0-6 |

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.

| | |
|-----|------|
| DS | Used |
| HPP | Used |

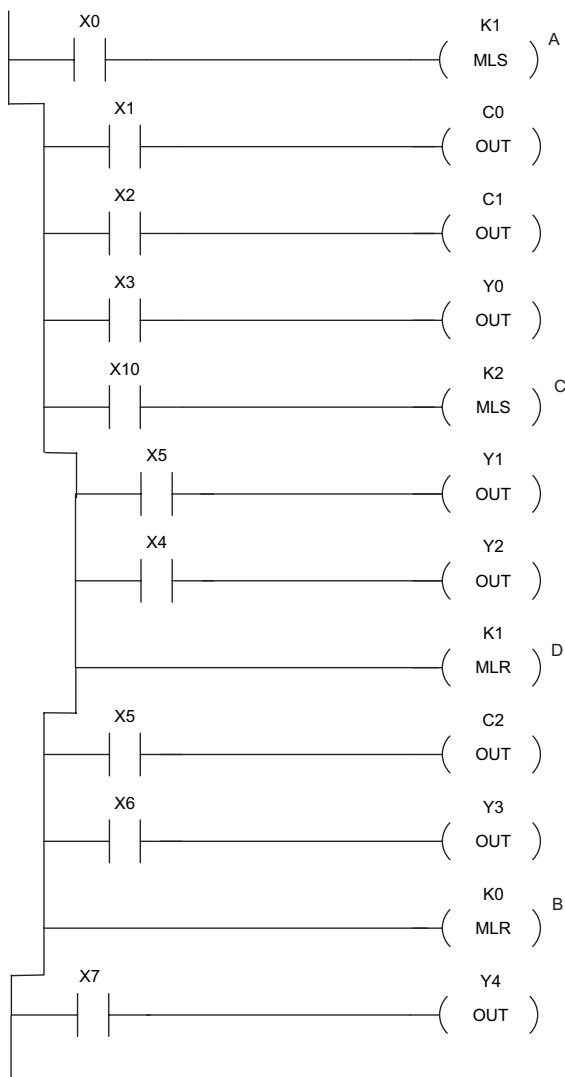
DirectSOFT



MLS/MLR Example

In the following MLS/MLR example, logic between the first MLS K1 (A) and MLR K0 (B) will function only if input X0 is on. The logic between the MLS K2 (C) and MLR K1 (D) will function only if input X10 and X0 is on. The last rung is not controlled by either of the MLS coils.

DirectSOFT



Handheld Programmer Keystrokes

| | | | | | |
|----|-----|---|------|---|-----|
| \$ | STR | → | A | 0 | ENT |
| Y | MLS | → | B | 1 | ENT |
| \$ | STR | → | B | 1 | ENT |
| GX | OUT | → | SHFT | C | 2 |
| | | | A | 0 | ENT |
| \$ | STR | → | C | 2 | ENT |
| GX | OUT | → | SHFT | C | 2 |
| | | | B | 1 | ENT |
| \$ | STR | → | D | 3 | ENT |
| GX | OUT | → | A | 0 | ENT |
| \$ | STR | → | B | 1 | A |
| | | | A | 0 | ENT |
| Y | MLS | → | C | 2 | ENT |
| \$ | STR | → | F | 5 | ENT |
| GX | OUT | → | B | 1 | ENT |
| \$ | STR | → | E | 4 | ENT |
| GX | OUT | → | C | 2 | ENT |
| T | MLR | → | B | 1 | ENT |
| \$ | STR | → | F | 5 | ENT |
| GX | OUT | → | SHFT | C | 2 |
| | | | C | 2 | ENT |
| \$ | STR | → | G | 6 | ENT |
| GX | OUT | → | D | 3 | ENT |
| T | MLR | → | A | 0 | ENT |
| \$ | STR | → | H | 7 | ENT |
| GX | OUT | → | E | 4 | ENT |

Interrupt Instructions

Interrupt (INT)

✗ 230

✓ 240

✓ 250-1

✓ 260

The Interrupt instruction allows a section of ladder logic to be placed outside the main body of the program and executed when needed. Interrupts can be called from the program or by external interrupts via the counter interface module (D2–CTRINT), which provides 4 interrupts.

INT O aaa

| | |
|-----|------|
| DS | Used |
| HPP | Used |

The software interrupt uses interrupt #00 which means the hardware interrupt #0 and the software interrupt cannot be used together.

Typically, interrupts will be used in an application where a fast response to an input is needed or a program section needs to execute faster than the normal CPU scan. The interrupt label and all associated logic must be placed after the End statement in the program. When the interrupt routine is called from the interrupt module or software interrupt, the CPU will complete execution of the instruction it is currently processing in ladder logic, then execute the designated interrupt routine. Interrupt module interrupts are labeled in octal to correspond with the hardware input signal (X1 will initiate interrupt INT1). There is only one software interrupt, and it is labeled INT 0. The program execution will continue from the point it was before the interrupt occurred once the interrupt is serviced.

The software interrupt is set up by programming the interrupt time in V7634. The valid range is 3 to 999 ms. The value must be a BCD value. The interrupt will not execute if the value is out of range.



NOTE: See the example program of a software interrupt.

| Operand Data Type | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|-------------|---------------|-------------|
| | aaa | aaa | aaa |
| Constant 0 | 0-3 | 0-3 | 0-3 |

| DL240/250-1/260 Software | | DL240/250-1/260 Hardware | |
|---------------------------|-------------------|--|-------------------|
| Interrupt Input | Interrupt Routine | Interrupt Input | Interrupt Routine |
| V7634 sets interrupt time | INT 0 | X0 (cannot be used along with s/w interrupt) | INT 0 |
| - | - | X1 | INT 1 |
| - | - | X2 | INT 2 |
| - | - | X3 | INT 3 |

Interrupt Return (IRT)

- ✓

230

✓

240

✓

250-1

✓

260
- When an Interrupt Return is executed in the interrupt routine, the CPU will return to the point in the main body of the program from which it was called. The Interrupt Return is programmed as the last instruction in an interrupt routine and is a stand alone instruction (no input contact on the rung).
- (IRT)

Interrupt Return Conditional (IRTC)

- ✗

230

✗

240

✓

250-1

✓

260
- The Interrupt Return Conditional instruction is a optional instruction used with an input contact to implement a conditional return from the interrupt routine. The Interrupt Return is required to terminate the interrupt routine.
- (IRTC)

Enable Interrupts (ENI)

- ✓

230

✓

240

✓

250-1

✓

260
- The Enable Interrupt instruction is programmed in the main body of the application program (before the End instruction) to enable hardware or software interrupts. Once the coil has been energized, interrupts will be enabled until they are disabled by the Disable Interrupt instruction.
- (ENI)

Disable Interrupts (DISI)

- ✓

230

✓

240

✓

250-1

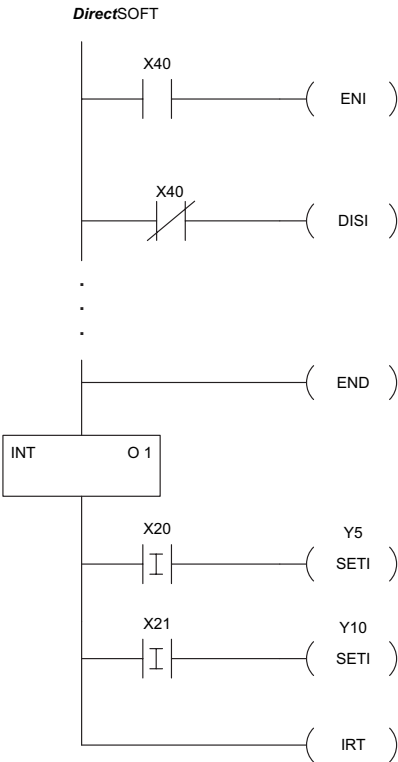
✓

260
- The Disable Interrupt instruction is programmed in the main body of the application program (before the End instruction) to disable both hardware or software interrupts. Once the coil has been energized, interrupts will be disabled until they are enabled by the Enable Interrupt instruction.
- (DISI)

| | |
|-----|------|
| DS | Used |
| HPP | Used |

Interrupt Example for Interrupt Module

In the following example, when X40 is on, the interrupts will be enabled. When X40 is off, the interrupts will be disabled. When an interrupt signal X1 is received, the CPU will jump to the interrupt label INT O 1. The application ladder logic in the interrupt routine will be performed. The CPU will return to the main body of the program after the IRT instruction is executed.



Handheld Programmer Keystrokes

| | | | | | |
|--------|----------------|------------------|------------------|----------------|-----|
| \$STR | → | E ₄ | A ₀ | ENT | |
| SHFT | E ₄ | N _{TMR} | I ₈ | ENT | |
| SPSTRN | → | E ₄ | A ₀ | ENT | |
| SHFT | D ₃ | I ₈ | S _{RST} | I ₈ | ENT |

| | | | | | | |
|-------|------|-------|-------|-----|-----|-----|
| SHFT | E4 | N TMR | D3 | ENT | | |
| SHFT | I8 | N TMR | T MLR | → | B1 | ENT |
| \$STR | SHFT | I8 | → | C2 | A0 | ENT |
| X SET | SHFT | I8 | → | F5 | ENT | |
| \$STR | SHFT | I8 | → | C2 | B1 | ENT |
| X SET | SHFT | I8 | → | B1 | A0 | ENT |
| SHFT | I8 | R ORN | T MLR | ENT | | |

In the following example, when X1 is on, the value 10 is copied to V7634. This value sets the software interrupt to 10ms. When X20 turns on, the interrupt will be enabled. When X20 turns off, the interrupt will be disabled. Every 10ms the CPU will jump to the interrupt label INT 0 0. The application ladder logic in the interrupt routine will be performed. If X35 is not on, Y0–Y17 will be reset to off and then the CPU will return to the main body of the program.



* The value entered, 3-999, must be followed by the digit 4 to complete the instruction.

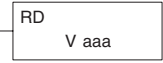


Intelligent I/O Instructions

Read from Intelligent Module (RD)

- ✓ 230
- ✓ 240
- ✓ 250-1
- ✓ 260

The Read from Intelligent Module instruction reads a block of data (1 to 128 bytes maximum) from an intelligent I/O module into the CPU's V-memory. It loads the function parameters into the first and second level of the accumulator stack, and the accumulator by three additional instructions.



Listed below are the steps to program the Read from Intelligent module function.

| | |
|-----|------|
| DS | Used |
| HPP | Used |

- Step 1: Load the base number (0 to 3) into the first byte and the slot number (0 to 7) into the second byte of the second level of the accumulator stack.
- Step 2: Load the number of bytes to be transferred into the first level of the accumulator stack (maximum of 128 bytes).
- Step 3: Load the address from which the data will be read into the accumulator. This parameter must be a HEX value.
- Step 4: Insert the RD instruction that specifies the starting V-memory location (Vaaa) into which the data will be read.

Helpful hint: Use the LDA instruction to convert an octal address to its HEX equivalent and load it into the accumulator when the hex format is required.

| Operand Data Type | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|---------------------|---------------------|---------------------|---------------------|
| | aaa | aaa | aaa | aaa |
| V-memory V | All (See page 3-53) | All (See page 3-54) | All (See page 3-55) | All (See page 3-56) |

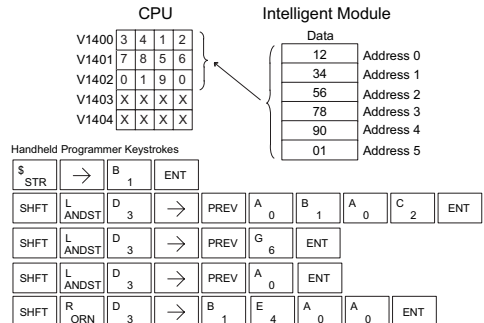
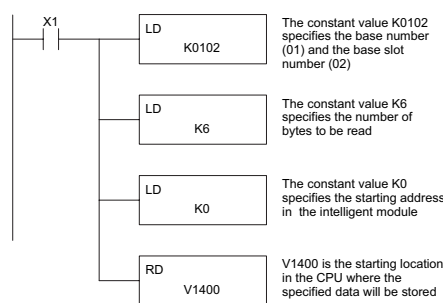
| Discrete Bit Flags | Description |
|--------------------|---|
| SP54 | On when RX, WX, RD, WT instructions are executed with the wrong parameters. |



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

In the following example, when X1 is on, the RD instruction will read six bytes of data from an intelligent module in base 1, slot 2 starting at address 0 in the intelligent module and copy the information into V-memory locations V1400–V1405.

DirectSOFT



Write to Intelligent Module (WT)

✓ 230

✓ 240

✓ 250-1

✓ 260

The Write to Intelligent Module instruction writes a block of data (1 to 128 bytes maximum) to an intelligent I/O module from a block of V-memory in the CPU. The function parameters are loaded into the first and second level of the accumulator stack and the accumulator by three additional instructions. Listed below are the steps necessary to program the Read from Intelligent module function.

WT
V aaa

| | |
|-----|------|
| DS | Used |
| HPP | Used |

- Step 1: Load the base number (0 to 3) into the first byte and the slot number (0 to 7) into the second byte of the second level of the accumulator stack.
- Step 2: Load the number of bytes to be transferred into the first level of the accumulator stack (maximum of 128 bytes).
- Step 3: Load the intelligent module address which will receive the data into the accumulator. This parameter must be a HEX value.
- Step 4: Insert the WT instruction which specifies the starting V-memory location (Vaaa) where the data will be written from in the CPU.

Helpful hint: Use the LDA instruction to convert an octal address to its HEX equivalent and load it into the accumulator when the hex format is required.

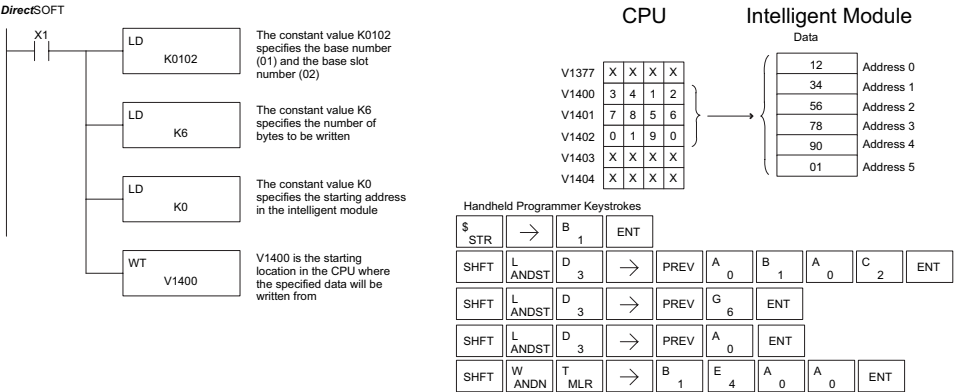
| Operand Data Type | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|-----------------------|---------------------|---------------------|---------------------|
| | aaa | aaa | aaa | aaa |
| V-memory | V All (See page 3-53) | All (See page 3-54) | All (See page 3-55) | All (See page 3-56) |

| Discrete Bit Flags | Description |
|--------------------|---|
| SP54 | On when RX, WX, RD, WT instructions are executed with the wrong parameters. |



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

In the following example, when X1 is on, the WT instruction will write six bytes of data to an intelligent module in base 1, slot 2 starting at address 0 in the intelligent module and copy the information from V-memory locations V1400–V1402.



Network Instructions

Read from Network (RX)

☒ 230

☒ 240

☒ 250-1

☒ 260

The Read from Network instruction is used by the master device on a network to read a block of data from another CPU. The function parameters are loaded into the first and second levels of the accumulator stack and the accumulator by three additional instructions. Listed below are the steps necessary to program the Read from Network function.

RX

A aaa

| | |
|-----|------|
| DS | Used |
| HPP | Used |

Step 1: Load the slave address (0 to 90 BCD) into the first byte, and load the PLC internal port (KF1) or slot number of the master DCM or ECOM (0 to 7) into the second byte of the second level of the accumulator stack.

Step 2: Load the number of bytes (0 to 128 BCD, multiple of 2) to be transferred into the first level of the accumulator stack.

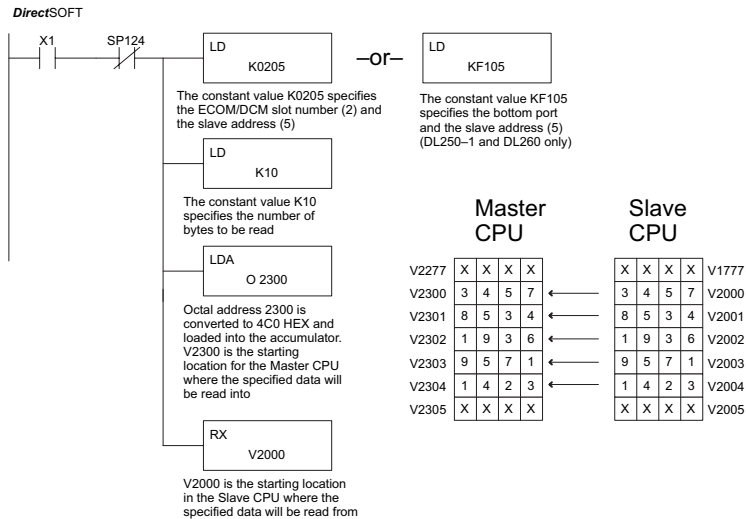
Step 3: Load the address of the data to be read into the accumulator. This parameter requires a HEX value.

Step 4: Insert the RX instruction which specifies the starting V-memory location (Aaaa) where the data will be read from in the slave.

Helpful hint: For parameters that require HEX values, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

| Operand Data Type | | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|-------|-----------------------------------|-----------------------------------|-----------------------------------|
| A | | aaa | aaa | aaa |
| V-memory | V | All (See page 3 - 54) | All (See page 3 - 55) | All (See page 3 - 56) |
| Pointer | P | All V-memory (See page 3 - 54) | All V-memory (See page 3 - 55) | All V-memory (See page 3 - 56) |
| Inputs | X | 0-477 | 0-777 | 0-1777 |
| Outputs | Y | 0-477 | 0-777 | 0-1777 |
| Control Relays | C | 0-377 | 0-1777 | 0-3777 |
| Stage | S | 0-777 | 0-1777 | 0-1777 |
| Timer | T | 0-177 | 0-377 | 0-377 |
| Counter | CT | 0-177 | 0-177 | 0-377 |
| Global I/O | GX/GY | - | - | 0-3777 |
| Special Relay | SP | 0-137 540-617 | 0-777 | 0-777 |

In the following example, when X1 is on and the module busy relay SP124 (see special relays) is not on, the RX instruction will access an ECOM or DCM operating as a master in slot 2. Ten consecutive bytes of data (V2000 – V2004) will be read from a CPU at station address 5 and copied into V-memory locations V2300–V2304 in the CPU with the master DCM or ECOM.



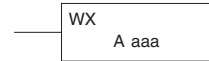
Handheld Programmer Keystrokes

| | | | | | | | | | |
|-----------|------------|----------|------------|--------|----------|--------|--------|--------|-----|
| \$ STR | → | B 1 | ENT | | | | | | |
| W ANDN | → | SHFT | SP STRN | B 1 | C 2 | E 4 | ENT | | |
| SHFT | L ANDST | D 3 | → | SHFT | K JMP | C 2 | A 0 | F 5 | ENT |
| SHFT | L ANDST | D 3 | → | SHFT | K JMP | B 1 | A 0 | ENT | |
| SHFT | L ANDST | D 3 | A 0 | → | C 2 | D 3 | A 0 | A 0 | ENT |
| SHFT | R ORN | X SET | → | C 2 | A 0 | A 0 | A 0 | ENT | |

Write to Network (WX)

- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260

The Write to Network instruction is used to write a block of data from the master device to a slave device on the same network. The function parameters are loaded into the first and second levels of the accumulator stack and the accumulator by three additional instructions. Listed below are the steps necessary to program the Write to Network function.



| | |
|-----|------|
| DS | Used |
| HPP | Used |

Step 1: Load the slave address (0 to 90 BCD) into the first byte and the PLC internal port (KF1) or slot number of the master DCM or ECOM (0 to 7) into the second byte of the second level of the accumulator stack.

Step 2: Load the number of bytes (0 to 128 BCD, multiple of 2) to be transferred into the first level of the accumulator stack.

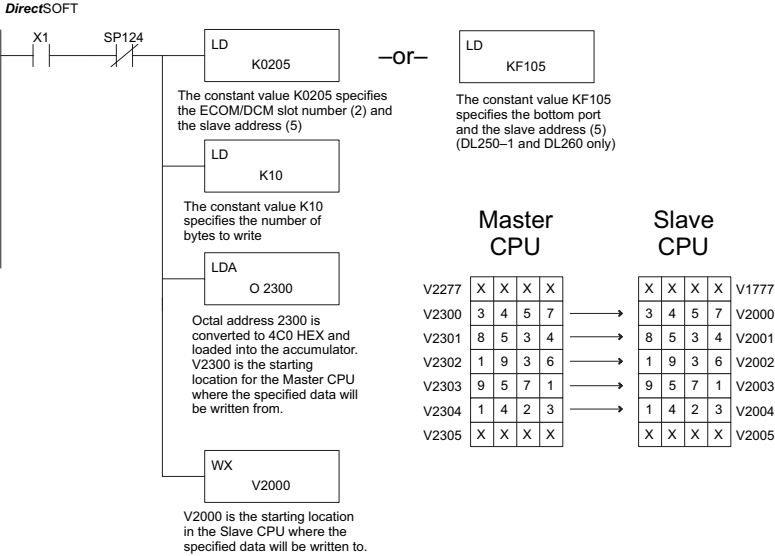
Step 3: Load the address of the data in the master that is to be written to the network into the accumulator. This parameter requires a HEX value.

Step 4: Insert the WX instruction which specifies the starting V-memory location (Aaaa) where the data will be written to the slave.

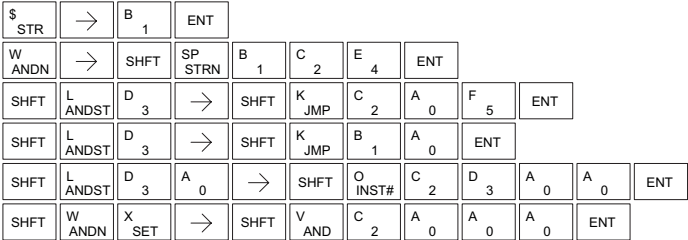
Helpful hint: — For parameters that require HEX values, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

| Operand Data Type | | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|-------|-----------------------------------|-----------------------------------|-----------------------------------|
| A | | aaa | aaa | aaa |
| V-memory | V | All (See page 3 - 54) | All (See page 3 - 55) | All (See page 3 - 56) |
| Pointer | P | All V-memory (See page 3 - 54) | All V-memory (See page 3 - 55) | All V-memory (See page 3 - 56) |
| Inputs | X | 0-477 | 0-777 | 0-1777 |
| Outputs | Y | 0-477 | 0-777 | 0-1777 |
| Control Relays | C | 0-377 | 0-1777 | 0-3777 |
| Stage | S | 0-777 | 0-1777 | 0-1777 |
| Timer | T | 0-177 | 0-377 | 0-377 |
| Counter | CT | 0-177 | 0-177 | 0-377 |
| Global I/O | GX/GY | - | - | 0-3777 |
| Special Relay | SP | 0-137 540-617 | 0-777 | 0-777 |

In the following example when X1 is on and the module busy relay SP124 (see special relays) is not on, the WX instruction will access a DCM or ECOM operating as a master in slot 2. Ten consecutive bytes of data is read from the CPU at station address 5 and copied to V-memory locations V2000–V2004 in the slave CPU.



Handheld Programmer Keystrokes



Message Instructions

Fault (FAULT)

 230

 240

 250-1

 260

The Fault instruction is used to display a message on the handheld programmer or *DirectSOFT*. The message has a maximum of 23 characters and can be either V-memory data, numerical constant data, or ASCII text. See Appendix G for the ASCII Conversion Table.

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.

FAULT
A aaa

| | |
|-----|------|
| DS | Used |
| HPP | Used |

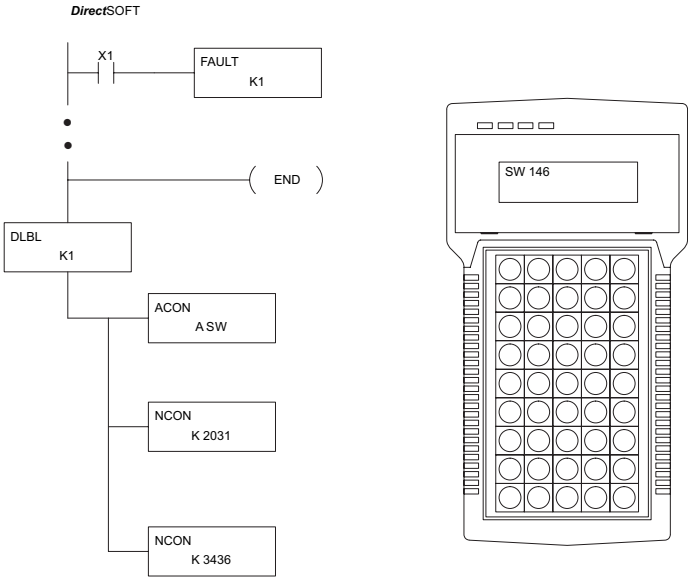
| Operand Data Type | | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|---|---------------------|---------------------|---------------------|
| A | | aaa | aaa | aaa |
| V-memory | V | All (See page 3-54) | All (See page 3-55) | All (See page 3-56) |
| Constant | K | 1-FFFF | 1-FFFF | 1-FFFF |



NOTE: The FAULT instruction takes a considerable amount of time to execute. This is because the FAULT parameters are stored in EEPROM. Make sure you consider the instruction 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

| | | | | | | | | | | | | | | |
|------|-----|-------|-------|-------|-----|-----|------|-----|---|-----|--|--|--|--|
| \$ | → | B | 1 | ENT | | | | | | | | | | |
| SHFT | F | A | U | L | T | → | B | ENT | | | | | | |
| | 5 | 0 | ISG | ANDST | MLR | | 1 | | | | | | | |
| • | | | | | | | | | | | | | | |
| • | | | | | | | | | | | | | | |
| SHFT | E | N | D | ENT | | | | | | | | | | |
| | 4 | TMR | 3 | | | | | | | | | | | |
| SHFT | D | L | B | L | → | B | ENT | | | | | | | |
| | 3 | ANDST | 1 | ANDST | | 1 | | | | | | | | |
| SHFT | A | C | O | N | → | S | W | ENT | | | | | | |
| | 0 | 2 | INST# | TMR | | RST | ANDN | | | | | | | |
| SHFT | N | C | O | N | → | C | A | D | B | ENT | | | | |
| | TMR | 2 | INST# | TMR | | 2 | 0 | 3 | 1 | | | | | |
| SHFT | N | C | O | N | → | D | E | D | G | ENT | | | | |
| | TMR | 2 | INST# | TMR | | 3 | 4 | 3 | 6 | | | | | |

Data Label (DLBL)

- ☒ **230** The Data Label instruction marks the beginning of an ASCII / numeric data area. DLBLs are programmed after the End statement. A maximum of 64 (DL240 and
- ☒ **240** DL250–1/260) or 32 (DL230) DLBL instructions can be
- ☒ **250-1** used in a program. Multiple NCONs and ACONs can be
- ☒ **260** used in a DLBL area.

| |
|---------------|
| DLBL K aaa |
|---------------|

| Operand Data Type | | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|---|-------------|-------------|---------------|-------------|
| | | aaa | aaa | aaa | aaa |
| Constant | K | 1-FFFF | 1-FFFF | 1-FFFF | 1-FFFF |

ASCII Constant (ACON)

- ☒ **230** The ASCII Constant instruction is used with the DLBL
- ☒ **240** instruction to store ASCII text for use with other
- ☒ **250-1** instructions. Two ASCII characters can be stored in an
- ☒ **260** ACON instruction. If only one character is stored in an ACON, a leading space will be printed in the Fault message.

| |
|---------------|
| ACON A aaa |
|---------------|

| Operand Data Type | | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|---|-------------|-------------|---------------|-------------|
| | | aaa | aaa | aaa | aaa |
| ASCII | A | 0-9 A-Z | 0-9 A-Z | 0-9 A-Z | 0-9 A-Z |

Numerical Constant (NCON)

- ☒ **230** The Numerical Constant instruction is used with the
- ☒ **240** DLBL instruction to store the HEX ASCII equivalent of
- ☒ **250-1** numerical data for use with other instructions. Two digits
- ☒ **260** can be stored in an NCON instruction.

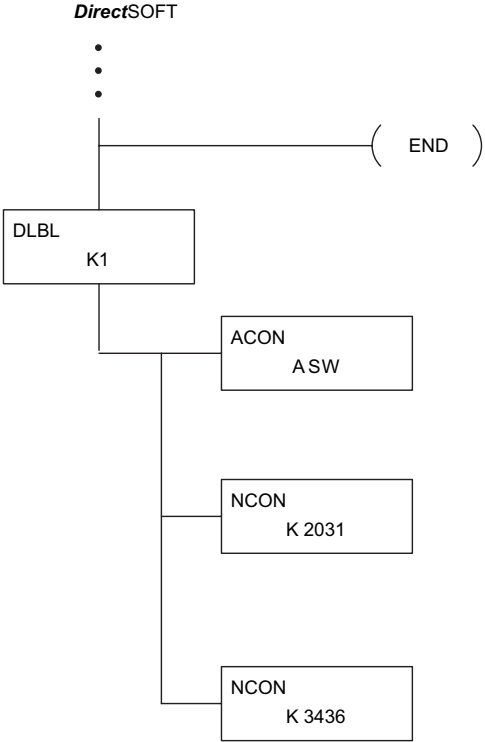
| |
|---------------|
| NCON K aaa |
|---------------|

| Operand Data Type | | DL230 Range | DL240 Range | DL250-1 Range | DL260 Range |
|-------------------|---|-------------|-------------|---------------|-------------|
| | | aaa | aaa | aaa | aaa |
| Constant | K | 0-FFFF | 0-FFFF | 0-FFFF | 0-FFFF |

| | |
|-----|------|
| DS | Used |
| HPP | Used |

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.



Handheld Programmer Keystrokes

| | | | | | | | | | | | | | | | | | | | |
|------|----------|------------|------------|------------|---|----------|-----------|--------|--------|-----|--|--|--|--|--|--|--|--|--|
| SHFT | E 4 | N TMR | D 3 | ENT | | | | | | | | | | | | | | | |
| SHFT | D 3 | L ANDST | B 1 | L ANDST | → | B 1 | ENT | | | | | | | | | | | | |
| SHFT | A 0 | C 2 | O INST# | N TMR | → | S RST | W ANDN | ENT | | | | | | | | | | | |
| SHFT | N TMR | C 2 | O INST# | N TMR | → | C 2 | A 0 | D 3 | B 1 | ENT | | | | | | | | | |
| SHFT | N TMR | C 2 | O INST# | N TMR | → | D 3 | E 4 | D 3 | G 6 | ENT | | | | | | | | | |

Print Message (PRINT)

-  230
-  240
-  250-1
-  260

The Print Message instruction prints the embedded text or text/data variable message to the specified communications port (2 on the DL250-1/260 CPU), which must have the communications port configured.

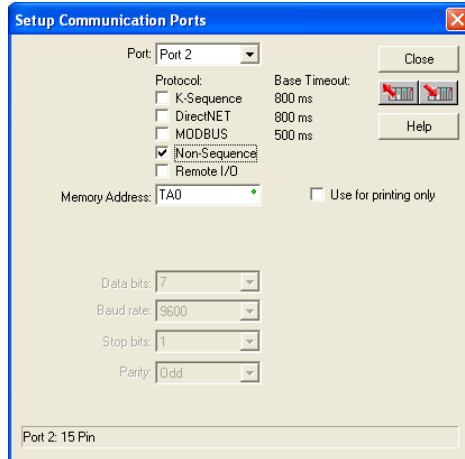
PRINT A aaa
"Hello, this is a PLC message"

| | |
|-----|------|
| DS | Used |
| HPP | N/A |

| Data Type | DL250-1 Range | DL260 Range |
|-----------|---------------|-------------|
| A | aaa | aaa |
| Constant | K | 2 |

You may recall from the CPU specifications in Chapter 3 that the DL250-1 and DL260 ports are capable of several protocols. To configure a port using the Handheld Programmer, use AUX 56 and follow the prompts, making the same choices as indicated below on this page. To configure a port in *DirectSOFT*, choose the PLC menu, then Setup, then Setup Secondary Comm Port.

- **Port:** From the port number list box at the top, choose "Port 2."
- **Protocol:** Click the check box to the left of "Non-sequence." The Setup Communication Ports dialog box opens.



The dialog box titled "Setup Communication Ports" has a blue title bar. It contains the following fields and controls:

- Port:** A dropdown menu set to "Port 2".
- Protocol:** A group box containing four checkboxes: "K-Sequence" (unchecked), "DirectNET" (unchecked), "MODBUS" (unchecked), and "Non-Sequence" (checked). Below these is a checkbox for "Remote I/O" (unchecked).
- Base Timeout:** A label with three values: "800 ms", "800 ms", and "500 ms".
- Memory Address:** A text field containing "TA0".
- Use for printing only:** An unchecked checkbox.
- Data bits:** A dropdown menu set to "7".
- Baud rate:** A dropdown menu set to "9600".
- Stop bits:** A dropdown menu set to "1".
- Parity:** A dropdown menu set to "Odd".
- Buttons:** "Close" (top right), "Help" (middle right), and a red arrow button (bottom right).
- Status bar:** At the bottom, it says "Port 2: 15 Pin".

- **Memory Address:** Choose a V-memory address for *DirectSOFT* to use to store the port setup information. You will need to reserve 66 contiguous words in V-memory for this purpose. Select "Use for printing only" if it applies.
- **Baud Rate:** Choose the baud rate that matches your printer.
- **Stop Bits, Parity:** Choose number of stop bits and parity setting to match your printer.

Then click the button indicated to send the Port 2 configuration to the CPU, and click Close. See Chapter 3 for port wiring information to connect your printer to the DL250-1/260.



Port 2 on the DL250–1/260 has standard RS232 levels, and should work with most printer serial input connections.

Text element - used for printing character strings. The character strings are defined as the character (more than 0) ranged by the double quotation marks. Two hex numbers preceded by the dollar sign means an 8-bit ASCII character code. Also, two characters preceded by the dollar sign is interpreted according to the following table:

| # | Character code | Description |
|---|----------------|----------------------------------|
| 1 | \$\$ | Dollar sign (\$) |
| 2 | \$" | Double quotation (") |
| 3 | \$L or \$1 | Line feed (LF) |
| 4 | \$N or \$n | Carriage return line feed (CRLF) |
| 5 | \$P or \$p | Form feed |
| 6 | \$R or \$r | Carriage return (CR) |
| 7 | \$T or \$t | Tab |

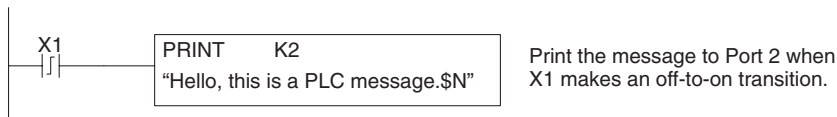
The following examples show various syntax conventions and the length of the output to the printer.

Example:

| | |
|-------------------|-------------------------------------|
| " " | Length 0 without character |
| "A" | Length 1 with character A |
| " " | Length 1 with blank |
| " \$ " " | Length 1 with double quotation mark |
| " \$ R \$ L " | Length 2 with one CR and one LF |
| " \$ 0 D \$ 0 A " | Length 2 with one CR and one LF |
| " \$ \$ " | Length 1 with one \$ mark |

In printing an ordinary line of text, you will need to include **double quotation** marks before and after the text string. Error code 499 will occur in the CPU when the print instruction contains invalid text or no quotations. It is important to test your PRINT instruction data during the application development.

The following example prints the message to port 2. We use a PD contact, which causes the message instruction to be active for just one scan. Note the \$N at the end of the message, which produces a carriage return / line feed on the printer. This prepares the printer to print the next line, starting from the left margin.



V-memory element – used for printing V-memory contents in the integer format or real format. Use V-memory number or V-memory number with “:” and data type. The data types are shown in the table below. The Character code must be capital letters.



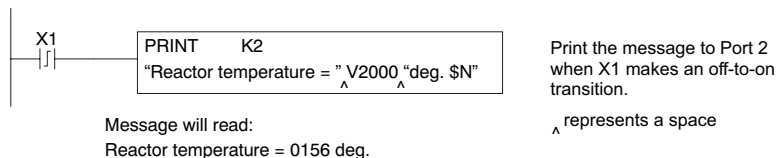
NOTE: There must be a space entered before and after the V-memory address to separate it from the text string. Failure to do this will result in an error code 499.

| # | Character code | Description |
|---|----------------|---|
| 1 | none | 16-bit binary (decimal number) |
| 2 | : B | 4-digit BCD |
| 3 | : D | 32-bit binary (decimal number) |
| 4 | : D B | 8-digit BCD |
| 5 | : R | Floating point number (real number) |
| 6 | : E | Floating point number (real number with exponent) |

Example:

| | |
|-------------|---|
| V2000 | Print binary data in V2000 for decimal number |
| V2000 : B | Print BCD data in V2000 |
| V2000 : D | Print binary number in V2000 and V2001 for decimal number |
| V2000 : D B | Print BCD data in V2000 and V2001 |
| V2000 : R | Print floating point number in V2000/V2001 as real number |
| V2000 : E | Print floating point number in V2000/V2001 as real number with exponent |

Example: The following example prints a message containing text and a variable. The “reactor temperature” labels the data, which is at V2000. You can use the ‘: B’ qualifier after the V2000 if the data is in BCD format, for example. The final string adds the units of degrees to the line of text, and the \$N adds a carriage return / line feed.



V-memory text element – used for printing text stored in V-memory. Use the % followed by the number of characters after V-memory number for representing the text. If you assign “0” as the number of characters, the print function will read the character count from the first location. Then it will start at the next V-memory location and read that number of ASCII codes for the text from memory.

Example:

| | |
|------------|---|
| V2000 % 16 | 16 characters in V2000 to V2007 are printed. |
| V2000 % 0 | The characters in V2001 to Vxxxx (determined by the number in V2000) will be printed. |

Bit element – used for printing the state of the designated bit in V-memory or a relay bit. The bit element can be assigned by the designating point (.) and bit number preceded by the V-memory number or relay number. The output type is described as shown in the table below.

| # | Data format | Description |
|---|-------------|--|
| 1 | none | Print 1 for an ON state, and 0 for an OFF state |
| 2 | : BOOL | Print “TRUE” for an ON state, and “FALSE” for an OFF state |
| 3 | : ONOFF | Print “ON” for an ON state, and “OFF” for an OFF state |

Example:

V2000.15 Prints the status of bit 15 in V2000, in 1/0 format
 C100 Prints the status of C100 in 1/0 format
 C100 : BOOL Prints the status of C100 in TRUE/FALSE format
 C100 : ON/OFF Prints the status of C100 in ON/OFF format
 V2000.15 : BOOL Prints the status of bit 15 in V2000 in TRUE/FALSE format

The maximum numbers of characters you can print is 128. The number of characters for each element is listed in the table below:

| Element type | Maximum Characters |
|-------------------------------------|--------------------|
| Text, 1 character | 1 |
| 16-bit binary | 6 |
| 32-bit binary | 11 |
| 4-digit BCD | 4 |
| 8-digit BCD | 8 |
| Floating point (real number) | 13 |
| Floating point (real with exponent) | 13 |
| V-memory/text | 2 |
| Bit (1/0 format) | 1 |
| Bit (TRUE/FALSE format) | 5 |
| Bit (ON/OFF format) | 3 |

The Handheld Programmer’s mnemonic is “PRINT,” followed by the DEF field.

Special relay flags SP116 and SP117 indicate the status of the DL250–1/260 CPU ports (busy, or communications error). See the appendix on special relays for a description.



NOTE: You must use the appropriate special relay in conjunction with the PRINT command to ensure the ladder program does not try to PRINT to a port that is still busy from a previous PRINT or WX or RX instruction.

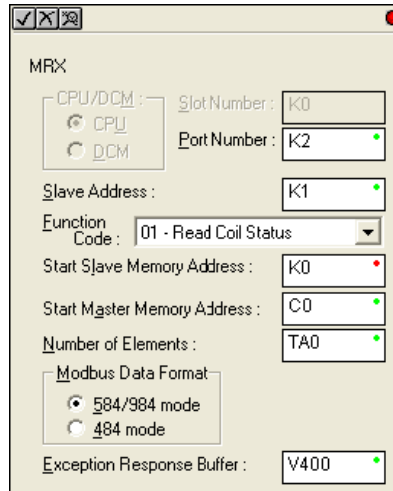
Modbus RTU Instructions (DL260)

Modbus Read from Network (MRX)

The Modbus Read from Network (MRX) instruction is used by the DL260 network master to read a block of data from a connected slave device and to write the data into V-memory addresses within the master. The instruction allows the user to specify the Modbus Function Code, slave station address, starting master and slave memory addresses, number of elements to transfer, Modbus data format and the Exception Response Buffer.

| | |
|---|-------|
|  | 230 |
|  | 240 |
|  | 250-1 |
|  | 260 |

| | |
|-----|------|
| DS | Used |
| HPP | N/A |



- **Port Number:** must be DL260 Port 2 (K2)
- **Slave Address:** specify a slave station address (1 to 247)
- **Function Code:** The following Modbus function codes are supported by the MRX instruction:
 - 01 – Read a group of coils
 - 02 – Read a group of inputs
 - 03 – Read holding registers
 - 04 – Read input registers
 - 07 – Read Exception status
- **Start Slave Memory Address:** specifies the starting slave memory address of the data to be read. See the table on the following page.
- **Start Master Memory Address:** specifies the starting memory address in the master where the data will be placed. See the table on the following page.
- **Number of Elements:** specifies how many coils, inputs, holding registers or input registers will be read. See the table on the following page.
- **Modbus Data Format:** specifies Modbus 584/984 or 484 data format to be used.

- **Exception Response Buffer:** specifies the master memory address where the Exception Response will be placed (6 bytes in length). See the table on the following page. The exception response buffer uses 3 words. These bytes are swapped in the MRX/MWX exception response buffer V-memory so:

V-Memory 1 Hi Byte = Function Code Byte (Most Significant Bit Set)

V-Memory 1 Lo Byte = Address Byte

V-Memory 2 Hi Byte = One of the CRC Bytes

V-Memory 2 Lo Byte = Exception Code

V-Memory 3 Hi Byte = 0

V-Memory 3 Lo Byte = Other CRC Byte

MRX Slave Memory Address

| MRX Slave Address Ranges | | |
|--------------------------|----------------------|---|
| Function Code | Modbus Data Format | Slave Address Range(s) |
| 01-Read Coil | 484 Mode | 1-999 |
| 01-Read Coil | 584/984 Mode | 1-65535 |
| 02-Read Input Status | 484 Mode | 1001-1999 |
| 02-Read Input Status | 584/984 Mode | 10001-19999 (5 digit) or 100001-165535 (6 digit) |
| 03-Read Holding Register | 484 Mode | 4001-4999 |
| 03-Read Holding Register | 584/984 | 40001-49999 9 (5 digit) or 4000001-465535 (6 digit) |
| 04-Read Input Register | 484 Mode | 3001-3999 |
| 04-Read Input Register | 584/984 Mode | 30001-39999 (5 digit) or 3000001-365535 (6 digit) |
| 07-Read Exception Status | 484 and 584/984 Mode | n/a |

MRX Master Memory Addresses

| MRX Master Memory Address Ranges | | |
|----------------------------------|----|---------------------|
| Operand Data Type | | DL260 Range |
| Inputs | X | 0-1777 |
| Outputs | Y | 0-1777 |
| Control Relays | C | 0-3777 |
| Stage Bits | S | 0-1777 |
| Timer Bits | T | 0-377 |
| Counter Bits | CT | 0-377 |
| Special Relays | SP | 0-777 |
| V-memory | V | all (see page 3-56) |
| Global Inputs | GX | 0-3777 |
| Global Outputs | GY | 0-3777 |

MRX Number of Elements

| Number of Elements | | |
|--------------------|---|----------------------------------|
| Operand Data Type | | DL260 Range |
| V-memory | V | All (see page 3-56) |
| Constant | K | Bits: 1-2000 Registers: 1-125 |

MRX Exception Response Buffer

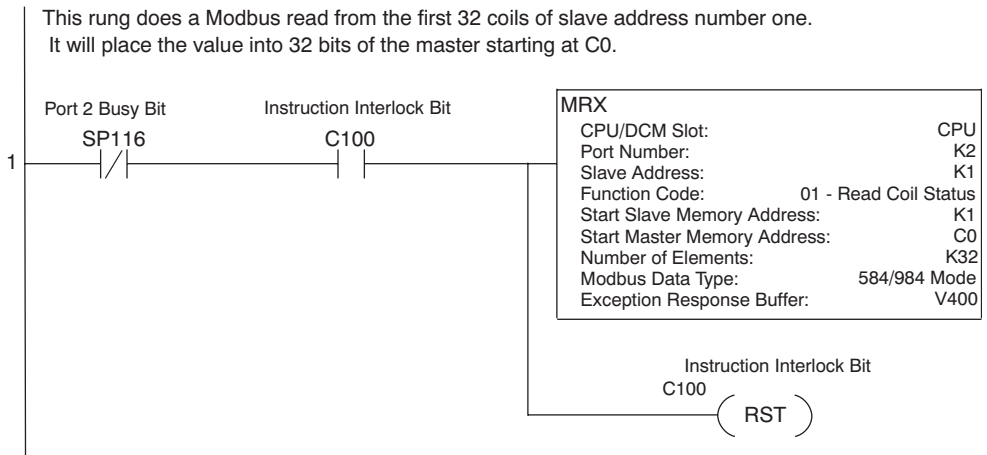
| Exception Response Buffer | | |
|---------------------------|---|---------------------|
| Operand Data Type | | DL260 Range |
| V-memory | V | All (see page 3-56) |

5

MRX Example

DL260 port 2 has two Special Relay contacts associated with it (see Appendix D for comm port special relays). One indicates “Port busy” (SP116), and the other indicates “Port Communication Error” (SP117). The “Port Busy” bit is on while the PLC communicates with the slave. When the bit is off, the program can initiate the next network request. The “Port Communication Error” bit turns on when the PLC has detected an error. Use of this bit is optional. When used, it should be ahead of any network instruction boxes since the error bit is reset when an MRX or MWX instruction is executed.

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

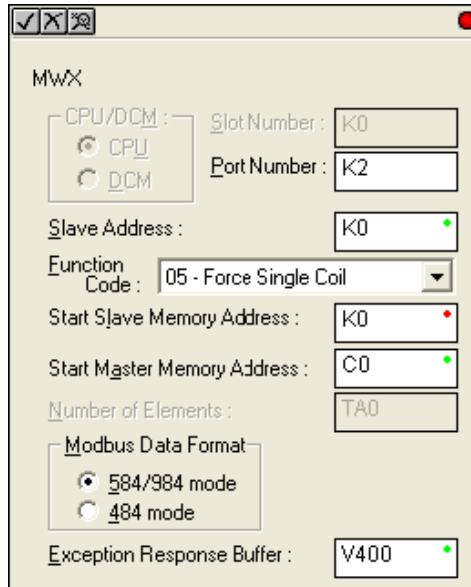


Modbus Write to Network (MWX)

| | |
|-------------------------------------|-------|
| <input type="checkbox"/> | 230 |
| <input type="checkbox"/> | 240 |
| <input type="checkbox"/> | 250-1 |
| <input checked="" type="checkbox"/> | 260 |

The Modbus Write to Network (MWX) instruction is used to write a block of data from the network master's (DL260) memory to Modbus memory addresses within a slave device on the network. The instruction allows the user to specify the Modbus Function Code, slave station address, starting master and slave memory addresses, number of elements to transfer, Modbus data format and the Exception Response Buffer.

| | |
|-----|------|
| DS | Used |
| HPP | N/A |



The MWX instruction configuration window is shown. It contains the following fields and options:

- CPU/DCM:** A group box containing two radio buttons: **CPU** (selected) and **DCM**.
- Slot Number:** A text box containing the value **K0**.
- Port Number:** A text box containing the value **K2**.
- Slave Address:** A text box containing the value **K0**, with a green checkmark icon to its right.
- Function Code:** A dropdown menu showing **05 - Force Single Coil**.
- Start Slave Memory Address:** A text box containing the value **K0**, with a red dot icon to its right.
- Start Master Memory Address:** A text box containing the value **C0**, with a green checkmark icon to its right.
- Number of Elements:** A text box containing the value **TA0**.
- Modbus Data Format:** A group box containing two radio buttons: **584/984 mode** (selected) and **484 mode**.
- Exception Response Buffer:** A text box containing the value **V400**, with a green checkmark icon to its right.

- **Port Number:** must be DL260 Port 2 (K2)
- **Slave Address:** specify a slave station address (0 to 247)
- **Function Code:** The following Modbus function codes are supported by the MWX instruction:
 - 05 – Force Single coil
 - 06 – Preset Single Register
 - 15 – Force Multiple Coils
 - 16 – Preset Multiple Registers
- **Start Slave Memory Address:** specifies the starting slave memory address where the data will be written
- **Start Master Memory Address:** specifies the starting address of the data in the master that is to be written to the slave
- **Number of Elements:** specifies how many consecutive coils or registers will be written to. This field is only active when either function code 15 or 16 is selected
- **Modbus Data Format:** specifies Modbus 584/984 or 484 data format to be used

- **Exception Response Buffer:** specifies the master memory address where the Exception Response will be placed (6-bytes in length). See the table on the following page. The exception response buffer uses 3 words. These bytes are swapped in the MRX/MWX exception response buffer V-memory so:

V-Memory 1 Hi Byte = Function Code Byte (Most Significant Bit Set)

V-Memory 1 Lo Byte = Address Byte

V-Memory 2 Hi Byte = One of the CRC Bytes

V-Memory 2 Lo Byte = Exception Code

V-Memory 3 Hi Byte = 0

V-Memory 3 Lo Byte = Other CRC Byte

MWX Slave Memory Address

| MWX Slave Address Ranges | | |
|--------------------------------|--------------------|---|
| Function Code | Modbus Data Format | Slave Address Range(s) |
| 05 - Force Single Coil | 484 Mode | 1-999 |
| 05 - Force Single Coil | 584/984 Mode | 1-65535 |
| 06 - Preset Single Register | 484 Mode | 4001-4999 |
| 06 - Preset Single Register | 584/984 Mode | 40001-49999 (5 digit) or 400001-465535 (6 digit) |
| 15 - Force Multiple Coils | 484 Mode | 1-999 |
| 15 - Force Multiple Coils | 584/984 Mode | 1-65535 |
| 16 - Preset Multiple Registers | 484 Mode | 4001-4999 |
| 16 - Preset Multiple Registers | 584/984 Mode | 40001-49999 (5 digit) or 400001-465535 (6 digit) |

MWX Master Memory Addresses

| MWX Master Memory Address Ranges | | |
|----------------------------------|----|---------------------|
| Operand Data Type | | DL260 Range |
| Inputs | X | 0-1777 |
| Outputs | Y | 0-1777 |
| Control Relays | C | 0-3777 |
| Stage Bits | S | 0-1777 |
| Timer Bits | T | 0-377 |
| Counter Bits | CT | 0-377 |
| Special Relays | SP | 0-777 |
| V-memory | V | all (see page 3-56) |
| Global Inputs | GX | 0-3777 |
| Global Outputs | GY | 0-3777 |

MWX Number of Elements

| Number of Elements | | |
|--------------------|---|----------------------------------|
| Operand Data Type | | DL260 Range |
| V-memory | V | all (see page 3-56) |
| Constant | K | Bits: 1-2000 Registers: 1-125 |

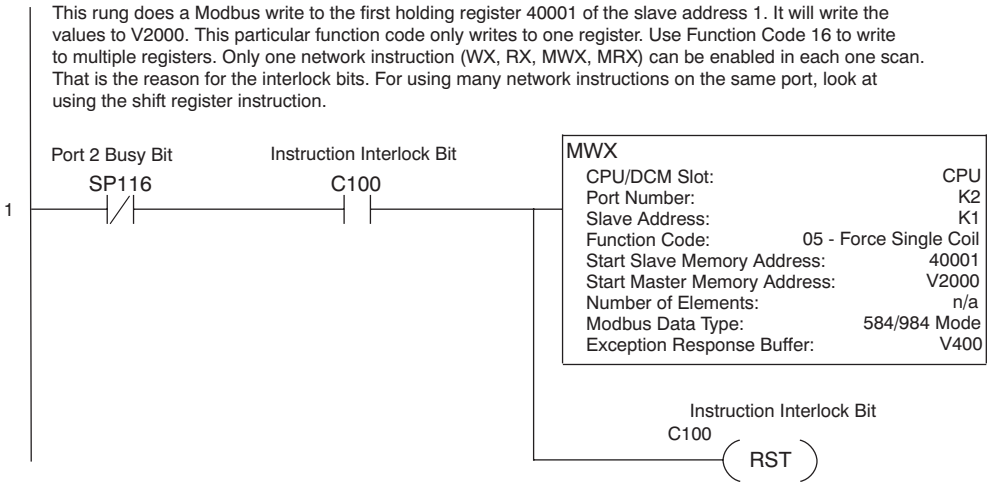
MWX Exception Response Buffer

| Exception Response Buffer | | |
|---------------------------|---|---------------------|
| Operand Data Type | | DL260 Range |
| V-memory | V | all (see page 3-56) |

MWX Example

DL260 port 2 has two Special Relay contacts associated with it (see Appendix D for comm port special relays). One indicates “Port busy” (SP116), and the other indicates ”Port Communication Error” (SP117). The “Port Busy” bit is on while the PLC communicates with the slave. When the bit is off, the program can initiate the next network request. The “Port Communication Error” bit turns on when the PLC has detected an error. Use of this bit is optional. When used, it should be ahead of any network instruction boxes since the error bit is reset when an MRX or MWX instruction is executed.

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



ASCII Instructions (DL260)

 230

 240

 250-1

 260

The DL260 CPU supports several instructions and methods that allow ASCII strings to be read into and written from the PLC communications ports.

Specifically, port 2 on the DL260 can be used for either reading or writing raw ASCII strings, but cannot be used for both on the same CPU.

The DL260 can also decipher ASCII embedded within a supported protocol (K-Sequence, *Direct*NET, Modbus, Ethernet) via the CPU ports, H2-ECOM or D2-DCM module.

ASCII character tables and descriptions can be found at www.asciitable.com.

| | |
|-----|------|
| DS | Used |
| HPP | N/A |

Reading ASCII Input Strings

There are several methods which the DL260 can use to read ASCII input strings:

- 1) **ASCII IN (AIN)** – This instruction configures port 2 for raw ASCII input strings with parameters such as fixed and variable length ASCII strings, termination characters, byte swapping options, and instruction control bits. Use barcode scanners, weight scales, etc., to write raw ASCII input strings into port 2 based on the (AIN) instruction's parameters.
- 2) Write embedded ASCII strings directly to V-memory from an external HMI or similar master device via a supported communications protocol using the CPU ports, H2-ECOM or D2-DCM module. The AIN instruction is not used in this case.
- 3) If a DL260 PLC is a master on a network, the Network Read instruction (RX) can be used to read embedded ASCII data from a slave device via a supported communications protocol using port 2, H2-ECOM or D2-DCM module. The RX instruction places the data directly into V-memory.

Writing ASCII Output Strings

The following instructions can be used to write ASCII output strings:

- 1) **Print from V-memory (PRINTV)** – Use this instruction to write raw ASCII strings out of port 2 to a display panel or a serial printer, etc. The instruction features the starting V-memory address, string length, byte swapping options, etc. When the instruction's permissive bit is enabled, the string is written to port 2.
- 2) **Print to V-memory (VPRINT)** – Use this instruction to create pre-coded ASCII strings in the PLC (i.e. alarm messages). When the instruction's permissive bit is enabled, the message is loaded into a pre-defined V-memory address location. Then use the PRINTV instruction to write the pre-coded ASCII string out of port 2. American, European and Asian Time/Date stamps are supported.

Additionally, if a DL260 PLC is a master on a network, the Network Write instruction (WX) can be used to write embedded ASCII data to an HMI or slave device directly from V-memory via a supported communications protocol using port 2, H2-ECOM or D2-DCM module.

Managing the ASCII Strings

The following instructions can be helpful in managing the ASCII strings within the CPU's V-memory:

ASCII Find (AFIND) – Finds where a specific portion of the ASCII string is located in continuous V-memory addresses. Forward and reverse searches are supported.

ASCII Extract (AEX) – Extracts a specific portion (usually some data value) from the ASCII find location or other known ASCII data location.

Compare V-memory (CMPV) – This instruction is used to compare two blocks of V-memory addresses and is usually used to detect a change in an ASCII string. Compared data types must be of the same format (i.e., BCD, ASCII, etc.).

Swap Bytes (SWAPB) – usually used to swap V-memory bytes on ASCII data that was written directly to V-memory from an external HMI or similar master device via a communications protocol. The AIN and AEX instructions have a built-in byte swap feature.

ASCII Input (AIN)

The ASCII Input instruction allows the CPU to receive ASCII strings through the specified communications port and places the string into a series of specified V-memory registers. The ASCII data can be received as a fixed number of bytes or as a variable length string with a specified termination character(s). Other features include Byte Swap preferences, Character Timeout, and user-defined flag bits for Busy, Complete and Timeout Error.

- 230
- 240
- 250-1
- 260

| | |
|-----|------|
| DS | Used |
| HPP | N/A |

AIN

Length Type

☒ Fixed Length

☐ Variable Length

CPU/DCM

☒ CPU

☐ DCM

Slot Number:

K0

Port Number:

K2

Data Destination:

V2000

* Data Destination = Byte count

* Data Destination + 1 = Start of data

Fixed Length:

K32

Interchar. Timeout:

None

First Char. Timeout:

None

Byte Swap:

☒ None

☐ All

☐ All but null

Termination Code Length:

☒ 1 Character

☐ 2 Characters

TermCode 1:

00

hexadecimal

TermCode 2:

00

hexadecimal

Overflow Error:

C0

Busy:

C0

Complete:

C1

Interchar. T/O Error:

K0

First Char. T/O Error:

K0

AIN Fixed Length Configuration

- **Length Type:** select fixed length based on the length of the ASCII string that will be sent to the CPU port.
- **Port Number:** must be DL260 port 2 (K2).
- **Data Destination:** specifies where the ASCII string will be placed in V-memory.
- **Fixed Length:** specifies the length, in bytes, of the fixed-length ASCII string the port will receive.
- **Inter-character Timeout:** if the amount of time between incoming ASCII characters exceeds the set time, the specified Timeout Error bit will be set. No data will be stored at the Data Destination V-memory location. The bit will reset when the AIN instruction permissive bits are disabled. *None* selection disables this feature.
- **First Character Timeout:** if the amount of time from when the AIN is enabled to the time the first character is received exceeds the set time, the specified First Character Timeout bit will be set. The bit will reset when the AIN instruction permissive bits are disabled. *None* selection disables this feature.
- **Byte Swap:** swaps the high-byte and low-byte within each V-memory register of the Fixed Length ASCII string. See the SWAPB instruction for details.
- **Busy Bit:** is ON while the AIN instruction is receiving ASCII data.
- **Complete Bit:** is set once the ASCII data has been received for the specified fixed length and reset when the AIN instruction permissive bits are disabled.
- **Inter-character Timeout Error Bit:** is set when the Character Timeout is exceed. See Character Timeout explanation above.
- **First Character Timeout Error Bit:** is set when the First Character Timeout is exceeded. See First Character Timeout explanation above.

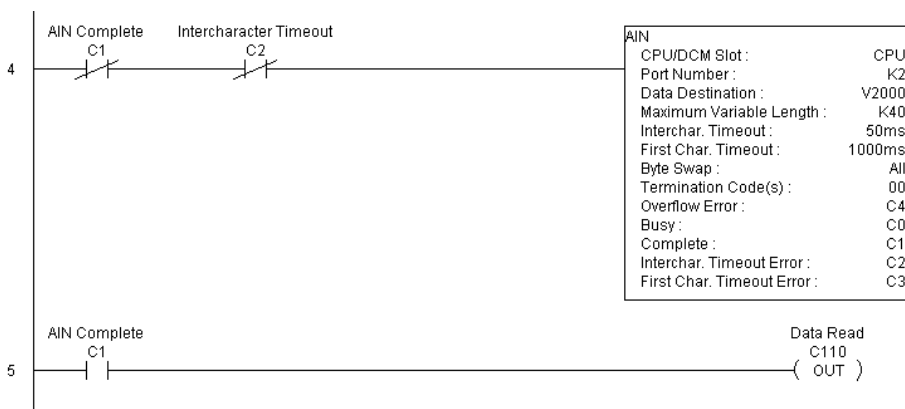
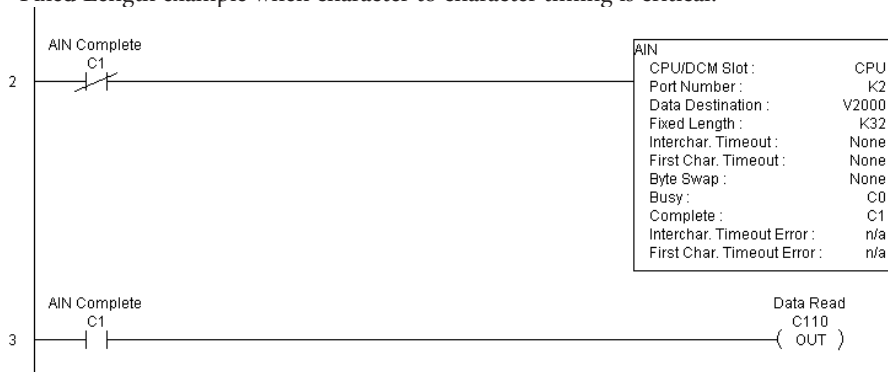
| Parameter | DL260 Range |
|---|-------------------------------|
| Data Destination | All V-memory (See page 3 -56) |
| Fixed Length | K1-128 |
| Bits: Busy, Complete, Timeout Error, Overflow | C0-3777 |

| Discrete Bit Flags | Description |
|--------------------|--|
| SP53 | On if the CPU cannot execute the instruction |
| SP71 | On when a value used by the instruction is invalid |
| SP116 | On when CPU port 2 is communicating with another device |
| SP117 | On when CPU port 2 has experienced a communication error |

AIN Fixed Length Examples

Fixed Length example when the PLC is reading the port continuously and timing is not critical.

Fixed Length example when character to character timing is critical.



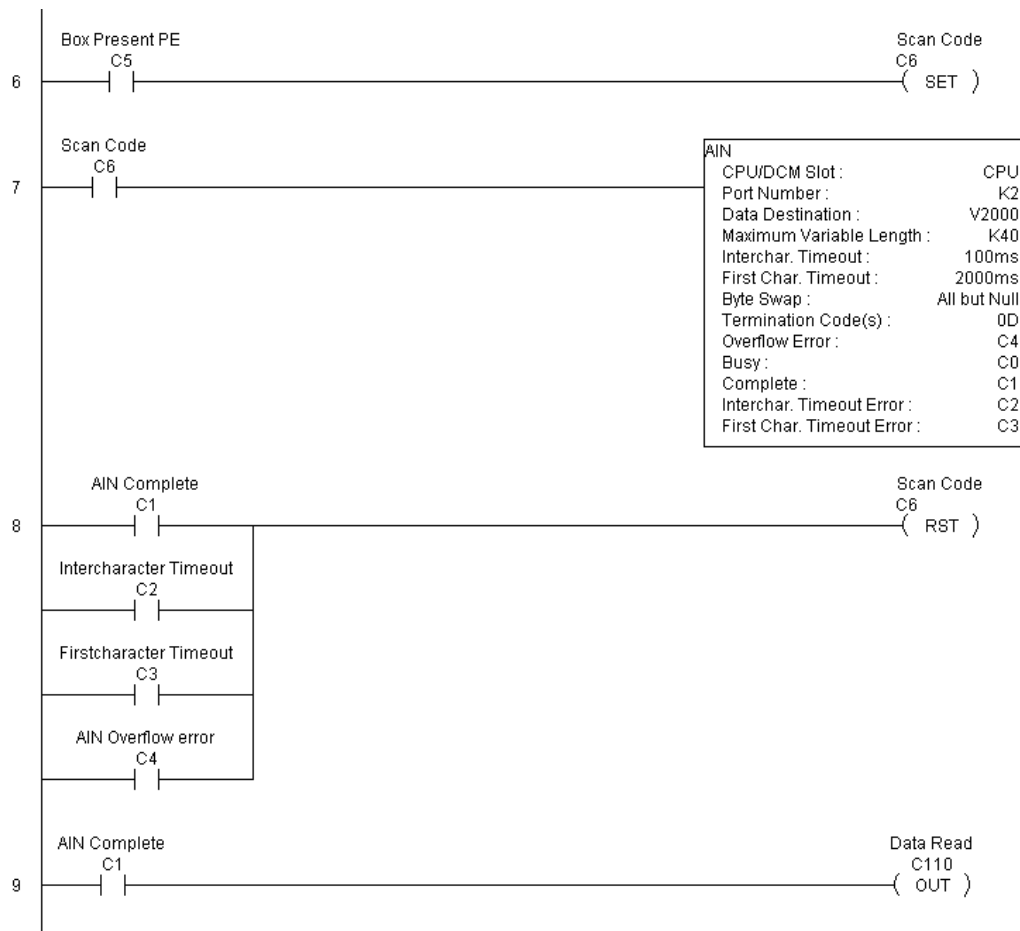
AIN Variable Length Configuration:

- **Length Type:** select Variable Length if the ASCII string length followed by termination characters will vary in length.
- **Port Number:** must be DL260 port 2 (K2).
- **Data Destination:** specifies where the ASCII string will be placed in V-memory.
- **Maximum Variable Length:** specifies, in bytes, the maximum length of a Variable Length ASCII string the port will receive.
- **Inter-character Timeout:** if the amount of time between incoming ASCII characters exceeds the set time, the Timeout Error bit will be set. No data will be stored at the Data Destination V-memory location. The Timeout Error bit will reset when the AIN instruction permissive bits are disabled. *None* selection disables this feature.
- **First Character Timeout:** if the amount of time from when the AIN is enabled to the time the first character is received exceeds the set time, the specified First Character Timeout bit will be set. The bit will reset when the AIN instruction permissive bits are disabled. *None* selection disables this feature.
- **Byte Swap:** swaps the high-byte and low-byte within each V-memory register of the Variable Length ASCII string. See the SWAPB instruction for details.
- **Termination Code Length:** consists of either 1 or 2 characters. Refer to the ASCII table in Appendix G.
- **Overflow Error Bit:** is set when the ASCII data received exceeds the Maximum Variable Length specified.
- **Busy Bit:** is ON while the AIN instruction is receiving ASCII data.
- **Complete Bit:** is set once the ASCII data has been received up to the termination code characters. It will be reset when the AIN instruction permissive bits are disabled.
- **Inter-character Timeout Error Bit:** is set when the Character Timeout is exceed. See Character Timeout explanation above.
- **First Character Timeout Error Bit:** is set when the First Character Timeout is exceeded. See First Character Timeout explanation above.

| Parameter | DL260 Range |
|---|------------------------------|
| Data Destination | All V-memory (See page 3-56) |
| Max. Variable Length | K1-128 |
| Bits: Busy, Complete, Timeout Error, Overflow | C0-3777 |

AIN Variable Length Example

AIN Variable Length example used to read barcodes on boxes (PE = photoelectric sensor).



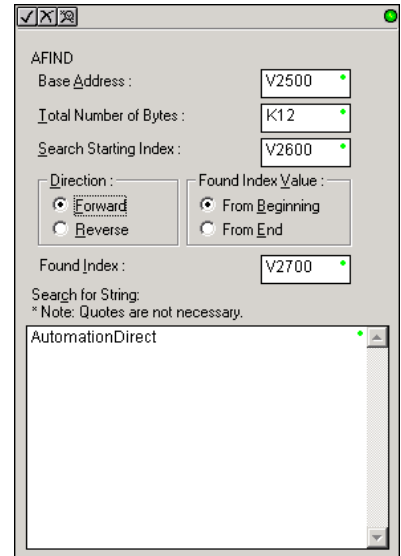
ASCII Find (AFIND)

-  230
-  240
-  250-1
-  260

The ASCII Find instruction locates a specific ASCII string or portion of an ASCII string within a range of V-memory registers and places the string's Found Index number (byte number where desired string is found) in Hex, into a specified V-memory register. Other features include, Search Starting Index number for skipping over unnecessary bytes before beginning the FIND operation, Forward or Reverse direction search, and From Beginning and From End selections to reference the Found Index Value.

| | |
|-----|------|
| DS | Used |
| HPP | N/A |

- **Base Address:** specifies the beginning V-memory register where the entire ASCII string is stored in memory.
- **Total Number of Bytes:** specifies the total number of bytes to search for the desired ASCII string.
- **Search Starting Index:** specifies which byte to skip to (with respect to the Base Address) before beginning the search.
- **Direction:** Forward begins the search from lower numbered V-memory registers to higher numbered V-memory registers. Reverse does the search from higher numbered V-memory registers to lower-numbered V-memory registers.
- **Found Index Value:** specifies whether the Beginning or the End byte of the ASCII string found will be loaded into the Found Index register.
- **Found Index:** specifies the V-memory register where the Found Index Value will be stored. A value of FFFF will result if the desired string is not located in the memory registers specified. A value of EEEE will result if there is a conflict in the AFIND search parameters specified.
- **Search for String:** up to 128 characters.



| Parameter | DL260 Range |
|-----------------------|--|
| Base Address | All V-memory (See page 3-56) |
| Total Number of Bytes | All V-memory (See page 3-56) or K1-128 |
| Search Starting Index | All V-memory (See page 3-56) or K0-127 |
| Found Index | All V-memory (See page 3-56) |

| Discrete Bit Flags | Description |
|--------------------|---|
| SP53 | On if the CPU cannot execute the instruction. |
| SP71 | On when a value used by the instruction is invalid. |

In the following example, the AFIND instruction is used to search for the “day” portion of “Friday” in the ASCII string “Today is Friday,” which had previously been loaded into V-memory. Note that a Search Starting Index of constant (K) 5 combined with a Forward Direction Search is used to prevent finding the “day” portion of the word “Today.” The Found Index will be placed into V4000.

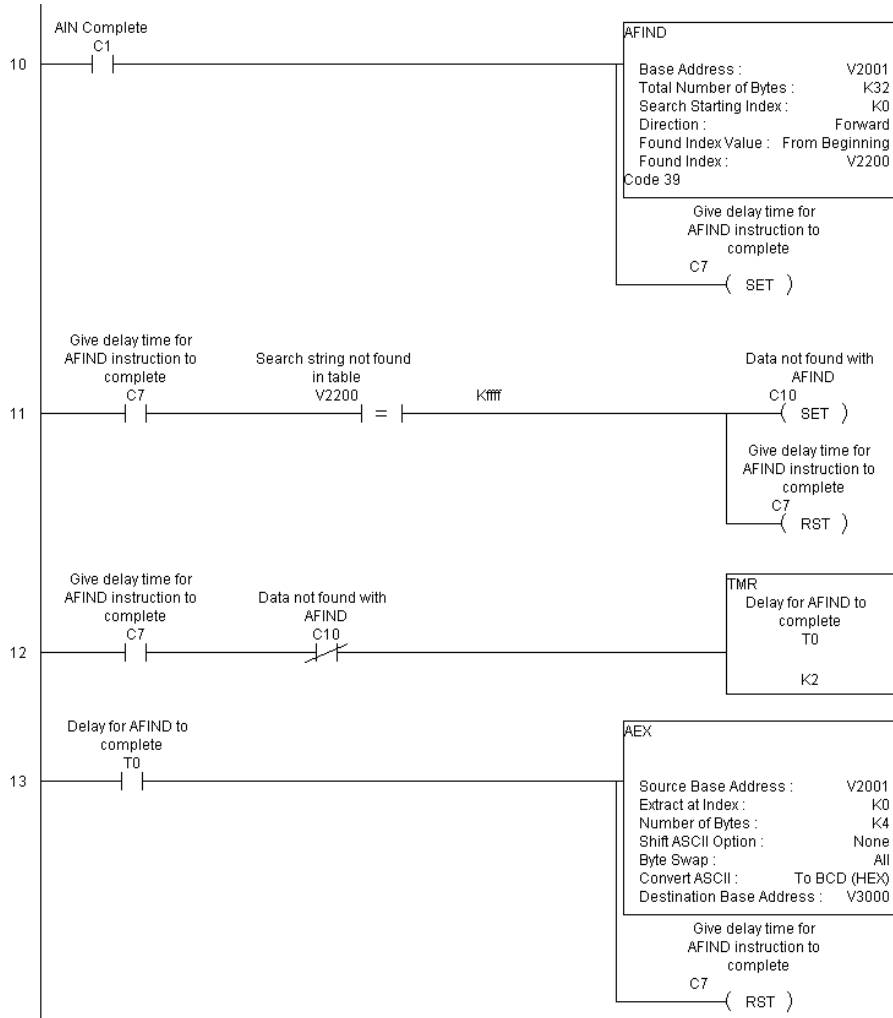
The diagram illustrates the search process for the word "array" in the ASCII Table. The table is organized into columns for ASCII Characters, their HEX Equivalents, and their categories (Low, High). The search starts at index 0 (Base Address) and proceeds downwards (Forward Direction Search) until the character 'y' is found at index 4. The search then proceeds to index 5 (Search start Index Number) and continues downwards (Forward Direction Search) until the character 'r' is found at index 10. The search then proceeds to index 11 (Beginning Index Number) and continues downwards (Forward Direction Search) until the character 'a' is found at index 13. The search then proceeds to index 14 (End Index Number) and continues downwards (Forward Direction Search) until the character 'y' is found at index 14. The search then proceeds to index 15 (Found Index Number) and continues downwards (Forward Direction Search) until the character 'r' is found at index 15. The final result is the word "array" found at index 15.

| Index | ASCII Character | HEX Equivalent | Category | Address |
|-------|-----------------|----------------|----------|---------|
| 0 | T | 54h | Low | V3000 |
| 1 | o | 6Fh | High | V3000 |
| 2 | d | 64h | Low | V3001 |
| 3 | a | 61h | High | V3001 |
| 4 | y | 79h | Low | V3002 |
| 5 | | 20h | High | V3002 |
| 6 | i | 69h | Low | V3003 |
| 7 | s | 73h | High | V3003 |
| 8 | | 20h | Low | V3004 |
| 9 | F | 46h | High | V3004 |
| 10 | r | 72h | Low | V3005 |
| 11 | i | 69h | High | V3005 |
| 12 | d | 64h | Low | V3006 |
| 13 | a | 61h | High | V3006 |
| 14 | y | 79h | Low | V3007 |
| 15 | . | 2Eh | High | V3007 |

Found Index Number = 0012 V4000

AFIND Example Combined with AEX Instruction

When an AIN instruction has executed, its Complete bit can be used to trigger an AFIND instruction to search for a desired portion of the ASCII string. Once the string is found, the AEX instruction can be used to extract the located string.



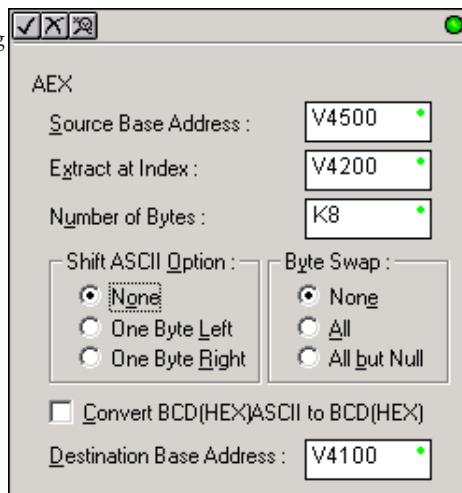
ASCII Extract (AEX)

-  230
-  240
-  250-1
-  260

The ASCII Extract instruction extracts a specified number of bytes of ASCII data from one series of V-memory registers and places them into another series of V-memory registers. Other features include Extract at Index for skipping over unnecessary bytes before beginning the Extract operation, Shift ASCII Option, for One Byte Left or One Byte Right, Byte Swap and Convert data to a BCD format number.

| | |
|-----|------|
| DS | Used |
| HPP | N/A |

- **Source Base Address:** specifies the beginning V-memory register where the entire ASCII string is stored in memory.
- **Extract at Index:** specifies which byte to skip to (with respect to the Source Base Address) before extracting the data.
- **Number of Bytes:** specifies the number of bytes to be extracted.
- **Shift ASCII Option:** shifts all extracted data one byte left or one byte right to displace “unwanted” characters, if necessary.
- **Byte Swap:** swaps the high-byte and the low-byte within each V-memory register of the extracted data. See the SWAPB instruction for details.
- **Convert BCD(Hex) ASCII to BCD (Hex):** if enabled, this will convert ASCII numerical characters to Hexadecimal numerical values.
- **Destination Base Address:** specifies the V-memory register where the extracted data will be stored.



| Parameter | DL260 Range |
|--------------------------|--|
| Source Base Address | All V-memory (See page 3-56) |
| Extract at Index | All V-memory (See page 3-56) or K0-127 |
| Number of Bytes | K1-128 |
| Destination Base Address | All V-memory (See page 3-56) |

See the previous page for an example using the AEX instruction.

| Discrete Bit Flags | Description |
|--------------------|---|
| SP53 | On if the CPU cannot execute the instruction. |
| SP71 | On when a value used by the instruction is invalid. |

ASCII Compare (CMPV)

- ☐ 230
☐ 240
☐ 250-1
☒ 260

| | |
|-----|------|
| DS | Used |
| HPP | N/A |

The ASCII Compare instruction compares two groups of V-memory registers. The CMPV will compare any data type (ASCII to ASCII, BCD to BCD, etc) of one series (group) of V-memory registers to another series of V-memory registers for a specified byte length.

“Compare from” Starting Address: specifies the beginning V-memory register of the first group of V-memory registers to be compared from.

“Compare to” Starting Address: specifies the beginning V-memory register of the second group of V-memory registers to be compared to.

Number of Bytes: specifies the length of each V-memory group to be compared.

SP61 = 1 (ON), the result is equal

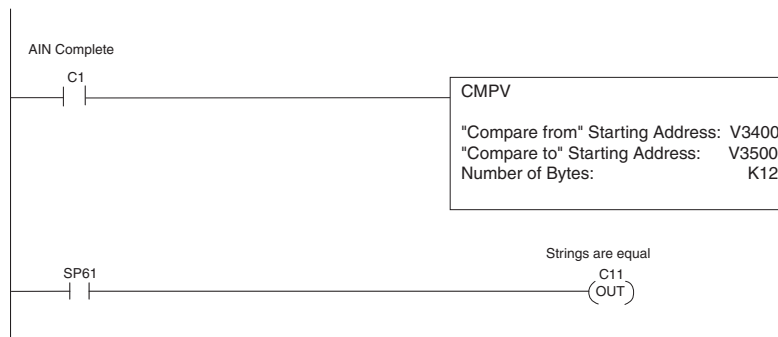
SP61 = 0 (OFF), the result is not equal

| Parameter | DL260 Range |
|-------------------------------|--|
| Compare from Starting Address | All V-memory (See page 3-56) |
| Compare to Starting Address | All V-memory (See page 3-56) |
| Number of Bytes | All V-memory (See page 3-56) or K0-127 |

| Discrete Bit Flags | Description |
|--------------------|---|
| SP53 | On if the CPU cannot execute the instruction. |
| SP61 | On when result is equal. |
| SP71 | On when a value used by the instruction is invalid. |

CMPV Example

The CMPV instruction executes when the AIN instruction is complete. If the compared V-memory tables are equal, SP61 will turn ON.



ASCII Print to V-memory (VPRINT)

230

240

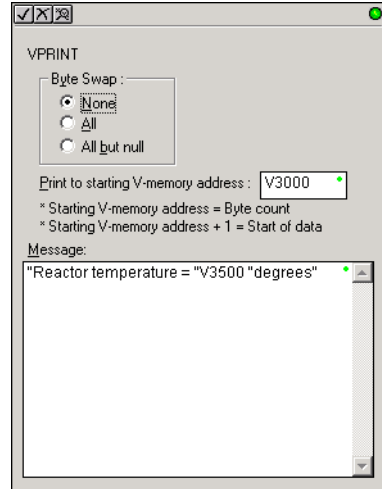
250-1

260

| | |
|-----|------|
| DS | Used |
| HPP | N/A |

The ASCII Print to V-memory instruction will write a specified ASCII string into a series of V-memory registers. Other features include Byte Swap, options to suppress or convert leading zeros or spaces, and `_Date` and `_Time` options for U.S., European, and Asian date formats and 12- or 24-hour time formats.

- **Byte Swap:** swaps the high-byte and low-byte within each V-memory register to which the ASCII string is printed. See the SWAPB instruction for details.
- **Print to Starting V-memory Address:** specifies the beginning of a series of V-memory addresses where the ASCII string will be placed by the VPRINT instruction.
- **Starting V-memory Address:** the first V-memory register of the series of registers specified will contain the ASCII string's length in bytes.
- **Starting V-memory Address + 1:** the 2nd and subsequent registers will contain the ASCII string printed to V-memory.



VPRINT Time/Date Stamping

| Parameter | DL260 Range |
|------------------------------------|------------------------------|
| Print to Starting V-memory Address | All V-memory (See page 3-56) |

| Discrete Bit Flags | Description |
|--------------------|---|
| SP53 | On if the CPU cannot execute the instruction. |
| SP71 | On when a value used by the instruction is invalid. |

The codes in the table below can be used in the VPRINT ASCII string message to “print to V-memory” the current time and/or date.

| # | Character Code | Date/Time Stamp Options |
|---|-----------------------|--|
| 1 | <code>_Date:us</code> | American standard (month/day/2 digit year) |
| 2 | <code>_Date:e</code> | European standard (day/month/2 digit year) |
| 3 | <code>_Date:a</code> | Asian standard (2 digit year/month/day) |
| 4 | <code>_Time:12</code> | standard 12 hour clock (0-12 hour:min am/pm) |
| 5 | <code>_Time:24</code> | standard 24 hour clock (0-23 hour:min am/pm) |

VPRINT V-memory element

The following modifiers can be used in the VPRINT ASCII string message to “print to V-memory” register contents in integer format or real format. Use V-memory number or V-memory number with “:” and data type. The data types are shown in the table below. The Character code must be capital letters.



NOTE: There must be a space entered before and after the V-memory address to separate it from the text string. Failure to do this will result in error code 499.

| # | Character Code | Description |
|---|----------------|---|
| 1 | none | 16-bit binary (decimal number) |
| 2 | : B | 4-digit BCD |
| 3 | : D | 32-bit binary (decimal number) |
| 4 | : D B | 8-digit BCD |
| 5 | : R | Floating point number (real number) |
| 6 | : E | Floating point number (real number with exponent) |

Examples:

V2000 Print binary data in V2000 for decimal number

V2000 : B Print BCD data in V2000

V2000 : D Print binary number in V2000 and V2001 for decimal number

V2000 : D B Print BCD data in V2000 and V2001

V2000 : R Print floating point number in V2000/V2001 as real number

V2000 : E Print floating point number in V2000/V2001 as real number with exponent

The following modifiers can be added to any of the modifiers above to suppress or convert leading zeros or spaces. The character code must be capital letters.

| # | Character Code | Description |
|---|----------------|----------------------------------|
| 1 | S | Suppresses leading spaces |
| 2 | C0 | Converts leading spaces to zeros |
| 3 | 0 | Suppresses leading zeros |

Example with V2000 = 0018 (binary format)

| V-memory Register with Modifier | Number of Characters | | | |
|------------------------------------|----------------------|---|---|---|
| | 1 | 2 | 3 | 4 |
| V2000 | 0 | 0 | 1 | 8 |
| V2000:B | 0 | 0 | 1 | 2 |
| V2000:BO | 1 | 2 | | |

Example with V2000 = sp sp18 (binary format) where sp = space

| V-memory Register with Modifier | Number of Characters | | | |
|---------------------------------|----------------------|----|---|---|
| | 1 | 2 | 3 | 4 |
| V2000 | sp | sp | 1 | 8 |
| V2000:B | sp | sp | 1 | 2 |
| V2000:BS | 1 | 2 | | |
| V2000:BC0 | 0 | 0 | 1 | 2 |

VPRINT V-memory text element

The following is used for “printing to V-memory” text stored in registers. Use the % followed by the number of characters after V-memory number for representing the text. If you assign “0” as the number of characters, the function will read the character count from the first location. Then it will start at the next V-memory location and read that number of ASCII codes for the text from memory.

Example:

V2000 % 16 16 characters in V2000 to V2007 are printed.

V2000 % 0 The characters in V2001 to Vxxxx (determined by the number in V2000) will be printed.

VPRINT Bit element

The following is used for “printing to V-memory” the state of the designated bit in V-memory or a control relay bit. The bit element can be assigned by the designating point (.) and bit number preceded by the V-memory number or relay number. The output type is described as shown in the table below.

| # | Data format | Description |
|---|-------------|--|
| 1 | none | Print 1 for an ON state, and 0 for an OFF state |
| 2 | : BOOL | Print “TRUE” for an ON state, and “FALSE” for an OFF state |
| 3 | : ONOFF | Print “ON” for an ON state, and “OFF” for an OFF state |

Example:

V2000.15 Prints the status of bit 15 in V2000, in 1/0 format

C100 Prints the status of C100 in 1/0 format

C100 : BOOL Prints the status of C100 in TRUE/FALSE format

C100 : ON/OFF Prints the status of C100 in ON/OFF format

V2000.15 : BOOL Prints the status of bit 15 in V2000 in TRUE/FALSE format

The maximum numbers of characters you can VPRINT is 128. The number of characters required for each element, regardless of whether the :S, :C0 or :0 modifiers are used, is listed in the table below.

| Element Type | Maximum Characters |
|-------------------------------------|--------------------|
| Text, 1 character | 1 |
| 16-bit binary | 6 |
| 32-bit binary | 11 |
| 4-digit BCD | 4 |
| 8-digit BCD | 8 |
| Floating point (real number) | 13 |
| Floating point (real with exponent) | 13 |
| V-memory/text | 2 |
| Bit (1/0 format) | 1 |
| Bit (TRUE/FALSE format) | 5 |
| Bit (ON/OFF format) | 3 |

Text element

The following is used for “printing to V-memory” character strings. The character strings are defined as the character (more than 0) ranged by the double quotation marks. Two hex numbers preceded by the dollar sign means an 8-bit ASCII character code. Also, two characters preceded by the dollar sign is interpreted according to the following table:

| # | Character code | Description |
|---|----------------|----------------------------------|
| 1 | \$\$ | Dollar sign (\$) |
| 2 | \$" | Double quotation (") |
| 3 | \$L or \$l | Line feed (LF) |
| 4 | \$N or \$n | Carriage return line feed (CRLF) |
| 5 | \$P or \$p | Form feed |
| 6 | \$R or \$r | Carriage return (CR) |
| 7 | \$T or \$t | Tab |

The following examples show various syntax conventions and the length of the output to the printer.

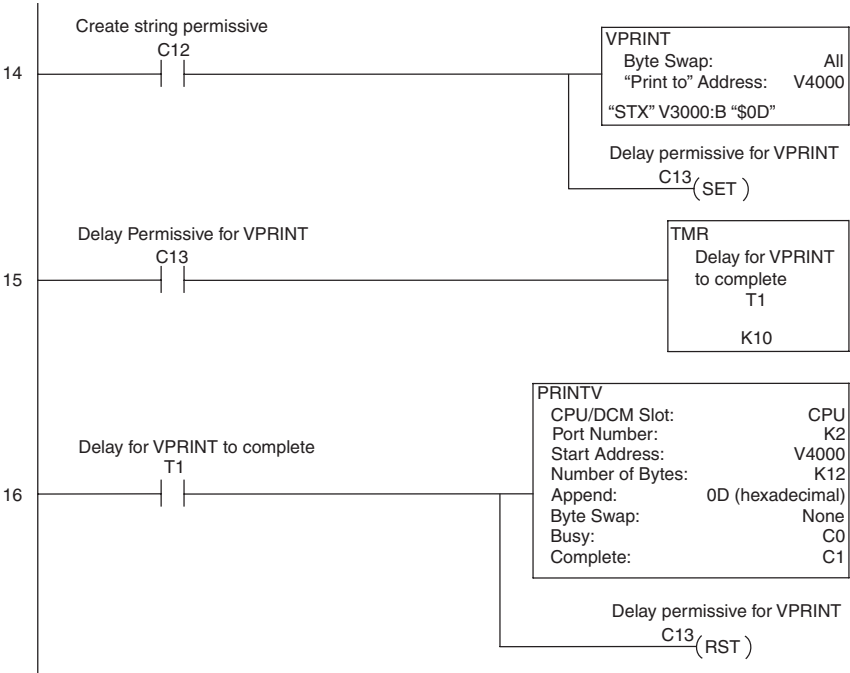
Example:

" " Length 0 without character
"A" Length 1 with character A
" " Length 1 with blank
" \$" Length 1 with double quotation mark
" \$ R \$ L " Length 2 with one CR and one LF
" \$ 0 D \$ 0 A " Length 2 with one CR and one LF
" \$ \$ " Length 1 with one \$ mark

In printing an ordinary line of text, you will need to include **double quotation** marks before and after the text string. Error code 499 will occur in the CPU when the print instruction contains invalid text or no quotations. It is important to test your VPRINT instruction data during application development.

VPRINT Example Combined with PRINTV Instruction

The VPRINT instruction is used to create a string in V-memory. The PRINTV is used to print the string out of port 2.



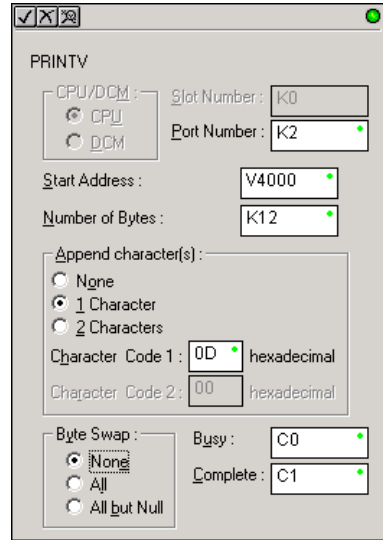
ASCII Print from V-memory (PRINTV)

-  230
-  240
-  250-1
-  260

| | |
|-----|------|
| DS | Used |
| HPP | N/A |

The ASCII Print from V-memory instruction will send an ASCII string out of the designated communications port from a specified series of V-memory registers for a specified length in number of bytes. Other features include user specified Append Characters to be placed after the desired data string for devices that require specific termination character(s), Byte Swap options, and user specified flags for Busy and Complete.

- **Port Number:** must be DL260 port 2 (K2).
- **Start Address:** specifies the beginning of series of V-memory registers that contain the ASCII string to print.
- **Number of Bytes:** specifies the length of the string to print.
- **Append Characters:** specifies ASCII characters to be added to the end of the string for devices that require specific termination characters.
- **Byte Swap:** swaps the high-byte and low-byte within each V-memory register of the string while printing. See the SWAPB instruction for details.
- **Busy Bit:** will be ON while the instruction is printing ASCII data.
- **Complete Bit:** will be set once the ASCII data has been printed and reset when the PRINTV instruction permissive bits are disabled.



| Parameter | DL260 Range |
|----------------------|--|
| Port Number | port 2 (K2) |
| Start Address | All V-memory (See page 3-56) |
| Number of Bytes | All V-memory (See page 3-56) or K1-128 |
| Bits: Busy, Complete | C0-3777 |

See the facing page for an example using the PRINTV instruction.

| Discrete Bit Flags | Description |
|--------------------|---|
| SP53 | On if the CPU cannot execute the instruction. |
| SP71 | On when a value used by the instruction is invalid. |
| SP116 | On when CPU port 2 is communicating with another device. |
| SP117 | On when CPU port 2 has experienced a communication error. |

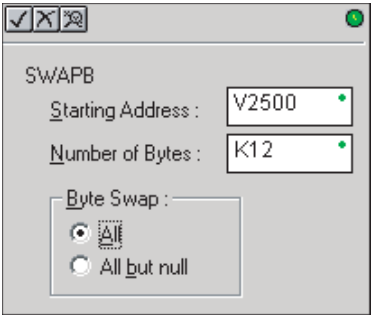
ASCII Swap Bytes (SWAPB)

- 230
- 240
- 250-1
- 260

| | |
|-----|------|
| DS | Used |
| HPP | N/A |

The ASCII Swap Bytes instruction swaps byte positions (high-byte to low-byte and low-byte to high-byte) within each V-memory register of a series of V-memory registers for a specified number of bytes.

- Starting Address: specifies the beginning of a series of V-memory registers the instruction will use to begin byte swapping
- Number of Bytes: specifies the number of bytes, beginning with the Starting Address, to byte swap



| Parameter | DL260 Range |
|------------------|---|
| Starting Address | All V-memory (See page 3-56) |
| Number of Bytes | All V-memory (See page 3-56) or K1 to 128 |

| Discrete Bit Flags | Description |
|--------------------|---|
| SP53 | On if the CPU cannot execute the instruction. |
| SP71 | On when a value used by the instruction is invalid. |

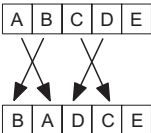
Byte Swap Preferences

No Byte Swapping (AIN, AEX, PRINTV, VPRINT)



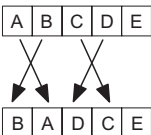
| Byte | |
|-------|-------|
| High | Low |
| V2000 | 0005h |
| V2001 | B A |
| V2002 | D C |
| V2003 | xx E |

Byte Swap All



| Byte | |
|-------|-------|
| High | Low |
| V2000 | 0005h |
| V2001 | A B |
| V2002 | C D |
| V2003 | E xx |

Byte Swap All but Null



| Byte | |
|-------|-------|
| High | Low |
| V2000 | 0005h |
| V2001 | A B |
| V2002 | C D |
| V2003 | xx E |

SWAPB Example

The AIN Complete bit is used to trigger the SWAPB instruction. Use a one-shot so the SWAPB only executes once.

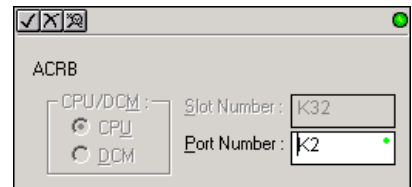


ASCII Clear Buffer (ACRB)

The ASCII Clear Buffer instruction will clear the ASCII receive buffer of the specified communications port number.

- ☐ 230
- ☐ 240
- ☐ 250-1
- ☒ 260

Port Number: must be DL260 port 2 (K2)



| | |
|-----|------|
| DS | Used |
| HPP | N/A |

ACRB Example

The AIN Complete bit or the AIN diagnostic bits are used to clear the ASCII buffer.



Intelligent Box (IBox) Instructions (DL250-1/DL260 Only)

A new class of instructions, called Ibox Instructions, became available with the introduction of *DirectSOFT*. These powerful, yet easy-to-use instructions simplify many of the more complicated tasks that could previously be accomplished only through the use of multiple RLL Instructions. The IBox Instructions are supported by DL250-1 and DL260 PLCs. The D2-250-1 CPU requires firmware version v4.60 or later, and the D2-260 CPU requires firmware version v2.40 or later. For more information on *DirectSOFT* or to download our free version, please visit our Web site at: www.automationdirect.com.

| Analog Helper IBoxes | | |
|--|--------|-------|
| Instruction | IBox # | Page |
| Analog Input / Output Combo Module Pointer Setup (ANLGCMB) | IB-462 | 5-232 |
| Analog Input Module Pointer Setup (ANLGIN) | IB-460 | 5-234 |
| Analog Output Module Pointer Setup (ANLGOUT) | IB-461 | 5-236 |
| Analog Scale 12-Bit BCD to BCD (ANSCL) | IB-423 | 5-238 |
| Analog Scale 12-Bit Binary to Binary (ANSCLB) | IB-403 | 5-239 |
| Filter Over Time - BCD (FILTER) | IB-422 | 5-240 |
| Filter Over Time - Binary (FILTERB) | IB-402 | 5-242 |
| Hi/Low Alarm - BCD (HILOAL) | IB-421 | 5-244 |
| Hi/Low Alarm - Binary (HILOALB) | IB-401 | 5-246 |

| Discrete Helper IBoxes | | |
|-------------------------------------|--------|-------|
| Instruction | Ibox # | Page |
| Off Delay Timer (OFFDTMR) | IB-302 | 5-248 |
| On Delay Timer (ONDTMR) | IB-301 | 5-250 |
| One Shot (ONESHOT) | IB-303 | 5-252 |
| Push On / Push Off Circuit (PONOFF) | IB-300 | 5-253 |

| Memory IBoxes | | |
|--------------------------|--------|-------|
| Instruction | Ibox # | Page |
| Move Single Word (MOVEW) | IB-200 | 5-254 |
| Move Double Word (MOVED) | IB-201 | 5-255 |

| Math IBoxes | | |
|--|--------|-------|
| Instruction | Ibox # | Page |
| BCD to Real with Implied Decimal Point (BCDTOR) | IB-560 | 5-256 |
| Double BCD to Real with Implied Decimal Point (BCDTORD) | IB-562 | 5-257 |
| Math - BCD (MATHBCD) | IB-521 | 5-258 |
| Math - Binary (MATHBIN) | IB-501 | 5-260 |
| Math - Real (MATHR) | IB-541 | 5-262 |
| Real to BCD with Implied Decimal Point and Rounding (RTOBCD) | IB-561 | 5-263 |
| Real to Double BCD with Implied Decimal Point and Rounding (RTOBCDD) | IB-563 | 5-264 |
| Square BCD (SQUARE) | IB-523 | 5-265 |
| Square Binary (SQUAREB) | IB-503 | 5-266 |
| Square Real(SQUARER) | IB-543 | 5-267 |
| Sum BCD Numbers (SUMBCD) | IB-522 | 5-268 |
| Sum Binary Numbers (SUMBIN) | IB-502 | 5-269 |
| Sum Real Numbers (SUMR) | IB-542 | 5-270 |

| Communication IBoxes | | |
|--|--------|-------|
| Instruction | Ibox # | Page |
| ECOM100 Configuration (ECOM100) | IB-710 | 5-272 |
| ECOM100 Disable DHCP (ECDHCPD) | IB-736 | 5-274 |
| ECOM100 Enable DHCP (ECDHCPD) | IB-735 | 5-276 |
| ECOM100 Query DHCP Setting (ECDHCPQ) | IB-734 | 5-278 |
| ECOM100 Send E-mail (ECEMAIL) | IB-711 | 5-280 |
| ECOM100 Restore Default E-mail Setup (ECEMRDS) | IB-713 | 5-283 |
| ECOM100 E-mail Setup (ECEMSUP) | IB-712 | 5-286 |
| ECOM100 IP Setup (ECIPSUP) | IB-717 | 5-290 |
| ECOM100 Read Description (ECRDDDES) | IB-726 | 5-292 |
| ECOM100 Read Gateway Address (ECRDGWA) | IB-730 | 5-294 |
| ECOM100 Read IP Address (ECRDIP) | IB-722 | 5-296 |
| ECOM100 Read Module ID (ECRDMID) | IB-720 | 5-298 |
| ECOM100 Read Module Name (ECRDNAM) | IB-724 | 5-300 |
| ECOM100 Read Subnet Mask (ECRDSNM) | IB-732 | 5-302 |
| ECOM100 Write Description (ECWRDES) | IB-727 | 5-304 |
| ECOM100 Write Gateway Address (ECWRGWA) | IB-731 | 5-306 |
| ECOM100 Write IP Address (ECWRIP) | IB-723 | 5-308 |
| ECOM100 Write Module ID (ECWRMID) | IB-721 | 5-310 |
| ECOM100 Write Name (ECWRNAM) | IB-725 | 5-312 |
| ECOM100 Write Subnet Mask (ECWRSNM) | IB-733 | 5-314 |
| ECOM100 RX Network Read (ECRX) | IB-740 | 5-316 |
| ECOM100 WX Network Write (ECWX) | IB-741 | 5-319 |
| NETCFG Network Configuration (NETCFG) | IB-700 | 5-322 |
| Network RX Read (NETRX) | IB-701 | 5-324 |
| Network WX Write (NETWX) | IB-702 | 5-327 |

| Counter I/O IBoxes (Work with H2-CTRIO and H2-CTRIO2) | | |
|---|---------|-------|
| Instruction | Ibox # | Page |
| CTRIO Configuration (CTRIO) | IB-1000 | 5-330 |
| CTRIO Add Entry to End of Preset Table (CTRADPT) | IB-1005 | 5-332 |
| CTRIO Clear Preset Table (CTRCLRT) | IB-1007 | 5-335 |
| CTRIO Edit Preset Table Entry (CTREDPT) | IB-1003 | 5-338 |
| CTRIO Edit Preset Table Entry and Reload (CTREDRL) | IB-1002 | 5-342 |
| CTRIO Initialize Preset Table (CTRINPT) | IB-1004 | 5-346 |
| CTRIO Initialize Preset Table (CTRINTR) | IB-1010 | 5-350 |
| CTRIO Load Profile (CTRLDPR) | IB-1001 | 5-354 |
| CTRIO Read Error (CTRRDER) | IB-1014 | 5-357 |
| CTRIO Run to Limit Mode (CTRRTLML) | IB-1011 | 5-359 |
| CTRIO Run to Position Mode (CTRRTPM) | IB-1012 | 5-362 |
| CTRIO Velocity Mode (CTRVELO) | IB-1013 | 5-365 |
| CTRIO Write File to ROM (CTRWFTFTR) | IB-1006 | 5-368 |



NOTE: Check your CPU firmware version using **DirectSOFT**: PLC Menu > Diagnostics > System Information. The latest firmware and update tool are available from:

<http://support.automationdirect.com/firmware/index.html>

Analog Input/Output Combo Module Pointer Setup (ANLGCMB) (IB-462)

The Analog Input/Output Combo Module Pointer Setup instruction generates the logic to configure the pointer method for an analog input/output combination module on the first PLC scan following a Program to Run transition.

230

240

250-1

260

The ANLGCMB IBox instruction determines the data format and Pointer addresses based on the CPU type, the Base# and the module Slot#.

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

The Input Data Address is the starting location in user V-memory where the analog input data values will be stored, one location for each input channel enabled.

The Output Data Address is the starting location in user V-memory where the analog output data values will be stored by ladder code or external device, one location for each output channel enabled.

Since the IBox logic only executes on the first scan, the instruction cannot have any input logic.

Analog Input/Output Combo Module Pointer Setup

ANLGCMB

IB-462

| | |
|----------------------------------|------|
| Base # (K0-Local) | K0 |
| Slot # | K0 |
| Number of Input Channels | K1 |
| Input Data Format (0-BCD 1-BIN) | K0 |
| Input Data Address | V400 |
| Number of Output Channels | K1 |
| Output Data Format (0-BCD 1-BIN) | K0 |
| Output Data Address | V400 |

ANLGCMB Parameters

- Base # (K0-Local): specifies which base the module is in.
- Slot #: specifies which slot is occupied by the analog module.
- Number of Input Channels: specifies the number of analog input channels to scan.
- Input Data Format (0-BCD 1-BIN): specifies the analog input data format (BCD or Binary) - the binary format may be used for displaying data on some OI panels.
- Input Data Address: specifies the starting V-memory location that will be used to store the analog input data.
- Number of Output Channels: specifies the number of analog output channels that will be used.
- Output Data Format (0-BCD 1-BIN): specifies the format of the analog output data (BCD or Binary).
- Output Data Address: specifies the starting V-memory location that will be used to source the analog output data.



NOTE The ANLGCMB instruction does not currently support the F2-8AD4DA-1 or F2-8AD4DA-2.

| Parameter | | DL205 Range |
|----------------------------------|---|-------------------------------------|
| Base # (K0-Local) | K | K0-3 |
| Slot # | K | K0-7 |
| Number of Input Channels | K | K1-8 |
| Input Data Format (0-BCD 1-BIN) | K | BCD: K0; Binary: K1 |
| Input Data Address | V | See DL205 V-memory map - Data Words |
| Number of Output Channels | K | K1-8 |
| Output Data Format (0-BCD 1-BIN) | K | BCD: K0; Binary: K1 |
| Output Data Address | V | See DL205 V-memory map - Data Words |

ANLGCMB Example

In the following example, the ANLGCMB instruction is used to set up the pointer method for an analog I/O combination module that is installed in option slot 2. Four input channels are enabled and the analog data will be written to V2000 - V2003 in BCD format. Two output channels are enabled and the analog values will be read from V2100 - V2101 in BCD format.

| | | | |
|---|--|---|---------------|
| 1 | No permissive contact or input logic is used with this instruction | Analog Input/Output Combo Module Pointer Setup | |
| | | ANLGCMB | IB-462 |
| | | Base # (K0-Local) | K0 |
| | | Slot # | K1 |
| | | Number of Input Channels | K4 |
| | | Input Data Format (0-BCD 1-BIN) | K0 |
| | | Input Data Address | V2000 |
| | | Number of Output Channels | K2 |
| | | Output Data Format (0-BCD 1-BIN) | K0 |
| | | Output Data Address | V2100 |



NOTE: An Analog I/O IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.

Analog Input Module Pointer Setup (ANLGIN) (IB-460)

Analog Input Module Pointer Setup generates the logic to configure the pointer method for one analog input module on the first PLC scan following a Program to Run transition.

230

240

250-1

260

This IBox determines the data format and Pointer addresses based on the CPU type, the Base#, and the Slot#.

The Input Data Address is the starting location in user V-memory where the analog input data values will be stored, one location for each input channel enabled.

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

Analog Input Module Pointer Setup

ANLGIN

IB-460

Base # (K0-Local)

K0

Slot #

K0

Number of Input Channels

K1

Input Data Format (0-BCD 1-BIN)

K0

Input Data Address

V400

Since this logic only executes on the first scan, this IBox cannot have any input logic.

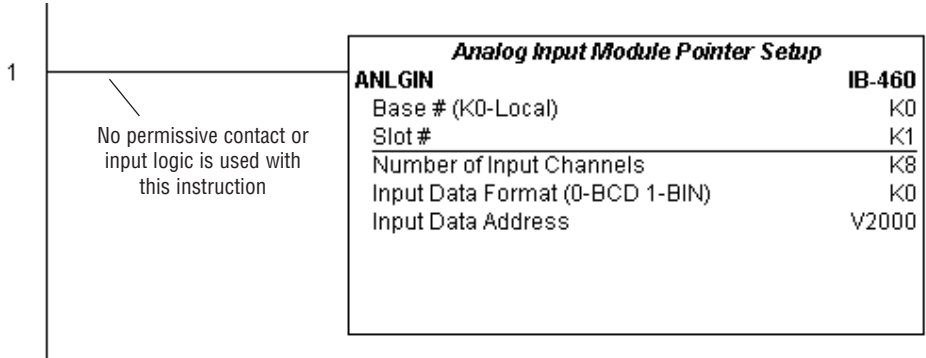
ANLGIN Parameters

- Base # (K0-Local): specifies which base the analog module is in.
- Slot #: specifies which PLC slot is occupied by the analog module.
- Number of Input Channels: specifies the number of input channels to scan.
- Input Data Format (0-BCD 1-BIN): specifies the analog input data format (BCD or Binary) - the binary format may be used for displaying data on some OI panels.
- Input Data Address: specifies the starting V-memory location that will be used to store the analog input data.

| Parameter | | DL205 Range |
|---------------------------------|---|-------------------------------------|
| Base # (K0-Local) | K | K0-3 |
| Slot # | K | K0-7 |
| Number of Input Channels | K | K1-8 |
| Input Data Format (0-BCD 1-BIN) | K | BCD: K0; Binary: K1 |
| Input Data Address | V | See DL205 V-memory map - Data Words |

ANLGIN Example

In the following example, the ANLGIN instruction is used to set up the pointer method for an analog input module that is installed in option slot 1. Eight input channels are enabled and the analog data will be written to V2000 - V2007 in BCD format.



NOTE: An Analog I/O IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.

Analog Output Module Pointer Setup (ANLGOUT) (IB-461)

Analog Output Module Pointer Setup generates the logic to configure the pointer method for one analog output module on the first PLC scan following a Program to Run transition.

This IBox determines the data format and Pointer addresses based on the CPU type, the Base#, and the Slot#.

The Output Data Address is the starting location in user V-memory where the analog output data values will be placed by ladder code or external device, one location for each output channel enabled.

Since this logic only executes on the first scan, this IBox cannot have any input logic.

Analog Output Module Pointer Setup IB-461

ANLGOUT

| | |
|----------------------------------|------|
| Base # (K0-Local) | K0 |
| Slot # | K0 |
| Number of Output Channels | K1 |
| Output Data Format (0-BCD 1-BIN) | K0 |
| Output Data Address | V400 |

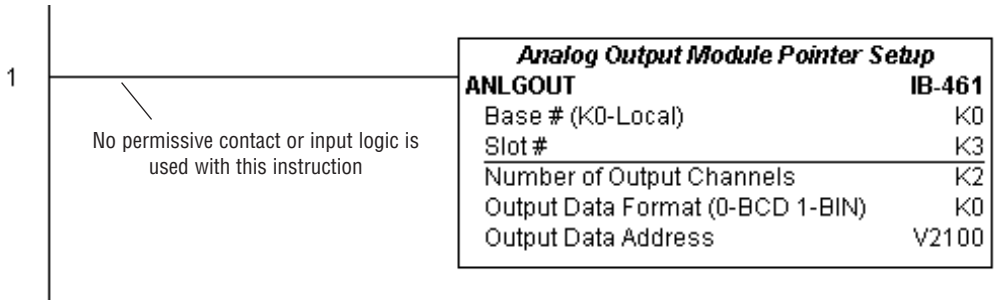
ANLGOUT Parameters

- Base # (K0-Local): specifies which base the analog module is in
- Slot #: specifies which PLC slot is occupied by the analog module
- Number of Output Channels: specifies the number of analog output channels that will be used
- Output Data Format (0-BCD 1-BIN): specifies the format of the analog output data (BCD or Binary)
- Output Data Address: specifies the starting V-memory location that will be used to source the analog output data

| Parameter | | DL205 Range |
|----------------------------------|---|-------------------------------------|
| Base # (K0-Local) | K | K0-3 |
| Slot # | K | K0-7 |
| Number of Output Channels | K | K1-8 |
| Output Data Format (0-BCD 1-BIN) | K | BCD: K0; Binary: K1 |
| Output Data Address | V | See DL205 V-memory map - Data Words |

ANLGOUT Example

In the following example, the ANLGOUT instruction is used to set up the pointer method for an analog output module that is installed in option slot 3. Two output channels are enabled and the analog data will be read from V2100 - V2101 in BCD format.



NOTE: An Analog I/O IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.

Analog Scale 12-Bit BCD to BCD (ANSCL) (IB-423)

Analog Scale 12-Bit BCD to BCD scales a 12-bit BCD analog value (0 to 4095 BCD) into BCD engineering units. You specify the engineering unit high value (when raw is 4095), and the engineering low value (when raw is 0), and the output V-memory address where you want to place the scaled engineering unit value. The engineering units are generated as BCD and can be the full range of 0 to 9999 (see ANSCLB - Analog Scale 12-Bit Binary to Binary if your raw units are in Binary format).

Note that this IBox only works with unipolar unsigned raw values. It does NOT work with bipolar or sign plus magnitude raw values.

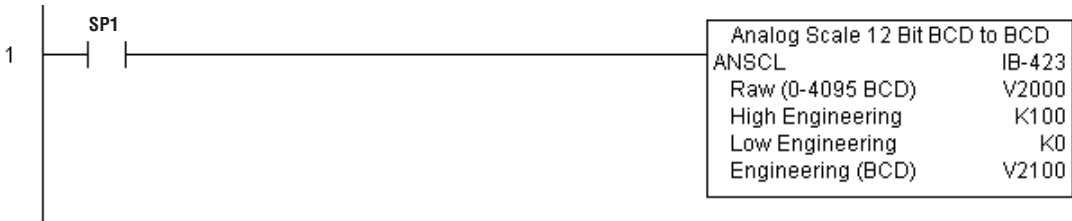
ANSCL Parameters

- Raw (0 to 4095 BCD): specifies the V-memory location of the unipolar unsigned raw 0 to 4095 unscaled value
- High Engineering: specifies the high engineering value when the raw input is 4095
- Low Engineering: specifies the low engineering value when the raw input is 0
- Engineering (BCD): specifies the V-memory location where the scaled engineering BCD value will be placed

ANSCL Example

| Parameter | | DL205 Range |
|-------------------|-----|-------------------------------------|
| Raw (0-4095 BCD) | V,P | See DL205 V-memory map - Data Words |
| High Engineering | K | K0-9999 |
| Low Engineering | K | K0-9999 |
| Engineering (BCD) | V,P | See DL205 V-memory map - Data Words |

In the following example, the ANSCL instruction is used to scale a raw value (0 to 4095 BCD) that is in V2000. The engineering scaling range is set 0 to 100 (low engineering value - high engineering value). The scaled value will be placed in V2100 in BCD format.



Analog Scale 12-Bit Binary to Binary (ANSCLB) (IB-403)

Analog Scale 12-Bit Binary to Binary scales a 12-bit binary analog value (0 to 4095 decimal) into binary (decimal) engineering units. You specify the engineering unit high value (when raw is 4095), and the engineering low value (when raw is 0), and the output V-memory address where you want to place the scaled engineering unit value. The engineering units are generated as binary and can be the full range of 0 to 65535 (see ANSCL - Analog Scale 12-Bit BCD to BCD if your raw units are in BCD format).

Note that this IBox only works with unipolar unsigned raw values. It does NOT work with bipolar, sign plus magnitude, or signed 2's complement raw values.

5

ANSCLB Parameters

- Raw (12-bit binary): specifies the V-memory location of the unipolar unsigned raw decimal unscaled value (12-bit binary = 0 to 4095 decimal)
- High Engineering: specifies the high engineering value when the raw input is 4095 decimal
- Low Engineering: specifies the low engineering value when the raw input is 0 decimal
- Engineering (binary): specifies the V-memory location where the scaled engineering decimal value will be placed

ANSCLB Example

| Parameter | | DL205 Range |
|----------------------|-----|-------------------------------------|
| Raw (12-bit binary) | V,P | See DL205 V-memory map - Data Words |
| High Engineering | K | K0-65535 |
| Low Engineering | K | K0-65535 |
| Engineering (binary) | V,P | See DL205 V-memory map - Data Words |

In the following example, the ANSCLB instruction is used to scale a raw value (0 to 4095 binary) that is in V2000. The engineering scaling range is set 0 to 1000 (low engineering value - high engineering value). The scaled value will be placed in V2100 in binary format.



Filter Over Time - BCD (FILTER) (IB-422)

Filter Over Time BCD will perform a first-order filter on the Raw Data on a defined time interval. The equation is:

$$\text{New} = \text{Old} + [(\text{Raw} - \text{Old}) / \text{FDC}]$$

where,

New: New Filtered Value

Old: Old Filtered Value

FDC: Filter Divisor Constant

Raw: Raw Data

The Filter Divisor Constant is an integer in the range K1 to K100, such that if it equaled K1 then no filtering would be done.

The rate at which the calculation is performed is specified by time in hundredths of a second (0.01 seconds) as the Filter Freq Time parameter. Note that this Timer instruction is embedded in the IBox and must NOT be used anywhere else in your program. Power flow controls whether the calculation is enabled. If it is disabled, the Filter Value is not updated. On the first scan from Program to Run mode, the Filter Value is initialized to 0 to give the calculation a consistent starting point.

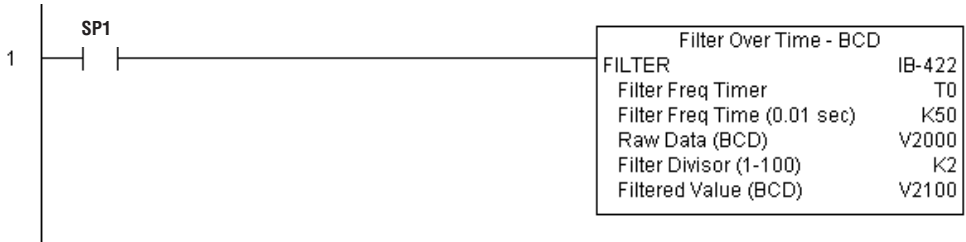
FILTER Parameters

- Filter Frequency Timer: specifies the Timer (T) number which is used by the Filter instruction.
- Filter Frequency Time (0.01sec): specifies the rate at which the calculation is performed.
- Raw Data (BCD): specifies the V-memory location of the raw unfiltered BCD value.
- Filter Divisor (1 to 100): this constant is used to control the filtering effect. A larger value will increase the smoothing effect of the filter. A value of 1 results with no filtering.
- Filtered Value (BCD): specifies the V-memory location where the filtered BCD value will be placed.

| Parameter | | DL205 Range |
|----------------------------------|---|-------------------------------------|
| Filter Frequency Timer | T | T0-377 |
| Filter Frequency Time (0.01 sec) | K | K0-9999 |
| Raw Data (BCD) | V | See DL205 V-memory map - Data Words |
| Filter Divisor (1-100) | K | K1-100 |
| Filtered Value (BCD) | V | See DL205 V-memory map - Data Words |

FILTER Example

In the following example, the Filter instruction is used to filter a BCD value that is in V2000. Timer(T0) is set to 0.5 sec, the rate at which the filter calculation will be performed. The filter constant is set to 2. A larger value will increase the smoothing effect of the filter. A value of 1 results with no filtering. The filtered value will be placed in V2100.



Filter Over Time - Binary (FILTERB) (IB-402)

Filter Over Time in Binary (decimal) will perform a first-order filter on the Raw Data on a defined time interval. The equation is:

$$\text{New} = \text{Old} + [(\text{Raw} - \text{Old}) / \text{FDC}]$$
 where

New: New Filtered Value

Old: Old Filtered Value

FDC: Filter Divisor Constant

Raw: Raw Data

The Filter Divisor Constant is an integer in the range K1 to K100, such that if it equaled K1 then no filtering would be done.

The rate at which the calculation is performed is specified by time in hundredths of a second (0.01 seconds) as the Filter Freq Time parameter. Note that this Timer instruction is embedded in the IBox and must NOT be used anywhere else in your program. Power flow controls whether the calculation is enabled. If it is disabled, the Filter Value is not updated. On the first scan from Program to Run mode, the Filter Value is initialized to 0 to give the calculation a consistent starting point.

Filter Over Time - Binary
IB-402

FILTERB

Filter Freq Timer: T0

Filter Freq Time (0.01 sec): K0

Raw Data (Binary): V2000

Filter Divisor (1-100): K1

Filtered Value (Binary): V3000

FILTERB Parameters

- Filter Frequency Timer: specifies the Timer (T) number that is used by the Filter instruction.
- Filter Frequency Time (0.01sec): specifies the rate at which the calculation is performed.
- Raw Data (Binary): specifies the V-memory location of the raw unfiltered binary (decimal) value.
- Filter Divisor (1 to 100): this constant is used to control the filtering effect. A larger value will increase the smoothing effect of the filter. A value of 1 results with no filtering.
- Filtered Value (Binary): specifies the V-memory location where the filtered binary (decimal) value will be placed.

| Parameter | | DL205 Range |
|----------------------------------|---|-------------------------------------|
| Filter Frequency Timer | T | T0-377 |
| Filter Frequency Time (0.01 sec) | K | K0-9999 |
| Raw Data (Binary) | V | See DL205 V-memory map - Data Words |
| Filter Divisor (1-100) | K | K1-100 |
| Filtered Value (Binary) | V | See DL205 V-memory map - Data Words |

FILTERB Example

In the following example, the FILTERB instruction is used to filter a binary value that is in V2000. Timer(T1) is set to 0.5 sec, the rate at which the filter calculation will be performed. The filter constant is set to 3. A larger value will increase the smoothing effect of the filter. A value of 1 results with no filtering. The filtered value will be placed in V2100



Hi/Low Alarm - BCD (HILOAL) (IB-421)

Hi/Low Alarm - BCD monitors a BCD value V-memory location and sets four possible alarm states, High-High, High, Low, and Low-Low whenever the IBox has power flow. You enter the alarm thresholds as constant (K) BCD values (K0-K9999) and/or BCD value V-memory locations.

You must ensure that threshold limits are valid, that is $HH \geq H > L \geq LL$. Note that when the High-High or Low-Low alarm condition is true, that the High and Low alarms will also be set, respectively. This means you may use the same threshold limit and same alarm bit for the High-High and the High alarms in case you only need one “High” alarm. Also note that the boundary conditions are inclusive. That is, if the Low boundary is K50, and the Low-Low boundary is K10, and if the Monitoring Value equals 10, then the Low Alarm AND the Low-Low alarm will both be ON. If there is no power flow to the IBox, then all alarm bits will be turned off regardless of the value of the Monitoring Value parameter.

Hi/Low Alarm - BCD IB-421

| | |
|------------------------|-----|
| Monitoring Value (BCD) | TA0 |
| High-High Limit | TA0 |
| High-High Alarm | C0 |
| High Limit | TA0 |
| High Alarm | C0 |
| Low Limit | TA0 |
| Low Alarm | C0 |
| Low-Low Limit | TA0 |
| Low-Low Alarm | C0 |

HILOAL Parameters

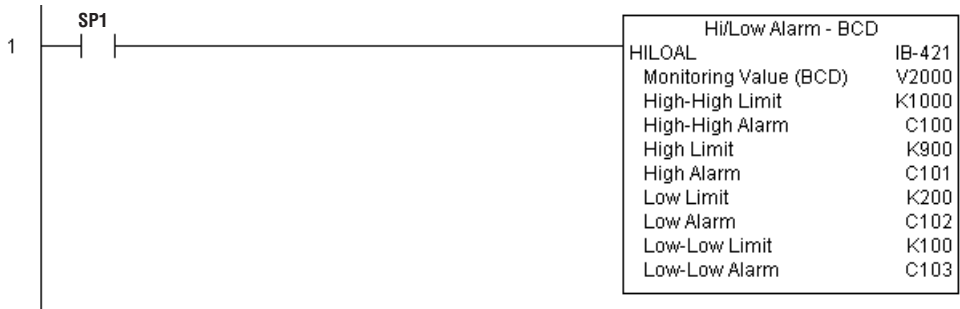
- Monitoring Value (BCD): specifies the V-memory location of the BCD value to be monitored
- High-High Limit: V-memory location or constant specifies the high-high alarm limit
- High-High Alarm: On when the high-high limit is reached
- High Limit: V-memory location or constant specifies the high alarm limit
- High Alarm: On when the high limit is reached
- Low Limit: V-memory location or constant specifies the low alarm limit
- Low Alarm: On when the low limit is reached
- Low-Low Limit: V-memory location or constant specifies the low-low alarm limit
- Low-Low Alarm: On when the low-low limit is reached

| Parameter | DL205 Range |
|-----------------------------------|---|
| Monitoring Value (BCD) V | See DL205 V-memory map - Data Words |
| High-High Limit V, K | K0-9999; or see DL205 V-memory map - Data Words |
| High-High Alarm X, Y, C, GX,GY, B | See DL205 V-memory map |
| High Limit V, K | K0-9999; or see DL205 V-memory map - Data Words |
| High Alarm X, Y, C, GX,GY, B | See DL205 V-memory map |
| Low Limit V, K | K0-9999; or see DL205 V-memory map - Data Words |
| Low Alarm X, Y, C, GX,GY, B | See DL205 V-memory map |
| Low-Low Limit V, K | K0-9999; or see DL205 V-memory map - Data Words |
| Low-Low Alarm X, Y, C, GX,GY, B | See DL205 V-memory map |

HILOAL Example

In the following example, the HILOAL instruction is used to monitor a BCD value that is in V2000. If the value in V2000 meets/exceeds the High limit of K900, C101 will turn on. If the value continues to increase to meet/exceed the High-High limit, C100 will turn on. Both bits would be on in this case. The High and High-High limits and alarms can be set to the same value if one “High” limit or alarm is desired to be used.

If the value in V2000 meets or falls below the Low limit of K200, C102 will turn on. If the value continues to decrease to meet or fall below the Low-Low limit of K100, C103 will turn on. Both bits would be on in this case. The Low and Low-Low limits and alarms can be set to the same value if one “Low” limit or alarm is desired to be used.



Hi/Low Alarm - Binary (HILOALB) (IB-401)

Hi/Low Alarm - Binary monitors a binary (decimal) V-memory location and sets four possible alarm states, High-High, High, Low, and Low-Low whenever the IBox has power flow. You enter the alarm thresholds as constant (K) decimal values (K0-K65535) and/or binary (decimal) V-memory locations.

You must ensure that threshold limits are valid, that is $HH \geq H > L \geq LL$. Note that when the High-High or Low-Low alarm condition is true, that the High and Low alarms will also be set, respectively. This means you may use the same threshold limit and same alarm bit for the High-High and the High alarms in case you only need one “High” alarm. Also note that the boundary conditions are inclusive. That is, if the Low boundary is K50, and the Low-Low boundary is K10, and if the Monitoring Value equals 10, then the Low Alarm AND the Low-Low alarm will both be ON. If there is no power flow to the IBox, then all alarm bits will be turned off regardless of the value of the Monitoring Value parameter.

HILOALB Parameters

- Monitoring Value (Binary): specifies the V-memory location of the Binary value to be monitored
- High-High Limit: V-memory location or constant specifies the High-High alarm limit
- High-High Alarm: On when the High-High limit is reached
- High Limit: V-memory location or constant specifies the High alarm limit
- High Alarm: On when the High limit is reached
- Low Limit: V-memory location or constant specifies the Low alarm limit
- Low Alarm: On when the Low limit is reached
- Low-Low Limit: V-memory location or constant specifies the Low-Low alarm limit
- Low-Low Alarm: On when the Low-Low limit is reached

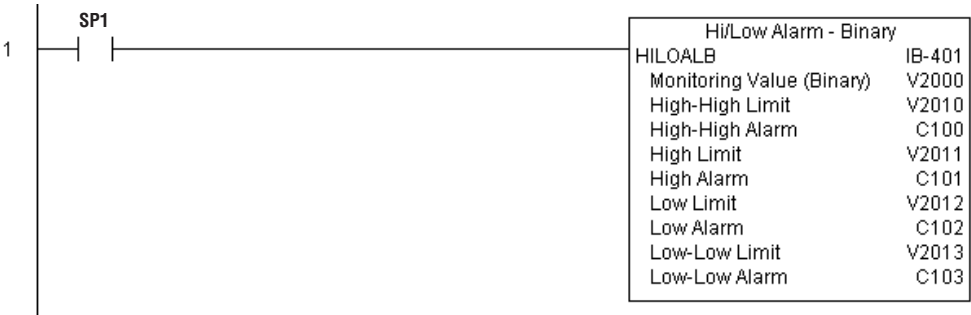
| HILOALB | | IB-401 |
|---------------------------|-----|--------|
| Monitoring Value (Binary) | TA0 | • |
| High-High Limit | TA0 | • |
| High-High Alarm | C0 | • |
| High Limit | TA0 | • |
| High Alarm | C0 | • |
| Low Limit | TA0 | • |
| Low Alarm | C0 | • |
| Low-Low Limit | TA0 | • |
| Low-Low Alarm | C0 | • |

| Parameter | | DL205 Range |
|---------------------------|-------------------|--|
| Monitoring Value (Binary) | V | See DL205 V-memory map - Data Words |
| High-High Limit | V, K | K0-65535; or see DL205 V-memory map - Data Words |
| High-High Alarm | X, Y, C, GX,GY, B | See DL205 V-memory map |
| High Limit | V, K | K0-65535; or see DL205 V-memory map - Data Words |
| High Alarm | X, Y, C, GX,GY, B | See DL205 V-memory map |
| Low Limit | V, K | K0-65535; or see DL205 V-memory map - Data Words |
| Low Alarm | X, Y, C, GX,GY,B | See DL205 V-memory map |
| Low-Low Limit | V, K | K0-65535; or see DL205 V-memory map - Data Words |
| Low-Low Alarm | X, Y, C, GX,GY, B | See DL205 V-memory map |

HILOALB Example

In the following example, the HILOALB instruction is used to monitor a binary value that is in V2000. If the value in V2000 meets/exceeds the High limit of the binary value in V2011, C101 will turn on. If the value continues to increase to meet/exceed the High-High limit value in V2010, C100 will turn on. Both bits would be on in this case. The High and High-High limits and alarms can be set to the same V-memory location/value if one “High” limit or alarm is desired to be used.

If the value in V2000 meets or falls below the low limit of the binary value in V2012, C102 will turn on. If the value continues to decrease to meet or fall below the Low-Low limit in V2013, C103 will turn on. Both bits would be on in this case. The Low and Low-Low limits and alarms can be set to the same V-memory location/value if one “Low” limit or alarm is desired to be used.



Off Delay Timer (OFFDTMR) (IB-302)

Off Delay Timer will delay the “turning off” of the Output parameter by the specified Off Delay Time (up to 99.99 seconds) based on the power flow into the IBox. Once the IBox receives power, the Output bit will turn on immediately. When the power flow to the IBox turns off, the Output bit WILL REMAIN ON for the specified amount of time (in hundredths of a second). Once the Off Delay Time has expired, the output will turn Off. If the power flow to the IBox comes back on BEFORE the Off Delay Time, then the timer is RESET and the Output will remain On - so you must continuously have NO power flow to the IBox for AT LEAST the specified Off Delay Time before the Output will turn Off.

This IBox utilizes a Timer resource (TMRF), which cannot be used anywhere else in your program.

- 230
- 240
- 250-1
- 260

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

OFFDTMR Parameters

- Timer Number: specifies the Timer (TMRF) number which is used by the OFFDTMR instruction
- Off Delay Time (0.01sec): specifies how long the Output will remain on once power flow to the Ibox is removed (up to 99.99 seconds).
- Output: specifies the output that will be delayed “turning off” by the Off Delay Time.

Off Delay Timer

IB-302

OFFDTMR

Timer Number

Off Delay Time (0.01 sec)

Output

T0

TA0

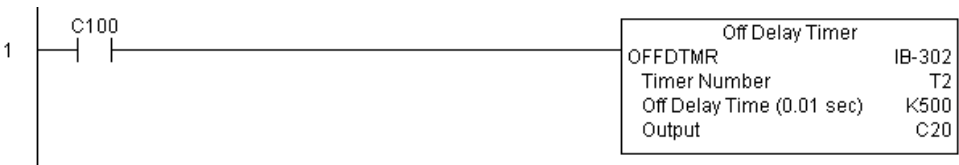
C0

| Parameter | | DL205 Range |
|----------------|-------------------|--|
| Timer Number | T | T0-377 |
| Off Delay Time | K,V | K0-9999; See DL205 V-memory map - Data Words |
| Output | X, Y, C, GX,GY, B | See DL205 V-memory map |

OFFDTMR Example

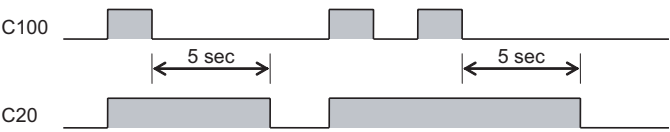
In the following example, the OFFDTMR instruction is used to delay the “turning off” of output C20. Timer 2 (T2) is set to 5 seconds, the “off-delay” period.

When C100 turns on, C20 turns on and will remain on while C100 is on. When C100 turns off, C20 will remain on for the specified Off Delay Time (5 secs), and then turn off.



5

Example timing diagram



On Delay Timer (ONDTMR) (IB-301)

On Delay Timer will delay the “turning on” of the Output parameter by the specified amount of time (up to 99.99 seconds) based on the power flow into the IBox. Once the IBox loses power, the Output is turned off immediately. If the power flow turns off BEFORE the On Delay Time, then the timer is RESET and the Output is never turned on, so you must have continuous power flow to the IBox for at least the specified On Delay Time before the Output turns On.

This IBox utilizes a Timer resource (TMRF), which cannot be used anywhere else in your program.

ONDTMR Parameters

- Timer Number: specifies the Timer (TMRF) number which is used by the ONDTMR instruction
- On Delay Time (0.01sec): specifies how long the Output will remain on once power flow to the Ibox is removed (up to 99.99 seconds).
- Output: specifies the output that will be delayed “turning on” by the On Delay Time.

| Parameter | DL205 Range |
|---------------|--|
| Timer Number | T0-377 |
| On Delay Time | K0-9999; See DL205 V-memory map - Data Words |
| Output | See DL205 V-memory map |

ONDTMR Example

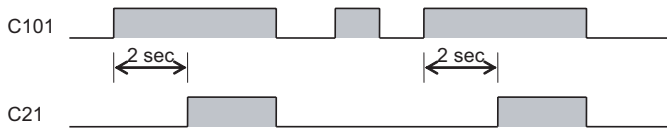
In the following example, the ONDTMR instruction is used to delay the “turning on” of output C21. Timer 1 (T1) is set to 2 seconds, the “on-delay” period.

When C101 turns on, C21 is delayed turning on by 2 seconds. When C101 turns off, C21 turns off immediately.



5

Example timing diagram



One Shot (ONESHOT) (IB-303)

One Shot will turn on the given bit output parameter for one scan on an OFF to ON transition of the power flow into the IBox. This IBox is simply a different name for the PD Coil (Positive Differential).

230

240

250-1

260

ONESHOT Parameters

- Discrete Output: specifies the output that will be on for one scan

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

One Shot

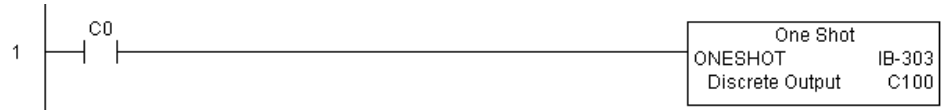
ONESHOT IB-303

Discrete Output C0

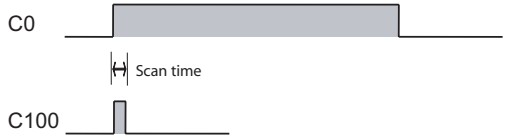
| Parameter | DL205 Range |
|-------------------------|------------------------|
| Discrete Output X, Y, C | See DL205 V-memory map |

ONESHOT Example

In the following example, the ONESHOT instruction is used to turn C100 on for one PLC scan after C0 goes from an off to on transition. The input logic must produce an off to on transition to execute the One Shot instruction.



Example Timing Diagram



Push On/Push Off Circuit (PONOFF) (IB-300)

Push On/Push Off Circuit toggles an output state whenever its input power flow transitions from off to on. Requires an extra bit parameter for scan-to-scan state information. This extra bit must NOT be used anywhere else in the program. This is also known as a “flip-flop circuit.”

230

240

250-1

260

PONOFF Parameters

- Discrete Input: specifies the input that will toggle the specified output
- Discrete Output: specifies the output that will be “turned on/off” or toggled
- Internal State: specifies a work bit that is used by the instruction

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

Push On/Push Off Circuit

PONOFF

IB-300

Discrete Input

C0

Discrete Output

C0

Internal State

C0

| Parameter | | DL205 Range |
|-----------------|----------------------------|------------------------|
| Discrete Input | X,Y,C,S,T,CT,GX,GY,SP,B,PB | See DL205 V-memory map |
| Discrete Output | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Internal State | X, Y, C | See DL205 V-memory map |

PONOFF Example

In the following example, the PONOFF instruction is used to control the on and off states of the output C20 with a single input C10. When C10 is pressed once, C20 turns on. When C10 is pressed again, C20 turns off. C100 is an internal bit used by the instruction.



NOTE: Neither a permissive nor input logic is required with this instruction.

Move Single Word (MOVEW) (IB-200)

Move Single Word moves (copies) a word to a memory location directly or indirectly via a pointer, either as a HEX constant, from a memory location, or indirectly through a pointer.

☒ 230

☒ 240

☒ 250-1

☒ 260

MOVEW Parameters

- From WORD: specifies the word that will be moved to another location
- To WORD: specifies the location to which where the “From WORD” will be moved

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

☒ ☒ ☒

Move Single Word

MOVEW

IB-200

From WORD

TA0

To WORD

TA0

| Parameter | | DL205 Range |
|-----------|-------|--|
| From WORD | V,P,K | K0-FFFF; See DL205 V-memory map - Data Words |
| To WORD | V,P | See DL205 V-memory map - Data Words |

MOVEW Example

In the following example, the MOVEW instruction is used to move 16 bits of data from V2000 to V3000 when C100 turns on.



Move Double Word (MOVED) (IB-201)

Move Double Word moves (copies) a double word to two consecutive memory locations directly or indirectly via a pointer, either as a double HEX constant, from a double memory location, or indirectly through a pointer to a double memory location.

 230

 240




 250-1

 260

MOVED Parameters

- From DWORD: specifies the double word that will be moved to another location
- To DWORD: specifies the location to which where the “From DWORD” will be moved

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |



Move Double Word

MOVED

IB-201

From DWORD

TA0

To DWORD

TA0

| Parameter | | DL205 Range |
|------------|-------|---|
| From DWORD | V,P,K | K0-FFFFFFF; See DL205 V-memory map - Data Words |
| To DWORD | V,P | See DL205 V-memory map - Data Words |

MOVED Example

In the following example, the MOVED instruction is used to move 32 bits of data from V2000 and V2001 to V3000 and V3001 when C100 turns on.



BCD to Real with Implied Decimal Point (BCDTOR) (IB-560)

BCD to Real with Implied Decimal Point converts the given 4-digit WORD BCD value to a Real number, with the implied number of decimal points (K0-K4).

For example, BCDTOR K1234 with an implied number of decimal points equal to K1, would yield R123.4

- 230
- 240
- 250-1
- 260

BCDTOR Parameters

- Value (WORD BCD): specifies the word or constant that will be converted to a Real number
- Number of Decimal Points: specifies the number of implied decimal points in the Result DWORD
- Result (DWORD REAL): specifies the location where the Real number will be placed

BCD to Real with Implied Decimal Point

BCDTOR IB-560

Value (WORD BCD) TA0

Number of Decimal Points K0

Result (DWORD REAL) V400

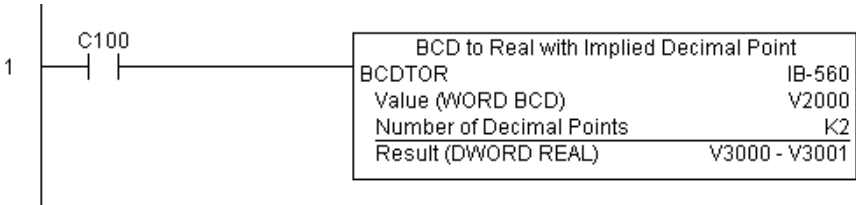
| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

| Parameter | DL205 Range |
|--------------------------|--|
| Value (WORD BCD) | V,P,K K0-9999; See DL205 V-memory map - Data Words |
| Number of Decimal Points | K K0-4 |
| Result (DWORD REAL) | V See DL205 V-memory map - Data Words |

BCDTOR Example

In the following example, the BCDTOR instruction is used to convert the 16-bit data in V2000 from a 4-digit BCD data format to a 32-bit REAL (floating point) data format and store into V3000 and V3001 when C100 turns on.

K2 in the Number of Decimal Points implies the data will have two digits to the right of the decimal point.



Double BCD to Real with Implied Decimal Point (BCDTORD) (IB-562)

Double BCD to Real with Implied Decimal Point converts the given 8-digit DWORD BCD value to a Real number, given an implied number of decimal points (K0-K8).

For example, BCDTORD K12345678 with an implied number of decimal points equal to K5, would yield R123.45678

230
240
250-1
260

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

BCDTORD Parameters

- Value (DWORD BCD): specifies the Dword or constant that will be converted to a Real number
- Number of Decimal Points: specifies the number of implied decimal points in the Result DWORD
- Result (DWORD REAL): specifies the location where the Real number will be placed

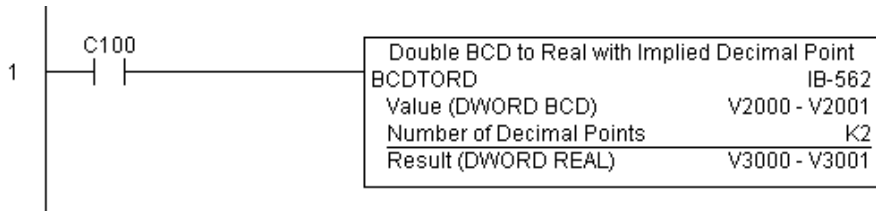
| Double BCD to Real with Implied Decimal Point | |
|---|--------|
| BCDTORD | IB-562 |
| Value (DWORD BCD) | TA0 |
| Number of Decimal Points | K0 |
| Result (DWORD REAL) | V400 |

| Parameter | | DL205 Range |
|--------------------------|-------|--|
| Value (DWORD BCD) | V,P,K | K0-99999999; See DL205 V-memory map - Data Words |
| Number of Decimal Points | K | K0-8 |
| Result (DWORD REAL) | V | See DL205 V-memory map - Data Words |

BCDTORD Example

In the following example, the BCDTORD instruction is used to convert the 32-bit data in V2000 from an 8-digit BCD data format to a 32-bit REAL (floating point) data format and store into V3000 and V3001 when C100 turns on.

K2 in the Number of Decimal Points implies the data will have two digits to the right of the decimal point.

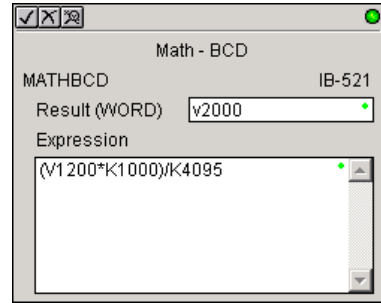


Math - BCD (MATHBCD) (IB-521)

- ☐ 230
- ☐ 240
- ☒ 250-1
- ☒ 260

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

Math - BCD Format lets you enter complex mathematical expressions like you would in Visual Basic, Excel, or C++ to do complex calculations, nesting parentheses up to 4 levels deep. In addition to + - * /, you can do Modulo (% aka Remainder), Bit-wise And (&) Or (|) Xor (^), and some BCD functions - Convert to BCD (BCD), Convert to Binary (BIN), BCD Complement (BCDCPL), Convert from Gray Code (GRAY), Invert Bits (INV), and BCD/ HEX to Seven Segment Display (SEG).



Example: $((V2000 + V2001) / (V2003 - K100)) * GRAY(V3000 \& K001F)$

Every V-memory reference MUST be to a single-word BCD formatted value. Intermediate results can go up to 32-bit values, but as long as the final result fits in a 16-bit BCD word, the calculation is valid. Typical example of this is scaling using multiply then divide, $(V2000 * K1000) / K4095$. The multiply term most likely will exceed 9999 but fits within 32 bits. The divide operation will divide 4095 into the 32-bit accumulator, yielding a result that will always fit in 16 bits.

You can reference binary V-memory values by using the BCD conversion function on a V-memory location but NOT an expression. That is $BCD(V2000)$ is okay and will convert V2000 from Binary to BCD, but $BCD(V2000 + V3000)$ will add V2000 as BCD, to V3000 as BCD, then interpret the result as Binary and convert it to BCD - NOT GOOD.

Also, the final result is a 16-bit BCD number and so you could do BIN around the entire operation to store the result as Binary.

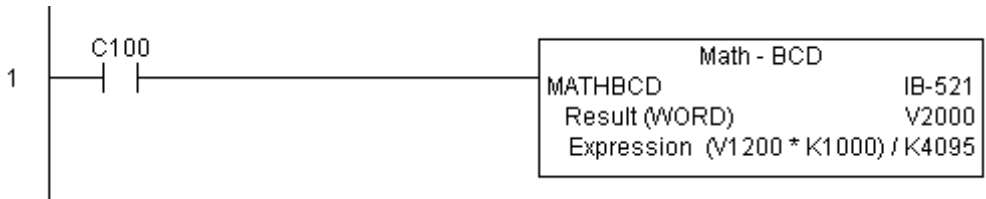
MATHBCD Parameters

- Result (WORD): specifies the location where the BCD result of the mathematical expression will be placed (result must fit into 16-bit single V-memory location).
- Expression: specifies the mathematical expression to be executed and the result is stored in specified Result (WORD). Each V-memory location used in the expression must be in BCD format.

| Parameter | | DL205 Range |
|-------------|---|-------------------------------------|
| WORD Result | V | See DL205 V-memory map - Data Words |
| Expression | | Text |

MATHBCD Example

In the following example, the MATHBCD instruction is used to calculate the math expression which multiplies the BCD value in V1200 by 1000, then divides by 4095 and loads the resulting value in V2000 when C100 turns on.



Math - Binary (MATHBIN) (IB-501)

- ☐ 230
- ☐ 240
- ☒ 250-1
- ☒ 260

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

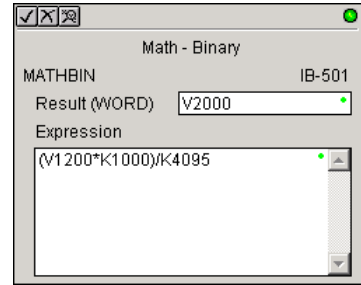
Math - Binary Format lets you enter complex mathematical expressions like you would in Visual Basic, Excel, or C++ to do complex calculations, nesting parentheses up to 4 levels deep. In addition to + - * /, you can do Modulo (% aka Remainder), Shift Right (>>) and Shift Left (<<), Bit-wise And (&) Or (|) Xor (^), and some binary functions - Convert to BCD (BCD), Convert to Binary (BIN), Decode Bits (DECO), Encode Bits (ENCO), Invert Bits (INV), HEX to Seven Segment Display (SEG), and Sum Bits (SUM).

Example: $((V2000 + V2001) / (V2003 - K10)) * SUM(V3000 \& K001F)$

Every V-memory reference MUST be to a single-word binary formatted value. Intermediate results can go up to 32-bit values, but as long as the final result fits in a 16-bit binary word, the calculation is valid. Typical example of this is scaling using multiply then divide, $(V2000 * K1000) / K4095$. The multiply term most likely will exceed 65535 but fits within 32 bits. The divide operation will divide 4095 into the 32-bit accumulator, yielding a result that will always fit in 16 bits.

You can reference BCD V-memory values by using the BIN conversion function on a V-memory location but NOT an expression. That is, $BIN(V2000)$ is okay and will convert V2000 from BCD to Binary, but $BIN(V2000 + V3000)$ will add V2000 as Binary, to V3000 as Binary, then interpret the result as BCD and convert it to Binary - NOT GOOD.

Also, the final result is a 16-bit binary number and so you could do BCD around the entire operation to store the result as BCD.



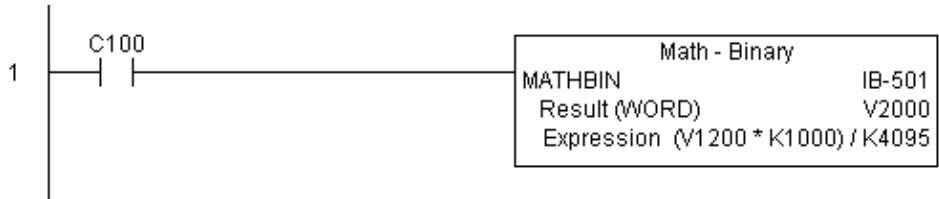
MATHBIN Parameters

- Result (WORD): specifies the location where the binary result of the mathematical expression will be placed (result must fit into 16-bit single V-memory location).
- Expression: specifies the mathematical expression to be executed and the result is stored in specified Result (WORD). Each V-memory location used in the expression must be in binary format.

| Parameter | DL205 Range |
|-------------|-------------|
| WORD Result | V |
| Expression | Text |

MATHBIN Example

In the following example, the MATHBIN instruction is used to calculate the math expression which multiplies the Binary value in V1200 by 1000 then divides by 4095 and loads the resulting value in V2000 when C100 turns on.

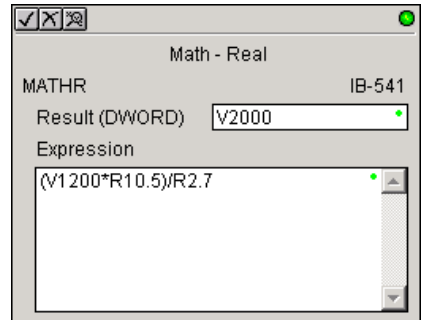


Math - Real (MATHR) (IB-541)

Math - Real Format lets you enter complex mathematical expressions like you would in Visual Basic, Excel, or C++ to do complex calculations, nesting parentheses up to 4 levels deep. In addition to + - * /, you can do Bit-wise And (&) Or (|) and Xor (^). The DL260 also supports several Real functions - Arc Cosine (ACOSR), Arc Sine (ASINR), Arc Tangent (ATANR), Cosine (COSR), Convert Radians to Degrees (DEGR), Invert Bits (INV), Convert Degrees to Radians (RADR), HEX to Seven Segment Display (SEG), Sine (SINR), Square Root (SQRT), and Tangent (TANR).

Example: $((V2000 + V2002) / (V2004 - R2.5)) * SINR(RADR(V3000 / R10.0))$

Every V-memory reference MUST be able to fit into a double-word Real formatted value.



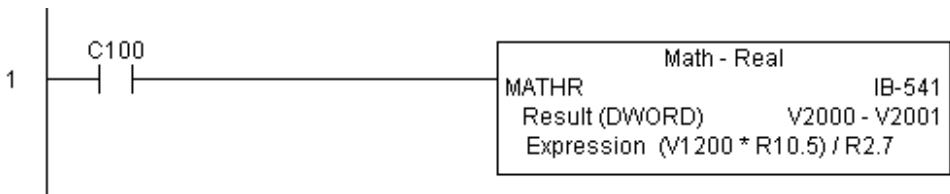
MATHR Parameters

- Result (DWORD): specifies the location where the Real result of the mathematical expression will be placed (result must fit into a double-word Real formatted location).
- Expression: specifies the mathematical expression to be executed and the result is stored in specified Result (DWORD) location. Each V-memory location used in the expression must be in Real format.

| Parameter | DL205 Range |
|--------------|-------------|
| DWORD Result | V |
| Expression | Text |

MATHR Example

In the following example, the MATHR instruction is used to calculate the math expression which multiplies the REAL (floating point) value in V1200 by 10.5 then divides by 2.7 and loads the resulting 32-bit value in V2000 and V2001 when C100 turns on.



Real to BCD with Implied Decimal Point and Rounding (RTOBCD) (IB-561)

Real to BCD with Implied Decimal Point and Rounding converts the absolute value of the given Real number to a 4-digit BCD number, compensating for an implied number of decimal points (K0-K4) and performs rounding.

230

240

250-1

260

For example, RTOBCD R56.74 with an implied number of decimal points equal to K1, would yield 567 BCD. If the implied number of decimal points was 0, then the function would yield 57 BCD (note that it rounded up).

If the Real number is negative, the Result will equal its positive, absolute value.

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

RTOBCD Parameters

- Value (DWORD Real): specifies the Real Dword location or number that will be converted and rounded to a BCD number with decimal points
- Number of Decimal Points: specifies the number of implied decimal points in the Result WORD
- Result (WORD BCD): specifies the location where the rounded/implied decimal points BCD value will be placed

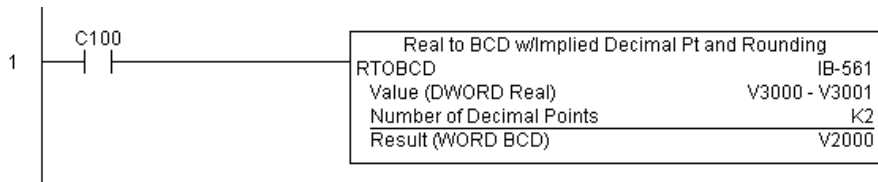
| Real to BCD w/Implied Decimal Pt and Rounding | |
|---|--------|
| RTOBCD | IB-561 |
| Value (DWORD Real) | TA0 |
| Number of Decimal Points | K0 |
| Result (WORD BCD) | V400 |

| Parameter | DL205 Range |
|--------------------------|--|
| Value (DWORD Real) | V,P,R R ; See DL205 V-memory map - Data Words |
| Number of Decimal Points | K K0-4 |
| Result (WORD BCD) | V See DL205 V-memory map - Data Words |

RTOBCD Example

In the following example, the RTOBCD instruction is used to convert the 32-bit REAL (floating point) data format in V3000 and V3001 to the 4-digit BCD data format and store in V2000 when C100 turns on.

K2 in the Number of Decimal Points implies the data will have two implied decimal points.



Real to Double BCD with Implied Decimal Point and Rounding (RTOBCDD) (IB-563)

230

240

250-1

260

Real to Double BCD with Implied Decimal Point and Rounding converts the absolute value of the given Real number to an 8-digit DWORD BCD number, compensating for an implied number of decimal points (K0-K8) and performs rounding.

For example, RTOBCDD R38156.74 with an implied number of decimal points equal to K1, would yield 381567 BCD. If the implied number of decimal points was 0, then the function would yield 38157 BCD (note that it rounded up).

If the Real number is negative, the Result will equal its positive, absolute value.

RTOBCDD Parameters

- Value (DWORD Real): specifies the Dword Real number that will be converted and rounded to a BCD number with decimal points
- Number of Decimal Points: specifies the number of implied decimal points in the Result DWORD
- Result (DWORD BCD): specifies the location where the rounded/implied decimal points DWORD BCD value will be placed

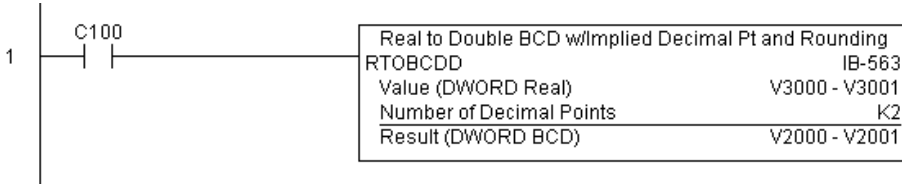
| Real to Double BCD w/Implied Decimal Pt and Rounding | |
|--|------|
| RTOBCDD IB-563 | |
| Value (DWORD Real) | TA0 |
| Number of Decimal Points | K0 |
| Result (DWORD BCD) | V400 |

| Parameter | DL205 Range |
|--------------------------|-------------|
| Value (DWORD Real) | V,P,R |
| Number of Decimal Points | K |
| Result (DWORD BCD) | V |

RTOBCDD Example

In the following example, the RTOBCDD instruction is used to convert the 32-bit REAL (floating point) data format in V3000 and V3001 to the 8-digit BCD data format and store in V2000 and V2001 when C100 turns on.

K2 in the Number of Decimal Points implies the data will have two implied decimal points.



Square BCD (SQUARE) (IB-523)

Square BCD squares the given 4-digit WORD BCD number and writes it as an 8-digit DWORD BCD result.

230

240

250-1

260

SQUARE Parameters

- Value (WORD BCD): specifies the BCD Word or constant that will be squared
- Result (DWORD BCD): specifies the location where the squared DWORD BCD value will be placed

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

Square BCD

SQUARE

IB-523

Value (WORD BCD)

TA0

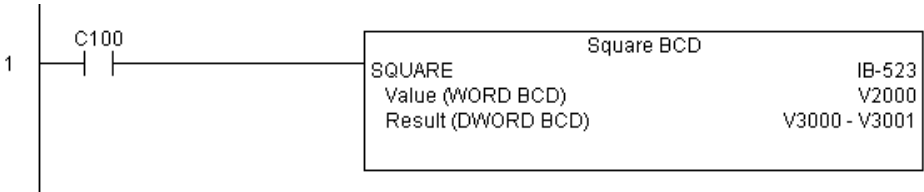
Result (DWORD BCD)

V400

| Parameter | | DL205 Range |
|--------------------|-------|---|
| Value (WORD BCD) | V,P,K | K0-9999 ; See DL205 V-memory map - Data Words |
| Result (DWORD BCD) | V | See DL205 V-memory map - Data Words |

SQUARE Example

In the following example, the SQUARE instruction is used to square the 4-digit BCD value in V2000 and store the 8-digit double word BCD result in V3000 and V3001 when C100 turns on.



Square Binary (SQUAREB) (IB-503)

Square Binary squares the given 16-bit WORD Binary number and writes it as a 32-bit DWORD Binary result.

 230

 240




 250-1

 260

SQUAREB Parameters

- Value (WORD Binary): specifies the binary Word or constant that will be squared
- Result (DWORD Binary): specifies the location where the squared DWORD binary value will be placed

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |



Square Binary

SQUAREB

IB-503

Value (WORD binary)

V2000

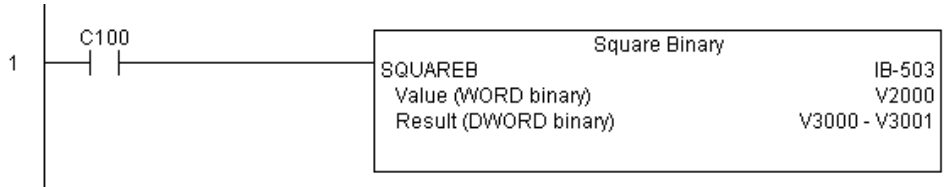
Result (DWORD binary)

V3000

| Parameter | | DL205 Range |
|-----------------------|-------|---|
| Value (WORD Binary) | V,P,K | K0-65535; See DL205 V-memory map - Data Words |
| Result (DWORD Binary) | V | See DL205 V-memory map - Data Words |

SQUAREB Example

In the following example, the SQUAREB instruction is used to square the single-word Binary value in V2000 and store the 8-digit double-word Binary result in V3000 and V3001 when C100 turns on.



Square Real (SQUARER) (IB-543)

Square Real squares the given REAL DWORD number and writes it to a REAL DWORD result.

230

240

250-1

260

SQUARER Parameters

- Value (REAL DWORD): specifies the Real DWORD location or number that will be squared
- Result (REAL DWORD): specifies the location where the squared Real DWORD value will be placed

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

☒☐☐

Square Real

IB-543

SQUARER

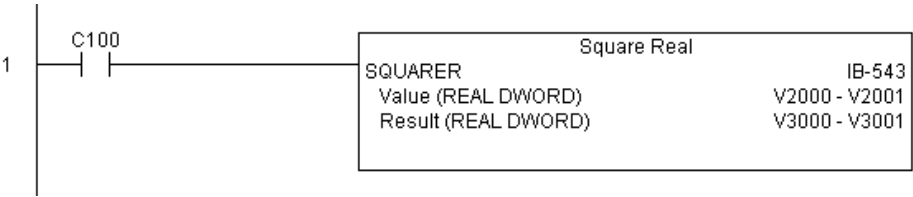
Value (REAL DWORD)

Result (REAL DWORD)

| Parameter | DL205 Range |
|--------------------------|---|
| Value (REAL DWORD) V,P,R | R ; See DL205 V-memory map - Data Words |
| Result (REAL DWORD) V | See DL205 V-memory map - Data Words |

SQUARER Example

In the following example, the SQUARER instruction is used to square the 32-bit floating point REAL value in V2000 and V2001 and store the REAL value result in V3000 and V3001 when C100 turns on.



Sum BCD Numbers (SUMBCD) (IB-522)

Sum BCD Numbers sums up a list of consecutive 4-digit WORD BCD numbers into an 8-digit DWORD BCD result.

You specify the group’s starting and ending V-memory addresses (inclusive). When enabled, this instruction will add all the numbers in the group (so you may want to place a differential contact driving the enable).

SUMBCD could be used as the first part of calculating an average.

- ☐ 230
- ☐ 240
- ☒ 250-1
- ☒ 260

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

SUMBCD Parameters

- Start Address: specifies the starting address of a block of V-memory location values to be added together (BCD)
- End Addr (inclusive): specifies the ending address of a block of V-memory location values to be added together (BCD)
- Result (DWORD BCD): specifies the location where the sum of the block of V-memory BCD values will be placed

☒☒☒

Sum BCD Numbers

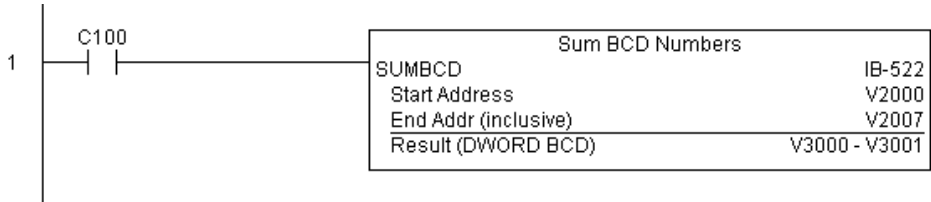
SUMBCD IB-522

| | |
|----------------------|-------|
| Start Address | V2000 |
| End Addr (inclusive) | V2007 |
| Result (DWORD BCD) | V3000 |

| Parameter | | DL205 Range |
|-------------------------|---|-------------------------------------|
| Start Address | V | See DL205 V-memory map - Data Words |
| End Address (inclusive) | V | See DL205 V-memory map - Data Words |
| Result (DWORD BCD) | V | See DL205 V-memory map - Data Words |

SUMBCD Example

In the following example, the SUMBCD instruction is used to total the sum of all BCD values in words V2000 thru V2007 and store the resulting 8-digit double word BCD value in V3000 and V3001 when C100 turns on.



Sum Binary Numbers (SUMBIN) (IB-502)

Sum Binary Numbers sums up a list of consecutive 16-bit WORD Binary numbers into a 32-bit DWORD binary result.

☐ 230

☐ 240

☒ 250-1

☒ 260

You specify the group's starting and ending V-memory addresses (inclusive). When enabled, this instruction will add all the numbers in the group (so you may want to place a differential contact driving the enable).

SUMBIN could be used as the first part of calculating an average.

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

SUMBIN Parameters

- Start Address: specifies the starting address of a block of V-memory location values to be added together (Binary)
- End Addr (inclusive): specifies the ending address of a block of V-memory location values to be added together (Binary)
- Result (DWORD Binary): specifies the location where the sum of the block of V-memory binary values will be placed

Sum Binary Numbers IB-502

SUMBIN

Start Address V2000

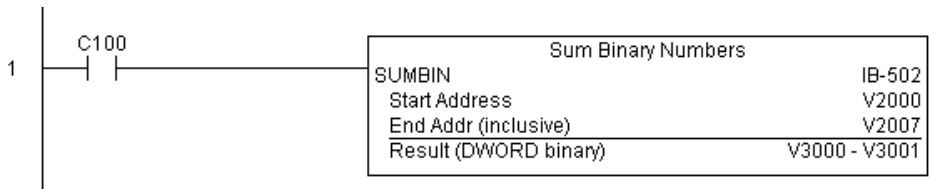
End Addr (inclusive) V2007

Result (DWORD binary) V3000

| Parameter | | DL205 Range |
|-------------------------|---|-------------------------------------|
| Start Address | V | See DL205 V-memory map - Data Words |
| End Address (inclusive) | V | See DL205 V-memory map - Data Words |
| Result (DWORD Binary) | V | See DL205 V-memory map - Data Words |

SUMBIN Example

In the following example, the SUMBIN instruction is used to total the sum of all Binary values in words V2000 thru V2007 and store the resulting 8-digit double word Binary value in V3000 and V3001 when C100 turns on.



Sum Real Numbers (SUMR) (IB-542)

Sum Real Numbers sums up a list of consecutive REAL DWORD numbers into a REAL DWORD result.

You specify the group's starting and ending V-memory addresses (inclusive).

Remember that Real numbers are DWORDs and occupy 2 words of V-memory each, so the number of Real values summed up is equal to half the number of memory locations. Note that the End Address can be EITHER word of the 2 word ending address, for example, if you wanted to add the 4 Real numbers stored in V2000 thru V2007 (V2000, V2002, V2004, and V2006), you can specify V2006 OR V2007 for the ending address and you will get the same result.

When enabled, this instruction will add all the numbers in the group (so you may want to place a differential contact driving the enable).

SUMR could be used as the first part of calculating an average.

☒ 230

☒ 240

☒ 250-1

☒ 260

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

SUMR Parameters

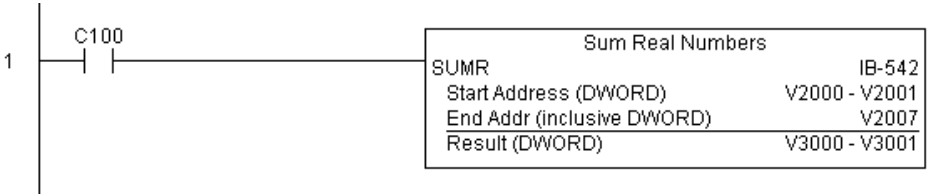
- Start Address (DWORD): specifies the starting address of a block of V-memory location values to be added together (Real)
- End Addr (inclusive) (DWORD): specifies the ending address of a block of V-memory location values to be added together (Real)
- Result (DWORD): specifies the location where the sum of the block of V-memory Real values will be placed

| Sum Real Numbers | |
|----------------------------|--------|
| SUMR | IB-542 |
| Start Address (DWORD) | V2000 |
| End Addr (inclusive DWORD) | V2007 |
| Result (DWORD) | V3000 |

| Parameter | | DL205 Range |
|-------------------------------|---|-------------------------------------|
| Start Address (DWORD) | V | See DL205 V-memory map - Data Words |
| End Address (inclusive DWORD) | V | See DL205 V-memory map - Data Words |
| Result (DWORD) | V | See DL205 V-memory map - Data Words |

SUMR Example

In the following example, the SUMR instruction is used to total the sum of all floating point REAL number values in words V2000 thru V2007 and store the resulting 32-bit floating point REAL number value in V3000 and V3001 when C100 turns on.



ECOM100 Configuration (ECOM100) (IB-710)

ECOM100 Configuration defines all the common information for one specific ECOM100 module which is used by the other ECOM100 IBoxes; for example, ECRX - ECOM100 Network Read, ECEMAIL - ECOM100 Send Email, ECIPSUP - ECOM100 IP Setup, etc.

You MUST have the ECOM100 Configuration IBox at the top of your ladder/stage program with any other configuration IBoxes. The Message Buffer parameter specifies the starting address of a 65 WORD buffer. This is 101 Octal addresses (e.g., V1400 thru V1500).

If you have more than one ECOM100 in your PLC, you must have a different ECOM100 Configuration IBox for EACH ECOM100 module in your system that utilizes any ECOM IBox instructions.

The Workspace and Status parameters and the entire Message Buffer are internal, private registers used by the ECOM100 Configuration IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

In order for MOST ECOM100 IBoxes to function, you must turn ON dip switch 7 on the ECOM100 circuit board. You can keep dip switch 7 off if you are ONLY using ECOM100 Network Read and Write IBoxes (ECRX, ECWX).

ECOM100 Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number.
- Slot: specifies which PLC slot is occupied by the ECOM100 module.
- Status: specifies a V-memory location that will be used by the instruction.
- Workspace: specifies a V-memory location that will be used by the instruction.
- Msg Buffer: specifies the starting address of a 65-word buffer that will be used by the module for configuration.

| Parameter | | DL205 Range |
|----------------------------|---|-------------------------------------|
| ECOM100# | K | K0-255 |
| Slot | K | K0-7 |
| Status | V | See DL205 V-memory map - Data Words |
| Workspace | V | See DL205 V-memory map - Data Words |
| Msg Buffer (65 words used) | V | See DL205 V-memory map - Data Words |

ECOM100 Example

The ECOM100 Config IBox coordinates all of the interaction with other ECOM100-based IBoxes (ECxxxx). You must have an ECOM100 Config IBox for each ECOM100 module in your system. Configuration IBoxes must be at the top of your program and must execute every scan.

This IBox defines ECOM100# K0 to be in slot 3. Any ECOM100 IBoxes that need to reference this specific module (such as ECEMAIL, ECRX, ...) would enter K0 for their ECOM100# parameter.

The Status register is for reporting any completion or error information to other ECOM100 IBoxes. This V-memory register must not be used anywhere else in the entire program.

The Workspace register is used to maintain state information about the ECOM100, along with proper sharing and interlocking with the other ECOM100 IBoxes in the program. This V-memory register must not be used anywhere else in the entire program.

The Message Buffer of 65 words (130 bytes) is a common pool of memory that is used by other ECOM100 IBoxes (such as ECEMAIL). This way, you can have a bunch of ECEMAIL IBoxes, but only need 1 common buffer for generating and sending each EMail. These V-memory registers must not be used anywhere else in your entire program.

| | | |
|---|-----------------------|---------------|
| 1 | ECOM100 Config | |
| | ECOM100 | IB-710 |
| | ECOM100 # | K0 |
| | Slot | K3 |
| | Status | V1501 |
| | Workspace | V1502 |
| | Msg Buffer (65 WORDs) | V1400 - V1500 |

No permissive contact or input logic is used with this instruction



NOTE: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in ***BOLD italics***, and the instruction name and ID will be in ***BOLD characters***.

ECOM100 Disable DHCP (ECDHCPD) (IB-736)

 230

 240

 250-1

 260

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

5

ECOM100 Disable DHCP will set up the ECOM100 to use its internal TCP/IP settings on a leading edge transition to the IBox. To configure the ECOM100's TCP/IP settings manually, use the NetEdit3 utility, or you can do it programmatically from your PLC program using the ECOM100 IP Setup (ECIPSUP), or the individual ECOM100 IBoxes: ECOM Write IP Address (ECWRIP), ECOM Write Gateway Address (ECWRGWA), and ECOM100 Write Subnet Mask (ECWRSNM).

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

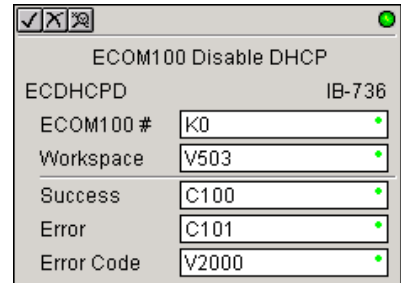
Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).

The "Disable DHCP" setting is stored in Flash-ROM in the ECOM100 and the execution of this IBox will disable the ECOM100 module for at least a half second until it writes the Flash-ROM. Therefore, it is **HIGHLY RECOMMENDED** that you only execute this IBox ONCE, on the second scan. Since it requires a LEADING edge to execute, use a **NORMALLY CLOSED SP0 (STR NOT First Scan)** to drive the power flow to the IBox.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

ECDHCPD Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not completed successfully
- Error Code: specifies the location where the Error Code will be written



| Parameter | | DL205 Range |
|------------|---------------|-------------------------------------|
| ECOM100# | K | K0-255 |
| Workspace | V | See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error Code | V | See DL205 V-memory map - Data Words |

ECDHCPD Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.



NOTE: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.

Rung 2: On the 2nd scan, disable DHCP in the ECOM100. DHCP is the same protocol used by PCs for using a DHCP Server to automatically assign the ECOM100's IP Address, Gateway Address, and Subnet Mask. Typically disabling DHCP is done by assigning a hard-coded IP Address either in NetEdit or using one of the ECOM100 IP Setup IBoxes, but this IBox allows you to disable DHCP in the ECOM100 using your ladder program. The ECDHCPD is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to disable DHCP will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON. If successful, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the specific error code.



ECOM100 Enable DHCP (ECDHCPE) (IB-735)

ECOM100 Enable DHCP will tell the ECOM100 to obtain its TCP/IP setup from a DHCP Server on a leading edge transition to the IBox.

The IBox will be successful once the ECOM100 has received its TCP/IP settings from the DHCP server. Since it is possible for the DHCP server to be unavailable, a Timeout parameter is provided so that the IBox can complete, but with an Error (Error Code = 1004 decimal).

See also the ECOM100 IP Setup (ECIPSUP) IBox 717 to directly set up ALL of the TCP/IP parameters in a single instruction - IP Address, Subnet Mask, and Gateway Address.

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).

The “Enable DHCP” setting is stored in Flash-ROM in the ECOM100 and the execution of this IBox will disable the ECOM100 module for at least a half second until it writes the Flash-ROM. Therefore, it is **HIGHLY RECOMMENDED** that you only execute this IBox **ONCE**, on the second scan. Since it requires a **LEADING** edge to execute, use a **NORMALLY CLOSED SP0 (STR NOT First Scan)** to drive the power flow to the IBox.

In order for this ECOM100 IBox to function, you must turn **ON** dip switch 7 on the ECOM100 circuit board.

ECDHCPE Parameters

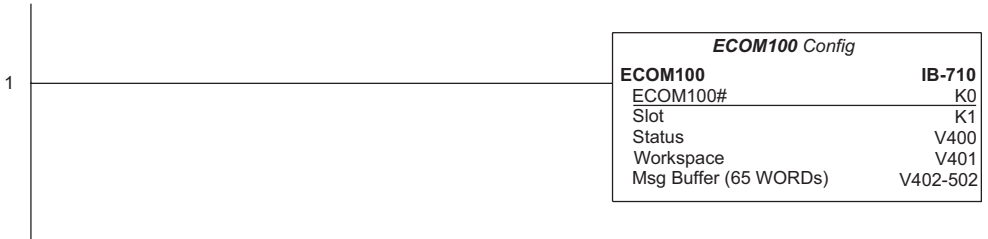
- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number.
- Timeout(sec): specifies a timeout period so that the instruction may have time to complete.
- Workspace: specifies a V-memory location that will be used by the instruction.
- Success: specifies a bit that will turn on once the request is completed successfully.
- Error: specifies a bit that will turn on if the instruction is not completed successfully.
- Error Code: specifies the location where the Error Code will be written.

| ECOM100 Enable DHCP | |
|---------------------|--------|
| ECDHCPE | IB-735 |
| ECOM100 # | K0 |
| Timeout(sec.) | K5 |
| Workspace | V400 |
| Success | C0 |
| Error | C0 |
| Error Code | V400 |

| Parameter | | DL205 Range |
|---------------|---------------|-------------------------------------|
| ECOM100# | K | K0-255 |
| Timeout (sec) | K | K5-127 |
| Workspace | V | See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error Code | V | See DL205 V-memory map - Data Words |

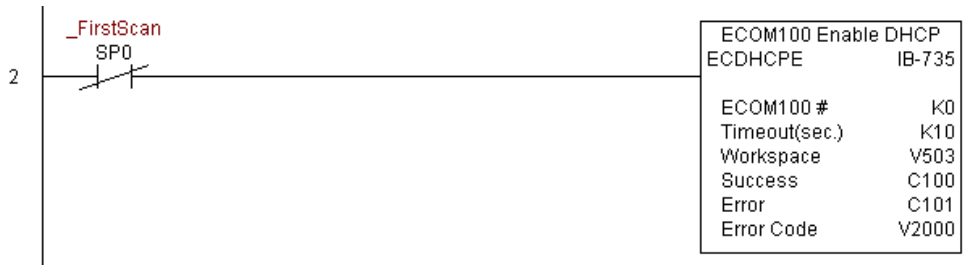
ECDHCPE Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.



NOTE: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.

Rung 2: On the second scan, enable DHCP in the ECOM100. DHCP is the same protocol used by PCs for using a DHCP Server to automatically assign the ECOM100's IP Address, Gateway Address, and Subnet Mask. Typically this is done using NetEdit, but this IBox allows you to enable DHCP in the ECOM100 using your ladder program. The ECDHCPE is leading edge triggered, not power-flow driven (similar to a counter input leg). The commands to enable DHCP will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON. The ECDHCPE does more than just set the bit to enable DHCP in the ECOM100, it polls the ECOM100 once every second to see if the ECOM100 has found a DHCP server and has a valid IP Address. Therefore, a timeout parameter is needed in case the ECOM100 cannot find a DHCP server. If a timeout does occur, the Error bit will turn on and the error code will be 1005 decimal. The Success bit will turn on only if the ECOM100 finds a DHCP Server and is assigned a valid IP Address. If successful, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the specific error code.



ECOM100 Query DHCP Setting (ECDHCPQ) (IB-734)

ECOM100 Query DHCP Setting will determine if DHCP is enabled in the ECOM100 on a leading edge transition to the IBox. The DHCP Enabled bit parameter will be ON if DHCP is enabled, OFF if disabled.

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

230

240

250-1

260

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

ECDHCPQ Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number.
- Workspace: specifies a V-memory location that will be used by the instruction.
- Success: specifies a bit that will turn on once the instruction is completed successfully.
- Error: specifies a bit that will turn on if the instruction is not completed successfully.
- DHCP Enabled: specifies a bit that will turn on if the ECOM100's DHCP is enabled or remain off if disabled - after instruction query, be sure to check the state of the Success/Error bit state along with DHCP Enabled bit state to confirm a successful module query.

| Parameter | DL205 Range |
|--------------|--|
| ECOM100# | K K0-255 |
| Workspace | V See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B See DL205 V-memory map |
| DHCP Enabled | X,Y,C,GX,GY,B See DL205 V-memory map |

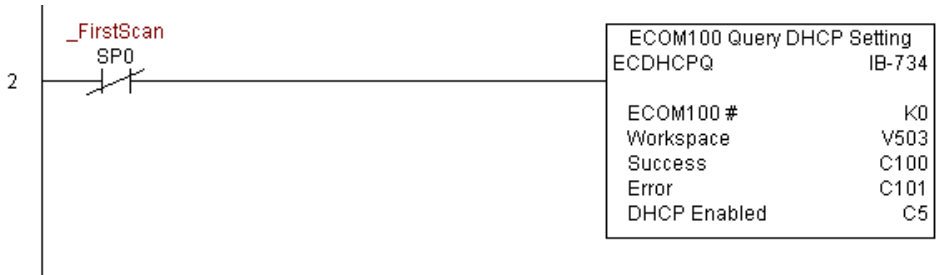
ECDHCPQ Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.



NOTE: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.

Rung 2: On the second scan, read whether DHCP is enabled or disabled in the ECOM100 and store it in C5. DHCP is the same protocol used by PCs for using a DHCP Server to automatically assign the ECOM100's IP Address, Gateway Address, and Subnet Mask. The ECDHCPQ is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to read (Query) whether DHCP is enabled or not will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON. If successful, turn on C100. If there is a failure, turn on C101.



ECOM100 Send E-mail (ECEMAIL) (IB-711)

| | |
|-------------------------------------|-------|
| <input type="checkbox"/> | 230 |
| <input type="checkbox"/> | 240 |
| <input checked="" type="checkbox"/> | 250-1 |
| <input checked="" type="checkbox"/> | 260 |

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

ECOM100 Send EMail, on a leading edge transition, will behave as an EMail client and send an SMTP request to your SMTP Server to send the EMail message to the EMail addresses in the To: field and also to those listed in the Cc: list hard coded in the ECOM100. It will send the SMTP request based on the specified ECOM100#, which corresponds to a specific unique ECOM100 Configuration (ECOM100) at the top of your program.

The Body: field supports what the PRINT and VPRINT instructions support for text and embedded variables, allowing you to embed real-time data in your EMail (e.g., "V2000 = " V2000:B).

The Workspace parameter is an internal, private register used by this IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the request is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), an SMPT protocol error (between 100 and 999), or a PLC logic error (greater than 1000).

Since the ECOM100 is only an EMail Client and requires access to an SMTP Server, you MUST have the SMTP parameters configured properly in the ECOM100 via the ECOM100's Home Page and/or the EMail Setup instruction (ECEMSUP). To get to the ECOM100's Home Page, use your favorite Internet browser and browse to the ECOM100's IP Address, e.g., <http://192.168.12.86>

You are limited to approximately 100 characters of message data for the entire instruction, including the To: Subject: and Body: fields. To save space, the ECOM100 supports a hard coded list of EMail addresses for the Carbon Copy field (cc:) so that you can configure those IN the ECOM100, and keep the To: field small (or even empty), to leave more room for the Subject: and Body: fields.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

ECOM100 Send EMail IB-711

| | |
|------------|--|
| ECOM100 # | K0 |
| Workspace | V503 |
| Success | C100 |
| Error | C101 |
| Error Code | V2000 |
| To | joe@acme.com |
| Subject | Machine Offline |
| Body | "Machine #" V5010:B "went offline at" V3000:B "on "V3010:B " \$N" |

ECEMAIL Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not completed successfully
- Error Code: specifies the location where the Error Code will be written
- To: specifies an E-mail address that the message will be sent to
- Subject: subject of the e-mail message
- Body: supports what the PRINT and VPRINT instructions support for text and embedded variables, allowing you to embed real-time data in the EMail message

| Parameter | | DL205 Range |
|------------|---------------|-------------------------------------|
| ECOM100# | K | K0-255 |
| Workspace | V | See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error Code | V | See DL205 V-memory map |
| To: | | Text |
| Subject: | | Text |
| Body: | | See PRINT and VPRINT instructions |

ECEMAIL Decimal Status Codes

This list of status codes is based on the list in the *ECOM100 Mock Slave Address 89 Command Specification*.

ECOM100 Status codes can be classified into four different areas based on its decimal value.

| ECOM100 Status Code Areas | |
|---------------------------|--|
| 0-1 | Normal Status - no error |
| 2-99 | Internal ECOM100 errors |
| 100-999 | Standard TCP/IP protocol errors (SMTP, HTTP, etc) |
| 1000+ | IBox ladder logic assigned errors (SP Slot Error, etc) |

For the ECOM100 Send EMail IBOX, the status codes below are specific to this IBox.

Normal Status 0 - 1

| ECOMM100 Send EMAIL IBOX Status Codes | |
|---------------------------------------|--|
| 0-1 | Success - ECEMAIL completed successfully. |
| 1 | Busy - ECEMAIL IBOX logic sets the Error register to this value when the ECEMAIL starts a new request. |

Internal ECOM100 Errors (2-99)

| Internal ECOM 100 Errors (2-99) | |
|---------------------------------|---|
| 10-19 | Timeout Errors - last digit shows where in ECOM100's SMTP state logic the timeout occurred; <i>regardless of the last digit</i> , the SMTP conversation with the SMTP Server timed out. |
| | SMTP Internal Errors (20-29) |
| 20 | TCP Write Error |
| 21 | No Sendee |
| 22 | Invalid State |
| 23 | Invalid Data |
| 24 | Invalid SMTP Configuration |
| 25 | Memory Allocation Error |

ECEMAIL IBox Ladder Logic Assigned Errors (1000+)

| ECEMAIL IBox Ladder Logic Assigned Errors (1000+) | |
|---|--|
| 101 | SP Slot Error - the SP error bit for the ECOM100's slot turned on. Possibly using RX or WX instructions on the ECOM100 and walking on the ECEMAIL execution. Use should use ECRX and ECX IBoxes. |

ECEMAIL Decimal Status Codes

SMTP Protocol Errors - SMTP (100-999)

5

| SMTP Protocol Errors - SMTP (100-999) | |
|---------------------------------------|---|
| Error | Description |
| 1xx | Informational replies |
| 2xx | Success replies |
| 200 | (Non-standard success response) |
| 211 | System status or system help reply |
| 214 | Help message |
| 220 | <domain> Service ready. Ready to start TLS |
| 221 | <domain> Service closing transmission channel |
| 250 | Ok, queuing for node <node> started. Requested mail action okay, completed |
| 251 | Ok, no messages waiting for node <node>. User not local; will forward to <forward-path> |
| 252 | OK, pending messages for node <node> started. Cannot VRFY user (e.g., info is not local), but will take message for this user and attempt delivery. |
| 253 | OK, messages pending messages for node <node> started |
| 3xx | (re) direction replies |
| 354 | Start mail input; end with <CRLF>.<CRLF> |
| 355 | Octet-offset is the transaction offset. |
| 4xx | client/request error replies |
| 421 | <domain> Service not available, closing transmission channel |
| 432 | A password transition is needed |
| 450 | Requested mail action not taken: mailbox unavailable. ATRN request refused. |
| 451 | Requested action aborted; local error in processing. Unable to process ATRN request now. |
| 452 | Requested action not taken: insufficient system storage |
| 453 | You have no mail |
| 454 | TLS is not available due to temporary reason. Encryption required for requested authentication mechanism. |
| 458 | Unable to queue messages for node <node> |
| 459 | Node <node> not allowed: <reason> |
| 5xx | Server/process error replies |
| 500 | Syntax error, command unrecognized. Syntax error. |
| 501 | Syntax error in parameters or arguments |
| 502 | Command not implemented |
| 503 | Bad sequence of commands |
| 504 | Command parameter not implemented |
| 521 | <domain> does not accept mail |
| 530 | Access denied. Must issue a STARTTLS command first. Encryption required for requested authentication mechanism. |
| <i>Continued next page</i> | |

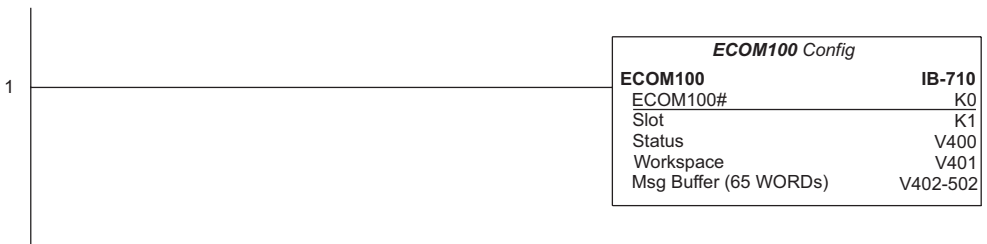
| SMTP Protocol Errors - SMTP (100-999) | |
|---------------------------------------|---|
| Error | Description |
| 534 | Authentication mechanism is too weak. |
| 538 | Encryption required for requested authentication mechanism. |
| 550 | Requested action not taken; mailbox unavailable. |
| 551 | User not local; please try <forward path> |
| 552 | Requested mail action aborted; exceeded storage allocation |
| 553 | Requested action not taken; mailbox name not allowed. |
| 554 | Transaction failed |

ECEMAIL Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.



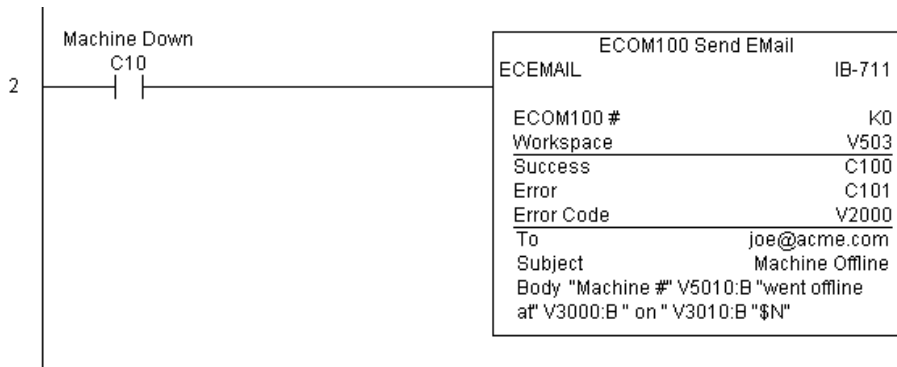
NOTE: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.



Rung 2: When a machine goes down, send an email to Joe in maintenance and to the VP over production showing what machine is down along with the date/time stamp of when it went down.

The ECEMAIL is leading edge triggered, not power-flow driven (similar to a counter input leg). An email will be sent whenever the power flow into the IBox goes from OFF to ON. This helps prevent self-inflicted spamming.

If the EMail is sent, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the SMTP error code or other possible error codes.



ECOM100 Restore Default E-mail Setup (ECEMRDS) (IB-713)

ECOM100 Restore Default EMail Setup, on a leading edge transition, will restore the original EMail Setup data stored in the ECOM100 back to the working copy based on the specified ECOM100#, which corresponds to a specific unique ECOM100 Configuration (ECOM100) at the top of your program.

When the ECOM100 is first powered up, it copies the EMail setup data stored in ROM to the working copy in RAM. You can then modify this working copy from your program using the ECOM100 EMail Setup (ECEMSUP) IBox. After modifying the working copy, you can later restore the original setup data via your program by using this IBox.

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

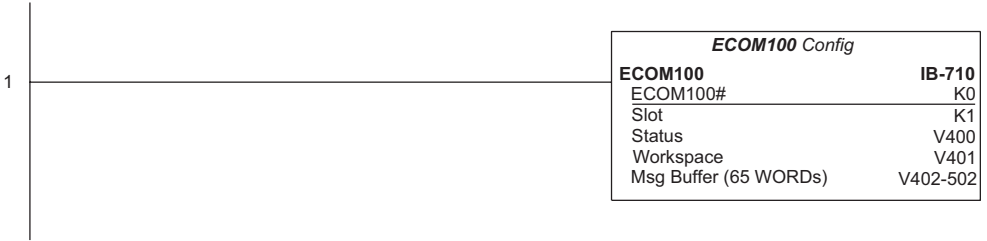
ECEMRDS Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not completed successfully
- Error Code: specifies the location where the Error Code will be written

| Parameter | | DL205 Range |
|------------|---------------|-------------------------------------|
| ECOM100# | K | K0-255 |
| Workspace | V | See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error Code | V | See DL205 V-memory map - Data Words |

ECEMRDS Example

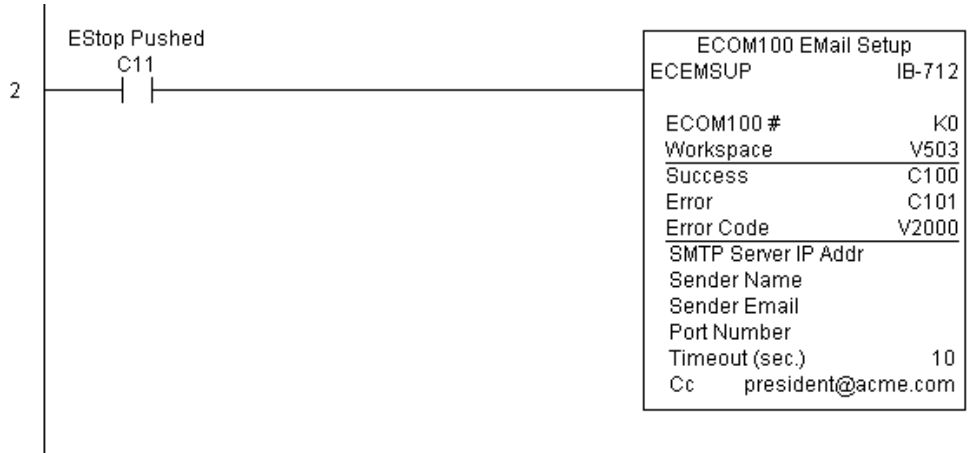
Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.



NOTE: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.

Rung 2: Whenever an EStop is pushed, ensure that the president of the company gets copies of all Emails being sent.

The ECOM100 EMail Setup IBox allows you to set/change the SMTP Email settings stored in the ECOM100.



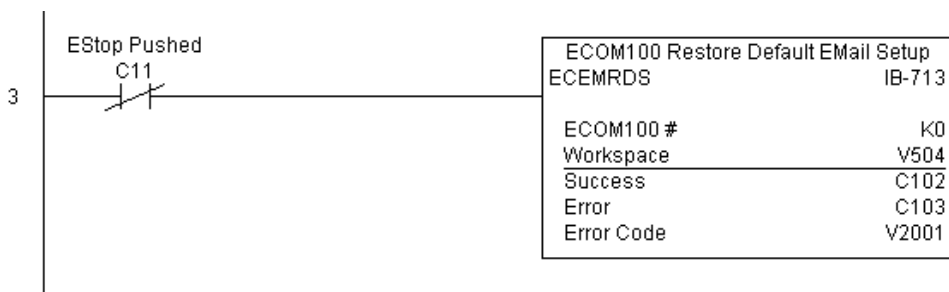
(Example continued on next page)

ECEMRDS Example (cont'd)

Rung 3: Once the EStop is pulled out, take the president off the cc: list by restoring the default EMail setup in the ECOM100.

The ECEMRDS is leading edge triggered, not power-flow driven (similar to a counter input leg). The ROM-based EMail configuration stored in the ECOM100 will be copied over the “working copy” whenever the power flow into the IBox goes from OFF to ON (the working copy can be changed by using the ECEMSUP IBox).

If successful, turn on C102. If there is a failure, turn on C103. If it fails, you can look at V2001 for the specific error code.



ECOM100 E-mail Setup (ECEMSUP) (IB-712)

✗ 230

✗ 240

✓ 250-1

✓ 260

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

ECOM100 EMail Setup, on a leading edge transition, will modify the working copy of the EMail setup currently in the ECOM100 based on the specified ECOM100#, which corresponds to a specific unique ECOM100 Configuration (ECOM100) at the top of your program.

You may pick and choose any or all fields to be modified using this instruction. Note that these changes are cumulative: if you execute multiple ECOM100 EMail Setup IBoxes, then all of the changes are made in the order they are executed. Also note that you can restore the original ECOM100 EMail Setup that is stored in the ECOM100 to the working copy by using the ECOM100 Restore Default EMail Setup (ECEMRDS) IBox.

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).

You are limited to approximately 100 characters/bytes of setup data for the entire instruction. So if needed, you could divide the entire setup across multiple ECEMSUP IBoxes on a field-by-field basis, for example do the Carbon Copy (cc:) field in one ECEMSUP IBox and the remaining setup parameters in another.

In order for this ECOM100 IBox to function, you must turn **ON** dip switch 7 on the ECOM100 circuit board.

ECEMSUP Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number.
- Workspace: specifies a V-memory location that will be used by the instruction.
- Success: specifies a bit that will turn on once the request is completed successfully.
- Error: specifies a bit that will turn on if the instruction is not completed successfully.
- Error Code: specifies the location where the Error Code will be written.
- SMTP Server IP Addr: optional parameter that specifies the IP Address of the SMTP Server on the ECOM100's network.
- Sender Name: optional parameter that specifies the sender name that will appear in the "From:" field to those who receive the e-mail.
- Sender EMail: optional parameter that specifies the sender EMail address that will appear in the "From:" field to those who receive the e-mail.

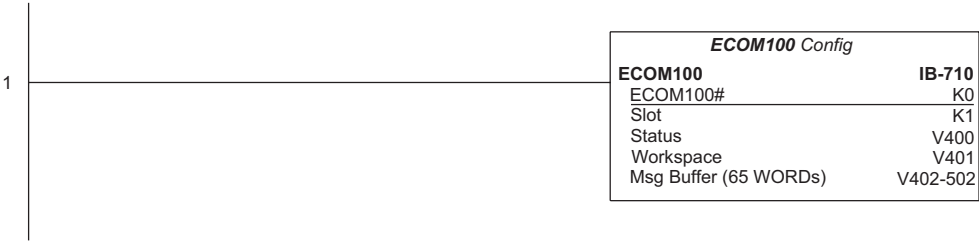
ECEMSUP Parameters (cont'd)

- Port Number: optional parameter that specifies the TCP/IP Port Number to send SMTP requests; usually this does not need to be configured (see your network administrator for information on this setting).
- Timeout (sec): optional parameter that specifies the number of seconds to wait for the SMTP Server to send the EMail to all the recipients.
- Cc: optional parameter that specifies a list of “carbon copy” Email addresses to send all EMail to.

| Parameter | | DL205 Range |
|------------|---------------|-------------------------------------|
| ECOM100# | K | K0-255 |
| Workspace | V | See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error Code | V | See DL205 V-memory map - Data Words |

ECEMSUP Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.



(Example continued on next page)

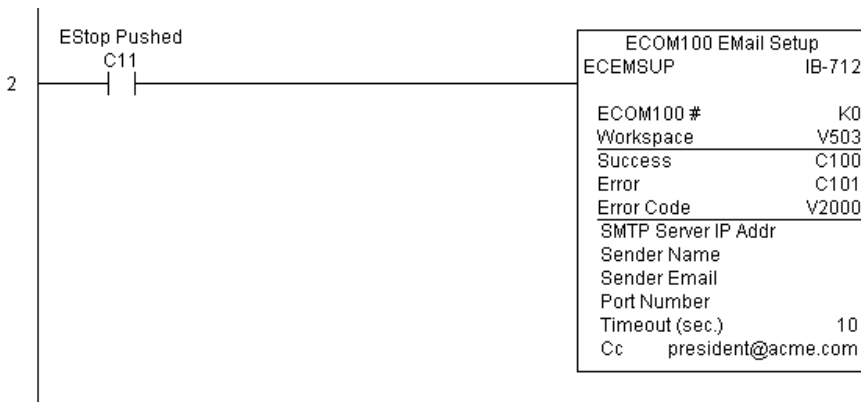


NOTE: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in ***BOLD italics***, and the instruction name and ID will be in ***BOLD characters***.

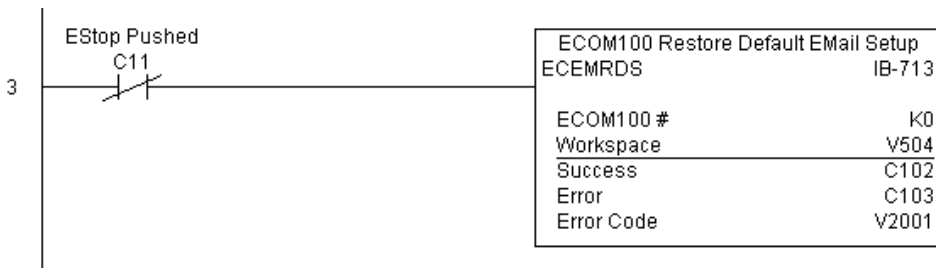
ECEMSUP Example (cont'd)

Rung 2: Whenever an EStop is pushed, ensure that president of the company gets copies of all EMail being sent. The ECOM100 EMail Setup IBox allows you to set/change the SMTP EMail settings stored in the ECOM100. The ECEMSUP is leading edge triggered, not power-flow driven (similar to a counter input leg). At power-up, the ROM-based EMail configuration stored in the ECOM100 is copied to a RAM-based "working copy". You can change this working copy by using the ECEMSUP IBox. To restore the original ROM-based configuration, use the Restore Default EMail Setup ECEMRDS IBox.

If successful, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the specific error code.



Rung 3: Once the EStop is pulled out, take the president off the cc: list by restoring the default EMail setup in the ECOM100.



ECOM100 IP Setup (ECIPSUP) (IB-717)

- ☐ 230
- ☐ 240
- ☒ 250-1
- ☒ 260

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

ECOM100 IP Setup will configure the three TCP/IP parameters in the ECOM100: IP Address, Subnet Mask, and Gateway Address, on a leading edge transition to the IBox. The ECOM100 is specified by the ECOM100#, which corresponds to a specific unique ECOM100 Configuration (ECOM100) IBox at the top of your program.

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).

This setup data is stored in Flash-ROM in the ECOM100 and will disable the ECOM100 module for at least a half second until it writes the Flash-ROM. Therefore, it is **HIGHLY RECOMMENDED** that you only execute this IBox **ONCE** on the second scan. Since it requires a **LEADING** edge to execute, use a **NORMALLY CLOSED SP0 (NOT First Scan)** to drive the power flow to the IBox.

In order for this ECOM100 IBox to function, you must turn **ON** dip switch 7 on the ECOM100 circuit board.

ECIPSUP Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number.
- Workspace: specifies a V-memory location that will be used by the instruction.
- Success: specifies a bit that will turn on once the request is completed successfully.
- Error: specifies a bit that will turn on if the instruction is not completed successfully.
- Error Code: specifies the location where the Error Code will be written.
- IP Address: specifies the module's IP Address.
- Subnet Mask: specifies the Subnet Mask for the module to use.
- Gateway Address: specifies the Gateway Address for the module to use.

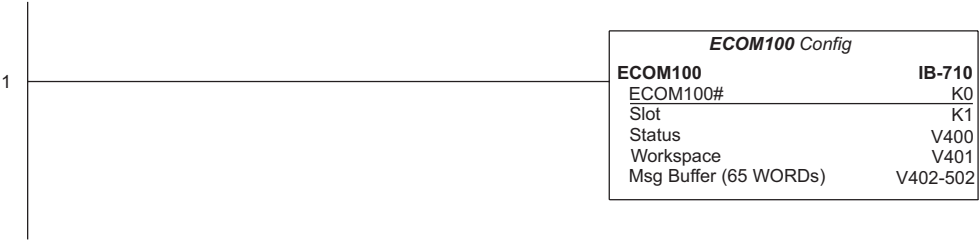
| ECOM100 IP Setup | |
|------------------|-----------------------|
| ECIPSUP | IB-717 |
| ECOM100 # | K0 |
| Workspace | V503 |
| Success | C100 |
| Error | C101 |
| Error Code | V2000 |
| IP Address | 192 . 168 . 012 . 100 |
| Subnet Mask | 255 . 255 . 0 . 0 |
| Gateway Address | 192 . 168 . 0 . 1 |

| Parameter | | DL205 Range |
|---------------------|-----------------|-------------------------------------|
| ECOM100# | K | K0-255 |
| Workspace | V | See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error Code | V | See DL205 V-memory map - Data Words |
| IP Address | IP Address | 0.0.0.1. to 255.255.255.254 |
| Subnet Mask Address | IP Address Mask | 0.0.0.1. to 255.255.255.254 |
| Gateway Address | IP Address | 0.0.0.1. to 255.255.255.254 |

ECIPSUP Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

5



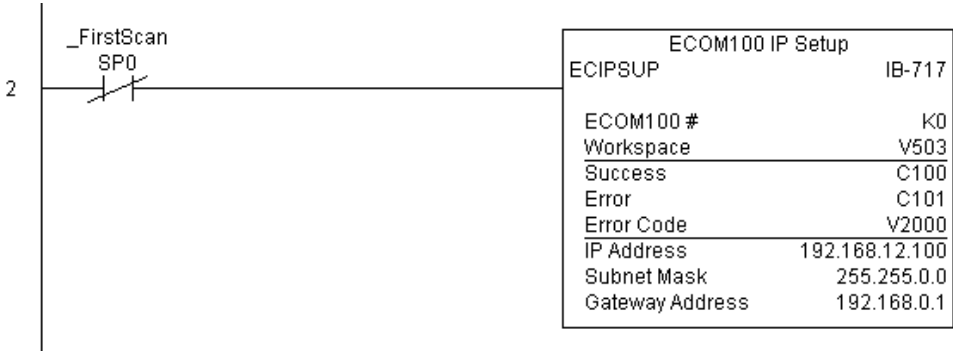
NOTE: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.

Rung 2: On the second scan, configure all of the TCP/IP parameters in the ECOM100:

IP Address: 192.168.12.100
Subnet Mask: 255.255.0.0
Gateway Address: 192.168.0.1

The ECIPSUP is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to write the TCP/IP configuration parameters will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the specific error code.



ECOM100 Read Description (ECRDDES) (IB-726)

ECOM100 Read Description will read the ECOM100's Description field up to the number of specified characters on a leading edge transition to the IBox.

 230

 240

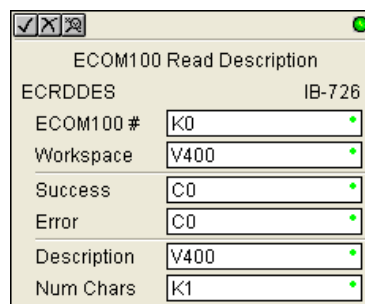
 250-1

 260

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.



5

ECRDDES Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not successfully completed
- Description: specifies the starting buffer location where the ECOM100's Description will be placed
- Num Chars: specifies the number of characters (bytes) to read from the ECOM100's Description field

| Parameter | | DL205 Range |
|-------------|---------------|-------------------------------------|
| ECOM100# | K | K0-255 |
| Workspace | V | See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Description | V | See DL205 V-memory map - Data Words |
| Num Chars | K | K1-128 |

ECRDDES Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

5



NOTE: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.

Rung 2: On the 2nd scan, read the Module Description of the ECOM100 and store it in V3000 thru V3007 (16 characters). This text can be displayed by an HMI.

The ECRDDES is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to read the module description will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101.



ECOM100 Read Gateway Address (ECRDGWA) (IB-730)

ECOM100 Read Gateway Address will read the four parts of the Gateway IP address and store them in four consecutive V-memory locations in decimal format, on a leading edge transition to the IBox.

 230

 240

 250-1

 260

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

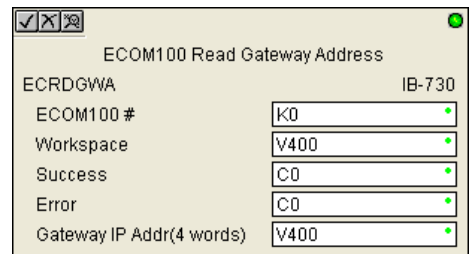
Either the Success or Error bit parameter will turn on once the command is complete.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

ECRDGWA Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number.
- Workspace: specifies a V-memory location that will be used by the instruction.
- Success: specifies a bit that will turn on once the request is completed successfully.
- Error: specifies a bit that will turn on if the instruction is not completed successfully.
- Gateway IP Addr: specifies the starting address where the ECOM100's Gateway Address will be placed in 4 consecutive V-memory locations.



ECOM100 Read Gateway Address

IB-730

ECRDGWA

ECOM100 # K0

Workspace V400

Success C0

Error C0

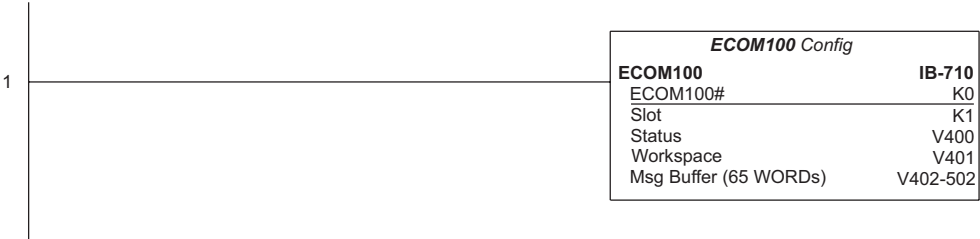
Gateway IP Addr(4 words) V400

| Parameter | | DL205 Range |
|------------------------------|---------------|-------------------------------------|
| ECOM100# | K | K0-255 |
| Workspace | V | See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Gateway IP Address (4 Words) | V | See DL205 V-memory map - Data Words |

ECRDGWA Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

5

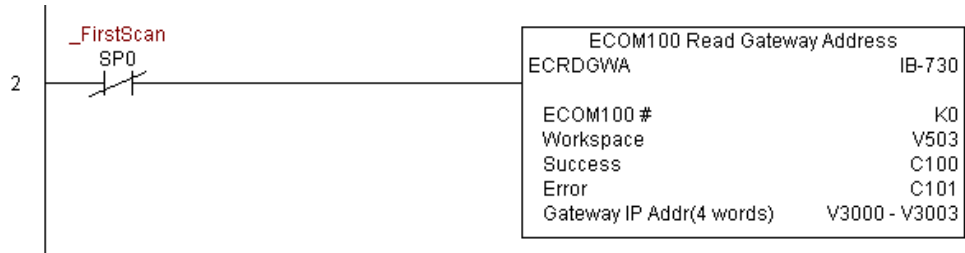


NOTE: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.

Rung 2: On the second scan, read the Gateway Address of the ECOM100 and store it in V3000 thru V3003 (4 decimal numbers). The ECOM100's Gateway Address could be displayed by an HMI.

The ECRDGWA is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to read the Gateway Address will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101.



ECOM100 Read IP Address (ECRDIP) (IB-722)

 230

 240

 250-1

 260

ECOM100 Read IP Address will read the four parts of the IP address and store them in four consecutive V-memory locations in decimal format, on a leading edge transition to the IBox.

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

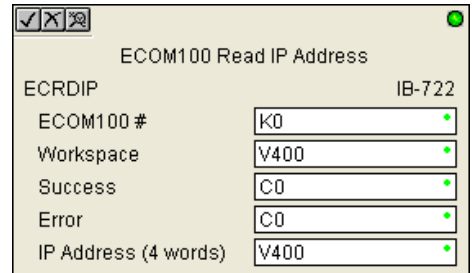
Either the Success or Error bit parameter will turn on once the command is complete.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

ECRDIP Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not completed successfully
- IP Address: specifies the starting address where the ECOM100's IP Address will be placed in four consecutive V-memory locations

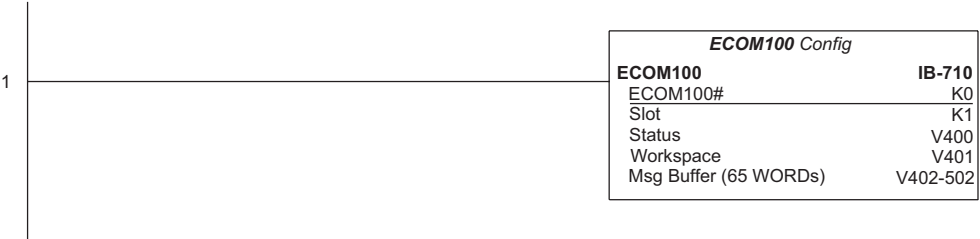


| Parameter | | DL205 Range |
|----------------------|---------------|-------------------------------------|
| ECOM100# | K | K0-255 |
| Workspace | V | See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |
| IP Address (4 Words) | V | See DL205 V-memory map - Data Words |

ECRDIP Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

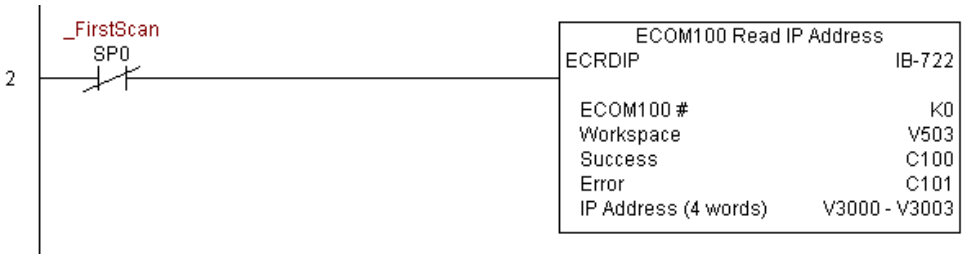
5



NOTE: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.

Rung 2: On the second scan, read the IP Address of the ECOM100 and store it in V3000 thru V3003 (four decimal numbers). The ECOM100's IP Address could be displayed by an HMI. The ECRDIP is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to read the IP Address will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101.



ECOM100 Read Module ID (ECRDMID) (IB-720)

ECOM100 Read Module ID will read the binary (decimal) WORD sized Module ID on a leading edge transition to the IBox.

 230

 240

 250-1

 260

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

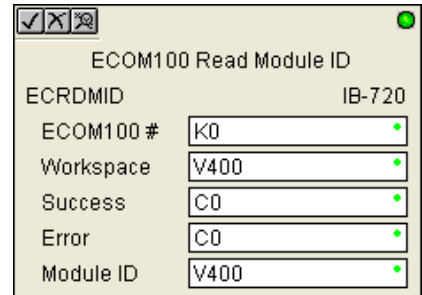
Either the Success or Error bit parameter will turn on once the command is complete.

In order for this ECOM100 IBox to function, you must turn **ON** dip switch 7 on the ECOM100 circuit board.

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

ECRDMID Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not completed successfully
- Module ID: specifies the location where the ECOM100's Module ID (decimal) will be placed



| Parameter | | DL205 Range |
|-----------|---------------|-------------------------------------|
| ECOM100# | K | K0-255 |
| Workspace | V | See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Module ID | V | See DL205 V-memory map - Data Words |

ECRDMID Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

5



NOTE: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.

Rung 2: On the second scan, read the Module ID of the ECOM100 and store it in V2000.

The ECRDMID is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to read the module ID will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101.



ECOM100 Read Module Name (ECRDNAM) (IB-724)

ECOM100 Read Name will read the Module Name up to the number of specified characters on a leading edge transition to the IBox.

 230

 240

 250-1

 260

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

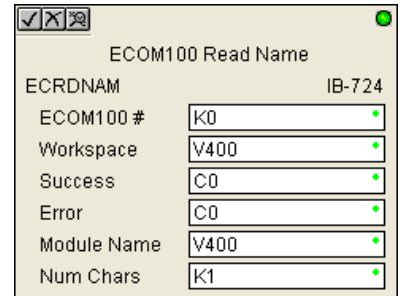
Either the Success or Error bit parameter will turn on once the command is complete.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

ECRDNAM Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number.
- Workspace: specifies a V-memory location that will be used by the instruction.
- Success: specifies a bit that will turn on once the request is completed successfully.
- Error: specifies a bit that will turn on if the instruction is not completed successfully.
- Module Name: specifies the starting buffer location where the ECOM100's Module Name will be placed.
- Num Chars: specifies the number of characters (bytes) to read from the ECOM100's Name field.

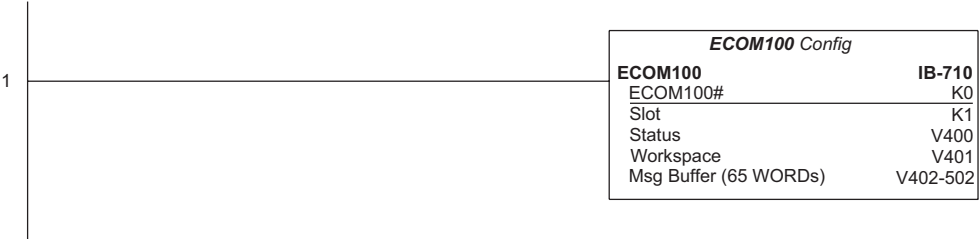


| Parameter | | DL205 Range |
|-------------|---------------|-------------------------------------|
| ECOM100# | K | K0-255 |
| Workspace | V | See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Module Name | V | See DL205 V-memory map - Data Words |
| Num Chars | K | K1-128 |

ECRDNAM Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

5



NOTE: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.

Rung 2: On the second scan, read the Module Name of the ECOM100 and store it in V3000 thru V3003 (8 characters). This text can be displayed by an HMI.

The ECRDNAM is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to read the module name will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101.



ECOM100 Read Subnet Mask (ECRDSNM) (IB-732)

ECOM100 Read Subnet Mask will read the four parts of the Subnet Mask and store them in 4 consecutive V-memory locations in decimal format, on a leading edge transition to the IBox.

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

ECRDSNM Parameters

- **ECOM100#:** this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number.
- **Workspace:** specifies a V-memory location that will be used by the instruction.
- **Success:** specifies a bit that will turn on once the request is completed successfully.
- **Error:** specifies a bit that will turn on if the instruction is not completed successfully.
- **Subnet Mask:** specifies the starting address where the ECOM100's Subnet Mask will be placed in four consecutive V-memory locations.

ECOM100 Read Subnet Mask IB-732

| | |
|-----------------------|--------|
| ECRDSNM | IB-732 |
| ECOM100 # | K0 |
| Workspace | V400 |
| Success | C0 |
| Error | C0 |
| Subnet Mask (4 words) | V400 |

| Parameter | | DL205 Range |
|-----------------------|---------------|-------------------------------------|
| ECOM100# | K | K0-255 |
| Workspace | V | See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Subnet Mask (4 Words) | V | See DL205 V-memory map - Data Words |

ECRDSNM Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

5

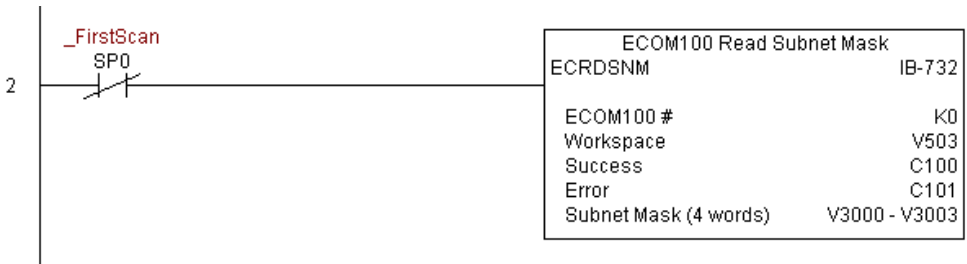


NOTE: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.

Rung 2: On the second scan, read the Subnet Mask of the ECOM100 and store it in V3000 thru V3003 (4 decimal numbers). The ECOM100's Subnet Mask could be displayed by an HMI.

The ECRDSNM is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to read the Subnet Mask will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101.



ECOM100 Write Description (ECWRDES) (IB-727)

ECOM100 Write Description will write the given Description to the ECOM100 module on a leading edge transition to the IBox. If you use a dollar sign (\$) or double quote ("), use the PRINT/VPRINT escape sequence of TWO dollar signs (\$\$) for a single dollar sign or dollar sign-double quote (\$") for a double quote character.

✗ 230

✗ 240

✓ 250-1

✓ 260

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).

The Description is stored in Flash-ROM in the ECOM100 and the execution of this IBox will disable the ECOM100 module for at least a half second until it writes the Flash-ROM. Therefore, it is **HIGHLY RECOMMENDED** that you only execute this IBox **ONCE** on the second scan. Since it requires a **LEADING** edge to execute, use a **NORMALLY CLOSED SP0** (STR NOT First Scan) to drive the power flow to the IBox.

In order for this ECOM100 IBox to function, you must turn **ON** dip switch 7 on the ECOM100 circuit board.

ECWRDES Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not completed successfully
- Error Code: specifies the location where the Error Code will be written
- Description: specifies the Description that will be written to the module

ECOM100 Write Description
IB-727

ECWRDES

ECOM100 # K0

Workspace V503

Success C100

Error C101

Error Code V2000

Description MODBUS/TCP Network #2

| Parameter | DL205 Range |
|-------------|--|
| ECOM100# | K K0-255 |
| Workspace | V See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B See DL205 V-memory map |
| Error Code | V See DL205 V-memory map - Data Words |
| Description | Text |

ECWRDES Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

5



NOTE: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.

Rung 2: On the second scan, set the Module Description of the ECOM100. Typically this is done using NetEdit, but this IBox allows you to configure the module description in the ECOM100 using your ladder program.

The ECWRDES is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to write the module description will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the specific error code.



ECOM100 Write Gateway Address (ECWRGWA) (IB-731)

 230

 240

 250-1

 260

ECOM100 Write Gateway Address will write the given Gateway IP Address to the ECOM100 module on a leading edge transition to the IBox. See also ECOM100 IP Setup (ECIPSUP) IBox 717 to set up ALL of the TCP/IP parameters in a single instruction - IP Address, Subnet Mask, and Gateway Address.

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).

The Gateway Address is stored in Flash-ROM in the ECOM100 and the execution of this IBox will disable the ECOM100 module for at least a half second until it writes the Flash-ROM. Therefore, it is **HIGHLY RECOMMENDED** that you only execute this IBox **ONCE**, on the second scan. Since it requires a **LEADING** edge to execute, use a **NORMALLY CLOSED** SP0 (STR NOT First Scan) to drive the power flow to the IBox.

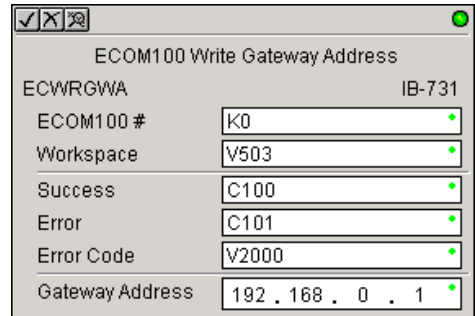
In order for this ECOM100 IBox to function, you must turn **ON** dip switch 7 on the ECOM100 circuit board.

5

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

ECWRGWA Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not completed successfully
- Error Code: specifies the location where the Error Code will be written
- Gateway Address: specifies the Gateway IP Address that will be written to the module

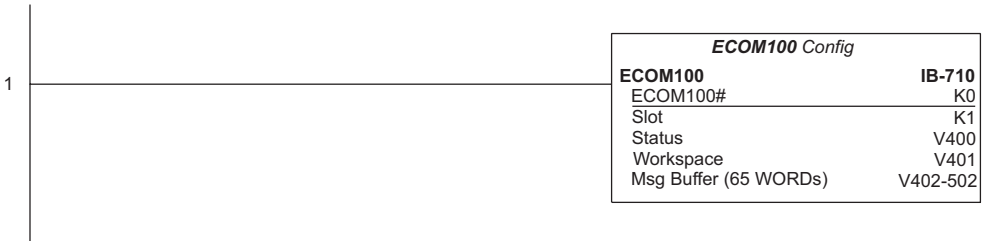


| Parameter | | DL205 Range |
|-----------------|---------------|-------------------------------------|
| ECOM100# | K | K0-255 |
| Workspace | V | See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error Code | V | See DL205 V-memory map - Data Words |
| Gateway Address | | 0.0.0.1. to 255.255.255.254 |

ECWRGWA Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

5

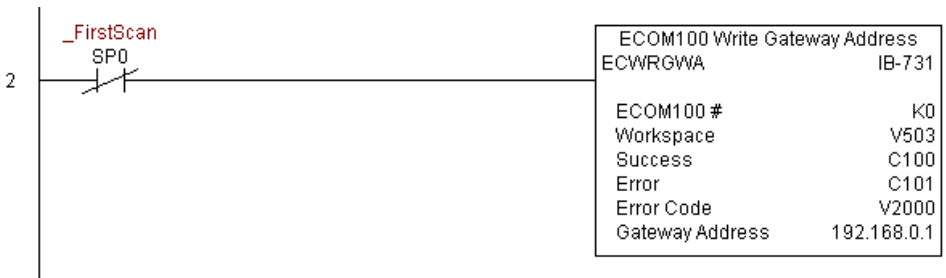


NOTE: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.

Rung 2: On the second scan, assign the Gateway Address of the ECOM100 to 192.168.0.1
The ECWRGWA is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to write the Gateway Address will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the specific error code.

To configure all of the ECOM100 TCP/IP parameters in one IBox, see the ECOM100 IP Setup (ECIPSUP) IBox.



ECOM100 Write IP Address (ECWRIP) (IB-723)

-  230
-  240
-  250-1
-  260

ECOM100 Write IP Address will write the given IP Address to the ECOM100 module on a leading edge transition to the IBox. See also ECOM100 IP Setup (ECIPSUP) IBox 717 to setup ALL of the TCP/IP parameters in a single instruction - IP Address, Subnet Mask, and Gateway Address.

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

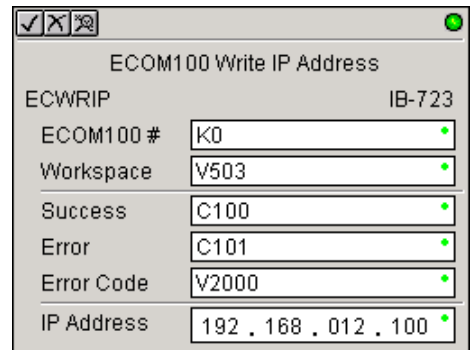
Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).

The IP Address is stored in Flash-ROM in the ECOM100 and the execution of this IBox will disable the ECOM100 module for at least a half second until it writes the Flash-ROM. Therefore, it is **HIGHLY RECOMMENDED** that you only execute this IBox **ONCE** on the second scan. Since it requires a **LEADING** edge to execute, use a **NORMALLY CLOSED** SP0 (STR NOT First Scan) to drive the power flow to the IBox.

In order for this ECOM100 IBox to function, you must turn **ON** dip switch 7 on the ECOM100 circuit board.

ECWRIP Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not successfully completed
- Error Code: specifies the location where the Error Code will be written



| ECOM100 Write IP Address | |
|--------------------------|-----------------------|
| ECWRIP | IB-723 |
| ECOM100 # | K0 |
| Workspace | V503 |
| Success | C100 |
| Error | C101 |
| Error Code | V2000 |
| IP Address | 192 . 168 . 012 . 100 |

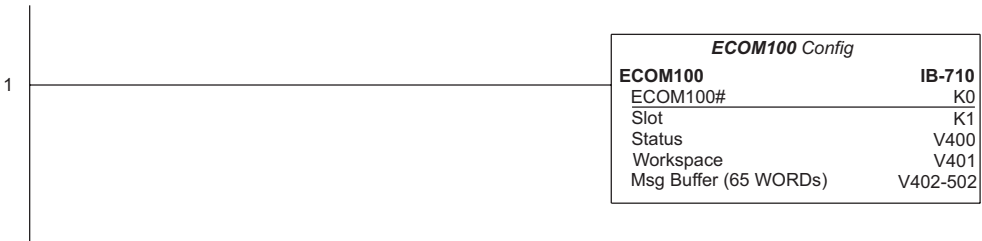
| Parameter | | DL205 Range |
|------------|---------------|-------------------------------------|
| ECOM100# | K | K0-255 |
| Workspace | V | See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error Code | V | See DL205 V-memory map - Data Words |
| IP Address | | 0.0.0.1. to 255.255.255.254 |

- IP Address: specifies the IP Address that will be written to the module

ECWRIP Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

5



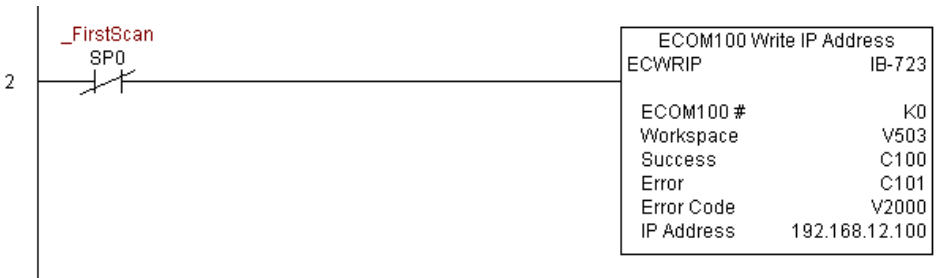
NOTE: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.

Rung 2: On the second scan, assign the IP Address of the ECOM100 to 192.168.12.100

The ECWRIP is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to write the IP Address will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the specific error code.

To configure all of the ECOM100 TCP/IP parameters in one IBox, see the ECOM100 IP Setup (ECIPSUP) IBox.



ECOM100 Write Module ID (ECWRMID) (IB-721)

ECOM100 Write Module ID will write the given Module ID on a leading edge transition to the IBox

230

240

250-1

260

If the Module ID is set in the hardware using the dipswitches, this IBox will fail and return error code 1005 (decimal).

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).

The Module ID is stored in Flash-ROM in the ECOM100 and the execution of this IBox will disable the ECOM100 module for at least a half second until it writes the Flash-ROM. Therefore, it is **HIGHLY RECOMMENDED** that you only execute this IBox **ONCE** on the second scan. Since it requires a **LEADING** edge to execute, use a **NORMALLY CLOSED SP0** (STR NOT First Scan) to drive the power flow to the IBox.

In order for this ECOM100 IBox to function, you must turn **ON** dip switch 7 on the ECOM100 circuit board.

ECWRMID Parameters

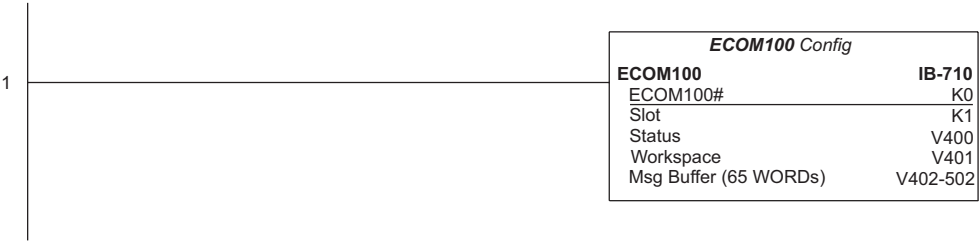
- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not completed successfully
- Error Code: specifies the location where the Error Code will be written
- Module ID: specifies the Module ID that will be written to the module

| Parameter | | DL205 Range |
|------------|---------------|-------------------------------------|
| ECOM100# | K | K0-255 |
| Workspace | V | See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error Code | V | See DL205 V-memory map - Data Words |
| Module ID | | K0-65535 |

ECWRMID Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

5



NOTE: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.

Rung 2: On the second scan, set the Module ID of the ECOM100. Typically this is done using NetEdit, but this IBox allows you to configure the module ID of the ECOM100 using your ladder program.

The ECWRMID is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to write the module ID will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the specific error code.



ECOM100 Write Name (ECWRNAM) (IB-725)

✗ 230

✗ 240

✓ 250-1

✓ 260

ECOM100 Write Name will write the given Name to the ECOM100 module on a leading edge transition to the IBox. If you use a dollar sign (\$) or double quote ("), use the PRINT/VPRINT escape sequence of TWO dollar signs (\$\$) for a single dollar sign or dollar sign-double quote (\$") for a double quote character.

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).

The Name is stored in Flash-ROM in the ECOM100 and the execution of this IBox will disable the ECOM100 module for at least a half second until it writes the Flash-ROM. Therefore, it is **HIGHLY RECOMMENDED** that you only execute this IBox **ONCE** on the second scan. Since it requires a **LEADING** edge to execute, use a **NORMALLY CLOSED SP0 (STR NOT First Scan)** to drive the power flow to the IBox.

In order for this ECOM100 IBox to function, you must turn **ON** dip switch 7 on the ECOM100 circuit board.

ECWRNAM Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number.
- Workspace: specifies a V-memory location that will be used by the instruction.
- Success: specifies a bit that will turn on once the request is completed successfully.
- Error: specifies a bit that will turn on if the instruction is not completed successfully.
- Error Code: specifies the location where the Error Code will be written.
- Module Name: specifies the Name that will be written to the module.

ECOM100 Write Name

ECWRNAM IB-725

ECOM100 # K0

Workspace V503

Success C100

Error C101

Error Code V2000

Module Name George

| Parameter | | DL205 Range |
|-------------|---------------|-------------------------------------|
| ECOM100# | K | K0-255 |
| Workspace | V | See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error Code | V | See DL205 V-memory map - Data Words |
| Module Name | | Text |

ECWRNAM Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.



NOTE: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.

Rung 2: On the second scan, set the Module Name of the ECOM100. Typically this is done using NetEdit, but this IBox allows you to configure the module name of the ECOM100 using your ladder program.

The ECWRNAM is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to write the module name will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the specific error code.



ECOM100 Write Subnet Mask (ECWRSNM) (IB-733)

ECOM100 Write Subnet Mask will write the given Subnet Mask to the ECOM100 module on a leading edge transition to the IBox. See also ECOM100 IP Setup (ECIPSUP) IBox 717 to set up ALL of the TCP/IP parameters in a single instruction - IP Address, Subnet Mask, and Gateway Address.

- ☒ 230
- ☒ 240
- ☒ 250-1
- ☒ 260

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).

The Subnet Mask is stored in Flash-ROM in the ECOM100 and the execution of this IBox will disable the ECOM100 module for at least a half second until it writes the Flash-ROM. Therefore, it is **HIGHLY RECOMMENDED** that you only execute this IBox **ONCE** on the second scan. Since it requires a **LEADING** edge to execute, use a **NORMALLY CLOSED SP0 (STR NOT First Scan)** to drive the power flow to the IBox.

In order for this ECOM100 IBox to function, you must turn **ON** dip switch 7 on the ECOM100 circuit board.

ECWRSNM Parameters

- **ECOM100#**: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number.
- **Workspace**: specifies a V-memory location that will be used by the instruction.
- **Success**: specifies a bit that will turn on once the request is completed successfully.
- **Error**: specifies a bit that will turn on if the instruction is not completed successfully.
- **Error Code**: specifies the location where the Error Code will be written.
- **Subnet Mask**: specifies the Subnet Mask that will be written to the module.

ECOM100 Write Subnet Mask
IB-733

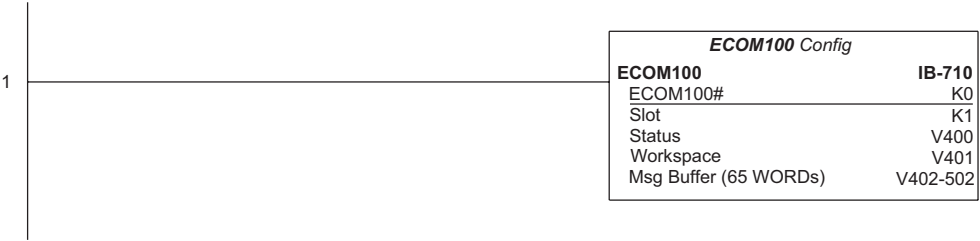
| | |
|-------------|-------------------|
| ECWRSNM | IB-733 |
| ECOM100 # | K0 |
| Workspace | V503 |
| Success | C100 |
| Error | C101 |
| Error Code | V2000 |
| Subnet Mask | 255 . 255 . 0 . 0 |

| Parameter | | DL205 Range |
|-------------|---------------|-------------------------------------|
| ECOM100# | K | K0-255 |
| Workspace | V | See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error Code | V | See DL205 V-memory map - Data Words |
| Subnet Mask | | Masked IP Address |

ECWRSNM Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

5



NOTE: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.

Rung 2: On the second scan, assign the Subnet Mask of the ECOM100 to 255.255.0.0

The ECWRSNM is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to write the Subnet Mask will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the specific error code.

To configure all of the ECOM100 TCP/IP parameters in one IBox, see the ECOM100 IP Setup (ECIPSUP) IBox.



ECOM100 RX Network Read (ECRX) (IB-740)

ECOM100 RX Network Read performs the RX instruction with built-in interlocking with all other ECOM100 RX (ECRX) and ECOM100 WX (ECWX) IBoxes in your program to simplify communications networking. It will perform the RX on the specified ECOM100#'s network, which corresponds to a specific unique ECOM100 Configuration (ECOM100) IBox at the top of your program.

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

Whenever this IBox has power, it will read element data from the specified slave into the given destination V-memory buffer, giving other ECOM100 RX and ECOM100 WX IBoxes on that ECOM100# network a chance to execute.

For example, if you wish to read and write data continuously from five different slaves, you can have all of these ECRX and ECWX instructions in ONE RUNG driven by SP1 (Always On). They will execute round-robin style, automatically!

ECRX Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Slave ID: specifies the slave ECOM(100) PLC that will be targeted by the ECRX instruction
- From Slave Element (Src): specifies the slave address of the data to be read
- Number of Bytes: specifies the number of bytes to read from the slave ECOM(100) PLC
- To Master Element (Dest): specifies the location where the slave data will be placed in the master ECOM100 PLC
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not completed successfully

| ECOM100 RX Network Read | |
|--------------------------|-------|
| IB-740 | |
| ECRX | |
| ECOM100 # | K0 |
| Workspace | V503 |
| Slave ID | K7 |
| From Slave Element (Src) | X0 |
| Number Of Bytes | K1 |
| To Master Element (Dest) | VC200 |
| Success | C100 |
| Error | C101 |

| Parameter | | DL205 Range |
|--------------------------|----------------------|-------------------------------------|
| ECOM100# | K | K0-255 |
| Workspace | V | See DL205 V-memory map - Data Words |
| Slave ID | K | K0-90 |
| From Slave Element (Src) | X,Y,C,S,T,CT,GX,GY,V | See DL205 V-memory map |
| Number of Bytes | K | K1-128 |
| To Master Element (Dest) | V | See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |

ECRX Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

5



(Example continued on next page)



NOTE: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in ***BOLD italics***, and the instruction name and ID will be in ***BOLD characters***.

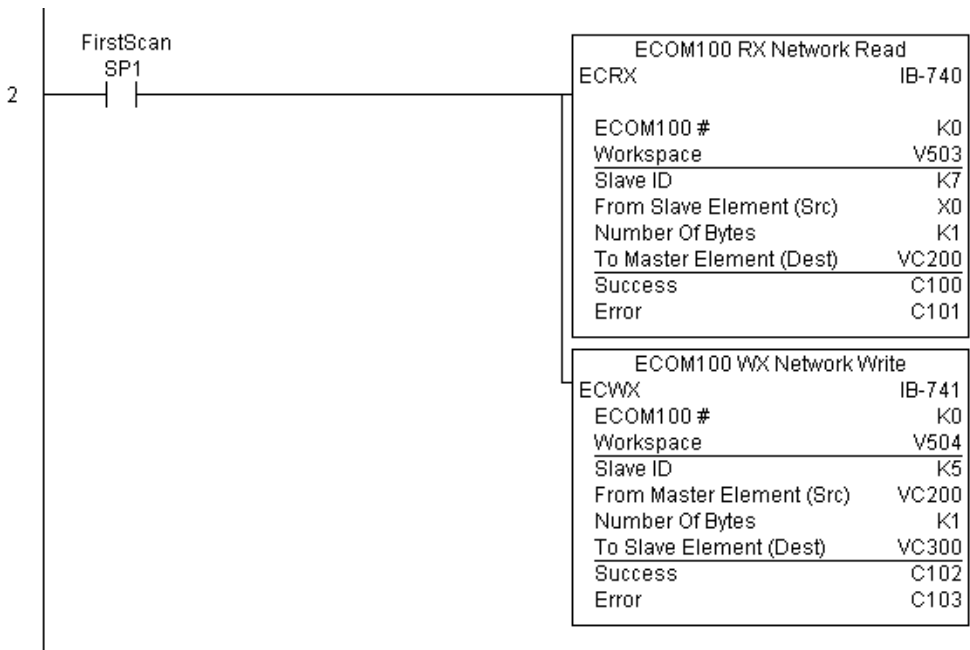
ECRX Example (cont'd)

Rung 2: Using ECOM100# K0, read X0-X7 from Slave K7 and write them to slave K5 as fast as possible. Store them in this local PLC in C200-C207, and write them to C300-C307 in slave K5.

Both the ECRX and ECWX work with the ECOM100 Config IBox to simplify all networking by handling all of the interlocks and proper resource sharing. They also provide very simplified error reporting. You no longer need to worry about any SP “busy bits” or “error bits,” or what slot number a module is in, or have any counters or shift registers or any other interlocks for resource management.

In this example, SP1 (always ON) is driving both the ECRX and ECWX IBoxes in the same rung. On the scan that the Network Read completes, the Network Write will start that same scan. As soon as the Network Write completes, any pending operations below it in the program would get a turn. If there are no pending ECOM100 IBoxes below the ECWX, then the very next scan the ECRX would start its request again.

Using the ECRX and ECWX for all of your ECOM100 network reads and writes is the fastest the PLC can do networking. For local Serial Ports, DCM modules, or the original ECOM modules, use the NETCFG and NETRX/NETWX IBoxes.



ECOM100 WX Network Write(ECWX) (IB-741)

ECOM100 WX Network Write performs the WX instruction with built-in interlocking with all other ECOM100 RX (ECRX) and ECOM100 WX (ECWX) IBoxes in your program to simplify communications networking. It will perform the WX on the specified ECOM100#'s network, which corresponds to a specific unique ECOM100 Configuration (ECOM100) IBox at the top of your program.

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

Whenever this IBox has power, it will write data from the master's V-memory buffer to the specified slave starting with the given slave element, giving other ECOM100 RX and ECOM100 WX IBoxes on that ECOM100# network a chance to execute.

For example, if you wish to read and write data continuously from five different slaves, you can have all of these ECRX and ECWX instructions in ONE RUNG driven by SP1 (Always On). They will execute round-robin style, automatically!

ECWX Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number.
- Workspace: specifies a V-memory location that will be used by the instruction.
- Slave ID: specifies the slave ECOM(100) PLC that will be targeted by the ECWX instruction.
- From Master Element (Src): specifies the location in the master ECOM100 PLC where the data will be sourced from.
- Number of Bytes: specifies the number of bytes to write to the slave ECOM(100) PLC.
- To Slave Element (Dest): specifies the slave address the data will be written to.
- Success: specifies a bit that will turn on once the request is completed successfully.
- Error: specifies a bit that will turn on if the instruction is not completed successfully.

| ECOM100 WX Network Write | |
|---------------------------|--------|
| ECWX | IB-741 |
| ECOM100 # | K0 |
| Workspace | V504 |
| Slave ID | K5 |
| From Master Element (Src) | VC200 |
| Number Of Bytes | K1 |
| To Slave Element (Dest) | VC300 |
| Success | C102 |
| Error | C103 |

| Parameter | | DL205 Range |
|---------------------------|----------------------|-------------------------------------|
| ECOM100# | K | K0-255 |
| Workspace | V | See DL205 V-memory map - Data Words |
| Slave ID | K | K0-90 |
| From Master Element (Src) | V | See DL205 V-memory map - Data Words |
| Number of Bytes | K | K1-128 |
| To Slave Element (Dest) | X,Y,C,S,T,CT,GX,GY,V | See DL205 V-memory map |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |

ECWX Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module number as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.



NOTE: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.

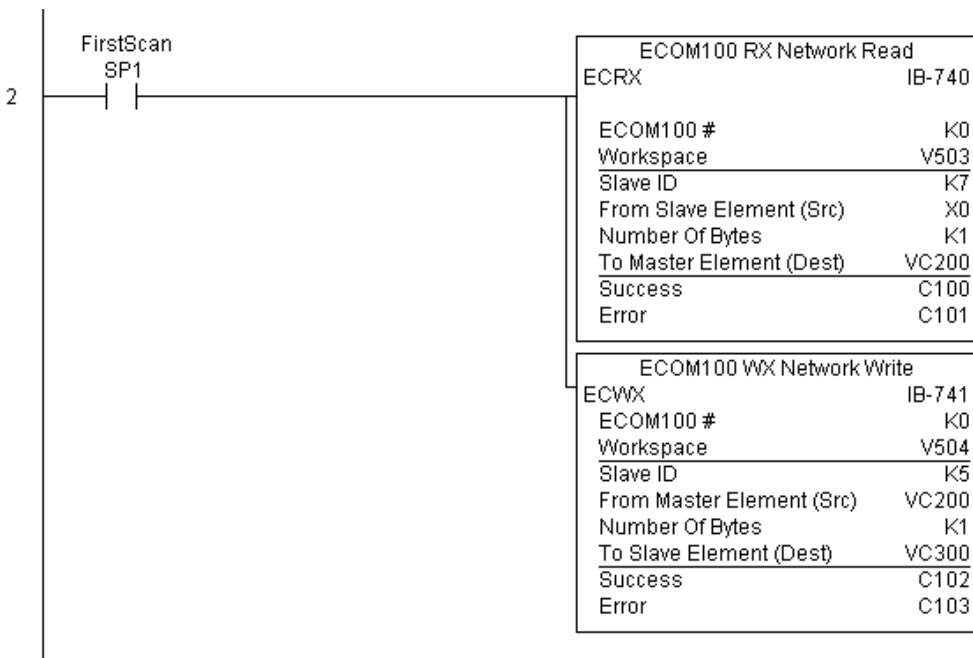
ECWX Example (cont'd)

Rung 2: Using ECOM100# K0, read X0-X7 from Slave K7 and write them to slave K5 as fast as possible. Store them in this local PLC in C200-C207, and write them to C300-C307 in slave K5.

Both the ECRX and ECWX work with the ECOM100 Config IBox to simplify all networking by handling all of the interlocks and proper resource sharing. They also provide very simplified error reporting. You no longer need to worry about any SP “busy bits” or “error bits,” or what slot number a module is in, or have any counters or shift registers or any other interlocks for resource management.

In this example, SP1 (always ON) is driving both the ECRX and ECWX IBoxes in the same rung. On the scan that the Network Read completes, the Network Write will start that same scan. As soon as the Network Write completes, any pending operations below it in the program would get a turn. If there are no pending ECOM100 IBoxes below the ECWX, then the very next scan the ECRX would start its request again.

Using the ECRX and ECWX for all of your ECOM100 network reads and writes is the fastest the PLC can do networking. For local Serial Ports, DCM modules, or the original ECOM modules, use the NETCFG and NETRX/NETWX IBoxes.



NETCFG Network Configuration (NETCFG) (IB-700)

Network Config defines all the common information necessary for performing RX/WX Networking using the NETRX and NETWX IBox instructions via a local CPU serial port, DCM or ECOM module.

 230

 240

 250-1

 260

You must have the Network Config instruction at the top of your ladder/stage program with any other configuration IBoxes.

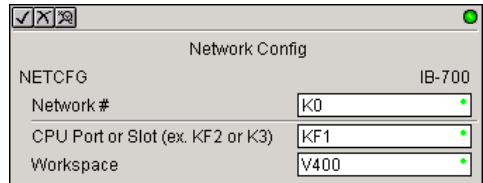
If you use more than one local serial port, DCM or ECOM in your PLC for RX/WX Networking, you must have a different Network Config instruction and Network number for EACH RX/WX network in your system that utilizes any NETRX/NETWX IBox instructions.

The second parameter “CPU Port or Slot” is the same value as in the high byte of the first LD instruction if you were coding the RX or WX rung yourself. This value is CPU and port specific. Use KF1 for local CPU serial port 2. Use K3 if a DCM or ECOM is located in slot 3 of a local 205 base.

The Workspace parameter is an internal, private register used by the Network Config IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

NETCFG Parameters

- Network#: specifies a unique number for each ECOM(100) or DCM network to use
- CPU Port or Slot: specifies the CPU port number or slot number of DCM/ECOM(100) used
- Workspace: specifies a V-memory location that will be used by the instruction



| Network Config | |
|----------------------------------|--------|
| NETCFG | IB-700 |
| Network # | K0 |
| CPU Port or Slot (ex. KF2 or K3) | KF1 |
| Workspace | V400 |

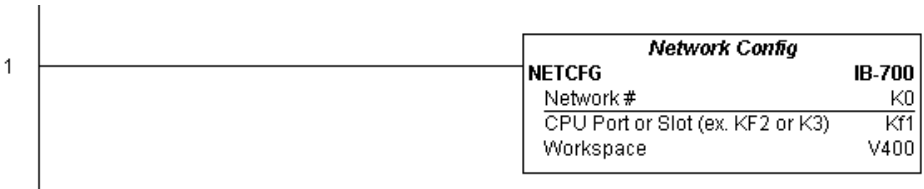
| Parameter | | DL205 Range |
|------------------|---|-------------------------------------|
| Network# | K | K0-255 |
| CPU Port or Slot | K | K0-FF |
| Workspace | V | See DL205 V-memory map - Data Words |

NETCFG Example

The Network Configuration IBox coordinates all of the interaction with other Network IBoxes (NETRX/NETWX). You must have a Network Configuration IBox for each serial port network, DCM module network, or original ECOM module network in your system. Configuration IBoxes must be at the top of your program and must execute every scan.

This IBox defines Network# K0 to be for the local CPU serial port #2 (KF1). For local CPU serial ports or DCM/ECOM modules, use the same value you would use in the most significant byte of the first LD instruction in a normal RX/WX rung to reference the port or module. Any NETRX or NETWX IBoxes that need to reference this specific network would enter K0 for their Network# parameter.

The Workspace register is used to maintain state information about the port or module, along with proper sharing and interlocking with the other NETRX and NETWX IBoxes in the program. This V-memory register must not be used anywhere else in the entire program.



NOTE: The Network Configuration IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.

Network RX Read (NETRX) (IB-701)

-  230
-  240
-  250-1
-  260

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

Network RX Read performs the RX instruction with built-in interlocking with all other Network RX (NETRX) and Network WX (NETWX) IBoxes in your program to simplify communications networking. It will perform the RX on the specified Network number, which corresponds to a specific unique Network Configuration (NETCFG) at the top of your program.

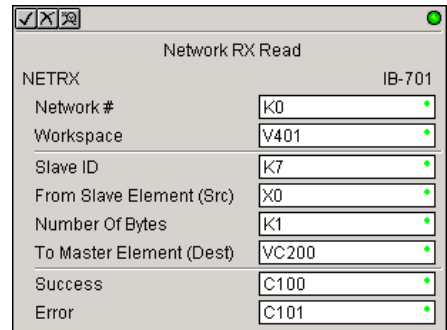
The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

Whenever this IBox has power, it will read element data from the specified slave into the given destination V-memory buffer, giving other Network RX and Network WX IBoxes on that Network number a chance to execute.

For example, if you wish to read and write data continuously from five different slaves, you can have all of these NETRX and NETWX instructions in ONE RUNG driven by SP1 (Always On). They will execute round-robin style, automatically!

NETRX Parameters

- Network#: specifies the (CPU ports, DCMs, ECOMs) Network # defined by the NETCFG instruction.
- Workspace: specifies a V-memory location that will be used by the instruction.
- Slave ID: specifies the slave PLC that will be targeted by the NETRX instruction.
- From Slave Element (Src): specifies the slave address of the data to be read.
- Number of Bytes: specifies the number of bytes to read from the slave device.
- To Master Element (Dest): specifies the location where the slave data will be placed in the master PLC.
- Success: specifies a bit that will turn on once the request is completed successfully.
- Error: specifies a bit that will turn on if the instruction is not completed successfully.

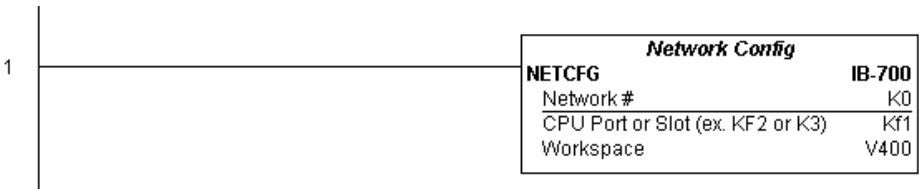


| Parameter | | DL205 Range |
|--------------------------|----------------------|-------------------------------------|
| Network# | K | K0-255 |
| Workspace | V | See DL205 V-memory map - Data Words |
| Slave ID | K,V | K0-90 |
| From Slave Element (Src) | X,Y,C,S,T,CT,GX,GY,V | See DL205 V-memory map |
| Number of Bytes | K | K1-128 |
| To Master Element (Dest) | V | See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |

NETRX Example

Rung 1: The Network Configuration IBox coordinates all of the interaction with other Network IBoxes (NETRX/NETWX). You must have a Network Configuration IBox for each serial port network, DCM module network, or original ECOM module network in your system. Configuration IBoxes must be at the top of your program and must execute every scan. This IBox defines Network# K0 to be for the local CPU serial port #2 (KF1). For local CPU serial ports or DCM/ECOM modules, use the same value you would use in the most significant byte of the first LD instruction in a normal RX/WX rung to reference the port or module. Any NETRX or NETWX IBoxes that need to reference this specific network would enter K0 for their Network# parameter.

The Workspace register is used to maintain state information about the port or module, along with proper sharing and interlocking with the other NETRX and NETWX IBoxes in the program. This V-memory register must not be used anywhere else in the entire program.



(Example continued on next page)



NOTE: The Network Configuration IBox instruction is used without a permissive contact. The top line will be identified in ***BOLD italics***, and the instruction name and ID will be in ***BOLD characters***.

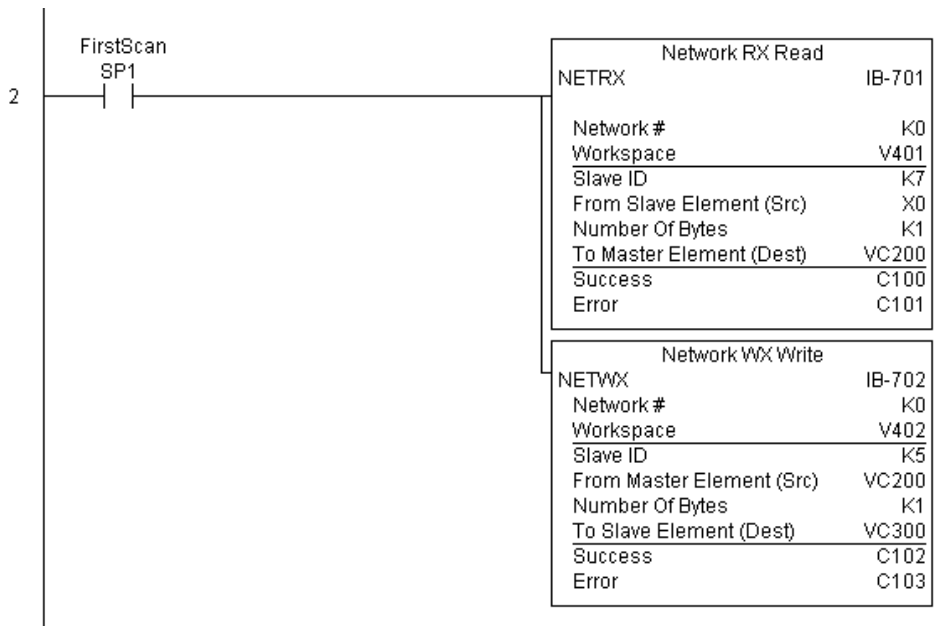
NETRX Example (cont'd)

Rung 2: Using Network# K0, read X0-X7 from Slave K7 and write them to slave K5 as fast as possible. Store them in this local PLC in C200-C207, and write them to C300-C307 in slave K5.

Both the NETRX and NETWX work with the Network Config IBox to simplify all networking by handling all of the interlocks and proper resource sharing. They also provide very simplified error reporting. You no longer need to worry about any SP “busy bits” or “error bits,” or what port number or slot number a module is in, or have any counters or shift registers or any other interlocks for resource management.

In this example, SP1 (always ON) is driving both the NETRX and NETWX IBoxes in the same rung. On the scan that the Network Read completes, the Network Write will start that same scan. As soon as the Network Write completes, any pending operations below it in the program would get a turn. If there are no pending NETRX or NETWX IBoxes below this IBox, then the very next scan the NETRX would start its request again.

Using the NETRX and NETWX for all of your serial port, DCM, or original ECOM network reads and writes is the fastest the PLC can do networking. For ECOM100 modules, use the ECOM100 and ECRX/ECWX IBoxes.



Network WX Write (NETWX) (IB-702)

Network WX Write performs the WX instruction with built-in interlocking with all other Network RX (NETRX) and Network WX (NETWX) IBoxes in your program to simplify communications networking. It will perform the WX on the specified Network number, which corresponds to a specific unique Network Configuration (NETCFG) at the top of your program.

The Workspace parameter is an internal, private register used by this IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

Whenever this IBox has power, it will write data from the master's V-memory buffer to the specified slave starting with the given slave element, giving other Network RX and Network WX IBoxes on that Network number a chance to execute.

For example, if you wish to read and write data continuously from five different slaves, you can have all of these NETRX and NETWX instructions in ONE RUNG driven by SP1 (Always On). They will execute round-robin style, automatically!

☐ 230

☐ 240

☒ 250-1

☒ 260

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

5

NETWX Parameters

- Network#: specifies the (CPU ports, DCMs, ECOMs) Network # defined by the NETCFG instruction
- Workspace: specifies a V-memory location that will be used by the instruction
- Slave ID: specifies the slave PLC that will be targeted by the NETWX instruction
- From Master Element (Src): specifies the location in the master PLC where the data will be sourced
- Number of Bytes: specifies the number of bytes to write to the slave PLC
- To Slave Element (Dest): specifies the slave address the data will be written to
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not completed successfully

| Network WX Write | | IB-702 |
|---------------------------|-------|--------|
| Network # | K0 | |
| Workspace | V402 | |
| Slave ID | K5 | |
| From Master Element (Src) | VC200 | |
| Number Of Bytes | K1 | |
| To Slave Element (Dest) | VC300 | |
| Success | C102 | |
| Error | C103 | |

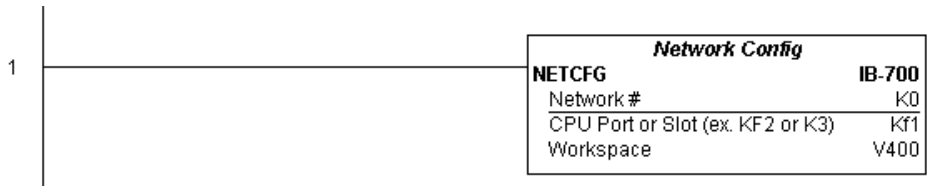
| Parameter | | DL205 Range |
|---------------------------|----------------------|-------------------------------------|
| Network# | K | K0-255 |
| Workspace | V | See DL205 V-memory map - Data Words |
| Slave ID | K,V | K0-90 |
| From Master Element (Src) | V | See DL205 V-memory map - Data Words |
| Number of Bytes | K | K1-128 |
| To Slave Element (Dest) | X,Y,C,S,T,CT,GX,GY,V | See DL205 V-memory map |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |

NETWX Example

Rung 1: The Network Configuration IBox coordinates all of the interaction with other Network IBoxes (NETRX/NETWX). You must have a Network Configuration IBox for each serial port network, DCM module network, or original ECOM module network in your system. Configuration IBoxes must be at the top of your program and must execute every scan.

This IBox defines Network# K0 to be for the local CPU serial port #2 (KF1). For local CPU serial ports or DCM/ECOM modules, use the same value you would use in the most significant byte of the first LD instruction in a normal RX/WX rung to reference the port or module. Any NETRX or NETWX IBoxes that need to reference this specific network would enter K0 for their Network# parameter.

The Workspace register is used to maintain state information about the port or module, along with proper sharing and interlocking with the other NETRX and NETWX IBoxes in the program. This V-memory register must not be used anywhere else in the entire program.



(Example continued on next page)



NOTE: The Network Configuration IBox instruction is used without a permissive contact. The top line will be identified in ***BOLD italics***, and the instruction name and ID will be in ***BOLD characters***.

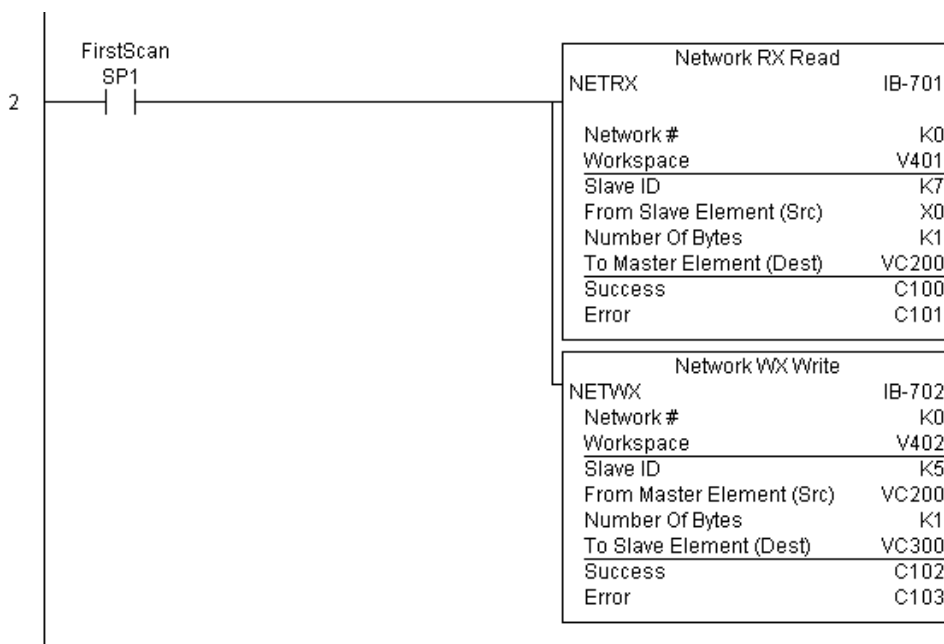
NETWX Example (cont'd)

Rung 2: Using Network# K0, read X0-X7 from Slave K7 and write them to slave K5 as fast as possible. Store them in this local PLC in C200-C207, and write them to C300-C307 in slave K5.

Both the NETRX and NETWX work with the Network Config IBox to simplify all networking by handling all of the interlocks and proper resource sharing. They also provide very simplified error reporting. You no longer need to worry about any SP “busy bits” or “error bits”, or what port number or slot number a module is in, or have any counters or shift registers or any other interlocks for resource management.

In this example, SP1 (always ON) is driving both the NETRX and NETWX IBoxes in the same rung. On the scan that the Network Read completes, the Network Write will start that same scan. As soon as the Network Write completes, any pending operations below it in the program would get a turn. If there are no pending NETRX or NETWX IBoxes below this IBox, then the very next scan the NETRX would start its request again.

Using the NETRX and NETWX for all of your serial port, DCM, or original ECOM network reads and writes is the fastest the PLC can do networking. For ECOM100 modules, use the ECOM100 and ECRX/ECWX IBoxes.



CTRIO Configuration (CTRIO) (IB-1000)

CTRIO Config defines all the common information for one specific CTRIO module which is used by the other CTRIO IBox instructions (for example, CTRLDPR - CTRIO Load Profile, CTREDRL - CTRIO Edit and Reload Preset Table, CTRRTL - CTRIO Run to Limit Mode, ...).

The Input/Output parameters for this instruction can be copied directly from the CTRIO Workbench configuration for this CTRIO module. Since the behavior is slightly different when the CTRIO module is in an EBC Base via an ERM, you must specify whether the CTRIO module is in a local base or in an EBC base.

CTRIO in Local Base

CTRIO in EBC Base

You must have the CTRIO Config IBox at the top of your ladder/stage program along with any other configuration IBoxes.

If you have more than one CTRIO in your PLC, you must have a different CTRIO Config IBox for EACH CTRIO module in your system that utilizes any CTRIO IBox instructions. Each CTRIO Config IBox must have a UNIQUE CTRIO# value. This is how the CTRIO IBoxes differentiate between the different CTRIO modules in your system.

The Workspace parameter is an internal, private register used by the CTRIO Config IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

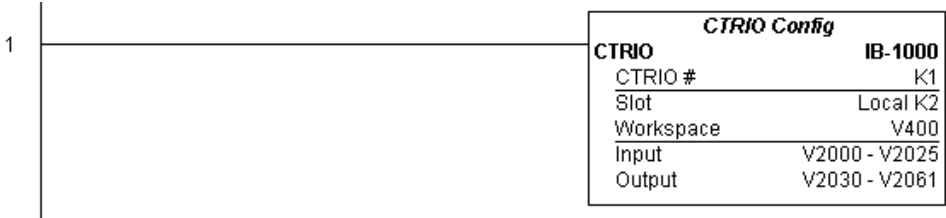
CTRIO Parameters

- CTRIO#: specifies a specific CTRIO module based on a user defined number
- Slot: (local base): specifies which PLC slot is occupied by the module (always K0 for EBC base)
- Workspace: specifies a V-memory location that will be used by the instruction
- CTRIO Location: specifies where the module is located (PLC local base or ERM to EBC base)
- Input (local base): This needs to be set to the same V-memory register as is specified in CTRIO Workbench as 'Starting V address for inputs' for this unique CTRIO.
- Output (local base): This needs to be set to the same V-memory register as is specified in CTRIO Workbench as 'Starting V address for outputs' for this unique CTRIO.
- Word Input (EBC base): The starting input V-memory address as defined by the I/O configuration in the ERM Workbench
- Bit Input (EBC base): The starting input Bit address as defined by the I/O configuration in the ERM Workbench
- Word Output (EBC base): The starting output V-memory address as defined by the I/O configuration in the ERM Workbench
- Bit Output (EBC base): The starting output Bit address as defined by the I/O configuration in the ERM Workbench

| Parameter | | DL205 Range |
|--------------------|-----|-------------------------------------|
| CTRIO# | K | K0-255 |
| Slot | K | K0-7 |
| Workspace | V | See DL205 V-memory map - Data Words |
| Input (Word, Bit) | V,B | See DL205 V-memory map - Data Words |
| Output (Word, Bit) | V,B | See DL205 V-memory map - Data Words |

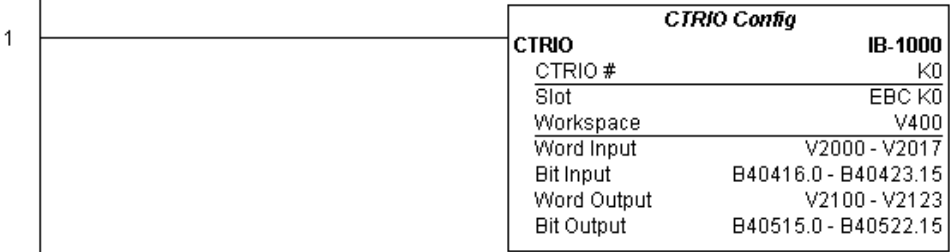
CTRIO Example (local base)

Rung 1: This sets up the CTRIO module in slot 2 of the local base. Each CTRIO module in the system will need a separate CTRIO Config IBox before any CTRxxxx IBoxes can be used. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



CTRIO Example (EBC base)

Overview: ERM Workbench must first be used to assign memory addresses to the I/O modules in the EBC base. Once the CTRIO module memory addresses are established using ERM Workbench, they are used in CTRIO Workbench and in a CTRIO IBox instruction to configure and define a specific CTRIO module. For this example, the CTRIO module uses V2000 - V2017 for its Word Input data and B40416.0 - B40423.15 for its Bit Input data. The module uses V2100 - V2123 for its Word Output data and B40515.0 - B40522.15 for its Bit Output data. The starting addresses, V2000 and V40416 (for inputs) and V2100 and V40515 (for outputs) are entered into CTRIO Workbench I/O Map to configure this specific CTRIO module. These starting addresses are the memory locations used in the CTRIO IBox instruction as the Word Input, Bit Input, Word Output and Bit Output addresses as shown below. For more information on this topic, refer to the CTRIO User Manual “Program Control” chapter.



NOTE: The CTRIO Configuration IBox instructions do not require a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD** characters.

CTRIO Add Entry to End of Preset Table (CTRADPT) (IB-1005)

CTRIO Add Entry to End of Preset Table, on a leading edge transition to this IBox, will append an entry to the end of a memory based Preset Table on a specific CTRIO Output resource. This IBox will take more than one PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.

-  230
-  240
-  250-1
-  260

Entry Type:

K0: Set

K1: Reset

K2: Pulse On (uses Pulse Time)

K3: Pulse Off (uses Pulse Time)

K4: Toggle

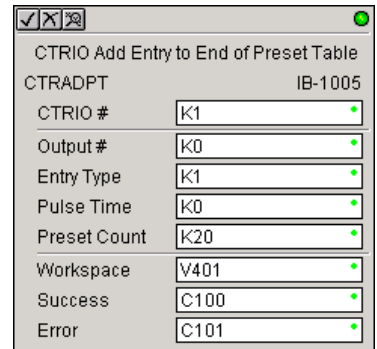
K5: Reset Count

Note that the Pulse Time parameter is ignored by some Entry Types.

The Workspace register is for internal use by this IBox instruction and MUST NOT be used anywhere else in your program.

CTRADPT Parameters

- CTRIO#: specifies a specific CTRIO module based on a user-defined number (see CTRIO Config)
- Output#: specifies a CTRIO output to be used by the instruction
- Entry Type: specifies the Entry Type to be added to the end of a Preset Table
- Pulse Time: specifies a pulse time in msec for the Pulse On and Pulse Off Entry Types
- Preset Count: specifies an initial count value to begin at after Reset
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the instruction has completed successfully
- Error: specifies a bit that will turn on if the instruction does not complete successfully



| Parameter | | DL205 Range |
|--------------|---------------|---|
| CTRIO# | K | K0-255 |
| Output# | K | K0-3 |
| Entry Type | V,K | K0-5; See DL205 V-memory map - Data Words |
| Pulse Time | V,K | K0-65535; See DL205 V-memory map - Data Words |
| Preset Count | V,K | K0-2147434528; See DL205 V-memory map |
| Workspace | V | See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |

CTRADPT Example

Rung 1: This sets up the CTRIO module in slot 2 of the local base. Each CTRIO module in the system will need a separate CTRIO Config IBox before any CTRxxxx IBoxes can be used. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.

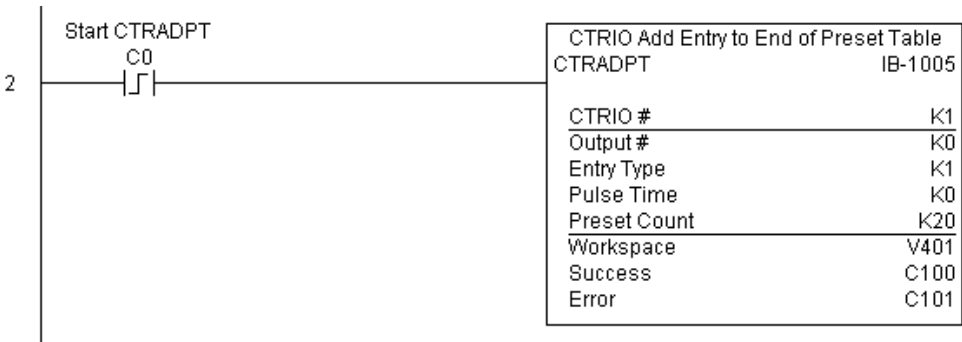


NOTE: The CTRIO Configuration IBox instruction does not require a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.

Rung 2: This rung is a sample method for enabling the CTRADPT command. A C-bit is used to allow the programmer to control the command from Data View for testing purposes.

Turning on C0 will cause the CTRADPT instruction to add a new preset to the preset table for output #0 on the CTRIO in slot 2. The new preset will be a command to RESET (entry type K1=reset), pulse time is left at zero as the reset type does not use this, and the count at which it will reset will be 20.

Operating procedure for this example code is to load the CTRADPT_ex1.cwb file to your CTRIO, then enter the code shown here, change to RUN mode, enable output #0 by turning on C2 in Data View, turn encoder on CTRIO to value above 10 and output #0 light will come on and stay on for all counts past 10. Now reset the counter with C1, enable C0 to execute CTRADPT command to add a reset for output #0 at a count of 20, turn on C2 to enable output #0, then turn encoder to value of 10+ (output #0 should turn on) and then continue on to count of 20+ (output #0 should turn off).



(Example continued on next page)

CTRADPT Example (cont'd)

Rung 3: This rung allows the programmer to reset the counter from the ladder logic.



Rung 4: This rung allows the operator to enable output #0 from the ladder code.



CTRIO Clear Preset Table (CTRCLRT) (IB-1007)

CTRIO Clear Preset Table will clear the RAM-based Preset Table on a leading edge transition to this IBox. This IBox will take more than one PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.




The Workspace register is for internal use by this IBox instruction and **MUST NOT** be used anywhere else in your program.

-  230
-  240
-  250-1
-  260

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

CTRCLRT Parameters

- CTRIO#: specifies a specific CTRIO module based on a user-defined number (see CTRIO Config)
- Output#: specifies a CTRIO output to be used by the instruction
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the instruction has completed successfully
- Error: specifies a bit that will turn on if the instruction does not complete successfully



CTRIO Clear Preset Table

CTRCLRT IB-1007

CTRIO #

Output #

Workspace

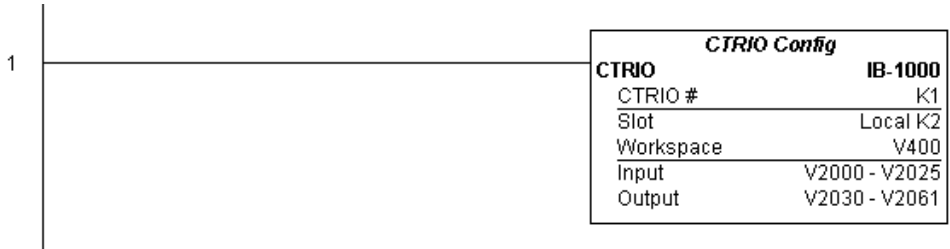
Success

Error

| Parameter | | DL205 Range |
|-----------|---------------|-------------------------------------|
| CTRIO# | K | K0-255 |
| Output# | K | K0-3 |
| Workspace | V | See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |

CTRCLRT Example

Rung 1: This sets up the CTRIO module in slot 2 of the local base. Each CTRIO module in the system will need a separate CTRIO Config IBox before any CTRxxxx IBoxes can be used. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



NOTE: The CTRIO Configuration IBox instruction does not require a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.

Rung 2: This rung is a sample method for enabling the CTRCLRT command. A C-bit is used to allow the programmer to control the command from Data View for testing purposes.

Turning on C0 will cause the CTRCLRT instruction to clear the preset table for output #0 on the CTRIO in slot 2.

Operating procedure for this example code is to load the CTRCLRT_ex1.cwb file to your CTRIO, then enter the code shown here, change to RUN mode, enable output #0 by turning on C2 in Data View, turn encoder on CTRIO to value above 10 and output #0 light will come on and stay on until a count of 20 is reached, where it will turn off. Now reset the counter with C1, enable C0 to execute CTRCLRT command to clear the preset table, turn on C2 to enable output #0, then turn encoder to value of 10+ (output #0 should NOT turn on).



CTRCLRT Example (cont'd)

Rung 3: This rung allows the programmer to reset the counter from the ladder logic.



Rung 4: This rung allows the operator to enable output #0 from the ladder code.



CTRIO Edit Preset Table Entry (CTREDPT) (IB-1003)



| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

CTRIO Edit Preset Table Entry, on a leading edge transition to this IBox, will edit a single entry in a Preset Table on a specific CTRIO Output resource. This IBox is good if you are editing more than one entry in a file at a time. If you wish to do just one edit and then reload the table immediately, see the CTRIO Edit and Reload Preset Table Entry (CTREDRL) IBox.

This IBox will take more than one PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.

Entry Type:

K0: Set

K1: Reset

K2: Pulse On (uses Pulse Time)

K3: Pulse Off (uses Pulse Time)

K4: Toggle

K5: Reset Count

Note that the Pulse Time parameter is ignored by some Entry Types.

The Workspace register is for internal use by this IBox instruction and **MUST NOT** be used anywhere else in your program.

CTREDPT Parameters

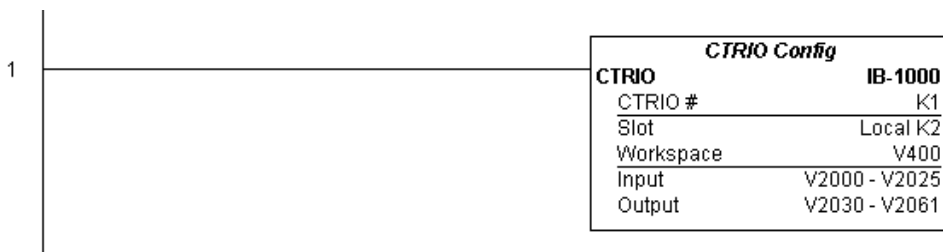
- CTRIO#: specifies a specific CTRIO module based on a user defined number (see CTRIO Config Ibox)
- Output#: specifies a CTRIO output to be used by the instruction
- Table#: specifies the Table number of which an Entry is to be edited
- Entry#: specifies the Entry location in the Preset Table to be edited
- Entry Type: specifies the Entry Type to add during the edit
- Pulse Time: specifies a pulse time in msec for the Pulse On and Pulse Off Entry Types
- Preset Count: specifies an initial count value to begin at after Reset
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the instruction has successfully completed
- Error: specifies a bit that will turn on if the instruction does not complete successfully

| CTRIO Edit Preset Table Entry | |
|-------------------------------|---------|
| IB-1003 | |
| CTREDPT | IB-1003 |
| CTRIO # | K1 |
| Output # | K0 |
| Table # | K1 |
| Entry # (0-based) | K1 |
| Entry Type | K1 |
| Pulse Time | K0 |
| Preset Count | K30 |
| Workspace | V401 |
| Success | C100 |
| Error | C101 |

| Parameter | | DL205 Range |
|--------------|---------------|---|
| CTRIO# | K | K0-255 |
| Output# | K | K0-3 |
| Table# | V,K | K0-255; See DL205 V-memory map - Data Words |
| Entry# | V,K | K0-255; See DL205 V-memory map - Data Words |
| Entry Type | V,K | K0-5; See DL205 V-memory map - Data Words |
| Pulse Time | V,K | K0-65535; See DL205 V-memory map - Data Words |
| Preset Count | V,K | K0-2147434528; See DL205 V-memory map |
| Workspace | V | See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |

CTREDPT Example

Rung 1: This sets up the CTRIO module in slot 2 of the local base. Each CTRIO module in the system will need a separate CTRIO Config IBox before any CTRxxxx IBoxes can be used. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



(Example continued on next page)



NOTE: The CTRIO Configuration IBox instruction does not require a permissive contact. The top line will be identified in ***BOLD italics***, and the instruction name and ID will be in ***BOLD characters***.

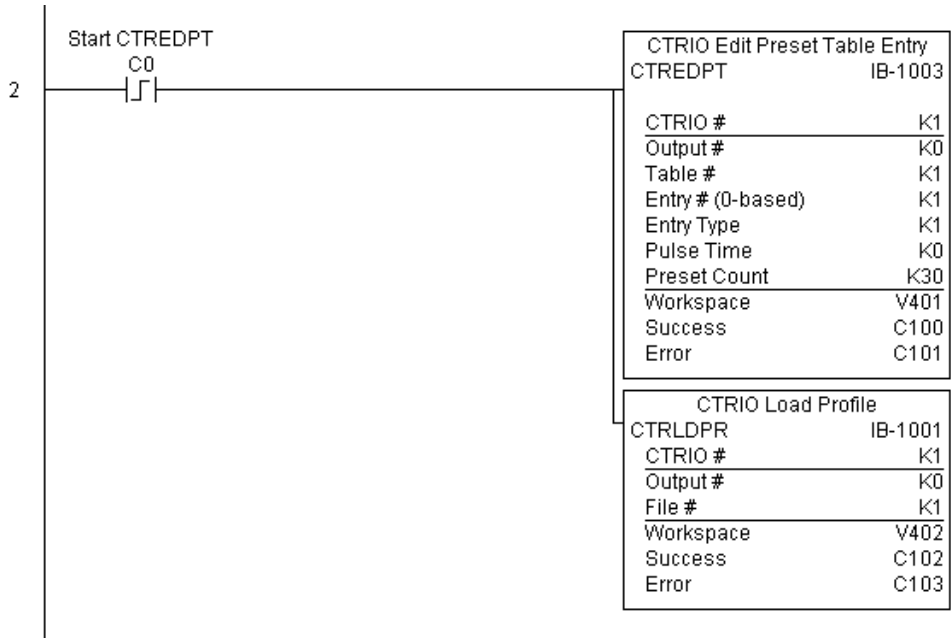
CTREDPT Example (cont'd)

Rung 2: This rung is a sample method for enabling the CTREDPT command. A C-bit is used to allow the programmer to control the command from Data View for testing purposes.

Turning on C0 will cause the CTREDPT instruction to change the second preset from a reset at a count of 20 to a reset at a count of 30 for output #0 on the CTRIO in slot 2.

Operating procedure for this example code is to load the CTREDPT_ex1.cwb file to your CTRIO, then enter the code shown here, change to RUN mode, enable output #0 by turning on C2 in Data View, turn encoder on CTRIO to value above 10 and output #0 light will come on and stay on until a count of 20 is reached, where it will turn off. Now reset the counter with C1, enable C0 to execute CTREDPT command to change the second preset, turn on C2 to enable output #0, then turn encoder to value of 10+ (output #0 should turn on) and then continue past a count of 30 (output #0 should turn off).

Note that we must also reload the profile after changing the preset(s); this is why the CTRLDPR command follows the CTREDPT command in this example.



CTREDPT Example (cont'd)

Rung 3: This rung allows the programmer to reset the counter from the ladder logic.



Rung 4: This rung allows the operator to enable output #0 from the ladder code.



CTRIO Edit Preset Table Entry and Reload (CTREDRL) (IB-1002)

CTRIO Edit Preset Table Entry and Reload, on a leading edge transition to this IBox, will perform this dual operation to a CTRIO Output resource in one CTRIO command. This IBox will take more than one PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.

☐ 230

☐ 240

☒ 250-1

☒ 260

Entry Type:

K0: Set

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

K1: Reset

K2: Pulse On (uses Pulse Time)

K3: Pulse Off (uses Pulse Time)

K4: Toggle

K5: Reset Count

Note that the Pulse Time parameter is ignored by some Entry Types.

The Workspace register is for internal use by this IBox instruction and **MUST NOT** be used anywhere else in your program.

CTREDRL Parameters

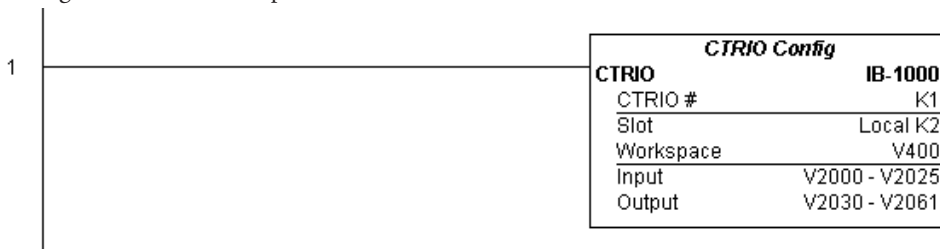
- CTRIO#: specifies a specific CTRIO module based on a user defined number (see CTRIO Config IBox)
- Output#: specifies a CTRIO output to be used by the instruction
- Table#: specifies the Table number of which an Entry is to be edited
- Entry#: specifies the Entry location in the Preset Table to be edited
- Entry Type: specifies the Entry Type to add during the edit
- Pulse Time: specifies a pulse time in msec for the Pulse On and Pulse Off Entry Types
- Preset Count: specifies an initial count value to begin at after Reset
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the instruction has completed successfully
- Error: specifies a bit that will turn on if the instruction does not complete successfully

| CTRIO Edit Preset Table Entry and Reload | |
|--|------|
| CTREDRL IB-1002 | |
| CTRIO # | K1 |
| Output # | K0 |
| Table # | K1 |
| Entry # (0-based) | K1 |
| Entry Type | K1 |
| Pulse Time | K0 |
| Preset Count | K30 |
| Workspace | V401 |
| Success | C100 |
| Error | C101 |

| Parameter | | DL205 Range |
|--------------|---------------|---|
| CTRIO# | K | K0-255 |
| Output# | K | K0-3 |
| Table# | V,K | K0-255; See DL205 V-memory map - Data Words |
| Entry# | V,K | K0-255; See DL205 V-memory map - Data Words |
| Entry Type | V,K | K0-5; See DL205 V-memory map - Data Words |
| Pulse Time | V,K | K0-65535; See DL205 V-memory map - Data Words |
| Preset Count | V,K | K0-2147434528; See DL205 V-memory map |
| Workspace | V | See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |

CTREDRL Example

Rung 1: This sets up the CTRIO module in slot 2 of the local base. Each CTRIO module in the system will need a separate CTRIO Config IBox before any CTRxxxx IBoxes can be used. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



(Example continued on next page)



NOTE: The CTRIO Configuration IBox instruction does not require a permissive contact. The top line will be identified in ***BOLD italics***, and the instruction name and ID will be in ***BOLD characters***.

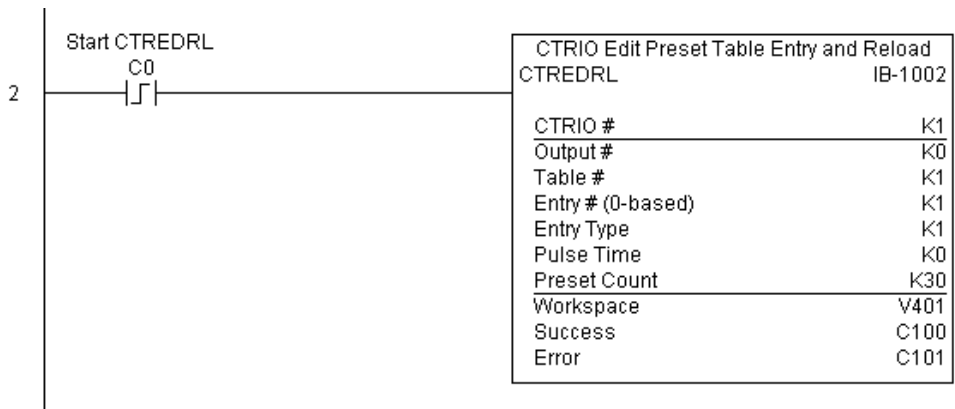
CTREDRL Example (cont'd)

Rung 2: This rung is a sample method for enabling the CTREDRL command. A C-bit is used to allow the programmer to control the command from Data View for testing purposes.

Turning on C0 will cause the CTREDRL instruction to change the second preset in file 1 from a reset value of 20 to a reset value of 30.

Operating procedure for this example code is to load the CTREDRL_ex1.cwb file to your CTRIO, then enter the code shown here, change to RUN mode, enable output #0 by turning on C2 in Data View, turn encoder on CTRIO to value above 10 and output #0 light will come on, continue to a count above 20 and the output #0 light will turn off. Now reset the counter with C1, enable C0 to execute CTREDRL command to change the second preset count value to 30, then turn encoder to value of 10+ (output #0 should turn on) and continue on to a value of 30+ and the output #0 light will turn off.

Note that it is not necessary to reload this file separately, however, the command can only change one value at a time.



(Example continued on next page)

CTREDRL Example (cont'd)

Rung 3: This rung allows the programmer to reset the counter from the ladder logic.



Rung 4: This rung allows the operator to enable output #0 from the ladder code.



CTRIO Initialize Preset Table (CTRINPT) (IB-1004)

CTRIO Initialize Preset Table, on a leading edge transition to this IBox, will create a single entry Preset Table in memory but not as a file, on a specific CTRIO Output resource. This IBox will take more than one PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.

✗ 230

✗ 240

✓ 250-1

✓ 260

Entry Type:

K0: Set

K1: Reset

K2: Pulse On (uses Pulse Time)

K3: Pulse Off (uses Pulse Time)

K4: Toggle

K5: Reset Count

Note that the Pulse Time parameter is ignored by some Entry Types.

The Workspace register is for internal use by this IBox instruction and **MUST NOT** be used anywhere else in your program.

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

CTRINPT Parameters

- CTRIO#: specifies a specific CTRIO module based on a user defined number (see CTRIO Config Ibox)
- Output#: specifies a CTRIO output to be used by the instruction
- Entry Type: specifies the Entry Type to add during the edit
- Pulse Time: specifies a pulse time in msec for the Pulse On and Pulse Off Entry Types
- Preset Count: specifies an initial count value to begin at after Reset
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the instruction has completed successfully
- Error: specifies a bit that will turn on if the instruction does not complete successfully

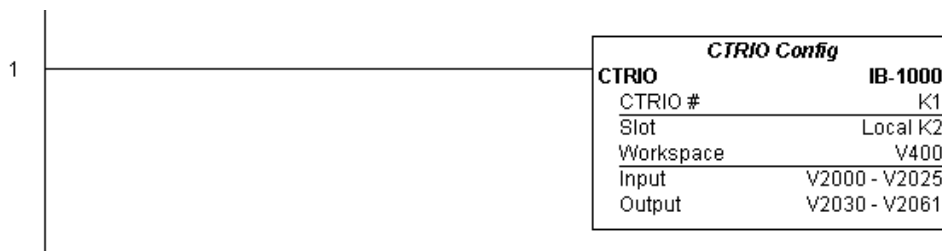
| CTRIO Initialize Preset Table | |
|-------------------------------|---------|
| CTRINPT | IB-1004 |
| CTRIO # | K1 |
| Output # | K0 |
| Entry Type | K0 |
| Pulse Time | K0 |
| Preset Count | K15 |
| Workspace | V401 |
| Success | C100 |
| Error | C101 |

| Parameter | | DL205 Range |
|--------------|---------------|---|
| CTRIO# | K | K0-255 |
| Output# | K | K0-3 |
| Entry Type | V,K | K0-5; See DL205 V-memory map - Data Words |
| Pulse Time | V,K | K0-65535; See DL205 V-memory map - Data Words |
| Preset Count | V,K | K0-2147434528; See DL205 V-memory map |
| Workspace | V | See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |

5

CTRINPT Example

Rung 1: This sets up the CTRIO module in slot 2 of the local base. Each CTRIO module in the system will need a separate CTRIO Config IBox before any CTRxxxx IBoxes can be used. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



(Example continued on next page)



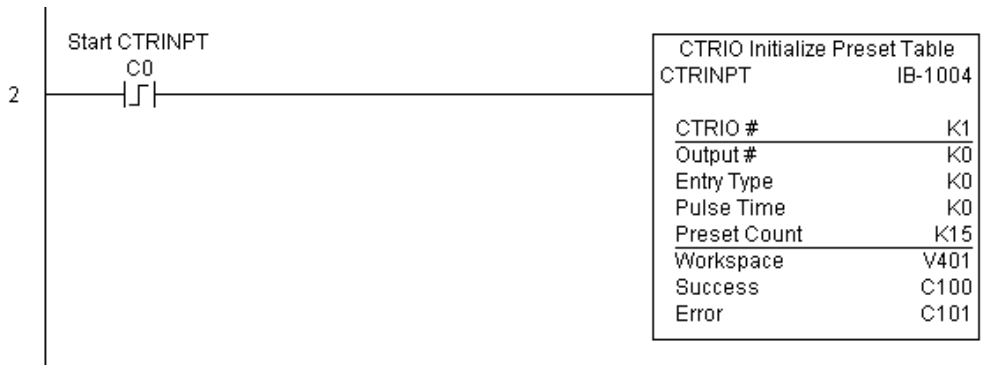
NOTE: The CTRIO Configuration IBox instruction does not require a permissive contact. The top line will be identified in ***BOLD italics***, and the instruction name and ID will be in ***BOLD characters***.

CTRINPT Example (cont'd)

Rung 2: This rung is a sample method for enabling the CTRINPT command. A C-bit is used to allow the programmer to control the command from Data View for testing purposes.

Turning on C0 will cause the CTRINPT instruction to create a single entry preset table, but not as a file, and use it for the output #0. In this case the single preset will be set at a count of 15 for output #0.

Operating procedure for this example code is to load the CTRINPT_ex1.cwb file to your CTRIO, then enter the code shown here, change to RUN mode, enable output #0 by turning on C2 in Data View, turn encoder on CTRIO to value above 15 and output #0 light will not come on. Now reset the counter with C1, enable C0 to execute CTRINPT command to create a single preset table with a preset to set output#0 at a count of 15, then turn encoder to value of 15+ (output #0 should turn on).



CTRINPT Example (cont'd)

Rung 3: This rung allows the programmer to reset the counter from the ladder logic.



Rung 4: This rung allows the operator to enable output #0 from the ladder code.



CTRIO Initialize Preset Table on Reset (CTRINTR) (IB-1010)

CTRIO Initialize Preset Table on Reset, on a leading edge transition to this IBox, will create a single entry Preset Table in memory but not as a file, on a specific CTRIO Output resource. This IBox will take more than 1 PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.

 230

 240

 250-1

 260

Entry Type:

K0: Set

K1: Reset

K2: Pulse On (uses Pulse Time)

K3: Pulse Off (uses Pulse Time)

K4: Toggle

K5: Reset Count

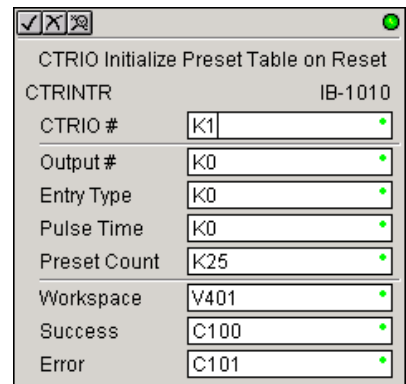
Note that the Pulse Time parameter is ignored by some Entry Types.

The Workspace register is for internal use by this IBox instruction and **MUST NOT** be used anywhere else in your program.

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

CTRINTR Parameters

- CTRIO#: specifies a specific CTRIO module based on a user defined number (see CTRIO Config IBox)
- Output#: specifies a CTRIO output to be used by the instruction
- Entry Type: specifies the Entry Type to add during the edit
- Pulse Time: specifies a pulse time in msec for the Pulse On and Pulse Off Entry Types
- Preset Count: specifies an initial count value to begin at after Reset
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the instruction has completed successfully
- Error: specifies a bit that will turn on if the instruction does not complete successfully



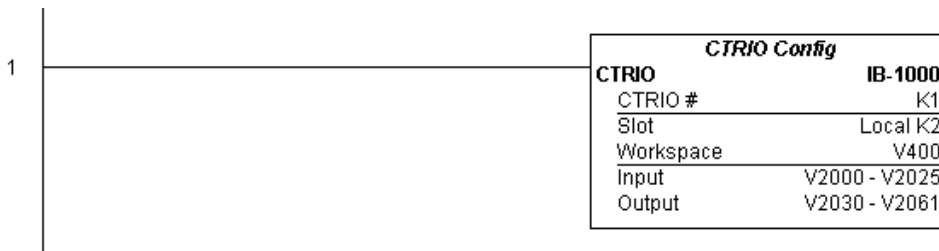
| CTRIO Initialize Preset Table on Reset | |
|--|------|
| CTRINTR IB-1010 | |
| CTRIO # | K1 |
| Output # | K0 |
| Entry Type | K0 |
| Pulse Time | K0 |
| Preset Count | K25 |
| Workspace | V401 |
| Success | C100 |
| Error | C101 |

| Parameter | | DL205 Range |
|--------------|---------------|---|
| CTRIO# | K | K0-255 |
| Output# | K | K0-3 |
| Entry Type | V,K | K0-5; See DL205 V-memory map - Data Words |
| Pulse Time | V,K | K0-65535; See DL205 V-memory map - Data Words |
| Preset Count | V,K | K0-2147434528; See DL205 V-memory map |
| Workspace | V | See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |

5

CTRINTR Example

Rung 1: This sets up the CTRIO module in slot 2 of the local base. Each CTRIO module in the system will need a separate CTRIO Config IBox before any CTRxxxx IBoxes can be used. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



(Example continued on next page)



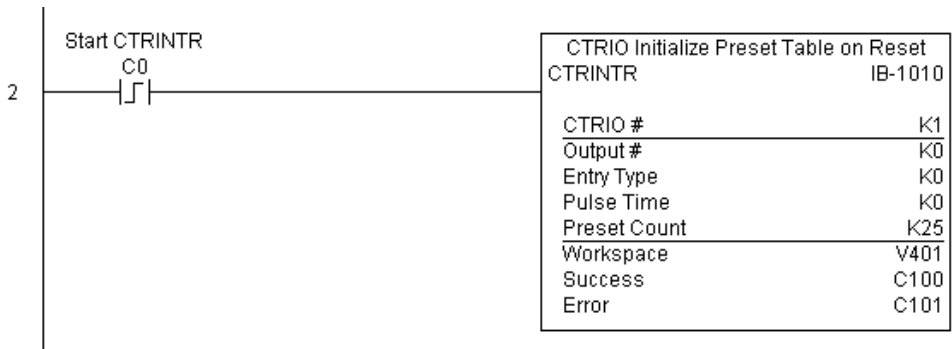
NOTE: The CTRIO Configuration IBox instruction does not require a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.

CTRINTR Example (cont'd)

Rung 2: This rung is a sample method for enabling the CTRINTR command. A C-bit is used to allow the programmer to control the command from Data View for testing purposes.

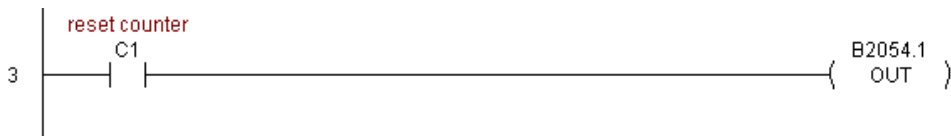
Turning on C0 will cause the CTRINTR instruction to create a single entry preset table, but not as a file, and use it for output #0, the new preset will be loaded when the current count is reset. In this case the single preset will be a set at a count of 25 for output #0.

Operating procedure for this example code is to load the CTRINTR_ex1.cwb file to your CTRIO, then enter the code shown here, change to RUN mode, enable output #0 by turning on C2 in Data View, turn encoder on CTRIO to value above 10 and output #0 light will come on. Now turn on C0 to execute the CTRINTR command, reset the counter with C1, then turn encoder to value of 25+ (output #0 should turn on).



CTRINTR Example (cont'd)

Rung 3: This rung allows the programmer to reset the counter from the ladder logic.



Rung 4: This rung allows the operator to enable output #0 from the ladder code.



CTRIO Load Profile (CTRLDPR) (IB-1001)

CTRIO Load Profile loads a CTRIO Profile File to a CTRIO Output resource on a leading edge transition to this IBox. This IBox will take more than one PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.

 230

 240

 250-1

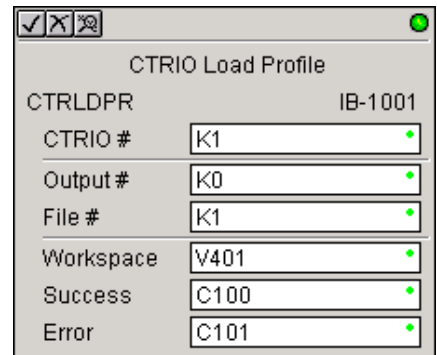
 260

The Workspace register is for internal use by this IBox instruction and **MUST NOT** be used anywhere else in your program.

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

CTRLDPR Parameters

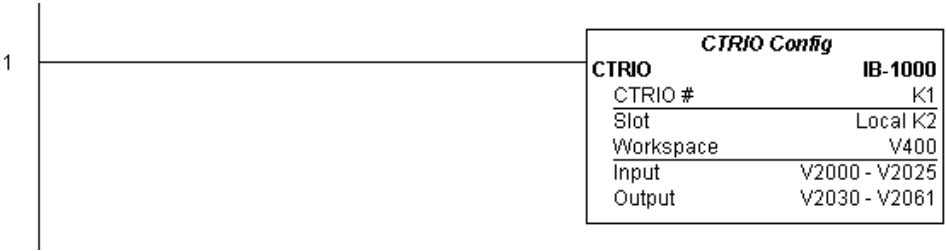
- **CTRIO#:** specifies a specific CTRIO module based on a user defined number (see CTRIO Config)
- **Output#:** specifies a CTRIO output to be used by the instruction
- **File#:** specifies a CTRIO profile File number to be loaded
- **Workspace:** specifies a V-memory location that will be used by the instruction
- **Success:** specifies a bit that will turn on once the instruction has completed successfully
- **Error:** specifies a bit that will turn on if the instruction does not complete successfully



| Parameter | | DL205 Range |
|-----------|---------------|---|
| CTRIO# | K | K0-255 |
| Output# | K | K0-3 |
| File# | V,K | K0-255; See DL205 V-memory map - Data Words |
| Workspace | V | See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |

CTRLDPR Example

Rung 1: This sets up the CTRIO module in slot 2 of the local base. Each CTRIO module in the system will need a separate CTRIO Config IBox before any CTRxxxx IBoxes can be used. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



NOTE: The CTRIO Configuration IBox instruction does not require a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.

Rung 2: This CTRIO Load Profile IBox will load File #1 into the working memory of Output 0 in CTRIO #1. This example program requires that you load CTRLDPR_IBox.cwb into your Hx-CTRIO(2) module.



(Example continued on next page)

CTRLDPR Example (cont'd)

Rung 3: If the file is loaded successfully, set Profile_Loaded.



CTRIO Read Error (CTRRDER) (IB-1014)

230

240

250-1

260

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

CTRIO Read Error Code, on a leading edge transition to this IBox, will read the decimal error code value (listed below) from the CTRIO module and place it in the specified Error Code register. This instruction is not supported when the CTRIO is used in an ERM/EBC configuration.

Since the Error Code in the CTRIO is only maintained until another CTRIO command is given, you must use this instruction immediately after the CTRIO IBox that reports an error via its Error bit parameter.

The Workspace register is for internal use by this IBox instruction and **MUST NOT** be used anywhere else in your program.

Error Codes:

0: No Error

100: Specified command code is unknown or unsupported

101: File number not found in the file system

102: File type is incorrect for specified output function

103: Profile type is unknown

104: Specified input is not configured as a limit on this output

105: Specified limit input edge is out of range

106: Specified input function is unconfigured or invalid

107: Specified input function number is out of range

108: Specified preset function is invalid

109: Preset table is full

110: Specified Table entry is out of range

111: Specified register number is out of range

112: Specified register is an unconfigured input or output

2001: Error reading Error Code - cannot access CTRIO via ERM

CTRRDER Parameters

- CTRIO#: specifies a specific CTRIO module based on a user-defined number (see CTRIO Config)
- Workspace: specifies a V-memory location that will be used by the instruction
- Error Code: specifies the location where the Error Code will be written

CTRIO Read Error Code

CTRRDER IB-1014

CTRIO # K1

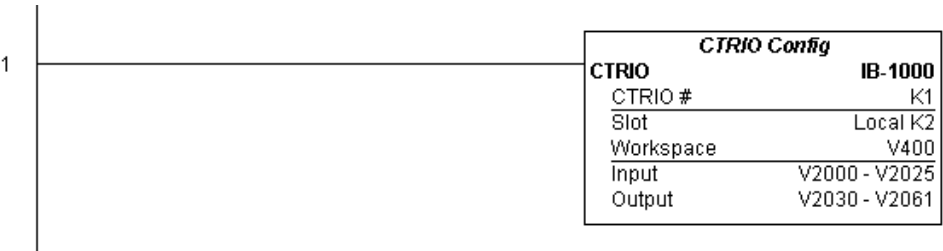
Workspace V401

Error Code V402

| Parameter | | DL205 Range |
|------------|---|-------------------------------------|
| CTRIO# | K | K0-255 |
| Workspace | V | See DL205 V-memory map - Data Words |
| Error Code | V | See DL205 V-memory map - Data Words |

CTRRDER Example

Rung 1: This sets up the CTRIO module in slot 2 of the local base. Each CTRIO module in the system will need a separate CTRIO Config IBox before any CTRxxxx IBoxes can be used. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



NOTE: The CTRIO Configuration IBox instruction does not require a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.

Rung 2: This CTRIO Read Error Code IBox will read the Extended Error information from CTRIO #1. This example program requires that you load CTRRDER_IBox.cwb into your Hx-CTRIO(2) module.



CTRIO Run to Limit Mode (CTRRTLTM) (IB-1011)

CTRIO Run To Limit Mode, on a leading edge transition to this IBox, loads the Run to Limit command and given parameters on a specific Output resource. The CTRIO's Input(s) must be configured as Limit(s) for this function to work.

 230

 240

 250-1

 260

Valid Hexadecimal Limit Values:

K00 - Rising Edge of Ch1/C

K10 - Falling Edge of Ch1/C

K20 - Both Edges of Ch1/C

K01 - Rising Edge of Ch1/D

K11 - Falling Edge of Ch1/D

K21 - Both Edges of Ch1/D

K02 - Rising Edge of Ch2/C

K12 - Falling Edge of Ch2/C

K22 - Both Edges of Ch2/C

K03 - Rising Edge of Ch2/D

K13 - Falling Edge of Ch2/D

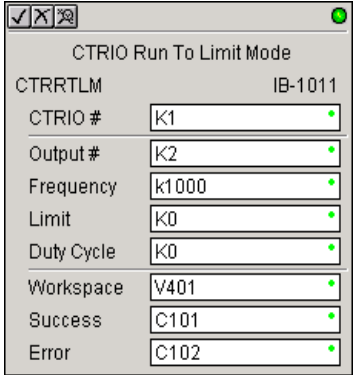
K23 - Both Edges of Ch2/D

This IBox will take more than one PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.

The Workspace register is for internal use by this IBox instruction and MUST NOT be used anywhere else in your program.

CTRRTLTM Parameters

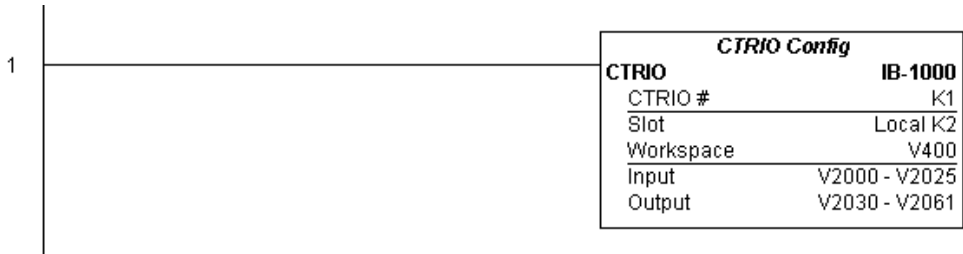
- CTRIO#: specifies a specific CTRIO module based on a user-defined number (see CTRIO Config IBox).
- Output#: specifies a CTRIO output to be used by the instruction.
- Frequency: specifies the output pulse rate (H2-CTRIO: 20Hz - 25KHz / H2-CTRIO2: 20Hz - 250 KHz).
- Limit: the CTRIO's Input(s) must be configured as Limit(s) for this function to operate.
- Duty Cycle: specifies the % of on time versus off time. This is a hex number. Default of 0 is 50%, also entering 50 will yield 50%. 50% duty cycle is defined as on half the time and off half the time.
- Workspace: specifies a V-memory location that will be used by the instruction.
- Success: specifies a bit that will turn on once the instruction has completed successfully.
- Error: specifies a bit that will turn on if the instruction does not complete successfully.



| Parameter | | DL205 Range |
|------------|---------------|--|
| CTRIO# | K | K0-255 |
| Output# | K | K0-3 |
| Frequency | V,K | K20-20000; See DL205 V-memory map - Data Words |
| Limit | V,K | K0-FF; See DL205 V-memory map - Data Words |
| Duty Cycle | V,K | K0-99; See DL205 V-memory map - Data Words |
| Workspace | V | See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |

CTRRTLM Example

Rung 1: This sets up the CTRIO module in slot 2 of the local base. Each CTRIO module in the system will need a separate CTRIO Config IBox before any CTRxxxx IBoxes can be used. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



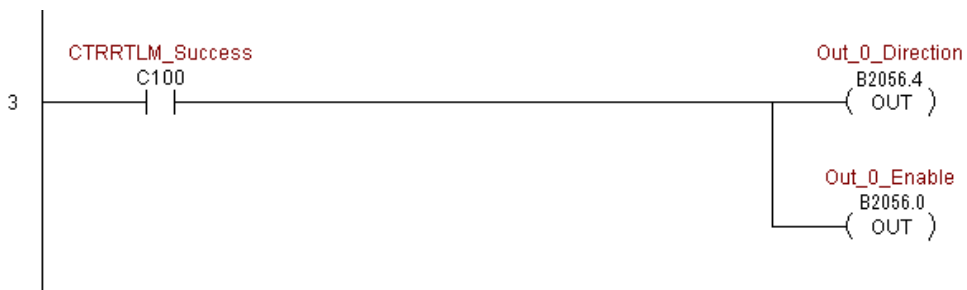
NOTE: The CTRIO Configuration IBox instruction does not require a permissive contact. The top line will be identified in ***BOLD italics***, and the instruction name and ID will be in ***BOLD characters***.

Rung 2: This CTRIO Run To Limit Mode IBox sets up Output #2 in CTRIO #1 to output pulses at a Frequency of 1000 Hz until Limit #0 comes on. This example program requires that you load CTRRTLM_IBox.cwb into your Hx-CTRIO(2) module.



CTRRTLM Example (cont'd)

Rung 3: If the Run To Limit Mode parameters are OK, set the Direction Bit and Enable the output.



CTRIO Run to Position Mode (CTRRTPM) (IB-1012)

CTRIO Run To Position Mode, on a leading edge transition to this IBox, loads the Run to Position command and given parameters on a specific Output resource.

 230

 240

 250-1

 260

Valid Function Values are:

00: Less Than Ch1/Fn1

10: Greater Than Ch1/Fn1

01: Less Than Ch1/Fn2

11: Greater Than Ch1/Fn2

02: Less Than Ch2/Fn1

12: Greater Than Ch2/Fn1

03: Less Than Ch2/Fn2

13: Greater Than Ch2/Fn2

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

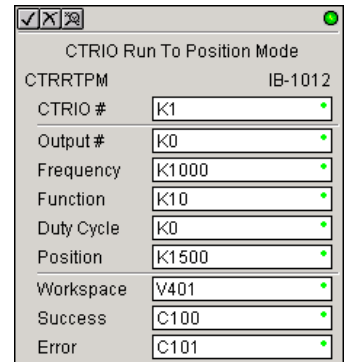
5

This IBox will take more than one PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.

The Workspace register is for internal use by this IBox instruction and **MUST NOT** be used anywhere else in your program.

CTRRTPM Parameters

- CTRIO#: specifies a specific CTRIO module based on a user-defined number (see CTRIO Config IBox).
- Output#: specifies a CTRIO output to be used by the instruction.
- Frequency: specifies the output pulse rate (H2-CTRIO: 20Hz - 25KHz / H2-CTRIO2: 20Hz - 250 KHz).
- Duty Cycle: specifies the % of on time versus off time. This is a hex number. Default of 0 is 50%, also entering 50 will yield 50%. 50% duty cycle is defined as on half the time and off half the time.
- Position: specifies the count value, as measured on the encoder input, at which the output pulse train will be turned off.
- Workspace: specifies a V-memory location that will be used by the instruction.
- Success: specifies a bit that will turn on once the instruction has completed successfully.
- Error: specifies a bit that will turn on if the instruction does not complete successfully.

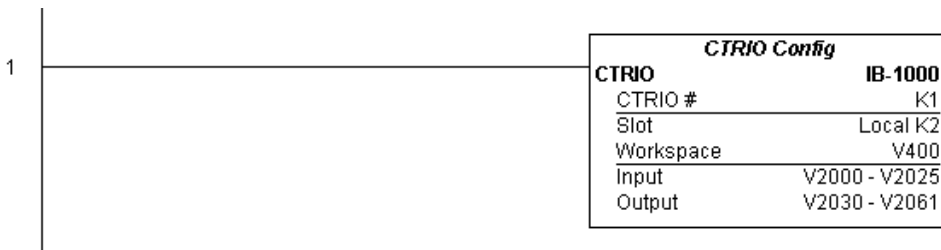


| CTRIO Run To Position Mode | |
|----------------------------|---------|
| IB-1012 | |
| CTRRTPM | IB-1012 |
| CTRIO # | K1 |
| Output # | K0 |
| Frequency | K1000 |
| Function | K10 |
| Duty Cycle | K0 |
| Position | K1500 |
| Workspace | V401 |
| Success | C100 |
| Error | C101 |

| Parameter | | DL205 Range |
|------------|---------------|--|
| CTRIO# | K | K0-255 |
| Output# | K | K0-3 |
| Frequency | V,K | K20-20000; See DL205 V-memory map - Data Words |
| Duty Cycle | V,K | K0-99; See DL205 V-memory map |
| Position | V,K | K0-2147434528; See DL205 V-memory map |
| Workspace | V | See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |

CTRRTPM Example

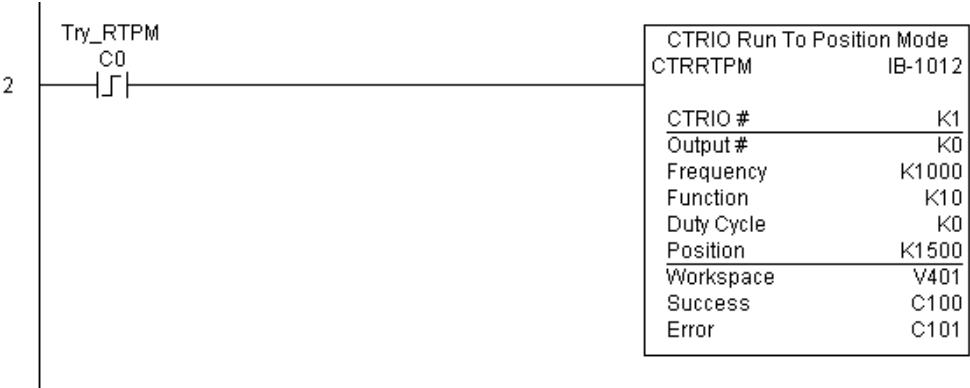
Rung 1: This sets up the CTRIO module in slot 2 of the local base. Each CTRIO module in the system will need a separate CTRIO Config IBox before any CTRxxxx IBoxes can be used. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



NOTE: The CTRIO Configuration IBox instruction does not require a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.

CTRRTPM Example (cont'd)

Rung 2: This CTRIO Run To Position Mode IBox sets up Output #0 in CTRIO #1 to output pulses at a Frequency of 1000 Hz, use the 'Greater than Ch1/Fn1' comparison operator, until the input position of 1500 is reached. This example program requires that you load CTRRTPM_IBox.cwb into your Hx-CTRIO(2) module.



Rung 3: If the Run To Position Mode parameters are OK, set the Direction Bit and Enable the output.



CTRIO Velocity Mode (CTRVELO) (IB-1013)

CTRIO Velocity Mode loads the Velocity command and given parameters on a specific Output resource on a leading edge transition to this IBox.

This IBox will take more than one PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.

The Workspace register is for internal use by this IBox instruction and **MUST NOT** be used anywhere else in your program.

- ☐ 230
- ☐ 240
- ☒ 250-1
- ☒ 260

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

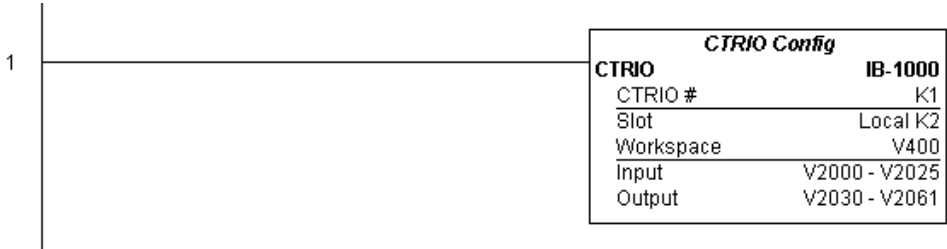
CTRVELO Parameters

- CTRIO#: specifies a specific CTRIO module based on a user defined number (see CTRIO Config Ibox)
- Output#: specifies a CTRIO output to be used by the instruction
- Frequency: specifies the output pulse rate (H2-CTRIO: 20Hz - 25KHz / H2-CTRIO2: 20Hz - 250 KHz)
- Duty Cycle: specifies the % of on time versus off time. This is a hex number. Default of 0 is 50%, also entering 50 will yield 50%. 50% duty cycle is defined as on half the time and off half the time
- Step Count: This DWORD value specifies the number of pulses to output. A Step Count value of -1 (or 0xFFFFFFFF) causes the CTRIO to output pulses continuously. Negative Step Count values must be V-Memory references.
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the instruction has successfully completed
- Error: specifies a bit that will turn on if the instruction does not complete successfully

| Parameter | | DL205 Range |
|------------|---------------|--|
| CTRIO# | K | K0-255 |
| Output# | K | K0-3 |
| Frequency | V,K | K20-20000; See DL205 V-memory map - Data Words |
| Duty Cycle | V,K | K0-99; See DL205 V-memory map |
| Step Count | V,K | K0-2147434528; See DL205 V-memory map |
| Workspace | V | See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |

CTRVELO Example

Rung 1: This sets up the CTRIO module in slot 2 of the local base. Each CTRIO module in the system will need a separate CTRIO Config IBox before any CTRxxxx IBoxes can be used. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



NOTE: The CTRIO Configuration IBox instruction does not require a permissive contact. The top line will be identified in ***BOLD italics***, and the instruction name and ID will be in ***BOLD characters***.

Rung 2: This CTRIO Velocity Mode IBox sets up Output #0 in CTRIO #1 to output 10,000 pulses at a Frequency of 1000 Hz. This example program requires that you load CTRVELO_IBox.cwb into your Hx-CTRIO(2) module.



CTRVELO Example (cont'd)

Rung 3: If the Velocity Mode parameters are OK, set the Direction Bit and Enable the output.



5

CTRIO Write File to ROM (CTRWFTFTR) (IB-1006)

 230

 240

 250-1

 260

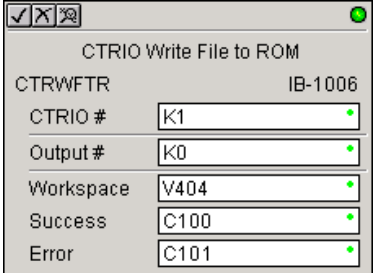
CTRIO Write File to ROM writes the runtime changes made to a loaded CTRIO Preset Table back to Flash ROM on a leading edge transition to this IBox. This IBox will take more than one PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.

The Workspace register is for internal use by this IBox instruction and **MUST NOT** be used anywhere else in your program.

| | |
|-----|------|
| DS5 | Used |
| HPP | N/A |

CTRWFTFTR Parameters

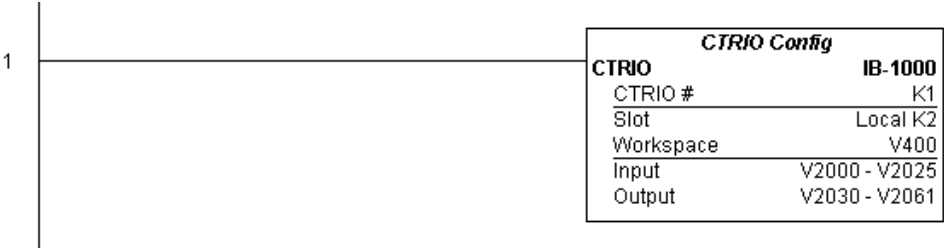
- CTRIO#: specifies a specific CTRIO module based on a user defined number (see CTRIO Config Ibox)
- Output#: specifies a CTRIO output to be used by the instruction
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the instruction has completed successfully
- Error: specifies a bit that will turn on if the instruction does not complete successfully



| Parameter | | DL205 Range |
|-----------|---------------|-------------------------------------|
| CTRIO# | K | K0-255 |
| Output# | K | K0-3 |
| Workspace | V | See DL205 V-memory map - Data Words |
| Success | X,Y,C,GX,GY,B | See DL205 V-memory map |
| Error | X,Y,C,GX,GY,B | See DL205 V-memory map |

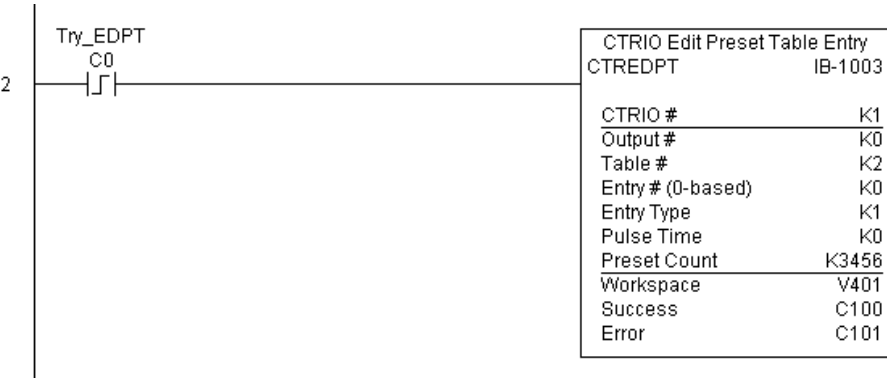
CTRWFTR Example

Rung 1: This sets up the CTRIO module in slot 2 of the local base. Each CTRIO module in the system will need a separate CTRIO Config IBox before any CTRxxxx IBoxes can be used. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



NOTE: The CTRIO Configuration IBox instruction does not require a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.

Rung 2: This CTRIO Edit Preset Table Entry IBox will change Entry 0 in Table #2 to be a RESET at Count 3456. This example program requires that you load CTRWFTR_IBox.cwb into your Hx-CTRIO(2) module.



(Example continued on next page)

CTRWFTTR Example (cont'd)

Rung 3: If the file is successfully edited, use a Write File To ROM IBox to save the edited table back to the CTRIO's ROM, thereby making the changes retentive.



Notes:

INDEX



A

- Accumulator/Stack Instructions, 5–53
 - Pointers, 5–57
- Analog Potentiometer, 3–19
- ASCII Conversion Table, G-2
- ASCII Instructions, 5–211
- Auto tuning error, 8–48
- Auto Tuning Procedure, 8–45
- Auxiliary Functions, A–2
 - Accessing, A–3
 - Direct*SOFT, A–3
 - Handheld Programmer, A–3

B

- Basic EMC Installation Guidelines, I-4
- Binary Numbering System, H–2
- Bit Map, 3–57
 - Control Relay, 3–59
 - Remote I/O, 3–66
 - Stage Control/Status, 3–63
 - Timer and Counter Status, 3–65
 - X Input/Y Output, 3–57
- Bit Operations, 5–123
- Boolean Instructions, 5–5, 5–10
- Bumpless Transfer, 8–13, 8–27, 8–75, 8–76

C

- Cascade Control, 8–64
 - Tuning, 8–66
- Clock and Calendar, 3–16
- Clock/Calendar Instructions, 5–175
- Comparative Boolean, 5–27
- Converge Jump instruction, 7–25
- Converge Stage instruction, 7–25
- Convergence Jump instruction, 7–20
- Convergence Stages, 7–19
- Counter Instructions, 5–46
- CPU Control Instructions, 5–177
- CPU Error Indicators, 9–10
 - PWR Indicator, 9–11
- CPU Features, 3–2
 - Mode Switch, 3–12
 - Program Storage Media, 3–9
 - Setting the CPU Network Address, 3–17
 - Setup, 3–6
 - Specifications, 3–4
 - Status Indicators, 3–12
- CPU Operation, 3–21
 - I/O Response Time, 3–27
 - Scan Time Considerations, 3–29

D

- Data Type Mismatch, H-7
- DL205 Aliases, 3-52
- DL205 PLC Memory, E-2
- Drum Instruction (DRUM), 6-12
 - chart representation, 6-3
 - counter assignments, 6-6
 - drum control techniques, 6-10
 - event drum (EDRUM), 6-14
 - handheld programmer mnemonics, 6-16
 - masked event drum (MDRMD)(MDRMW), 6-19, 6-21
 - Operation, 6-8
 - powerup state, 6-9
 - self-resetting, 6-11
 - step transitions, 6-11
 - Terminology, 6-2

E

- Environmental Specifications, 2-8
- Error Codes, 9-6, B-2
- Error Term Selection, 8-34
- European Union Directives, I-2
- Expanding DL205 I/O, 4-17
 - Ethernet Base Controller, 4-22
 - Ethernet Remote Master, 4-17
 - Serial Remote I/O Master/Slave, 4-26

F

- Feedforward Control, 8-69
- Freeze Bias, 8-11, 8-35

H

- Hardware Maintenance, 9-2
 - Communications, 9-13
 - Diagnostics, 9-3

- Hexadecimal Numbering System, H-3
- Hysteresis, 8-13, 8-37, 8-38, 8-39

I

- I/O Module Troubleshooting, 9-14
- I/O System Configurations, 4-2
- Immediate Instructions, 5-33
- Initial Stage (ISG), 7-24
- Installation, 2-10
 - Base Wiring, 2-13
 - Connecting DC I/O, 2-19
 - I/O Modules Position, 2-26
 - Transient Suppression, 2-21
- Instruction Execution Times, C-2
- Instruction Table, 5-2
- Instructions
 - stage, 7-23
 - stage programming, 7-2
- Intelligent Box (IBox) Table, 5-230
- Intelligent I/O Instructions, 5-191
- Interrupt Instructions, 5-187

J

- Jump instruction, 7-7

L

- Local Expansion I/O, 4-11
- Loop Mode, 8-28
- Loop Table Word Definitions, 8-20

M

- Machine Startup, 9-18
- Memory Map, 3-37, 3-53
 - DL230, 3-53
 - DL240, 3-54

DL250-1, 3-55

DL260, 3-56

Message Instructions, 5-197

Modbus RTU Instructions, 5-205

Mounting Guidelines, 2-5

Component Dimensions, 2-5

N

Network Connections, 4-32

Configuring Port 2, 4-32

Network Instructions, 5-193

Network Master Operation, 4-41

Network Modbus RTU Master Operation, 4-45

Network Slave Operation, 4-35

Modbus, 4-35

Noise Troubleshooting, 9-17

Non-volatile V-memory, E-3

Non-Sequence Protocol, 4-54

Configure the DL250-1 Port 2, 4-56

Not Jump, 7-24

Number Conversion Instructions, 5-130

O

Octal Numbering System, H-4

P

Parallel Processing, 7-19

PID

Analog Filter, 8-54

DirectSOFT Filter Intelligent Box Instruction, 8-56

Error Flags, 8-18

Example Program, 8-71

Loop Modes, 8-28, 8-52, 8-53, 8-65

Parameters, 8-33

Setup Alarms, 8-36

Special Features, 8-52

PID Alarms, 8-36

Calculation Overflow/Underflow Error, 8-39

Hysteresis, 8-39

Mode/Alarm Bit Description, 8-23

Monitor Limit, 8-36

Programming Error, 8-39

PV Deviation, 8-37

Rate-of-Change, 8-38

PID Loop

Alarms, 8-13

Auto Tuning, 8-41

Configure, 8-26

Manual Tuning, 8-41, 8-42, 8-44, 8-47, 8-54

Mode, 8-28

Operating Modes, 8-14

Operation, 8-9

Program Setup, 8-71

Reverse Acting, 8-12, 8-14, 8-27, 8-41

Setup, 8-18

Terminology, 8-76

Troubleshooting Tips, 8-74

PID Mode 2 Word Description, 8-22

PID Mode Setting 1 Description, 8-21

PLC Numbering Systems, 3-35

Position Algorithm, 8-9, 8-15, 8-77

Position Form, 8-9

Power Budget, 4-7

Product Weight Table, F-2

Products and Data Types, H-9

Program Control Instructions, 5-179

Q

Quick Start, 1-10

R

- Ramp/Soak Generator, 8–57
 - Controls, 8–60
 - Direct*SOFT Ramp/Soak Example, 8–62
 - Profile Monitoring, 8–61
 - Ramp/Soak Flag Bit Description, 8–23
 - Ramp/Soak Table Location, 8–24
 - Relay Ladder, 8–62
 - Table, 8–58
 - Table Flags, 8–60
 - Test Profile with PID View, 8–63
 - Testing, 8–61
- Rate-of-Change, 8–13, 8–14, 8–38
- Real Numbering System, H–5
- Reset Windup, 8–10, 8–35, 8–77

S

- Safety Guidelines, 2–2
 - Emergency Stops, 2–3
- Shift Register Instruction, 5–52
- Signed vs. Unsigned Integers, H–8
- Special Relays, D–2
- Square Root, 8–14
- Stage instructions, 7–23
- Stage Programming, 7–2, 7–15
 - convergence, 7–19
 - emergency stop, 7–14
 - Example, 7–10
 - four steps to stage programming, 7–9
 - introduction, 7–2
 - jump instruction, 7–7
 - mutually exclusive transitions, 7–14
 - parallel processing concepts, 7–19
 - program organization, 7–15
 - questions and answers, 7–29
 - stage instruction characteristics, 7–6
 - state transition diagrams, 7–3
 - timer inside stage, 7–13

- State Diagram, 7–11
- Step Transitions, 6–4
- System Components, 1–4
 - Direct*LOGIC™ Part Numbering System, 1–8
- System V-memory, 3–41
 - DL230, 3–41
 - DL240, 3–43
 - DL250–1, 3–46
 - DL260, 3–49

T

- Table Instructions, 5–144
- Technical Support, 1–2
- Time-Proportioning Control, 8–67
 - On/Off Control Program, 8–68
- Timer Instructions, 5–41
- Transcendental Functions, 5–121
- Transfer Mode, 8–27

U

- Using a Password, 3–18
- Using Battery Backup, 3–14

V

- Velocity Algorithm, 8–9, 8–15, 8–77
 - Velocity Form, 8–12