

**SYSMAC CVM1D**  
**Duplex System**  
**Programmable Controllers**


**OPERATION MANUAL**


**OMRON**


## Notice:

OMRON products are manufactured for use according to proper procedures by a qualified operator and only for the purposes described in this manual.

The following conventions are used to indicate and classify precautions in this manual. Always heed the information provided with them. Failure to heed precautions can result in injury to people or damage to property.

 **DANGER** Indicates an imminently hazardous situation which, if not avoided, will result in death or serious injury.

 **WARNING** Indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury.

 **Caution** Indicates a potentially hazardous situation which, if not avoided, may result in minor or moderate injury, or property damage.

## OMRON Product References

All OMRON products are capitalized in this manual. The word “Unit” is also capitalized when it refers to an OMRON product, regardless of whether or not it appears in the proper name of the product.

The abbreviation “Ch,” which appears in some displays and on some OMRON products, often means “word” and is abbreviated “Wd” in documentation in this sense.

The abbreviation “PC” means Programmable Controller and is not used as an abbreviation for anything else.

## Visual Aids

The following headings appear in the left column of the manual to help you locate different types of information.

**Note** Indicates information of particular interest for efficient and convenient operation of the product.

**1, 2, 3...** 1. Indicates lists of one sort or another, such as procedures, checklists, etc.

## © OMRON, 1999

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, mechanical, electronic, photocopying, recording, or otherwise, without the prior written permission of OMRON.

No patent liability is assumed with respect to the use of the information contained herein. Moreover, because OMRON is constantly striving to improve its high-quality products, the information contained in this manual is subject to change without notice. Every precaution has been taken in the preparation of this manual. Nevertheless, OMRON assumes no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained in this publication.

# TABLE OF CONTENTS

<b>PRECAUTIONS</b> .....	<b>xi</b>
1 Intended Audience .....	xii
2 General Precautions .....	xii
3 Safety Precautions .....	xii
4 Operating Environment Precautions .....	xiii
5 Application Precautions .....	xiii
6 Conformance to EC Directives .....	xv
<b>SECTION 1</b>	
<b>Introduction</b> .....	<b>1</b>
1-1 Overview .....	2
1-2 Relay Circuits: The Roots of PC Logic .....	2
1-3 PC Terminology .....	3
1-4 OMRON Product Terminology .....	3
1-5 Overview of PC Operation .....	4
1-6 PC Operating Modes .....	5
1-7 Programming Devices .....	7
1-8 CVM1D Program Memory .....	9
1-9 Related Manuals .....	9
<b>SECTION 2</b>	
<b>Hardware Considerations</b> .....	<b>11</b>
2-1 CPU Unit Switch Settings .....	12
2-2 Duplex Unit Switch Settings .....	14
2-3 EM Units .....	17
2-4 Memory Cards .....	18
2-5 Setting Rack Numbers .....	22
2-6 I/O Control Unit and I/O Interface Unit Displays .....	22
<b>SECTION 3</b>	
<b>Memory Areas</b> .....	<b>25</b>
3-1 Introduction .....	27
3-2 Data Area Structure .....	27
3-3 CIO (Core I/O) Area .....	31
3-4 TR (Temporary Relay) Area .....	39
3-5 CPU Bus Link Area .....	39
3-6 Auxiliary Area .....	41
3-7 Timer Area .....	59
3-8 Counter Area .....	60
3-9 DM and EM Areas .....	60
3-10 Index and Data Registers (IR and DR) .....	62
<b>SECTION 4</b>	
<b>Writing Programs</b> .....	<b>65</b>
4-1 Basic Procedure .....	66
4-2 Instruction Terminology .....	66
4-3 Basic Ladder Diagrams .....	67
4-4 Mnemonic Code .....	72
4-5 Branching Instruction Lines .....	79
4-6 Jumps .....	83
4-7 Controlling Bit Status .....	85
4-8 Intermediate Instructions .....	87
4-9 Work Bits (Internal Relays) .....	87
4-10 Programming Precautions .....	89
4-11 Program Execution .....	90
4-12 Data Formats .....	90

# TABLE OF CONTENTS

<b>SECTION 5</b>	
<b>Instruction Set</b> .....	<b>95</b>
5-1 Notation .....	100
5-2 Instruction Format .....	100
5-3 Data Areas, Definers, and Flags .....	100
5-4 Differentiated and Immediate Refresh Instructions .....	103
5-5 Coding Right-hand Instructions .....	105
5-6 Ladder Diagram Instructions .....	107
5-7 Bit Control Instructions .....	112
5-8 INTERLOCK and INTERLOCK CLEAR: IL(002) and ILC(003) .....	120
5-9 JUMP and JUMP END: JMP(004) and JME(005) .....	122
5-10 CONDITIONAL JUMP: CJP(221)/CJPN(222) .....	125
5-11 END: END(001) .....	126
5-12 NO OPERATION: NOP(000) .....	126
5-13 Timer and Counter Instructions .....	127
5-14 Shift Instructions .....	147
5-15 Data Movement Instructions .....	175
5-16 Comparison Instructions .....	193
5-17 Conversion Instructions .....	207
5-18 BCD Calculation Instructions .....	237
5-19 Binary Calculation Instructions .....	249
5-20 Symbol Math Instructions .....	259
5-21 Floating-point Math Instructions .....	280
5-22 Increment/Decrement Instructions .....	301
5-23 Special Math Instructions .....	306
5-24 PID and Related Instructions .....	317
5-25 Logic Instructions .....	330
5-26 Time Instructions .....	337
5-27 Special Instructions .....	342
5-28 Flag/Register Instructions .....	354
5-29 STEP DEFINE and STEP START: STEP(008)/SNXT(009) .....	356
5-30 Subroutines .....	365
5-31 Stack Instructions .....	370
5-32 Data Tracing .....	374
5-33 Special I/O Instructions .....	377
5-34 Network Instructions .....	385
5-35 Block Programming Instructions .....	400
<b>SECTION 6</b>	
<b>Program Execution Timing</b> .....	<b>413</b>
6-1 PC Operation .....	414
6-2 Cycle Time .....	418
6-3 Calculating Cycle Time .....	423
6-4 Instruction Execution Times .....	425
6-5 I/O Response Time .....	437
<b>SECTION 7</b>	
<b>PC Setup</b> .....	<b>441</b>
7-1 PC Setup Overview .....	442
7-2 PC Setup Details .....	443
7-3 PC Setup Default Settings .....	446
<b>Index</b> .....	<b>449</b>
<b>Revision History</b> .....	<b>459</b>

## About this Manual:

This manual describes the operation and programming of the CVM1D Duplex System Programmable Controllers and includes the sections described below.

Please read this and all related manuals carefully and be sure you understand the information provide before attempting to operate or program a CVM1D Duplex System Programmable Controller. The main manuals required for operation are listed below. A list of other related manuals is provided at the end of *Section 1 Introduction*.

Manual	Contents	Cat. No.
CVM1D Duplex System PC Installation Guide	Describes the system configuration, specifications, installation, and wiring.	W350
CVM1D Duplex System PC Operation Manual	Describes ladder-diagram programming and programming instructions.	W351
CVM1-PRS21-EV1 Programming Console Operation Manual	Describes Programming Console operations.	W222
CVM1/CV-series PCs Operation Manual: Host Link System	Describes communications methods with host computers.	W205

**Section 1** gives a brief overview of the history of Programmable Controllers and explains terms commonly used in ladder-diagram programming. It provides an overview of the process of programming and operating a PC and explains basic terminology used with OMRON PCs. It also describes the Programming Devices that can be used and provides a list of related FA products and manuals.

**Section 2** provides information on hardware aspects of CVM1D PCs that are relevant to programming and software operation. These include switches on the CPU Unit and basic PC configuration. This information is covered in more detail in the CVM1D Installation Guide.


**Section 3** describes the way in which PC memory is broken into various areas used for different purposes. The contents of each area and addressing conventions, including the use of indirect addressing and addressing registers, are also described.

**Section 4** explains the basic steps and concepts involved in writing a basic ladder diagram program. It introduces the instructions that are used to build the basic structure of the ladder diagram and control its execution, along with a few other instructions of special interest in programming.

**Section 5** explains each instruction in the CVM1D PC instruction set and provides the ladder diagram symbols, data areas, and flags used with each.

**Section 6** explains the execution cycle of the PC and shows how to calculate the cycle time and I/O response times. I/O response times in Link Systems are described in the individual System Manuals.

**Section 7** lists the parameters in the PC Setup, provide examples of normal application, and provide the default values. The use of each parameter in the PC Setup is described where relevant in this manual and in other CVM1D manuals.

 <b>WARNING</b> Failure to read and understand the information provided in this manual may result in personal injury or death, damage to the product, or product failure. Please read each section in its entirety and be sure you understand the information provided in the section and related sections before attempting any of the procedures or operations given.
--

# PRECAUTIONS

This section provides general precautions for using the Programmable Controller (PC) and related devices.

**The information contained in this section is important for the safe and reliable application of the Programmable Controller. You must read this section and understand the information contained before attempting to set up or operate a PC system.**

1 Intended Audience .....	xii
2 General Precautions .....	xii
3 Safety Precautions .....	xii
4 Operating Environment Precautions .....	xiii
5 Application Precautions .....	xiii
6 Conformance to EC Directives .....	xv

## 1 Intended Audience

This manual is intended for the following personnel, who must also have knowledge of electrical systems (an electrical engineer or the equivalent).

- Personnel in charge of installing FA systems.
- Personnel in charge of designing FA systems.
- Personnel in charge of managing FA systems and facilities.


## 2 General Precautions

The user must operate the product according to the performance specifications described in the operation manuals.






Before using the product under conditions which are not described in the manual or applying the product to nuclear control systems, railroad systems, aviation systems, vehicles, combustion systems, medical equipment, amusement machines, safety equipment, and other systems, machines, and equipment that may have a serious influence on lives and property if used improperly, consult your OMRON representative.

Make sure that the ratings and performance characteristics of the product are sufficient for the systems, machines, and equipment, and be sure to provide the systems, machines, and equipment with double safety mechanisms.


This manual provides information for programming and operating the Unit. Be sure to read this manual before attempting to use the Unit and keep this manual close at hand for reference during operation.


-  **WARNING** It is extremely important that a PC and all PC Units be used for the specified purpose and under the specified conditions, especially in applications that can directly or indirectly affect human life. You must consult with your OMRON representative before applying a PC System to the above-mentioned applications.


## 3 Safety Precautions

-  **WARNING** Do not attempt to take any Unit apart while the power is being supplied. Doing so may result in electric shock.
-  **WARNING** Do not touch any of the terminals while the power is being supplied. Doing so may result in electric shock.
-  **WARNING** Do not attempt to disassemble, repair, or modify any Units. Any attempt to do so may result in malfunction, fire, or electric shock.
-  **WARNING** There is a lithium battery built into the SRAM Memory Cards. Do not short the positive and negative terminals of the battery, charge the battery, attempt to take it apart, subject it to pressures that would deform it, incinerate it, or otherwise mistreat it. Doing any of these could cause the battery to erupt, ignite, or leak.
-  **WARNING** The CPU Unit refreshes I/O even when the program is stopped (i.e., even in PROGRAM mode). Confirm safety thoroughly in advance before changing the status of any part of memory allocated to I/O Units, Special I/O Units, or CPU Bus Units. Any changes to the data allocated to any Unit may result in unexpected operation of the loads connected to the Unit. Any of the following operation may result in changes to memory status.
- Transferring I/O memory data to the CPU Unit from a Programming Device.


- Changing present values in memory from a Programming Device.
- Force-setting/-resetting bits from a Programming Device.
- Transferring I/O memory from a host computer or from another PC on a network.

 **Caution** Execute online edit only after confirming that no adverse effects will be caused by extending the cycle time. Otherwise, the input signals may not be readable.


 **Caution** Confirm safety at the destination node before transferring a program to another node or changing the I/O memory area. Doing either of these without confirming safety may result in injury.

 **Caution** Tighten the screws on the terminal block of the AC Power Supply Unit to the torque specified in the operation manual. The loose screws may result in burning or malfunction.


## 4 Operating Environment Precautions

 **Caution** Do not operate the control system in the following places:

- Locations subject to direct sunlight.
- Locations subject to temperatures or humidity outside the range specified in the specifications.
- Locations subject to condensation as the result of severe changes in temperature.
- Locations subject to corrosive or flammable gases.
- Locations subject to dust (especially iron dust) or salts.
- Locations subject to exposure to water, oil, or chemicals.
- Locations subject to shock or vibration.


 **Caution** Take appropriate and sufficient countermeasures when installing systems in the following locations:

- Locations subject to static electricity or other forms of noise.
- Locations subject to strong electromagnetic fields.
- Locations subject to possible exposure to radioactivity.
- Locations close to power supplies.

 **Caution** The operating environment of the PC System can have a large effect on the longevity and reliability of the system. Improper operating environments can lead to malfunction, failure, and other unforeseeable problems with the PC System. Be sure that the operating environment is within the specified conditions at installation and remains within the specified conditions during the life of the system.

## 5 Application Precautions

Observe the following precautions when using the PC System.

 **WARNING** Always heed these precautions. Failure to abide by the following precautions could lead to serious or possibly fatal injury.

- Always ground the system to 100  $\Omega$  or less when installing the Units to protect against electric shock.



- Always turn OFF the power supply to the PC before attempting any of the following. Not turning OFF the power supply may result in malfunction or electric shock.
  - Mounting or dismounting Power Supply Unit, I/O Units, CPU Units, Memory Cassettes, or any other Units.
  - Assembling the Units.
  - Setting DIP switches or rotary switches.
  - Connecting or wiring the cables.
  - Connecting or disconnecting the connectors.

**Caution**

Failure to abide by the following precautions could lead to faulty operation of the PC or the system, or could damage the PC or PC Units. Always heed these precautions.

- Fail-safe measures must be taken by the customer to ensure safety in the event of incorrect, missing, or abnormal signals caused by broken signal lines, momentary power interruptions, or other causes.
- Interlock circuits, limit circuits, and similar safety measures in external circuits (i.e., not in the Programmable Controller) must be provided by the customer.
- Always use the power supply voltage specified in the operation manuals. An incorrect voltage may result in malfunction or burning.
- Take appropriate measures to ensure that the specified power with the rated voltage and frequency is supplied. Be particularly careful in places where the power supply is unstable. An incorrect power supply may result in malfunction.
- Install external breakers and take other safety measures against short-circuiting in external wiring. Insufficient safety measures against short-circuiting may result in burning.
- Do not mount a CVM1D Power Supply Unit in a CVM1 or CV-series PC. Doing so may result in burning or malfunction.
- Do not mount a CVM1 or CV-series Power Supply Unit on a Duplex CVM1D PC. Doing so may result in unexpected stoppage of the PC.
- Do not apply voltages to the Input Units in excess of the rated input voltage. Excess voltages may result in burning.
- Do not apply voltages or connect loads to the Output Units in excess of the maximum switching capacity. Excess voltage or loads may result in burning.
- Disconnect the functional ground terminal when performing withstand voltage tests. Not disconnecting the functional ground terminal may result in burning.
- Install the Unit properly as specified in the operation manual. Improper installation of the Unit may result in malfunction.
- Be sure that all the mounting screws, terminal screws, and cable connector screws are tightened to the torque specified in the relevant manuals. Incorrect tightening torque may result in malfunction.
- Leave the label attached to the Unit when wiring. Removing the label may result in malfunction.
- Remove the label after the completion of wiring to ensure proper heat dissipation. Leaving the label attached may result in malfunction.
- Use crimp terminals for wiring. Do not connect bare stranded wires directly to terminals. Connection of bare stranded wires may result in burning.
- Double-check all the wiring before turning on the power supply. Incorrect wiring may result in burning.
- Mount the Unit only after checking the terminal block completely.
- Be sure that the terminal blocks, Memory Units, expansion cables, and other items with locking devices are properly locked into place. Improper locking may result in malfunction.

- Check the user program for proper execution before actually running it on the Unit. Not checking the program may result in an unexpected operation.
- Confirm that no adverse effect will occur in the system before attempting any of the following. Not doing so may result in an unexpected operation.
  - Changing the operating mode of the PC.
  - Force-setting/force-resetting any bit in memory.
  - Changing the present value of any word or any set value in memory.
- Do not pull on the cables or bend the cables beyond their natural limit. Doing either of these may break the cables.
- Do not place objects on top of the cables. Doing so may break the cables.
- When replacing parts, be sure to confirm that the rating of a new part is correct. Not doing so may result in malfunction or burning.
- Before touching the Unit, be sure to first touch a grounded metallic object in order to discharge any static built-up. Not doing so may result in malfunction or damage.

## 6 Conformance to EC Directives

The CVM1D PCs conform to EMC and Low Voltage Directives as follows:

### **EMC Directives**

OMRON devices that comply with EC Directives also conform to the related EMC standards so that they can be more easily built into other devices or the overall machine. The actual products have been checked for conformity to EMC standards (see the following note). Whether the products conform to the standards in the system used by the customer, however, must be confirmed by the customer.

EMC-related performance of the OMRON devices that comply with EC Directives will vary depending on the configuration, wiring, and other conditions of the equipment or control panel on which the OMRON devices are installed. The customer must, therefore, perform the final check to confirm that devices and the overall machine conform to EMC standards.

**Note** Applicable EMC (Electromagnetic Compatibility) standards are as follows:

EMS (Electromagnetic Susceptibility): EN61131-2  
 EMI (Electromagnetic Interference): EN50081-2  
 (Radiated emission: 10-m regulations)

### **Low Voltage Directive**

Always ensure that devices operating at voltages of 50 to 1,000 VAC and 75 to 1,500 VDC meet the required safety standards for the PC (EN61131-2).

The CVM1D, CVM1, and CV-series PCs that comply with EC Directives must be installed as follows:

- 1, 2, 3...** 1. The CVM1D, CVM1, and CV-series PCs are designed for installation inside control panels. All PCs must be securely grounded within control panels.
2. Use reinforced insulation or double insulation for the DC power supplies used for the communications power supply and I/O power supplies.
3. The CVM1D, CVM1, and CV-series PCs that comply with EC Directives also conform to the Common Emission Standard (EN50081-2). Radiated emission characteristics (10-m regulations) may vary depending on the configuration of the control panel used, other devices connected to the control panel, wiring, and other conditions. You must therefore confirm that the overall machine or equipment complies with EC Directives.

Precautions Regarding Relay Output Switching:

Noise generated by the switching of relay outputs may exceed the standards of the Common Emission Standard (EN50081-2). In this case, take the necessary countermeasures to reduce the emissions, such as connecting a surge suppressor or shielding the PC. Some methods of reducing emissions are shown below.

Countermeasures are required when the load switching frequency of the overall PC exceeds 5 times/minute. Refer to *3-8 Compliance with EC Directives* in the *CVM1D Installation Guide* for examples of circuits that can be used to reduce noise generated by switching.

# SECTION 1

## Introduction

This section gives a brief overview of the history of Programmable Controllers and explains terms commonly used in ladder-diagram programming. It provides an overview of the process of programming and operating a PC and explains basic terminology used with OMRON PCs. It also describes the Programming Devices that can be used and provides a list of related FA products and manuals.

1-1	Overview .....	2
1-2	Relay Circuits: The Roots of PC Logic .....	2
1-3	PC Terminology .....	3
1-4	OMRON Product Terminology .....	3
1-5	Overview of PC Operation .....	4
1-6	PC Operating Modes .....	5
1-6-1	Operating Modes .....	5
1-6-2	RUN Output .....	6
1-7	Programming Devices .....	7
1-7-1	Compatible Programming Devices .....	7
1-7-2	Connecting Programming Devices .....	8
1-7-3	Transferring Programs .....	8
1-7-4	Recommended Programming Device Operations .....	8
1-8	CVM1D Program Memory .....	9
1-9	Related Manuals .....	9

## 1-1 Overview

A PC (Programmable Controller) is basically a CPU (Central Processing Unit) containing a program and connected to input and output (I/O) devices. The program controls the PC so that when an input signal from an input device turns ON or OFF, the appropriate response is made. The response normally involves turning ON or OFF an output signal to some sort of output device. The input devices could be photoelectric sensors, pushbuttons on control panels, limit switches, or any other device that can produce a signal that can be input into the PC. The output devices could be solenoids, indicator lamps, relays turning on motors, or any other devices that can be activated by signals output from the PC.

For example, a sensor detecting a passing product turns ON an input to the PC. The PC responds by turning ON an output that activates a pusher that pushes the product onto another conveyor for further processing. Another sensor, positioned higher than the first, turns ON a different input to indicate that the product is too tall. The PC responds by turning on another pusher positioned before the pusher mentioned above to push the too-tall product into a rejection box.

Although this example involves only two inputs and two outputs, it is typical of the type of control operation that PCs can achieve. Actually even this example is much more complex than it may at first appear because of the timing that would be required, i.e., "How does the PC know when to activate each pusher?" Much more complicated operations are also possible.

To achieve proper control, CVM1D PCs use a form of PC logic called ladder-diagram programming. A single ladder-diagram program is used, as in C-series PCs. CVM1D PCs cannot use SFC programs.

## 1-2 Relay Circuits: The Roots of PC Logic

PCs historically originate in relay-based control systems. And although the integrated circuits and internal logic of the PC have taken the place of the discrete relays, timers, counters, and other such devices, actual PC operation proceeds as if those discrete devices were still in place. PC control, however, also provides computer capabilities and accuracy to achieve a great deal more flexibility and reliability than is possible with relays.

The symbols and other control concepts used to describe PC operation also come from relay-based control and form the basis of the ladder-diagram programming method. Most of the terms used to describe these symbols and concepts, however, have come in from computer terminology.

### Relay vs. PC Terminology

The terminology used throughout this manual is somewhat different from relay terminology, but the concepts are the same.

The following table shows the relationship between relay terms and the PC terms used for OMRON PCs.

Relay term	PC equivalent
contact	input or condition
coil	output or work bit
NO relay	normally open condition
NC relay	normally closed condition

Actually there is not a total equivalence between these terms. The term condition is only used to describe ladder diagram programs in general and is specifically equivalent to one of a certain set of basic instructions. The terms input and output are not used in programming per se, except in reference to I/O bits that are assigned to input and output signals coming into and leaving the PC. Normally open conditions and normally closed conditions are explained in *4-3 Basic Ladder Diagrams*.

## 1-3 PC Terminology

Although also provided in the *Glossary* at the back of this manual, the following terms are crucial to understanding PC operation and are thus introduced here.

### PC

Because CVM1D PCs are Rack PCs, there is no single product that is a CVM1D PC. That is why we talk about the configuration of the PC, because a PC is a configuration of smaller Units.

To have a functional PC, you would need to have a CPU Rack with at least one Unit mounted to it that provides I/O points. When we refer to the PC, however, we are generally talking about the CPU and all of the Units directly controlled by it through the program. This does not include the I/O devices connected to PC inputs and outputs. The term PC is also used to refer to the controlling element of the PC, i.e., the CPU.

If you are not familiar with the terms used above to describe a PC, refer to *Section 2 Hardware Considerations* for explanations.

### Inputs and Outputs

A device connected to the PC that sends a signal to the PC is called an input device; the signal it sends is called an input signal. A signal enters the PC through terminals or through pins on a connector on a Unit. The place where a signal enters the PC is called an input point. This input point is allocated a location in memory that reflects its status, i.e., either ON or OFF. This memory location is called an input bit. The CPU, in its normal processing cycle, monitors the status of all input points and turns ON or OFF corresponding input bits accordingly.

There are also output bits in memory that are allocated to output points on Units through which output signals are sent to output devices, i.e., an output bit is turned ON to send a signal to an output device through an output point. The CPU periodically turns output points ON or OFF according to the status of the output bits.

These terms are used when describing different aspects of PC operation. When programming, one is concerned with what information is held in memory, and so I/O bits are referred to. When talking about the Units that connect the PC to the controlled system and the places on these Units where signals enter and leave the PC, I/O points are referred to. When wiring these I/O points, the physical counterparts of the I/O points, either terminals or connector pins, are referred to. When talking about the signals that enter or leave the PC, one refers to input signals and output signals, or sometimes just inputs and outputs. It all depends on what aspect of PC operation is being talked about.

### Controlled System and Control System

The Control System includes the PC and all I/O devices it uses to control an external system. A sensor that provides information to achieve control is an input device that is clearly part of the Control System. The controlled system is the external system that is being controlled by the PC program through these I/O devices. I/O devices can sometimes be considered part of the controlled system, e.g., a motor used to drive a conveyor belt.

## 1-4 OMRON Product Terminology

OMRON products are divided into several functional groups that have generic names. *Appendix A Standard Models* lists products according to these groups. The term **Unit** is used to refer to all of the OMRON PC products. Although a Unit is any one of the building blocks that goes together to form a CV-series PC, its meaning is generally, but not always, limited in context to refer to the Units that are mounted to a Rack. Most, but not all, of these products have names that end with the word Unit.

The largest group of OMRON products is the **I/O Units**. These include all of the Rack-mounting Units that provide non-dedicated input or output points for general use. I/O Units come with a variety of point connections and specifications.

**Special I/O Units** are dedicated Units that are designed to meet specific needs. These include Position Control Units, High-speed Counter Units, and Analog I/O Units. This group also includes some programmable Units, such as the ASCII Unit, which is programmed in BASIC.

**CPU Bus Units** connect to the CPU bus and must be mounted on either the CPU Rack or an Expansion CPU Rack. These include the SYSMAC NET Link Unit, SYSMAC LINK Unit, SYSMAC BUS/2 Remote I/O Master Unit, and BASIC Unit.

**Link Units** are used to create communications links between PCs or between PCs and other devices. Link Units include SYSMAC NET Link Unit, SYSMAC LINK Unit, and, sometimes, SYSMAC BUS/2 Remote I/O Master Unit.

Other product groups include **Programming Devices**, **Programming Devices**, and **DIN Track Products**.

## 1-5 Overview of PC Operation

The following are the basic steps involved in programming and operating a CV-series PC. Assuming you have already purchased one or more of these PCs, you must have a reasonable idea of the required information for steps one and two, which are discussed briefly below. The relevant sections of this manual that provide more information are listed with relevant steps.

- 1, 2, 3...
  1. Determine what the controlled system must do, in what order, and at what times.
  2. Determine what Racks and what Units will be required. Refer to the *CVM1D PC Installation Guide*. If a Link System is required, refer to the appropriate *System Manual*.
  3. On paper, assign all input and output devices to I/O points on Units and determine which I/O bits will be allocated to each. If the PC includes Special I/O Units, CPU Bus Units, or Link Systems, refer to the individual *Operation Manuals* or *System Manuals* for details on I/O bit allocation. (*Section 3 Memory Areas*)
  4. Using relay ladder symbols, write a program that represents the sequence of required operations within each process and their inter-relationships. Be sure to also program appropriate responses for all possible emergency situations. (*Section 4 Writing Programs*, *Section 5 Instruction Set*, *Section 6 Program Execution Timing*)
  5. Write the program in the SSS offline, and then switch to online operation and transfer the program to Program Memory in the CPU. The program can also be written or altered online. Refer to the *SSS Operation Manual* for details.
  6. Generate the I/O table with I/O Units installed. The I/O table can be generated either online from the SSS, or edited offline and then transferred. Always turn the PC off and on after transferring the I/O table. The PC will not run until the I/O table has been registered. Refer to the *SSS Operation Manual* for details.
  7. The PC Setup controls a variety of basic options in PC operation (such as the PC mode at start-up). The operating parameters in the PC Setup can be left in their default settings or changed with the SSS as required. Refer to *Section 7 PC Setup* for details.
  8. Debug the program, first to eliminate any syntax errors, and then to find execution errors. Refer to the SSS operation manuals for details on debugging operations. (Refer to *Section 6 Troubleshooting* in the *CVM1D PC Installation Guide*.)

9. Wire the PC to the controlled system. This step can actually be started as soon as step 3 has been completed. Refer to the *CV-series PCs Installation Guide* and to other *Operation Manuals* and *System Manuals* for details on individual Units.
10. Test the program in an actual control situation and carry out fine tuning as required. Refer to the SSS operation manuals for details on debugging operations. (Refer to *Section 6 Troubleshooting* in the *CVM1D PC Installation Guide*.)
11. Record two copies of the finished program on masters and store them safely in different locations. Refer to the SSS operation manuals for details.

- Note**
1. The date and time are not set when the CPU is shipped. Set the date and time by the procedure described in the *SSS Operation Manuals*.
  2. There is an error log in the PC. This log can be cleared by turning ON the Error Log Reset Bit (A00014).

<b>Control System Design</b>	Designing the Control System is the first step in automating any process. A PC can be programmed and operated only after the overall Control System is fully understood. Designing the Control System requires, first of all, a thorough understanding of the system that is to be controlled. The first step in designing a Control System is thus determining the requirements of the controlled system.
<b>Input/Output Requirements</b>	The first thing that must be assessed is the number of input and output points that the controlled system will require. This is done by identifying each device that is to send an input signal to the PC or which is to receive an output signal from the PC. Keep in mind that the number of I/O points available depends on the configuration of the PC.
<b>Sequence, Timing, and Relationships</b>	Next, determine the sequence in which control operations are to occur and the relative timing of the operations. Identify the physical relationships between the I/O devices as well as the kinds of responses that should occur between them. For instance, a photoelectric switch might be functionally tied to a motor by way of a counter within the PC. When the PC receives an input from a start switch, it could start the motor. The PC could then stop the motor when the counter has received a specified number of input signals from the photoelectric switch. Each of the related tasks must be similarly determined, from the beginning of the control operation to the end.
<b>Unit Requirements</b>	The actual Units that will be mounted or connected to PC Racks must be determined according to the requirements of the I/O devices. Actual hardware specifications, such as voltage and current levels, as well as functional considerations, such as those that require Special I/O Units, CPU Bus Units, or Link Systems will need to be considered. In many cases, Special I/O Units, CPU Bus Units or Link Systems can greatly reduce the programming burden. Details on these Units and Link Systems are available in appropriate <i>Operation Manuals</i> and <i>System Manuals</i> .  Once the entire Control System has been designed, the task of programming, debugging, and operation as described in the remaining sections of this manual can begin.

## 1-6 PC Operating Modes


### 1-6-1 Operating Modes

CVM1D PCs have four operation modes: PROGRAM, DEBUG, MONITOR, and RUN. The Unit will automatically enter the mode specified in the PC Setup (default setting: PROGRAM mode). Refer to *Section 7 PC Setup* for details. The PC mode can be changed from a Programming Device. The function of each mode is described briefly below.



**PROGRAM Mode**

PROGRAM mode is used when making basic changes to the PC program or settings, such as transferring, writing, changing, or checking the program, generating or changing the I/O table, or changing the PC Setup. The program cannot be executed in PROGRAM mode. Output points at Output Units will remain OFF, even when the corresponding output bit is ON.

 **WARNING** The CPU Unit will continue refreshing I/O even though PC operation is stopped in PROGRAM mode. If one of the following methods has been used to change the status of output bits allocated to Output Units or bits allocated to Special I/O Units or CPU Bus Units, be sure that it is safe to operate the PC before switching to MONITOR mode or RUN mode. Loads connected to the Output Unit, Special I/O Unit, or CPU Bus Unit may operate unexpectedly if the status of the corresponding output bit has been changed.

- Transferring I/O memory to the CPU Unit from a Programming Device (SSS)
- Changing PVs from a Programming Device
- Force-setting or force-resetting bits from a Programming Device
- Transferring I/O memory from another PC or host computer in the network

**DEBUG Mode**

DEBUG mode cannot be used in CVM1D mode.

DEBUG mode is used to check program execution and I/O operation after syntax errors in the program have been corrected. With SFC programs, step execution or section operation can be used to check for errors from a Programming Device using the DEBUG operation.

**MONITOR Mode**

MONITOR mode is used when monitoring program execution, such as making a trial run of a program. The program is executed just as it is in RUN mode, but bit status, timer and counter SV/PV, and the data content of most words can be changed online. PC operation in MONITOR mode is significantly slower than it is in RUN mode. Output points at Output Units will be turned ON when the corresponding output bit is ON.

**RUN Mode**

RUN mode is used when operating the PC in normal control conditions. Bit status cannot be Force Set or Reset, and SVs, PVs, and the data cannot be changed online. Output points at Output Units will be turned ON when the corresponding output bit is ON.

**1-6-2 RUN Output**

The following table shows the status of the RUN output on the CPU Rack's Power Supply Unit for the four operating modes.

Operating mode	Run output status	Start input status
(Waiting)	OFF	OFF
PROGRAM	OFF	ON
DEBUG	OFF	
MONITOR	ON	
RUN	ON	

The RUN output will be turned OFF when a fatal operating error occurs.

# 1-7 Programming Devices

## 1-7-1 Compatible Programming Devices

There are three Programming Devices that can be used with CVM1D PCs: The SYSMAC Support Software (SSS), the Programming Console and the Data Access Console.

Name	Function
SYSMAC Support Software (C500-ZL3AT1-E)	The SYSMAC Support Software (SSS) is the main Programming Device used to program and monitor CVM1D PCs. You must have the SSS to program and operate these PCs.
Programming Console (CVM1-PRS21-EV1)	The Programming Console can be used for onsite monitoring and programming of PC, but does not support advanced programming/debugging operations.  Use a version 2.0 or higher Programming Console with CVM1D PCs. Earlier versions cannot perform online I/O Unit replacement.
Data Access Console (CV500-DAC11)	The Data Access Console can be used for onsite monitoring and and setting adjustments.

- Note**
1. If the SYSMAC-CPT Support Software is used, AR Area words unique to the CVM1D PCs cannot be programmed. Furthermore, programs containing these AR Area words cannot be transferred.
  2. Errors that have been added to the CVM1D (the duplex bus error, duplex verification error, and duplex power supply error) will not be displayed on Programming Consoles other than the CVM1-PRS21-EV1 version 2.0. These errors will not be displayed in the error history, either.

### Connecting Cables

The following table shows which cables can be used when connecting a host computer or Programming Console to the Duplex Unit's peripheral connector. Make the required connecting cable when connecting to the host link connector.

Programming Device	Cable model number	Cable length
Personal Computer (SSS)	CV500-CIF01	6 m
Programming Console	CV500-CN224	2 m
	CV500-CN424	4 m
	CV500-CN624	6 m

### Allowed Number of Programming Devices

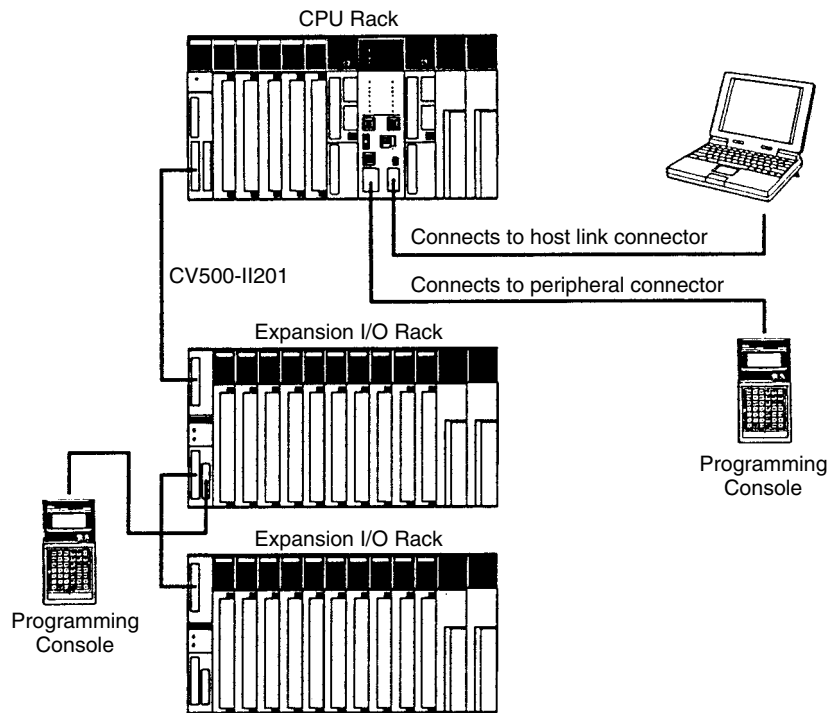
A total of four Programming Consoles can be connected to a CVM1D PC, as shown in the following table. Only one Programming Console can be connected to the CPU Unit or an I/O Interface Unit.

If a Programming Console is connected to the CPU or an I/O Interface Unit, 3 more Programming Devices can be connected to SYSMAC BUS/2 Remote I/O Slave Units. If no Programming Console is connected to the CPU Unit or I/O Interface Unit, 4 Programming Consoles can be connected to SYSMAC BUS/2 Remote I/O Slave Units. Up to 2 Programming Consoles can be connected to Remote I/O Slaves under a single Remote I/O Master Unit.

Connecting Unit	Max. connection combinations			Comments
CPU Unit	1	0	0	The same functions are supported when connected to either Unit.
CV500-II201 I/O Interface Unit	0	1	0	
SYSMAC BUS/2 Slave Unit	3	3	4	The I/O Table Register, I/O Table Change, and SYSMAC LINK Broadcast Test operations are not supported.

### 1-7-2 Connecting Programming Devices

Programming Devices can be connected to the Duplex Unit's peripheral connector, the Duplex Unit's host link connector, or a CV500-II201 I/O Interface Unit's peripheral connector.



The baud rate for transmissions with a personal computer running SSS depends upon the location where the computer is connected.

Connection location		Programming Device	Baud rate
Duplex Unit	Peripheral connector	Programming Console	50 Kbps
		SSS	19.2, 9.6, or 4.8 Kbps
	Host link connector	SSS	9.6 Kbps
I/O Interface Unit (CV500-II201)	Peripheral connector	Programming Console	50 Kbps
	Host link connector	SSS	19.2, 9.6, or 4.8 Kbps

Set the baud rate on the Duplex Unit's DIP Switch. When the host link connector is being used, the baud rate can be changed in the PC Setup (19.2, 9.6, 4.8, or 2.4 Kbps). To enable the baud rate setting in the PC Setup, turn OFF pins 3 and 4 of the Duplex Unit's DIP Switch.

### 1-7-3 Transferring Programs

A program can be written or edited offline with the SSS and then transferred to PC online. Refer to the *SSS Operation Manual* for details on writing, editing, and transferring programs.

### 1-7-4 Recommended Programming Device Operations

#### Checking the PC Setup

The CVM1D's PC Setup settings can be changed with the SSS. The PC Setup settings are all set to their default values when the CPU Unit is shipped. Verify that the PC Setup settings are appropriate for the planned PC applications. Refer to *Section 7 PC Setup* for details on the PC Setup. Refer to the *SSS Operation Manual* for details on changing the PC Setup settings.

- Setting the Clock**                      The CPU Unit's internal clock is not set when the CPU Unit is shipped. Be sure to set the clock with a Programming Device. Refer to the Programming Device's Operation Manual for details.
- Clearing the Error Log**              Before using the CVM1D, clear the Error Log Area by force-setting (forcing ON) the Error Log Reset Bit (A00014).

## 1-8 CVM1D Program Memory

Program Memory is contained in the CPU and is divided into two areas, the PC Setup and the Program Area. There are 64K words available in the CVM1D. The first 2K words are taken up by the PC Setup, leaving 62K words in the Program Area. (One word contains two bytes.)

Program Memory is backed up by the CPU battery, so data will not be lost during a power interruption.

**Note** The program memory chip is built into CVM1D PCs and does not need to be installed by the user.

- PC Setup**                                This area of Program Memory contains the settings described in *Section 7 PC Setup*. Basic options in PC operation (such as the method of I/O refreshing and the PC mode at start-up) are specified in these settings.  
The PC Setup is stored in EEPROM, so this data will not be lost even if the back-up battery power is interrupted.

- Program Area**                        This area of Program Memory contains the ladder program. There are approximately 60K words available for the ladder program because 1.85K words are reserved for system use.

## 1-9 Related Manuals

The following manuals are available for the various CVM1D-series and related products. Catalog number suffixes have been omitted; be sure you have the most recent version for your region.

Product	Model number	Manual	Cat. No.
CVM1D Duplex System PCs	---	CVM1D Duplex System PC Installation Guide	W350
	---	CVM1D Duplex System PC Operation Manual	W351
CV-series Host Link Unit	CV500-LK201	CV-series PCs Operation Manual: Host Link System	W205
SYSMAC Support Software (SSS)	C500-ZL3AT1-E	SYSMAC Support Software Operation Manual: Basics	W247
		SYSMAC Support Software Operation Manual: CVM1 PCs	W249
Programming Console	CVM1-PRS21-V1E	Programming Console Operation Manual	W222
CompoBus/D Master Unit	CVM1-DRM21-V1	CompoBus/D Master Unit Operation Manual	W267
Controller Link Unit	CVM1-CLK21	Controller Link Unit Operation Manual	W309
Controller Link Support Software	C200H-ZW3AT2-E	Controller Link Support Software Operation Manual	W308
SYSMAC NET Link Unit	CVM1-SNT31	SYSMAC NET Link System Manual	W213
SYSMAC LINK Unit	CV500-SLK11/SLK21	SYSMAC LINK System Manual	W212
SYSMAC BUS/2 Remote I/O Units	CV500-RM211/RM221/RT211/ RT221	SYSMAC BUS/2 Remote I/O System Manual	W204
SYSMAC BUS Wired Remote I/O Units	3G2A5-RM201/RT201	SYSMAC BUS Wired Remote I/O System Manual	W120
Ethernet Unit	CV500-ETN01	CV-series Ethernet System Manual	W242
BASIC Unit	CV500-BSC11/BSC21/BSC31/BSC41/BSC51/BSC61	BASIC Unit Reference Manual	W207
		BASIC Unit Operation Manual	W206

Product	Model number	Manual	Cat. No.
Temperature Controller Data Link Unit	CV500-TDL21	Temperature Controller Data Link Unit Operation Manual	W244
Personal Computer Unit	CV500-VP213-E/ VP223-E/VP217-E/ VP227-E	Personal Computer Unit Operation Manual	W251
		Personal Computer Unit Technical Manual	W252
Analog I/O Units	3G2A5-AD001/AD002/ AD003/AD004/AD005/A D006/AD007, C500-AD101 3G2A5-DA001/DA002/ DA003/DA004/DA005, C500-DA101/DA103	Analog I/O Units Operation Manual	W258
	C500-AD501	Analog I/O Unit Operation Manual	W259
High-speed Counter Unit	3G2A5-CT001, C500-CT012	High-speed Counter Unit Operation Manual	W129
	C500-CT021	High-speed Counter Unit Operation Manual	W253
	C500-CT041	High-speed Counter Unit Operation Manual	W185
Motion Control Unit	CV500-MC221/MC421	Motion Control Unit Operation Manual: Introduction	W254
		Motion Control Unit Operation Manual: Details	W255
Position Control Unit	C500-NC113/NC211	Position Control Unit Operation Manual	W323
	3G2A5-NC111-EV1	Position Control Unit Operation Manual	W142
	C500-NC222-E	Position Control Unit Operation Manual	W231
MC Support Software	CV500-ZN3AT1-E	MC Support Software Operation Manual	W247
Teaching Box	C500-TU002-E	Teaching Box Operation Manual	W145
	CVM1-PRS71	Teaching Box Operation Manual	W257
Cam Positioner Unit	C500-CP131	Cam Positioner Unit Operation Manual	W144
Fuzzy Logic Unit	C500-FZ001	Fuzzy Logic Unit Operation Manual	W209
ASCII Unit	C500-ASC04	ASCII Unit Operation Manual	W186
ID Sensor Unit	C500-IDS01-V2/ IDS02-V1/IDS21/ IDS22	ID Sensor Unit Operation Manual	W126
Memory Card Writer	CV500-MCW01-E	Memory Card Writer Operation Manual	W214

## SECTION 2

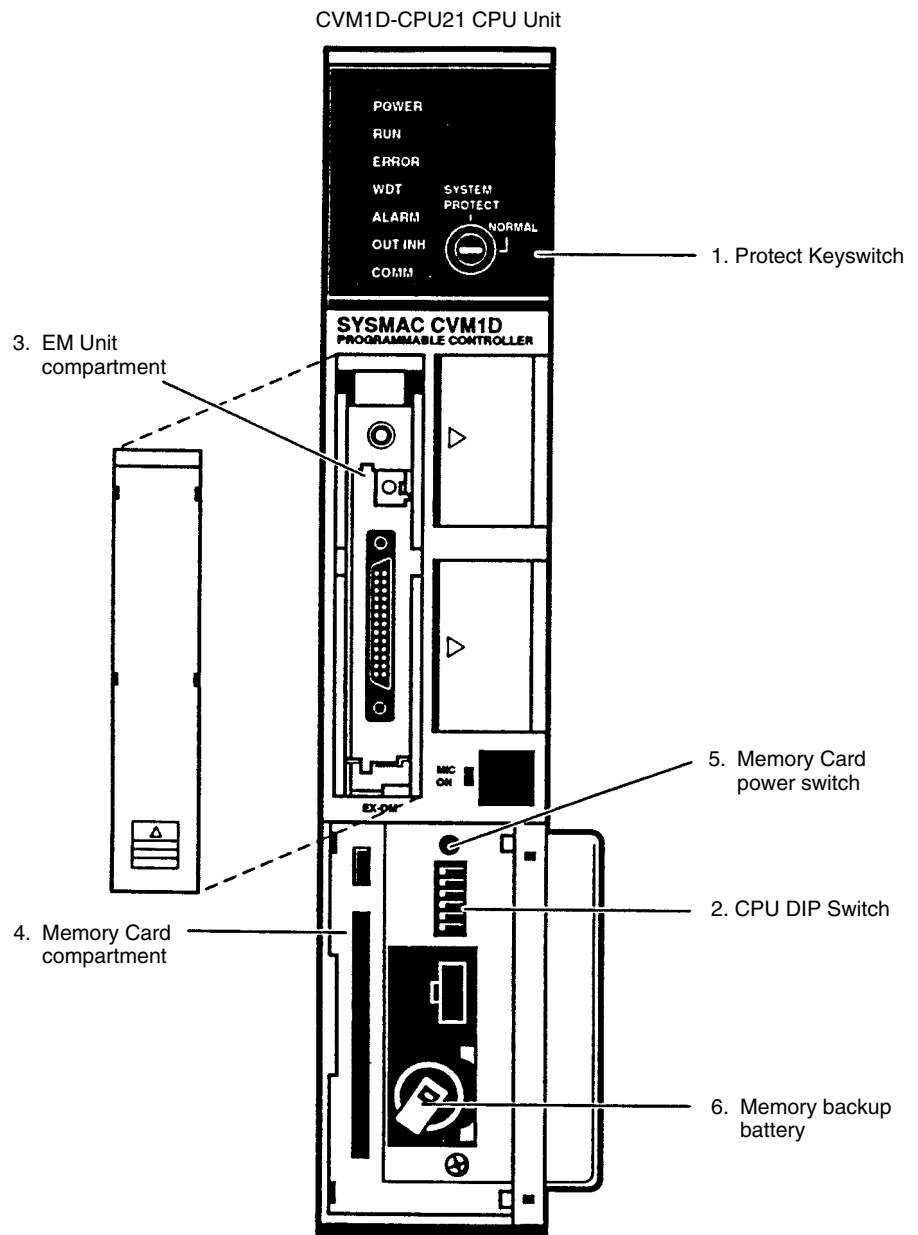
# Hardware Considerations

This section provides information on hardware aspects of CVM1D PCs that are relevant to programming and software operation. These include switches on the CPU Unit and basic PC configuration. This information is covered in more detail in the *CVM1D Installation Guide*.

2-1	CPU Unit Switch Settings .....	12
2-2	Duplex Unit Switch Settings .....	14
2-3	EM Units .....	17
2-4	Memory Cards .....	18
2-4-1	Mounting and Removing Memory Cards .....	18
2-4-2	File Transfer between the CPU and Memory Card .....	19
2-4-3	Simplified Backup Function .....	21
2-5	Setting Rack Numbers .....	22
2-6	I/O Control Unit and I/O Interface Unit Displays .....	22



## 2-1 CPU Unit Switch Settings

There aren't any special preparations required for the CPU Unit during programming, but the switch settings described in this section may be required before starting operation.

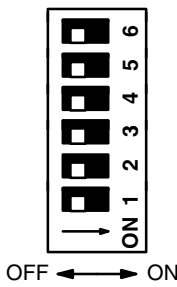


1, 2, 3... 1. Protect Keyswitch

Be sure that the Protect Keyswitch is set to NORMAL when transferring the program, writing the program, or changing the PC Setup. The program cannot be edited or transferred when the keyswitch is set to PROTECT.

Position	Function
 <b>NORMAL</b>	Memory is not write-protected.
<b>PROTECT</b> 	Write-protects the user program, PC Setup, and CPU Bus Unit settings and prevents program transfers from SSS.  The Protect Keyswitch's write-protection is temporarily suspended during duplex synchronization or program transfer in duplex mode. Write-protection is enabled again after the program transfer is completed.

2. CPU DIP Switch

Pin number(s)	Position	Function	
	1, 2	---  Pins 1 and 2 determine which files will be transferred from the CPU Unit to the Memory Card. (In this case, UM indicates the user program.)  Start the data transfer by pressing the Memory Card power switch for 3 seconds.	
	1	2	Back up UM, PC Setup, I/O memory, and DM.
	OFF	OFF	Back up PC Setup.
	ON	OFF	Back up UM.
	OFF	ON	Back up I/O memory and DM.
	ON	ON	Not used. (Leave this pin set to OFF.)
	3	OFF	Not used. (Leave this pin set to OFF.)
4	OFF	Not used. (Leave this pin set to OFF.)	
5	OFF	Files are not transferred from the Memory Card automatically at start-up.	
	ON	The program file, PC Setup file, and CPU Bus Unit settings file will be transferred from the Memory Card to the CPU Unit automatically at start-up.  (In a duplex system, the files are transferred in the active CPU Unit only. The files aren't transferred in the standby CPU Unit. The automatic transfer at start-up setting can be made in the PC Setup as well, but the setting on pin 5 supercedes the PC Setup setting. The files will not be transferred if the Protect Keyswitch is set to PROTECT.)	
6	OFF	Not used. (Leave this pin set to OFF.)	

3. EM Unit Compartment

When the EM Area is being used in a duplex system, be sure to install EM Units with the same specifications in the active and standby CPU Units. Refer to 2-3 EM Units for more details.

4. Memory Card Compartment

When Memory Cards are being used in a duplex system, be sure to install Memory Cards with the same specifications in the active and standby CPU Units. Refer to 2-4 Memory Cards for more details.



5. Memory Card Power Switch

Press and release this switch to turn the power to the Memory Card off and on. The M/C ON indicator will be lit when the power is on. Never remove the Memory Card while the power is on. The Memory Card will not function when the M/C ON indicator is not lit.

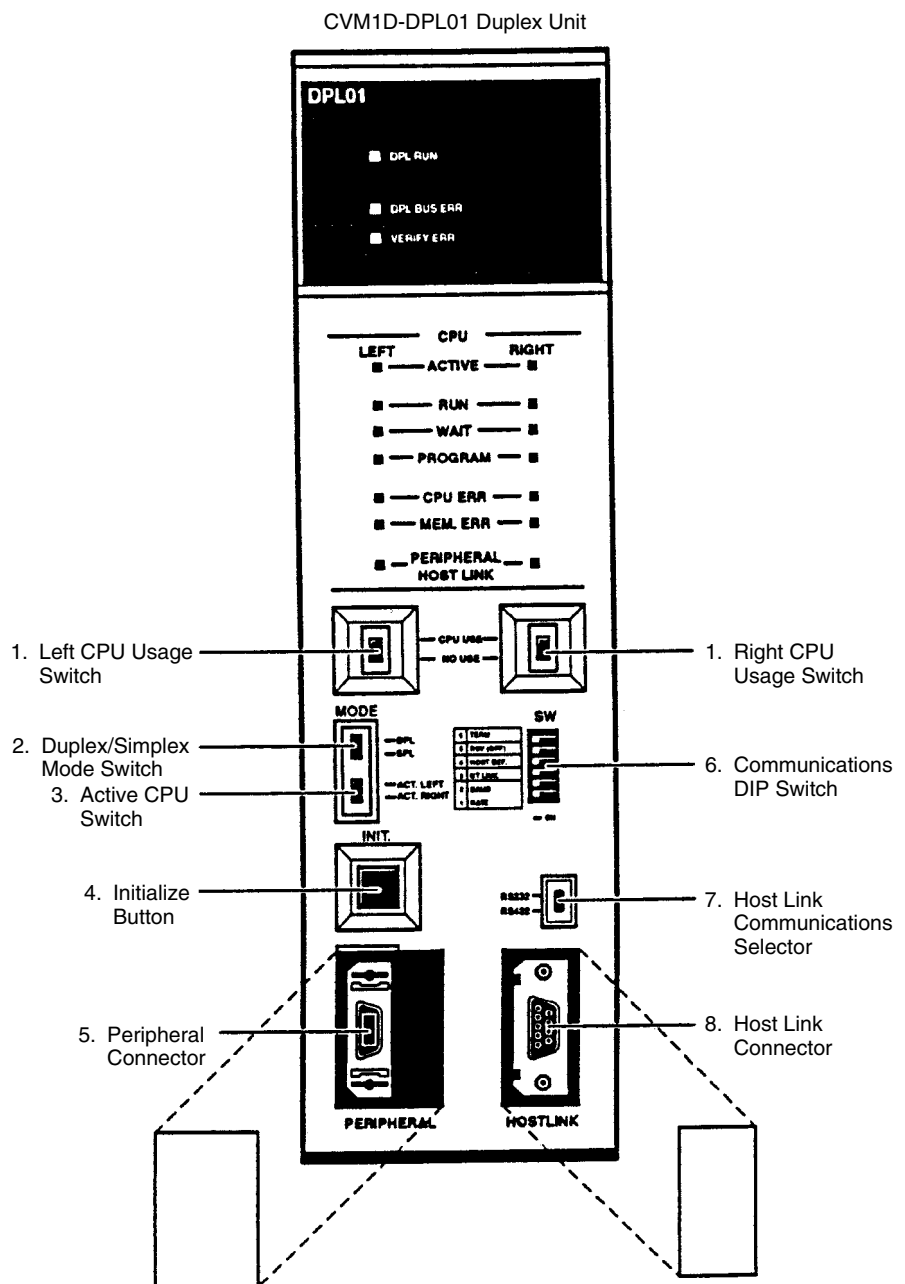
The Memory Card power switch also controls the simplified backup function. Press and hold the switch for 3 seconds to start the backup. Refer to 2-4-3 Simplified Backup Function for details.

6. Memory Backup Battery

Verify that the backup battery is installed before operating the PC.

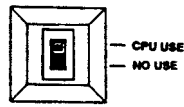
## 2-2 Duplex Unit Switch Settings

Some Duplex Unit switch settings are required before programming and others are required before starting operation.



1, 2, 3... 1. CPU Usage Switches

These switches control the power supply to the left and right CPU Units.

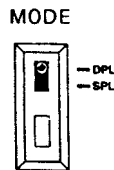


In a duplex system, set both CPU Usage Switches to "CPU USE." In a simplex system, set the desired CPU Unit's Usage Switch to "CPU USE" and the other CPU Unit's switch to "NO USE."

Power will be supplied to a CPU Unit when its CPU Usage Switch is set to "CPU USE." Set the switch to "NO USE" when the CPU Unit isn't being used or the CPU Unit is being replaced with the PC power on.

2. Duplex/Simplex Mode Switch

Set this switch to DPL to operate the PC as a duplex system; set it to SPL to operate the PC as a simplex system.

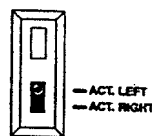


DPL: Set to DPL for duplex operation.  
SPL: Set to SPL for simplex operation.

This switch setting becomes effective when the power is turned on, a CPU Usage Switch is switched from NO USE to CPU USE, or the Initialize Button is pressed.

3. Active CPU Switch

In a duplex system, this switch determines whether the left or right CPU Unit is the active CPU Unit. Select which CPU Unit is the active CPU when operating the PC in duplex mode.

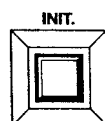


ACT. LEFT: Sets the left CPU Unit as the active CPU Unit.  
ACT. RIGHT: Sets the right CPU Unit as the active CPU Unit.

This switch setting becomes effective when the power is turned on or a CPU Usage Switch is switched from NO USE to CPU USE. Setting changes during operation are ignored.

4. Initialize Button

Initializes changes in duplex system operation. For example, press this button when you want to switch from simplex mode to duplex mode after replacing a CPU Unit in which an error occurred.



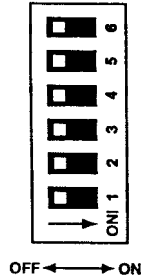
5. Peripheral Connector

This connector is used to connect a Programming Device such as a computer running SSS or a Programming Console. The baud rate depends upon the Programming Device that is connected, so be sure to set the baud rate on the Communications DIP Switch. (See below.)

6. Communications DIP Switch

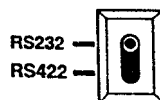
Sets the CPU Unit's communications specifications.

Pin number(s)	Position	Function
1, 2	---	Pins 1 and 2 set the baud rate for the peripheral connector. Set the baud rate to 50,000 bps for a Programming Console or 9,600 bps for a personal computer running SSS.
	1   2	
	OFF   OFF	Programming Device communications: 50,000 bps
	ON   OFF	Programming Device communications: 19,200 bps
	OFF   ON	Programming Device communications: 9,600 bps
	ON   ON	Programming Device communications: 4,800 bps
3	OFF	Communicate via Host Link communications
	ON	Communicate with PT (Programmable Terminal) via NT Link communications.
4	OFF	Host link communications governed by PC Set-up. (Effective only when pin 3 is OFF.)
	ON	Initializes host link communications specifications to the following settings when connecting a personal computer. (Effective only when pin 3 is OFF.)  Baud rate: 9,600 bps Unit number 00 Even parity 7-bit data 2 stop bits
5	OFF	Not used. (Leave this pin set to OFF.)
6	---	Sets the termination resistance for RS-422 communications. (This setting is not used for RS-232C communications. Leave OFF when using RS-232C communications.)
	OFF	Termination resistance OFF.
	ON	Termination resistance ON.



7. Host Link Communications Selector

Set the switch to the communications specification desired for the host link connector.



8. Host Link Connector

This connector is used for RS-232C or RS-422 communications. Select RS-232C or RS-422 communications with the host link communications selector (7). The PERIPHERAL/HOST LINK indicators indicate whether a device is connected to the left or right CPU Unit.

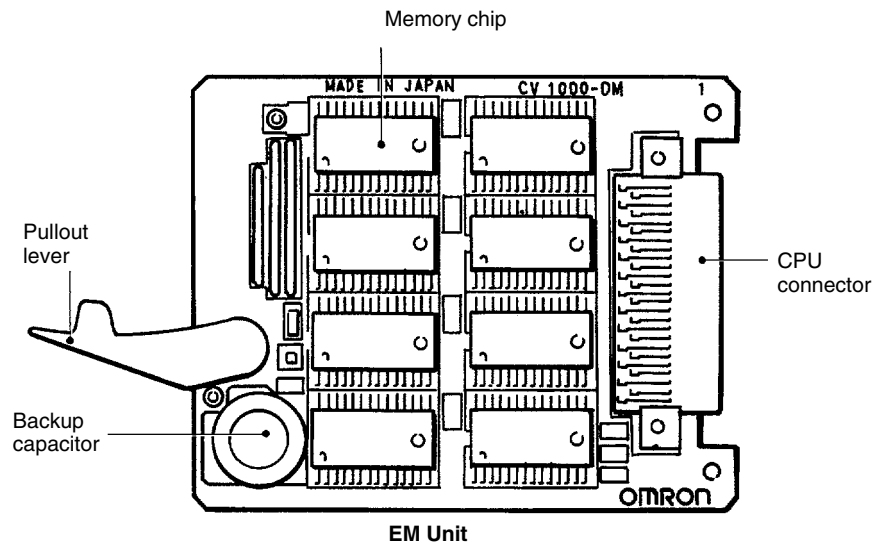
## 2-3 EM Units

If the DM Area capacity of 24K words is insufficient, an EM Unit (sold separately) can be added to create an EM (Expansion Data Memory) Area providing up to 256K words of additional memory. In a duplex system, the left and right CPU Units must have EM Units with the same specifications.

There are three models of EM Units available, as shown in the following table.

Model	Memory capacity	Memory banks
CV1000-DM641	64K words	2 (0 and 1)
CV1000-DM151	128K words	4 (0 to 3)
CV1000-DM251	256K words	8 (0 to 7)

The following diagram shows the structure of the EM Unit and identifies its main components.



The EM Area operates the same as the DM Area, but the EM Area memory is contained in the EM Unit, while DM Area memory is internal.

EM Area memory is divided into banks of 32K words each. Words E00000 to E32765 of the current bank can be accessed. The current bank number is contained in the least significant digit of A511. A511 is in a read-only area, but the current bank number can be changed with the EMBC(171) instruction. Refer to *Section 5 Instruction Set* for details.

Refer to the *CVM1D Installation Guide* for details on installing and removing EM Units.

## 2-4 Memory Cards

Memory Cards can be installed in the CPU Unit to store programs and other data as file memory. In a duplex system, the left and right CPU Units must have Memory Cards with the same capacity.

The CVM1D does not support Memory Card instructions, but the program and other data can be transferred from the Memory Card to the CPU Unit automatically at start-up. (This function is enabled with pin 5 of the CPU DIP Switch.)

The following table shows the Memory Cards that are available.

Memory type	Total capacity	File capacity	Model No.	Battery life (See note 1)
RAM	64K bytes	61K bytes	HMC-ES641	About 5 yrs
	128K bytes	125K bytes	HMC-ES151	About 3 yrs
	256K bytes	251K bytes	HMC-ES251	About 1 yr
	512K bytes	506K bytes	HMC-ES551	About 0.5 yrs
EEPROM (See note 2)	64K bytes	61K bytes	HMC-EE641	Not applicable
	128K bytes	125K bytes	HMC-EE151	
EPROM (See note 2)	512K bytes	506K bytes	HMC-EP551	
	1M bytes	1016K bytes	HMC-EP161	

- Note**
1. Batteries should be replaced before the end of their life expectancy. Refer to the *CVM1D Installation Guide* for details on battery replacement.
  2. Cannot be used without a CV500-MCW□□ Memory Card Writer.


The SSS can be used to write data to RAM and EEPROM Cards, but a CV500-MCW□□ Memory Card Writer must be used to write data to EPROM Memory Cards.

The following commercially available memory cards can be used. The procedures and applications for using these memory cards is exactly the same as for the Memory Cards provided by OMRON.

- RAM Memory Cards conforming to JEIDA4.0 and of the following sizes: 64 Kbytes, 128 Kbytes, 256 Kbytes, 512 Kbytes, 1 Mbyte, and 2 Mbytes.

- Note** The 2-Mbyte Memory Cards cannot be used in the CV500-MCW01 Memory Card Writer.

Memory Cards must be formatted before use. RAM and EEPROM Cards can be formatted with the SSS or the CV500-MCW□□ Memory Card Writer; EPROM Memory Cards can be formatted with the CV500-MCW□□ Memory Card Writer only.

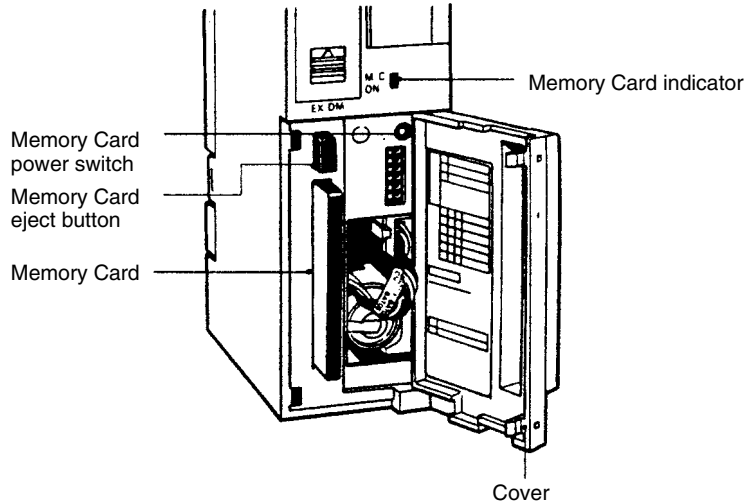
-  **Caution** Memory Cards can be damaged by twisting, shock, or exposure to high temperature, humidity, or direct sunlight. Handle them with care.

### 2-4-1 Mounting and Removing Memory Cards

**Mounting a Memory Card** Mount a Memory Card to the CPU using the following procedure.

- 1, 2, 3... 1. Open the cover of the Memory Card compartment.
2. If the Memory Card is RAM or EEPROM, set the write-protect switch to OFF so that data can be written to the Card.
3. Insert the Memory Card into its compartment. In doing so, a slight resistance will be felt as the connector on the Memory Card mates with the connector on the CPU. Continue pushing until the Memory Card is inserted completely into the CPU. If the Memory Card power switch is ON, the Memory Card indicator will light.

4. Close the cover.



**Removing a Memory Card**

1, 2, 3...

1. Open the cover of the Memory Card compartment.
2. Press the Memory Card power switch once if the Memory Card indicator is lit. The Memory Card indicator will turn OFF.
3. Press the Memory Card eject button. The Memory Card will be released allowing it to be removed.
4. Pull out the Memory Card.
5. Close the cover.

**Note**

1. Do not expose the Memory Card to high temperature, humidity, or direct sunlight.
2. Do not bend the Card or subject it to shock.
3. Do not apply excess force to the Card when inserting or removing it.
4. Do not remove the Card while the M/C ON indicator is lit; doing so may result in data errors in the memory.

**2-4-2 File Transfer between the CPU and Memory Card**

**Memory Card Files**

Memory Card files are identified by both their filename and filename extension. The following table lists the filenames and filename extensions that are used with the PC. Filenames are eight characters long and recorded in ASCII. If fewer than eight characters are needed, enter spaces (ASCII 20) in the remaining bytes.

Type of file	Filename
Extended PC Setup <sup>1</sup>	filename.STD
Data files	filename.IOM
Ladder program files (files saved with the partial save operation)	filename.LDP
Program file (complete program)	filename.OBJ
Extended PC Setup <sup>1</sup> (transferred at start-up)	AUTOEXEC.STD
Program file (complete program transferred at start-up)	AUTOEXEC.OBJ

**Note**

1. Extended PC Setup includes the PC Setup, I/O table, routing tables, data link tables for data links in SYSMAC LINK, SYSMAC NET Link, and Controller Link Systems, Communications Unit settings, BASIC Unit memory switches, and customized settings (function codes and data areas).

2. The files that will be transferred at start-up must be named "AUTOEXEC."
3. Files called BACKUP are created when the simplified backup function is used. Refer to 2-4-3 *Simplified Backup Function* for details.

### File Transfer at Start-up

1, 2, 3...

There are two methods for automatic transfer of files at start-up:

1. When pin 5 of the CPU DIP switch is ON, the Extended PC Setup file (AUTOEXEC.STD) and the program file (AUTOEXEC.OBJ) are both transferred to Program Memory at start-up. If either of the files is missing, a memory error will occur and neither file will be transferred.
2. The PC Setup can be set (setting D, Program Transfer at Start-up) to transfer the program file (AUTOEXEC.OBJ) from the Memory Card to the PC automatically when the PC is turned on. In this case the extended PC Setup file (AUTOEXEC.STD) is not transferred.

With either method, the transfer will not proceed if the write-protect switch is ON, but will proceed even if the program memory access right is restricted from the SSS. The transfer normally takes about 4 seconds.

To enable file transfer at start-up, the proper files must be recorded on a Memory Card in advance from the SSS. The Extended PC Setup file (AUTOEXEC.STD) can be transferred directly from the PC to the Memory Card in online operations from the SSS. The program file (AUTOEXEC.OBJ) can be created on the Memory Card using one of the following two operations.

- In online operations, transfer the program and other files to the PC and then transfer the program file to the Memory Card from the PC.
- Convert the program into an object file in offline SSS operations, and then transfer it directly to the Memory Card in online operations.

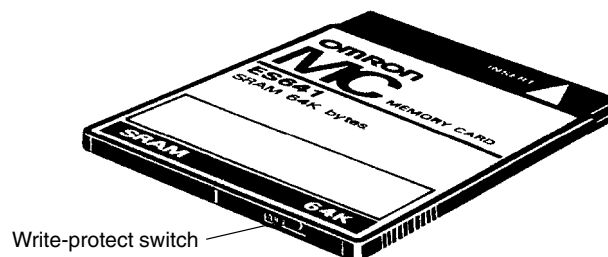
If the PC is set to transfer the program at start-up but the transfer cannot be completed for some reason, the Memory Card Startup Transfer Error Flag (A40309) will be turned ON, a memory error will occur, and the PC will not begin operation. When the program is not transferred, either find and eliminate the cause of the error or change the PC settings so that the program won't be transferred, and then turn the PC off and on. The following are possible reasons that the program cannot be transferred:

- The Protect Keyswitch is set to PROTECT.
- One or both AUTOEXEC files are missing.
- The Memory Card power switch is OFF. (If the M/C ON indicator is not lit when the PC power is ON, press the Memory Card power switch.)
- The Memory Card is not installed.

### Reading and Writing Memory Card Files

It is not possible to write to an EPROM Card installed in the CPU. Use the CV500-MCW□□ Memory Card Writer to write to an EPROM Card. Refer to the *Memory Card Writer Operation Manual* for details. Set the drive name to "0" when accessing a Memory Card.

The RAM and EEPROM cards have a write-protect switch, as shown in the diagram below. Turn this switch to OFF when writing to or erasing the Memory Card.



The three methods of reading and writing Memory Card files are listed below.

1, 2, 3...

1. Reading and writing can be performed as an online operation with a Programming Device, e.g., the SSS.

2. Reading and writing can be performed by a command from a host computer.
3. Reading and writing can be performed by using the simplified backup function. Refer to 2-4-3 *Simplified Backup Function* for details.

### 2-4-3 Simplified Backup Function

The user program, Extended PC Setup, and IOM/DM data can be backed up from the CPU Unit to a Memory Card very simply without using a Programming Device. The backup data in the Memory Card can be transferred back to the CPU Unit later. (This method is provided as an easy way to backup and restore data, but we still recommend that a Programming Device be used to verify the data.)

#### Backing Up Data to a Memory Card

Use the following procedure to prepare to backup data in the memory of the CPU Unit to a Memory Card.

- 1, 2, 3...
  1. Insert a Memory Card that is not write-protected and check to be sure the available capacity is sufficient for the files that will be created.
  2. Confirm that the Memory Card is not being accessed by file memory operations or from a Programming Device.
  3. Turn OFF pin 5 on the CPU DIP switch.
  4. In a duplex system, confirm that the CPU Unit is the active Unit.

#### Transferring Data Back to CPU Unit Memory

Use the following procedure to prepare to transfer data on the Memory Card to the memory of the CPU Unit.

- 1, 2, 3...
  1. Insert the Memory Card and be sure that it contains the desired files.
  2. Check the file checksums and sizes to be sure that they are correct.
  3. Confirm that the CPU Unit is in PROGRAM mode.
  4. Confirm that the Memory Card is not being accessed from a Programming Device.
  5. Turn ON pin 5 on the CPU DIP switch.
  6. In a duplex system, confirm that the CPU Unit is the active Unit.

#### Specifying Files

Pins 1 and 2 on the DIP switch are used to specify the files to be transferred. Set pins 1 and 2 as shown in the following table.

Pin 1	Pin2	User program	Extended PC Setup	IOM/DM
OFF	OFF	Transferred.	Transferred.	Transferred.
OFF	ON	Transferred.	Not transferred.	Not transferred.
ON	OFF	Not transferred.	Transferred.	Not transferred.
ON	ON	Not transferred.	Not transferred.	Transferred.
File name (See note.)		BACKUP.OBJ	BACKUP.STD	IOM: BACKUP.IOM DM: BACKUPDM.IOM EM: BACKUPE*.IOM (* = bank number)

**Note** Any files of the same name will be automatically overwritten when backing up to Memory Card.

#### Starting and Confirming Data Transfers

Data transfers are started by pressing the Memory Card power switch for 3 seconds. If the transfer ends normally, the Memory Card indicator will flash once and will then go out when the transfer has completed. The time required will depend on the amount of data being transferred. If there is insufficient memory available on the Memory Card to back up the specified data or if the specified files are not present on the Memory Card when restoring data, the Memory Card indicator will flash 5 times and then go out.



## 2-5 Setting Rack Numbers

I/O words are allocated to Units mounted on the CPU, Expansion CPU, and CV-series Expansion I/O Racks by rack number, regardless of the order in which the Racks are connected. The CPU Rack number is fixed at 0, so I/O bits are always allocated first to Units on the CPU Rack. Set the rack numbers of Expansion CPU and Expansion I/O Racks to unique numbers between 1 and 7.

Be sure to set the rack numbers before registering the I/O table.

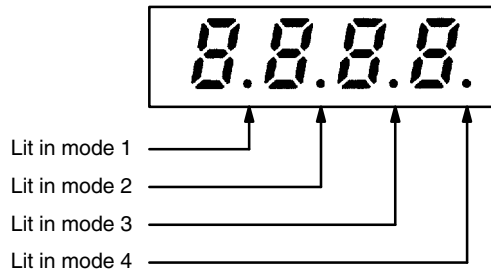
**Note** The PC Setup can be used to control I/O word allocation to Racks and override allocation by rack number. Refer to *Section 7 PC Setup* for details.

The rack numbers for Expansion CPU and Expansion I/O Racks are set with the rack number switch (RACK No.) on the I/O Interface Unit mounted on the Rack. Set the rack number with a standard screwdriver after turning off the Rack power supply and be careful not to damage the switch groove.

- Note**
1. A duplication error will occur if 2 or more Racks have the same rack number.
  2. If a rack number is set to 8 or 9, the Rack will not be recognized by the CPU.

## 2-6 I/O Control Unit and I/O Interface Unit Displays

The I/O Control Unit and I/O Interface Unit have four-character 7-segment displays on the front. There are four display modes that display various information from the CPU, and the current display mode is indicated by the position of the decimal point on the display, as shown in the following diagram.

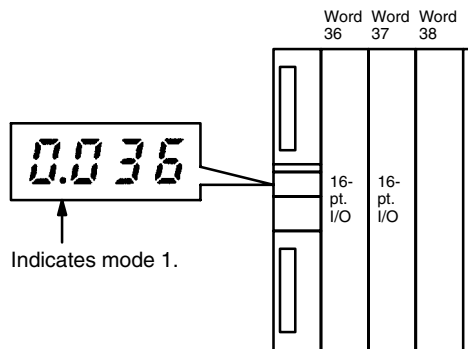


Pressing the mode selector switch changes the display to the next mode. The Unit will automatically enter the mode specified in the PC Setup (default setting: mode 1). Refer to *Section 7 PC Setup* for details.

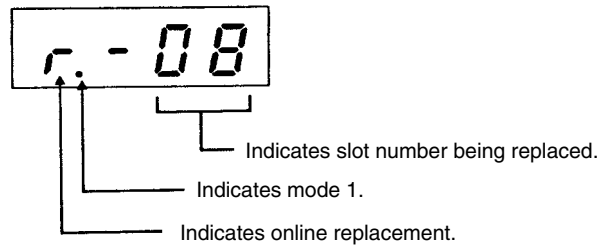
If the CPU Rack power supply is OFF or an initialization error has occurred, the displays will show "—" and the rack number will be displayed when the mode selector switch is held down, but the mode will not be changed.

### Display Mode 1

In mode 1, the first I/O word allocated to that Rack is displayed. If the I/O table hasn't been registered yet, or an error occurred during registration, the display will show "0000." In the following example, the first word allocated is CIO 0036.



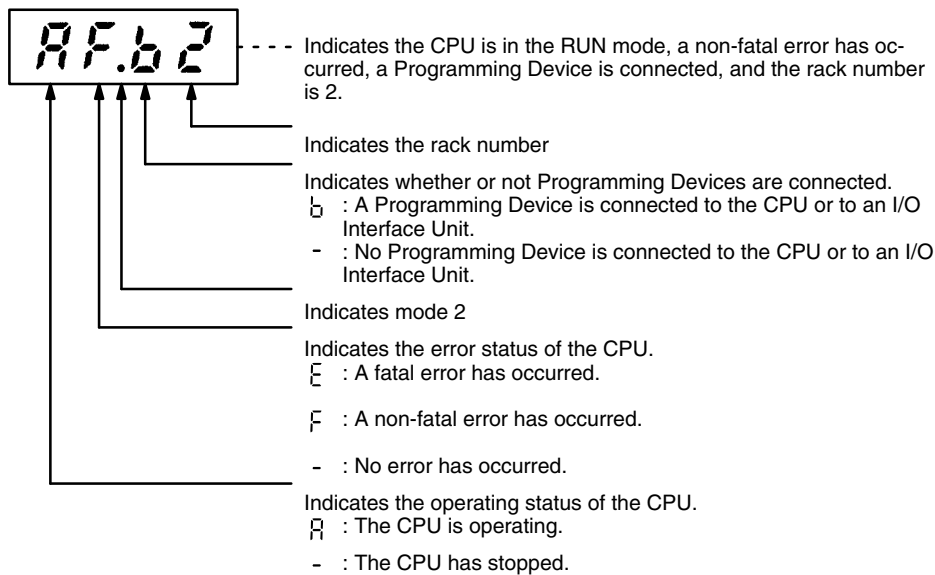
During online I/O Unit replacement, the slot number of the Unit that can be replaced is displayed. In the following example, the Unit in slot number 8 can be replaced.



**Display Mode 2**

In mode 2, the current CPU status and the rack number of that Rack are displayed. The information displayed by the four digits is listed below.

- 1, 2, 3...
- The leftmost digit indicates whether or not the CPU is operating.
    - "R" indicates it is operating.
    - "-" indicates it is stopped.
  - The second digit indicates whether or not an error has occurred in the PC.
    - "E" indicates that a fatal error has occurred.
    - "F" indicates that a non-fatal error has occurred.
    - "-" indicates that no errors have occurred.
  - The third digit from the left indicates whether or not a Programming Device is connected to the Rack (to the CPU Unit or I/O Interface Unit). If a Programming Device is already connected, another cannot be connected.
    - "b" indicates that a Programming Device is connected.
    - "-" indicates that a Programming Device is not connected.
- Note** Only one Programming Device can be connected to a CPU Unit or I/O Interface Unit for each PC, but three additional Programming Devices can be connected to the SYSMAC BUS/2 Slave Racks.
- The rightmost digit indicates the rack number.

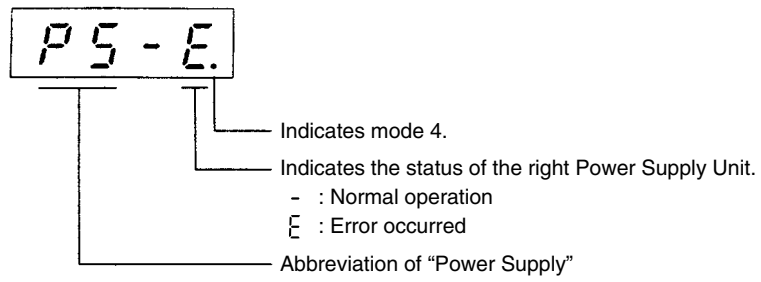


**Display Mode 3**

In mode 3, the display shows a 4-character message when an IODP(189) instruction is executed in the program for that Unit. The display mode of the destination unit can be changed to mode 3 automatically by the instruction. Refer to Section 5 Instruction Set for details on IODP(189).

**Display Mode 4**

In mode 4, the display shows the status of the duplex power supply. When an error has occurred, that information takes priority.



# SECTION 3

## Memory Areas

This section describes the way in which PC memory is broken into various areas used for different purposes. The contents of each area and addressing conventions, including the use of indirect addressing and addressing registers, are also described.

3-1	Introduction .....	27
3-2	Data Area Structure .....	27
3-3	CIO (Core I/O) Area .....	31
3-3-1	I/O Area .....	32
3-3-2	Work Areas .....	36
3-3-3	SYSMAC BUS/2 Area .....	37
3-3-4	Link Area .....	37
3-3-5	Holding Area .....	37
3-3-6	CPU Bus Unit Area .....	38
3-3-7	CompoBus/D Areas .....	38
3-3-8	SYSMAC BUS Area .....	38
3-4	TR (Temporary Relay) Area .....	39
3-5	CPU Bus Link Area .....	39
3-6	Auxiliary Area .....	41
3-6-1	IOM Hold Bit .....	46
3-6-2	Forced Status Hold Bit .....	46
3-6-3	Error Log Reset Bit .....	47
3-6-4	Output OFF Bit .....	47
3-6-5	CPU Bus Unit Restart Bits .....	47
3-6-6	SYSMAC BUS Error Check Bits .....	47
3-6-7	SSS Flags .....	47
3-6-8	Start-up Time .....	48
3-6-9	Power Interruption Time .....	48
3-6-10	Number of Power Interruptions .....	48
3-6-11	Service Disable Bits .....	48
3-6-12	Message Flags .....	49
3-6-13	Error Log Area .....	49
3-6-14	MACRO Input/Output Areas .....	50
3-6-15	CPU Bus Unit Initializing Flags .....	50
3-6-16	Wait Flags .....	50
3-6-17	Duplex Startup Error Flags .....	51
3-6-18	Programming Device Flags .....	51
3-6-19	CPU Bus Unit Service Interval .....	51
3-6-20	Memory Card Flags .....	51
3-6-21	Alternate CPU Unit Status Flags .....	51
3-6-22	Duplex Verification Error Flags .....	52
3-6-23	Power Supply Unit OFF/Error Flags .....	52
3-6-24	Online I/O Replacement Flags .....	53
3-6-25	Error Code .....	53
3-6-26	FALS Flag .....	53
3-6-27	Cycle Time Too Long Flag .....	53
3-6-28	Program Error Flag .....	53
3-6-29	I/O Setting Error Flag .....	53
3-6-30	Too Many I/O Points Flag .....	53
3-6-31	CPU Bus Error and Unit Flags .....	53
3-6-32	Duplication Error Flag and Duplicate Rack/CPU Bus Unit Numbers .....	54

3-6-33	I/O Bus Error Flag and I/O Bus Error Slot/Rack Numbers	54
3-6-34	Memory Error Flag	54
3-6-35	Duplex Bus Error Flag	54
3-6-36	Power Interruption Flag	54
3-6-37	CPU Bus Unit Setting Error Flag and Unit Number	54
3-6-38	Battery Low Flags	54
3-6-39	SYSMAC BUS Error Flag, Check Bits, and Master/Unit Numbers	55
3-6-40	SYSMAC BUS/2 Error Flag and Master/Unit Numbers	55
3-6-41	CPU Bus Unit Error Flag and Unit Numbers	56
3-6-42	I/O Verification Error Flag	56
3-6-43	Indirect DM BCD Error Flag	56
3-6-44	Jump Error Flag	56
3-6-45	Duplex Power Supply Error Flag	56
3-6-46	FAL Flag and FAL Number	56
3-6-47	Memory Error Area Location	56
3-6-48	Memory Card Start-up Transfer Error Flag	57
3-6-49	CPU-recognized Rack Numbers	57
3-6-50	CPU Bus Unit Number Setting Error Flag	57
3-6-51	CPU Bus Link Error Flag	57
3-6-52	Maximum Cycle Time	57
3-6-53	Present Cycle Time	57
3-6-54	Instruction Execution Error Flag, ER	57
3-6-55	Arithmetic Flags	57
3-6-56	Step Flag	58
3-6-57	First Cycle Flag	58
3-6-58	Clock Pulse Bits	58
3-6-59	Network Status Flags	59
3-6-60	EM Status Flags	59
3-7	Timer Area	59
3-8	Counter Area	60
3-9	DM and EM Areas	60
3-10	Index and Data Registers (IR and DR)	62

## 3-1 Introduction

Various types of data are required to achieve effective and correct control. To facilitate managing this data, the PC is provided with various **memory areas** for data, each of which performs a different function. The areas generally accessible by the user for use in programming are classified as **data areas**. Details, including the name, range, and function of each area are summarized in the following table. The internal I/O memory addresses are shown in parentheses. These memory address are used for indirect addressing. Refer to *3-9 DM and EM Areas* and to *5-3 Data Areas, Definers, and Flags* for details on indirect addressing.

Area	Range	Function
CIO Area (Core I/O)	Words: CIO 0000 to CIO 2555 Bits: CIO 000000 to CIO 255515 (\$0000 to \$09FB)	The CIO (Core I/O) Area is divided into eight sections, five controlling I/O and three used to store and manipulate data internally. Refer to <i>3-3 CIO (Core I/O) Area</i> for details.
Temporary Relay Area	TR0 to TR7 (bits only) (\$09FF)	Used to temporarily store execution conditions. TR bits are not input when programming directly in ladder diagrams, and are used only when programming in mnemonic form.
CPU Bus Link Area	Words: G000 to G255 Bits: G00000 to G25515 (\$0A00 to \$0AFF)	G000 is the PC Status Area; G001 to G004, the Clock Area. G008 to G127 contain PC output bits; G128 to G255, CPU Bus Unit output bits.
Auxiliary Area	Words: A000 to A511 Bits: A00000 to A51115 (\$0B00 to \$0CFF)	Contains flags and bits with special functions.
Timer Area	T0000 to T1023 (Completion Flags: \$0F00 to \$0F3F Present Values: \$1000 to \$13FF)	Used to define timers (normal, high-speed, and totalizing) and to access Completion Flags, PV, and SV.
Counter Area	C0000 to C1023 (Completion Flags: \$0F80 to \$0FBF Present Values: \$1800 to \$1BFF)	Used to define counters (normal, reversible, and transition) and to access Completion Flags, PV, and SV.
DM Area	D00000 to D24575 (\$2000 to \$7FFF)	Used for internal data storage and manipulation.
EM Area	E00000 to E32765 for each bank; 2, 4, or 8 banks (\$8000 to \$8FFD)	EM functions just like DM. An Extended Data Memory Unit must be installed.
Index registers	IR0 to IR2	Used for indirect addressing.
Data registers	DR0 to DR2	Generally used for indirect addressing.

### Flags and Control Bits

Some data areas contain flags and/or control bits. Flags are bits that are automatically turned ON and OFF to indicate particular operation status. Although some flags (e.g., the Carry Flag) can be turned ON and OFF by the user, most flags are read only; they cannot be controlled directly.

Control bits are bits turned ON and OFF by the user to control specific aspects of operation. Any bit given a name using the word bit rather than the word flag is a control bit, e.g., Restart Bits are control bits.

## 3-2 Data Area Structure

### Addresses

There are two different sets of addresses that can be used to access PC memory: data area addresses or memory addresses. Data area addresses are used when specifying an address directly as an operand for an instruction. Memory addresses are used when using indirect addressing.

When designating a data area address, the acronym for the area (the letter(s) identifying the data area) is always required for any area except the CIO (Core I/O) Area. Although the CIO acronym is given for clarity in text explanations, it is not required and not entered when programming.

It is also possible to access any memory location through its hexadecimal internal I/O memory address with indirect addressing. Refer to 3-9 DM and EM Areas, and 3-10 IR and DR Areas, for details on indirect addressing.

**Word Structure**

Memory areas are divided up into words, each of which consists of 16 bits numbered 00 through 15 from right (least significant) to left (most significant). CIO words 0000 and 0001 are shown below with bit numbers. Here, the content of each word is shown as all zeros. Bit 00 is called the rightmost bit; bit 15, the leftmost bit.

The term least significant bit is often used for rightmost bit; the term most significant bit, for leftmost bit.

<b>Bit number</b>	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
<b>CIO word 0000</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<b>CIO word 0001</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Data in the DM Area and EM Area, as well as Timer and Counter PVs can be accessed as words only. Transition Flags, Step Flags, and Timer and Counter Completion Flags can be accessed as bits only. You cannot designate any of these for operands requiring bit data. Data in the CIO, CPU Bus Link, and Auxiliary Areas is accessible either by word or by bit, depending on the instruction in which the data is being used.

To designate one of these areas by word, all that is necessary is the acronym, if required, and the two-, three-, or four-digit word address. To designate an area by bit, the word address is combined with the bit number as a single four- to six-digit address. The following table shows examples of this. The two rightmost digits of a bit address must indicate a bit between 00 and 15, e.g., the rightmost digit must be 5 or less when the next digit to the left is 1.

The same timer and counter numbers can be used to designate either the present value (PV) of the timer or counter, or the Completion Flag for the timer or counter. This is explained in more detail in 3-7 Timer Area and 3-8 Counter Area.

Area	Word designation	Bit designation
CIO	0000	000015 (leftmost bit in word CIO 0000)
CIO	0252	025200 (rightmost bit in word CIO 0252)
DM	D01250	Not possible
T	T215 (designates PV)	T215 (designates Completion Flag)
A	A012	A01200

To designate a word by its internal I/O memory address, write the hexadecimal address to an Index Register, DM, or EM word and indirectly address the operand through that register or word. Refer to 3-9 DM and EM Areas and 3-10 IR and DR Areas for details on indirect addressing.

**Data Structure**

Word data input as decimal values is stored in binary-coded decimal (BCD); word data entered as hexadecimal is stored in binary form. Each four bits of a word represent one digit, either a hexadecimal or decimal digit, numerically equivalent to the value of the binary bits. One word of data thus contains four digits, which are numbered from right to left. These digit numbers and the corresponding bit numbers for one word are shown below.

<b>Digit number</b>	3				2				1				0			
<b>Bit number</b>	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
<b>Contents</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

When referring to the entire word, the digit numbered 0 is called the rightmost digit; the one numbered 3, the leftmost digit.

When inputting data, it must be input in the proper form for the intended purpose. This is no problem when designating bits, which are turned ON (equivalent to a binary value of 1) or OFF (a binary value of 0). When inputting word data, however, it is important to input it either as decimal or as hexadecimal, depending on what is called for by the instruction it is to be used for. *Section 5 Instruction Set* specifies when a particular form of data is required for an instruction.

**Converting Different Forms of Data**

Binary and hexadecimal can be easily converted back and forth because each four bits of a binary number is numerically equivalent to one digit of a hexadecimal number. The binary number 0101111101011111 is converted to hexadecimal by considering each set of four bits in order from the right. Binary 1111 is hexadecimal F; binary 0101 is hexadecimal 5. The hexadecimal equivalent would thus be 5F5F, or 24,415 in decimal ( $16^3 \times 5 + 16^2 \times 15 + 16 \times 5 + 15$ ).

Decimal and BCD are easily converted back and forth. In this case, each BCD digit (i.e., each group of four BCD bits) is numerically equivalent to the corresponding decimal digit. The BCD bits 0101011101010111 are converted to decimal by considering each four bits from the right. Binary 0101 is decimal 5; binary 0111 is decimal 7. The decimal equivalent would thus be 5,757. Note that this is not the same numeric value as the hexadecimal equivalent of 0101011101010111, which would be 5,757 hexadecimal, or 22,359 in decimal ( $16^3 \times 5 + 16^2 \times 7 + 16 \times 5 + 7$ ).

Because the numeric equivalent of each four BCD binary bits must be numerically equivalent to a decimal value, any four bit combination numerically greater than 9 cannot be used, e.g., 1011 is not allowed because it is numerically equivalent to 11, which cannot be expressed as a single digit in decimal notation. The binary bits 1011 are of course allowed in hexadecimal and are equivalent to the hexadecimal digit B.

There are instructions provided to convert data between BCD and hexadecimal. Refer to *5-15 Data Conversion* for details. Tables of binary equivalents to hexadecimal and BCD digits are provided in the appendices for reference.

**Decimal Points**

Decimal points are used in timers only. The least significant digit represents tenths of a second. All arithmetic instructions operate on integers only. When inputting data for use by Special I/O Units or other special applications, be sure to check on the type of data required for the application.

**Signed and Unsigned Data**

This section explains signed and unsigned binary data formats. Three instructions, MAX(165), MIN(166), and SUM(167), can use either signed or unsigned data.

**Unsigned binary**

Unsigned binary is the standard format used in OMRON PCs. Data in this manual are unsigned unless otherwise stated. Unsigned binary values are always positive and range from 0 (\$0000) to 65,535 (\$FFFF). Eight-digit values range from 0 (\$0000 0000) to 4,294,967,295 (\$FFFF FFFF).

Digit value	$16^3$				$16^2$				$16^1$				$16^0$			
Bit number	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Contents	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



**Signed Binary**

Signed binary data can have either a positive and negative value. The sign is indicated by the status of bit 15. If bit 15 is OFF, the number is positive and if bit 15 is ON, the number is negative. Positive signed binary values range from 0 (\$0000) to 32,767 (\$7FFF), and negative signed binary values range from -32,768 (\$8000) to -1 (\$FFFF).

Sign indicator	----- ----- ----- -----															
Digit value	16 <sup>3</sup>				16 <sup>2</sup>				16 <sup>1</sup>				16 <sup>0</sup>			
Bit number	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Contents	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Eight-digit positive values range from 0 (\$0000 0000) to 2,147,483,647 (\$7FFF FFFF), and eight-digit negative values range from -2,147,483,648 (\$8000 0000) to -1 (\$FFFF FFFF).

**Converting Decimal to Signed Binary**

Positive signed binary data is identical to unsigned binary data (up to 32,767) and can be converted using BIN(100). The following procedure converts negative decimal values between -32,768 and -1 to signed binary. In this example -12345 is converted to CFC7.

1. First take the absolute value (12345) and convert to unsigned binary:

Bit number	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Contents	0	0	1	1	0	0	0	0	0	0	1	1	1	0	0	1

2. Next take the complement:

Bit number	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Contents	1	1	0	0	1	1	1	1	1	1	0	0	0	1	1	0

3. Finally add one:

Bit number	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Contents	1	1	0	0	1	1	1	1	1	1	0	0	0	1	1	1

Reverse the procedure to convert negative signed binary data to decimal.

### 3-3 CIO (Core I/O) Area

CIO Area addresses run from words CIO 0000 through CIO 2555 and bits CIO 000000 through CIO 255515 and are divided into eight data areas. Five of these data areas are used to control I/O points and Special Units, and three data areas are used to manipulate and store data internally. The CIO Area is accessible either by bit or by word. No prefix is required when inputting data area addresses; the CIO prefix is used only for clarity in descriptions.

The name, range, and function of each data area within the CIO Area are summarized in the following table. internal I/O memory addresses are in parentheses.

Area	Range	Function
I/O Area	Words: CIO 0000 to CIO 0127 Bits: CIO 000000 to CIO 012715 (\$0000 to \$007F)	Allocated to I/O in the System and used to control I/O points. Bits not used to control I/O points can be used as work bits. The PC Setup can be used to control allocations.  Once I/O table has been registered, input bits are displayed on SSS with an I; output bits, with a Q.
Work Area	Words: CIO 0128 to CIO 0199 Bits: CIO 012800 to CIO 019915 (\$0080 to \$00C7)	These bits are used in the program to manipulate or to temporarily store data.
SYSMAC BUS/2 Area	Words: CIO 0200 to CIO 0999 Bits: CIO 020000 to CIO 099915 (\$00C8 to \$03E7)	These bits are used for remote I/O points in the SYSMAC BUS/2 Remote I/O System unless the default allocations are changed in the PC Setup.  Bits not used to control I/O points can be used as work bits.
Link Area	Words: CIO 1000 to CIO 1199 Bits: CIO 100000 to CIO 119915 (\$03E8 to \$04AF)	These bits are used for SYSMAC NET Link and SYSMAC LINK (Controller Link) Systems. Bits not used for data links can be used as work bits. These bits can be set as holding bits via PC Setup.
Holding Area	Words: CIO 1200 to CIO 1499 Bits: CIO 120000 to CIO 149915 (\$04B0 to \$05DB)	Used to store data and to retain the data values when the power is turned off.
CPU Bus Unit Area	Words: CIO 1500 to CIO 1899 Bits: CIO 150000 to CIO 189915 (\$05DC to \$076B)	Used to store the operating status of CPU Bus Units. Bits not used by CPU Bus Units can be used as work bits. These bits can be set as holding bits via the PC Setup.
CompoBus/D Areas	Words: CIO 1900 to CIO 1963 Bits: CIO 190000 to CIO 196315 (\$076C to \$0AB)  Words: CIO 2000 to CIO 2063 Bits: CIO 200000 to CIO 206315 (\$07D0 to \$080F)	These bits are used in CompoBus/D networks. Bits not used for CompoBus/D can be used as work bits.
Work Areas	Words: CIO 1964 to CIO 1999 Bits: CIO 196400 to CIO 199915 (\$07AC to \$07CF)  Words: CIO 2064 to CIO 2299 Bits: CIO 206400 to CIO 229915 (\$0810 to \$08FB)	These bits are used in the program to manipulate or to temporarily store data. These bits can be set as holding bits via the PC Setup.
SYSMAC BUS Area	Words: CIO 2300 to CIO 2555 Bits: CIO 230000 to CIO 255515 (\$08FC to \$09FB)	These bits are used for remote I/O points in the SYSMAC BUS Remote I/O System unless the default allocations are changed in the PC Setup.  Bits not used to control I/O points can be used as work bits. Up to word 2399 can be set as holding bits via the PC Setup.

### 3-3-1 I/O Area

The I/O Area is used as data to control I/O points. Those words that are used to control I/O points are called I/O words. Bits in I/O words are called I/O bits. I/O Area bits that are not allocated as I/O bits are reset when power is interrupted or PC operation is stopped. I/O Area word addresses run from CIO 0000 through CIO 0127 (I/O bits CIO 000000 through CIO 012715).

#### I/O Words

The maximum number of I/O bits is 16 (bits/word) times the number of I/O words, i.e., 2,048 bits. I/O bits are assigned to input or output points on Units connected at various locations in the PC System, as described later in this section (see *Word Allocations*).

If an I/O point on a Unit brings an input into the PC, the bit assigned to it is an input bit; if the point sends an output from the PC, the bit assigned to it is an output bit. To turn ON an output, the output bit assigned to it must be turned ON from the program or from a Programming Device. When an input turns ON, the input bit assigned to it also turns ON and the status of the input can be accessed indirectly by reading the status of the input bit assigned to it. Input status and control output status is thus manipulated through I/O bits.

After the I/O Table has been registered (see *Word Allocations*, below), an “I” will appear before input bit addresses and a “Q” will appear before output bit addresses on SSS (SYSMAC Support Software) displays.

I/O bits that are not assigned to I/O points can be used as work bits.

#### Input Bit Usage

Input bits record external signals input to the PC and can be used in any order in programming. Each input bit can also be used in as many instructions as required to achieve effective and proper control. They cannot be used as operands in instructions that control bit status, e.g., the OUTPUT, DIFFERENTIATE UP, and KEEP instructions. In other words, input bits should be treated as read-only bits.

#### Output Bit Usage

Output bits are used to output program execution results and can be used in any order in programming. Generally speaking, any one output bit should be used in only one instruction that controls its status, including OUT, KEEP(11), DIFU(13), DIFD(14), and SFT(10). If an output bit is used in more than one such instruction, only the status determined by the last instruction will actually be output from the PC during the normal I/O refresh period.

If you control the status of an output bit in more than one instruction, be sure to consider proper output timing and test the program before actual application. See 5-14-1 *SHIFT REGISTER – SFT(050)* for an example that uses an output bit in two “bit-control” instructions.

#### Word Allocations

I/O words in the CIO Area are allocated to Units mounted on Racks or otherwise connected to the PC by performing the I/O Table Registration operation. This operation creates in memory a table called an I/O table that records what words and how many words are allocated to the Units and whether these words are input or output words. The actual procedure for this operation is described in the *SSS Operation Manuals*.

The first word allocated to each Rack can be set with the SSS under the PC Setup. When the I/O Table Registration operation is performed, the system assigns word addresses to Units in the order in which they are mounted left to right on each Rack, beginning with the first word set in the PC Setup. The assigned words must be between CIO 0000 and CIO 0511.

For any Racks not assigned a first word in the PC Setup menu when the I/O Table is registered, the system automatically assigns word addresses to Units.

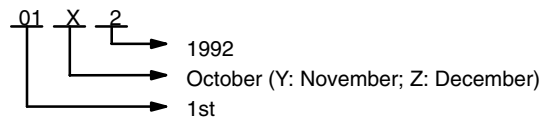
Word allocation begins with the leftmost Unit on the CPU Rack, and then continues left to right on the CPU Expansion Rack or Expansion I/O Rack with the lowest rack number set on its I/O Interface Unit. The order in which the Expansion I/O Racks are connected is not relevant in word allocation, only the rack numbers. I/O words start from CIO 0000 for the first Unit on the CPU Rack and continue consecutively: CIO 0001, CIO 0002, etc.

If the lowest word assigned to a Rack in the PC Setup menu is not higher than the total number of words required by Racks that aren't assigned a first word, the same word will be assigned to two Units and a duplication error will occur. A duplication error will also occur if words assigned to Racks overlap those assigned to Units controlled through Remote I/O Masters in the SYSMAC BUS/2 Area, which begins at CIO 0200. Be careful when setting areas from the SSS to avoid overlapping allocations.

There are no specific words associated with any particular slot because different Units can require a different number of words. Rather, each Unit is assigned the next word(s) following the word(s) assigned to the previous Unit. If there are any empty slots, no words will be assigned to those slots. Words are only assigned when a Unit is mounted; all empty slots are skipped. The numbers of I/O words allocated to the most common types of Unit are shown below.

Unit	Words required
16-pt I/O Units	1 word
24- or 32-pt I/O Units	2 words
64-pt I/O Units	4 words
Interrupt Input Unit (See note 8.)	1 word
Dummy I/O Unit	Set to 1, 2, or 4 words
Analog I/O Units	2 or 4 words
High-speed Counter Units	CT012/CT041: 2 words CT021: 2 or 4 words
MCR Units (See note 1.)	4 words
PID Unit (See notes 1 and 2.)	4 words
Position Control Units (See note 2.)	NC111/NC103/NC112/NC121: 4 words NC222: 2 words
I/O Interface Unit	None
Cam Positioner	2 or 4 words
Ladder Program I/O Unit	2 words
ASCII Unit (ASC03 not applicable; use ASC04.)	2 or 4 words
SYSMAC NET Link Unit	None (assigned CIO Link Area words)
SYSMAC LINK Unit	None (assigned CIO Link Area words)
Controller Link Unit	None (assigned CIO Link Area words)
SYSMAC BUS/2 Remote I/O Master Unit	None (See note 1.)
CompoBus/D Master Unit	None
BASIC Unit	None
Personal Computer Unit	None (See note 3.)
Motion Control Units	None
Temperature Control Data Link Unit	None
Ethernet Unit	None
Remote I/O Master Unit	None (See note 4.)
Remote I/O Slave Unit	None (See note 4.)
I/O Link Unit	1 or 2 words (See note 5.)
I/O Control Unit	None

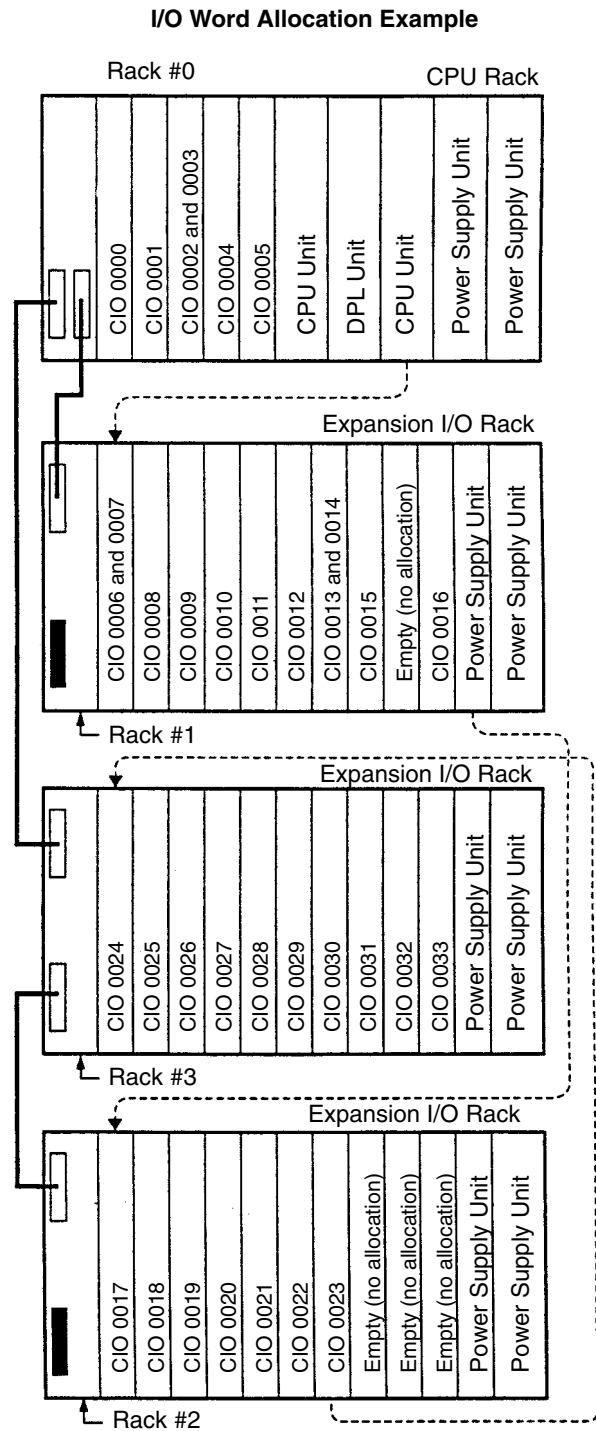
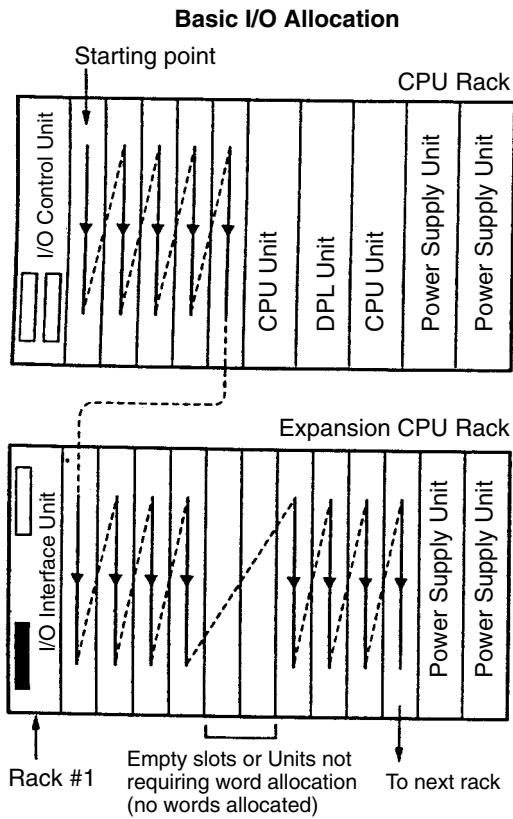
- Note**
1. PID Units, Magnetic Card Reader Units, Fuzzy Logic Units, and Cam Position Units cannot be mounted to Slave Racks in SYSMAC BUS/2 Systems.
  2. The PID Unit and some Position Control Units require two slots on a Rack.
  3. The Personal Computer Unit requires four slots on a Rack.
  4. Although no words are allocated to the Remote I/O Master and Slave Units themselves, words are allocated to Units mounted to Slave Racks or otherwise connected to the Remote I/O System. Refer to 3-3-3 SYSMAC BUS/2 Area and 3-3-8 SYSMAC BUS Area, for details.
  5. 3G2A5-LK010-E I/O Link Units and C500-ETL01 Teaching Tool cannot be set to 16 point input/16 point output on a CVM1, CVM1D, or CV-series PC.
  6. The I/O READ and I/O WRITE instructions (READ(190)/WRIT(191)) can be used for Units mounted to Slave Racks in SYSMAC BUS/2 Systems (but not in SYSMAC BUS Systems) under the following conditions.
    - a) The lot number of the Remote I/O Master Unit and Remote I/O Slave Unit must be the same as or latter than the following.



- b) The DIP switch on the Remote I/O Slave Unit must be set to “54MH.”
  - c) The Special I/O Unit must be one of the following: AD101, CT012, CT041, ASC04, IDS01-V1, IDS02, IDS21, IDS22, LDP01-V1, or NC222.
7. Refer to the *CVM1D Installation Guide* or to the operation manuals for individual Units for specific mounting procedures and limitations.
  8. Interrupt Input Units operate as normal 16-pt Input Units.

Once the word(s) assigned to a Unit has been determined, the use of individual bits in the word(s) is determined by the type of Unit. If the Unit is a Special I/O Unit, I/O Link Unit, or CPU Bus Unit, each bit will have a dedicated function. Refer to the *Operation Manuals* for the relevant Units for details.

With I/O Units, bits within a word are assigned to terminals starting at the top of the I/O Unit with bit 00 and going sequentially to the bottom. If the first Unit on the left of the CPU Rack is an Input Unit, the top terminals (i.e., the top input point) will be assigned CIO 000000, the next terminals, CIO 000001, and so forth for all of the terminals on the Unit. The allocation order is illustrated below. Arrows indicate the order in which words are allocated to Units for the rack number settings indicated.




### Rack Changes

Once Units have been mounted and the I/O Table Registration operation has been performed, a change to any Unit mounted to a Rack that affects the type of I/O word, or the number of words required by the Unit will cause an I/O verification error to occur. This includes adding Units to previously unused slots or removing Units that have already been allocated word(s). A Unit can, however, be replaced with another Unit that requires the same number of input words and the same number of output words without generating an I/O verification error. Dummy I/O Units are available to fill slots for future use or to replace Units that are no longer needed (see *Word Reservations*, below).

There are two ways, however, to change the I/O table registered in memory. One is to allocate words to a slot that is not currently being used. This method is described below in *Word Reservations*.

The other way is to perform the I/O Table Registration operation again. When this is done, all I/O words will be reallocated according to the Units mounted to the Racks at the time. If the number of words allocated to any one slot changes, all word allocations past that slot will also change, requiring that the program be changed to allow for this.

Sometimes program changes can be avoided when a Unit is removed from a Rack or you know that you are going to have to add a Unit later by reserving words. Although designed to enable slot reservations for future use, a slot reservation can be left permanently to prevent what could be extensive program changes.

 **Caution** Always be sure to change word and bit addresses in the program whenever a change to Units on a Rack affects word allocations. Failure to do so may cause improper I/O operations.

### Word Reservations

Words can be reserved at a certain slot for future use either by mounting a Dummy I/O Unit to the slot before performing the I/O Table Registration operation or by performing an I/O Table Change operation after performing the I/O Table Registration operation.

A Dummy I/O Unit provides settings to designate word types (input or output) and length (one, two, or four words). After I/O Table Generation has been performed and a Dummy I/O Unit has been allocated the words designated by these settings, it can be replaced at any time with a Unit that requires the same type and number of words, e.g., if a Dummy I/O Unit is set for two input words, it can be replaced with any 24- or 32-point Input Unit or any other Unit that requires two input words.

Once an I/O table has been registered, it can be changed using the I/O Table Change operation described in *SSS Operation Manuals*. This operation can be used to reserve up to four input words, output words, or non-defined words at a time. The I/O Table Change operation must be performed after the I/O Table Registration operation. If I/O Table Registration is repeated, all word reservations will be cancelled, and I/O Table Change will have to be repeated.

### 3-3-2 Work Areas

There are three Work Areas available in PC memory. Words and bits in the Work Areas can be used in programming as required to control other bits, but are not used for direct external I/O. Other bits and words in the CIO Area which are not being used for their intended purpose can also be used as work words and work bits. Actual application of work bits and work words is described in *Section 4 Writing Programs*.

Work words and bits are reset when power is interrupted or PC operation is stopped, but they are not reset when a FALS error instruction is executed in the program.

Work words	Work bits
CIO 0128 to CIO 0199 (72 words)	CIO 012800 to CIO 019915 (1,152 bits)
CIO 1964 to CIO 1999 (36 words)	CIO 196400 to CIO 199915 (576 bits)
CIO 2064 to CIO 2299 (236 words)	CIO 206400 to CIO 229915 (3,776 bits)

### 3-3-3 SYSMAC BUS/2 Area

I/O bits allocated in the SYSMAC BUS/2 Area correspond to I/O points on I/O Terminals (group-1 and group-2 Slaves), Units mounted to Slave Racks (group-3 Slaves), or other Units connected to SYSMAC BUS/2 Remote I/O Master Units (RM/2). Up to four Masters can be connected. The total number of I/O points required for I/O Terminals, Units on Slave Racks, and other Units in the SYSMAC BUS/2 Remote I/O System must not exceed 2,048 (128 words).

SYSMAC BUS/2 Area address allocation can be customized with the PC Setup using the SSS. The first word allocated to the group-1, group-2, and group-3 Slaves, as well as the size of each of these areas, can be changed. The following table shows the default address allocations.

RM/2 #	Group-1 Slaves*	Group-2 Slaves*	Group-3 Slaves
0	CIO 0200 to CIO 0249	CIO 0250 to CIO 0299	CIO 0300 to CIO 0399
1	CIO 0400 to CIO 0449	CIO 0450 to CIO 0499	CIO 0500 to CIO 0599
2	CIO 0600 to CIO 0649	CIO 0650 to CIO 0699	CIO 0700 to CIO 0799
3	CIO 0800 to CIO 0849	CIO 0850 to CIO 0899	CIO 0900 to CIO 0999

\*Group-1 Slaves allocated up to 64 I/O points. Group-2 Slaves are medium-sized Units allocated up to 128 I/O points. Group-3 Slaves are used to form Slave Racks.

As with I/O area allocations to CPU, Expansion CPU, and Expansion I/O Racks, word allocation begins with the Slaves connected to the Master with the lowest unit number, RM/2 #0, regardless of the order that the Masters are mounted. Likewise, word allocation to Units connected to RM/2 #0 begins with the Slaves that have the lowest unit numbers, regardless of the order that the Slaves are mounted.

Up to 8 Slave Racks can be connected to each RM/2 Master. Word addresses are assigned to Units on Slave Racks in the order in which they are mounted left to right. Refer to the *SYSMAC BUS/2 Remote I/O System Manual* for details on word allocation to Slaves and Units on Slave Racks.

After the I/O Table has been registered or edited, an “I” will appear before input bit addresses and a “Q” will appear before output bit addresses on SSS displays. Refer to the *SSS Operation Manuals* for details on the PC Setup.

### 3-3-4 Link Area

The Link Area is used as a common data area to automatically transfer information between PCs. This data transfer is achieved through data links in either a SYSMAC LINK System, SYSMAC NET Link System, or a Controller Link System. Link Area addresses run from CIO 1000 through CIO 1199. Link Area words CIO 1000 through CIO 1063 and DM Area words D00000 through D00127 are automatically used for data link tables (in a SYSMAC LINK or SYSMAC NET Link System) unless specific link words are designated. Allocations can be designated from the SSS. Refer to the *SSS Operation Manuals*, and the *SYSMAC LINK System Manual*, *SYSMAC NET Link System Manual*, or *Controller Link Unit Operation Manual* for details.

### 3-3-5 Holding Area

The Holding Area is used to store/manipulate various kinds of data and can be accessed either by word or by bit. Holding Area bits can be used in any order required and can be programmed as often as required.

The default Holding Area word addresses range from CIO 1200 through CIO 1499; bit addresses, from CIO 120000 through CIO 149915. The range of the Holding Area can be changed to any size between CIO 1000 through CIO 2399 with the PC Setup from the SSS. If the Holding Area is increased, it will overlap other areas. An “H” will appear before Holding Area bit addresses on the SSS screen. Refer to the *SSS Operation Manuals* for details.



The Holding Area retains status when the operating mode is changed, power is interrupted, or PC operation is stopped.

Holding Area bits and words can be used to preserve data whenever PC operation is stopped. Holding bits also have various special applications, such as creating latching relays with the KEEP instruction and forming self-holding outputs. These are discussed in *Section 4 Writing Programs* and *Section 5 Instruction Set*.

### 3-3-6 CPU Bus Unit Area

Two types of external bus are provided for CVM1, CVM1D, and CV-series PCs: the high-speed CPU bus (S Bus) and the I/O bus. Units that connect to the CPU bus on the CPU or Expansion CPU Rack are called CPU Bus Units and include the SYSMAC NET Link Unit, SYSMAC LINK Unit, Controller Link Unit, Host Link Unit, CompoBus/D Master Unit, SYSMAC BUS/2 Remote I/O Master Unit, BASIC Unit, and Personal Computer Unit.

CPU Bus Unit Area addresses range from CIO 1500 through CIO 1899. These 400 words are divided into 16 groups of 25 words each. These are allocated to CPU Bus Units according to their unit number settings as shown in the following tables.

Unit #	0	1	2	3	4	5	6	7
CIO words	1500	1525	1550	1575	1600	1625	1650	1675
	to 1524	to 1549	to 1574	to 1599	to 1624	to 1649	to 1674	to 1699

Unit #	8	9	10	11	12	13	14	15
CIO words	1700	1725	1750	1775	1800	1825	1850	1875
	to 1724	to 1749	to 1774	to 1799	to 1824	to 1849	to 1874	to 1899

An additional 1600 words in the DM Area (D02000 to D03599) are provided for CPU Bus Units. The particular function of words allocated to the Unit depends on the CPU Bus Unit being used.

### 3-3-7 CompoBus/D Areas

I/O bits allocated to CompoBus/D correspond to external I/O points on the devices connected to the CompoBus/D device network. Refer to *CompoBus/D (DeviceNet) Operation Manual (W267)* for further information.

Words	Bits
CIO 1900 to CIO 1963 (64 words)	CIO 190000 to CIO 196315 (1,024 bits)
CIO 2000 to CIO 2063 (64 words)	CIO 200000 to CIO 206315 (1,024 bits)

**Note** Bits not used for CompoBus/D can be used as work bits.

### 3-3-8 SYSMAC BUS Area

I/O bits allocated in the SYSMAC BUS Area correspond to external I/O points on I/O Terminals, Optical I/O Units, or I/O Units mounted to Slave Racks that are connected to SYSMAC BUS Remote I/O Master Units (RM). Up to 8 Masters can be connected to the PC and the total number of I/O points in the SYSMAC BUS System must not exceed 2,048 (128 words).

Unit numbers are assigned to Masters automatically when the I/O Table is registered or edited, according to the order in which the Masters are mounted (taking into account rack number settings). The first word allocated to each Master can be changed with the PC Setup using the SSS.

SYSMAC BUS Area addresses range from CIO 2300 through CIO 2555. These 256 words are divided into 8 groups of 32 words each and are allocated to Masters according their number setting. The following table shows the default address allocation.

RM #	0	1	2	3	4	5	6	7
<b>CIO words</b>	2300 to 2331	2332 to 2363	2364 to 2395	2396 to 2427	2428 to 2459	2460 to 2491	2492 to 2523	2524 to 2555

Words are allocated to Units on Slave Racks in order beginning with the Slave Rack with the lowest unit number. Up to 8 Slave Racks can be connected to each Master. Word addresses are assigned to Units in the first Slave Rack in the order in which they are mounted left to right. Word allocation then continues left to right on the Slave Rack with the next lowest unit number, and so on until words have been allocated to all of the Slave Racks.

Words are allocated to I/O Terminals and Optical I/O Units according to word settings on the Unit. The word allocated is calculated by adding the first word of the Master and the word setting on the Unit. To minimize the chance of overlapping with words allocated to Slave Racks, it is recommended to set I/O Terminal and Optical I/O Unit settings beginning from 31, the last word allocated to the Master, and continuing down to lower settings.

Refer to the *SYSMAC BUS Remote I/O System Manual* for details on word allocation to I/O Terminals and Slave Racks.

After the I/O Table has been registered or edited, an “I” will appear before input bit addresses and a “Q” will appear before output bit addresses on SSS displays. Refer to the *SSS Operation Manuals* for details on the PC Setup.

### 3-4 TR (Temporary Relay) Area

The TR Area provides eight bits that are used only with the LD and OUT instructions to enable certain types of branching ladder diagram programming. It is only necessary to use TR bits when entering the program using mnemonic code. The SSS enters TR bits automatically, although the TR bits are not shown on the SSS screen. The use of TR bits is described in *Section 4 Writing Programs*.

TR addresses range from TR0 though TR7. Each of these bits can be used as many times as required and in any order required as long as the same TR bit is not used twice in the same instruction block.

### 3-5 CPU Bus Link Area

The CPU Bus Link Area is indicated by a G prefix. Addresses range from G000 to G255. The CPU Bus Link Area can be divided into 3 sections, the PC Status Area, Clock/Calendar Area, and Data Link Area.

G000 is the PC Status Area and contains flags and control bits relating to PC status. G001 to G004 are the Clock/Calendar Area, and G005 to G007 are not used.

Most of the CPU Bus Link Area (G008 to G255) is taken up by the Data Link Area which is used to transfer information between CPU Bus Units and the CPU. CPU Bus Units connect to the CPU bus on the CPU Rack or Expansion CPU Rack.



**Caution**

The CPU Bus Link Area words G000 through G007 cannot be written to from the user program and can only be read from to access the data provided there.

**PC Status Area**

The following table shows the specific functions of flags and control bits in the PC Status Area, G000.

G000 bit(s)	Function
00	ON when the PC is in PROGRAM mode.
01	ON when the PC is in DEBUG mode.
02	ON when the PC is in MONITOR mode.
03	ON when the PC is in RUN mode.
04	ON when the program is being executed (RUN or MONITOR mode).
05	Not used.
06	ON when a non-fatal error has occurred. (PC operation continues.)
07	ON when a fatal error has occurred. (PC stops.)
08	<p>SPL/DPL Mode Flag    OFF when operating in simplex mode                                                ON when operating in duplex mode</p> <p>The following diagram shows an example using G00008 (SPL/DPL Mode Flag) and A39115 (Duplex Initializing Wait Flag).</p> <p>G00008: OFF when operating in simplex mode, ON when operating in duplex mode, and OFF during duplex initialization.                      A39115: ON during duplex initialization.                      G00008 turns OFF during duplex initialization, so duplex errors cannot be detected independently. Use G00008 together with A39115.                      Duplex initializing is performed when writing during online edit operations, pressing the Initialize Switch Button, and processing communications commands or Programming Device commands.                      The duplex initializing time depends on the cycle time, mostly basically regardless of the UM size or EM size, and is calculated as the cycle time multiplied by 390.</p>
09	Active CPU Unit Flag    OFF when the CPU Unit on the left is active. ON when the CPU Unit on the right is active.
10	Not used.
11	UM Protect Bit. Prevents both reading out and writing to Program Memory when turned ON. Set with the SSS.
12	Memory Card Protect Bit. Prevents writing to Memory Cards when turned ON. Set with the Memory Card Protect Switch.
13 and 14	Not used.
15	UM Protect Bit. Prevents writing to Program Memory when turned ON. Set with the System Protect Key Switch.

**Calendar/Clock Area**

The following table shows the function of bits in the Calendar/Clock Area, G001 to G004. The clock is set with the SSS. Refer to the *SSS Operation Manuals* for more details.

Word	Bits	Contents	Possible values
G001	00 to 07	Seconds	00 to 59
	08 to 15	Minutes	00 to 59
G002	00 to 07	Hours	00 to 23 (24-hour system)
	08 to 15	Day of month	01 to 31 (adjusted by month and for leap year)
G003	00 to 07	Month	1 to 12
	08 to 15	Year	00 to 99 (Rightmost two digits of year)
G004	00 to 07	Day of week	00 to 06 (00: Sun.; 01: Mon.; 02: Tues.; 03: Wed.; 04: Thurs.; 05: Fri.; 06: Sat.)

**Note** The accuracy of the internal clock depends on the ambient temperature. Refer to the following table.

Ambient temperature	Error per month
55°C	-3 to 0 min
25°C	±1 min
0°C	-2 to 0 min

**Data Link Area**

The CPU Bus Link Area is disabled by default in the PC Setup and must be enabled with the SSS in order to use the Data Link Area.

The 120 words of CPU Bus Link Area from G008 to G127 are used for outputs from the CPU to BASIC Units. The 128 words from G128 to G255 are used for outputs from the BASIC Units. These are divided into 16 groups of 8 words each and allocated to CPU Bus Units according their unit number settings as shown in the following tables. All words not output by a particular BASIC Unit are read by it as inputs from the other BASIC Units.

Unit #	0	1	2	3	4	5	6	7
<b>Words</b>	G128 to G135	G136 to G143	G144 to G151	G152 to G159	G160 to G167	G168 to G175	G176 to G183	G184 to G191

Unit #	8	9	10	11	12	13	14	15
<b>Words</b>	G192 to G199	G200 to G207	G208 to G215	G216 to G223	G224 to G231	G232 to G239	G240 to G247	G248 to G255

When the PC Setup have been changed to enable the CPU Bus Link, bit 15 of the first word allocated to each Unit (e.g., bit G12815 for Unit #0) will be OFF during data reception.

### 3-6 Auxiliary Area

The Auxiliary Area contains flags and control bits used for monitoring and controlling PC operation, accessing clock pulses, and signalling errors. Auxiliary Area word addresses range from A000 through A511; bit addresses, from A00000 through A51115. Addresses A000 through A255 are read/write, but addresses A256 through A511 are read only.

The Force Set/Reset operations from the SSS behave like the SET(016) and RSET(017) instructions when applied to words A000 through A255.

Unused Auxiliary Area words and bits **cannot** be used as work words and bits.



**Caution**

The Auxiliary Area contains two sections. The section between A000 and A255 can be read from or written to from the user program. The section between A256 and A511, however, can be read from to access the data provided there, but it cannot be written to from the user program.

The following table lists the functions of Auxiliary Area flags and control bits. Most of these bits are described in more detail following the table. Descriptions are in order by address, except that some bits/words with related functions are explained together.

Word(s)	Bit(s)	Function
A000	00 to 11	Not used.
	12	IOM Hold Bit
	13	Forced Status Hold Bit
	14	Error Log Reset Bit
	15	Output OFF Bit
A001	00 to 15	CPU Bus Unit Restart Bits
A002 to A004	00 to 15	Not used.

Word(s)	Bit(s)	Function
A005	00 to 07	SYSMAC BUS Error Check Bits
	08 to 15	Not used.
A006 and A007	00 to 15	Not used.
A008 (See note 1.)	00 to 06	Not used.
	07	Stop Monitor Flag
	08	Execution Time Measured Flag
	09	Differentiate Monitor Completed Flag
	10	Stop Monitor Completed Flag
	11	Trace Trigger Monitor Flag
	12	Trace Completed Flag
	13	Trace Busy Flag
	14	Trace Start Bit
15	Sampling Start Bit	
A009	00 to 15	Not used.
A010 to A011 (See note 1.)	00 to 15	Startup Time (BCD)
A012 to A013 (See note 1.)	00 to 15	Power Interruption Time (BCD)
A014 (See note 1.)	00 to 15	Number of Power Interruptions (BCD)
A015 (See note 1.)	00 to 15	CPU Bus Service Disable Bits
A016	00 to 15	Not used.
A017	00 to 02	Not used.
	03	Host Link Service Disable Bit
	04	Programming Device Service Disable Bit
	05	SYSMAC BUS Refresh Disable Bit
	06 to 15	Not used.
A018 to A089	00 to 15	Not used.
A090 to A097	00 to 15	Reserved for system use
A098	00	FPD(177) Teaching Bit
	01 to 15	Not used.
A099	00 to 07	Message #0 to #7 Flags
	08 to 15	Not used.
A100 to A199	00 to 15	Error Log Area (20 × 5 words)
A200 to A203	00 to 15	Macro area inputs
A204 to A207	00 to 15	Macro area outputs
A208 to A255	00 to 15	Not used. (End of Read/Write Area)
A256 to A299	00 to 15	Not used. (Beginning of Read-only Area)
A300	00 to 15	Error Log Pointer (binary)
A301	00 to 15	Not used.
A302	00 to 15	CPU Bus Unit Initializing Flags
A303 to A305	00 to 15	Not used.

Word(s)	Bit(s)	Function
A306	00	Start Input Wait Flag
	01	I/O Verification Error Wait Flag
	02	SYSMAC BUS Terminator Wait Flag
	03	CPU Bus Unit Initializing Wait Flag
	04	Duplex Bus Error At Startup Flag
	05	Duplex Verification Error At Startup Flag
	06	Duplex Alternate CPU Waiting At Startup Flag
	07 to 14	Not used.
	15	Programming Device Connected Flag
A307	00 to 07	Programming Device Connected Flags for RT #0 to RT #7 of RM/2 #0
	08 to 15	Programming Device Connected Flags for RT #0 to RT #7 of RM/2 #1
A308	00 to 07	Programming Device Connected Flags for RT #0 to RT #7 of RM/2 #2
	08 to 15	Programming Device Connected Flags for RT #0 to RT #7 of RM/2 #3
A309	00 to 15	Programming Device Cycle Time (binary)
A310 to A325	00 to 15	CPU Bus Unit Service Interval (binary)
A326 to A342	00 to 15	Not used.
A343 (See note 1.)	00 to 02	Memory Card Type
	03 to 06	Not used.
	07	Memory Card Format Error Flag
	08 to 13	Not used.
	14	Accessing Memory Card Flag
	15	Memory Card Protected Flag
A344 to A390	---	Not used.
A391	00 to 14	Not used.
	15	Duplex Initializing Wait Flag (See note 2.)
A392 to 393	---	Not used.
A394	00 to 08	Not used.
	09	Duplex CPU Replacement Enabled Flag
	10	Duplex CPU Error Flag
	11	Duplex CPU Memory Error Flag
	12 to 15	Not used.
A395	00 to 15	Not used.
A396	00 to 11	Not used.
	12	Duplex Verification Error: EM Size Mismatch
	13	Duplex Verification Error: PC Setup Mismatch
	14	Duplex Verification Error: PC Setup Transfer Error
	15	Duplex Verification Error: Program Mismatch
A397	00	CPU Rack Right PSU OFF Flag
	01	CPU Rack Left PSU OFF Flag
	02 to 15	Not used.

Word(s)	Bit(s)	Function
A398	00	CPU Rack Right PSU Error Flag
	01	CPU Rack Left PSU Error Flag
	02	Rack #1 Right PSU Error Flag
	03	Rack #1 Left PSU Error Flag
	04	Rack #2 Right PSU Error Flag
	05	Rack #2 Left PSU Error Flag
	06	Rack #3 Right PSU Error Flag
	07	Rack #3 Left PSU Error Flag
	08	Rack #4 Right PSU Error Flag
	09	Rack #4 Left PSU Error Flag
	10	Rack #5 Right PSU Error Flag
	11	Rack #5 Left PSU Error Flag
	12	Rack #6 Right PSU Error Flag
	13	Rack #6 Left PSU Error Flag
	14	Rack #7 Right PSU Error Flag
	15	Rack #7 Left PSU Error Flag
A399	00 to 07	Slot Undergoing Online I/O Replacement (2-digit BCD)
	08 to 11	Rack Undergoing Online I/O Replacement (1-digit BCD)
	12 to 13	Not used.
	14	Online I/O Replacement Status Retention Flag
	15	Online I/O Replacement In Progress Flag
A400	00 to 15	Error Code
A401	00 to 05	Not used.
	06	FALS Error Flag
	07	Not used.
	08	Cycle Time Too Long Flag
	09	Program Error Flag
	10	I/O Setting Error Flag
	11	Too Many I/O Points Flag
	12	CPU Bus Error Flag
	13	Duplication Error Flag
	14	I/O Bus Error Flag
15	Memory Error Flag	
A402	00	Duplex Verification Error Flag
	01	Duplex Bus Error Flag
	02	Power Interruption Flag
	03	CPU Bus Unit Setting Error Flag
	04	Battery Low Flag
	05	SYSMAC BUS Error Flag
	06	SYSMAC BUS/2 Error Flag
	07	CPU Bus Unit Error Flag
	08	Not used.
	09	I/O Verification Error Flag
	10 to 11	Not used.
	12	Indirect DM Error Flag
	13	Jump Error Flag
	14	Duplex Power Supply Error Flag
15	FAL Error Flag	

Word(s)	Bit(s)	Function
A403	00 to 08	Memory Error Area Location
	09	Memory Card Startup Transfer Error Flag
	10 to 15	Not used.
A404	00 to 07	I/O Bus Error Slot Number (BCD)
	08 to 15	I/O Bus Error Rack Number (BCD)
A405	00 to 15	CPU Bus Unit Error Unit Number
A406	00 to 15	Not used.
A407	00 to 15	Total I/O Words on CPU and Expansion Racks (BCD)
A408	00 to 15	Total SYSMAC BUS/2 I/O Words (BCD)
A409	00 to 07	Duplicate Rack Number
	08 to 14	Not used
	15	Duplicate System Parameter Words Flag
A410	00 to 15	CPU Bus Unit Duplicate Number
A411 to A418	00 to 15	Not used.
A419	00 to 07	CPU-recognized Rack Numbers
	08 to 15	Not used.
A420 to A421	00 to 15	Not used.
A422	00 to 15	CPU Bus Unit Error Unit Number
A423	00 to 13	Not used.
	14	CPU Bus Unit Number Setting Error Flag
	15	CPU Bus Link Error Flag
A424	00 to 03	SYSMAC BUS/2 Error Master Number
	04 to 15	Not used.
A425	00 to 07	SYSMAC BUS Error Master Number
	08 to 15	Not used.
A426	00 to 09	Not used.
	10	Memory Card Battery Error in Standby CPU
	11	CPU Battery Error in Standby CPU
	12	Memory Card Battery Error in Active CPU
	13	CPU Battery Error in Active CPU
	14	Memory Card Battery Low Flag
	15	PC Battery Low Flag
A427	00 to 15	CPU Bus Unit Setting Error Unit Number
A428 to A429	00 to 15	Not used.
A430 to A461	00 to 15	Executed FAL Number
A462 to A463	00 to 15	Maximum Cycle Time (BCD, 8 digits)
A464 to A465	00 to 15	Present Cycle Time (BCD, 8 digits)
A466 to A469	00 to 15	Not used.
A470 to A477	00 to 15	SYSMAC BUS Error Codes: RM # 0 (A470)    RM #1 (A471) RM # 2 (A472)    RM #3 (A473) RM # 4 (A474)    RM #5 (A475) RM # 6 (A476)    RM #7 (A477)
A478	00 to 15	Total SYSMAC BUS I/O Words (BCD)
A479	00 to 15	Not used.
A480 to A499	00 to 15	SYSMAC BUS/2 Error Unit Number: RM # 0 (A480 to A484)    RM #1 (A485 to A489) RM # 2 (A490 to A494)    RM #3 (A495 to A499)



Word(s)	Bit(s)	Function
A500	00 to 02	Not used.
	03	Instruction Execution Error Flag
	04	Carry Flag
	05	Greater Than Flag
	06	Equals Flag
	07	Less Than Flag
	08	Negative Flag
	09	Overflow Flag
	10	Underflow Flag
	11	Not used.
	12	First Cycle Flag when one-step operation is started with STEP instruction
	13	Always ON Flag
	14	Always OFF Flag
	15	First Cycle Flag
	A501	00
01		0.2-s Clock Pulse
02		1.0-s Clock Pulse
03		0.02-s Clock Pulse
04 to 15		Not used.
A502	00 to 07	Port #0 to #7 Enabled Flags
	08 to 15	Port #0 to #7 Execute Error Flags
A503 to A510	00 to 15	Port #0 to #7 Completion Codes
A511	00 to 04	Current EM Bank (0 to 7)
	05 to 14	Not used.
	15	EM Installed Flag

- Note**
1. During duplex operation, the status of A008 through A015 will not agree in the active and standby CPU Units.
  2. The Duplex Initializing Wait Flag is ON during duplex initialization. For details about duplex initialization, refer to *6-2-2 Duplex Initialization*.
  3. Do not use A50013 (Always ON Flag), A50014 (Always OFF Flag), or A50015 (First Cycle Flag) to control execution of differentiated instructions. The instructions will never be executed.

### 3-6-1 IOM Hold Bit

Bit A00012 can be turned ON to preserve the status of the CIO Area, Transition Flags, Timer Flags, Timer PVs, index registers, data registers, and the Current EM Bank Number when shifting from PROGRAM or DEBUG to MONITOR or RUN mode or when shifting from MONITOR or RUN mode to PROGRAM or DEBUG mode. (I/O Memory includes the CIO Area, TR Area, CPU Bus Link Area, Auxiliary Area, Transition Flags, Step Flags, Timer Completion Flags, and Counter Completion Flags.)

When the IOM Hold Bit is OFF, the CIO Area, Transition Flags, Timer Flags, Timer PVs, index registers, data registers, and the Current EM Bank Number are cleared when switching between these modes.

If the IOM Hold Bit is ON, and the status of the IOM Hold Bit itself is preserved in the PC Setup (Setting B, IOM Hold Bit status), then I/O Memory is also preserved when the PC is turned ON or power is interrupted.

### 3-6-2 Forced Status Hold Bit

Bit A00013 can be turned ON to preserve the status of bits that have been force-set or force-reset when switching modes (except RUN mode). When the Forced

Status Hold Bit is OFF, bits that have been force-set or force-reset will return to default status when switching between modes.

If the Forced Status Hold Bit is ON, and the status of the Forced Status Hold Bit itself is preserved in the PC Setup (Setting B, Forced Status Hold Bit status), then the status of bits that have been force-set or force-reset is also preserved when the PC is turned ON or power is interrupted.

In any case, bits that have been force-set or force-reset will return to default status when switching to RUN mode.

### 3-6-3 Error Log Reset Bit

Bit A00014 can be turned ON to clear the contents of the Error Log Area (words A100 to A199), and reset the Error Record Pointer to 0. The Error Log Reset Bit is automatically turned OFF after the Error Log Area is cleared.

### 3-6-4 Output OFF Bit

Bit A00015 can be turned ON to turn OFF all outputs from the PC. The OUT INH. indicator on the front panel of the CPU will light.

### 3-6-5 CPU Bus Unit Restart Bits

Bits A00100 through A00115 can be turned ON to reset CPU Bus Units number #0 through #15, respectively. The Restart Bits are turned OFF automatically when restarting is completed.

Do not turn these bits ON and OFF in the program; manipulate them from the SSS.

### 3-6-6 SYSMAC BUS Error Check Bits

Bits A00500 through A00507 can be turned ON to read out the error codes (stored in words A470 through A477) for Masters numbered #0 through #7, respectively. The Error Check Bits are turned OFF automatically after the information has been read out. Refer to 3-6-39 SYSMAC BUS Error Flag for more details.

### 3-6-7 SSS Flags

Word A008 contains flags that indicate the status of commands and instructions performed with the SSS.

**Note** During duplex operation, the status of the SSS Flags in the active CPU Unit will not agree with the same flags in the standby CPU Unit.

<b>Stop Monitor Flag (A00807)</b>	Bit A00807 is turned ON when the Stop Monitor is used from the SSS, and is turned OFF when it is completed.
<b>Execution Time Measured Flag (A00808)</b>	Bit A00808 is turned ON when the execution time has been measured with MARK(174) instructions with the SSS.
<b>Differentiate Monitor Completed Flag (A00809)</b>	Bit A00809 is turned ON when the differentiate monitor condition has been established with the SSS.
<b>Stop Monitor Completed Flag (A00810)</b>	Bit A00810 is turned ON when the Stop Monitor operation has been completed with the SSS.
<b>Trace Trigger Monitor Flag (A00811)</b>	Bit A00811 is turned ON when one of the trigger conditions has been established during execution of a Data or Program Trace with the SSS.
<b>Trace Completed Flag (A00812)</b>	Bit A00812 is turned ON upon when the sampling of a region of trace memory has been completed during execution of a Data or Program Trace with the SSS.

- Trace Busy Flag (A00813)** Bit A00813 is turned ON when a Data or Program Trace is executed with the SSS, and is turned OFF when it is completed.
- Trace Start Bit (A00814)** The Trigger conditions are established when bit A00814 is turned ON by one of trigger conditions of a Data or Program Trace of the SSS.
- Sampling Start Bit (A00815)** Bit A00815 is turned ON to start a Data Trace.

### 3-6-8 Start-up Time

Words A010 and A011 contain the start-up time, in BCD format, as shown in the following table. The start-up time is updated every time the power is turned ON.

Word	Bits	Contents	Possible values
A010	00 to 07	Seconds	00 to 59
	08 to 15	Minutes	00 to 59
A011	00 to 07	Hours	00 to 23 (24-hour system)
	08 to 15	Day of month	01 to 31 (adjusted by month and for leap year)

**Note** During duplex operation, the start-up time in the active CPU Unit will not agree with the start-up time in the standby CPU Unit.

### 3-6-9 Power Interruption Time

Words A012 and A013 contain, in BCD format, the time at which power was interrupted, as shown in the following table. The power interruption time is updated every time the power is interrupted.

Word	Bits	Contents	Possible values
A012	00 to 07	Seconds	00 to 59
	08 to 15	Minutes	00 to 59
A013	00 to 07	Hours	00 to 23 (24-hour system)
	08 to 15	Day of month	01 to 31 (adjusted by month and for leap year)

**Note** During duplex operation, the power interruption time in the active CPU Unit will not agree with the power interruption time in the standby CPU Unit.

### 3-6-10 Number of Power Interruptions

Word A014 contains the number of times that power has been interrupted since the PC was first turned on. The number is in 4-digit BCD, and can be reset by writing #0000 to word A014.

**Note** During duplex operation, the number of power interruptions in the active CPU Unit will not agree with the number in the standby CPU Unit.

### 3-6-11 Service Disable Bits

Words A015 and A017 contain control bits that disable I/O servicing to certain Units and periodic refreshing. Turn these bits ON and OFF in the program. The service disable bits are automatically turned OFF when power is turned on or PC operation is stopped.

- CPU Service Disable Bits** Bits A01500 through A01515 can be turned ON to stop service to CPU Bus Units numbered #0 through #15, respectively. Turn the appropriate bit OFF again to resume service to the CPU Bus Unit.

**Note** During duplex operation, the status of the CPU Service Disable Bits in the active CPU Unit will not agree with the same bits in the standby CPU Unit.

- Host Link Service Disable Bit**      Bit A01703 can be turned ON to stop Host Link System servicing. Turn OFF again to resume service to the Host Link System.
  
- Programming Device Service Disable Bit**      Bit A01704 can be turned ON to stop service to Programming Devices. Turn OFF again to resume service to Programming Devices.
  
- SYSMAC BUS Refresh Disable Bit**      Bit A01705 can be turned ON to stop SYSMAC BUS refreshing. Turn OFF again to resume SYSMAC BUS refreshing.

### 3-6-12 Message Flags

When the MESSAGE instruction (MSG(195)) is executed, the bit in A099 corresponding to the message number is turned ON. Bits 00 through 07 correspond to message numbers 0 through 7, respectively.

### 3-6-13 Error Log Area

Words A100 through A199 contain up to 20 records that show the nature, time, and date of errors that have occurred in the PC. The Error Log Area will store system-generated or FAL(006)/FALS(007)-generated error codes. Refer to *Section 6 Troubleshooting* in the *CVM1D PCs Installation Guide* for details on error codes.

The Error Log Area can be moved to the DM or EM Areas and its size can be increased to store up to 2,047 records with the PC Setup.

#### Area Structure

With the default PC Setup, error records occupy five words each stored between words A100 and A199. The last record that was stored can be obtained via the content of word A300 (Error Record Pointer). The record number, Auxiliary Area words, and pointer value for each of the twenty records are as follows:

Record	Addresses	Pointer value*
None	N.A.	0000
1	A100 to A104	0001
2	A105 to A109	0002
3	A110 to A114	0003
4	A115 to A119	0004
5	A120 to A124	0005
6	A125 to A129	0006
7	A130 to A134	0007
8	A135 to A139	0008
9	A140 to A144	0009
10	A145 to A149	000A
11	A150 to A154	000B
12	A155 to A159	000C
13	A160 to A164	000D
14	A165 to A169	000E
15	A170 to A174	000F
16	A175 to A179	0010
17	A180 to A184	0011
18	A185 to A189	0012
19	A190 to A194	0013
20	A195 to A199	0014

\*The pointer value is in word A300, which is in the read-only area (words A256 to A511).

Although each of them contains a different record, the structure of each record is the same: the first word contains the error code; the second word, the error con-

tents, and the third, fourth, and fifth words, the time, day, and date. The error code will be either one generated by the system or by FAL(006)/FALS(007); the time and date will be the time and date from the Calendar/Clock Area, words G001 to G004. This structure is shown below.

Word	Bit	Content
First	00 to 15	Error code
Second	00 to 15	Error contents
Third	00 to 07	Seconds
	08 to 15	Minutes
Fourth	00 to 07	Hours
	08 to 15	Day of month
Fifth	00 to 07	Month
	08 to 15	Year

**Operation**

When the first error code is generated, the relevant data will be placed in the error record after the one indicated by the Log Record Pointer (initially this will be record 1) and the Pointer will be incremented. Any other error codes generated thereafter will be placed in consecutive records until the last one is used.

If there are words allocated for n errors and n errors occur, the next error will be written into the last position, n, the contents of previous error will be moved to record n-1, and so on until the contents of record 1 is moved off the end and lost, i.e., the area functions like a shift register that moves data in units of error records (5 words). The Record Pointer will remain set to n (binary).

The Error Log Area can be reset by turning ON bit A00014 (Error Log Reset Bit). When this is done, the Record Pointer will be reset to 0000, the Error Log Area will be cleared, and any further error codes will be recorded from the beginning of the Error Log Area.

**3-6-14 MACRO Input/Output Areas**

When MCRO(156) is executed, the contents of the four source words are copied to the MACRO input area (A200 through A203) and these input arguments are passed to the specified subroutine.

When the subroutine is completed, the results of the subroutine are copied to the MACRO output area (A204 through A207) and these results are passed to the four destination words.

**3-6-15 CPU Bus Unit Initializing Flags**

Bits A30200 through A30215 turn ON while the corresponding CPU Bus Units (Units #0 through #15, respectively) are initializing.

**3-6-16 Wait Flags**

- Start-up Wait Flag (A30600)** Bit A30600 is ON when the start input terminals are open.
- I/O Verification Error Wait Flag (A30601)** Bit A30601 is ON when the PC is not running because an I/O Verification Error has occurred.
- SYSMAC BUS Terminator Wait Flag (A30602)** Bit A30602 is ON when the PC is not running because there is a terminator missing in the SYSMAC BUS System.
- CPU Bus Unit Initializing Wait Flag (A30603)** Bit A30603 is ON when the PC is not running because a CPU Bus Unit is initializing.
- Duplex Initializing Wait Flag (A39115)** Bit A39115 is turned ON during duplex initialization processing. For details of duplex initialization, refer to 6-2-2 *Duplex Initialization*.

### 3-6-17 Duplex Startup Error Flags

- Duplex Bus Error At Startup Flag (A30604)** Bit A30604 is ON when a duplex bus error occurs at the start of duplex operation after power is turned on. (Startup mode set to RUN or MONITOR.)
- Duplex Verification Error At Startup Flag (A30605)** Bit A30605 is ON when a duplex verification error occurs at the start of duplex operation after power is turned on. (Startup mode set to RUN or MONITOR.)
- Alternate CPU Waiting At Startup Flag (A30606)** Bit A30606 is ON when the other CPU is not operating at the start of duplex operation after power is turned on. (Startup mode set to RUN or MONITOR.)

### 3-6-18 Programming Device Flags

**Programming Device Connected Flag (A30615)** Bit A30615 is ON when a Programming Device is connected to the CPU, the Expansion CPU, or an Expansion I/O Rack.

**SYSMAC BUS/2 Programming Device Flags (A307 and A308)** Bits A30700 through A30815 are turned ON when a Programming Device is connected to the corresponding Slave Rack, as shown in the following table.

Word	Bits	
	00 to 07	08 to 15
A307	Racks #0 to #7 on Master #0	Racks #0 to #7 on Master #1
A308	Racks #0 to #7 on Master #2	Racks #0 to #7 on Master #3

**Programming Device Cycle Time (A309)** Word A309 contains the cycle time in ms (in binary) required to service Programming Devices, Host Link, and CPU Bus Units. Refer to *6-2 Cycle Time* for details.

### 3-6-19 CPU Bus Unit Service Interval

Words A310 through A325 contain the interval in ms (binary) between CPU Bus Unit services for Units #0 through #15, respectively. Measuring the service interval can be enabled or disabled in the PC Setup.

### 3-6-20 Memory Card Flags

- Memory Card Type (A34300 to A34303)** The binary number stored in A34300 to A34303 indicates the type of Memory Card, if any, installed in the Memory Card Drive. (0: None, 1: RAM, 2: EPROM, 3: EEPROM)
- Memory Card Format Error Flag (A34307)** Bit A34307 is turned ON when the Memory Card has not been initialized or initialization was not normal.
- Accessing Memory Card Flag (A34314)** Bit A34314 is turned ON when the Memory Card is being accessed, i.e., Memory Card data is being written, read, or compared.
- Memory Card Protected Flag (A34315)** Bit A34315 is turned ON when the Memory Card is write-protected by the write-protect switch on the card.

### 3-6-21 Alternate CPU Unit Status Flags

- Duplex CPU Replacement Enabled Flag (A39409)** Bit A39409 is turned ON when the other CPU Unit can be removed or connected during duplex operation.
- Duplex CPU Error Flag (A39410)** Bit A39410 is turned ON when there is a CPU error in the other CPU Unit during duplex operation.

**Duplex CPU Memory Error Flag (A39411)** Bit A39411 is turned ON when there is a memory error in the other CPU Unit during duplex operation.

### 3-6-22 Duplex Verification Error Flags

**Duplex Verification Error Flag (A40200)** Bit A40200 is turned ON when an error is detected in comparisons between the CPU Units. Refer to the four flags below (A39612 to A39615) to determine the cause of the verification error.

**Duplex EM Size Mismatch Flag (A39612)** Bit A39612 is turned ON when the EM size settings do not match.

**Duplex PC Setup Mismatch Flag (A39613)** Bit A39613 is turned ON when the PC Setup settings do not match.

**Duplex PC Setup Transfer Error Flag (A39614)** Bit A39614 is turned ON when the PC Setup is not transferred properly.

**Duplex Program Mismatch Flag (A39614)** Bit A39615 is turned ON when the programs do not match.

### 3-6-23 Power Supply Unit OFF/Error Flags

**CPU Rack PSU OFF Flags (A39700 and A39701)** Bit A39700 is turned ON when the right Power Supply Unit in the CPU Rack is OFF or its start input terminals are open.  
Bit A39701 is turned ON when the left Power Supply Unit in the CPU Rack is OFF or its start input terminals are open.

**CPU Rack PSU Error Flags (A39800 and A39801)** Bit A39800 is turned ON when there is an error in the right Power Supply Unit in the CPU Rack. Bit A39801 is turned ON when there is an error in the left Power Supply Unit in the CPU Rack.

**Rack #1 PSU Error Flags (A39802 and A39803)** Bit A39802 is turned ON when there is an error in the right Power Supply Unit in the Expansion CPU Rack or Expansion I/O Rack #1.  
Bit A39803 is turned ON when there is an error in the left Power Supply Unit in the Expansion CPU Rack or Expansion I/O Rack #1.

**PSU Error Flags for Racks #2 through #7 (A39804 to A39815)** Bits A39804 through A39815 contain the Error Flags for Power Supply Units in Expansion I/O Racks #2 through #7, as shown in the following table.

Bit	Function
A39804	Expansion I/O Rack #2 Right Power Supply Unit Error Flag
A39805	Expansion I/O Rack #2 Left Power Supply Unit Error Flag
A39806	Expansion I/O Rack #3 Right Power Supply Unit Error Flag
A39807	Expansion I/O Rack #3 Left Power Supply Unit Error Flag
A39808	Expansion I/O Rack #4 Right Power Supply Unit Error Flag
A39809	Expansion I/O Rack #4 Left Power Supply Unit Error Flag
A39810	Expansion I/O Rack #5 Right Power Supply Unit Error Flag
A39811	Expansion I/O Rack #5 Left Power Supply Unit Error Flag
A39812	Expansion I/O Rack #6 Right Power Supply Unit Error Flag
A39813	Expansion I/O Rack #6 Left Power Supply Unit Error Flag
A39814	Expansion I/O Rack #7 Right Power Supply Unit Error Flag
A39815	Expansion I/O Rack #7 Left Power Supply Unit Error Flag

### 3-6-24 Online I/O Replacement Flags

A399	00 to 07	Slot Undergoing Online I/O Replacement (2-digit BCD)
	08 to 11	Rack Undergoing Online I/O Replacement (1-digit BCD)
	12 to 13	Not used.
	14	Online I/O Replacement Status Retention Bit
	15	Online I/O Replacement In Progress Flag

#### Unit Undergoing Online I/O Replacement

Bit A39900 to A39911 indicate the rack number and slot number of the Unit undergoing online I/O replacement. Bits 08 to 11 contain the rack number (1-digit BCD) and bits 00 to 07 contain the slot number (2-digit BCD).

#### Online I/O Replacement Status Retention Flag (A39914)

Bit A39914 indicates whether the status of data in I/O Units will be retained while the rack is undergoing online I/O replacement. Bit A39914 is turned ON if the status of the I/O bits is retained and turned OFF if the bits are cleared.

#### Online I/O Replacement In Progress Flag (A39915)

Bit A39915 is turned ON during online I/O replacement.

### 3-6-25 Error Code

When an error or alarm occurs, the error code is written to A400. If two errors occur simultaneously, the more serious error, with a higher error code, is recorded. Refer to *Section 6 Troubleshooting* in the *CVM1D PCs Installation Guide* for details on error codes.

### 3-6-26 FALS Flag

Bit A40106 is turned ON when the SEVERE ALARM FAILURE instruction (FALS(007)) is executed. The FAL number is written to word A400.

### 3-6-27 Cycle Time Too Long Flag

Bit A40108 is turned ON if the cycle time exceeds the cycle time monitoring time (i.e., the maximum cycle time) set in the PC Setup.

### 3-6-28 Program Error Flag

Bit A40109 is turned ON if there is a program syntax error (including no END(001) instruction).

### 3-6-29 I/O Setting Error Flag

Bit A40110 is turned ON if the I/O designation of a slot has changed, e.g., an Input Unit has been installed in an Output Unit's slot, or vice versa.

### 3-6-30 Too Many I/O Points Flag

Bit A40111 is turned ON if the total number of I/O points being used exceeds the maximum for the PC. The total number of I/O points being used on CPU and Expansion Racks is written to word A407; in the SYSMAC BUS/2 system, to word A408; and in the SYSMAC BUS system, to word A478.

### 3-6-31 CPU Bus Error and Unit Flags

Bit A40112 is turned ON when an error occurs during the transmission of data between the CPU and CPU Bus Units, or a WDT (watchdog timer) error occurs in a CPU Bus Unit. The unit number of the CPU Bus Unit involved is contained in word A405.



Bits A40500 through A40515 correspond to CPU Bus Units #0 through #15, respectively. When a CPU Bus Error occurs, the bit corresponding to the unit number of the CPU Bus Unit involved is turned ON.

### 3-6-32 Duplication Error Flag and Duplicate Rack/CPU Bus Unit Numbers

Bit A40113 is turned ON when two Racks are assigned the same rack number, two CPU Bus Units are assigned the same unit number, or the same words are allocated to more than one Rack or Unit in the PC Setup. The duplicate Expansion I/O Rack number is written to word A409, and the duplicate CPU Bus Unit number is written to word A410.

Bits A40900 through A40907 correspond to Racks #0 through #7, respectively. When two Racks have the same rack number, the bits corresponding to the rack numbers involved are turned ON. Bit A40915 is also turned ON to indicate that the same words are allocated to more than one Rack or Unit in the PC Setup.

Bits A41000 through A41015 correspond to CPU Bus Units #0 through #15, respectively. When two CPU Bus Units have the same unit number, the bits corresponding to the unit numbers of the CPU Bus Units involved are turned ON.

### 3-6-33 I/O Bus Error Flag and I/O Bus Error Slot/Rack Numbers

Bit A40114 is turned ON when an error occurs during the transmission of data between the CPU and I/O Units through the I/O bus, or a terminator is not installed correctly. The rack/slot number of the Unit involved is written to word A404.

Bits A40400 through A40407 contain the slot number, in BCD, of the I/O Unit where the error occurred. If the error did not occur with an I/O Unit, then these bits contain #0F. Bits A40408 through A40415 contain the rack number, in BCD, of the Rack where the error occurred.

If the error occurred because of a terminator setting, word A404 will contain #0E0F for line 0 (IOC right connector), or #0F0F for line 1 (IOC left connector).

### 3-6-34 Memory Error Flag

Bit A40115 is turned ON when an error occurs in memory. The memory area involved is written to word A403.

### 3-6-35 Duplex Bus Error Flag

Bit A40201 is turned ON when an error is detected in the duplex bus error check.

### 3-6-36 Power Interruption Flag

Bit A40202 is turned ON when power is momentarily interrupted if a momentary power interruption is set as an error in the PC Setup (see "Error on power off" in the PC Setup). The time and date of the most recent power interruption is written to words A012 and A013, and the number of power interruptions is written to word A014.

### 3-6-37 CPU Bus Unit Setting Error Flag and Unit Number

Bit A40203 is turned ON when the CPU Bus Units actually installed differ from the Units registered in the I/O table. The unit number of the CPU Bus Unit involved is written to word A427.

Bits A42700 through A42715 correspond to CPU Bus Units #0 through #15, respectively. When an error occurs, the bit corresponding to the unit number of the CPU Bus Unit involved is turned ON.

### 3-6-38 Battery Low Flags

Bit A40204 is turned ON if the voltage of the CPU or Memory Card battery drops. If the problem has occurred with the Memory Card battery, bit A42614 will be

turned ON, and if the problem has occurred with the CPU battery, bit A42615 will be turned ON.

### 3-6-39 SYSMAC BUS Error Flag, Check Bits, and Master/Unit Numbers

Bit A40205 is turned ON when an error occurs during the transmission of data in the SYSMAC BUS system. The number of the Master involved is written to word A425, and information about the Unit(s) involved is written to words A470 through A477.

Bits A42500 through A42507 correspond to Masters #0 through #7, respectively. When a error occurs, the bit corresponding to the number of the Master involved is turned ON.

Words A470 through A477 are used to indicate which Unit is involved in the error on Masters #0 through #7, respectively. The function of each bit is described below. Refer to the *Optical* and *Wired Remote I/O System Manuals* for details.

**Bits 00 to 02**

Not used.

**Bit 03 – Remote I/O Error Flag**

Bit 03 turns ON when an error has occurred in remote I/O.

**Bits 04 to 15**

If the content of bits 12 through 15 is B, an error has occurred in a Remote I/O Master or Slave Unit, and the content of bits 08 through 11 will indicate the number of the Master of the Remote I/O Subsystem involved. These numbers are assigned to Masters in the order that they are mounted to the CPU and Expansion Racks. If the error is in the Master, the value of bits 4 to 7 will be 8. If the error is in a Slave, bits 4 to 7 will contain the unit number of the Slave where the error occurred.

If the content of bits 12 through 15 is other than B, an error has occurred in an Optical I/O Unit, I/O Link Unit, or I/O Terminal. Here, bits 08 through 15 will provide the word address (#00 to #31) that has been set on the Unit.

When this Unit is an Optical I/O Unit, bit 04 will be ON if the Unit is assigned leftmost bits (08 through 15), and OFF if it is assigned rightmost bits (00 through 07).

**Error Check Bits (A00500 to A00507)**

If there are errors in more than one Unit for a single Master, words A470 through A477 will contain error information for only the first one. Data for the remaining Units will be stored in memory and can be accessed by turning ON the Error Check Bit for that Master. Bits A00500 through A00507 are the Error Check Bits for Masters #0 through #7, respectively. Error Check Bits are automatically turned OFF when data has been accessed. Write down the data for the first error if required before using the Error Check Bit; previous data will be cleared when data for the next error is displayed.

### 3-6-40 SYSMAC BUS/2 Error Flag and Master/Unit Numbers

Bit A40206 is turned ON when an error occurs during the transmission of data in the SYSMAC BUS/2 System. The number of the Master involved is written to word A424, and information about the Slave Unit(s) involved is written to words A480 through A499.

Bits A42400 through A42403 are turned ON when the error involves Masters #0 through #3, respectively.

Information identifying the Slave Unit(s) involved is contained in words A480 through A499, which are divided into four groups of five words, one group for each Master, as shown below.

Words	Master number
A480 to A484	0
A485 to A489	1

Words	Master number
A490 to A494	2
A495 to A499	3

Bits are turned ON to indicate which of the Slaves connected to the Master was involved in the error, as shown below.

Word	Bits	Slave
First	00 to 15	Group-1 Slaves #0 to #15
Second	00 to 15	Group-1 Slaves #16 to #31
Third	00 to 15	Group-2 Slaves #0 to #15
Fourth	00 to 07	Slave Racks #0 to #7 (Group-3 Slaves)
	08 to 15	Not used.
Fifth	00 to 15	Not used

### 3-6-41 CPU Bus Unit Error Flag and Unit Numbers

Bit A40207 is turned ON when a parity error occurs during the transmission of data between the CPU and CPU Bus Units. The unit number of the CPU Bus Unit involved is written to word A422.

Bits A42200 through A42215 correspond to CPU Bus Units #0 through #15, respectively. When a CPU Bus Unit Error occurs, the bit corresponding to the unit number of the CPU Bus Unit involved is turned ON.

### 3-6-42 I/O Verification Error Flag

Bit A40209 is turned ON when the Units mounted in the system disagree with the I/O table registered in the CPU. To ensure proper operation, PC operation should be stopped, Units checked, and the I/O table corrected whenever this flag goes ON.

### 3-6-43 Indirect DM BCD Error Flag

Bit A40212 is turned ON if the content of an indirectly addressed DM word is not BCD when BCD is specified in the PC Setup.

The contents of indirectly addressed DM words can be set to either binary or BCD with the PC Setup. Binary addresses will access memory according to internal I/O memory addresses. BCD will access other DM words according to DM Area addresses. If binary addresses are used, this flag will not operate.

### 3-6-44 Jump Error Flag

Bit A40213 is turned ON if there is no destination for a JMP(004) instruction.

### 3-6-45 Duplex Power Supply Error Flag

Bit A40214 is turned ON if an error is detected during the duplex power supply error check. Refer to *3-6-23 Power Supply Unit OFF/Error Flags* for details on the flags that indicate which Power Supply Unit is the cause of the error.

### 3-6-46 FAL Flag and FAL Number

Bit A40215 is turned ON when the FAL(006) instruction is executed. The FAL number is then written to words A430 to A461. Bits from A43001 to A46115 correspond consecutively to FAL numbers 001 to 511

### 3-6-47 Memory Error Area Location

Bits A40300 to A40308 are turned ON to indicate the memory area in which a memory error has occurred. The bits correspond to memory areas as follows:

00:	Program Memory	05:	I/O Table
01:	Memory Card	06:	System Memory
02:	I/O Memory	07:	Routing Tables
03:	EM	08:	CPU Bus Unit Software Switches
04:	PC Setup		

### 3-6-48 Memory Card Start-up Transfer Error Flag

Bit A40309 is turned ON when an error occurs during the transmission of the program from the Memory Card when power is turned ON. An error can occur because the AUTOEXEC file is missing, the Memory Card is not installed, or the System Protect setting is ON.

### 3-6-49 CPU-recognized Rack Numbers

Bits A41900 through A41907 are turned ON when Expansion Racks #0 through #7, respectively, are recognized by the CPU.

### 3-6-50 CPU Bus Unit Number Setting Error Flag

Bit A42314 is turned ON when a CPU Bus Unit is not set to an acceptable unit number (0 to 15).

### 3-6-51 CPU Bus Link Error Flag

Bit A42315 is turned ON when a parity error occurs with CPU bus links.

### 3-6-52 Maximum Cycle Time

Words A462 and A463 contain the maximum cycle time that has occurred since operation was started. If the maximum cycle time is exceeded, however, the previous maximum cycle time will remain in words A462 and A463. The time is recorded in 8-digit BCD in tenths of milliseconds (0000000.0 ms to 9999999.9 ms), as shown in the following table.

Word	Bits			
	15 to 12	11 to 08	07 to 04	03 to 00
A463	10 <sup>6</sup>	10 <sup>5</sup>	10 <sup>4</sup>	10 <sup>3</sup>
A462	10 <sup>2</sup>	10 <sup>1</sup>	10 <sup>0</sup>	10 <sup>-1</sup>

### 3-6-53 Present Cycle Time

Words A464 and A465 contain the present cycle time unless the maximum cycle time is exceeded, in which case the previous cycle time will remain. The time is recorded in 8-digit BCD in tenths of milliseconds (0000000.0 ms to 9999999.9 ms), as shown in the following table.


Word	Bits			
	15 to 12	11 to 08	07 to 04	03 to 00
A465	10 <sup>6</sup>	10 <sup>5</sup>	10 <sup>4</sup>	10 <sup>3</sup>
A464	10 <sup>2</sup>	10 <sup>1</sup>	10 <sup>0</sup>	10 <sup>-1</sup>

### 3-6-54 Instruction Execution Error Flag, ER

Bit A50003 is turned ON if an attempt is made to execute an instruction with incorrect operand data. Common causes of an instruction error are non-BCD operand data when BCD data is required, or an indirectly addressed DM word that is non-existent. **When the ER Flag is ON, the current instruction will not be executed.**

### 3-6-55 Arithmetic Flags

The following flags are used in data shifting, arithmetic calculation, and comparison instructions. They are generally referred to only by their two-letter abbreviations.

 **Caution** These flags are all reset when the END instruction is executed, and therefore cannot be monitored from a Programming Device.

Refer to *5-14 Shift Instructions*, *5-16 Comparison Instructions*, *5-18 BCD Calculation Instructions*, and *5-19 Binary Calculation Instructions* for details.

**Carry Flag, CY**

Bit A50004 is turned ON when there is a carry in the result of an arithmetic operation or when a rotate or shift instruction moves a “1” into CY. The content of CY is also used in some arithmetic operations, e.g., it is added or subtracted along with other operands. This flag can be set and cleared from the program using the SET CARRY and CLEAR CARRY instructions. This Flag is also used by the I/O READ and I/O WRITE instructions. Refer to page 378 for details.

**Greater Than Flag, GR**

Bit A50005 is turned ON when the result of a comparison shows the first of two operands to be greater than the second.

**Equals Flag, EQ**

Bit A50006 is turned ON when the result of a comparison shows two operands to be equal or when the result of an arithmetic operation is zero.

**Less Than Flag, LE**

Bit A50007 is turned ON when the result of a comparison shows the first of two operands to be less than the second.

**Negative Flag, N**


Bit A50008 is turned ON when the highest bit in the result of a calculation is ON.

**Overflow Flag, OF**

Bit A50009 is turned ON when the absolute value of the result is greater than the maximum value that can be expressed.

**Underflow Flag, UF**

Bit A50010 is turned ON when absolute value of the result is less than the minimum value that can be expressed.

 **Caution** The previous seven flags are cleared when END(001) is is executed.

**3-6-56 Step Flag**

Bit A50012 is turned ON for one cycle when step execution is started with the STEP(008) instruction.

**3-6-57 First Cycle Flag**

When ladder-only programming is used, bit A50015 turns ON when PC operation begins and then turns OFF after one cycle of the program. The First Cycle Flag is useful in initializing counter values and other operations. An example of this is provided in *5-13 Timer and Counter Instructions*.

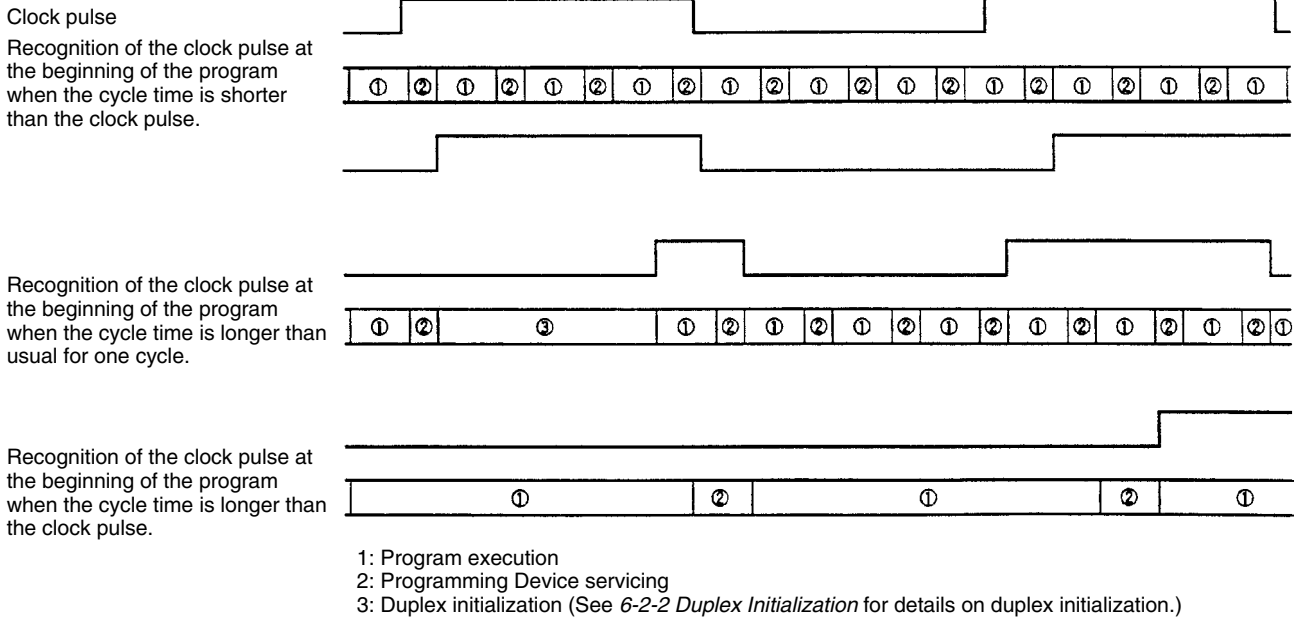
**Note** Do not use A50015 to control execution of differentiated instructions. The instructions will never be executed.

**3-6-58 Clock Pulse Bits**

Four clock pulses are available to control program timing. Each clock pulse bit is ON for the first half of the rated pulse time, then OFF for the second half. In other words, each clock pulse has a duty factor of 50%.

<b>Pulse width</b>	0.1 s	0.2 s	1.0 s	0.02 s
<b>Bit</b>	A50100	A50101	A50102	A50103

These clock pulse bits are often used with counter instructions to create timers. Refer to *5-13 Timer and Counter Instructions* for an example of this.



**Caution** The CPU may not be able to accurately read clock pulses if the cycle time is longer than the pulse width.

### 3-6-59 Network Status Flags

Bits A50200 through A50207 are turned ON to indicate that ports #0 through #7, respectively, are enabled for the SEND(192), RECV(193), and CMND(194). Bits A50208 through A50215 are turned ON to indicate that an error has occurred in ports #0 through #7, respectively, during data communications using SEND(192), RECV(193), or CMND(194).

A503 through A510 contain the completion codes for ports #0 through #7, respectively, following data communications using SEND(192), RECV(193), or CMND(194). Refer to the manuals for Communications Units for details on completion codes.

### 3-6-60 EM Status Flags

The rightmost digit of A511 will contain the current bank number. Bit A51115 (the EM Installed Flag) is turned ON when a EM Unit is mounted to the CPU.

## 3-7 Timer Area

Timer Completion Flags and present values (PV) are accessed through timer numbers ranging from T0000 through T1023. Each timer number and its set value (SV) are defined using timer instructions. No prefix is required when using a timer number to create a timer in one of these instructions.

The same timer number can be defined using more than one of these instructions as long as the instructions are not executed in the same cycle. If the same timer number is defined in more than one of these instructions or in the same instruction twice, an error will be generated during the program check, but as long as the instructions are not executed in the same cycle, they will operate correctly. There are no restrictions on the order in which timer numbers can be used.

Once defined, a timer number can be designated as an operand in one or more of certain instructions. Timer numbers can be designated for operands that re-

quire bit data or for operands that require word data. When designated as an operand that requires bit data, the timer number accesses the Completion Flag of the timer. The Completion Flag will be ON when the timer has timed out. When designated as an operand that requires word data, the timer number accesses a memory location that holds the PV of the timer.

Timer PVs are reset when PC operation is begun, when the CNR(236) instruction is executed, and when in interlocked program sections when the execution condition for IL(002) is OFF. Refer to *5-8 Interlock and Interlock Clear – IL(02) and ILC(03)* for details on timer operation in interlocked program sections.

Timer Completion Flags are allocated to internal I/O memory addresses (bit addresses) F000 through F3FF, corresponding to timer numbers T0000 through T1023. Timer PVs are allocated to internal I/O memory addresses (word addresses) 1000 through 13FF, corresponding to timer numbers T0000 through T1023. Completion Flags and PVs can be accessed directly with their internal I/O memory addresses, but they are normally accessed by using the timer numbers in the program.

All timer instructions can be affected by the cycle time. Refer to *5-13 Timer and Counter Instructions* for details.

## 3-8 Counter Area

Counter Completion Flags and present values (PV) are accessed through counter numbers ranging from C0000 through C1023. Each counter number and its set value (SV) are defined using counter instructions. No prefix is required when using a counter number to create a counter in a counter instruction. The same counter number can be defined using more than one of these instructions as long as the instructions are not executed in the same cycle. If the same counter number is defined in more than one of these instructions or in the same instruction twice, an error will be generated during the program check, but as long as the instructions are not executed in the same cycle, they will operate correctly. There are no restrictions on the order in which counter numbers can be used.

Once defined, a counter number can be designated as an operand in one or more of certain instructions other than those listed above. Counter numbers can be designated for operands that require bit data or for operands that require word data. When designated as an operand that requires bit data, the counter number accesses the completion flag of the counter. When designated as an operand that requires word data, the counter number accesses a memory location that holds the PV of the counter.

Counter PVs are reset when the CNR(236) instruction is executed, but unlike timers, counters maintain their status when PC operation is begun, and when in interlocked program sections when the execution condition for IL(002) is OFF.

Counter Completion Flags are allocated to internal I/O memory addresses (bit addresses) F800 through F8FF, corresponding to counter numbers C0000 through C1023. Counter PVs are allocated to internal I/O memory addresses (word addresses) 1800 through 1BFF, corresponding to counter numbers C0000 through C1023. Completion Flags and PVs can be accessed directly with their internal I/O memory addresses, but they are normally accessed by using the counter numbers in the program.

## 3-9 DM and EM Areas

The DM (Data Memory) Area is used for internal data storage and manipulation and is accessible only by word. Addresses range from D0000 through D24575.

The EM (Extended Data Memory) Area is contained in the CV1000-DM□□1 EM Unit, a card which must be purchased separately and installed into a slot on

the CPU Unit in order for the EM Area to be used. There are three models of Memory Units available, with 64K words (E00000 to E32765 × 2 banks), 128K words (E00000 to E32765 × 4 banks), and 256K words (E00000 to E32765 × 8 banks).

When the PC is turned on, the EM bank number is automatically set to 0, but can be changed with the EMBC(171) instruction.

When using the SYSMAC NET Link or SYSMAC LINK systems, D00000 through D00127 are automatically allocated as part of the Data Link Table unless data link are set manually from the SSS. The 1,600 words from D02000 to D03599 are allocated for CPU Bus Units, 100 words for each Unit. The particular function depends on the type of CPU Bus Unit being used. Refer to the CPU Bus Unit's *Operation Manual* for details.

**Note** D02000 to D03599 are not used by SYSMAC BUS/2 Remote I/O Master Units.

Although composed of 16 bits just like any other word in memory, DM and EM words cannot be specified by bit for use in instructions with bit-size operands, such as LD, OUT, AND, and OR, nor can DM words be used with the SHIFT instruction.

The DM and EM Areas retain status during power interruptions.

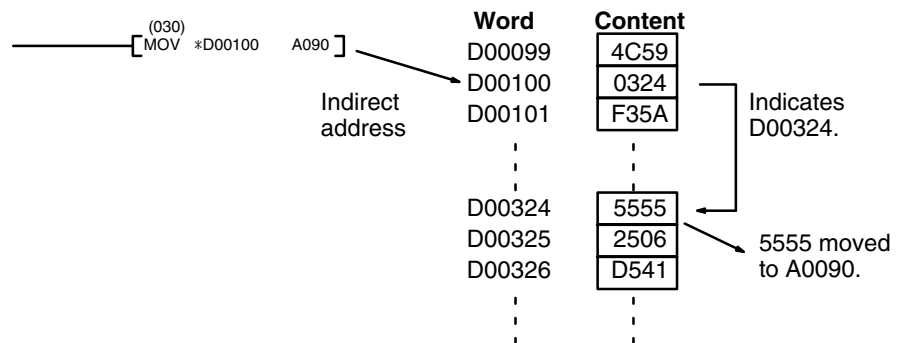
**Indirect Addressing**

Normally, when the content of a data area word is specified for an instruction, the instruction is performed directly on the content of that word. For example, suppose CMP(020) (COMPARE) is used in the program with CIO 0005 as the first operand and D00010 as the second operand. When this instruction is executed, the content of CIO 0005 is compared with that of D00010.

It is also possible, however, to use indirect DM and EM addresses as operands for instructions. If \*D00100 is specified as the data for a programming instruction, the asterisk in front of D indicates that it is an indirect address that specifies another which contains the actual operand data. Likewise, EM indirect addressing is indicated by an asterisk in front of the E, \*E. When addressed indirectly, the content of \*D00100 can be read as either BCD or binary (hexadecimal) data, depending on the PC Setup for indirect addressing.

If the content of \*DM (or \*EM) addresses is defined as BCD in the PC Setup, the number indicates another DM (or EM) address. If the contents of the \*DM address are not BCD, a \*DM BCD error will occur, and an error flag, A50003, will be turned ON. Because only the last four digits of the final address can be specified in one word, the range of possible BCD numbers is #0000 to #9999, and the range of DM addresses that can be addressed indirectly is D00000 to D09999.

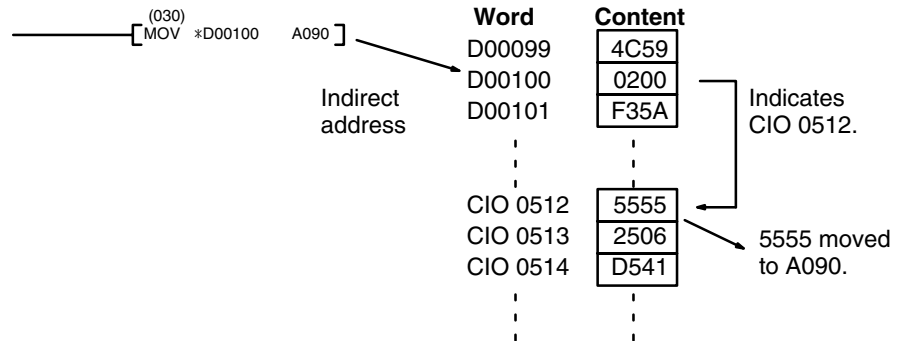
If the content of D00100 is #0324, then \*D00100 indicates D00324 as the word that contains the desired data, and the content of D00324 is used as the operand in the instruction. The following shows an example of this with the MOVE instruction.





If the content of \*DM addresses is defined as binary in the PC Setup, the number indicates a internal I/O memory address. The range of possible binary numbers, \$0000 to \$FFFF, allows all memory areas, including EM, to be indirectly addressed.

If, in this case, the content of D00100 is \$0324, then \*D00100 indicates internal I/O memory address \$0324, which is CIO 0804 in the SYSMAC BUS/2 Area, as the word that contains the desired data, and the content of CIO 0804 is used as the operand in the instruction. The following example shows this type of indirect addressing with the MOVE instruction.



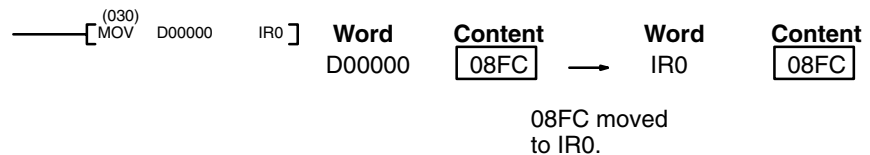
Indirect addressing can also be used in instructions that require bit operands for bits in the Core I/O Area (\$0000 to \$0FFF). These bits are designated by using the rightmost digits of the memory address as the leftmost three digits of the hexadecimal address and adding the bit number as the rightmost digit. For example, the CIO bit 190000 is designated by \$76CA where 76C is the rightmost three digits of the memory address (CIO word 1900 is \$076C) and A is bit 10.

### 3-10 Index and Data Registers (IR and DR)

The Index Registers, IR0, IR1, and IR2, which contain a single word of data, are used for indirect addressing. A “,” prefix is included before an Index Register to indicate indirect addressing, just as the “\*” prefix is used to indicate indirect addressing with DM and EM.

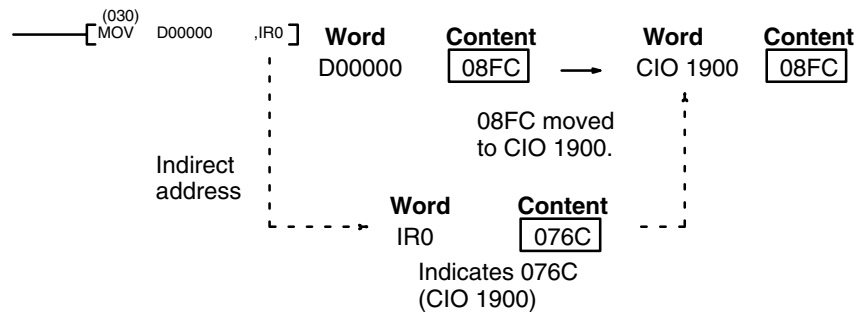
#### Direct Addressing

If an Index Register is used as an operand in an instruction without the “,” prefix, the instruction is performed directly on the content of that Index Register, as in the following example.



#### Indirect Addressing

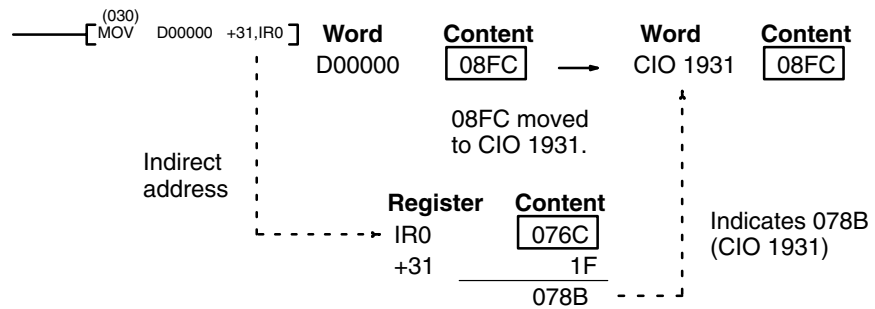
If an Index Register is used as an operand in an instruction with the “,” prefix, the instruction is performed on the word at the internal I/O memory address indicated by that Index Register, as in the following example.



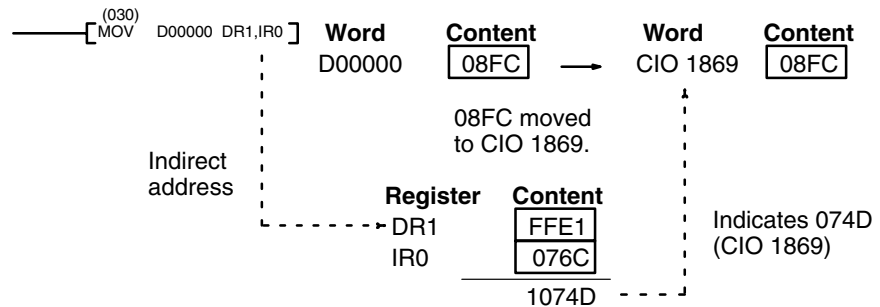
Indirect addressing can also be used in instructions that require bit operands for bits in the Core I/O Area (`$0000` to `$0FFF`). These bits are designated by using the rightmost digits of the memory address as the leftmost three digits of the hexadecimal address and adding the bit number as the rightmost digit. For example, the bit `CIO 190000` is designated by `$76CA` where `76C` is the rightmost three digits of the memory address (word `CIO 1900` is `$076C`) and `A` is bit 10.

**Offset Indirect Addressing**

The internal I/O memory address indicated in an Index Register can be offset by a specified constant or by the content of a Data Register (`DR0`, `DR1`, or `DR2`) by inputting the constant or the Data Register before the “,” prefix. The constant must be in BCD between `-2047` and `+2047`. To offset the indirect addressing by `+31` words, simply input `+31`, before the “,” prefix, as shown.



If a Data Register is input before the “,” prefix, the content of the Data Register will be added to the content of the Index Register, and the result is the internal I/O memory address that is indirectly addressed. If the result exceeds `$FFFF`, the carry to the fifth digit is truncated (effectively subtracting `$10000` (65,536, decimal) from the result). In the following example, `DR1` is added to `IR0`. The content of `DR1` is `$FFE1`, so adding `$FFE1` is equivalent to subtracting `1F` (`$0FFE1 - $10000 = -1F`), for all `IR0` values greater than or equal to `$001F`.



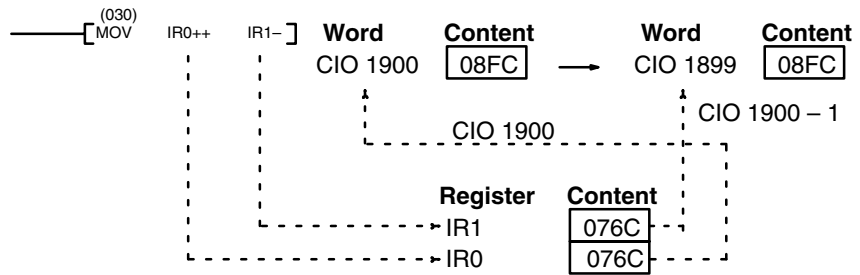
**Auto-increments and Auto-decrements**

An auto-increment increases the contents of an Index Register by 1 or 2 after executing the instruction. A “+” suffix indicates an auto-increment of 1, and a “++” suffix indicates an auto-increment of 2.

An auto-decrement decreases the contents of an Index Register by 1 or 2 before executing the instruction. A “-” prefix indicates an auto-decrement of 1, and a “-” prefix indicates an auto-decrement of 2. The notation for auto-increments and auto-decrements is as follows:

- ,IRn+: After execution, increase the contents of IRn by 1.
- ,IRn++: After execution, increase the contents of IRn by 2.
- ,-IRn: Decrease the contents of IRn by 1 before execution.
- ,-IRn: Decrease the contents of IRn by 2 before execution.

Both an auto-increment and an auto-decrement are used in the following example. The data movement for the first execution is shown. The second execution would move the contents of CIO 1902 to CIO 1898; the third execution would move the contents of CIO 1904 to CIO 1897; etc.



# SECTION 4

## Writing Programs

This section explains the basic steps and concepts involved in writing a basic ladder diagram program. It introduces the instructions that are used to build the basic structure of the ladder diagram and control its execution, along with a few other instructions of special interest in programming.

The entire set of instructions used in programming is described in *Section 5 Instruction Set*.

4-1	Basic Procedure	66
4-2	Instruction Terminology	66
4-3	Basic Ladder Diagrams	67
4-3-1	Basic Terms	67
4-3-2	Basic Mnemonic Code	68
4-3-3	Ladder Instructions	69
4-3-4	OUTPUT and OUTPUT NOT	71
4-3-5	The END Instruction	72
4-4	Mnemonic Code	72
4-4-1	Logic Block Instructions	72
4-4-2	Coding Multiple Right-hand Instructions	79
4-5	Branching Instruction Lines	79
4-5-1	TR Bits	80
4-5-2	Interlocks	82
4-6	Jumps	83
4-7	Controlling Bit Status	85
4-7-1	DIFFERENTIATE UP and DIFFERENTIATE DOWN	85
4-7-2	SET and RESET	85
4-7-3	KEEP	86
4-7-4	Self-maintaining Bits (Seal)	86
4-8	Intermediate Instructions	87
4-9	Work Bits (Internal Relays)	87
4-10	Programming Precautions	89
4-11	Program Execution	90
4-12	Data Formats	90
4-12-1	Unsigned Binary Data	90
4-12-2	Signed Binary Data	91
4-12-3	BCD Data	94
4-12-4	Signed BCD Data	94
4-12-5	Floating-point Data	94

## 4-1 Basic Procedure

There are several basic steps involved in writing a program. Sheets that can be copied to aid in programming are provided in *Appendix E I/O Assignment Sheets* and *Appendix F Program Coding Sheet*.

- 1, 2, 3...
  1. Obtain a list of all I/O devices and the I/O points that have been assigned to them and prepare a table that shows the I/O bit allocated to each I/O device.
  2. If the PC has any Units that are allocated words in data areas other than the CIO area or are allocated CIO words in which the function of each bit is specified by the Unit, prepare similar tables to show what words are used for which Units and what function is served by each bit within the words. These Units include CPU Bus Units, Special I/O Units, and Link Units.
  3. Determine what words are available for work bits and prepare a table in which you can allocate these as you use them.
  4. Also prepare tables of timer and counter numbers and jump numbers so that you can allocate these as you use them. Remember, timer and counter numbers can be defined only once within the program; jump numbers can be used only once each. (timer/counter numbers are described in *5-13 Timer and Counter Instructions*; jump numbers are described in this section.)
  5. Draw the ladder diagram. (The CVM1D does not support SFC programming.)
  6. Input the program into the CPU. Actual input is done from the SSS and is possible in either ladder diagram or mnemonic form.
  7. Check the program for syntax errors and correct these.
  8. Execute the program to check for execution errors and correct these.
  9. After the entire Control System has been installed and is ready for use, execute the program and fine tune it if required.

The basics of ladder-diagram programming and conversion to mnemonic code are described in *4-3 Basic Ladder Diagrams*. The rest of Section 4 covers more advanced programming, programming precautions, and program execution. All instructions are covered in *Section 5 Instruction Set*. *Section 6 Troubleshooting* in the *CVM1D PCs Installation Guide* provides information required for debugging. Refer to the *SSS Operation Manuals* for program input, debugging, and monitoring procedures.

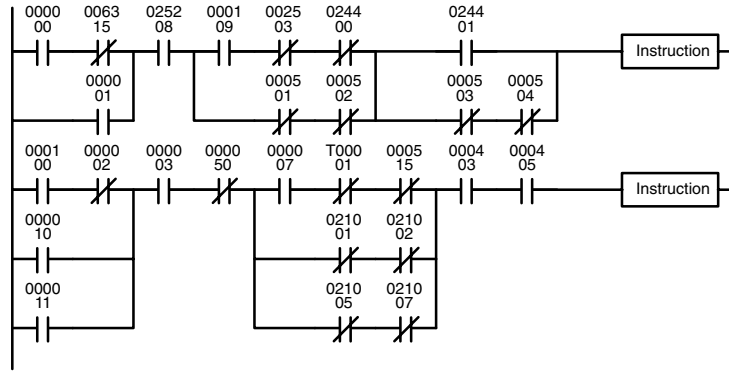
## 4-2 Instruction Terminology

There are basically two types of instructions used in ladder-diagram programming: instructions that correspond to the conditions on the rungs of the ladder diagram and are used in instruction form only when converting a program to mnemonic code, and instructions that are used on the right side of the ladder diagram and are executed according to the conditions on the instruction lines leading to them.

Most instructions have at least one or more **operands** associated with them. Operands indicate or provide the data on which an instruction is performed. These are sometimes input as the actual numeric values, but are usually the addresses of data area words or bits that contain the data to be used. For instance, a MOVE instruction that has CIO 0000 designated as the source operand will move the contents of CIO 0000 to some other location. The other location is also designated as an operand. A bit whose address is designated as an operand is called an **operand bit**; a word whose address is designated as an operand is called an **operand word**. If the value is entered as a constant, it is preceded by # to indicate it is not an address, but the actual value to be used in the instruction. Refer to *Section 5 Instruction Set* for other terms used in describing instructions.

### 4-3 Basic Ladder Diagrams

A ladder diagram consists of two vertical lines running down the sides with lines branching in between them. The vertical lines are called **bus bars**; the branching lines, **instruction lines** or rungs. Along the instruction lines are placed conditions that lead to other instructions next to the right bus bar. The logical combinations of the conditions on the instruction lines determine when and how the instructions at the right are executed. A ladder diagram is shown below.



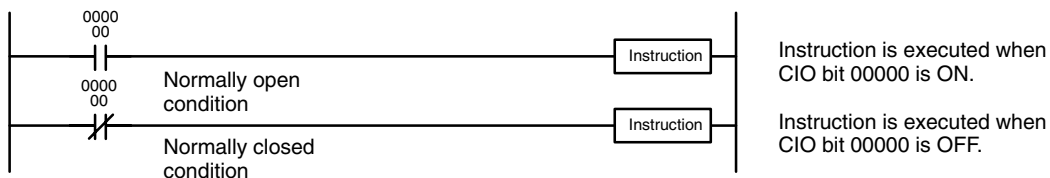
As shown in the diagram above, instruction lines can branch apart and they can join back together. The short vertical pairs of lines are called **conditions**. Conditions without diagonal lines through them are called **normally open conditions** and correspond to a LOAD, AND, or OR instruction. The conditions with diagonal lines through them are called **normally closed conditions** and correspond to a LOAD NOT, AND NOT, or OR NOT instruction. The number above each condition indicates the operand bit for the instruction. It is the status of the bit associated with each condition that determines the **execution condition** for following instructions. Only these conditions and a limited number of **intermediate instructions** can appear along the instruction lines. All other instructions must appear next to the right bus bar. Instructions that appear next to the right bus bar are called **right-hand instructions**.

The way the operation of each of the instructions corresponds to a condition is described below. Before we consider these, however, there are some basic terms that must be explained.

#### 4-3-1 Basic Terms

##### Normally Open and Normally Closed Conditions

Each condition in a ladder diagram is either ON or OFF depending on the status of the operand bit that has been assigned to it. A normally open condition is ON if the operand bit is ON; OFF if the operand bit is OFF. A normally closed condition is ON if the operand bit is OFF; OFF if the operand bit is ON. Generally speaking, you use a normally open condition when you want something to happen when a bit is ON, and a normally closed condition when you want something to happen when a bit is OFF.



<b>Execution Conditions</b>	In ladder diagram programming, the logical combination of ON and OFF conditions before an instruction determines the compound condition under which the instruction is executed. This condition, which is either ON or OFF, is called the execution condition for the instruction. All instructions other than LOAD instructions have execution conditions. Execution conditions are maintained in buffers in memory and are continuously changed by each instruction that is executed until a LOAD or LOAD NOT instruction is used to start a new instruction line and thus a new execution condition.
<b>Operand Bits</b>	The operands designated for any of the ladder instructions can be any bit in the data areas accessible by bit (e.g., not the DM or EM Areas). This means that the conditions in a ladder diagram can be determined by I/O bits, flags, work bits, timers/counters, etc. LOAD and OUTPUT instructions can also use TR Area bits, but they do so only in special applications. Refer to <i>4-5-1 TR Bits</i> for details.
<b>Logic Blocks</b>	The relationship between the conditions on the instruction lines that lead to an instruction determine the execution condition for the instruction. Any group of conditions that go together to create an execution condition for an instruction is called a logic block. Although ladder diagrams can be written without actually analyzing individual logic blocks, understanding logic blocks is necessary for efficient programming and is essential when programs are to be input in mnemonic code.
<b>Block Programming</b>	Block programming is a form of programming that can make it easier to program complex operations such as a series of data calculations that would be difficult to program using ladder diagrams. Creating structured programs can shorten cycle time, thereby improving overall system processing speed.

## 4-3-2 Basic Mnemonic Code

Programs can be input from a Programming Device in either graphic form (i.e., as a ladder diagram) or in mnemonic form (i.e., as a list of code). The mnemonic code provides exactly the same information as the ladder diagram. You can program directly in mnemonic code, although it is not recommended for beginners or for complex programs. Programming in mnemonic code is also necessary when a logic block contains more than twenty instruction lines.

Because of the importance of mnemonic code in complete understanding of a program, we will introduce and describe the mnemonic code along with ladder diagrams.

<b>Program Memory Structure</b>	The program is input into addresses in Program Memory. Addresses in Program Memory are slightly different to those in other memory areas because each address does not necessarily hold the same amount of data. Rather, each address holds one instruction and all of the definers and operands (described in more detail later) required for that instruction.
---------------------------------	--

Instructions can require between one and eight words in memory. The length of an instruction depends not only on the instruction, but also on the operands used for the instruction. If an index register is addressed directly or a data register is used as an operand, the instruction will require one word less than when specifying a word address for the operand. If a constant is designated for instructions that use 2-word operands, the instruction will require one word more than when specifying a word address for the operand. The possible lengths for each instruction are provided in *Section 6 Program Execution Timing*.

Program Memory addresses start at 00000 and run until the capacity of Program Memory has been exhausted. The first word at each address defines the instruction. Any definers used by the instruction are placed on the same line of code. Also, if an instruction requires only a single bit operand (with no definer), the bit operand is also placed on the same line as the instruction. The rest of the words required by an instruction contain the operands that specify what data is to be

used. All other instructions are written with the instruction on the first line followed by the operands one to a line. An example of mnemonic code is shown below. The instructions used in it are described later in the manual. When inputting programs in mnemonic form from the SSS, most operands are separated only by spaces. Refer to the *SSS Operation Manuals* for details.

Address	Instruction	Operands
00000	LD	000000
00001	AND	000001
00002	OR	000002
00003	LD NOT	000100
00004	AND	000101
00005	AND LD	000102
00006	MOV(030)	
		0000
		D00000
00007	CMP(020)	
		D00000
		0000
00008	LD	025505
00009	OUT	000501
00010	MOV(030)	
		D00000
		D00500
00011	DIFU(013)	000502
00012	AND	000005
00013	OUT	000503

The address and instruction columns of the mnemonic code table are filled in for the instruction word only. For all other lines, the left two columns are left blank. If the instruction requires no definer or bit operand, the operand column is left blank for first line. It is a good idea to cross through any blank data column spaces (for all instruction words that do not require data) so that the data column can be quickly scanned to see if any addresses have been left out.

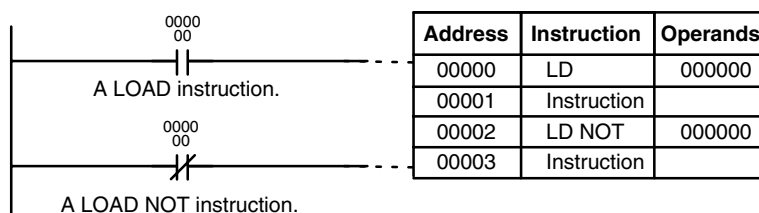
When programming, addresses are automatically displayed and do not have to be input unless for some reason a different location is desired for the instruction. When converting to mnemonic code, it is best to start at Program Memory address 00000 unless there is a specific reason for starting elsewhere.

### 4-3-3 Ladder Instructions

The ladder instructions are those instructions that correspond to the conditions on the ladder diagram. Ladder instructions, either independently or in combination with the logic block instructions described later, form the execution conditions upon which the execution of all other instructions are based.

#### LOAD and LOAD NOT

The first condition that starts any logic block within a ladder diagram corresponds to a LOAD or LOAD NOT instruction. Each of these instructions is written on one line of mnemonic code. "Instruction" is used as a dummy instruction in the following examples and could be any of the right-hand instructions described later in this manual.

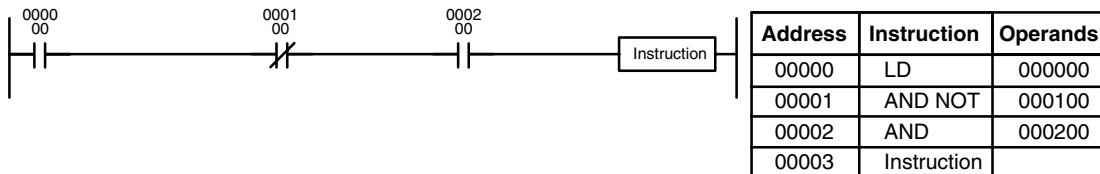




When this is the only condition on the instruction line, the execution condition for the instruction at the right is ON when the execution condition is ON. For the LOAD instruction (i.e., a normally open condition), an ON execution condition would be produced when CIO 000000 was ON; for the LOAD NOT instruction (i.e., a normally closed condition), an ON execution condition would be produced when CIO 000000 was OFF.

**AND and AND NOT**

When two or more conditions lie in series on the same instruction line, the first one corresponds to a LOAD or LOAD NOT instruction, and the rest of the conditions correspond to AND or AND NOT instructions. The following example shows three conditions which correspond in order from the left to a LOAD, an AND NOT, and an AND instruction. Again, each of these instructions is written on one line of mnemonic code.



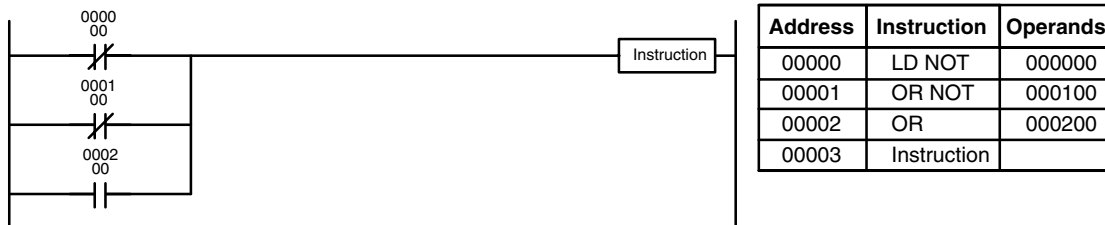
The instruction would have an ON execution condition only when CIO 000000 was ON, CIO 000100 was OFF, and CIO 000200 was ON.

AND instructions in series can be considered individually, with each taking the logical AND of the execution condition produced by the preceding instruction and the status of the AND instruction's operand bit. If both of these are ON, an ON execution condition will be produced for the next instruction. If either is OFF, the resulting execution condition will also be OFF.

Each AND NOT instruction in a series would take the logical AND between the execution condition produced by the preceding instruction and the inverse of its operand bit.

**OR and OR NOT**

When two or more conditions lie on separate instruction lines running in parallel and then joining together, the first condition corresponds to a LOAD or LOAD NOT instruction; the rest of the conditions correspond to OR or OR NOT instructions. The following example shows three conditions which correspond in order from the top to a LOAD NOT, an OR NOT, and an OR instruction. Again, each of these instructions requires one line of mnemonic code.

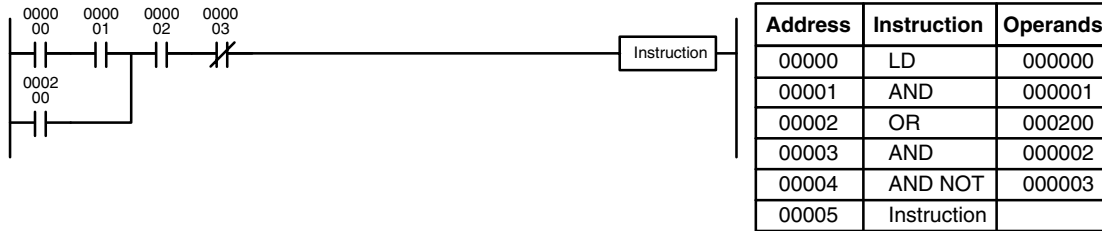


The instruction at the right would have an ON execution condition when any one of the three conditions was ON, i.e., when CIO 000000 was OFF, when CIO 00100 was OFF, or when CIO 000200 was ON.

OR and OR NOT instructions can be considered individually, each taking the logical OR between the execution condition produced by the preceding instructions and the status of the OR instruction's operand bit. If either one of these were ON, an ON execution condition would be produced for the next instruction.

**Combining AND and OR Instructions**

When AND and OR instructions are combined in more complicated diagrams, they can sometimes be considered individually, with each instruction performing a logic operation on the current execution condition and the status of the operand bit. The following is one example. Study this example until you are convinced that the mnemonic code follows the same logic flow as the ladder diagram.

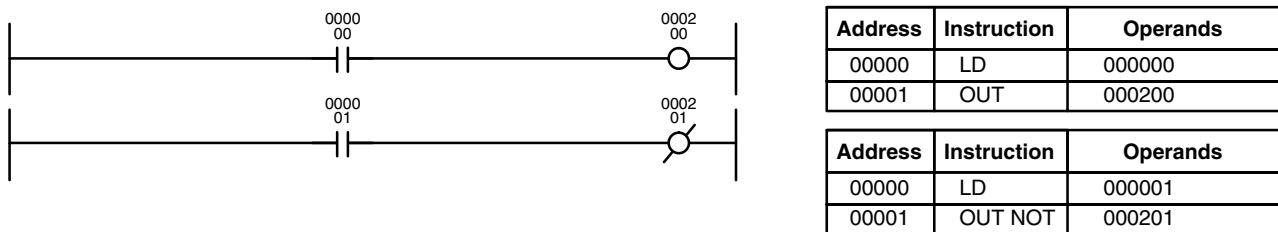


Here, an AND is taken between the status of CIO 000000 and that of CIO 000001 to determine the execution condition for an OR with the status of CIO 000200. The result of this operation determines the execution condition for an AND with the status of CIO 000002, which in turn determines the execution condition for an AND with the inverse (i.e., and AND NOT) of the status of CIO 000003.

In more complicated diagrams, it is necessary to consider logic blocks before an execution condition can be determined for the final instruction, and that’s where AND LOAD and OR LOAD instructions are used. Before we consider more complicated diagrams, however, we’ll look at the instructions required to complete a simple “input-output” program.

**4-3-4 OUTPUT and OUTPUT NOT**

The simplest way to output the results of combining execution conditions is to output it directly with the OUTPUT and OUTPUT NOT. These instructions are used to control the status of the designated operand bit according to the execution condition. With the OUTPUT instruction, the operand bit will be turned ON as long as the execution condition is ON and will be turned OFF as long as the execution condition is OFF. With the OUTPUT NOT instruction, the operand bit will be turned ON as long as the execution condition is OFF and turned OFF as long as the execution condition is ON. These appear as shown below. In mnemonic code, each of these instructions requires one line.

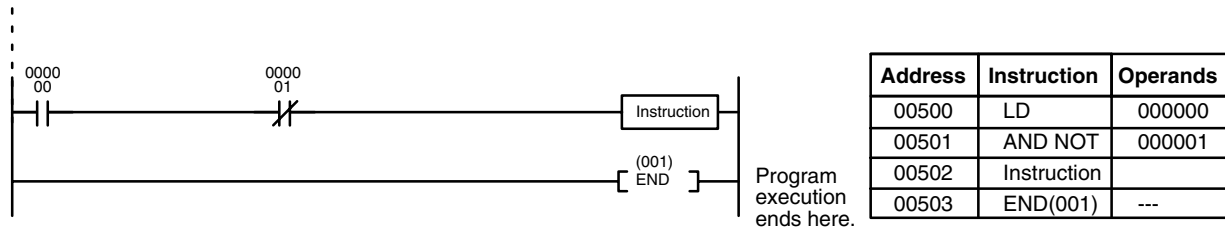


In the above examples, CIO 000200 will be ON as long as CIO 000000 is ON and CIO 000201 will be ON as long as CIO 000001 is OFF. Here, CIO 000000 and CIO 000001 would be input bits and CIO 000200 and CIO 000201 output bits assigned to the Units controlled by the PC, i.e., the signals coming in through the input points assigned CIO 000000 and CIO 000001 are controlling the output points to which CIO 000200 and CIO 000201 are allocated.

The length of time that a bit is ON or OFF can be controlled by combining the OUTPUT or OUTPUT NOT instruction with Timer instructions. Refer to Examples under 5-13-1 Timer – TIM for details.

### 4-3-5 The END Instruction

The last instruction required to complete any program is the END instruction. When the CPU scans a program, it executes all instructions up to the first END instruction before returning to the beginning of the program and beginning execution again. Although an END instruction can be placed at any point in a program, which is sometimes done when debugging, no instructions past the first END instruction will be executed. The number following the END instruction in the mnemonic code is its function code, which is used when inputting most instructions into the PC. Function codes are described in more detail later. The END instruction requires no operands and no conditions can be placed on the instruction line with it.



If there is no END instruction anywhere in a program, the program will not be executed at all.

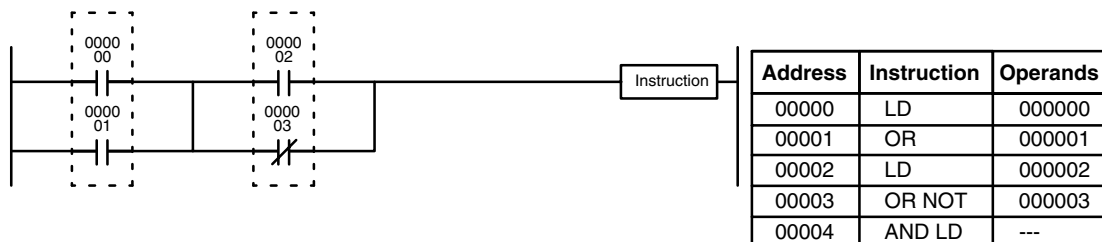
## 4-4 Mnemonic Code

### 4-4-1 Logic Block Instructions

Logic block instructions do not correspond to specific conditions on the ladder diagram; rather, they describe relationships between logic blocks. Each logic block is started with a LOAD or LOAD NOT instruction. Whenever a LOAD or LOAD NOT instruction is executed, a new execution condition is created and the previous execution condition is stored in a buffer. The AND LOAD instruction logically ANDs the execution conditions produced by two logic blocks, i.e., general speaking, it ANDs the current execution condition with the last execution condition stored in a buffer. The OR LOAD instruction logically ORs the execution conditions produced by two logic blocks.

#### AND LOAD

Although simple in appearance, the diagram below requires an AND LOAD instruction.



The two logic blocks are indicated by dotted lines. Studying this example shows that an ON execution condition will be produced when: either of the conditions in the left logic block is ON (i.e., when either CIO 000000 or CIO 000001 is ON) **and** either of the conditions in the right logic block is ON (i.e., when either CIO 000002 is ON or CIO 000003 is OFF).

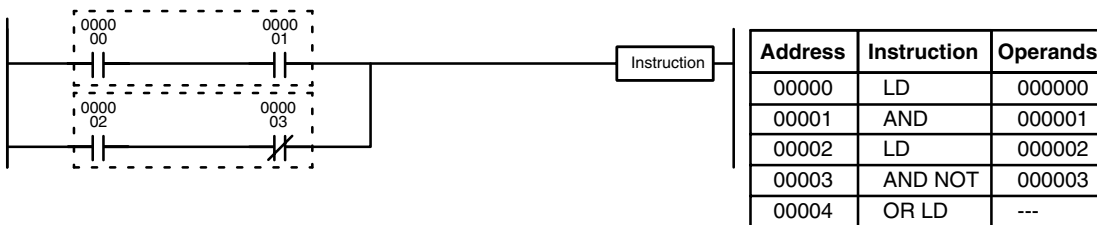
The above ladder diagram cannot be converted to mnemonic code using AND and OR instructions alone. If an AND between CIO 000002 and the results of an OR between CIO 000000 and CIO 000001 is attempted, the OR NOT between CIO 000002 and CIO 000003 is lost and the OR NOT ends up being an OR NOT between just CIO 000003 and the result of an AND between CIO 000002 and the first OR. What we need is a way to do the OR (NOT)'s independently and then combine the results.

To do this, we can use the LOAD or LOAD NOT instruction in the middle of an instruction line. When LOAD or LOAD NOT is executed in this way, the current execution condition is saved in special buffers and the logic process is begun over. To combine the results of the current execution condition with that of a previous "unused" execution condition, an AND LOAD or an OR LOAD instruction is used. Here "LOAD" refers to loading the last unused execution condition. An unused execution condition is produced by using the LOAD or LOAD NOT instruction for any but the first condition on an instruction line.

Analyzing the above ladder diagram in terms of mnemonic instructions, the condition for CIO 000000 is a LOAD instruction and the condition below it is an OR instruction between the status of CIO 000000 and that of CIO 000001. The condition at CIO 000002 is another LOAD instruction and the condition below is an OR NOT instruction, i.e., an OR between the status of CIO 000002 and the inverse of the status of CIO 000003. To arrive at the execution condition for the instruction at the right, the logical AND of the execution conditions resulting from these two blocks would have to be taken. AND LOAD does this. The mnemonic code for the ladder diagram is shown to the right of the diagram. The AND LOAD instruction requires no operands of its own, because it operates on previously determined execution conditions. Here too, dashes are used to indicate that no operands needs designated or input.

**OR LOAD**

The following diagram requires an OR LOAD instruction between the top logic block and the bottom logic block. An ON execution condition would be produced for the instruction at the right either when CIO 000000 is ON and CIO 000001 is OFF or when CIO 000002 and CIO 000003 are both ON. The operation of and mnemonic code for the OR LOAD instruction is exactly the same as those for a AND LOAD instruction except that the current execution condition is ORed with the last unused execution condition.



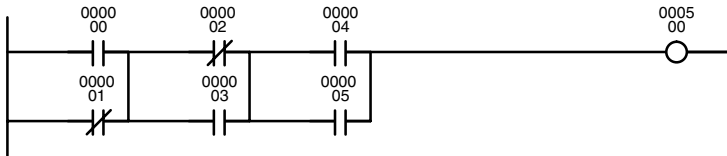
Naturally, some diagrams will require both AND LOAD and OR LOAD instructions.

**Logic Block Instructions in Series**

To code diagrams with logic block instructions in series, the diagram must be divided into logic blocks. Each block is coded using a LOAD instruction to code the first condition, and then AND LOAD or OR LOAD is used to logically combine the blocks. With both AND LOAD and OR LOAD there are two ways to achieve this. One is to code the logic block instruction after the first two blocks and then after each additional block. The other is to code all of the blocks to be combined, starting each block with LOAD or LOAD NOT, and then to code the logic block instructions which combine them. In this case, the instructions for the last pair of blocks should be combined first, and then each preceding block should be combined, working progressively back to the first block. Although either of these methods will produce exactly the same result, the second method, that of coding

all logic block instructions together, can be used only if eight or fewer blocks are being combined, i.e., if seven or fewer logic block instructions are required.

The following diagram requires AND LOAD to be converted to mnemonic code because three pairs of parallel conditions lie in series. The two means of coding the programs are also shown.



Address	Instruction	Operands
00000	LD	000000
00001	OR NOT	000001
00002	LD NOT	000002
00003	OR	000003
00004	AND LD	---
00005	LD	000004
00006	OR	000005
00007	AND LD	---
00008	OUT	000500

Address	Instruction	Operands
00000	LD	000000
00001	OR NOT	000001
00002	LD NOT	000002
00003	OR	000003
00004	LD	000004
00005	OR	000005
00006	AND LD	---
00007	AND LD	---
00008	OUT	000500

Again, with the second method, a maximum of eight blocks can be combined. There is no limit to the number of blocks that can be combined with the first method.

The following diagram requires OR LOAD instructions to be converted to mnemonic code because three pairs of conditions in series lie in parallel to each other.



Address	Instruction	Operands
00000	LD	000000
00001	AND NOT	000001
00002	LD NOT	000002
00003	AND NOT	000003
00004	OR LD	—
00005	LD	000004
00006	AND	000005
00007	OR LD	—
00008	OUT	000501

Address	Instruction	Operands
00000	LD	000000
00001	AND NOT	000001
00002	LD NOT	000002
00003	AND NOT	000003
00004	LD	000004
00005	AND	000005
00006	OR LD	—
00007	OR LD	—
00008	OUT	000501

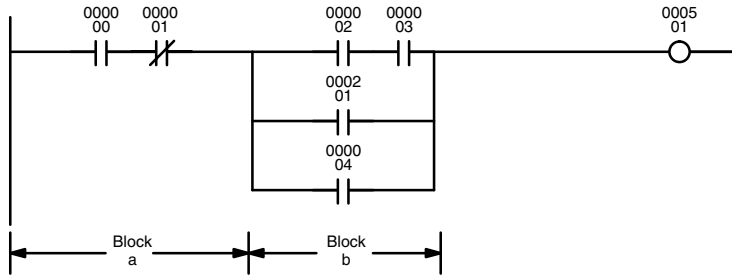
The first of each pair of conditions is converted to LOAD with the assigned bit operand and then ANDed with the other condition. The first two blocks can be coded first, followed by OR LOAD, the last block, and another OR LOAD, or the three blocks can be coded first followed by two OR LOADS. The mnemonic code for both methods is shown to the right of the ladder diagram.

Again, with the second method, a maximum of eight blocks can be combined. There is no limit to the number of blocks that can be combined with the first method.

**Combining AND LOAD and OR LOAD**

Both of the coding methods described above can also be used when using AND LOAD and OR LOAD, as long as the number of blocks being combined does not exceed eight.

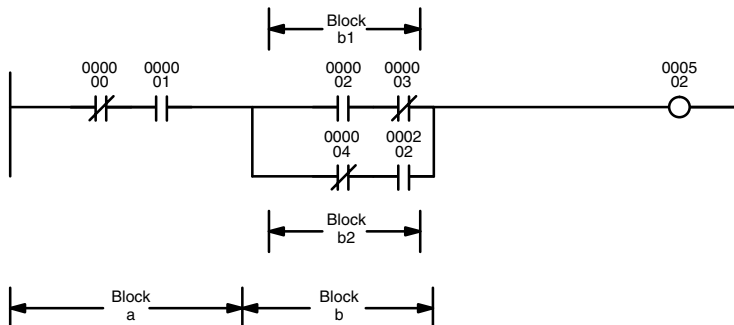
The following diagram contains only two logic blocks as shown. It is not necessary to further separate block b components, because it can be coded directly using only AND and OR.



Address	Instruction	Operands
00000	LD	000000
00001	AND NOT	000001
00002	LD	000002
00003	AND	000003
00004	OR	000201
00005	OR	000004
00006	AND LD	—
00007	OUT	000501

Although the following diagram is similar to the one above, block b in the diagram below cannot be coded without separating it into two blocks combined with OR LOAD. In this example, the three blocks have been coded first and then OR LOAD has been used to combine the last two blocks followed by AND LOAD to combine the execution condition produced by the OR LOAD with the execution condition of block a.

When coding the logic block instructions together at the end of the logic blocks they are combining, they must, as shown below, be coded in reverse order, i.e., the logic block instruction for the last two blocks is coded first, followed by the one to combine the execution condition resulting from the first logic block instruction and the execution condition of the logic block third from the end, and on back to the first logic block that is being combined.



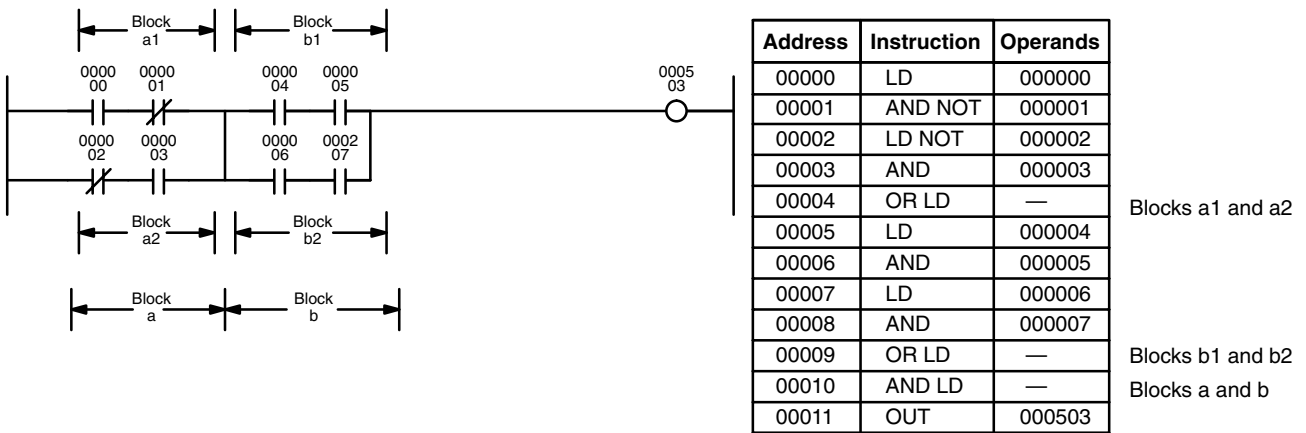
Address	Instruction	Operands
00000	LD NOT	000000
00001	AND	000001
00002	LD	000002
00003	AND NOT	000003
00004	LD NOT	000004
00005	AND	000202
00006	OR LD	—
00007	AND LD	—
00008	OUT	000502

**Complicated Diagrams**

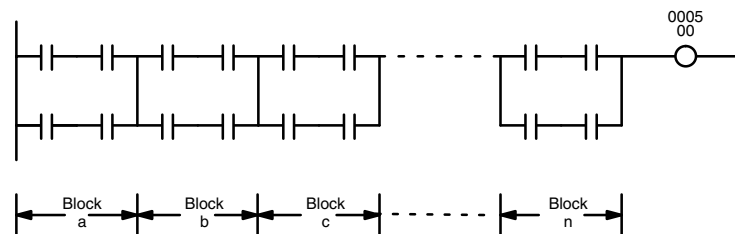
When determining what logic block instructions will be required to code a diagram, it is sometimes necessary to break the diagram into large blocks and then continue breaking the large blocks down until logic blocks that can be coded without logic block instructions have been formed. These blocks are then coded, combining the small blocks first, and then combining the larger blocks. Either AND LOAD or OR LOAD is used to combine the blocks, i.e., AND LOAD or OR LOAD always combines the last two execution conditions that existed, regardless of whether the execution conditions resulted from a single condition, from logic blocks, or from previous logic block instructions.

When working with complicated diagrams, blocks will ultimately be coded starting at the top left and moving down before moving across. This will generally mean that, when there might be a choice, OR LOAD will be coded before AND LOAD.

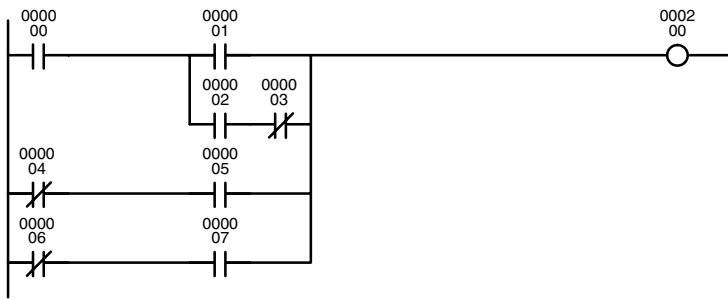
The following diagram must be broken down into two blocks and each of these then broken into two blocks before it can be coded. As shown below, blocks a and b require an AND LOAD. Before AND LOAD can be used, however, OR LOAD must be used to combine the top and bottom blocks on both sides, i.e., to combine a1 and a2; b1 and b2.



The following type of diagram can be coded easily if each block is coded in order: first top to bottom and then left to right. In the following diagram, blocks a and b would be combined using AND LOAD as shown above, and then block c would be coded and a second AND LOAD would be used to combined it with the execution condition from the first AND LOAD. Then block d would be coded, a third AND LOAD would be used to combine the execution condition from block d with the execution condition from the second AND LOAD, and so on through to block n.

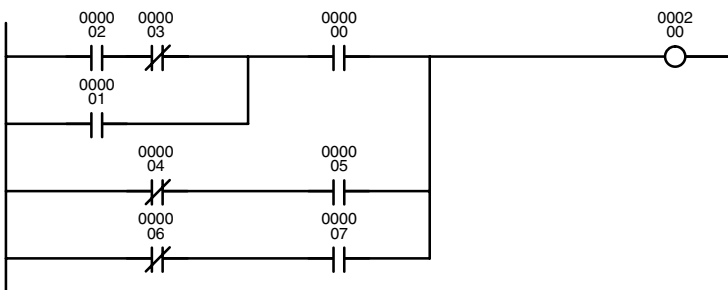


The following diagram requires an OR LOAD followed by an AND LOAD to code the top of the three blocks, and then two more OR LOADs to complete the mnemonic code.



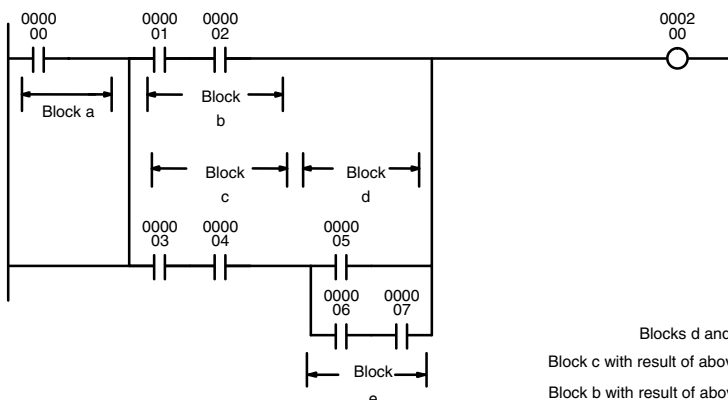
Address	Instruction	Operands
00000	LD	000000
00001	LD	000001
00002	LD	000002
00003	AND NOT	000003
00004	OR LD	--
00005	AND LD	--
00006	LD NOT	000004
00007	AND	000005
00008	OR LD	--
00009	LD NOT	000006
00010	AND	000007
00011	OR LD	--
00012	OUT	000200

Although the program will execute as written, this diagram could be drawn as shown below to eliminate the need for the first OR LOAD and the AND LOAD, simplifying the program and saving memory space.



Address	Instruction	Operands
00000	LD	000002
00001	AND NOT	000003
00002	OR	000001
00003	AND	000000
00004	LD NOT	000004
00005	AND	000005
00006	OR LD	--
00007	LD NOT	000006
00008	AND	000007
00009	OR LD	--
00010	OUT	000200

The following diagram requires five blocks, which here are coded in order before using OR LOAD and AND LOAD to combine them starting from the last two blocks and working backward. The OR LOAD at program address 00008 combines blocks blocks d and e, the following AND LOAD combines the resulting execution condition with that of block c, etc.

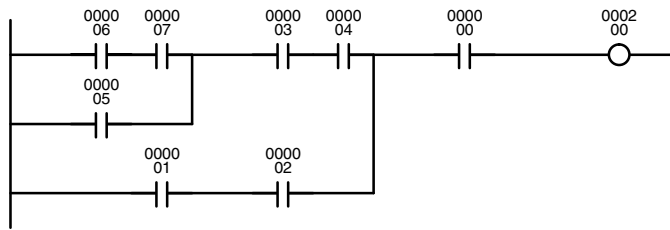


Blocks d and e  
Block c with result of above  
Block b with result of above  
Block a with result of above

Address	Instruction	Operands
00000	LD	000000
00001	LD	000001
00002	AND	000002
00003	LD	000003
00004	AND	000004
00005	LD	000005
00006	LD	000006
00007	AND	000007
00008	OR LD	--
00009	AND LD	--
00010	OR LD	--
00011	AND LD	--
00012	OUT	000200

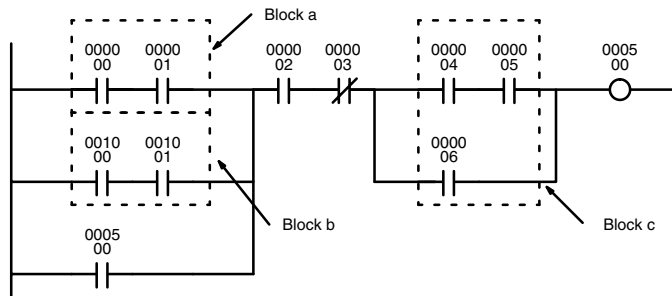


Again, this diagram can be redrawn as follows to simplify program structure and coding and to save memory space.

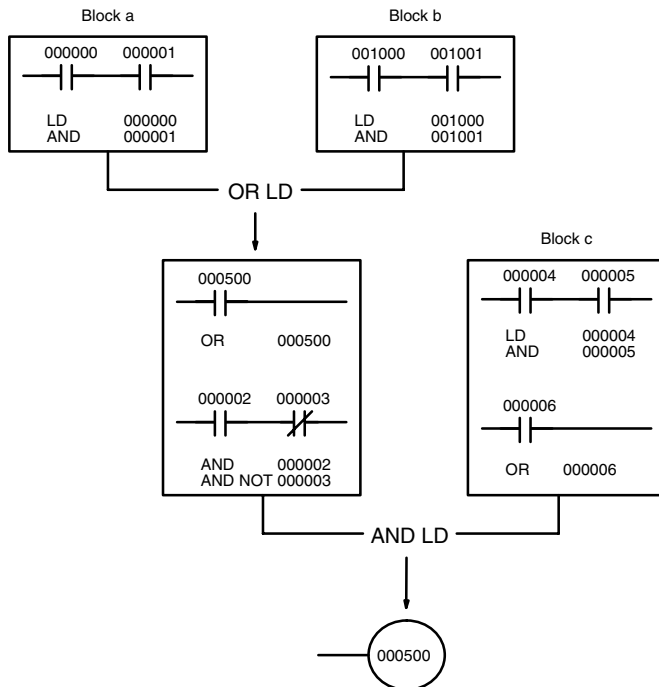


Address	Instruction	Operands
00000	LD	000006
00001	AND	000007
00002	OR	000005
00003	AND	000003
00004	AND	000004
00005	LD	000001
00006	AND	000002
00007	OR LD	--
00008	AND	000000
00009	OUT	000200

The next and final example may at first appear very complicated but can be coded using only two logic block instructions. The diagram appears as follows:



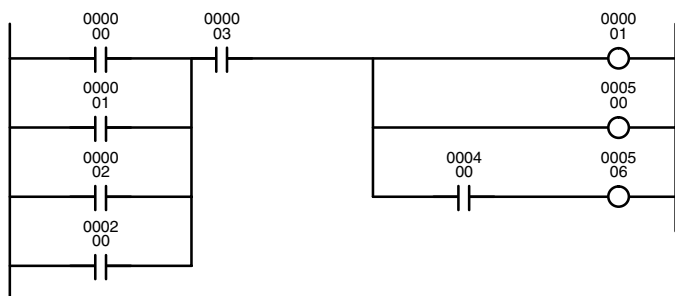
The first logic block instruction is used to combine the execution conditions resulting from blocks a and b, and the second one is to combine the execution condition of block c with the execution condition resulting from the normally closed condition assigned CIO 000003. The rest of the diagram can be coded with OR, AND, and AND NOT instructions. The logical flow for this and the resulting code are shown below.



Address	Instruction	Operands
00000	LD	000000
00001	AND	000001
00002	LD	001000
00003	AND	001001
00004	OR LD	--
00005	OR	000500
00006	AND	000002
00007	AND NOT	000003
00008	LD	000004
00009	AND	000005
00010	OR	000006
00011	AND LD	--
00012	OUT	000500

### 4-4-2 Coding Multiple Right-hand Instructions

If there is more than one right-hand instruction executed with the same execution condition, they are coded consecutively following the last condition on the instruction line. In the following example, the last instruction line contains one more condition that corresponds to an AND with CIO 000400.



Address	Instruction	Operands
00000	LD	000000
00001	OR	000001
00002	OR	000002
00003	OR	000200
00004	AND	000003
00005	OUT	000001
00006	OUT	000500
00007	AND	000400
00008	OUT	000506

## 4-5 Branching Instruction Lines

When an instruction line branches into two or more lines, it is sometimes necessary to use either interlocks or TR bits to maintain the execution condition that existed at a branching point. This is because instruction lines are executed across to a right-hand instruction before returning to the branching point to execute instructions one a branch line. If a condition exists on any of the instruction lines after the branching point, the execution condition could change during this time making proper execution impossible. The following diagrams illustrate this. In both diagrams, instruction 1 is executed before returning to the branching point and moving on to the branch line leading to instruction 2.

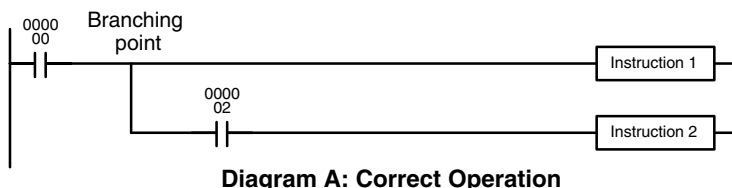


Diagram A: Correct Operation

Address	Instruction	Operands
00000	LD	000000
00001	Instruction 1	
00002	AND	000002
00003	Instruction 2	

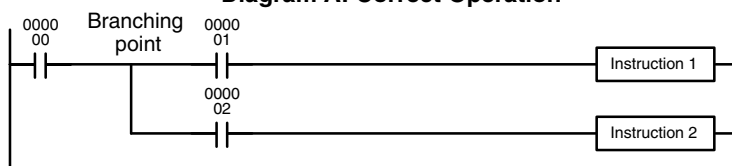


Diagram B: Incorrect Operation

Address	Instruction	Operands
00000	LD	000000
00001	AND	000001
00002	Instruction 1	
00003	AND	000002
00004	Instruction 2	

If, as shown in diagram A, the execution condition that existed at the branching point cannot be changed before returning to the branch line (instructions at the far right do not change the execution condition), then the branch line will be executed correctly and no special programming measure is required.

If, as shown in diagram B, a condition exists between the branching point and the last instruction on the top instruction line, the execution condition at the branching point and the execution condition after completing the top instruction line will sometimes be different, making it impossible to ensure correct execution of the branch line.

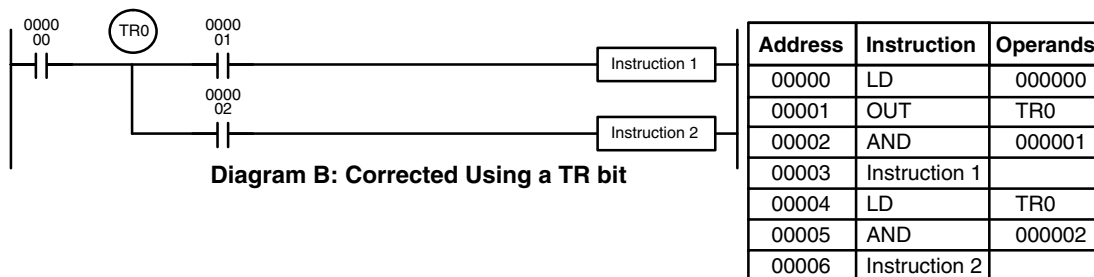
There are two means of programming branching programs to preserve the execution condition. One is to use TR bits; the other, to use interlocks (IL(002)/ILC(003)).

### 4-5-1 TR Bits

The TR area provides eight bits, TR0 through TR7, that can be used to temporarily preserve execution conditions. If a TR bit is placed at a branching point, the current execution condition will be stored at the designated TR bit. When returning to the branching point, the TR bit restores the execution status that was saved when the branching point was first reached in program execution.

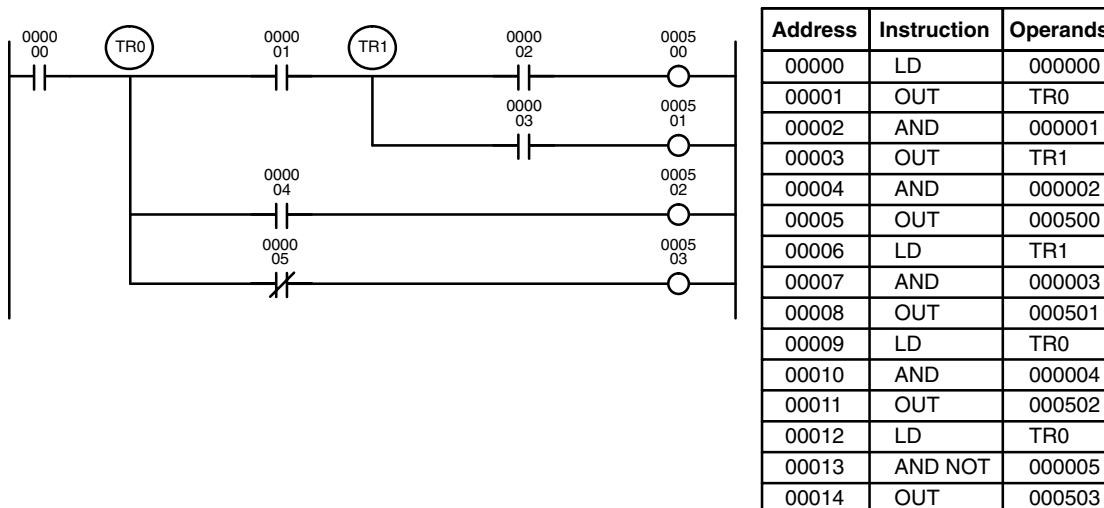
**Note** When programming in graphic ladder diagram form from the SSS, it is not necessary to input TR bits and none will appear on the screen. The SSS will automatically process TR bits for you as required and input them into the program. You will have to input TR bit when programming in mnemonic form.

The previous diagram B can be written as shown below to ensure correct execution. In mnemonic code, the execution condition is stored at the branching point using the TR bit as the operand of the OUTPUT instruction. This execution condition is then restored after executing the right-hand instruction by using the same TR bit as the operand of a LOAD instruction.



In terms of actual instructions the above diagram would be as follows: The status of CIO 000000 is loaded (a LOAD instruction) to establish the initial execution condition. This execution condition is then output using an OUTPUT instruction to TR0 to store the execution condition at the branching point. The execution condition is then ANDed with the status of CIO 000001 and instruction 1 is executed accordingly. The execution condition that was stored at the branching point is then re-loaded (a LOAD instruction with TR0 as the operand), this is ANDed with the status of CIO 000002, and instruction 2 is executed accordingly.

The following example shows an application using two TR bits.

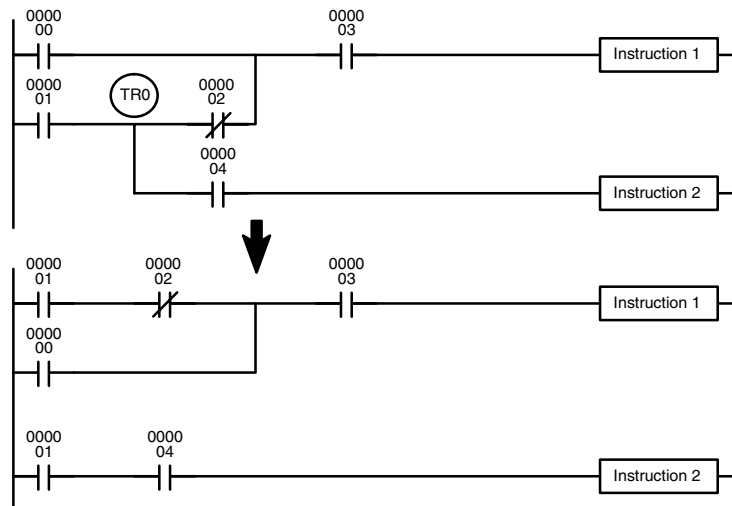
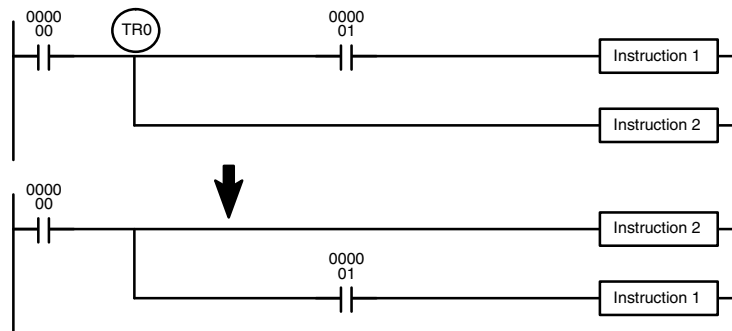


In this example, TR0 and TR1 are used to store the execution conditions at the branching points. After executing instruction 1, the execution condition stored in TR1 is loaded for an AND with the status CIO 000003. The execution condition stored in TR0 is loaded twice, the first time for an AND with the status of CIO 000004 and the second time for an AND with the inverse of the status of CIO 000005.

TR bits can be used as many times as required as long as the same TR bit is not used more than once in the same instruction block. A new instruction block is begun each time execution returns to the bus bar. If, in a single instruction block, it is necessary to have more than eight branching points that require the execution condition be saved, interlocks (which are described next) must be used.

When drawing a ladder diagram, be careful not to use TR bits unless necessary. Often the number of instructions required for a program can be reduced and ease of understanding a program increased by redrawing a diagram that would otherwise require TR bits. In both of the following pairs of diagrams, the bottom versions require fewer instructions and do not require TR bits. In the first example, this is achieved by reorganizing the parts of the instruction block: the bottom one, by separating the second OUTPUT instruction and using another LOAD instruction to create the proper execution condition for it.

**Note** Although simplifying programs is always a concern, the order of execution of instructions is sometimes important. For example, a MOVE instruction may be required before the execution of a BINARY ADD instruction to place the proper data in the required operand word. Be sure that you have considered execution order before reorganizing a program to simplify it.

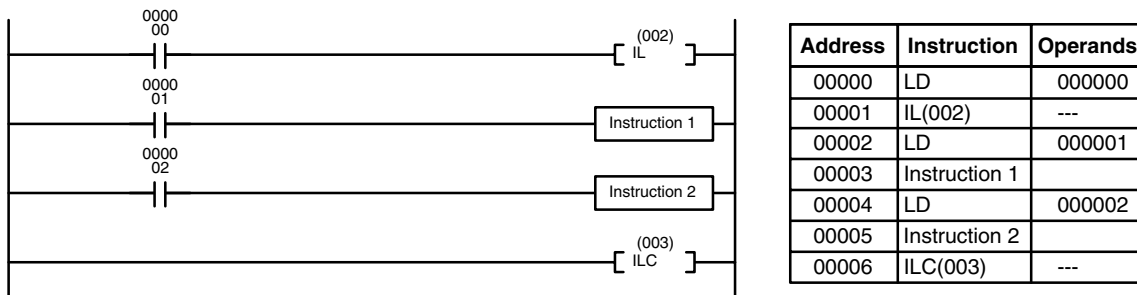


### 4-5-2 Interlocks

The problem of storing execution conditions at branching points can also be handled by using the INTERLOCK (IL(002)) and INTERLOCK CLEAR (ILC(003)) instructions to eliminate the branching point completely while allowing a specific execution condition to control a group of instructions. The INTERLOCK and INTERLOCK CLEAR instructions are always used together.

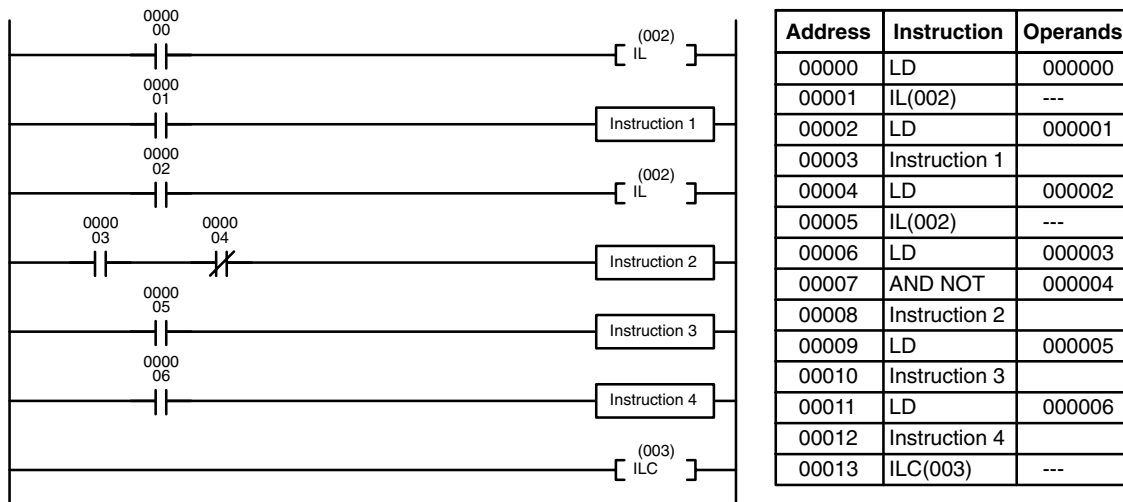
When an INTERLOCK instruction is placed before a section of a ladder program, the execution condition for the INTERLOCK instruction will control the execution of all instruction up to the next INTERLOCK CLEAR instruction. If the execution condition for the INTERLOCK instruction is OFF, all right-hand instructions through the next INTERLOCK CLEAR instruction will be executed with OFF execution conditions to reset the entire section of the ladder diagram. The effect that this has on particular instructions is described in 5-8 INTERLOCK and INTERLOCK CLEAR - IL(002) and ILC(003).

Diagram B can also be corrected with an interlock. Here, the conditions leading up to the branching point are placed on an instruction line for the INTERLOCK instruction, all of lines leading from the branching point are written as separate instruction lines, and another instruction line is added for the INTERLOCK CLEAR instruction. No conditions are allowed on the instruction line for INTERLOCK CLEAR. INTERLOCK and INTERLOCK CLEAR does not use operands.



If CIO 000000 is ON in the revised version of diagram B, above, the status of CIO 000001 and that of CIO 000002 would determine the execution conditions for instructions 1 and 2, respectively. Because CIO 000000 is ON, this would produce the same results as ANDing the status of each of these bits. If CIO 000000 is OFF, the INTERLOCK instruction would produce an OFF execution condition for instructions 1 and 2 and then execution would continue with the instruction line following the INTERLOCK CLEAR instruction.

As shown below, multiple INTERLOCK instructions can be used in one instruction block; each is effective through the next INTERLOCK CLEAR instruction.



If CIO 000000 in the above diagram is OFF (i.e., if the execution condition for the first INTERLOCK instruction is OFF), instructions 1 through 4 would be executed with OFF execution conditions and execution would move to the instruction following the INTERLOCK CLEAR instruction. If CIO 000000 is ON, the status of CIO 000001 would be loaded as the execution condition for instruction 1 and then the status of CIO 000002 would be loaded to form the execution condition for the second INTERLOCK instruction. If CIO 000002 is OFF, instructions 2 through 4 will be executed with OFF execution conditions. If CIO 000002 is ON, CIO 000003, CIO 000005, and CIO 000006 will determine the first execution condition in new instruction lines.

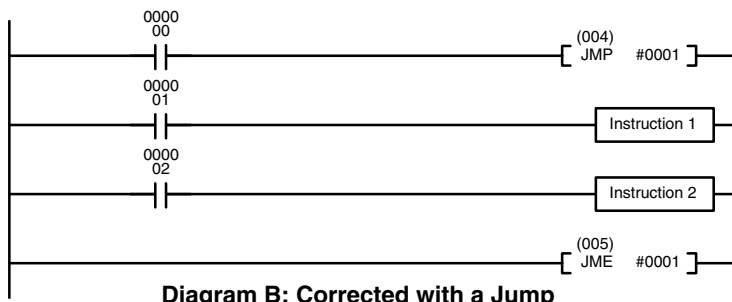
## 4-6 Jumps

A specific section of a program can be skipped according to a designated execution condition. Although this is similar to what happens when the execution condition for an INTERLOCK instruction is OFF, with jumps, the operands for all instructions maintain status. Jumps can therefore be used to control devices that require a sustained output, e.g., pneumatics and hydraulics, whereas interlocks can be used to control devices that do not require a sustained output, e.g., electronic instruments.

Jumps are created using the JUMP (JMP(004)) and JUMP END (JME(005)) instructions. If the execution condition for a JUMP instruction is ON, the program is executed normally as if the jump did not exist. If the execution condition for the JUMP instruction is OFF, program execution moves immediately to a JUMP END instruction without changing the status of anything between the JUMP and JUMP END instruction.

All JUMP and JUMP END instructions are assigned jump numbers ranging between 0000 and 0999. A jump can be defined once using any of the jump numbers 0000 through 0999. When a JUMP instruction assigned one of these numbers is executed, execution moves immediately to the JUMP END instruction that has the same number as if all of the instruction between them did not exist. The JUMP END instruction may be either before or after the JUMP instruction. Diagram B from the TR bit and interlock example could be redrawn as shown below using a jump. Although 0001 has been used as the jump number, any number between 0001 and 0999 could be used. JUMP and JUMP END require

no other operand and JUMP END never has conditions on the instruction line leading to it.



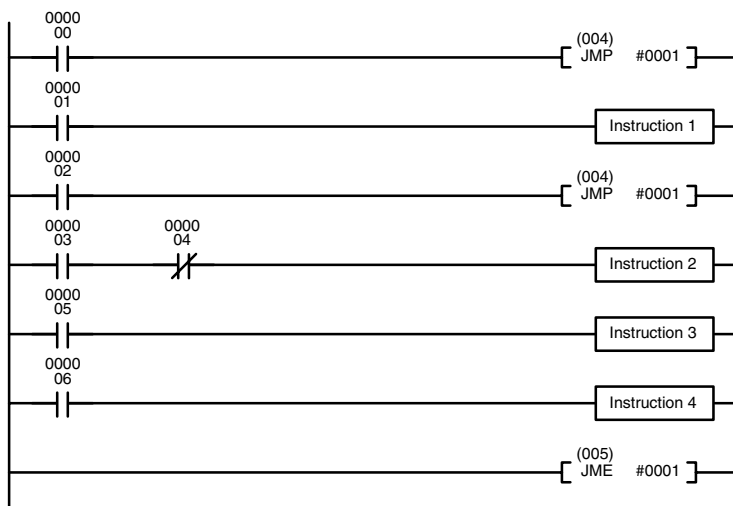
Address	Instruction	Operands
00000	LD	000000
00001	JMP(004)	#0001
00002	LD	000001
00003	Instruction 1	
00004	LD	000002
00005	Instruction 2	
00006	JME(005)	#0001

Diagram B: Corrected with a Jump

This version of diagram B would have a shorter execution time when 00000 was OFF than any of the other versions.

There must be a JUMP END with the same jump number for each JUMP instruction in the program.

The same jump number cannot be used in more than one JUMP END instruction. If you include more than one JUMP END instruction with the same jump number, all JUMP instructions with that jump number will jump to the first JUMP END instruction in the program with the same jump number. An exception to this is when jump number 0000 is set for multiple usage in the PC Setup (see following explanation and page 444). The same jump number can be used in more than one JUMP instruction to jump to the same destination in the program. The following example illustrates a program with two jumps to the same destination.



Address	Instruction	Operands
00000	LD	000000
00001	JMP(004)	#0001
00002	LD	000001
00003	Instruction 1	
00004	LD	000002
00005	JMP(004)	#0001
00006	LD	000003
00007	AND NOT	000004
00008	Instruction 2	
00009	LD	000005
00010	Instruction 3	
00011	LD	000006
00012	Instruction 4	
00013	JME(005)	#0001



**Caution**

Because instructions are not examined when jumps are made in the program, differentiated outputs can remain ON for more than one cycle if programmed within the area of the program that is jumped.

**JUMP 0000**

The PC Setup can be used to control the operation of jumps created using jump number 0000. If multiple jumps with 0000 are disabled, jumps created with 0000 will operate as described above. If multiple jumps are enabled, any JMP 0000 instruction will jump to the next JME 0000 in the program (and not the first JME 0000 in the program). When multiple jumps for 0000 are enabled, you cannot overlap or nest the jumps, i.e., each JMP 0000 must be followed by a JME 0000 before the next JMP 0000 in the program and each JME 0000 must be followed by a JMP 0000 before the next JME 0000 in the program.

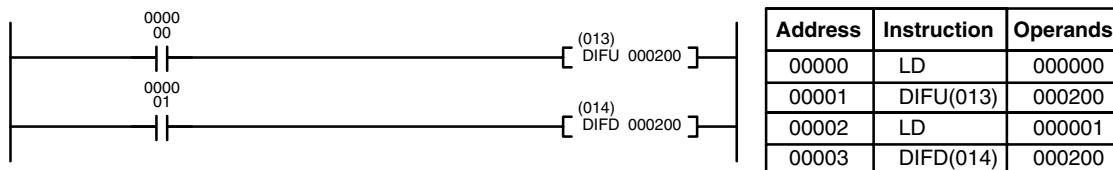
CVM1D CPUs also support the CJP(221) and CJPN(222) jump instructions that can also be used to create jumps in programs. Refer to *Section 5 Instruction Set* for details.

## 4-7 Controlling Bit Status

There are instructions that can be used generally to control individual bit status. These include the OUTPUT, OUTPUT NOT, DIFFERENTIATE UP, DIFFERENTIATE DOWN, SET, RESET and KEEP instructions. All of these instructions appear as the rightmost instruction in an instruction line and take a bit address for an operand. Although details are provided in *5-7 Bit Control Instructions*, these instructions (except for OUTPUT and OUTPUT NOT, which have already been introduced) are described here because of their importance in most programs. Although these instructions are used to turn ON and OFF output bits in the I/O Memory (i.e., to send or stop output signals to external devices), they are also used to control the status of other bits in the I/O memory or in other data areas.

### 4-7-1 DIFFERENTIATE UP and DIFFERENTIATE DOWN

DIFFERENTIATE UP and DIFFERENTIATE DOWN instructions are used to turn the operand bit ON for one scan at a time. The DIFFERENTIATE UP instruction turns ON the operand bit for one scan after the execution condition for it goes from OFF to ON; the DIFFERENTIATE DOWN instruction turns ON the operand bit for one scan after the execution condition for it goes from ON to OFF. Both of these instructions require only one line of mnemonic code.

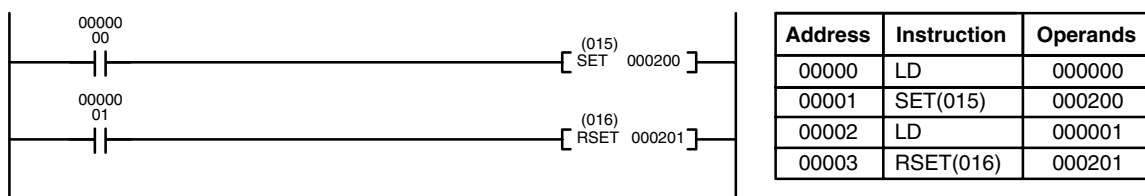


Here, CIO 000200 will be turned ON for one scan after CIO 000000 goes ON. The next time DIFU(013) 000200 is executed, CIO 000200 will be turned OFF, regardless of the status of CIO 000000. With the DIFFERENTIATE DOWN instruction, CIO 000200 will be turned ON for one scan after CIO 000001 goes OFF (CIO 000200 will be kept OFF until then), and will be turned OFF the next time DIFD(014) 000200 is executed.

CVM1D CPUs also provide UP(018) and DOWN(019) that can be used to differentiate changes in the execution condition to control execution. Refer to *Section 5 Instruction Set* for details.

### 4-7-2 SET and RESET

SET and RESET instructions are used to control the status of the operand bit while the execution condition for them is ON. When the execution condition is OFF, the status of the operand bit will not be changed.



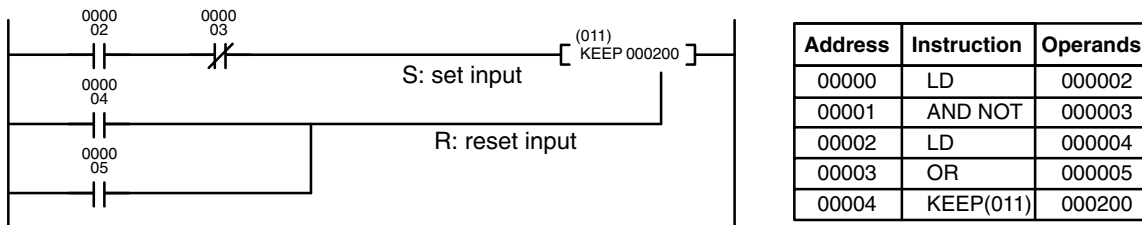
In the above example, CIO 000200 will be turned ON when CIO 000000 goes ON and will remain ON even after CIO 000000 goes OFF unless turned OFF somewhere else in the program. CIO 000201 will be turned OFF when CIO 000001 goes ON and will remain OFF even after CIO 000001 goes OFF unless turned ON somewhere else in the program.



### 4-7-3 KEEP

The KEEP instruction is used to maintain the status of the operand bit based on two execution conditions. To do this, the KEEP instruction is connected to two instruction lines. When the execution condition at the end of the first instruction line is ON, the operand bit of the KEEP instruction is turned ON. When the execution condition at the end of the second instruction line is ON, the operand bit of the KEEP instruction is turned OFF. The operand bit for the KEEP instruction will maintain its ON or OFF status even if it is located in an interlocked section of the diagram.

In the following example, CIO 000200 will be turned ON when CIO 000002 is ON and CIO 000003 is OFF. CIO 000200 will then remain ON until either CIO 000004 or CIO 000005 turns ON. With KEEP, as with all instructions requiring more than one instruction line, the instruction lines are coded first before the instruction that they control.

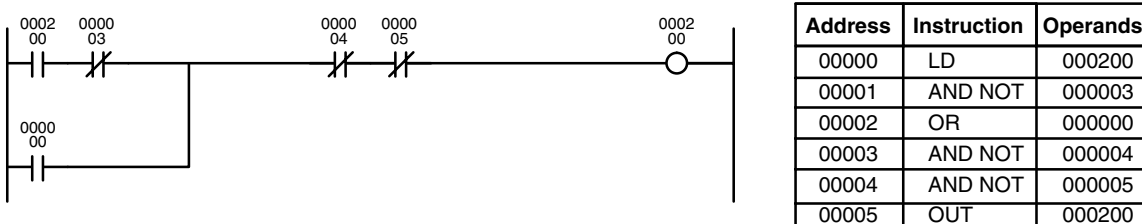


### 4-7-4 Self-maintaining Bits (Seal)

Although the KEEP instruction can be used to create self-maintaining bits, it is sometimes necessary to create self-maintaining bits in another way so that they can be turned OFF when in an interlocked section of a program.

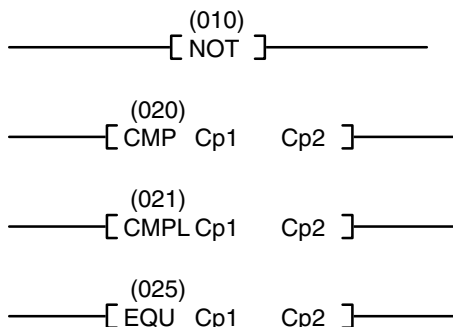
To create a self-maintaining bit, the operand bit of an OUTPUT instruction is used as a condition for the same OUTPUT instruction in an OR setup so that the operand bit of the OUTPUT instruction will remain ON or OFF until changes occur in other bits. At least one other condition is used just before the OUTPUT instruction to function as a reset. Without this reset, there would be no way to control the operand bit of the OUTPUT instruction.

The above diagram for the KEEP instruction can be rewritten as shown below. The only difference in these diagrams would be their operation in an interlocked program section when the execution condition for the INTERLOCK instruction was ON. Here, just as in the same diagram using the KEEP instruction, two reset bits are used, i.e., CIO 000200 can be turned OFF by turning ON either CIO 000004 or CIO 000005.



## 4-8 Intermediate Instructions

There are some instructions that can appear on instructions lines with conditions to help determine the execution conditions for other instructions. These instructions are called **intermediate instructions**. Intermediate instructions cannot be placed next to the right bus bar, only between conditions or between a condition and a right-hand instruction. The four instructions shown below, NOT(010), CMP(020), CMPL(021), and EQU(025), are intermediate instructions, and are described in *Section 5 Instruction Set*. The input comparison instructions described in *5-16-7 Input Comparison Instructions* are also intermediate instructions.



## 4-9 Work Bits (Internal Relays)

In programming, combining conditions to directly produce execution conditions is often extremely difficult. These difficulties are easily overcome, however, by using certain bits to trigger other instructions indirectly. Such programming is achieved by using work bits. Sometimes entire words are required for these purposes. These words are referred to as work words.

Work bits are not transferred to or from the PC. They are bits selected by the programmer to facilitate programming as described above. I/O bits and other dedicated bits cannot be used as work bits. All bits in the I/O Memory that are not allocated as I/O bits are available for use as work bits. Be careful to keep an accurate record of how and where you use work bits. This helps in program planning and writing, and also aids in debugging operations.

### Work Bit Applications

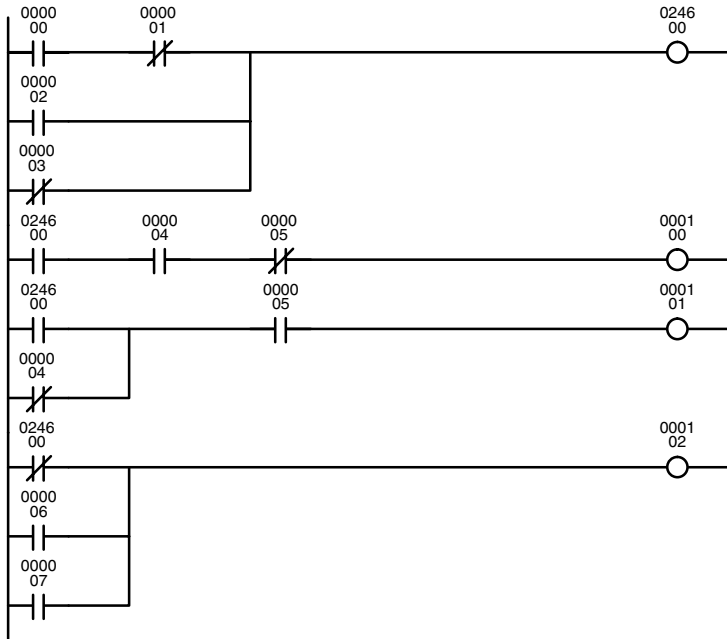
Examples given later in this subsection show two of the most common ways to employ work bits. These should act as a guide to the almost limitless number of ways in which the work bits can be used. Whenever difficulties arise in programming a control action, consideration should be given to work bits and how they might be used to simplify programming.

Work bits are often used with instructions that control bit status. The work bit is used first as the operand for one of these instructions so that later it can be used as a condition that will determine how other instructions will be executed. Work bits can also be used with other instructions, e.g., with the SHIFT REGISTER instruction (SFT(050)). An example of the use of work words and bits with the SHIFT REGISTER instruction is provided in *5-14-1 SHIFT REGISTER – SFT(050)*.

Although they are not always specifically referred to as work bits, many of the bits used in the examples in *Section 5 Instruction Set* use work bits. Understanding the use of these bits is essential to effective programming.

**Reducing Complex Conditions**

Work bits can be used to simplify programming when a certain combination of conditions is repeatedly used in combination with other conditions. In the following example, CIO 000000, CIO 000001, CIO 000002, and CIO 000003 are combined in a logic block that stores the resulting execution condition as the status of CIO 024600. CIO 024600 is then combined with various other conditions to determine output conditions for CIO 000100, CIO 000101, and CIO 000102, i.e., to turn the outputs allocated to these bits ON or OFF.

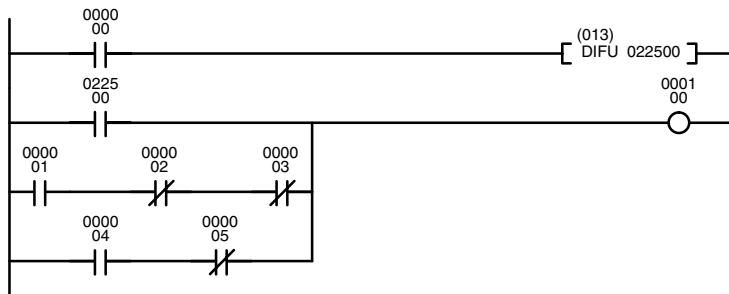


Address	Instruction	Operands
00000	LD	000000
00001	AND NOT	000001
00002	OR	000002
00003	OR NOT	000003
00004	OUT	024600
00005	LD	024600
00006	AND	000004
00007	AND NOT	000005
00008	OUT	000100
00009	LD	024600
00010	OR NOT	000004
00011	AND	000005
00012	OUT	000101
00013	LD NOT	024600
00014	OR	000006
00015	OR	000007
00016	OUT	000102

**Differentiated Conditions**

Work bits can also be used if differential treatment is necessary for some, but not all, of the conditions required for execution of an instruction. In this example, CIO 000100 must be left ON continuously as long as CIO 000001 is ON and both CIO 000002 and CIO 000003 are OFF, or as long as CIO 000004 is ON and CIO 000005 is OFF. It must be turned ON for only one scan each time CIO 000000 turns ON (unless one of the preceding conditions is keeping it ON continuously).

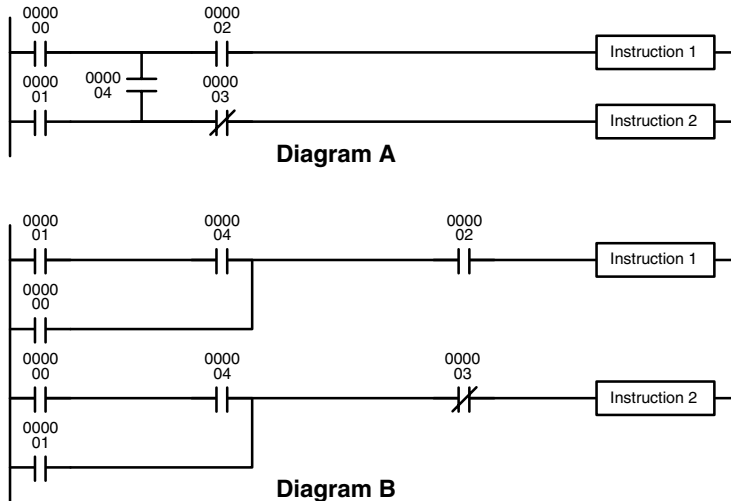
This action is easily programmed by using CIO 022500 as a work bit as the operand of the DIFFERENTIATE UP instruction (DIFU(013)). When CIO 000000 turns ON, CIO 022500 will be turned ON for one scan and then be turned OFF the next scan by DIFU(013). Assuming the other conditions controlling CIO 000100 are not keeping it ON, the work bit CIO 022500 will turn CIO 000100 ON for one scan only.



Address	Instruction	Operands
00000	LD	000000
00001	DIFU(013)	022500
00002	LD	022500
00003	LD	000001
00004	AND NOT	000002
00005	AND NOT	000003
00006	OR LD	---
00007	LD	000004
00008	AND NOT	000005
00009	OR LD	---
00010	OUT	000100

## 4-10 Programming Precautions

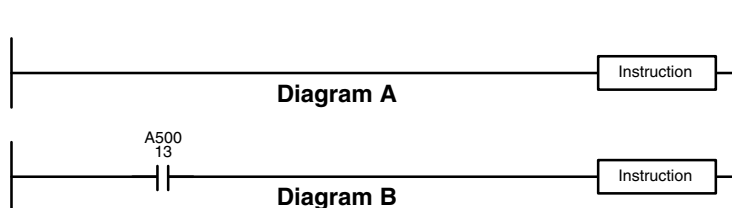
The number of conditions that can be used in series or parallel is unlimited as long as the memory capacity of the PC is not exceeded. Therefore, use as many conditions as required to draw clear diagrams. Although very complicated diagrams can be drawn with instruction lines, there must not be any conditions on lines running vertically between two other instruction lines. Diagram A shown below, for example, is not possible, and should be drawn as diagram B. Mnemonic code is provided for diagram B only; coding diagram A would be impossible.



Address	Instruction	Operands
00000	LD	000001
00001	AND	000004
00002	OR	000000
00003	AND	000002
00004	Instruction 1	
00005	LD	000000
00006	AND	000004
00007	OR	000001
00008	AND NOT	000003
00009	Instruction 2	

The number of times any particular bit can be assigned to conditions is not limited, so use them as many times as required to simplify your program. Often, complicated programs are the result of attempts to reduce the number of times a bit is used.

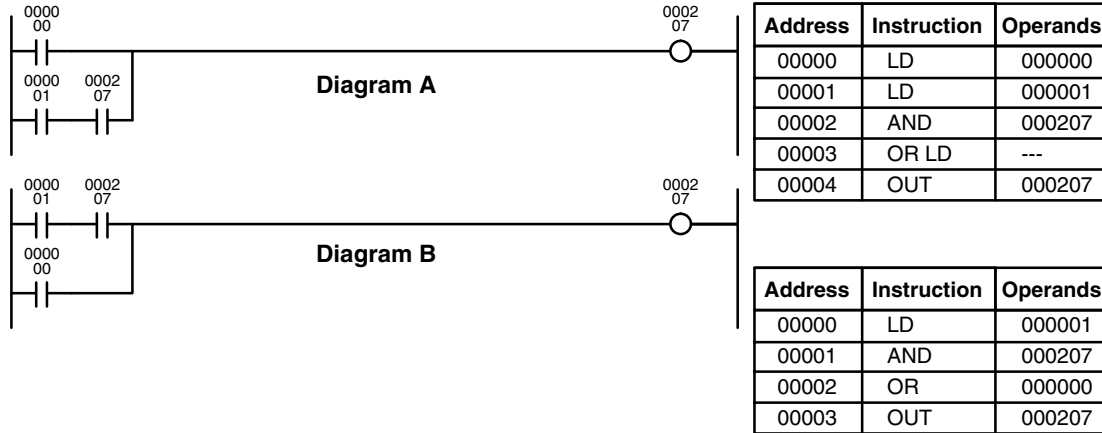
Except for instructions for which conditions are not allowed (e.g., INTERLOCK CLEAR and JUMP END, see below), every instruction line must also have at least one condition on it to determine the execution condition for the instruction at the right. Again, diagram A, below, must be drawn as diagram B. If an instruction must be continuously executed (e.g., if an output must always be kept ON while the program is being executed), the Always ON Flag (A50013) in the Auxiliary Area can be used.



Address	Instruction	Operands
00000	LD	A50013
00001	Instruction	

There are a few exceptions to this rule, including the INTERLOCK CLEAR, JUMP END, and step instructions. Each of these instructions is used as the second of a pair of instructions and is controlled by the execution condition of the first of the pair. Conditions should not be placed on the instruction lines leading to these instructions. Refer to *Section 5 Instruction Set* for details.

When drawing ladder diagrams, it is important to keep in mind the number of instructions that will be required to input it. In diagram A, below, an OR LOAD instruction will be required to combine the top and bottom instruction lines. This can be avoided by redrawing as shown in diagram B so that no AND LOAD or OR LOAD instructions are required. Refer to 5-6-5 AND LOAD and OR LOAD for more details and 4-4-1 Logic Block Instructions for further examples of diagrams requiring AND LOAD and OR LOAD.



### 4-11 Program Execution

When execution of a ladder diagram is started, the CPU scans the program from top to bottom, checking all conditions and executing all instructions accordingly as it moves down the bus bar. It is important that instructions be placed in the proper order so that, for example, the desired data is moved to a word before that word is used as the operand for an instruction. Remember that an instruction line is completed to the terminal instruction at the right before executing instruction lines branching from the first instruction line to other terminal instructions at the right.

Program execution is only one of the tasks carried out by the CPU as part of the cycle time. Refer to Section 6 Program Execution Timing for details.

### 4-12 Data Formats

The following data formats can be handled by the various calculation and conversion instructions.

- Unsigned binary
- Signed binary
- Unsigned BCD
- Signed BCD
- Floating-point

#### 4-12-1 Unsigned Binary Data

Data is configured in words, with 16 bits per word. This data is regarded as 16-bit binary data. Unsigned binary data is often written as four-digit hexadecimal (0000 to FFFF).

Bit	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
	2 <sup>15</sup>	2 <sup>14</sup>	2 <sup>13</sup>	2 <sup>12</sup>	2 <sup>11</sup>	2 <sup>10</sup>	2 <sup>9</sup>	2 <sup>8</sup>	2 <sup>7</sup>	2 <sup>6</sup>	2 <sup>5</sup>	2 <sup>4</sup>	2 <sup>3</sup>	2 <sup>2</sup>	2 <sup>1</sup>	2 <sup>0</sup>
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
	2 <sup>3</sup>	2 <sup>2</sup>	2 <sup>1</sup>	2 <sup>0</sup>	2 <sup>3</sup>	2 <sup>2</sup>	2 <sup>1</sup>	2 <sup>0</sup>	2 <sup>3</sup>	2 <sup>2</sup>	2 <sup>1</sup>	2 <sup>0</sup>	2 <sup>3</sup>	2 <sup>2</sup>	2 <sup>1</sup>	2 <sup>0</sup>
Digit	16 <sup>3</sup>				16 <sup>2</sup>				16 <sup>1</sup>				16 <sup>0</sup>			

The following example shows the bit status of CIO 0000 as “0011110000001110.” This would be represented as “3C0E” in hexadecimal.

Bit	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ON/OFF	0	0	1	1	1	1	0	0	0	0	0	0	1	1	1	0
	$2^3$	$2^2$	$2^1$	$2^0$	$2^3$	$2^2$	$2^1$	$2^0$	$2^3$	$2^2$	$2^1$	$2^0$	$2^3$	$2^2$	$2^1$	$2^0$
	$2^1 + 2^0 = 3$				$2^3 + 2^2 = 12$				0				$2^3 + 2^2 + 2^1 = 14$			
Digit	3				C				0				E			

#### Conversion to Decimal

With unsigned binary data, the digits expressed in hexadecimal can be converted to decimal by multiplying the value of each digit by its respective factor. For example, the hexadecimal value “3C0E” would be converted as follows:

$$(3 \times 16^3) + (12 \times 16^2) + (0 \times 16^1) + (14 \times 16^0) = 15,374$$

#### Range of Expression

The range that can be expressed in hexadecimal 0000 to FFFF (i.e., 0 to 65,535 decimal).

#### Double Data

Two-word data is handled as 32-bit binary data. Values can be expressed as eight-digit hexadecimal (0000 0000 to FFFF FFFF), and the equivalent decimal range is 0 to 4,294,967,295.

### 4-12-2 Signed Binary Data

Data is configured in words, with 16 bits per word. This data is regarded as 16-bit binary data, with the leftmost bit (i.e., the most significant bit, or MSB) used as the sign bit. Signed binary data is often written as four digits hexadecimal.

When the leftmost bit is OFF, (i.e., set to 0), the number is positive and the value is expressed as four-digit hexadecimal, from 0000 to 7FFF.

When the leftmost bit is ON, (i.e., set to 1), the number is negative. The value is expressed as four-digit hexadecimal, from 8000 to FFFF, in 2's complement.

Because the leftmost bit is used as the sign bit, the absolute value that can be expressed is less than that for unsigned binary data.

#### Conversion to Decimal

With signed binary data, the status of the sign bit (i.e., the MSB) determines whether the number will be positive or negative. When the sign bit is OFF, the number will be either positive or zero. As with unsigned binary data, the value can be converted to decimal by multiplying the value of each digit by its respective factor. For example, the hexadecimal value “258C” would be converted as follows:

$$(2 \times 16^3) + (5 \times 16^2) + (8 \times 16^1) + (12 \times 16^0) = +9,612$$

When the sign bit is ON, on the other hand, the number will be negative, and the method for converting to decimal will be different. Because the value is expressed in 2's complement, it must first be converted to a negative number and then the value of each digit can be multiplied by its respective factor. For example, the hexadecimal value "CFC7" would be converted as follows:

2's complement	C	F	C	7
	1100	1111	1100	0111
Subtract 1.				
	C	F	C	6
	1100	1111	1100	0110
Reverse the status of each bit.				
True value (negative)	3	0	3	9
	0011	0000	0011	1001

The negative decimal number is then calculated as follows:  
 $-\left[(3 \times 16^3) + (0 \times 16^2) + (3 \times 16^1) + (9 \times 16^0)\right] = -12,345$

**Note** To convert a negative decimal number into signed binary data, follow the above procedure in reverse. In other words, first convert the absolute value into 2's complement, then reverse the bits and add one.

**Range of Expression**

The hexadecimal range is 0000 to 7FFF for a positive number and 8000 to FFFF for a negative number. These are equivalent in decimal to 0 to +32,767 for a positive number and -32,768 to -1 for a negative number.

**Double Data**

Two-word data is handled as 32 bits of binary data, with the leftmost bit of the leftmost word used as the sign bit. Values can be expressed as eight-digit hexadecimal (0000 0000 to 7FFF FFFF, 8000 0000 to FFFF FFFF), and the equivalent decimal range is 0 to +2,147,483,647 (positive) and -1 to -2,147,483,648 (negative).

**Correlation Between Binary Data and Decimal Numbers**

Unsigned binary data	Decimal number	Signed binary data
FFFF	+65,535	Cannot be expressed.
FFFE	+65,534	
etc.	etc.	
8001	+32,769	7FFF
8000	+32,768	
7FFF	+32,767	
7FFE	+32,766	7FFE
etc.	etc.	etc.
0002	+2	0002
0001	+1	0001
0000	0	0000
Cannot be expressed.	-1	FFFF
	-2	FFFE
	etc.	etc.
	-32,767	8001
	-32,768	8000

Signed Binary Data  
Calculations

The following instructions carry out calculations on signed binary data.

Operation	Mnemonic	Code	Name
Addition	+	400	SIGNED BINARY ADD WITHOUT CARRY
	+L	401	DOUBLE SIGNED BINARY ADD WITHOUT CARRY
	+C	402	SIGNED BINARY ADD WITH CARRY
	+CL	403	DOUBLE SIGNED BINARY ADD WITH CARRY
Subtraction	-	410	SIGNED BINARY SUBTRACT WITHOUT CARRY
	-L	411	DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY
	-C	412	SIGNED BINARY SUBTRACT WITH CARRY
	-CL	413	DOUBLE SIGNED BINARY SUBTRACT WITH CARRY
Multiplication	*	420	SIGNED BINARY MULTIPLY
	*L	421	DOUBLE SIGNED BINARY MULTIPLY
Division	/	430	SIGNED BINARY DIVIDE
	/L	431	DOUBLE SIGNED BINARY DIVIDE
Comparison	=S	302	SIGNED EQUAL
	=SL	303	DOUBLE SIGNED EQUAL
	<>S	307	SIGNED NOT EQUAL
	<>SL	308	DOUBLE SIGNED NOT EQUAL
	<S	312	SIGNED LESS THAN
	<SL	313	DOUBLE SIGNED LESS THAN
	<=S	317	SIGNED LESS THAN OR EQUAL
	<=SL	318	DOUBLE SIGNED LESS THAN OR EQUAL
	>S	322	SIGNED GREATER THAN
	>SL	323	DOUBLE SIGNED GREATER THAN
	>=S	327	SIGNED GREATER THAN OR EQUAL
	>=SL	328	DOUBLE SIGNED GREATER THAN OR EQUAL



### 4-12-3 BCD Data

With BCD data, 16-bit word data is expressed as 4-digit binary data (0000 to 9999) using only the hexadecimal numbers 0 to 9. If the data in any digit corresponds to the hexadecimal numbers A to F, an error will be generated.

Bit	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
	$2^{15}$	$2^{14}$	$2^{13}$	$2^{12}$	$2^{11}$	$2^{10}$	$2^9$	$2^8$	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
	$2^3$	$2^2$	$2^1$	$2^0$	$2^3$	$2^2$	$2^1$	$2^0$	$2^3$	$2^2$	$2^1$	$2^0$	$2^3$	$2^2$	$2^1$	$2^0$
Digit	$10^3$				$10^2$				$10^1$				$10^0$			

In the following example, the bit status of CIO 0000 is shown as "0011100000000111." This value is "3807" in BCD, and would thus be 3,807 in decimal format.

Bit	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ON/OFF	0	0	1	1	1	0	0	0	0	0	0	0	0	1	1	1
	$2^3$	$2^2$	$2^1$	$2^0$	$2^3$	$2^2$	$2^1$	$2^0$	$2^3$	$2^2$	$2^1$	$2^0$	$2^3$	$2^2$	$2^1$	$2^0$
	$2^1 + 2^0 = 3$				$2^3 = 8$				0				$2^2 + 2^1 + 2^0 = 7$			
Digit	3				8				0				7			

#### Range of Expression

The range that can be expressed as BCD data is 0000 to 9999 (0 to 9,999 decimal).

#### Double Data

Two-word data is handled as 8-digit BCD data, with a decimal range of 0 to 99,999,999.

### 4-12-4 Signed BCD Data

Signed BCD data is formatted in special data patterns in order to express negative numbers for 16-bit word data. This format depends on the application, but in the CVM1D CPUs four formats are used.

The BINS(275), BISL(277), BCDS(276), and BDSL(278) instructions are provided for converting between BCD and binary. For details, refer to the explanations of individual instructions in *Section 5 Instruction Set*.

### 4-12-5 Floating-point Data

Floating-point data is stored as 2-word (32-bit) data in a format defined in IEEE754. The CVM1D CPUs provide a number of floating-point operation instructions, including math instructions, logarithms, exponents. All of these handle floating-point data.

The FIX(450), FIXL(451), FLT(452), and FLTL(453) instructions are provided for converting between floating-point and signed binary data. For details, refer to the explanations of individual instructions in *Section 5 Instruction Set*.

# SECTION 5

## Instruction Set

This section explains each instruction in the CVM1D PC instruction set and provides the ladder diagram symbols, data areas, and flags used with each. The instructions provided by the CVM1D PC are described in following subsections by instruction group.

Some instructions, such as Timer and Counter instructions, are used to control execution of other instructions. For example, a timer Completion Flag might be used to turn ON a bit when the time period set for the timer has expired. Although these other instructions are often used to control output bits through the OUTPUT instruction, they can be used to control execution of other instructions as well. The OUTPUT instructions used in examples in this manual can therefore generally be replaced by other instructions to modify the program for specific applications other than controlling output bits directly.

5-1	Notation .....	100
5-2	Instruction Format .....	100
5-3	Data Areas, Definers, and Flags .....	100
5-4	Differentiated and Immediate Refresh Instructions .....	103
5-5	Coding Right-hand Instructions .....	105
5-6	Ladder Diagram Instructions .....	107
5-6-1	LOAD, LOAD NOT, AND, AND NOT, OR, and OR NOT .....	107
5-6-2	CONDITION ON/OFF: UP(018) and DOWN(019) .....	109
5-6-3	BIT TEST: TST(350) and TSTN(351) .....	110
5-6-4	NOT: NOT(010) .....	111
5-6-5	AND LOAD and OR LOAD .....	111
5-7	Bit Control Instructions .....	112
5-7-1	OUTPUT and OUTPUT NOT: OUT and OUT NOT .....	112
5-7-2	DIFFERENTIATE UP/DOWN: DIFU(013) and DIFD(014) .....	113
5-7-3	SET and RESET: SET(016) and RSET(017) .....	115
5-7-4	MULTIPLE BIT SET/RESET: SETA(047)/RSTA(048) .....	116
5-7-5	KEEP: KEEP(011) .....	118
5-8	INTERLOCK and INTERLOCK CLEAR: IL(002) and ILC(003) .....	120
5-9	JUMP and JUMP END: JMP(004) and JME(005) .....	122
5-10	CONDITIONAL JUMP: CJP(221)/CJPN(222) .....	125
5-11	END: END(001) .....	126
5-12	NO OPERATION: NOP(000) .....	126
5-13	Timer and Counter Instructions .....	127
5-13-1	TIMER: TIM .....	129
5-13-2	HIGH-SPEED TIMER: TIMH(015) .....	133
5-13-3	ACCUMULATIVE TIMER: TTIM(120) .....	135
5-13-4	LONG TIMER: TIML(121) .....	137
5-13-5	MULTI-OUTPUT TIMER: MTIM(122) .....	138
5-13-6	COUNTER: CNT .....	140
5-13-7	REVERSIBLE COUNTER: CNTR(012) .....	144
5-13-8	RESET TIMER/COUNTER: CNR(236) .....	145
5-14	Shift Instructions .....	147
5-14-1	SHIFT REGISTER: SFT(050) .....	147
5-14-2	REVERSIBLE SHIFT REGISTER: SFTR(051) .....	150
5-14-3	ASYNCHRONOUS SHIFT REGISTER: ASFT(052) .....	151
5-14-4	WORD SHIFT: WSFT(053) .....	153
5-14-5	SHIFT N-BIT DATA LEFT: NSFL(054) .....	154
5-14-6	SHIFT N-BIT DATA RIGHT: NSFR(055) .....	155
5-14-7	SHIFT N-BITS LEFT: NASL(056) .....	156
5-14-8	SHIFT N-BITS RIGHT: NASR(057) .....	157

5-14-9	DOUBLE SHIFT N-BITS LEFT: NSLL(058)	158
5-14-10	DOUBLE SHIFT N-BITS RIGHT: NSRL(059)	160
5-14-11	ARITHMETIC SHIFT LEFT: ASL(060)	161
5-14-12	ARITHMETIC SHIFT RIGHT: ASR(061)	162
5-14-13	ROTATE LEFT: ROL(062)	163
5-14-14	ROTATE RIGHT: ROR(063)	164
5-14-15	DOUBLE SHIFT LEFT: ASLL(064)	165
5-14-16	DOUBLE SHIFT RIGHT: ASRL(065)	166
5-14-17	DOUBLE ROTATE LEFT: ROLL(066)	167
5-14-18	ROTATE LEFT WITHOUT CARRY: RLNC(260)	168
5-14-19	DOUBLE ROTATE LEFT WITHOUT CARRY: RLNL(262)	169
5-14-20	DOUBLE ROTATE RIGHT: RORL(067)	170
5-14-21	ROTATE RIGHT WITHOUT CARRY: RRNC(261)	171
5-14-22	DOUBLE ROTATE RIGHT W/O CARRY: RRNL(263)	172
5-14-23	ONE DIGIT SHIFT LEFT: SLD(068)	173
5-14-24	ONE DIGIT SHIFT RIGHT: SRD(069)	174
5-15	Data Movement Instructions	175
5-15-1	MOVE: MOV(030)	175
5-15-2	MOVE NOT: MVN(031)	176
5-15-3	DOUBLE MOVE: MOVL(032)	177
5-15-4	DOUBLE MOVE NOT: MVNL(033)	178
5-15-5	DATA EXCHANGE: XCHG(034)	179
5-15-6	DOUBLE DATA EXCHANGE: XCGL(035)	180
5-15-7	MOVE TO REGISTER: MOVR(036)	181
5-15-8	MOVE QUICK: MOVQ(037)	182
5-15-9	MULTIPLE BIT TRANSFER: XFRB(038)	183
5-15-10	BLOCK TRANSFER: XFER(040)	185
5-15-11	BLOCK SET: BSET(041)	186
5-15-12	MOVE BIT: MOVB(042)	187
5-15-13	MOVE DIGIT: MOVD(043)	189
5-15-14	SINGLE WORD DISTRIBUTE: DIST(044)	190
5-15-15	DATA COLLECT: COLL(045)	191
5-15-16	INTERBANK BLOCK TRANSFER: BXFR(046)	192
5-16	Comparison Instructions	193
5-16-1	COMPARE: CMP(020)	193
5-16-2	DOUBLE COMPARE: CMPL(021)	195
5-16-3	BLOCK COMPARE: BCMP(022)	196
5-16-4	TABLE COMPARE: TCMP(023)	198
5-16-5	MULTIPLE COMPARE: MCMP(024)	199
5-16-6	EQUAL: EQU(025)	200
5-16-7	Input Comparison Instructions (300 to 328)	201
5-16-8	SIGNED BINARY COMPARE: CPS(026)	203
5-16-9	DOUBLE SIGNED BINARY COMPARE: CPSL(027)	204
5-16-10	UNSIGNED COMPARE: CMP(028)	205
5-16-11	DOUBLE UNSIGNED COMPARE: CMPL(029)	205
5-17	Conversion Instructions	207
5-17-1	BCD-TO-BINARY: BIN(100)	207
5-17-2	BINARY-TO-BCD: BCD(101)	208
5-17-3	DOUBLE BCD-TO-DOUBLE BINARY: BINL(102)	209
5-17-4	DOUBLE BINARY-TO-DOUBLE BCD: BCDL(103)	210
5-17-5	2'S COMPLEMENT: NEG(104)	211
5-17-6	DOUBLE 2'S COMPLEMENT: NEGL(105)	212
5-17-7	SIGN: SIGN(106)	213
5-17-8	DATA DECODER: MLPX(110)	214

5-17-9	DATA ENCODER: DMPX(111)	216
5-17-10	7-SEGMENT DECODER: SDEC(112)	219
5-17-11	ASCII CONVERT: ASC(113)	222
5-17-12	BIT COUNTER: BCNT(114)	224
5-17-13	COLUMN TO LINE: LINE(115)	225
5-17-14	LINE TO COLUMN: COLM(116)	226
5-17-15	ASCII TO HEX: HEX(117)	227
5-17-16	SIGNED BCD-TO-BINARY: BINS(275)	230
5-17-17	SIGNED BINARY-TO-BCD: BCDS(276)	232
5-17-18	DOUBLE SIGNED BCD-TO-BINARY: BISL(277)	234
5-17-19	DOUBLE SIGNED BINARY-TO-BCD: BDSL(278)	236
5-18	BCD Calculation Instructions	237
5-18-1	SET CARRY: STC(078)	238
5-18-2	CLEAR CARRY: CLC(079)	238
5-18-3	BCD ADD: ADD(070)	238
5-18-4	BCD SUBTRACT: SUB(071)	239
5-18-5	BCD MULTIPLY: MUL(072)	241
5-18-6	BCD DIVIDE: DIV(073)	242
5-18-7	DOUBLE BCD ADD: ADDL(074)	243
5-18-8	DOUBLE BCD SUBTRACT: SUBL(075)	244
5-18-9	DOUBLE BCD MULTIPLY: MULL(076)	245
5-18-10	DOUBLE BCD DIVIDE: DIVL(077)	246
5-19	Binary Calculation Instructions	249
5-19-1	BINARY ADD: ADB(080)	249
5-19-2	BINARY SUBTRACT: SBB(081)	250
5-19-3	BINARY MULTIPLY: MLB(082)	252
5-19-4	BINARY DIVIDE: DVB(083)	253
5-19-5	DOUBLE BINARY ADD: ADBL(084)	254
5-19-6	DOUBLE BINARY SUBTRACT: SBBL(085)	255
5-19-7	DOUBLE BINARY MULTIPLY: MLBL(086)	256
5-19-8	DOUBLE BINARY DIVIDE: DVBL(087)	257
5-20	Symbol Math Instructions	259
5-20-1	Binary Addition: +(400)/+L(401)/+C(402)/+CL(403)	259
5-20-2	BCD Addition: +B(404)/+BL(405)/+BC(406)/+BCL(407)	261
5-20-3	Binary Subtraction: -(410)/-L(411)/-C(412)/-CL(413)	263
5-20-4	BCD Subtraction: -B(414)/-BL(415)/-BC(416)/-BCL(417)	268
5-20-5	Binary Multiplication: *(420)/*L(421)/*U(422)/*UL(423)	272
5-20-6	BCD Multiplication: *B(424)/*BL(425)	274
5-20-7	Binary Division: /(430)//L(431)//U(432)//UL(433)	276
5-20-8	BCD Division: /B(434)//BL(435)	278
5-21	Floating-point Math Instructions	280
5-21-1	FLOATING TO 16-BIT: FIX(450)	283
5-21-2	FLOATING TO 32-BIT: FIXL(451)	284
5-21-3	16-BIT TO FLOATING: FLT(452)	285
5-21-4	32-BIT TO FLOATING: FLTL(453)	285
5-21-5	FLOATING-POINT ADD: +F(454)	286
5-21-6	FLOATING-POINT SUBTRACT: -F(455)	287
5-21-7	FLOATING-POINT MULTIPLY: *F(456)	288
5-21-8	FLOATING-POINT DIVIDE: /F(457)	289
5-21-9	DEGREES TO RADIANS: RAD(458)	290
5-21-10	RADIANS TO DEGREES: DEG(459)	291
5-21-11	SINE: SIN(460)	292
5-21-12	COSINE: COS(461)	293
5-21-13	TANGENT: TAN(462)	294

5-21-14	SINE TO ANGLE: ASIN(463)	295
5-21-15	COSINE TO ANGLE: ACOS(464)	296
5-21-16	TANGENT TO ANGLE: ATAN(465)	297
5-21-17	SQUARE ROOT: SQRT(466)	298
5-21-18	EXPONENT: EXP(467)	299
5-21-19	LOGARITHM: LOG(468)	300
5-22	Increment/Decrement Instructions	301
5-22-1	INCREMENT BCD: INC(090)	301
5-22-2	DECREMENT BCD: DEC(091)	301
5-22-3	INCREMENT BINARY: INCB(092)	302
5-22-4	DECREMENT BINARY: DECB(093)	303
5-22-5	DOUBLE INCREMENT BCD: INCL(094)	303
5-22-6	DOUBLE DECREMENT BCD: DECL(095)	304
5-22-7	DOUBLE INCREMENT BINARY: INBL(096)	304
5-22-8	DOUBLE DECREMENT BINARY: DCBL(097)	305
5-23	Special Math Instructions	306
5-23-1	FIND MAXIMUM: MAX(165)	306
5-23-2	FIND MINIMUM: MIN(166)	307
5-23-3	SUM: SUM(167)	309
5-23-4	BCD SQUARE ROOT: ROOT(140)	310
5-23-5	BINARY ROOT: ROTB(274)	312
5-23-6	FLOATING POINT DIVIDE: FDIV(141)	313
5-23-7	ARITHMETIC PROCESS: APR(142)	315
5-24	PID and Related Instructions	317
5-24-1	PID CONTROL: PID(270)	317
5-24-2	LIMIT CONTROL: LMT(271)	326
5-24-3	DEAD-BAND CONTROL: BAND(272)	327
5-24-4	DEAD-ZONE CONTROL: ZONE(273)	328
5-25	Logic Instructions	330
5-25-1	LOGICAL AND: ANDW(130)	330
5-25-2	LOGICAL OR: ORW(131)	331
5-25-3	EXCLUSIVE OR: XORW(132)	331
5-25-4	EXCLUSIVE NOR: XNRW(133)	332
5-25-5	DOUBLE LOGICAL AND: ANDL(134)	333
5-25-6	DOUBLE LOGICAL OR: ORWL(135)	334
5-25-7	DOUBLE EXCLUSIVE OR: XORL(136)	334
5-25-8	DOUBLE EXCLUSIVE NOR: XNRL(137)	335
5-25-9	COMPLEMENT: COM(138)	336
5-25-10	DOUBLE COMPLEMENT: COML(139)	337
5-26	Time Instructions	337
5-26-1	HOURS TO SECONDS: SEC(143)	337
5-26-2	SECONDS TO HOURS: HMS(144)	338
5-26-3	CALENDAR ADD: CADD(145)	339
5-26-4	CALENDAR SUBTRACT: CSUB(146)	340
5-26-5	CLOCK COMPENSATION: DATE(179)	341
5-27	Special Instructions	342
5-27-1	FAILURE/SEVERE FAILURE ALARM: FAL(006) and FALS(007)	342
5-27-2	FAILURE POINT DETECTION: FPD(177)	344
5-27-3	MAXIMUM CYCLE TIME EXTEND: WDT(178)	349
5-27-4	I/O REFRESH: IORF(184)	350
5-27-5	I/O DISPLAY: IODP(189)	350
5-27-6	SELECT EM BANK: EMBC(171)	352
5-27-7	DATA SEARCH: SRCH(164)	353
5-28	Flag/Register Instructions	354

5-28-1	LOAD FLAGS: CCL(172)	354
5-28-2	SAVE FLAGS: CCS(173)	355
5-28-3	LOAD REGISTER: REGL(175)	355
5-28-4	SAVE REGISTER: REGS(176)	356
5-29	STEP DEFINE and STEP START: STEP(008)/SNXT(009)	356
5-30	Subroutines	365
5-30-1	SUBROUTINE ENTRY and RETURN: SBN(150)/RET(152)	365
5-30-2	SUBROUTINE CALL: SBS(151)	366
5-30-3	MACRO: MCRO(156)	368
5-31	Stack Instructions	370
5-31-1	SET STACK: SSET(160)	370
5-31-2	PUSH ONTO STACK: PUSH(161)	371
5-31-3	LAST IN FIRST OUT: LIFO(162)	372
5-31-4	FIRST IN FIRST OUT: FIFO(163)	373
5-32	Data Tracing	374
5-32-1	TRACE MEMORY SAMPLING: TRSM(170)	374
5-32-2	MARK TRACE: MARK(174)	376
5-33	Special I/O Instructions	377
5-33-1	I/O READ: READ(190)	377
5-33-2	I/O READ 2: RD2(280)	379
5-33-3	I/O WRITE: WRIT(191)	381
5-33-4	I/O WRITE 2: WR2(281)	383
5-34	Network Instructions	385
5-34-1	DISABLE ACCESS: IOSP(187)	385
5-34-2	ENABLE ACCESS: IORS(188)	386
5-34-3	DISPLAY MESSAGE: MSG(195)	386
5-34-4	NETWORK SEND: SEND(192)	388
5-34-5	NETWORK RECEIVE: RECV(193)	390
5-34-6	DELIVER COMMAND: CMND(194)	392
5-34-7	About Network Instruction Operations	395
5-35	Block Programming Instructions	400
5-35-1	Overview	400
5-35-2	BLOCK PROGRAM BEGIN/END: BPRG(250) / BEND<001>	401
5-35-3	Branching—IF<002>, ELSE<003>, and IEND<004>	402
5-35-4	ONE CYCLE AND WAIT: WAIT<005>	405
5-35-5	CONDITIONAL BLOCK EXIT: EXIT<006>	406
5-35-6	Loop Control—LOOP<009>/LEND<010>	407
5-35-7	BLOCK PROGRAM PAUSE/RESTART : BPPS<011>/BPRS<012>	408
5-35-8	HIGH-SPEED TIMER/TIMER WAIT: TIMW<013>/TMHW<015>	409
5-35-9	COUNTER WAIT: CNTW<014>	410

## 5-1 Notation

In the remainder of this manual, instructions will be referred to by their mnemonics. For example, the OUTPUT instruction will be called OUT; the AND LOAD instruction, AND LD. If you're not sure of the instruction a mnemonic is for, refer to *Appendix B Programming Instructions*.

If an instruction is assigned a function code, it will be given in parentheses after the mnemonic. These function codes, which are 3-digit decimal numbers, can be used to input instructions into the CPU and are described briefly below. A table of instructions listed in order of function codes is also provided in *Appendix B*.

An up or down arrow, ↑ or ↓ at the beginning of a mnemonic indicates a differentiated up or down version of that instruction. An exclamation mark, !, before a mnemonic indicates an immediate refresh variation of that instruction. The immediate refresh variations can be used in simplex mode only. Differentiated and immediate refresh instructions are explained on page 103.

## 5-2 Instruction Format

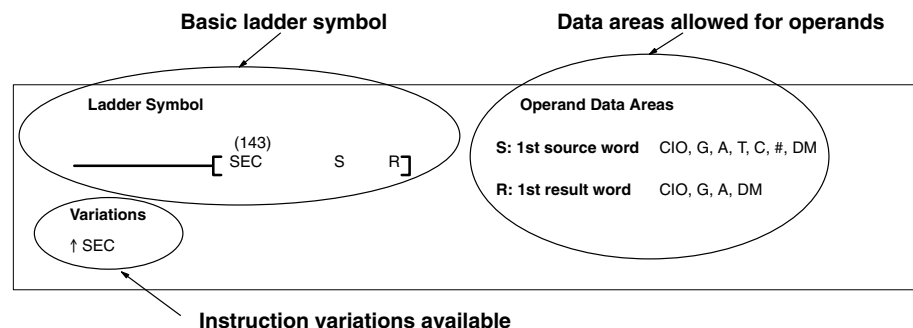
Most instructions have at least one or more operands associated with them. Operands indicate or provide the data on which an instruction is to be performed. These are sometimes input as the actual numeric values (i.e., as constants), but are usually the addresses of data area words or bits that contain the data to be used. A bit whose address is designated as an operand is called an **operand bit**; a word whose address is designated as an operand is called an **operand word**. In some instructions, the word address designated in an instruction indicates the first of multiple words containing the desired data.

Each instruction requires one or more words in Program Memory. The first word is the **instruction word**, which specifies the instruction and contains any definers (described below) or operand bits required by the instruction. Other operands required by the instruction are contained in following words, one operand per word. Some instructions require up to four words.

A **definer** is an operand associated with an instruction and contained in the same word as the instruction itself. These operands define the instruction rather than telling what data it is to use. Examples of definers are timer and counter numbers, which are used in timer and counter instructions to create timers and counters, as well as jump numbers, which define which JUMP instruction is paired with which JUMP END instruction. Bit operands are also contained in the same word as the instruction itself, although these are not considered definers.

## 5-3 Data Areas, Definers, and Flags

Each instruction is introduced with a frame that shows the basic form of the instruction, the variations of the instruction, and the data areas that can be used for each operand, as shown in the following illustration:



**Basic Ladder Symbol**

The ladder symbol shows how the instruction will appear in a program. The function code (here, 143) is provided above the mnemonic (SEC) and the operands are provided to the right (here, S and R). The ladder symbol is the same for any of the variations of the instruction except that the mnemonic changes.

**Variations**

The alternate forms of the instruction are listed here, including immediate refresh and differentiated forms.

**Operand Data Area Precautions**

The data areas are listed that can be used for each instruction. The actual operand will be a number, such as a word address, a bit address, an indirect address, or a constant, depending on the requirements of the instruction and the needs of the program.

Not all addresses in the specified data areas are necessarily allowed for an operand, e.g., if an operand requires two words, the last word in a data area cannot be designated as the first word of the operand because all words for a single operand must be within the same data area. Refer to *Section 3 Memory Areas* for addressing conventions and the addresses of specific flags and control bits.

For example, the second operand (CB) in the BLOCK COMPARE instruction (BCMP(022), shown below) specifies the first word of a comparison table that is 32 words long. This operand thus cannot be any of the last 31 words in an data area, e.g., if the CPU Bus Link Area is used, the last word that could be designated would be G224. Designating G245 would cause an error and the instruction would not be executed.

Ladder Symbol	Operand Data Areas
<pre> (022) ——— [ BCMP  S   CB   R ]           </pre>	<p><b>S: Source data</b>    CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>CB: 1<sup>st</sup> block word</b>    CIO, G, A, T, C, DM</p> <p><b>R: Result word</b>    CIO, G, A, T, C, DM, DR, IR</p>
<p><b>Variations</b></p> <p>↑ BCMP(022)</p>	

**Note:** The DM Area, IR, and DR are not listed as operand data areas unless they can be addressed directly. These areas can be used for indirectly addressing operands provided that the address being pointed to is a legal address. For example, for BCMP(022) (shown above) and Index Register could be used to indirectly address a DM address for the second operand, CB. Refer to the discussion on *Indirect Addressing* later in this section.

**Caution**

The Auxiliary Area words between A000 and A255 and the CPU Bus Link Area words G008 through G255 can be read from or written to from the user program. A256 to A511 and G000 to G007, however, can be read from to access the data provided there, but **cannot** be written to from the user program, i.e., they **cannot** be used as operands if the instruction alters the contents of the operand during processing.

**Designating Constants**

Although data area addresses are most often given as operands, many operands can be input as constants. The available value range for a given operand depends on the particular instruction that uses it. Constants must also be entered in the form required by the instruction, i.e., in BCD or in hexadecimal.

Constants are also input as either four digits or as either digits, depending on the requirements of the instruction (e.g., constants for double, or long, instructions require eight digits).



**Flags**

The *Flags* subsection lists flags that are affected by execution of an instruction. These flags include the following Auxiliary Area flags.

Abbreviation	Name	Bit
ER	Instruction Execution Error Flag	A50003
CY	Carry Flag	A50004
GR	Greater Than Flag	A50005
EQ	Equals Flag	A50006
LE	Less Than Flag	A50007
N	Negative Flag	A50008

ER is the flag most commonly used for monitoring an instruction’s execution. When ER goes ON, it indicates that an error has occurred in attempting to execute an instruction. The *Flags* subsection of each instruction lists possible reasons for ER going ON. ER will turn ON if operands are not entered correctly.



**Caution**

Most instructions are not executed when ER is ON. A table of instructions and the flags they affect is provided in *Appendix B Error and Arithmetic Flag Operation*.

**Indirect Addressing**

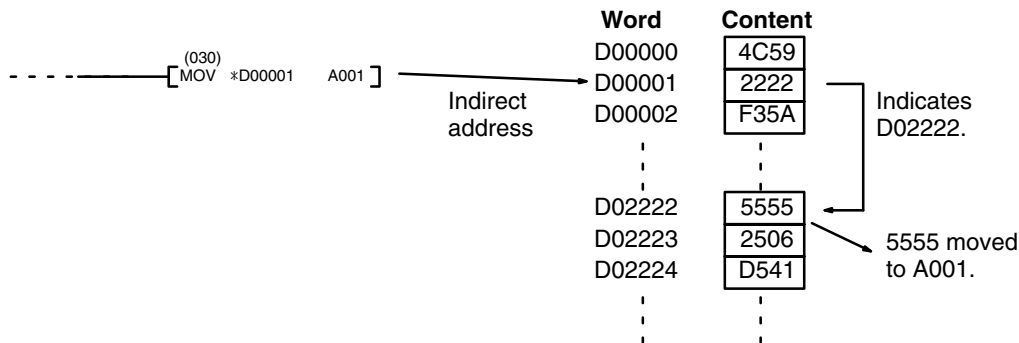
The DM or EM Area can be used to indirectly address an operand. Indirect DM or EM addressing is specified by placing an asterisk before the D or E: \*D or \*E. (The EM Area is contained in the CV1000-DM□□1 EM Unit, sold separately.)

The operation of indirect addressing is affected by the PC Setup specified from the SSS. The PC Setup can be used to specify whether the content of a word containing an indirect address contains the BCD data area address or contains the binary (hexadecimal) PC memory address.

**BCD Addressing**

When indirect DM data is designated as BCD, the address of the desired word must be in BCD and it must specify the data area address of a word within the DM or EM Area. The content of the operand word containing the indirect address (e.g., \*D00000) has to be in BCD and has to be between 0000 and 9999. Although DM and EM Area addresses go to D24575 and E32765, only the first 10,000 words can be indirectly addressed when indirect DM data is designated as BCD.

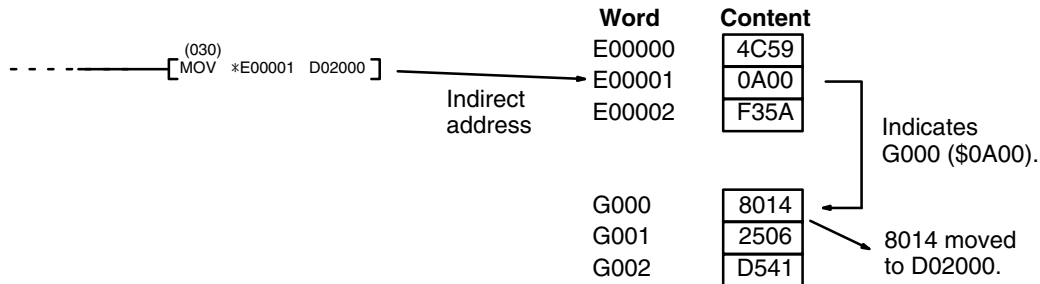
When an indirect DM or EM address is specified in BCD, the DM or EM word specified for the operand will contain the address of the DM or EM word that contains the data that will be used as the operand of the instruction. If, for example, \*D00001 was designated as the first operand of MOV(030), the contents of D00001 was 2222, and D02222 contained 5555, the value 5555 would be moved to the word specified for the second operand.



**Binary Indirect Addressing**

When indirect DM data is designated as binary, the content of the \*D or \*E address specifies the PC memory address, and thus can have any value between \$0000 and \$FFFF, as long as the instruction can be executed with the specified PC memory address.

When an indirect DM or EM address is specified in binary (hexadecimal), the designated DM or EM word will contain the PC memory address of the word that contains the data that will be used as the operand of the instruction. If, for example, \*E00001 was designated as the first operand of MOV(030), the contents of E00001 was \$0A00, and \$0A00 (G000, CPU Bus Link Area) contained 8014, the value 8014 would be moved to the word specified for the second operand.



**Index and Data Registers**

Index and data registers can also be used to indirectly address memory. Refer to 3-10 *Index and Data Registers (IR and DR)* for details and examples.

## 5-4 Differentiated and Immediate Refresh Instructions

**Differentiated Instructions**

Most instructions are provided in both non-differentiated and differentiate up forms, and some instructions are also provided with a differentiate down form. Differentiated instructions are distinguished by an up or down arrow, ↑ or ↓, just before the instruction mnemonic.

A non-differentiated instruction is executed each time it is scanned. A differentiate up instruction is executed only once after its execution condition goes from OFF to ON. If the execution condition has not changed or has changed from ON to OFF since the last time the instruction was scanned, the instruction will not be executed.

A differentiate down instruction is executed only once after its execution condition goes from ON to OFF. If the execution condition has not changed or has changed from OFF to ON since the last time the instruction was scanned, the instruction will not be executed.

**Note:** Do not use A50013 (Always ON Flag), A50014 (Always OFF Flag), or A50015 (First Cycle Flag) to control execution of differentiated instructions. The instructions will never be executed.

The following examples show how this works with MOV(030) and ↑MOV(030) which are used to move the data in the address designated by the first operand to the address designated by the second operand.

The execution condition is always compared to the execution condition that existed the last time the instruction was scanned, which may not be the previous cycle if an instruction is in a section of the program skipped by a jump, in a subroutine, etc. In the following examples, we will assume that the MOVE instruction is scanned each cycle.

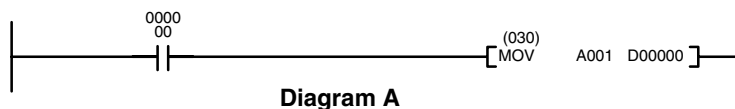


Diagram A

Address	Instruction	Operands
00000	LD	000000
00001	MOV(030)	
		A0001
		D00000

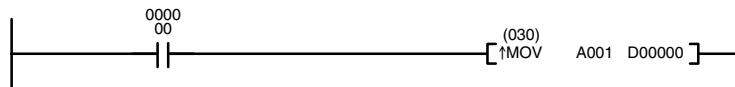


Diagram B

Address	Instruction	Operands
00000	LD	000000
00001	↑MOV(030)	
		A001
		D00000

In diagram A, the non-differentiated MOV(030) will move the content of A001 to D00000 whenever it is scanned with 000000 ON. If the cycle time is 80 ms and 000000 remains ON for 2.0 seconds, this move operation will be performed 25 times and D00000 will contain the last value moved to it.

In diagram B, the differentiate up instruction ↑MOV(030) will move the content of A001 to D00000 only once after 000000 goes ON. Even if 000000 remains ON for 2.0 seconds, the move operation will be executed only during the first cycle in which 000000 has changed from OFF to ON. Because the content of A001 could very well change during the 2 seconds while 000000 is ON, the final content of D00000 after the 2 seconds could be different depending on whether MOV(030) or ↑MOV(030) was used.

All operands and other specifications for instructions are the same regardless of whether the differentiated or non-differentiated form of an instruction is used. When inputting, the same function codes are also used.

Operation of differentiated instructions can be uncertain when the instructions are programmed between IL and ILC, between JMP and JME, or in subroutines. Refer to 5-8 INTERLOCK and INTERLOCK CLEAR – IL(002) and ILC(003), 5-9 JUMP and JUMP END – JMP(004) and JME(005), and 5-30 Subroutines for details.

CVM1D PCs also provide differentiation instructions: DIFU(013) and DIFD(014). These instruction operate as the differentiated variations of the OUTPUT instruction: DIFU(013) turns ON a bit for one cycle when the execution condition has changed from OFF to ON and DIFD(014) turns ON a bit for one cycle when the execution condition has changed from ON to OFF. Refer to 5-7-2 DIFFERENTIATE UP/DOWN – DIFU(013) and DIFD(014) for details.

Up or down differentiation can be combined with immediate refreshing in a single instruction.

**Immediate Refreshing**

Many instructions are provided in an immediate refresh variation, distinguished by an exclamation mark, !, at the beginning of the mnemonic. An immediate refresh instruction updates the status of input bits just before, or output bits just after, the instruction is executed. If the instruction has a word operand, the whole word is updated, and if the instruction has a bit operand, only the byte (leftmost or rightmost 8 bits) containing the bit operand is updated.

**Note:** Immediate-refresh variations can be used in simplex mode only; they cannot be used in duplex mode.

The I/O response time is reduced with an immediate refresh instruction because status is read from the input bit or written to the output bit without waiting for the next I/O refresh period. Refer to *6-5 I/O Response Time* for details on the effects of immediate refresh instructions on I/O response time.

Immediate refreshing and up or down differentiation can be combined in a single instruction. Immediate refresh instructions cannot be used for I/O points on Units mounted to Slave Racks in a SYSMAC BUS or SYSMAC BUS/2 Remote I/O System.

## 5-5 Coding Right-hand Instructions

Writing mnemonic code for ladder instructions is described in *Section 4 Writing Programs*. Converting the information in the ladder diagram symbol for all other instructions follows the same pattern, as described below, and is not specified for each instruction individually.

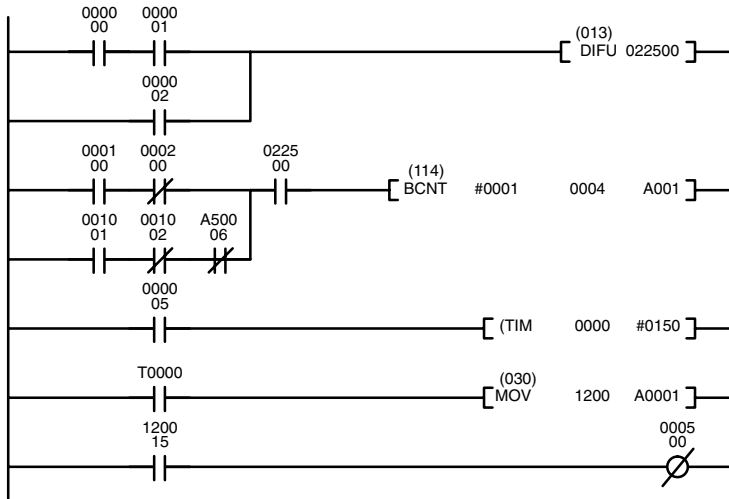
The first word of any instruction defines the instruction and provides any definers. The bit operand is also placed on the same line as the mnemonic for some instructions with certain operands. All other operands are placed on lines after the instruction line, one operand per line and in the same order as they appear in the ladder symbol for the instruction.

The address and instruction columns of the mnemonic code table are filled in for the instruction word only. For all other lines, the left two columns are left blank. If the instruction requires no definer or bit operand, the data column is left blank for first line. It is a good idea to cross through any blank data column spaces (for all instruction words that do not require data) so that the data column can be quickly scanned to see if any addresses have been left out.

If a CIO address is used in the data column, the left side of the column is left blank. If any other data area is used, the data area abbreviation is placed on the left side and the address is placed on the right side. If a constant is to be input, the number symbol (#) is placed on the left side of the data column and the number to be input is placed on the right side. Any numbers input as definers in the instruction word do not require the number symbol on the right side.

When coding an instruction that has a function code, be sure to write in the function code, which can be used when inputting the instruction via the Programming Device. Also be sure to designate differentiated instructions with the ↑ or ↓ symbol.

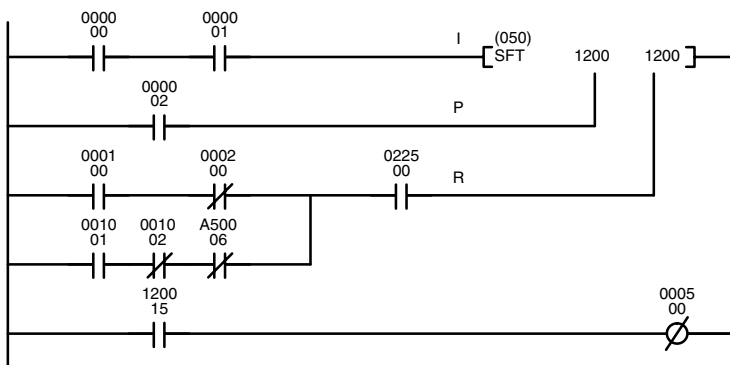
The following diagram and corresponding mnemonic code illustrates the points described above.



Address	Instruction	Operands
00000	LD	000000
00001	AND	000001
00002	OR	000002
00003	DIFU(013)	022500
00004	LD	000100
00005	AND NOT	000200
00006	LD	001001
00007	AND NOT	001002
00008	AND NOT	A50006
00009	OR LD	--
00010	AND	022500
00011	BCNT(114)	--
		#0001
		0004
		A001
00012	LD	000005
00013		T0000
		#0150
00014	LD	T0000
00015	MOV(030)	--
		1200
		A001
00016	LD	120015
00017	OUT NOT	000500

**Multiple Instruction Lines**

If a right-hand instruction requires multiple instruction lines, all of the lines for the instruction are entered before the right-hand instruction when inputting in mnemonic form (although this is not always true when inputting using ladder diagrams). Each line of the instruction is coded first to form 'logic blocks' combined by the right-hand instruction. An example of this for SFT(050) is shown below.



Address	Instruction	Operands
00000	LD	000000
00001	AND	000001
00002	LD	000002
00003	LD	000100
00004	AND NOT	000200
00005	LD	001001
00006	AND NOT	001002
00007	AND NOT	A50006
00008	OR LD	--
00009	AND	022500
00010	SFT(050)	--
		1200
		1200
00011	LD	120015
00012	OUT NOT	000500

**END(001)**

When you have finished coding the program, make sure you have placed END(001) at the last address.

## 5-6 Ladder Diagram Instructions

**Ladder Diagram Instructions** include Ladder Instructions and Logic Block Instructions. **Ladder Instructions** correspond to the conditions on the ladder diagram. **Logic Block Instructions** are used to relate more complex parts of the diagram that cannot be programmed with Ladder Instructions alone.

### 5-6-1 LOAD, LOAD NOT, AND, AND NOT, OR, and OR NOT

#### LOAD: LD

Ladder Symbols	Operand Data Area
	<b>B: Bit</b> CIO, G, A, T, C
	<b>Mnemonics</b> LD                      ! ↑ LD                      ! ↓ LD ↑ LD                      ↓ LD                      ! LD

#### LOAD NOT: LD NOT

Ladder Symbols	Operand Data Area
	<b>B: Bit</b> CIO, G, A, T, C
	<b>Mnemonics</b> LD NOT                      ! LD NOT

#### AND: AND

Ladder Symbols	Operand Data Area
	<b>B: Bit</b> CIO, G, A, T, C
	<b>Mnemonics</b> AND                      ! ↑ AND                      ! ↓ AND ↑ AND                      ↓ AND                      ! AND

#### AND NOT: AND NOT

Ladder Symbols	Operand Data Area
	<b>B: Bit</b> CIO, G, A, T, C
	<b>Mnemonics</b> AND NOT                      ! AND NOT

**OR: OR**

Ladder Symbols	Operand Data Area
	<b>B: Bit</b> CIO, G, A, T, C
	<b>Mnemonics</b> OR                      ! ↑ OR                      ! ↓ OR ↑ OR                      ↓ OR                      ! OR

**OR NOT: OR NOT**

Ladder Symbols	Operand Data Area
	<b>B: Bit</b> CIO, G, A, T, C
	<b>Mnemonics</b> OR NOT                      ! OR NOT

**Description**

These six basic instructions correspond to the conditions on a ladder diagram. As described in *Section 4 Writing Programs*, the status of the bits assigned to each instruction determines the execution conditions for all other instructions. Each of these instructions and each bit address can be used as many times as required. Each bit can be used in as many of these instructions as required.

The status of the bit operand (B) assigned to LD or LD NOT determines the first execution condition. AND takes the logical AND between the execution condition and the status of its bit operand; AND NOT, the logical AND between the execution condition and the inverse of the status of its bit operand. OR takes the logical OR between the execution condition and the status of its bit operand; OR NOT, the logical OR between the execution condition and the inverse of the status of its bit operand.

These six instructions use only one word of program memory, not two, when the operand is in the CIO Area between CIO 000000 and CIO 051115, saving program memory and reducing the instruction execution time. Two words of program memory are required for all other operands.

TR bits are added to the program automatically when creating the program with the ladder diagram using the SSS. Input TR bits only when inputting the program with mnemonics. The ladder symbol for loading TR bits is different from that shown above for LD and LD NOT. Refer to *4-3-3 Ladder Instructions* for details.

**Note:** Immediate-refresh variations can be used in simplex mode only; they cannot be used in duplex mode.

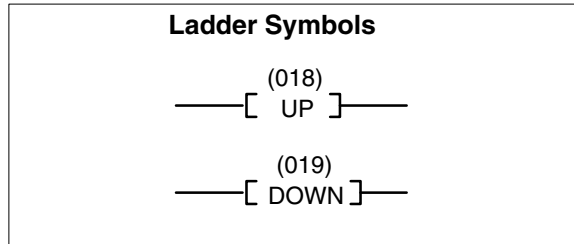
**Precautions**

There is no limit to the number of any of these instructions, or restrictions in the order in which they must be used, as long as the program memory capacity of the PC is not exceeded.

**Flags**

There are no flags affected by these instructions.

### 5-6-2 CONDITION ON/OFF: UP(018) and DOWN(019)



**Description**

UP(018) turns ON the execution condition for one cycle at the rising edge (OFF to ON) of the execution condition and then turns OFF the execution condition until the next time a rising edge is detected.

DOWN(019) turns ON the execution condition for one cycle at the falling edge (ON to OFF) of the execution condition and then turns OFF the execution condition until the next time a falling edge is detected.

Another instruction must follow UP(018) or DOWN(019), i.e., they cannot be used as right-hand instructions.

**Precautions**

Be careful when using UP(018) and DOWN(019) in subroutines between IL and ILC, and between JMP and JME instructions, because the execution condition may remain ON for more than one scan. Refer to 5-8 INTERLOCK and INTERLOCK CLEAR: IL(002) and ILC(003), 5-9 JUMP and JUMP END: JMP(004) and JME(005), and 5-30 Subroutines for details.

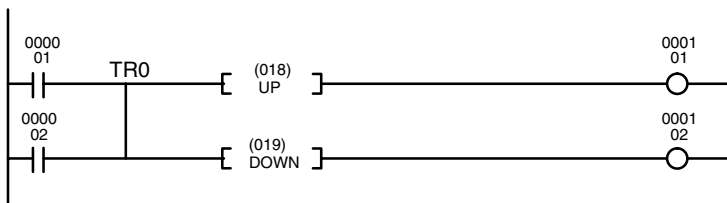
The DIFU(013) and DIFD(014) instructions can also be used for the same purpose, but they require work bits. UP(018) and DOWN(019) simplify programming by reducing the number of work bits and program addresses needed.

**Flags**

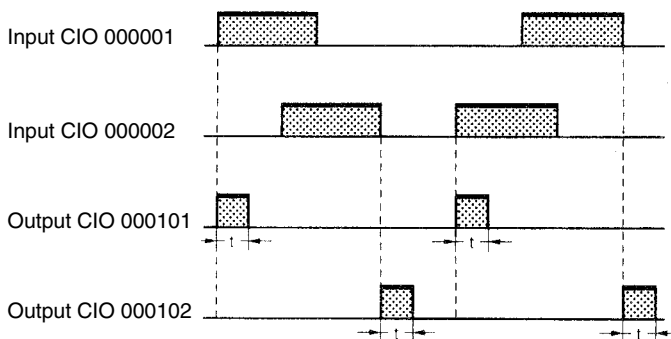
There are no flags affected by UP(018) or DOWN(019).

**Example**

The timing chart illustrates the operation of UP(018) and DOWN(019) in the following example.



Address	Instruction	Operands
00001	LD	000001
00002	OR	000002
00003	OUT	TR0
00004	UP(018)	
00005	OUT	00101
00006	LD	TR0
00007	DOWN(019)	
00008	OUT	000102



t: Cycle time



### 5-6-3 BIT TEST: TST(350) and TSTN(351)

Ladder Symbol	Operand Data Areas
	<b>S: Source word</b> CIO, G, A, DM, DR, IR <b>N: Bit number</b> CIO, G, A, T, C, #, DM, DR, IR

**Description**

TST(350) turns ON the execution condition when the specified bit in the specified word is ON and turns OFF the execution condition when the bit is OFF.

TSTN(351) turns OFF the execution condition when the specified bit in the specified word is ON and turns ON the execution condition when the bit is OFF.

The bit position is designated in N between 0000 and 0015 in BCD.

**Precautions**

TST(350) and TSTN(351) cannot be used as right-hand instructions, i.e., another instruction must appear between them and the right bus bar.

N must be BCD between 0000 and 0015.

**Note:** Refer to page 101 for general precautions on operand data areas.

**Flags**

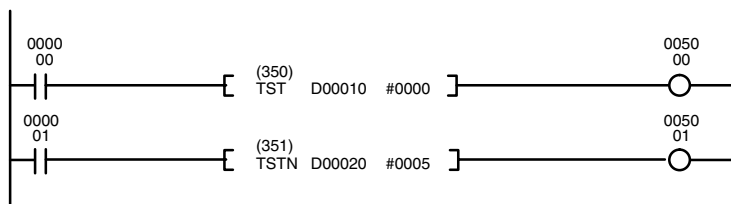
ER (A50003): N is not 0000 to 0015 BCD.

Content of \*DM word is not BCD.

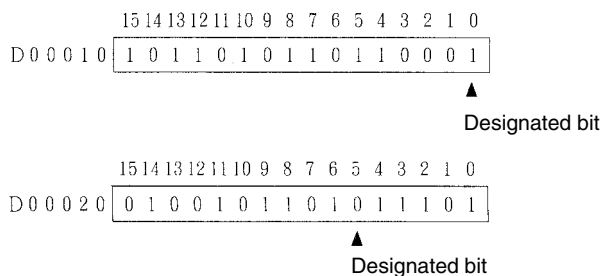
**Example**

In the first instruction line below, when CIO 000000 turns ON, TST(350) checks whether the designated bit (bit 00 in D00010) is ON or OFF. In this case, because it is ON, CIO 005000 is turned ON.

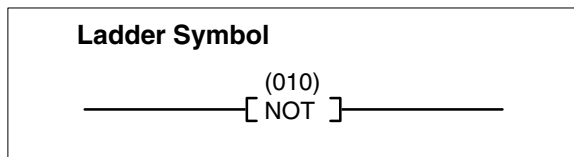
In the second instruction line below, when CIO 000001 turns ON, TST(350) checks whether the designated bit (bit 05 in D00020) is ON or OFF. In this case, because it is OFF, CIO 005001 is turned ON.



Address	Instruction	Operands
00000	LD	000000
00001	TST(350)	
		D00010
		#0000
00002	OUT	005000
00003	LD	000001
00004	TSTN(351)	
		D00020
		#0005
00005	OUT	005001



### 5-6-4 NOT: NOT(010)



**Description**

NOT(010) reverses the execution condition. NOT(010) is an intermediate instruction that inverts the execution condition that precedes it. As an intermediate instruction, it cannot be placed at the end of an instruction line, only between conditions or between a condition and a right-hand instruction.

**Precautions**

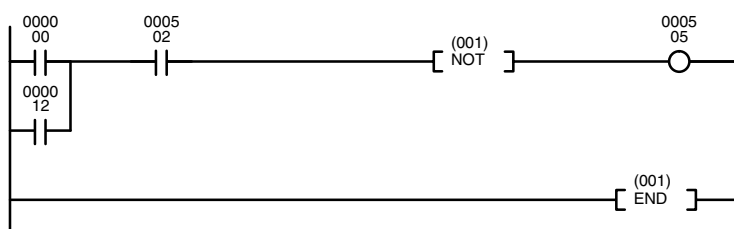
NOT(010) cannot be used as right-hand instructions, i.e., another instruction must appear between them and the right bus bar.

**Flags**

There are no flags affected by NOT(010).

**Example**

The following example and bit status table show the operation of NOT(010).

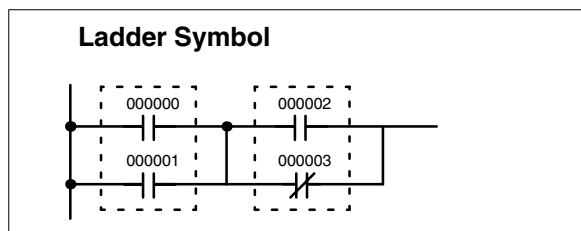


Address	Instruction	Operands
00000	LD	000000
00001	OR	000012
00002	AND	000502
00003	NOT(010)	---
00004	OUT	000505
00005	END(001)	---

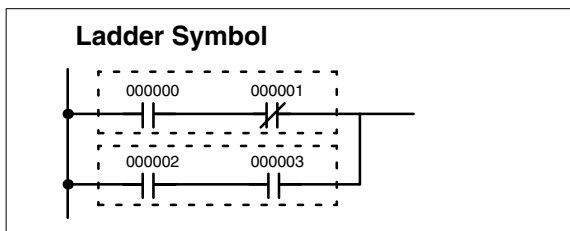
Bit	Bit status							
	ON	OFF	ON	OFF	ON	OFF	ON	OFF
000000	ON	OFF	ON	OFF	ON	OFF	ON	OFF
000012	ON	ON	OFF	OFF	ON	ON	OFF	OFF
000502	ON	ON	ON	ON	OFF	OFF	OFF	OFF
000505	OFF	OFF	OFF	ON	ON	ON	ON	ON

### 5-6-5 AND LOAD and OR LOAD

**AND LOAD: AND LD**



**OR LOAD: OR LD**



**Description**

When instructions are combined into blocks that cannot be logically combined using only OR and AND operations, AND LD and OR LD are used. Whereas AND and OR operations logically combine a bit status and an execution condition, AND LD and OR LD logically combine two execution conditions, the current one and the last unused one.

AND LD and OR LD are not necessary when drawing ladder diagrams or when inputting ladder diagrams using ladder diagram programming. They are required, however, to convert the program to and input it in mnemonic form.

In order to reduce the number of programming instructions required, a basic understanding of logic block instructions is required. For an introduction to logic blocks, refer to 4-4-1 Logic Block Instructions.

**Flags**



There are no flags affected by these instructions.

## 5-7 Bit Control Instructions



The instructions in this section are used to control bit status. These instructions are used to turn bits ON and OFF in different ways.

### 5-7-1 OUTPUT and OUTPUT NOT: OUT and OUT NOT

#### OUTPUT: OUT

Ladder Symbols	Operand Data Area
 	<b>B: Bit</b> CIO, G, A, TR  <b>Mnemonics</b> OUT                      ! OUT

#### OUTPUT NOT: OUT NOT

Ladder Symbols	Operand Data Area
 	<b>B: Bit</b> CIO, G, A  <b>Mnemonics</b> OUT NOT                      ! OUT NOT

#### Description

OUT and OUT NOT are used to control the status of the designated bit according to the execution condition.

OUT turns ON the designated bit for an ON execution condition, and turns OFF the designated bit for an OFF execution condition. With a TR bit, OUT appears at a branching point rather than at the end of an instruction line. Refer to *4-5 Branching Instruction Lines* for details.

OUT NOT turns ON the designated bit for a OFF execution condition, and turns OFF the designated bit for an ON execution condition.

OUT and OUT NOT can be used to control execution by turning ON and OFF bits that are assigned to conditions on the ladder diagram, thus determining execution conditions for other instructions. This is particularly helpful and allows a complex set of conditions to be used to control the status of a single work bit, and then that work bit can be used to control other instructions.

The length of time that a bit is ON or OFF can be controlled by combining the OUT or OUT NOT with TIM. Refer to Examples under *5-13-1 TIMER: TIM* for details.

**Note:** Immediate-refresh variations can be used in simplex mode only; they cannot be used in duplex mode.

#### Precautions

Any output bit is generally used in only one instruction that controls its status.

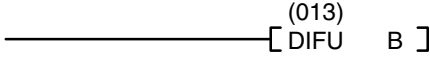
**Note:** Refer to page 101 for general precautions on operand data areas.

#### Flags

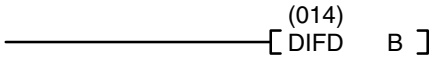
There are no flags affected by these instructions.

## 5-7-2 DIFFERENTIATE UP/DOWN: DIFU(013) and DIFD(014)

### DIFFERENTIATE UP: DIFU(013)

Ladder Symbol	Operand Data Area
	<b>B: Bit</b> CIO, G, A
<b>Variations</b> !DIFU(013)	

### DIFFERENTIATE DOWN: DIFD(014)

Ladder Symbol	Operand Data Area
	<b>B: Bit</b> CIO, G, A
<b>Variations</b> !DIFD(014)	

#### Description

DIFU(013) and DIFD(014) are used to turn the designated bit ON for one cycle only.

Whenever executed, DIFU(013) compares its current execution with the previous execution condition. If the previous execution condition was OFF and the current one is ON, DIFU(013) will turn ON the designated bit. If the previous execution condition was ON and the current execution condition is either ON or OFF, DIFU(013) will either turn the designated bit OFF or leave it OFF (i.e., if the designated bit is already OFF). The designated bit will thus never be ON for longer than one cycle, assuming it is executed each cycle (see *Precautions*, below).

Whenever executed, DIFD(014) compares its current execution with the previous execution condition. If the previous execution condition is ON and the current one is OFF, DIFD(014) will turn ON the designated bit. If the previous execution condition was OFF and the current execution condition is either ON or OFF, DIFD(014) will either turn the designated bit OFF or leave it OFF. The designated bit will thus never be ON for longer than one cycle, assuming it is executed each cycle (see *Precautions*, below).

These instructions are used when differentiated instructions (i.e., those prefixed with a ↑ or ↓,) are not available and single-cycle execution of a particular instruction is desired. They can also be used with non-differentiated forms of instructions that have differentiated forms when their use will simplify programming. Examples of these are shown below.

**Note:** Immediate-refresh variations can be used in simplex mode only; they cannot be used in duplex mode.

#### Precautions

Any output bit is generally used in only one instruction that controls its status. DIFU(013) and DIFD(014), operation can be uncertain when the instructions are programmed between IL and ILC, between JMP and JME, or in subroutines. Refer to 5-8 INTERLOCK and INTERLOCK CLEAR: IL(002) and ILC(003), 5-9 JUMP and JUMP END: JMP(004) and JME(005), and 5-30 Subroutines for details.

**Note:** Refer to page 101 for general precautions on operand data areas.

#### Flags

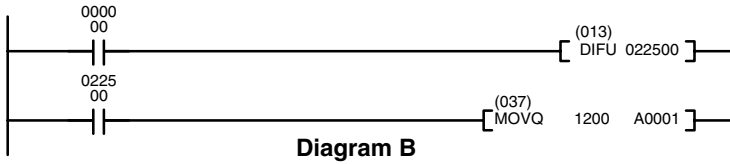
There are no flags affected by these instructions.

**Example 1: Use when There's No Differentiated Instruction**

In diagram A, below, whenever MOVQ(037) is executed with an ON execution condition it will move the contents of CIO 1200 to A001. If the execution condition remains ON, the content of A001 will be changed each cycle that the content of CIO 1200 changes. Diagram B, however, is an example of how DIFU(013) can be used to ensure that MOVQ(037) is executed only once each time the desired execution condition goes ON. Here, the contents of A001 will remain the same until CIO 022500 goes from OFF to ON.



Address	Instruction	Operands
00000	LD	000000
00001	MOVQ(037)	
		1200
		A001

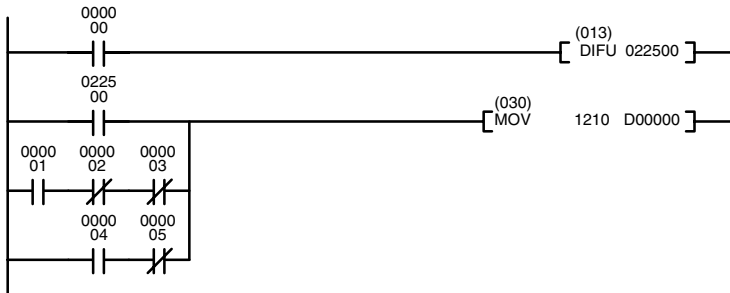


Address	Instruction	Operands
00000	LD	000000
00001	DIFU(013)	022500
00002	LD	022500
00003	MOVQ(037)	
		1200
		A001

**Note:** UP(018) and DOWN(019) can also be used to control differentiated execution of instructions. Refer to page 109 for details.

**Example 2: Use to Simplify Programming**

Although a differentiated form of MOV(030) is available, the following diagram would be very complicated to draw using it because only one of the conditions determining the execution condition for MOV(030) requires differentiated treatment.



Address	Instruction	Operands
00000	LD	000000
00001	DIFU(013)	022500
00002	LD	022500
00003	LD	000001
00004	AND NOT	000002
00005	AND NOT	000003
00006	OR LD	---
00007	LD	000004
00008	AND NOT	000005
00009	OR LD	---
00010	MOV(030)	
		1210
		D00000

### 5-7-3 SET and RESET: SET(016) and RSET(017)

#### SET: SET(016)

Ladder Symbol	Operand Data Area
	<b>B: Bit</b> CIO, G, A
<b>Variations</b> ↑SET(016)      ↓SET(016)      !SET(016) !↑SET(016)     ↓↓SET(016)	

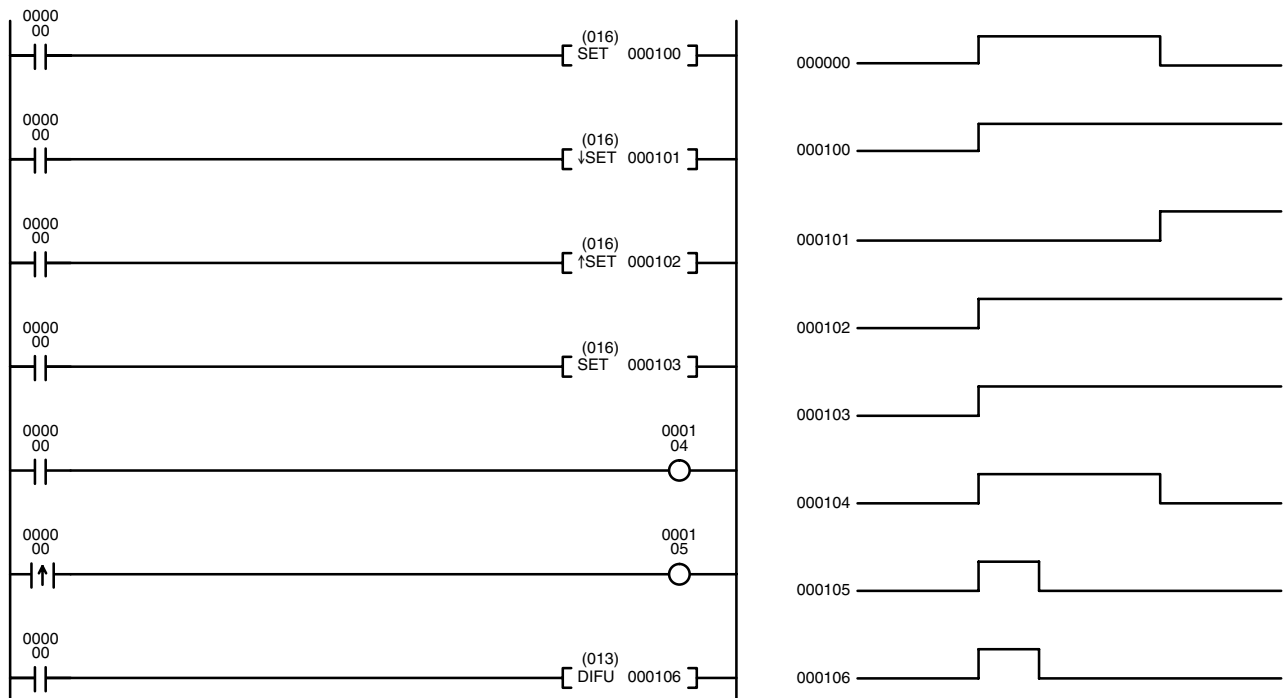
#### RESET: RSET(017)

Ladder Symbol	Operand Data Area
	<b>B: Bit</b> CIO, G, A
<b>Variations</b> ↑RSET(017)      ↓RSET(017)      !RSET(017) !↑RSET(017)     ↓↓RSET(017)	

#### Description

SET(016) turns the operand bit ON when the execution condition is ON, and does not affect the status of the operand bit when the execution condition is OFF. RSET(017) turns the operand bit OFF when the execution condition is ON, and does not affect the status of the operand bit when the execution condition is OFF. The operation of SET(016) differs from that of OUT because the OUT instruction turns the operand bit OFF when its execution condition is OFF. Likewise, RSET(017) differs from OUT NOT because OUT NOT turns the operand bit ON when its execution condition is OFF.

The following example shows the operations of the variations of SET(016).



**Note:** Immediate-refresh variations can be used in simplex mode only; they cannot be used in duplex mode.

**Precautions**

The status of operand bits for SET(016) and RSET (017) programmed between IL(002) and ILC(003) or JMP(004) and JME(005) will not change when the interlock or jump condition is met (i.e., when IL(002) or JMP(004) is executed with an OFF execution condition).

**Note:** Refer to page 101 for general precautions on operand data areas.

**Flags**

There are no flags affected by these instructions.

**Example**

In the example below, CIO 050000 is turned ON whenever CIO 000000 is ON, and turned OFF whenever CIO 000001 is ON.



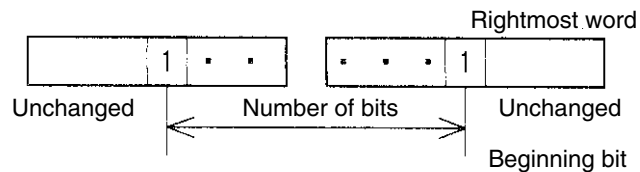
Address	Instruction	Operands
00000	LD	000000
00001	SET(016)	050000
00002	LD	050000
00003	OUT	005000
00004	LD	000001
00005	RSET(017)	050000

**5-7-4 MULTIPLE BIT SET/RESET: SETA(047)/RSTA(048)**

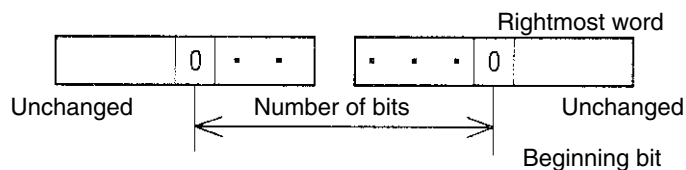
Ladder Symbol	Operand Data Area
	<p><b>D:</b> Rightmost word for set CIO, G, A</p> <p><b>N<sub>1</sub>:</b> Beginning bit CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>N<sub>2</sub>:</b> Number of bits CIO, G, A, T, C, #, DM, DR, IR</p>
<p><b>Variations</b></p> <p>↑SETA(047), ↑RSTA(048)</p>	

**Description**

When the execution condition is OFF, SETA(047) is not executed. When the execution condition is ON, SETA(047) turns ON a designated number of bits, beginning from the designated bit of the designated word, and continuing to the left (more-significant bits). All other bits are left unchanged.



When the execution condition is OFF, RSTA(048) is not executed. When the execution condition is ON, RSTA(048) turns OFF a designated number of bits, beginning from the designated bit of the designated word, and continuing to the left (more-significant bits). All other bits are left unchanged.



**Precautions**

N<sub>1</sub> must be between 0000 and 0015 and must be BCD. N<sub>2</sub> must be BCD.

**Note:** Refer to page 101 for general precautions on operand data areas.

**Flags**

ER (A50003): N<sub>1</sub> is not 0000 to 0015 BCD.  
 N<sub>2</sub> is not BCD.  
 Content of \*DM word is not BCD.

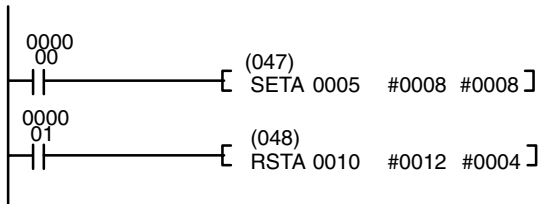
**Example 1**

**SETA Operation**

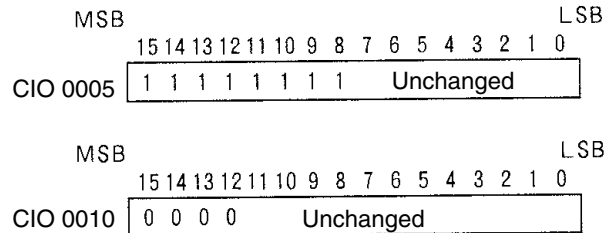
When CIO 000000 turns ON in the first instruction line in the following example, eight bits beginning with bit 08 in CIO 0005 are all turned ON.

**RSTA Operation**

When CIO 000001 turns ON in the second instruction line, the eight bits beginning with bit 12 in CIO 0010 are all turned OFF.



Address	Instruction	Operands
00000	LD	000000
00001	SETA(047)	
		0005
		#0008
		#0008
00002	LD	000001
00003	RSTA(048)	
		0010
		#0012
		#0004



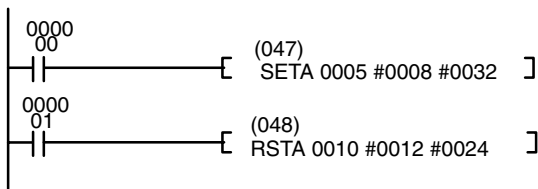
**Example 2**

**SETA Operation**

When CIO 000000 turns ON in the first instruction line in the following example, 32 bits beginning with bit 08 in CIO 0005 are all turned ON.

**RSTA Operation**

When CIO 000001 turns ON in the second instruction line, 24 bits beginning with bit 12 in CIO 0010 are all turned OFF.



Address	Instruction	Operands
00000	LD	000000
00001	SETA(047)	
		0005
		#0008
		#0032
00002	LD	000001
00003	RSTA(048)	
		0010
		#0012
		#0024



	MSB	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	LSB
CIO 0005	1	1	1	1	1	1	1	1	1	Unchanged								
CIO 0006	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
CIO 0007	Unchanged									1	1	1	1	1	1	1	1	1

	MSB	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	LSB
CIO 0010	0	0	0	0	Unchanged													
CIO 0011	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
CIO 0012	Unchanged												0	0	0	0		

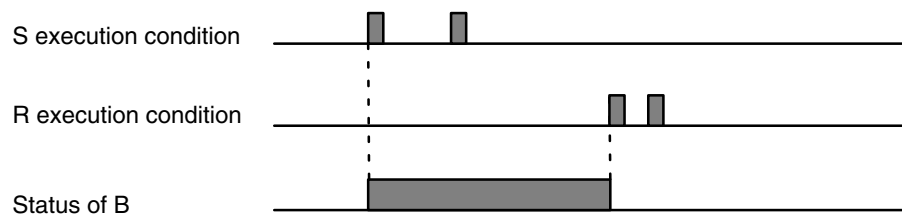
### 5-7-5 KEEP: KEEP(011)

<p><b>Ladder Symbol</b></p> <p><b>Variations</b> !KEEP(011)</p>	<p><b>Operand Data Area</b></p> <p><b>B: Bit</b>                      CIO, G, A</p>
---	---

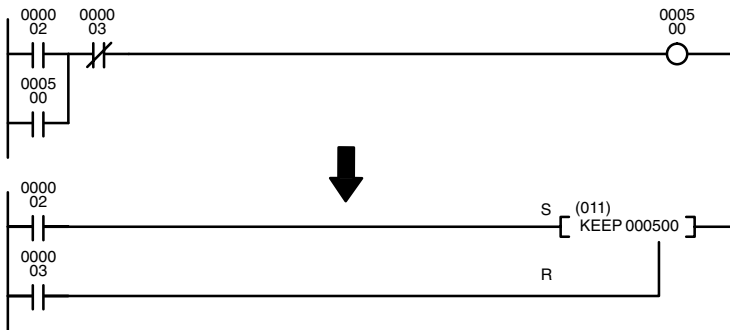
#### Description

KEEP(011) is used to maintain the status of the designated bit based on two execution conditions. These execution conditions are labeled S and R. S is the set input; R, the reset input. KEEP(011) operates like a latching relay that is set by S and reset by R.

When S turns ON, the designated bit will go ON and stay ON until reset, regardless of whether S stays ON or goes OFF. When R turns ON, the designated bit will go OFF and stay OFF until reset, regardless of whether R stays ON or goes OFF. The relationship between execution conditions and KEEP(011) bit status is shown below.



KEEP(011) operates like the self-maintaining bit described in 4-7-4 *Self-maintaining Bits (Seal)*. The following two diagrams would function identically, though the one using KEEP(011) requires one less instruction to program and would maintain status even in an interlocked program section.

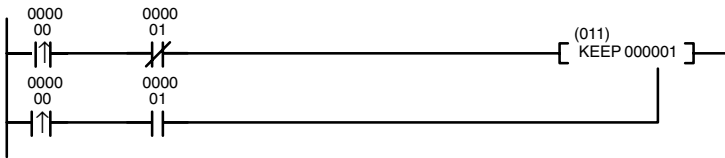


Address	Instruction	Operands
00000	LD	000002
00001	OR	000500
00002	AND NOT	000003
00003	OUT	000500

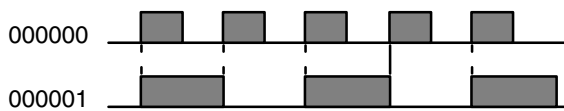
Address	Instruction	Operands
00000	LD	000002
00001	LD	000003
00002	KEEP(011)	000500

**Example**

KEEP(011) can be used to create flip-flops as shown below.



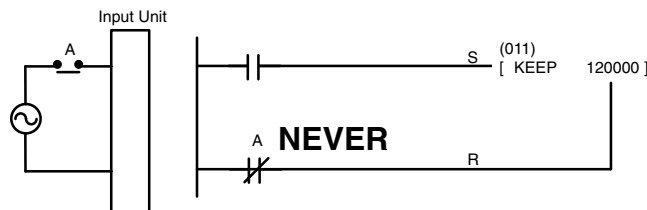
Address	Instruction	Operands
00000	LD↑	000000
00001	AND NOT	000001
00002	LD↑	000000
00003	AND	000001
00004	KEEP(011)	000001



**Note:** Immediate-refresh variations can be used in simplex mode only; they cannot be used in duplex mode.

**Precautions**

Any output bit is generally used in only one instruction that controls its status. Never use an input bit in a normally closed condition on the reset (R) for KEEP(011) when the input device uses an AC power supply. The delay in shutting down the PC's DC power supply (relative to the AC power supply to the input device) can cause the operand bit of KEEP(011) to be reset. This situation is shown below.



Bits used in KEEP are not reset in interlocks. Refer to the 5-8 *INTERLOCK and INTERLOCK CLEAR: IL(002) and ILC(003)* for details.

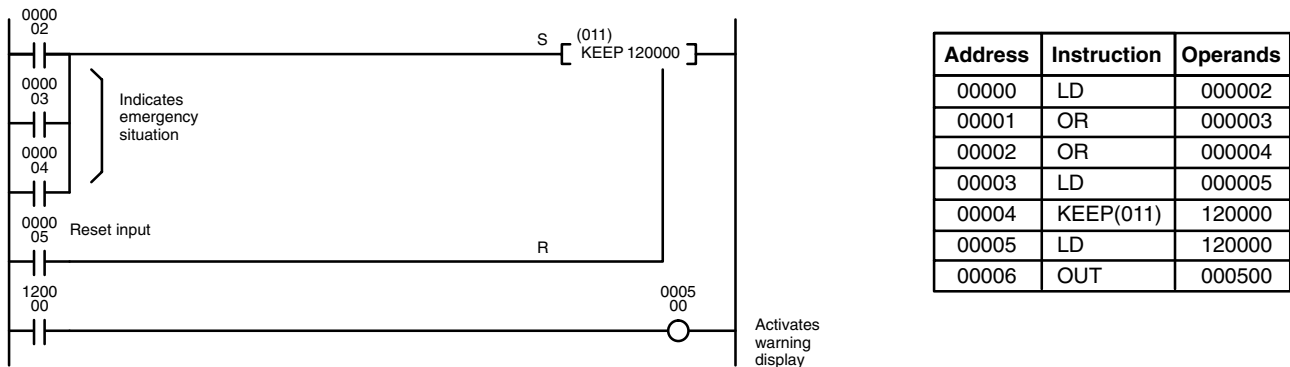
**Note:** Refer to page 101 for general precautions on operand data areas.

**Flags**

There are no flags affected by this instruction.

**Example**

If a holding bit (default range: CIO 1200 to CIO 1499) is used, bit status will be retained even during a power interruption. KEEP(011) can thus be used to program bits that will maintain status after restarting the PC following a power interruption. An example of this that can be used to produce a warning display following a system shutdown for an emergency situation is shown below. Bits 000002, 000003, and 000004 would be turned ON to indicate some type of error. Bit 000005 would be turned ON to reset the warning display. Bit 120000, which is turned ON when any one of the three bits indicates an emergency situation, is used to turn ON the warning indicator through 000500.

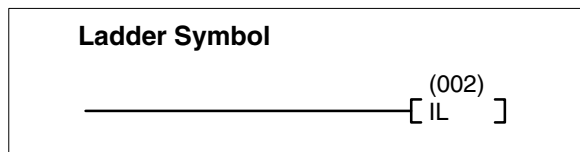


The status of I/O Area bits can be retained in the event of a power interruption by turning ON the IOM Hold Bit and setting IOM Hold Bit Hold in the PC Setup. If the IOM Hold Bit is not specified to be held in the PC Setup, all I/O Area bits will be turned OFF when the power is turned ON. Be sure to restart the PC after changing the PC Setup; otherwise the new settings will not be used.

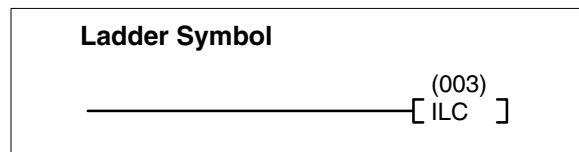
KEEP(011) can also be combined with TIM to produce delays in turning bits ON and OFF. Refer to 5-13-1 *TIMER: TIM* for details.

## 5-8 INTERLOCK and INTERLOCK CLEAR: IL(002) and ILC(003)

### INTERLOCK: IL(002)



### INTERLOCK CLEAR: ILC(003)



### Description

IL(002) is always used in conjunction with ILC(003) to create interlocks. Interlocks are used to create program sections that are executed normally when a specific execution condition is ON or reset when the specific execution condition is OFF. Logically, the treatment is similar to enabling branching with TR bits, but treatment of instructions between IL(002) and ILC(003) differs from that with TR bits when the execution condition for IL(002) is OFF. The execution condition of IL(002) is call the interlock condition and controls execution of the interlocked section of program. When the interlock condition is ON, the program will be executed as written.

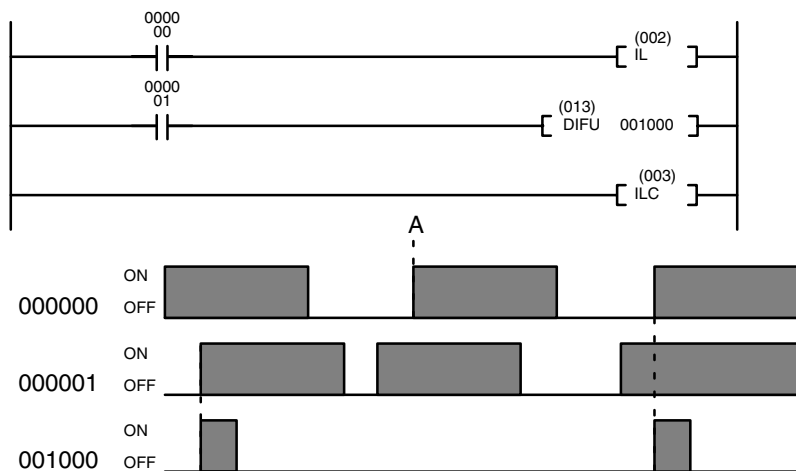
If the execution condition for IL(002) is OFF, the interlocked section between IL(002) and ILC(003) will be treated as shown in the following table:

Instruction	Treatment
OUT and OUT NOT	Designated bit turned OFF.
TIM, TIMH(015), and TIML(121)	Reset.
CNT, CNTR(012), TTIM(120), and MTIM(122)	PV maintained.
KEEP(011), SFT(050)	Bit status maintained.
DIFU(013) and DIFD(014)	Not executed (see below).
All others	Not executed.

IL(002) and ILC(003) do not necessarily have to be used in pairs. IL(002) can be used several times in a row, with each IL(002) creating an interlocked section through the next ILC(003). ILC(003) cannot be used unless there is at least one IL(002) between it and any previous ILC(003).

**Differentiation in Interlocks**

Changes in the execution condition for DIFU(013), DIFD(014), or a differentiated instruction are not recorded if the DIFU(013) or DIFD(014) is in an interlocked section and the execution condition for the IL(002) is OFF. When DIFU(013), DIFD(014), or a differentiated instruction is executed in an interlocked section immediately after the execution condition for the IL(002) has gone ON, the execution condition for the DIFU(013), DIFD(014), or differentiated instruction will be compared to the execution condition that existed before the interlock became effective (i.e., before the interlock condition for IL(002) went OFF). The ladder diagram and bit status changes for a DIFU(013) instruction in an interlock are shown below. The interlock is in effect while 000000 is OFF. Bit 001000 is not turned ON at the point labeled A even though 000001 has turned OFF and then back ON because the OFF status of 000001 just before A was not detected while the interlock condition was OFF.



Address	Instruction	Operands
00000	LD	000000
00001	IL(002)	
00002	LD	000001
00003	DIFU(013)	001000
00004	ILC(003)	

**Precautions**

There must be an ILC(003) following any one or more IL(002). Although as many IL(002) instructions as are necessary can be used with one ILC(003), ILC(003) instructions cannot be used consecutively without at least one IL(002) in between, i.e., nesting is not possible. Whenever a ILC(003) is executed, all interlocks between the active ILC(003) and the preceding ILC(003) are cleared. When more than one IL(002) is used with a single ILC(003), an error message will appear when the program check is performed, but execution will proceed normally.

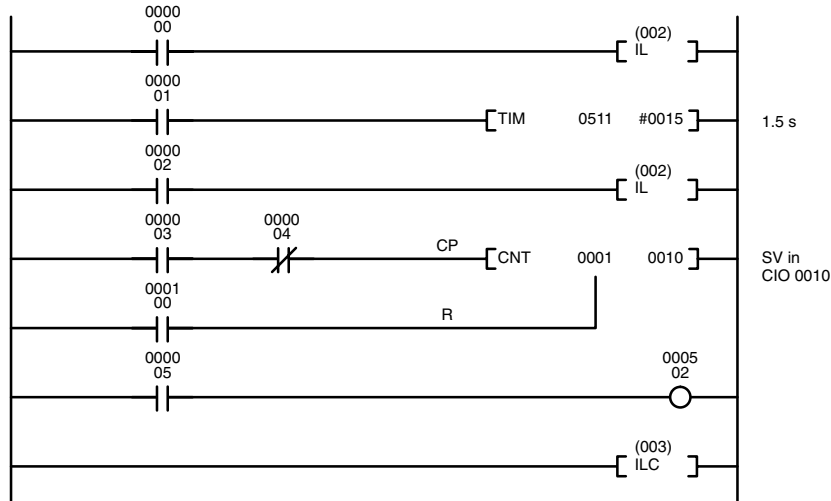
**Note:** Refer to page 101 for general precautions on operand data areas.

**Flags**

There are no flags affected by these instructions.

**Example**

The following diagram shows IL(002) being used twice with one ILC(003).



Address	Instruction	Operands
00000	LD	000000
00001	IL(002)	
00002	LD	000001
00003		T00511
		#0015
00004	LD	000002
00005	IL(002)	
00006	LD	000003
00007	AND NOT	000004
00008	LD	000100
00009		C00001
		0010
00010	LD	000005
00011	OUT	000502
00012	ILC(003)	

When the execution condition for the first IL(002) is OFF, T0511 will be reset to 1.5 s, C0001 will not be changed, and 000502 will be turned OFF. When the execution condition for the first IL(002) is ON and the execution condition for the second IL(002) is OFF, T0511 will be executed according to the status of 000001, C0001 will not be changed, and 000502 will be turned OFF. When the execution conditions for both the IL(002) are ON, the program will execute as written.

## 5-9 JUMP and JUMP END: JMP(004) and JME(005)

### JUMP: JMP(004)

Ladder Symbol	Operand Data Area
	<b>N: Jump number</b> CIO, G, A, T, C, #, DM, DR, IR

### JUMP END: JME(005)

Ladder Symbol	Operand Data Area
	<b>N: Jump number</b> #

### Description

JMP(004) is always used in conjunction with JME(005) to create jumps, i.e., to skip from one point in a ladder diagram to another point. JMP(004) defines the point from which the jump will be made; JME(005) defines the destination of the jump. When the execution condition for JMP(004) is ON, no jump is made and the program is executed consecutively as written.

When the execution condition for JMP(004) is OFF, program execution will go immediately to the first JME(005) in the program with the same jump number without executing any instructions in between (See *JUMP 0000*, below, for exception). The status of timers, counters, bits used in OUT, bits used in OUT NOT, and all other status controlled by the instructions between JMP(004) and JME(005) will not be changed. Because all of instructions between JMP(004) and JME(005) are skipped, jumps can be used to reduce cycle time.

Only one JME(005) instruction per jump number should be used in a program. If two or more JME(005) instructions with the same jump number are used in a program, program execution will skip to the JME(005) instruction at the lowest program address, even if it precedes the JMP(004) instruction. Programming multiple JMP(005) instructions for the same JME(004) instruction can be useful in programming.

**Differentiation in Jumps**

Although DIFU(013) and DIFD(014) are designed to turn ON the designated bit for one cycle, they will not necessarily do so when written between JMP(004) and JME(005). Once DIFU(013) or DIFD(014) has turned ON a bit, it will remain ON until the next time DIFU(013) or DIFD(014) is executed again. In normal programming, this means the next cycle. In a jump, this means the next time the jump from JMP(004) to JME(005) is not made, i.e., if a bit is turned ON by DIFU(013) or DIFD(014) and then a jump is made in the next cycle so that DIFU(013) or DIFD(014) are skipped, the designated bit will remain ON until the next time the execution condition for the JMP(004) controlling the jump is ON. When DIFU(013), DIFD(014), or a differentiated instruction is executed in a jumped section immediately after the execution condition for the JMP(004) has gone ON, the execution condition for the DIFU(013), DIFD(014), or differentiated instruction will be compared to the execution condition that existed before the jump became effective (i.e., before the execution condition for JMP(004) went OFF).

**JUMP 0000**

The PC Setup can be used to control the operation of jumps created using jump number 0000. If multiple jumps with 0000 are disabled, jumps created with 0000 will operate as described above. If multiple jumps are enabled, any JMP 0000 instruction will jump to the next JME 0000 in the program (and not the first JME 0000 in the program). When multiple jumps for 0000 are enabled, you cannot overlap or nest the jumps, i.e., each JMP 0000 must be followed by a JME 0000 before the next JMP 0000 in the program and each JME 0000 must be followed by a JMP 0000 before the next JME 0000 in the program.

Even if the JMP condition is OFF, all instructions between JMP 0000 and JME 0000 are still processed as NOPs, increasing the cycle time accordingly.

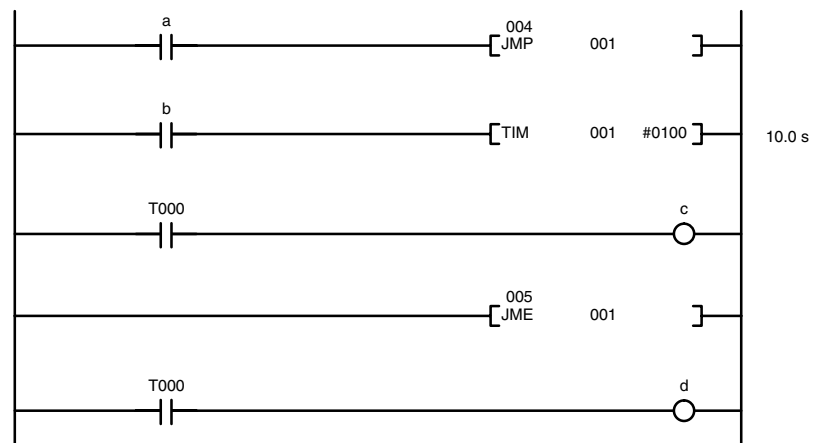
If the JMP condition is OFF when JMP 0001 through JMP 0999 are being used, the program will jump directly to JME(005). Any instructions between JMP(004) and JME(005) are not executed at all, and the cycle time is shortened accordingly.

**Timer Operation in Jumps**

The operation of timers in CVM1D PCs is the same as in C-series, CVM1, and CV-series PCs, although timer PVs are refreshed differently in jumped program sections. The following table describes the operation of timers in a jumped program section, i.e., when the execution condition for the JMP(004) controlling the jump is OFF.

Instruction	CVM1D Operation	C/CVM1/CV-series Operation
TIMER: TIM	<ul style="list-style-type: none"> <li>Count decrementing continues while the timer's execution condition is ON, but the timer PV is not refreshed.</li> <li>When the execution condition for JMP(004) goes ON, the elapsed time (while the timer was jumped) is subtracted and count decrementing continues.</li> </ul>	<ul style="list-style-type: none"> <li>Count decrementing continues and the timer PV is refreshed while the timer's execution condition is ON.</li> <li>The Timer Completion Flag will not be turned ON while the timer is jumped, even if the timer times out.</li> <li>If the timer has timed out while jumped, the Completion Flag is turned ON when the execution condition for JMP(004) goes ON.</li> </ul>
HIGH-SPEED TIMER: TIMH(015)	<ul style="list-style-type: none"> <li>The Timer Completion Flag will not be turned ON while the timer is jumped, even if the timer times out.</li> <li>If the timer has timed out while jumped, the Completion Flag is turned ON when the execution condition for JMP(004) goes ON.</li> </ul>	
ACCUMULATIVE TIMER: TTIM(120)	<ul style="list-style-type: none"> <li>Count incrementing continues while the timer's execution condition is ON, but the timer PV is not refreshed.</li> <li>When the execution condition for JMP(004) goes ON, the elapsed time (while the timer was jumped) is added and count incrementing continues.</li> <li>The Timer Completion Flag will not be turned ON while the timer is jumped, even if the timer times out.</li> <li>If the timer has timed out while jumped, the Completion Flag is turned ON when the execution condition for JMP(004) goes ON.</li> </ul>	<ul style="list-style-type: none"> <li>Timing is disabled when the execution condition for JMP(004) is OFF, even if the timer's execution condition is ON.</li> </ul>
LONG TIMER: TIML(121)		
MULTI-OUTPUT TIMER: MTIM(122)	<ul style="list-style-type: none"> <li>Timing is disabled when the execution condition for JMP(004) is OFF, even if the timer's execution condition is ON.</li> </ul>	

The following program example demonstrates the operation of a timer instruction programmed between JMP(004) and JME(005).



- The timer PV is refreshed every scan during program execution.
- While the timer is jumped (execution condition of JMP(004) OFF), timing (decrementing or incrementing) is not started and the timer PV is not refreshed even if the timer's execution condition (b) is ON.

When the JMP(004) instruction's execution condition is turned OFF after timing has been started by turning ON the timer's execution condition (b), timing will continue but the timer PV will not be refreshed.  
 Be sure to consider the effects of the JMP(004)/JME(005) instructions on timer PVs when using the timer PVs as data in the program.

- As long as a timer is jumped (execution condition of JMP(004) OFF), its Timer Completion Flag will not be turned ON even if the timer's SV has elapsed.
- When an operating timer has been jumped and the execution condition of JMP(004) is then turned ON again, the elapsed time is subtracted from the decrementing timer's PV (or added to an incrementing timer's PV) and the timing continues.  
 If the timer's SV has elapsed while it was jumped, its Completion Flag will be turned ON when the execution condition of JMP(004) is turned ON again.

**Precautions**

The jump number N must be BCD between 0000 and 0999.

When JMP(004) and JME(005) are not used in pairs, an error message will appear when the program check is performed. If a JME(005) instruction precedes a JMP(004) instruction with the same jump number, a loop might occur, so the END(001) instruction is never executed, causing a Cycle Time Over error.

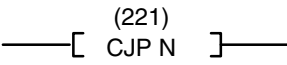
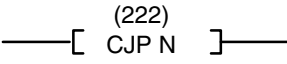
**Flags**

ER (A50003): Content of \*DM word is not BCD when set for BCD.  
 Jump number is not BCD or not between 0000 and 0999.  
 JMP(004) in the program without a corresponding JME(005).  
 Also turns ON the Jump Error Flag A40213.

**Example**

Examples of jump programs are provided in *4-6 Jumps*.

## 5-10 CONDITIONAL JUMP: CJP(221)/CJPN(222)

Ladder Symbol	Operand Data Areas
	<b>N: Jump number</b> CIO, G, A, T, C, #, DM, DR, IR
	

**Description**

CJP(221) operates in the reverse of JMP(004). When the execution condition turns ON, the program up until JME(005) is skipped. When the execution condition is OFF, the instructions after CJP(221) are executed normally.

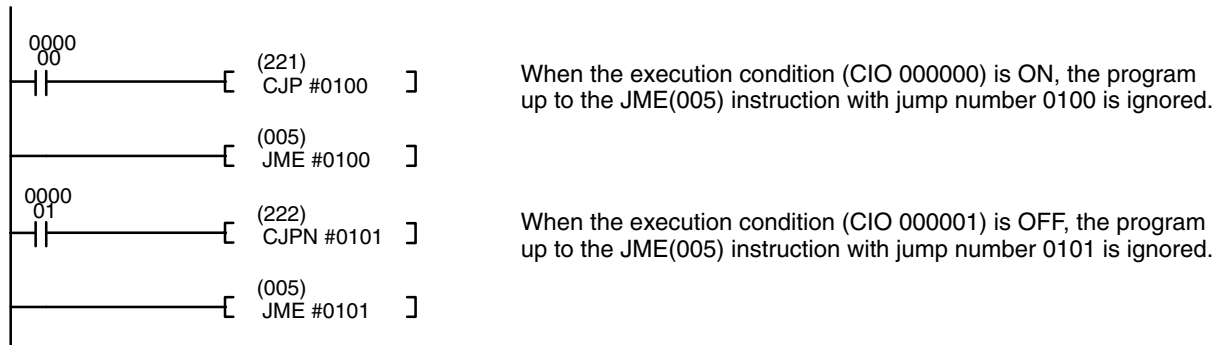
The CJPN(222) operates similar to JMP(004). When the execution condition is ON, the instructions after CJPN(222) are executed normally. When the execution condition is OFF, the program up until JME(005) is skipped.

JMP(004), CJP(221), and CJPN(222) all operate differently, however, when used in a block program. With JMP(004), the program jumps to JME(005) unconditionally. With CJP(221), the program jumps to JME(005) when the condition just before the CJP(221) instruction is ON. With CJPN(222), the program jumps to JME(005) when the condition just before the CJP(221) instruction is OFF.

When a jump occurs, the status of outputs from the program (output bits, timers, counters, shift registers, keep, etc.) is maintained and timing continues for TIM/TIMH(015).



If there are two or more JME(005) instructions in a program for the same jump number, the one at the lower address is valid and the ones at higher addresses are ignored.



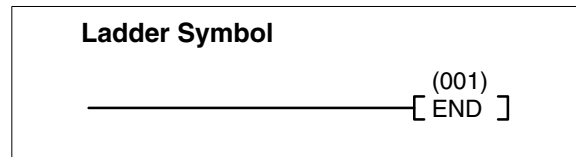
**Precautions**

The jump number (N) must be BCD between 0000 and 0999.

**Flags**

ER (A50003): Content of \*DM word is not BCD when set for BCD.  
 Jump number is not BCD or not between 0000 and 0999.  
 CJP(221) or CJPN(222) in program without a corresponding JME(005). Also turns ON the Jump Error Flag A40213.

### 5-11 END: END(001)



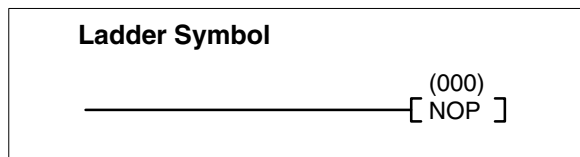
**Description**

END(001) is required as the last instruction in any program. No instruction written after END(001) will be executed. The END(001) instruction indicates the end of the entire ladder program for that cycle.  
 If there is no END(001) in the program, no instructions will be executed and the error message "NO END INST" will appear.

**Flags**

END(001) turns OFF the ER, CY, GR, EQ, LE, and N Flags.

### 5-12 NO OPERATION: NOP(000)



**Description**

NOP(000) is not generally required in programming. When NOP(000) is found in a program, nothing is executed and the program execution moves to the next instruction. When memory is cleared prior to programming, NOP(000) is written at all addresses.

**Precautions**

NOP(000) can only be used with mnemonic display, and not with ladder programs.

**Example**

NOP(000) can be inserted in a program at the position where an instruction is to be inserted later. Then when the instruction is inserted there will be no gap in the addresses.

**Flags**

There are no flags affected by NOP(000).

## 5-13 Timer and Counter Instructions

### Timers

The timer instructions in this section are used to create timers. Most timers require a timer number and a set value (SV). Timer numbers run from T0000 through T1023, and are used to access timer PVs and Completion Flags in memory areas set aside specifically for this purpose.

TIML(121) and MTIM(122) do not require timer numbers. The PVs and Completion Flags for these timers are contained in addresses specified by the user when inputting the instructions.

Any one timer number cannot be defined twice, i.e., once it has been used in any of the timer instructions it cannot be used again unless the two timers are never active simultaneously. If two timers share a single timer number, but are not used simultaneously, a duplication error will be generated when the program is checked, but the timers will operate normally. Once defined, a timer number can be used as many times as required as an operand in other instructions to access the present value and Completion Flag of the timer.

### Counters

The counter instructions in this section are used to create counter. Most counters require a counter number and a SV, and are connected to multiple instruction lines which serve as input signals, resets, etc.

Any one counter number cannot be defined twice, i.e., once it has been used in any of the counter instructions it cannot be used again unless the two counters are never active simultaneously. If two counters share a single counter number, but are not used simultaneously, a duplication error will be generated when the program is checked, but the counters will operate normally. Once defined, a counter number can be used as many times as required as an operand in other instructions to access the present value and Completion Flag of the counter.

### Set Values

A timer or counter SV can be input as a constant or as a word address in a data area. If an I/O Area word assigned to an Input Unit is designated as the word address, the Input Unit can be wired so that the SV can be set externally through thumbwheel switches or similar devices. Timers wired in this way can only be set externally during RUN or MONITOR mode. All SVs, including those set externally, must be in BCD.

Although set values may be set to 0 for timers and counters, it will disable them, i.e., turn ON the Completion Flag immediately.

### T/C Numbers as Operands

No prefix is required when using a timer or counter number as a definer in a timer or counter instruction. Once a timer or counter number has been used to create a timer/counter, it can be prefixed with T or C for use as an operand in various instructions.

Timer and counter numbers can be designated as operands that require either bit or word data. When designated as an operand that requires bit data, the timer or counter number accesses a bit that functions as a **Completion Flag** that indicates when the timer or counter has completed counting, i.e., the Completion Flag, which is normally OFF, will turn ON when the timer has timed out or counter counted out.

When designated as an operand that requires word data, the timer or counter number accesses a memory location that holds the present value (PV) of the timer or counter. The PV of a timer or counter can thus be used as an operand in CMP(020), or any other instruction for which the Timer or Counter Area is allowed.

Note that "T0000" is used to designate both the Completion Flag for the timer and to designate the PV of the timer. The meaning of the term in context should be clear, i.e., the first is always a bit operand and the second is always a word operand. The same is true of all other timer or counter numbers.

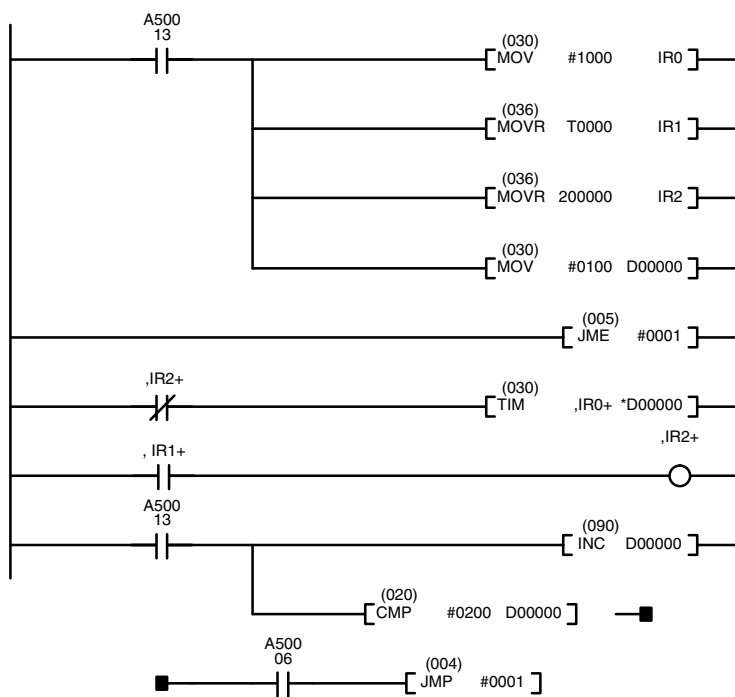
Indirect Addressing

10.Timer and counter numbers for TIM, TIMH(015), TTIM(120), CNT, CNTR(012), TIMW<013>, CNTW<014>, and TMHW<015> can be indirectly addressed using the Index Registers by moving the PC memory address of the PV of the timer or counter number to the Index Register. PVs for timers T0000 through T1023 are contained in PC memory addresses \$1000 through \$13FF, and PVs for counters C0000 through C1023 are contained in PC memory addresses \$1800 through \$1BFF. MOVR(036) can be used to move memory addresses for Completion Flags to Index Registers.

**Caution** If the Index Register doesn't contain a valid address for a timer or counter PV, the instruction will not be executed, and the ER (A50003) Flag will **not** be turned ON.

The following example shows a program section that uses indirect addressing to define and start 100 timers with SVs contained in D00100 through D00199. IR0 contains the PC memory address of the timer PV and IR1 contains the PC memory address of the timer Completion Flag.

DM address	Content	Function
D00100	0010	SV for T0000
D00101	0100	SV for T0001
D00102	0050	SV for T0002
.	.	.
.	.	.
.	.	.
D00199	0999	SV for T0099



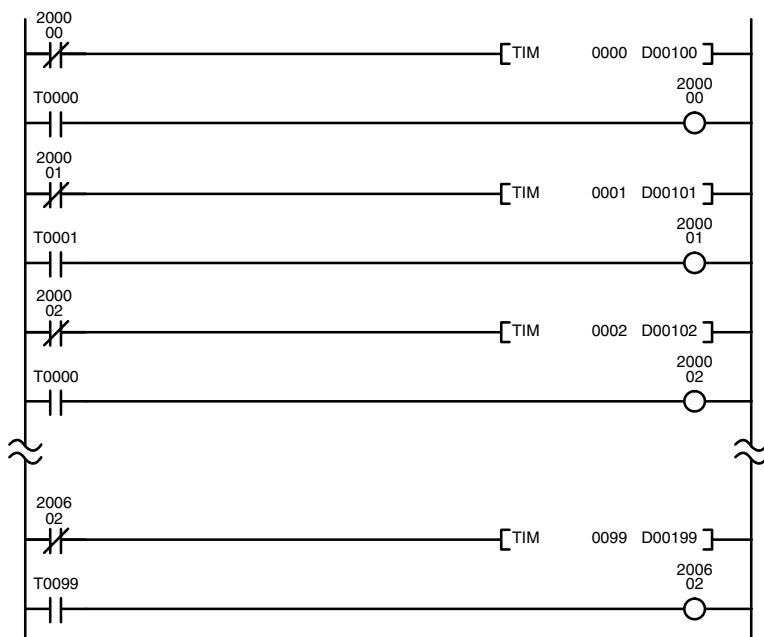
Address	Instruction	Operands
00000	LD	A50013
00001	MOV(030)	
		#1000
		IR0
00002	MOVR(036)	
		T0000
		IR1
00003	MOVR(036)	
		200000
		IR2
00004	MOV(030)	#0100
		D00000
00005	JME(005)	#0001
00006	LD NOT	,IR2+
00007	TIM	,IR0+
		*D00000
00008	LD	,IR1+
00009	OUT	,IR2+
00010	LD	A50013
00011	INC(090)	
		D00000
00012	CMP(020)	
		#0200
		D00000
00013	LD	A50006
00014	JMP(004)	#0001

**Caution** Do not use jump number 0000 in the above type of programming.

The first MOV(030) instruction moves the PC memory address of the PV for timer T0000 (\$1000) to IR0. The first MOVR(036) instruction moves the PC memory address of the Completion Flag for timer T0000 to IR1, and the second one moves the starting address into IR2. The second MOV(030) instruction moves the address (00100) of the DM word that contains the SV for timer T0000 to D00000. A50013 is an Always ON Flag.

JME(005) and JMP(004) form a loop in which the content of IR0, IR1, and D00000 are incremented by one each time the program executes the loop, successively defining and starting the 100 timers T0000 through T0199. The loop continues until the content of D00000 is 0200, i.e., until all 100 timers have been defined and started. A50006 is the Equals Flag.

The subroutine above is equivalent to the 400 instructions below.



Address	Instruction	Operands
00000	LD NOT	200000
00001	TIM	0000
		D00100
00002	LD	T0000
00003	OUT	200000
00004	LD NOT	200001
00005	TIM	0001
		D00101
00006	LD	T0001
00007	OUT	200001
00008	LD NOT	200002
00009	TIM	0002
		D00102
00010	LD	T0002
00011	OUT	200002
~ ~ ~		
00396	LD NOT	200602
00397	TIM	0099
		D00199
00398	LD	T0000
00399	OUT	200602

### 5-13-1 TIMER: TIM

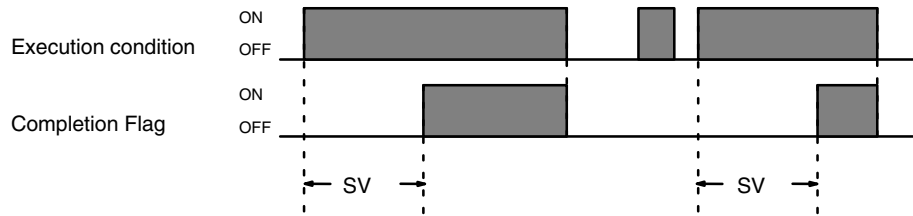
<p><b>Ladder Symbol</b></p> <p>— [ TIM    N    S ]</p>	<p><b>Operand Data Areas</b></p> <p><b>N: Timer number</b>    #</p> <p><b>S: Set value</b>        CIO, G, A, T, C, #, DM, DR, IR</p> <p>*Refer to page 128 for details on indirectly addressing timers.</p>
--	---

**Description**

A timer is activated when its execution condition goes ON and is reset (to SV) when the execution condition goes OFF. Once activated, TIM measures in units of 0.1 second from the SV.

If the execution condition remains ON long enough for TIM to time down to zero, the Completion Flag for the timer number used will turn ON and will remain ON until TIM is reset (i.e., until its execution condition goes OFF or CNR(236) is executed).

The following figure illustrates the relationship between the execution condition for TIM and the Completion Flag assigned to it.



**Differences in CVM1D and C/CVM1/CV-series PCs**

Operation of the TIM instruction is the same in CVM1D and C/CVM1/CV-series PCs except for the timer precision and PV refreshing in jumped program sections.

1, 2, 3...

1. Timer Precision

TIMER instruction errors can be one cycle time or more (up to two cycle times). The error range is  $\pm (10 \text{ ms} + \text{cycle time})$ .

2. Operation of the TIMER Instruction between JMP(004) and JME(005)

- The timer PV is refreshed every cycle during program execution.
- The timer operates normally while the execution condition of JMP(004) is ON.
- The timer PV is not refreshed while the timer is jumped (while the execution condition of JMP(004) is OFF.)  
If the timer is jumped after it has started timing, the timer count will continue decrementing but the PV will not be refreshed.  
Be sure to consider the effects of the JMP(004)/JME(005) instructions on timer PVs when using the timer PVs as data in the program.
- The Completion Flag will not be turned ON while a timer is jumped, even if the SV elapses.
- When the execution condition for JMP(004) is changed to ON, a timer that had been operating will continue timing down from the full time that elapsed since the timer's execution condition went ON. If the SV elapsed while the timer was jumped, the timer's Completion Flag will be turned ON when the execution condition for JMP(004) is turned ON again.

**Precautions**

SV must be between 000.0 and 999.9 and must be BCD. The decimal point is not entered.

Each timer number can be used to define only one timer instruction unless the timers are never active simultaneously. The TIM, TIMH(015), and TTIM(120) timers all use timer numbers.

Power interruptions reset timers. If a timer that is not reset by power interruptions is required, Auxiliary Area clock pulse bits can be counted to produce timers using CNT. Refer to 5-13-6 COUNTER: CNT for details.

If the IOM Hold Bit and PC Setup are set to retain IOM (which includes timer PVs and Completion Flags), the timer must be reset before starting it to prevent possible malfunctions.

If you convert a TIM instruction to a TIMH(015) instruction (or vice versa) using online programming operations, always reset the instruction's Completion Flag. Proper operation will not be possible unless the Completion Flag is reset.

**Note:** Refer to page 101 for general precautions on operand data areas.

**Flags**

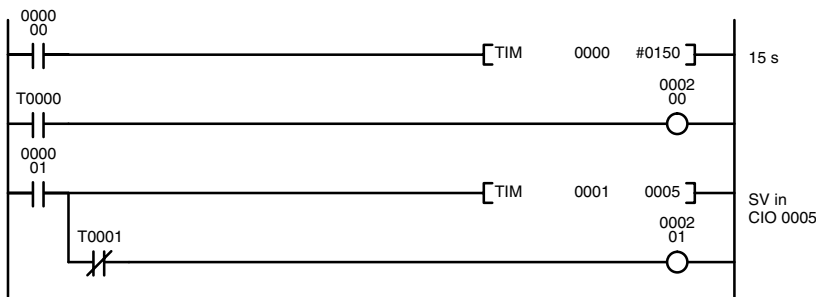
ER (A50003): Content of \*DM word is not BCD when set for BCD.  
Content of S (SV) is not BCD.

**Examples**

All of the following examples use OUT in diagrams that would generally be used to control output bits in the I/O Area. These diagrams can be modified to control execution of other instructions.

**Example 1:  
Basic Application**

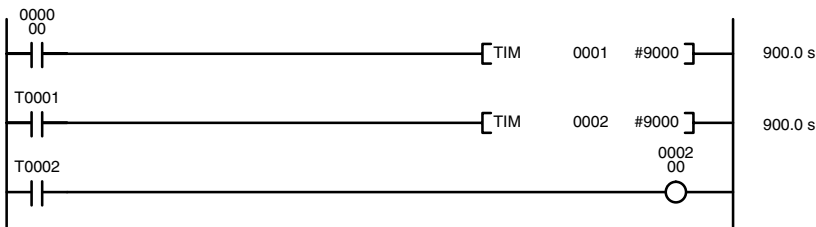
The following example shows two timers, one set with a constant and one set via input word 0005. Here, 000200 will be turned ON after 000000 goes ON and stays ON for at least 15 seconds. When 000000 goes OFF, the timer will be reset and 000200 will be turned OFF. When 000001 goes ON, T0001 is started from the SV provided through word 0005. Bit 000201 is also turned ON when 000001 goes ON. When the SV in 0005 has timed out, 000201 is turned OFF. This bit will also be turned OFF when T0001 is reset, regardless of whether or not SV has expired.



Address	Instruction	Operands
00000	LD	000000
00001	TIM	0000
		#0150
00002	LD	T0000
00003	OUT	000200
00004	LD	000001
00005	TIM	0001
		0005
00006	AND NOT	T0001
00007	OUT	000201

**Example 2:  
Extended Timers**

There are three ways to achieve timers that operate for longer than 999.9 seconds. One way is to use TIML(121). The second way is to program consecutive timers, with the Completion Flag of each timer used to activate the next timer. A simple example with two 900.0-second (15-minute) timers combined to functionally form a 30-minute timer is shown below.



Address	Instruction	Operands
00000	LD	000000
00001	TIM	0001
		#9000
00002	LD	T0001
00003	TIM	0002
		#9000
00004	LD	T0002
00005	OUT	000200

In this example, 000200 will be turned ON 30 minutes after 000000 goes ON assuming that 000000 stays ON.

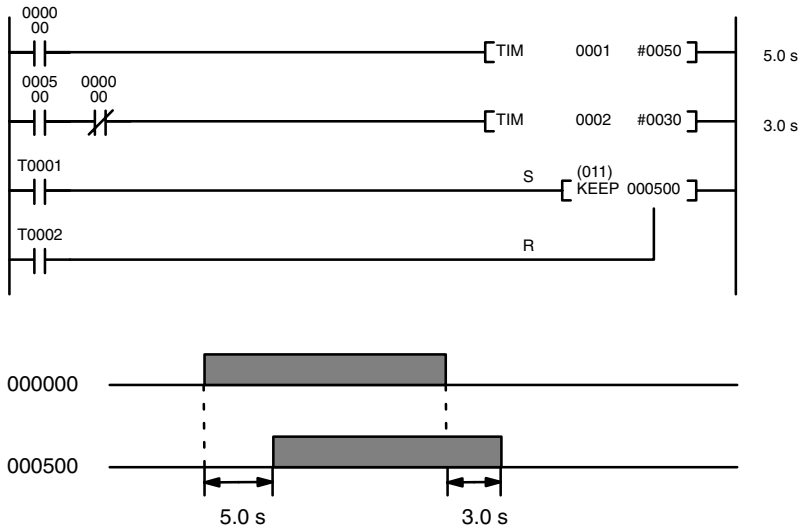
The third way to create longer timers is to combined TIM with CNT or to used CNT to count Auxiliary Area clock pulse bits to produce longer timers. An example is provided in 5-13-6 COUNTER: CNT.

**Example 3:  
ON/OFF Delays**

TIM can be combined with KEEP(011) to delay turning a bit ON and OFF in reference to a desired execution condition. KEEP(011) is described in 5-7-5 KEEP: KEEP(011).

To create delays, the Completion Flags for two TIM are used to determine the execution conditions for setting and resetting the bit designated for KEEP(011). The bit whose manipulation is to be delayed is used in KEEP(011). Turning ON and OFF the bit designated for KEEP(011) is thus delayed by the SV for the two TIM timers. The two SV could naturally be the same if desired.

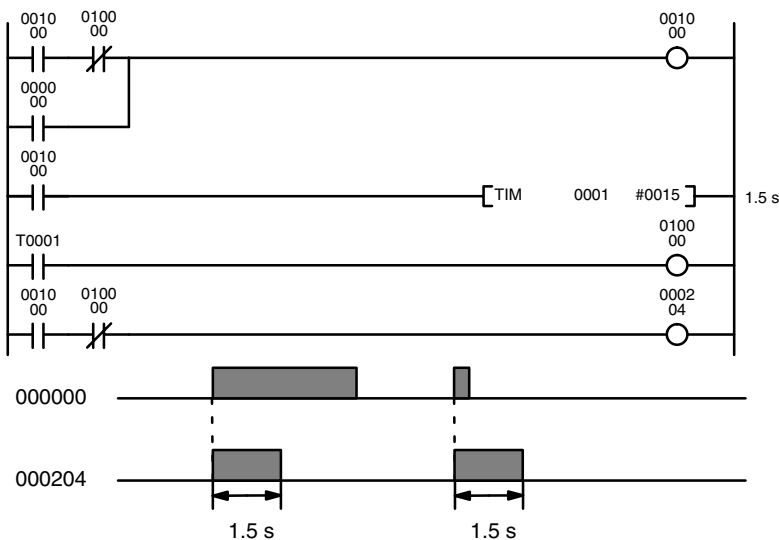
In the following example, 000500 would be turned ON 5.0 seconds after 000000 goes ON and then turned OFF 3.0 seconds after 000000 goes OFF. It is necessary to use both 000500 and 000000 to determine the execution condition for T0002; 000000 in a normally closed condition is necessary to reset T0002 when 000000 goes ON and 000500 is necessary to activate T0002 (when 000000 is OFF).



Address	Instruction	Operands
00000	LD	000000
00001	TIM	0001 #0050
00002	LD	000500
00003	AND NOT	000000
00004	TIM	0002 #0030
00005	LD	T0001
00006	LD	T0002
00007	KEEP(011)	000500

**Example 4:  
One-Shot Bits**

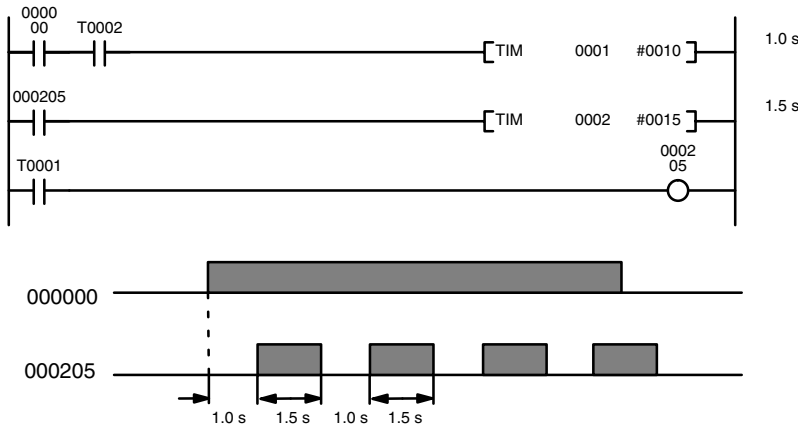
The length of time that a bit is kept ON or OFF can be controlled by combining TIM with OUT or OUT NOT. The following diagram demonstrates how this is possible. In this example, 000204 would remain ON for 1.5 seconds after 000000 goes ON regardless of the time 000000 stays ON. This is achieved by using 001000 as a self-maintaining bit activated by 000000 and turning ON 000204 through it. When T0001 comes ON (i.e., when the SV of T0001 has expired), 000204 will be turned OFF through T0001 (i.e., T0001 will turn ON which, as a normally closed condition, creates an OFF execution condition for OUT 000204).



Address	Instruction	Operands
00000	LD	001000
00001	AND NOT	010000
00002	OR	000000
00003	OUT	001000
00004	LD	001000
00005	TIM	0001 #0015
00006	LD	T0001
00007	OUT	010000
00008	LD	001000
00009	AND NOT	010000
00010	OUT	000204

**Example 5:  
Flicker Bits**

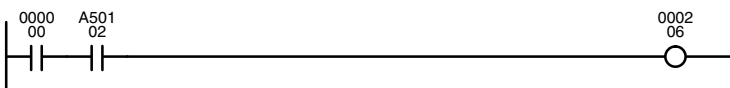
Bits can be programmed to turn ON and OFF at regular intervals while a designated execution condition is ON by using TIM twice. One TIM functions to turn ON and OFF a specified bit, i.e., the Completion Flag of this TIM turns the specified bit ON and OFF. The other TIM functions to control the operation of the first TIM, i.e., when the first TIM's Completion Flag goes ON, the second TIM is started and when the second TIM's Completion Flag goes ON, the first TIM is started.



Address	Instruction	Operands
00000	LD	000000
00001	AND	T0002
00002	TIM	0001
		#0010
00003	LD	000205
00004	TIM	0002
		#0015
00005	LD	T0001
00006	OUT	000205

A simpler but less flexible method of creating a flicker bit is to AND one of the Auxiliary Area clock pulse bits with the execution condition that is to be ON when the flicker bit is operating. Although this method does not use TIM, it is included here for comparison. This method is more limited because the ON and OFF times must be the same and they depend on the clock pulse bits available in the Auxiliary Area.

In the following example the 1-second clock pulse is used (A50102) so that 000206 would be turned ON and OFF every second, i.e., it would be ON for 0.5 seconds and OFF for 0.5 seconds. Precise timing and the initial status of 000206 would depend on the status of the clock pulse when 000000 goes ON.



Address	Instruction	Operands
00000	LD	000000
00001	AND	A50102
00002	OUT	000206

**5-13-2 HIGH-SPEED TIMER: TIMH(015)**

<p><b>Ladder Symbol</b></p>	<p><b>Operand Data Areas</b></p> <p><b>N:</b> Timer number #</p> <p><b>S:</b> Set value CIO, G, A, T, C, #, DM, DR, IR</p> <p>*Refer to page 128 for details on indirectly addressing timers.</p>
-----------------------------	---

**Description**

TIMH(015) operates in the same way as TIM except that TIMH measures in units of 0.01 second. Refer to 5-13-1 *TIMER: TIM* for operational details and examples.

**Differences in CVM1D and C/CVM1/CV-series PCs**

Operation of TIMH(015) is the same in CVM1D and C/CVM1/CV-series PCs except for the timer precision and PV refreshing in jumped program sections.

**1, 2, 3...**

**1. Timer Precision**

TIMH(015) instruction errors can be one cycle time or more (up to two cycle times). The error range is ± (10 ms + cycle time).



2. Operation of the TIMH(015) Instruction between JMP(004) and JME(005)
  - The timer PV is refreshed every cycle during program execution.
  - The timer operates normally while the execution condition of JMP(004) is ON.
  - The timer PV is not refreshed while the timer is jumped (while the execution condition of JMP(004) is OFF.)  
If the timer is jumped after it has started timing, the timer count will continue decrementing but the PV will not be refreshed.  
Be sure to consider the effects of the JMP(004)/JME(005) instructions on timer PVs when using the timer PVs as data in the program.
  - The Completion Flag will not be turned ON while a timer is jumped, even if the SV elapses.
  - When the execution condition for JMP(004) is changed to ON, a timer that had been operating will continue timing down from the full time that elapsed since the timer's execution condition went ON. If the SV elapsed while the timer was jumped, the timer's Completion Flag will be turned ON when the execution condition for JMP(004) is turned ON again.

**Precautions**

SV must be between 00.02 and 99.99 and must be BCD. The decimal point is not entered.

Each timer number can be used to define only one timer instruction unless the timers are never active simultaneously. The TIM, TIMH(015), and TTIM(120) timers all use timer numbers.

Timers in interlocked program sections are reset when the execution condition for IL(002) is OFF.

If the IOM Hold Bit and PC Setup are set to retain IOM (which includes timer PVs and Completion Flags), the timer must be reset before starting it to prevent possible malfunctions.

If you convert a TIM instruction to a TIMH(015) instruction (or vice versa) using online programming operations, always reset the instruction's Completion Flag. Proper operation will not be possible unless the Completion Flag is reset.

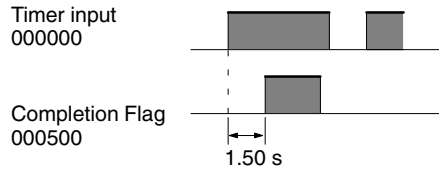
**Note:** Refer to page 101 for general precautions on operand data areas.

**Flags**

ER (A50003): Content of \*DM word is not BCD when set for BCD.  
Content of S (SV) is not BCD.

**Example**

The following timing chart illustrates the operation of the first TIMH(015) in the following example.



When CIO 000001 is ON, the SV for the second TIMH(015) in the following example will be read from CIO 0020, allowing the SV of the timer to be control from an external device connected through CIO 0020.



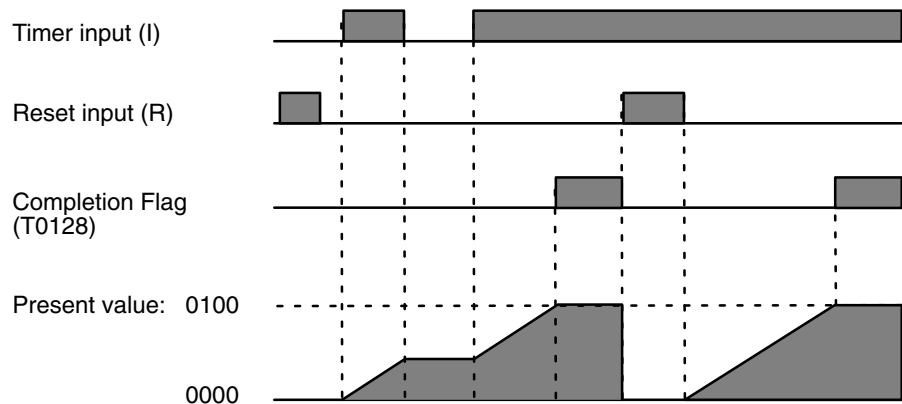
Address	Instruction	Operands
00000	LD	000000
00001	TIMH(015)	0000 #0150
00002	LD	T0000
00003	OUT	000500
00004	LD	000001
00005	TIMH(015)	0001 00020
00006	AND NOT	T0001
00007	OUT	000501

### 5-13-3 ACCUMULATIVE TIMER: TTIM(120)

<p><b>Ladder Symbol</b></p>	<p><b>Operand Data Areas</b></p> <p><b>N:</b> Timer number #</p> <p><b>S:</b> Set value CIO, G, A, T, C, #, DM, DR, IR</p> <p>*Refer to page 128 for details on indirectly addressing timers.</p>
-----------------------------	---

**Description**

A TTIM(120) timer is based on two execution conditions. These execution conditions are labeled I and R. I is the timer input; R, the reset input. The timer PV is clocked while the timer input is ON, maintained when I is OFF, and reset to zero when R is ON. If both I and R are ON simultaneously, the timer is reset. TTIM(120) increments in units of 0.1 second from zero.



**Differences in CVM1D and C/CVM1/CV-series PCs**

Operation of TTIM(120) is the same in CVM1D and C/CVM1/CV-series PCs except for the timer precision and PV refreshing in jumped program sections.

- 1, 2, 3... 1. Timer Precision  
 TTIM(120) instruction errors can be one cycle time or more (up to two cycle times). The error range is  $\pm (10 \text{ ms} + \text{cycle time})$ .
2. Operation of the TTIM(120) Instruction between JMP(004) and JME(005)
- The timer PV is refreshed every cycle during program execution.
  - The timer operates normally while the execution condition of JMP(004) is ON.
  - The timer PV is not refreshed while the timer is jumped (while the execution condition of JMP(004) is OFF.)  
 If the timer is jumped after it has started timing, the timer count will continue incrementing but the PV will not be refreshed.
  - The Completion Flag will not be turned ON while a timer is jumped, even if the SV elapses.
  - When the execution condition for JMP(004) is changed to ON, a timer that had been operating will continue timing up from the full time that elapsed since the timer's execution condition went ON. If the SV elapsed while the timer was jumped, the timer's Completion Flag will be turned ON when the execution condition for JMP(004) is turned ON again.
  - If the timer's execution condition goes ON while the timer is jumped, it may appear as if the timer is not timing because the PV is not refreshed, but the timer will be incrementing. When the execution condition for JMP(004) is changed to ON, the timer will continue incrementing from the time that elapsed while TTIM(120) was jumped.

**Precautions**

SV must be between 000.0 and 999.9 and must be BCD. The decimal point is not entered.

The PVs of accumulative timers in interlocked program sections are maintained when the execution condition for IL(002) is OFF.

Power interruptions will reset timers unless the IOM Hold Bit and PC Setup are set to retain timer PVs during power interruptions.

Accumulative timers will not restart after timing out unless the PV is changed to a value below the SV, the reset input is turned ON, or CNR(236) is executed to reset the timer. Accumulative timers are not reset in action programs when the step goes inactive.

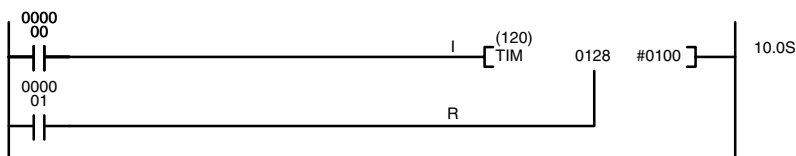
**Note:** Refer to page 101 for general precautions on operand data areas.

**Flags**

ER (A50003): Content of \*DM word is not BCD when set for BCD.  
 SV is not BCD.

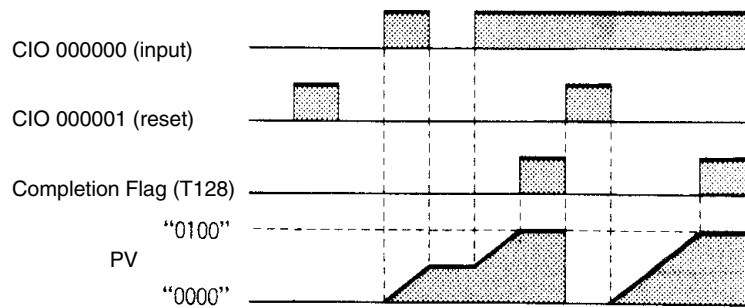
**Example**

When CIO 000000 is ON in the following example, the PV will be incremented by 1 every 0.1 s and the Completion Flag (T128) will turn ON when the PV reaches 0100. If CIO 000001 turns ON, the PV will be reset to 0000 and the Completion Flag will be turned OFF. The PV will be maintained whenever CIO 000000 is OFF.



Address	Instruction	Operands
00000	LD	000000
00001	LD	000001
00002	TTIM(120)	0128
		#0100

The following figure illustrates the relationship between the execution conditions for an accumulative timer with a set value of 10 s, its PV, and the Completion Flag.



### 5-13-4 LONG TIMER: TIML(121)

Ladder Symbol	Operand Data Areas
	<b>D<sub>1</sub>: Completion Flag</b> CIO, G, A, DM <b>D<sub>2</sub>: PV word</b> CIO, G, A, DM <b>S: SV word</b> CIO, G, A, T, C, #, DM

#### Description

TIML(121) is a decremting ON-delay timer that can time up to 9,999,999.9 s (approx. 115 days). A long timer is activated when its execution condition goes ON and is reset (to SV) when the execution condition goes OFF. Once activated, TIML(121) times down in units of 0.1 second from the SV.

Unlike most other timers, long timers are not defined using a timer number and timer numbers are not needed to access the Completion Flag and PV. The Completion Flag is bit 00 of D<sub>1</sub>, the PV is maintained in D<sub>2</sub> and D<sub>2</sub>+1, and the SV is specified in S and S+1. Bits 01 through 15 of D<sub>1</sub> cannot be used.

#### Differences in CVM1D and C/CVM1/CV-series PCs

Operation of TIML(121) is the same in CVM1D and C/CVM1/CV-series PCs except for the timer precision and PV refreshing in jumped program sections.

#### 1, 2, 3...

1. Timer Precision  
 TIML(121) instruction errors can be one cycle time or more (up to two cycle times). The error range is ± (100 ms + cycle time).
2. Operation of the TIML(121) Instruction between JMP(004) and JME(005)
  - The timer PV is refreshed every cycle during program execution.
  - The timer operates normally while the execution condition of JMP(004) is ON.
  - The timer does not operate while the timer is jumped (while the execution condition of JMP(004) is OFF), whether the timer's execution condition is ON or OFF.

#### Precautions

D<sub>2</sub> and S cannot be the last address in a data area because these operands designate the first of two words.

The set value must be BCD between 0,000,000.0 and 9,999,999.9 and must be BCD. The decimal point is not entered.

The same D<sub>1</sub>, D<sub>2</sub>, and D<sub>2</sub>+1 can be used in only one TIML(121) instruction.

Long timers in interlocked program sections are reset when the execution condition for IL(002) is OFF. Timing is disabled for long timers in jumped program sections, even if the timer's execution condition is ON.

Power interruptions will reset timers unless the IOM Hold Bit and PC Setup are set to retain timer PVs during power interruptions. If a timer that is not reset under these conditions is desired, Auxiliary Area clock pulse bits can be counted to produce accumulative timers using CNT. Refer to 5-13-6 COUNTER: CNT for details.

Long timers will not restart after timing out unless reset by turning OFF the execution condition or unless the PV is set to a value other than zero.

The long timer is inaccurate when the cycle time exceeds 25 s.

Physical limitations restrict TIML(121) accuracy to a maximum of ±0.36 s/h.

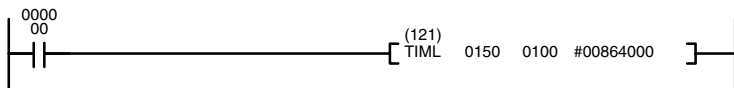
**Note:** Refer to page 101 for general precautions on operand data areas.

**Flags**

ER (A50003): Content of \*DM word is not BCD when set for BCD.  
SV is not BCD.

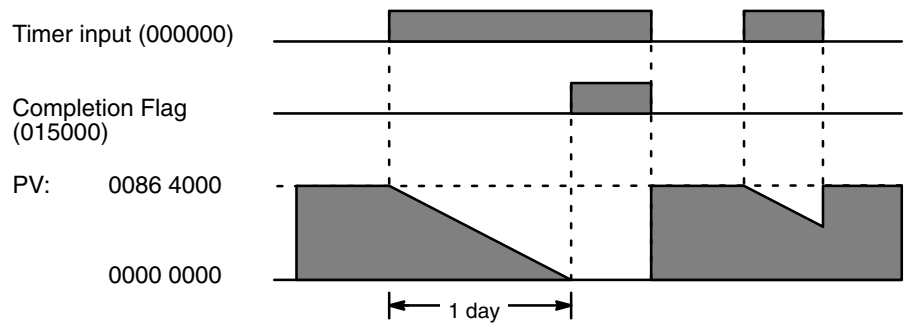
**Example**

When CIO 00000 turns ON in the following example, the SV (0086 4000) will be set into CIO 0101 and CIO 0100. As long as CIO 000000 remains ON, the PV in CIO 0101 and CIO 0100 will be decremented by -1 every 0.1 s. If the PV reaches 0000 0000, the Completion Flag (CIO 015000) will turn ON. If CIO 000000 turns OFF, the PV will again be set to the SV.



Address	Instruction	Operands
00000	LD	0000000
00001	TIML(121)	0150
		0100
		#00864000

The following figure illustrates the relationship between the execution condition for a long timer with a SV of 0086400.0 s (1 day), its PV, and the Completion Flag assigned to it.



**5-13-5 MULTI-OUTPUT TIMER: MTIM(122)**

Ladder Symbol	Operand Data Areas
	<p><b>D<sub>1</sub>: Completion Flags</b> CIO, G, A, T, C, DM</p> <p><b>D<sub>2</sub>: PV word</b> CIO, G, A, T, C, DM, DR, IR</p> <p><b>S: First SV word</b> CIO, G, A, T, C, #, DM</p>

**Description**

MTIM(122) is an accumulative timer that can have up to eight pairs of set values and Completion Flags. Unlike most timer instructions, MTIM(122) is not defined using a timer number and timer numbers are not needed to access the Completion Flags and PV. The timer is activated when the MTIM(122) execution condition is ON and the reset bit (bit 08 of D<sub>1</sub>) goes from ON to OFF.

Once activated, MTIM(122) increments the content of D<sub>2</sub> (the PV) from zero in units of 0.1 second. If the PV reaches 999.9 s, it continues counting from 000.0, and all Completion Flags are reset to zero.

Each time the instruction is executed, the PV (content of D<sub>2</sub>) is compared to the eight SVs in S through S+7, and if any of the SVs is less than or equal to the PV, the corresponding Completion Flag (D<sub>1</sub> bits 00 through 07) is turned ON. For greater accuracy, the same MTIM(122) instruction can be input into the program several times so the PV:SV comparison is made more frequently.

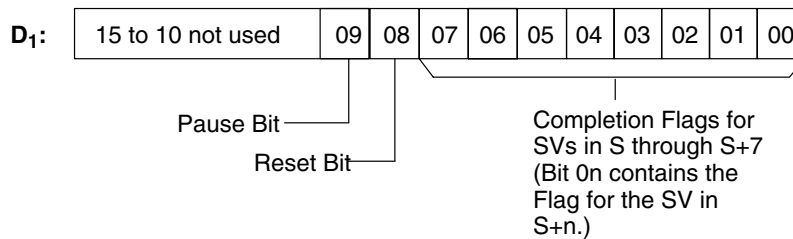
The timer can be reset by turning ON the reset bit (bit 08 of D<sub>1</sub>), which resets the PV and all Completion Flags to zero. Counting resumes from zero when the reset bit is turned OFF if the instruction execution condition is ON.

The pause bit (bit 09 of D<sub>1</sub>) can be turned ON to stop counting while maintaining the status of the PV and Completion Flags. The MTIM(122) instruction is treated as a NOP(000) instruction when the pause bit is ON and the reset bit is OFF. When the pause bit is turned OFF, counting resumes from the previous PV.

If D<sub>1</sub> is in the CIO, G, or A Area, the reset bit and pause bit can be controlled with SET(016) and RSET(017). If D<sub>1</sub> is in the DM or EM Area, these bits can be controlled with the ANDW(130) and ORW(131) instructions. The pause bit and the reset bit are effective only when the instruction execution condition is ON.

When fewer than eight outputs are needed, set the SV that follows the last one to 0000. SVs following the one set to 0000 are not compared to the PV.

The following figure shows the functions of bits in D<sub>1</sub>.



**Differences in CVM1D and C/CVM1/CV-series PCs**

Operation of MTIM(122) is the same in CVM1D and C/CVM1/CV-series PCs except for the timer precision and PV refreshing in jumped program sections.

1, 2, 3...

1. Timer Precision  
MTIM(122) instruction errors can be one cycle time or more (up to two cycle times). The error range is ± (100 ms + cycle time).
2. Operation of the MTIM(122) Instruction between JMP(004) and JME(005)
  - The timer PV is refreshed every cycle during program execution.
  - The timer operates normally while the execution condition of JMP(004) is ON.
  - The timer does not operate while the timer is jumped (while the execution condition of JMP(004) is OFF), whether the timer's execution condition is ON or OFF.

**Precautions**

S cannot be one of the last seven addresses in a data area because this operand designates the first of eight words.

The set value must be BCD between 000.0 and 999.9. The decimal point is not entered.



**Caution**

A50003 (Error Flag) will not turn ON and execution will continue even if the SV is not BCD.

The MTIM(122) PV is inaccurate when the time between instruction executions exceeds 1.6 s. The same MTIM(122) instruction can be input into the program several times so the instruction is executed more frequently, but only as many as necessary should be added to minimize the program execution time.

The set values in S through S+7 must be BCD, but the ER (A50003) Flag will **not** be turned ON, and the instruction will be executed even if the contents are not BCD.

**Note:** Refer to page 101 for general precautions on operand data areas.

**Flags**

ER (A50003): Content of \*DM word is not BCD when set for BCD.

**Example**

Timing will start in the following example when CIO 000000 is ON and CIO 000508 changes from OFF to ON. The PV will be output to D02000

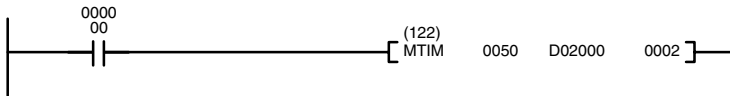
Comparisons will be made between the value in D02000 and the SV in CIO 0002 through CIO 0009 (S to S+7) and the corresponding bits in CIO 0050 will be turned on whenever the PV is greater than or equal to an SV, for example, CIO 005000 will be turned ON when the PV reaches 0155 (15.5 s).

The timer will restart from 0000 when the PV reaches 9999.

If CIO 005008 turns ON, the PV will be reset to 0000 and all Completion Flags will be turned OFF and then counting will restart when CIO 005008 turns OFF again.

If CIO 005009 turns ON, timing will stop until CIO 005009 turns OFF again.

CIO 005008 and CIO 005009 are effective only while CIO 00000 is ON.



Address	Instruction	Operands
00000	LD	000000
00001	MTIM(122)	
		0050
		D02000
		0002

	SVs	Completion Flags
S	CIO 0002 0 1 5 5	CIO 005000
S+1	CIO 0003 2 5 0 6	CIO 005001
S+2	CIO 0004 1 0 2 9	CIO 005002
S+3	CIO 0005 6 0 4 7	CIO 005003
S+4	CIO 0006 4 1 0 6	CIO 005004
S+5	CIO 0007 7 9 1 0	CIO 005005
S+6	CIO 0008 1 0 5 0	CIO 005006
S+7	CIO 0009 9 8 0 0	CIO 005007

**5-13-6 COUNTER: CNT**

<p><b>Ladder Symbol</b></p>	<p><b>Operand Data Areas</b></p> <p><b>N:</b> Counter number #</p> <p><b>S:</b> Set value CIO, G, A, T, C, #, DM, DR, IR</p> <p>*Refer to page 128 for details on indirectly addressing counters.</p>
-----------------------------	---

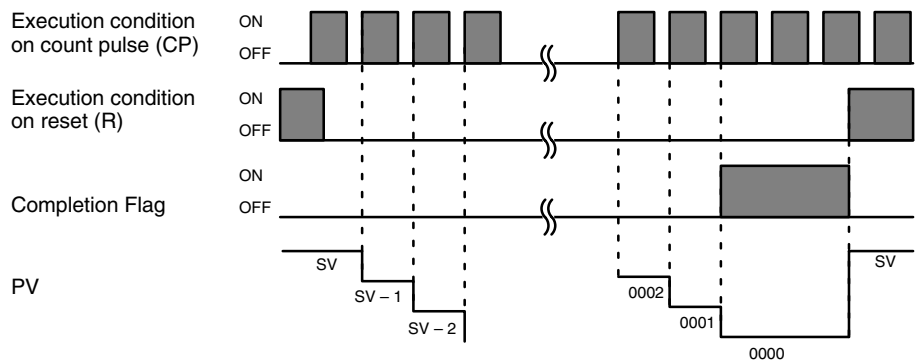
**Description**

CNT is used to count down from the SV when the execution condition on the count pulse, CP, goes from OFF to ON, i.e., the present value (PV) will be decremented by one whenever CNT is executed with an ON execution condition for CP and the execution condition was OFF for the last execution. If the execution condition has not changed or has changed from ON to OFF, the PV of CNT will not be changed. The Completion Flag for a counter is turned ON when the PV reaches zero and will remain ON until the counter is reset.

CNT is reset with a reset input, R. When R goes from OFF to ON, the PV is reset to the SV. The PV will not be decremented while R is ON. Counting down from SV will begin again when R goes OFF. The PV for CNT will not be reset in interlocked program sections or by power interruptions.

The counter will not restarted after it has counted down to zero until it is reset by turning ON R or executing CNR(236).

Changes in execution conditions, the Completion Flag, and the PV are illustrated below. PV line height is meant only to indicate changes in the PV and not absolute values.



When inputting CNT using ladder diagrams, first input the count pulse (CP), then the CNT instruction, and then the reset input (R). When using mnemonics, first input the count pulse, then the reset input, and then the CNT instruction.

**Precautions**

SV must be BCD between 0000 and 9999.

Counter numbers range from C0000 through C1023. Each counter number can be used to define only one counter instruction unless the counters are never active simultaneously. The CNT and CNTR(012) counters use counter numbers.

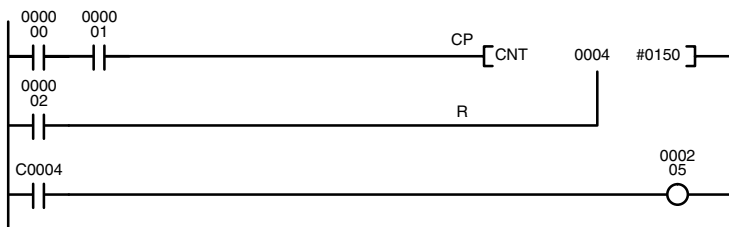
**Note:** Refer to page 101 for general precautions on operand data areas.

**Flags**

ER (A50003): Content of \*DM word is not BCD when set for BCD.  
Content of S (SV) is not BCD.

**Example 1:  
Basic Application**

In the following example, the PV will be decremented whenever both 000000 and 000001 are ON provided that 000002 is OFF and either 000000 or 000001 was OFF the last time C0004 was executed. When 150 pulses have been counted down (i.e., when PV reaches zero), 000205 will be turned ON.

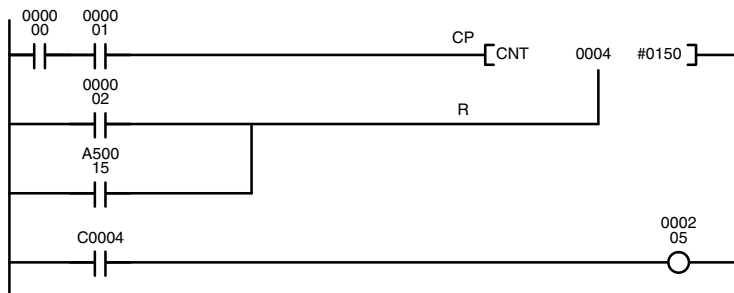


Address	Instruction	Operands
00000	LD	000000
00001	AND	000001
00002	LD	000002
00003	CNT	0004
		#0150
00004	LD	C0004
00005	OUT	000205

Here, 000000 can be used to control when CNT is operative and 000001 can be used as the bit whose OFF to ON changes are being counted.



The above CNT can be modified to restart from SV each time power is turned ON to the PC. This is done by using the First Cycle Flag in the Auxiliary Area (A50015) to reset CNT as shown below.



Address	Instruction	Operands
00000	LD	000000
00001	AND	000001
00002	LD	000002
00003	OR	A50015
00004	CNT	0004
		#0150
00005	LD	C0004
00006	OUT	000205

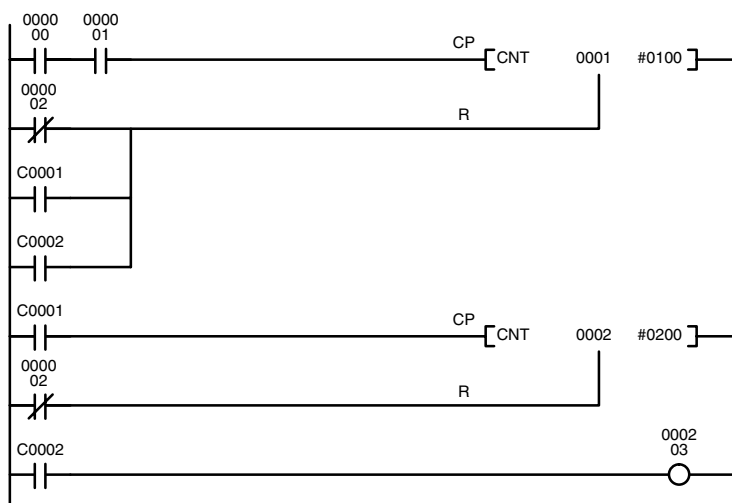
**Example 2: Extended Counter**

Counters that can count past 9,999 can be programmed by using one CNT to count the number of times another CNT has counted to zero from SV.

In the following example, 000000 is used to control when C0001 operates. When 000000 is ON, C0001 counts down the number of OFF to ON changes in 000001. C0001 is reset by its Completion Flag, i.e., it starts counting again as soon as its PV reaches zero. C0002 counts the number of times the Completion Flag for C0001 goes ON. Bit 000002 serves as a reset for the entire extended counter, resetting both C0001 and C0002 when it is OFF. The Completion Flag for C0002 is also used to reset C0001 to inhibit C0001 operation once the SV for C0002 has been reached and until the entire extended counter is reset via 000002.

Because in this example the SV for C0001 is 100 and the SV for C0002 is 200, the Completion Flag for C0002 turns ON when 100 x 200 or 20,000 OFF to ON changes have been counted in 000001. This would result in 000203 being turned ON.

CNT can be used in sequence as many times as required to produce counters capable of counting any desired values.



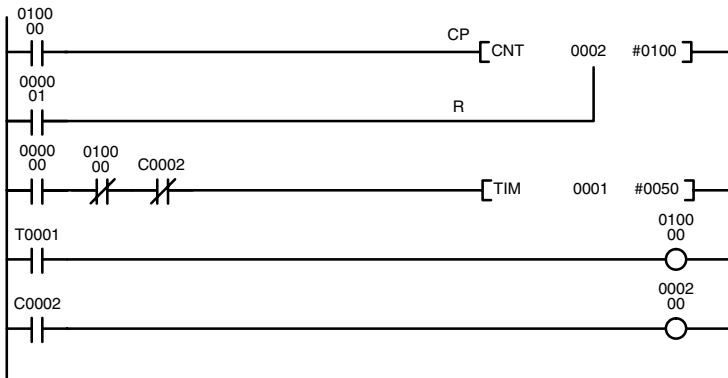
Address	Instruction	Operands
00000	LD	000000
00001	AND	000001
00002	LD NOT	000002
00003	OR	C0001
00004	OR	C0002
00005	CNT	0001
		#0100
00006	LD	C0001
00007	LD NOT	000002
00008	CNT	0002
		#0200
00009	LD	C0002
00010	OUT	000203

**Example 3: Extended Timers**

CNT can be used to create extended timers in two ways: by combining TIM with CNT and by counting Auxiliary Area clock pulse bits.

In the following example, C0002 counts the number of times T0001 reaches zero from its SV. The Completion Flag for T0001 is used to reset T0001 so that it runs continuously and C0002 counts the number of times the Completion Flag for T0001 goes ON (C0002 would be executed once each time between when the Completion Flag for T0001 goes ON and T0001 is reset by its Completion Flag). T0001 is also reset by the Completion Flag for C0002 so that the extended timer would not start again until C0002 was reset by 000001, which serves as the reset for the entire extended timer.

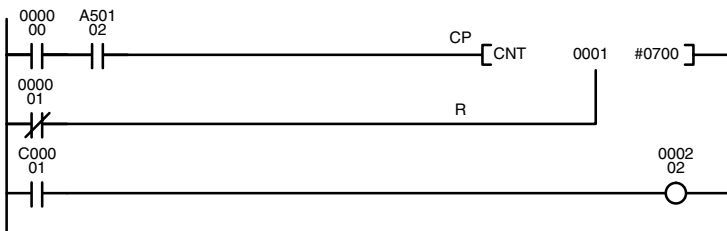
Because in this example the SV for T0001 is 5.0 seconds and the SV for C0002 is 100, the Completion Flag for C0002 turns ON when 5 seconds x 100 times, i.e., 500 seconds (or 8 minutes and 20 seconds), have expired. This would result in 000201 being turned ON.



Address	Instruction	Operands
00000	LD	010000
00001	LD	000001
00002	CNT	0002
		#0100
00003	LD	000000
00004	AND NOT	010000
00005	AND NOT	C0002
00006	TIM	0001
		#0050
00007	LD	T0001
00008	OUT	010000
00009	LD	C0002
00010	OUT	000200

In the following example, C0001 counts the number of times the 1-second clock pulse bit (A50102) goes from OFF to ON. Here again, 000000 is used to control CNT operation.

Because in this example the SV for C0001 is 700, the Completion Flag for C0002 turns ON when 1 second x 700 times, or 11 minutes and 40 seconds, have expired. This would result in 000202 being turned ON.



Address	Instruction	Operands
00000	LD	000000
00001	AND	A50102
00002	LD NOT	000001
00003	CNT	0001
		#0700
00004	LD	C0001
00005	OUT	000202

**Caution** The shorter clock pulses will not necessarily produce accurate timers because their short ON times might not be read accurately during longer cycles. In particular, the 0.02-second and 0.1-second clock pulses should not be used to create timers.

### 5-13-7 REVERSIBLE COUNTER: CNTR(012)

<p><b>Ladder Symbol</b></p>	<p><b>Operand Data Areas</b></p> <p><b>N: Counter number #</b></p> <p><b>S: Set value</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p>*Refer to page 128 for details on indirectly addressing counters.</p>
-----------------------------	---

**Description**

The CNTR(012) is a reversible, up/down circular counter, i.e., it is used to count between zero and SV according to changes in two execution conditions, those on the increment input (II) and those in the decrement input (DI).

The present value (PV) will be incremented by one whenever CNTR(012) is executed with an ON execution condition for II and the last execution condition for II was OFF. The present value (PV) will be decremented by one whenever CNTR(012) is executed with an ON execution condition for DI and the last execution condition for DI was OFF. If OFF to ON changes have occurred in both II and DI since the last execution, the PV will not be changed.

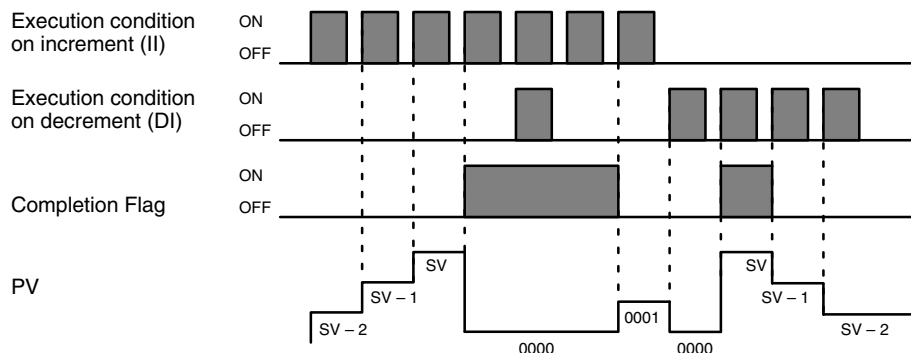
If the execution conditions have not changed or have changed from ON to OFF for both II and DI, the PV of CNTR(012) will not be changed.

When decremented below 0000, the present value is set to SV and the Completion Flag is turned ON until the PV is decremented again. When incremented above SV, the PV is set to 0000 and the Completion Flag is turned ON until the PV is incremented again.

CNTR(012) is reset with a reset input, R. When R goes from OFF to ON, the PV is reset to zero. The PV will not be incremented or decremented while R is ON. Counting will begin again when R goes OFF. The PV for CNTR(012) will not be reset in interlocked program sections or by power interruptions.

When inputting the CNTR(012) instruction with mnemonics, first enter the increment input (II), then the decrement input (DI), the reset input (R), and finally the CNTR(012) instruction. When entering with the ladder diagrams, first input the increment input (II), then the CNTR(012) instruction, the decrement input (DI), and finally the reset input (R).

Changes in II and DI execution conditions, the Completion Flag, and the PV are illustrated below starting from part way through CNTR(012) operation (i.e., when reset, counting begins from zero). PV line height is meant only to indicate changes in the PV and not absolute values.



**Precautions**

SV must be BCD between 0000 and 9999.

Counter numbers range from C0000 through C1023. Each counter number can be used to define only one counter instruction unless the counters are never active simultaneously. The CNT and CNTR(012) counters use counter numbers.

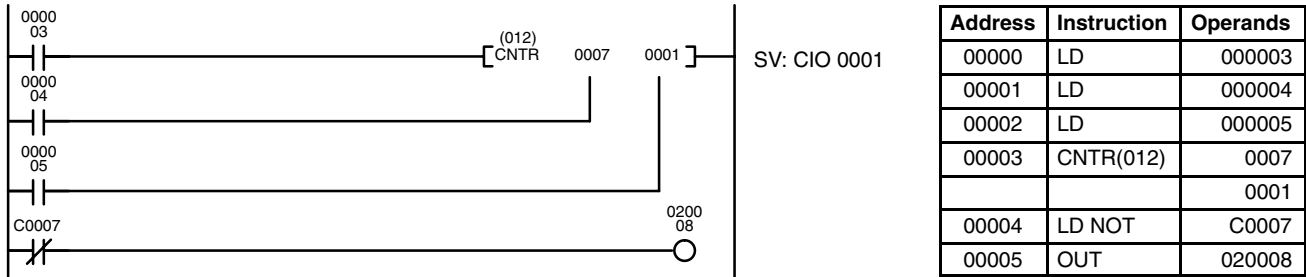
**Note:** Refer to page 101 for general precautions on operand data areas.

**Flags**

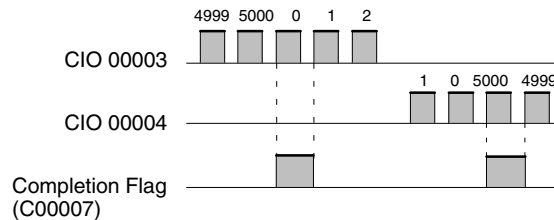
ER (A50003): Content of \*DM word is not BCD when set for BCD.  
Content of S (SV) is not BCD.

**Example**

The counter in the following example will count up when CIO 000003 changes from OFF to ON and will count down when CIO 000004 changes from OFF to ON. If CIO 000005 turns ON, the PV will be reset to 0000 and counting will be disabled until CIO 000005 turns OFF.



The following diagram illustrates the operation of the Completion Flag (C0007) when the content of CIO 0001 (i.e., the SV) is 5000.



**5-13-8 RESET TIMER/COUNTER: CNR(236)**

<p><b>Ladder Symbol</b></p> <p><b>Variations</b></p> <p>↑CNR(236)</p>	<p><b>Operand Data Areas</b></p> <p><b>D<sub>1</sub>: First Word</b> CIO, G, A, T, C, DM</p> <p><b>D<sub>2</sub>: Second Word</b> CIO, G, A, T, C, DM</p>
---	---

**Description**

When D<sub>1</sub> and D<sub>2</sub> are timer or counter numbers, CNR(236) resets the PV of timers from D<sub>1</sub> through D<sub>2</sub> without starting the timers or counters. PVs for TIM, TIMH(015), CNT, and TCNT(123) are reset to the SV, while PVs for CNTR(012) and TTIM(120) are reset to zero. TIML(121) and MTIM(122) timers cannot be reset with CNR(236).

If only one timer or counter needs to be reset, that timer or counter number can be entered alone. It is not necessary to enter both D<sub>1</sub> and D<sub>2</sub>.

If two or more timer or counter instructions are defined with the same timer or counter number and have different SVs, the PV for that timer or counter number will be reset to one or the other SV when CNR(236) is executed. The operation of the timer or counter instruction will not be affected, however, because the correct SV will be reset the next time that each timer or counter instruction is executed.

When D<sub>1</sub> and D<sub>2</sub> are addresses in a data area, CNR(236) sets the content of words D<sub>1</sub> through D<sub>2</sub> to zero.

**Precautions**

D<sub>1</sub> and D<sub>2</sub> must be in the same data area, and D<sub>1</sub> must be less than or equal to D<sub>2</sub>. If D<sub>1</sub> ≥ D<sub>2</sub>, only D<sub>1</sub> will be reset.

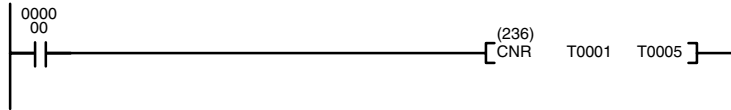
**Note:** Refer to page 101 for general precautions on operand data areas.

**Flags**

ER (A50003): Content of \*DM word is not BCD when set for BCD.

**Example**

The CNR(236) instruction in the example below resets timers T0001 to T0005 to their SV whenever CIO 000000 turns ON.



Address	Instruction	Operands
00000	LD	000000
00001	CNR(236)	
		T0001
		T0005

## 5-14 Shift Instructions

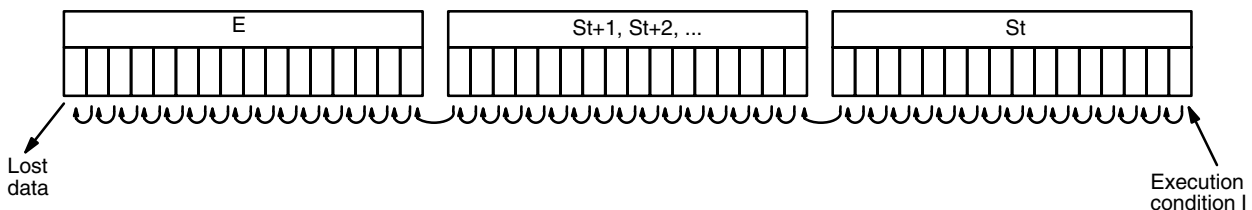
All of the Shift Instructions are used to shift data within or between words, but in differing amounts and directions.

### 5-14-1 SHIFT REGISTER: SFT(050)

Ladder Symbol	Operand Data Areas
	<b>St: Starting word</b> CIO, G, A <b>E: End word</b> CIO, G, A

#### Description

SFT(050) is controlled by three execution conditions, I, P, and R. If SFT(050) is executed and 1) execution condition P is ON and was OFF the last execution and 2) R is OFF, then execution condition I is shifted into the rightmost bit of a shift register defined between St and E, i.e., if I is ON, a 1 is shifted into the register; if I is OFF, a 0 is shifted in. When I is shifted into the register, all bits previously in the register are shifted to the left and the leftmost bit of the register is lost.



The execution condition on P functions like a differentiated instruction, i.e., I will be shifted into the register only when P is ON and was OFF the last time SFT(050) was executed. If execution condition P has not changed or has gone from ON to OFF, the shift register will remain unaffected.

St designates the rightmost word of the shift register; E designates the leftmost. The shift register includes both of these words and all words between them. The same word may be designated for St and E to create a 16-bit (i.e., 1-word) shift register.

When execution condition R goes ON, all bits in the shift register will be turned OFF (i.e., set to 0) and the shift register will not operate until R goes OFF again.

#### Precautions

St must be less than or equal to E. St and E must be in the same data area.

If a bit address in one of the words used in a shift register is also used in an instruction that controls individual bit status (e.g., OUT, KEEP(011), SET(016)), an error (“COIL DUPL”) will be generated when program syntax is checked on a Programming Device. The program, however, will be executed as written. See *Example 2: Controlling Bits in Shift Registers* for a programming example that does this.

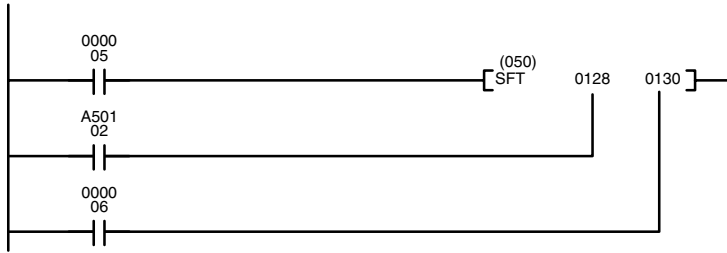
**Note** Refer to page 101 for general precautions on operand data areas.

#### Flags

There are no flags affected by SFT(050).

**Example 1:  
Basic Application**

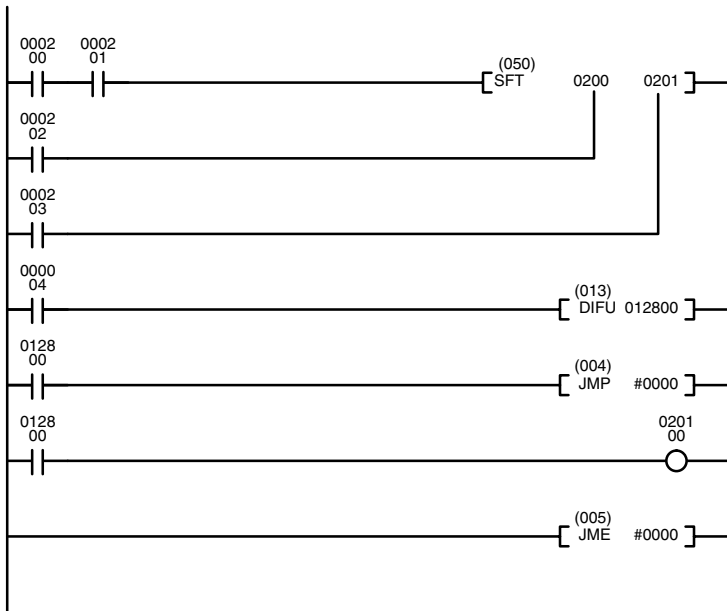
The following example uses the 1-second clock pulse bit (A50102) so that the execution condition produced by CIO 000005 is shifted into a 3-word register between CIO 0128 and CIO 0130 every second.



Address	Instruction	Operands
00000	LD	000005
00001	LD	A50102
00002	LD	000006
00003	SFT(050)	
		0128
		0130

**Example 2:  
Controlling Bits in Shift Registers**

The following program is used to control the status of the 17th bit of a shift register running from CIO 0200 through CIO 0201. When the 17th bit is to be set, CIO 000004 is turned ON. This causes the jump for JMP(004) 0000 not to be made for that one scan, and bit 020100 (the 17th bit) will be turned ON. When bit 012800 is OFF (i.e., at all times except during the first scan after CIO 000004 has changed from OFF to ON), the jump is executed and the status of bit 020100 will not be changed.



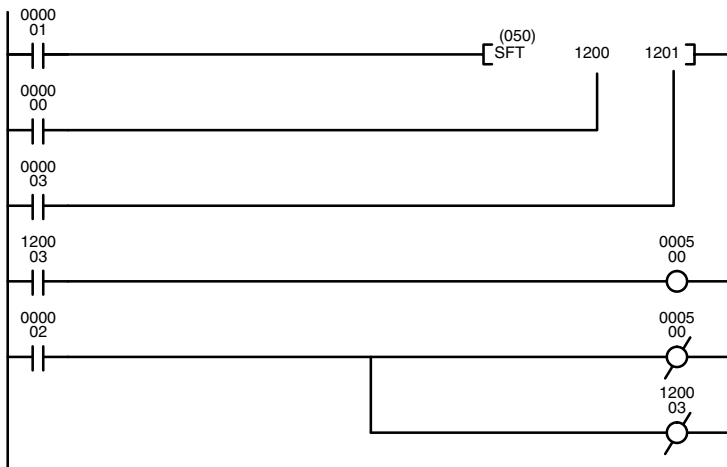
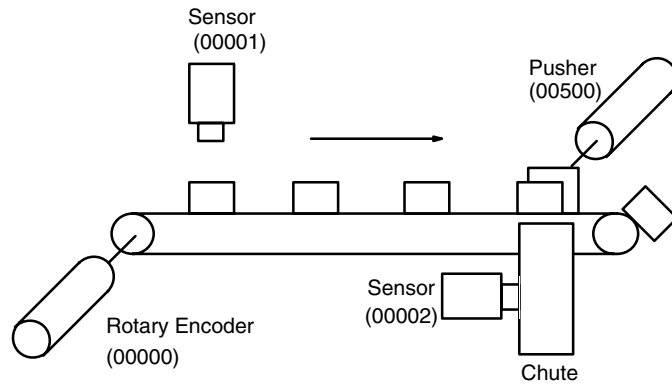
Address	Instruction	Operands
00000	LD	000200
00001	AND	000201
00002	LD	000202
00003	LD	000203
00004	SFT(050)	
		0200
		0201
00005	LD	000004
00006	DIFU(013)	012800
00007	LD	012800
00008	JMP(004)	#0000
00009	LD	012800
00010	OUT	020100
00011	JME(005)	#0000

When a bit that is part of a shift register is used in OUT (or any other instruction that controls bit status), a syntax error will be generated during the program check, but the program will be executed properly (i.e., as written).

**Example 3:  
Control Action**

The following program controls the conveyor line shown below so that faulty products detected at the sensor are pushed down a chute. To do this, the execution condition determined by inputs from the first sensor (000001) are stored in a shift register: ON for good products; OFF for faulty ones. Conveyor speed has been adjusted so that bit 120003 (in the Holding Area) of the shift register can be used to activate a pusher (000500) when a faulty product reaches it, i.e., when bit 120003 turns ON, 000500 is turned ON to activate the pusher.

The program is set up so that a rotary encoder (000000) controls execution of SFT(050) through a DIFU(013), the rotary encoder is set up to turn ON and OFF each time a product passes the first sensor. Another sensor (000002) is used to detect faulty products in the chute so that the pusher output and bit 120003 of the shift register can be reset as required.



Address	Instruction	Operands
00000	LD	000001
00001	LD	000000
00002	LD	000003
00003	SFT(050)	
		1200
		1201
00004	LD	120003
00005	OUT	000500
00006	LD	000002
00007	OUT NOT	000500
00008	OUT NOT	120003

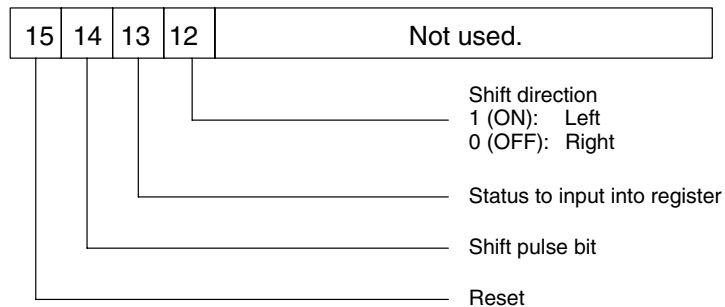


### 5-14-2 REVERSIBLE SHIFT REGISTER: SFTR(051)

Ladder Symbol	Operand Data Areas
$\text{---} \overset{(051)}{\text{[ SFTR C St E ]}}$	<b>C: Control word</b> CIO, G, A, DM, DR, IR <b>St: Starting word</b> CIO, G, A, DM <b>E: End word</b> CIO, G, A, DM
<b>Variations</b> ↑ SFTR(051)	

**Description**

SFTR(051) is used to create a single- or multiple-word shift register that can shift data to either the right or the left. To create a single-word register, designate the same word for St and E. The control word provides the shift direction, the status to be put into the register, the shift pulse, and the reset input. The control word is allocated as follows:



The data in the shift register will be shifted one bit in the direction indicated by bit 12, shifting one bit out to CY and the status of bit 13 into the other end whenever SFTR(051) is executed with an ON execution condition as long as the reset bit is OFF and as long as bit 14 is ON. If SFTR(051) is executed with an OFF execution condition or if SFTR(051) is executed with bit 14 OFF, the shift register will remain unchanged. If SFTR(051) is executed with an ON execution condition and the reset bit (bit 15) is OFF, the entire shift register and CY will be set to zero.

**Precautions**

St must be less than or equal to E, and St and E must be in the same data area.

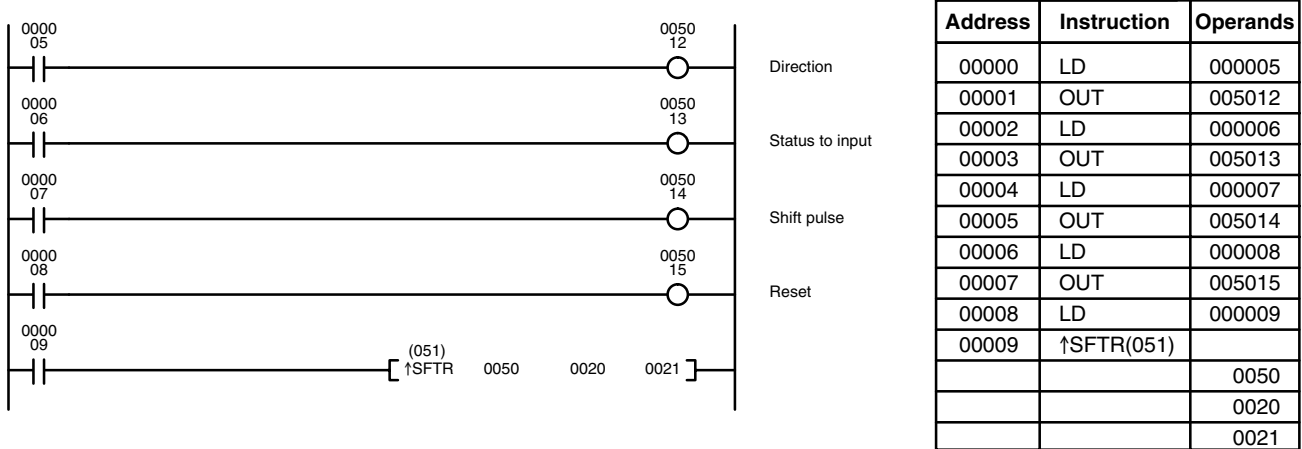
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

- ER (A50003): Content of a \*DM word is not BCD when set for BCD.  
St and E are not in the same data area or St is greater than E.
- CY (A50004): Receives the status of bit 00 of St or bit 15 of E, depending on the shift direction.

**Example**

In the following example, CIO bits 000005, 000006, 000007, and 000008 are used to control the bits of C used in  $\uparrow$  SFTR(051). The shift register is between words 0020 and 0021, and it is controlled through bit 000009.



**5-14-3 ASYNCHRONOUS SHIFT REGISTER: ASFT(052)**

Ladder Symbol	Operand Data Areas
$\text{---} \left[ \begin{matrix} (052) \\ \text{ASFT} \quad C \quad St \quad E \end{matrix} \right]$	<b>C: Control word</b> CIO, G, A, DM, DR, IR <b>St: Starting word</b> CIO, G, A, DM <b>E: End word</b> CIO, G, A, DM
<b>Variations</b> $\uparrow$ ASFT(052)	

**Description**

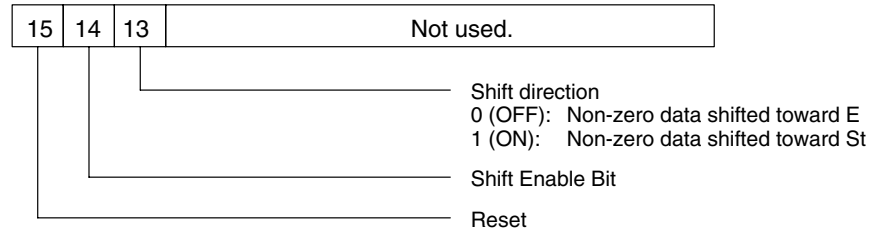
When the execution condition is OFF, ASFT(052) is not executed. When the execution condition is ON, ASFT(052) is used to create and control a reversible asynchronous word shift register between St and E. This shift register reverses the contents of adjacent words when the content of one of the words is zero and the other is non-zero.

Bit 13 of C determines whether the non-zero is shifted toward St or toward E. By repeating the instruction several times, all of the words with a content of zero accumulate at the lower or higher end of the range defined by St and E. If no words in the register contain zero or all of the words with a content of zero have accumulated at one end of the range, nothing is shifted.

When the Reset Bit is ON, the content of every word from St to E is set to zero.

**Control Word**

Bits 00 through 12 of C are not used. Bit 13 is the shift direction: turn bit 13 OFF to shift the non-zero data toward E (toward higher addressed words) and ON to shift toward St (toward lower addressed words). Bit 14 is the Shift Enable Bit: turn bit 14 OFF to enable shift register operation according to bit 13, and ON to disable the register. Bit 15 is the Reset Bit: the register will be reset (set to zero) between St and E when ASFT(052) is executed with bit 15 OFF. Turn bit 15 ON for normal operation.



Content of C	Function of ASFT(052)
#4000	Shift non-zero data toward E
#6000	Shift non-zero data toward St
#8000	Reset all words to zero

**Precautions**

St must be less than or equal to E. St and E must be in the same data area.

**Note** Refer to page 101 for general precautions on operand data areas.

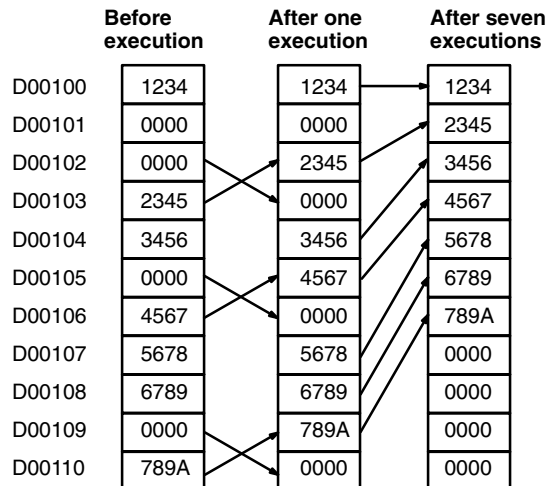
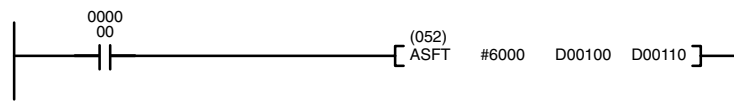
**Flags**

ER (A50003): Content of \*DM word is not BCD when set for BCD.

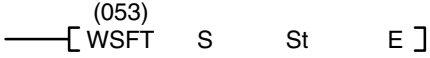
The St and E words are in different areas, or St is greater than E.

**Example**

The following instruction is used to shift words in an 11-word shift register created between D 00100 and D 00110 with C=#6000. Non-zero data is shifted towards St (D00110).

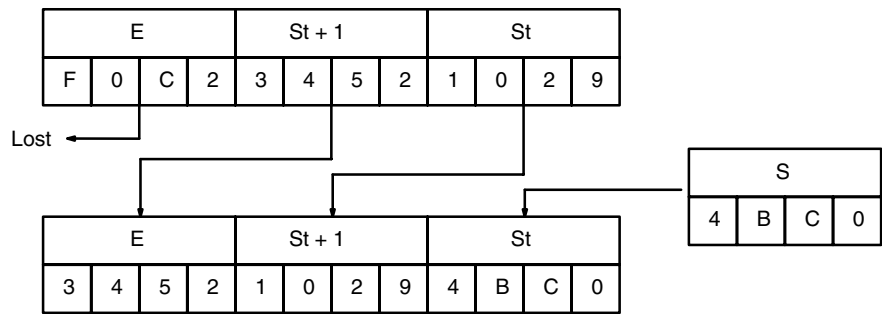


### 5-14-4 WORD SHIFT: WSFT(053)

<p><b>Ladder Symbol</b></p> <p>(053)  </p> <p><b>Variations</b></p> <p>↑ WSFT(053)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: Source word</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>St: Starting word</b> CIO, G, A, DM</p> <p><b>E: End word</b> CIO, G, A, DM</p>
---	--

**Description**

When the execution condition is OFF, WSFT(053) is not executed. When the execution condition is ON, WSFT(053) shifts data between St and E in word units. The data in S is copied into St and the content of E is lost.



**Precautions**

St must be less than or equal to E. St and E must be in the same data area. The shift operation might not be completed if a power interruption occurs during execution of the instruction.

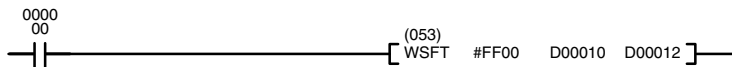
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

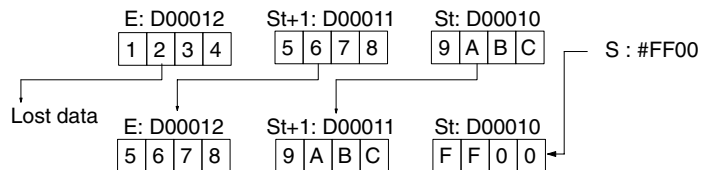
ER (A50003): Content of \*DM word is not BCD when set for BCD. The St and E words are in different areas, or St is greater than E.

**Example**

When CIO 000000 is ON in the following example, FF00 is shifted into D00010, the word data in D00010 is shifted to D00011, the word data in D00011 is shifted to D00012, and the word data in D00012 is lost.



Address	Instruction	Operands
00000	LD	000000
00001	WSFT(053)	
		#FF00
		D00010
		D00012



### 5-14-5 SHIFT N-BIT DATA LEFT: NSFL(054)

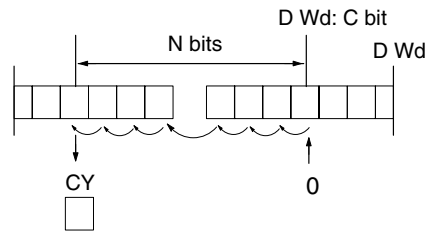
Ladder Symbol	Operand Data Areas
	<p><b>D: Beginning word for shift</b> CIO, G, A</p> <p><b>C: Beginning bit</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>N: Shift data length</b> CIO, G, A, T, C, #, DM, DR, IR</p>
<p><b>Variations</b></p> <p>↑NSFL(054)</p>	

**Description**

When the execution condition is OFF, NSFL(054) is not executed. When the execution condition is ON, NSFL(054) shifts the specified number of bits (i.e., the shift data length) from the beginning bit in the beginning word (bit C of word D), one bit to the left. A “0” is shifted into the beginning bit. The status of the Nth bit is shifted to CY (A50004).

If the shift data length (D) is “0,” the beginning bit (C) data will be copied to CY and the status of the beginning bit (C) will not be changed.

Set the beginning bit to a value from 0000 to 0015 BCD.



**Precautions**

C must be between 0000 and 0015 and must be BCD.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

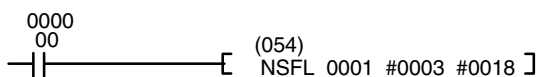
ER (A50003): C is not 0000 to 0015 BCD.

Content of a \*DM word is not BCD when set for BCD.

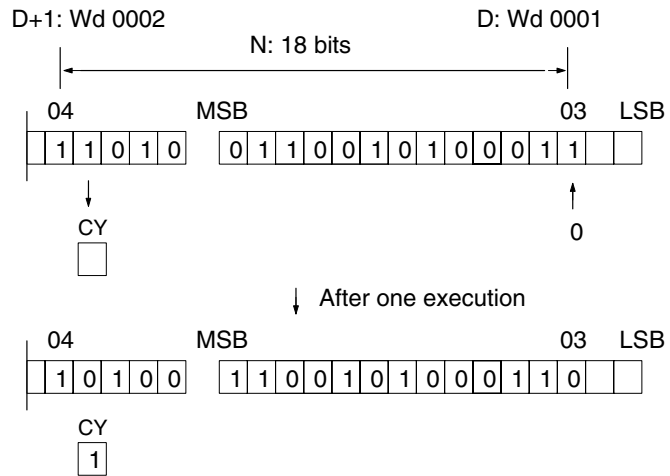
CY (A50004): “1” has been shifted to CY.

**Example**

When the CIO 000000 is ON in the following example, the 18 bits of data starting from bit 03 of CIO 0001 are shifted to the left one bit at a time. A “0” is entered for the beginning bit (CIO 000103) of the shift. The status of bit CIO 000204 is shifted to CY.



Address	Instruction	Operands
00000	LD	000000
00001	NSFL(054)	
		0001
		#0003
		#0018



### 5-14-6 SHIFT N-BIT DATA RIGHT: NSFR(055)

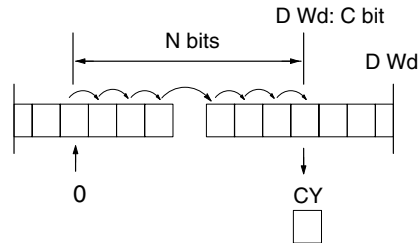
Ladder Symbol	Operand Data Areas
$\text{---} \overset{(055)}{[ \text{NSFR } D \text{ C } N \ ]}$	<b>D: Beginning word for shift</b> CIO, G, A <b>C: Beginning bit</b> CIO, G, A, T, C, #, DM, DR, IR <b>N: Shift data length</b> CIO, G, A, T, C, #, DM, DR, IR
<b>Variations</b> ↑NSFR(055)	

**Description**

When the execution condition is OFF, NSFR(055) is not executed. When the execution condition is ON, NSFR(055) shifts the specified number of bits (i.e., the shift data length) from the beginning bit of the beginning word (bit C of word D), one bit to the right. A "0" is entered for the beginning bit. The status of the Nth bit is shifted to CY (A50004).

If the shift data length (D) is "0," the beginning bit (C) data will be copied to C and the status of the beginning bit (C) will not be changed.

Set the beginning bit to a value from 0000 to 0015 BCD.



**Precautions**

C must be between 0000 and 0015 and must be BCD.

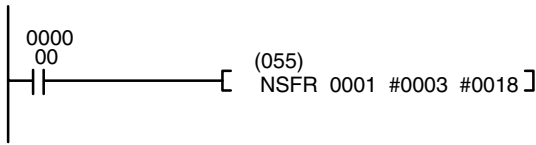
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

- ER (A50003): C is not 0000 to 0015 BCD.  
Content of a\*DM word is not BCD when set for BCD.
- CY (A50004): "1" has been shifted to CY.

**Example**

When CIO 000000 is ON in the following example, the 18 bits of data beginning from bit 03 of CIO 0001 are shifted to the right, one at a time. A “0” is entered for the beginning bit (CIO 000204) of the shift. The status of bit CIO 000103 is shifted to CY.



Address	Instruction	Operands
00000	LD	000000
00001	NSFR(055)	
		0001
		#0003
		#0018



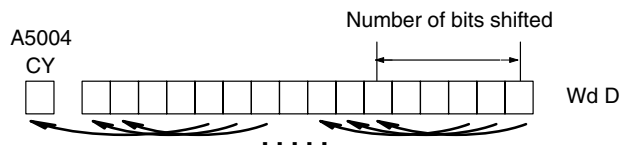
**5-14-7 SHIFT N-BITS LEFT: NASL(056)**

<p><b>Ladder Symbol (NASL)</b></p> <p><b>Variations</b></p> <p>↑NASL(056)</p>	<p><b>Operand Data Areas</b></p> <p><b>D: Shift word</b>      CIO, G, A, DM, DR, IR</p> <p><b>C: Control word</b>    CIO, G, A, T, C, #, DM, DR, IR</p>
---	---

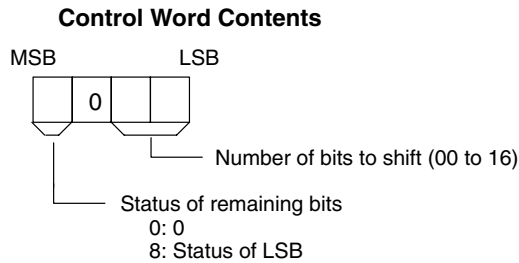
**Description**

When the execution condition is OFF, NASL(056) is not executed. When the execution condition is ON, NASL(056) shifts the status of the 16 bits in the specified word to the left the specified number of bits.

The number of bits to shift is set in the two rightmost digits of the control word. If the number is “0,” the data will not be shifted. The appropriate flags (see below) will turn ON and OFF, however, according to data in the specified word.



After the bits have been shifted, the status of the bits from which data was shifted (i.e., the number of bits shifted, beginning with the rightmost bit of the specified word) will be set to “0” or to the status of the LSB, depending on the control word setting.



**Precautions**

Refer to page 101 for general precautions on operand data areas.

**Flags**

- ER (A50003): Number of bits to shift is out of range.  
Content of a\*DM word is not BCD when set for BCD.
- CY (A50004): “1” has been shifted to CY.
- EQ (A50006) Content of D after the shift is all zeros
- N (A50008) Same status as leftmost bit (MSB) of word D after shift.

**Example**

See the example provided in 5-14-9 DOUBLE SHIFT N-BITS LEFT: NSLL(058).

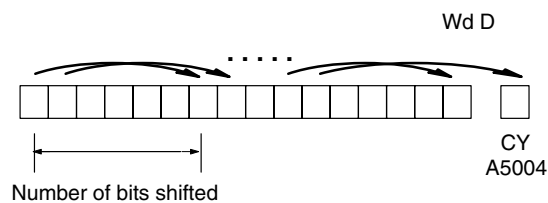
**5-14-8 SHIFT N-BITS RIGHT: NASR(057)**

Ladder Symbol	Operand Data Areas	
<p style="text-align: center;"> <math>\xrightarrow{\quad\quad\quad} \overset{(057)}{[ \text{NASR D C } ]}</math> </p> <p><b>Variations</b></p> <p>↑NASR(057)</p>	<p><b>D: Shift word</b></p>	<p>CIO, G, A, DM, DR, IR</p>
	<p><b>C: Control word</b></p>	<p>CIO, G, A, T, C, #, DM, DR, IR</p>

**Description**

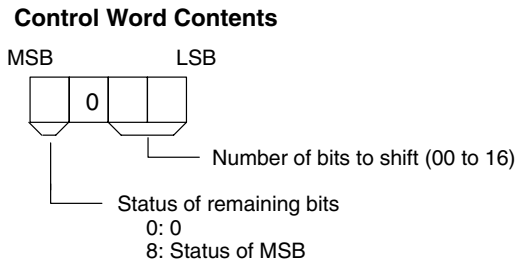
When the execution condition is OFF, NASR(057) is not executed. When the execution condition is ON, NASR(057) shifts the status of the 16 bits in the specified word to the right the specified number of bits

The number of bits to shift is set in the two rightmost digits of the control word. If the number is “0,” the data will not be shifted. The appropriate flags will turn ON and OFF, however, according to data in the specified word.





After the bits have been shifted, the status of the bits from which data was shifted (i.e., the number of bits shifted, beginning with the leftmost bit of the specified word) will be set to “0” or to the status of the MSB, depending on the control word setting.



**Precautions**

Refer to page 101 for general precautions on operand data areas.

**Flags**

- ER (A50003): Number of bits to shift is out of range.  
Content of a\*DM word is not BCD when set for BCD.
- CY (A50004): “1” has been shifted to CY.
- EQ (A50006) Content of word D after the shift is all zeros.
- N (A50008) Same status as leftmost bit (MSB) of word D after shift.

**Example**

See the example provided in 5-14-10 *DOUBLE SHIFT N-BITS RIGHT: NSRL(059)*.

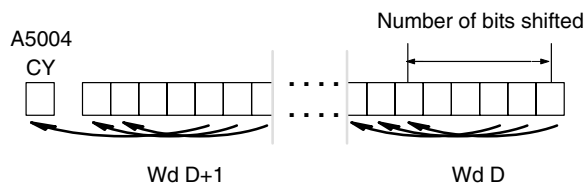
**5-14-9 DOUBLE SHIFT N-BITS LEFT: NSLL(058)**

Ladder Symbol	Operand Data Areas
$\text{---} \left[ \overset{(058)}{\text{NSLL D C}} \right]$	<b>D: Beginning shift word</b> CIO, G, A, DM <b>C: Control word</b> CIO, G, A, T, C, #, DM
<b>Variations</b> ↑NSLL(058)	

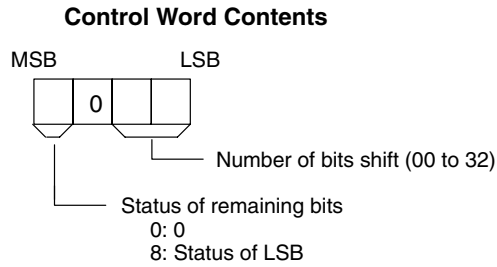
**Description**

When the execution condition is OFF, NSLL(058) is not executed. When the execution condition is ON, NSLL(058) shifts the status of the 32 bits in the specified words to the left the specified number of bits.

The number of bits to shift is set in the two rightmost digits of the control word. If the number is “0,” the word data will not be shifted. The appropriate flags will turn ON and OFF, however, according to data in the specified word.



After the bits have been shifted, the status of the bits from which data was shifted (i.e., the number of bits shifted, beginning with the rightmost bit of the specified word) will be set to “0” or to the status of the LSB, depending on the control word setting.



**Precautions**

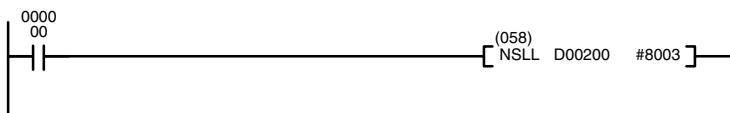
Refer to page 101 for general precautions on operand data areas.

**Flags**

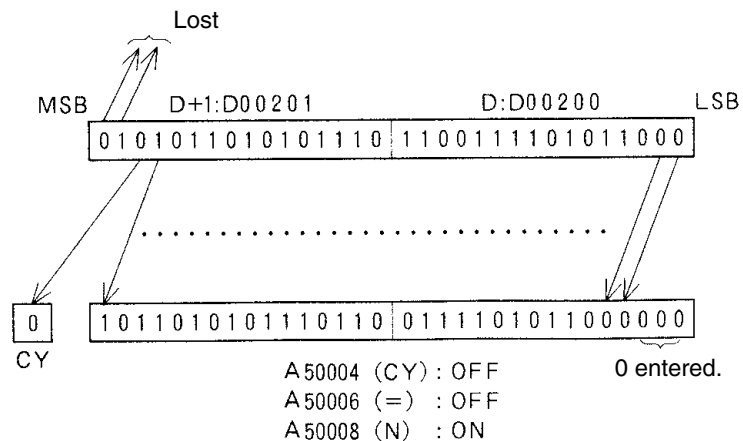
- ER (A50003): Number of bits to shift is out of range.  
Content of a\*DM word is not BCD when set for BCD.
- CY (A50004): “1” has been shifted to CY.
- EQ (A50006) Content of words D and D+1 after the shift is all zeros.
- N (A50008) Same status as leftmost bit (MSB) of word D+1 after shift.

**Example**

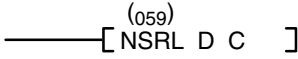
When CIO 000000 is ON in the following example, the 32 bits in D00201 and D00200 are shifted three bits to the left. The status of the three rightmost bits of D00200 are set to “0.”



Address	Instruction	Operands
00000	LD	000000
00001	NSLL(058)	
		D00200
		#8003



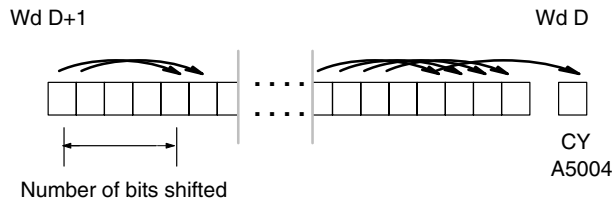
### 5-14-10 DOUBLE SHIFT N-BITS RIGHT: NSRL(059)

<p><b>Ladder Symbol</b></p>  <p><b>Variations</b></p> <p>↑NSRL(059)</p>	<p><b>Operand Data Areas</b></p> <p><b>D: Shift word address</b> CIO, G, A, DM,  <b>C: Control word</b> CIO, G, A, T, C, #, DM,</p>
--	---

**Description**

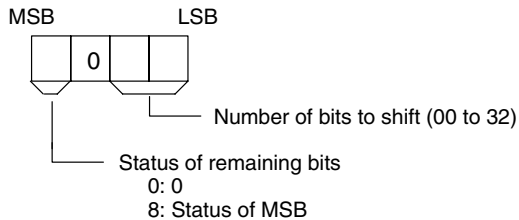
When the execution condition is OFF, NSRL(059) is not executed. When the execution condition is ON, NSRL(059) shifts the status of the 32 bits in the specified words to the right the specified number of bits.

The number of bits to shift is set in the two rightmost digits of the control word. If the number is "0," the word data will not be shifted. The appropriate flags will turn ON and OFF, however, according to data in the specified word.



After the bits have been shifted, the status of the bits from which data was shifted (i.e., the number of bits shifted, beginning with the leftmost bit of the specified word) will be set to "0" or to the status of the MSB, depending on the control word setting.

**Control Word Contents**



**Precautions**

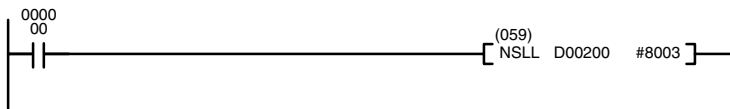
Refer to page 101 for general precautions on operand data areas.

**Flags**

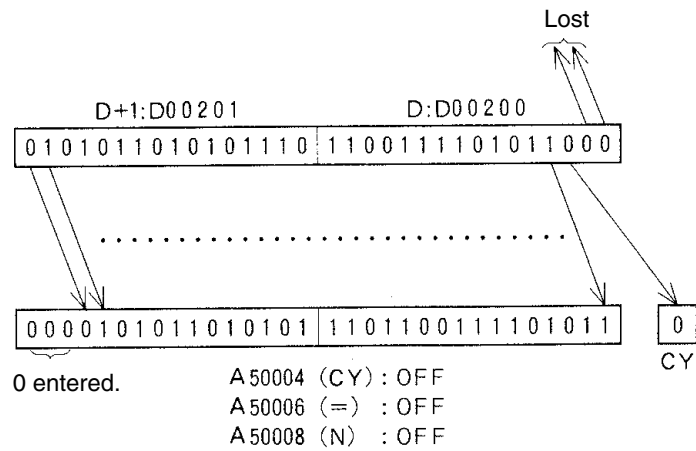
- ER (A50003): Number of bits to shift is out of range.  
Content of a\*DM word is not BCD when set for BCD.
- CY (A50004): "1" has been shifted to CY.
- EQ (A50006) Content of word D and D+1 after the shift is all zeros.
- N (A50008) Same as leftmost bit (MSB) of word D+1 after shift.

**Example**

When CIO 00000 is ON in the following example, the 32 bits in D00200 and D00201 are shifted three bits to the right. The status of the three leftmost bits of D00201 are set to "0."



Address	Instruction	Operands
00000	LD	000000
00001	NSLL(059)	
		D00200
		#8003

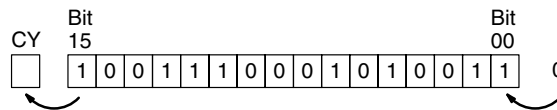


### 5-14-11 ARITHMETIC SHIFT LEFT: ASL(060)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;"> </p> <p><b>Variations</b></p> <p>↑ ASL(060)</p>	<p><b>Operand Data Area</b></p> <p><b>Wd: Word</b>      CIO, G, A, DM,</p>
---	--

**Description**

When the execution condition is OFF, ASL(060) is not executed. When the execution condition is ON, ASL(060) shifts a 0 into bit 00 of Wd, shifts the bits of Wd one bit to the left, and shifts the status of bit 15 into CY.

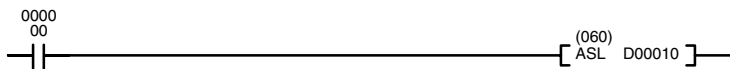


**Flags**

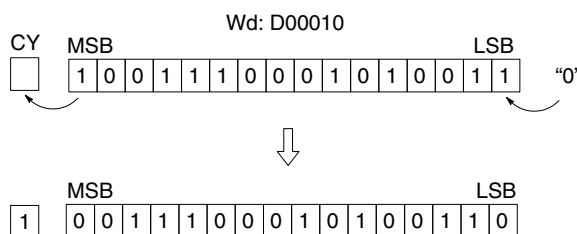
- ER (A50003): Content of \*DM word is not BCD when set for BCD.
- CY (A50004): Receives the status of bit 15.
- EQ (A50006): Content of Wd is 0 after a shift.
- N (A50008): Same status as bit 15 of D + 1 after shift.

**Example**

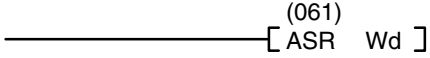
When CIO 000000 is ON in the following example, 0 is shifted into bit 00 of D00010, the status of all bits within D00010 are shifted left one position, and the status of bit 15 is shifted to CY.



Address	Instruction	Operands
00000	LD	000000
00001	ASL(060)	
		D00010

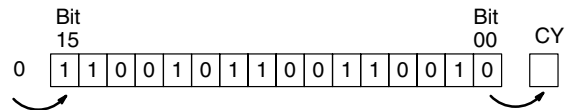


### 5-14-12 ARITHMETIC SHIFT RIGHT: ASR(061)

<p><b>Ladder Symbol</b></p>  <p><b>Variations</b></p> <p>↑ ASR(061)</p>	<p><b>Operand Data Area</b></p> <p><b>Wd: Word</b>      CIO, G, A, DM, DR, IR</p>
--	---

**Description**

When the execution condition is OFF, ASR(061) is not executed. When the execution condition is ON, ASR(061) shifts a 0 into bit 15 of Wd, shifts the bits of Wd one bit to the right, and shifts the status of bit 00 into CY.

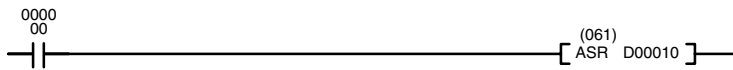


**Flags**

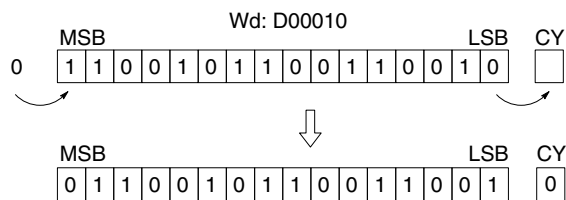
- ER (A50003): Content of \*DM word is not BCD when set for BCD.
- CY (A50004): Receives the status of bit 00.
- EQ (A50006): Content of Wd is 0 after a shift.
- N (A50008): OFF.

**Example**

When CIO 000000 is ON in the following example, 0 is shifted into bit 15 of D00010, the status of all bits within D00010 are shifted right one position, and the status of bit 00 is shifted to CY.



Address	Instruction	Operands
00000	LD	000000
00001	ASR(061)	
		D00010

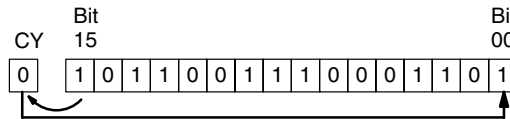


### 5-14-13 ROTATE LEFT: ROL(062)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;"> </p> <p><b>Variations</b></p> <p>↑ ROL(062)</p>	<p><b>Operand Data Area</b></p> <p><b>Wd: Word</b>                      CIO, G, A, DM, DR, IR</p>
---	---

**Description**

When the execution condition is OFF, ROL(062) is not executed. When the execution condition is ON, ROL(062) shifts all Wd bits one bit to the left, shifting CY into bit 00 of Wd and shifting bit 15 of Wd into CY.



**Precautions**

Use STC(078) to set CY to 1 or CLC(079) to set CY to 0 if necessary before doing a rotate operation to ensure that CY contains the proper status before executing ROL(062).

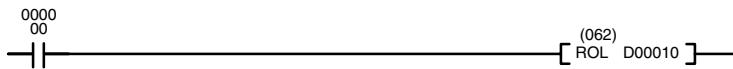
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

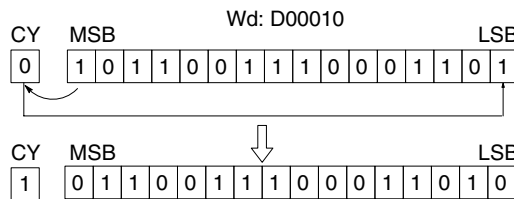
- ER (A50003): Content of \*DM word is not BCD when set for BCD.
- CY (A50004): Receives the status of bit 15 from Wd.
- EQ (A50006): Content of Wd is 0 after execution.
- N (A50008): Same status as bit 15 of Wd after execution.

**Example**

When CIO 000000 is ON in the following example, the status of CY is shifted into bit 00 of D00010, the status of all bits within D00010 are shifted left one position, and the status of bit 15 is shifted to CY.



Address	Instruction	Operands
00000	LD	000000
00001	ROL(062)	
		D00010

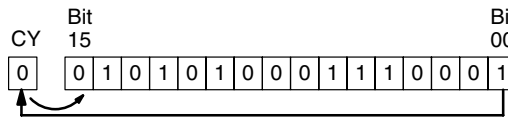


### 5-14-14 ROTATE RIGHT: ROR(063)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;"> </p> <p><b>Variations</b></p> <p>↑ ROR(063)</p>	<p><b>Operand Data Area</b></p> <p><b>Wd: Word</b>                      CIO, G, A, DM, DR, IR</p>
---	---

**Description**

When the execution condition is OFF, ROR(063) is not executed. When the execution condition is ON, ROR(063) shifts all Wd bits one bit to the right, shifting CY into bit 15 of Wd and shifting bit 00 of Wd into CY.



**Precautions**

Use STC(078) to set CY to 1 or CLC(079) to set CY to 0 if necessary before doing a rotate operation to ensure that CY contains the proper status before executing ROR(063).

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

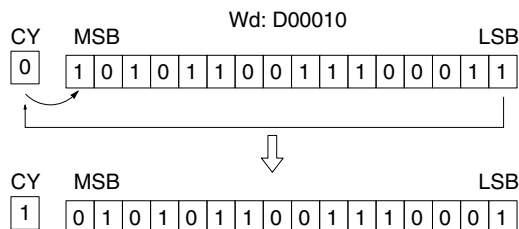
- ER (A50003): Content of \*DM word is not BCD when set for BCD.
- CY (A50004): Receives the status of bit 15 from Wd.
- EQ (A50006): Content of Wd is 0 after execution.
- N (A50008): Same status as bit 15 of Wd after execution.

**Example**

When CIO 000000 is ON in the following example, the status of CY is shifted into bit 15 of D00010, the status of all bits within D00010 are shifted right one position, and the status of bit 00 is shifted to CY.



Address	Instruction	Operands
00000	LD	000000
00001	ROR(063)	
		D00010

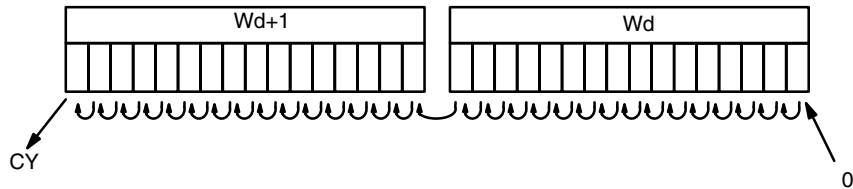


### 5-14-15 DOUBLE SHIFT LEFT: ASLL(064)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;"> </p> <p><b>Variations</b></p> <p>↑ ASLL(064)</p>	<p style="text-align: center;"><b>Operand Data Area</b></p> <p><b>Wd: Word</b>                      CIO, G, A, DM</p>
--	---

**Description**

When the execution condition is OFF, ASLL(064) is not executed. When the execution condition is ON, ASLL(064) shifts a 0 into bit 00 of Wd, all bits previously in Wd and Wd+1 are shifted to the left, and bit 15 of Wd+1 is shifted into CY.



**Precautions**

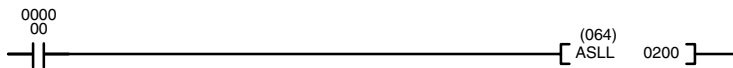
Refer to page 101 for general precautions on operand data areas.

**Flags**

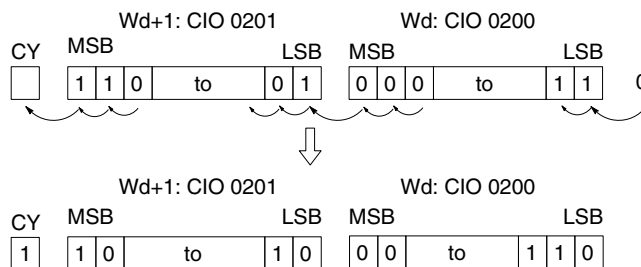
- ER (A50003): Content of \*DM word is not BCD when set for BCD.
- CY (A50004): Receives the status of bit 15 from Wd+1.
- EQ (A50006): Content of Wd and Wd+1 are 0 after a shift.
- N (A50008): Same status as bit 15 of Wd+1 after shift.

**Example**

When CIO 000000 is ON in the following example, 0 is shifted into bit 00 of CIO 0200, the status of all bits within CIO 0200 are shifted left one position, the status of bit 15 is shifted to bit 00 of CIO 0201, the status of all bits within CIO 0201 are shifted left one position, and the status of bit 15 is shifted to CY.



Address	Instruction	Operands
00000	LD	000000
00001	ASLL(064)	
		0200



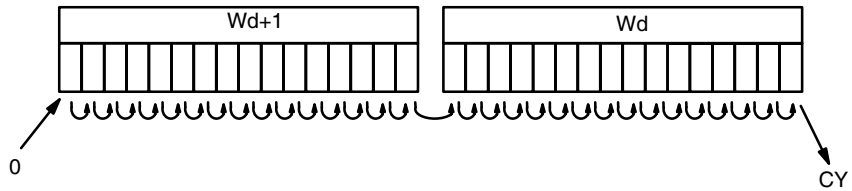


### 5-14-16 DOUBLE SHIFT RIGHT: ASRL(065)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;"> </p> <p><b>Variations</b></p> <p>↑ ASRL(065)</p>	<p><b>Operand Data Area</b></p> <p><b>Wd: Word</b>      CIO, G, A, DM</p>
--	---

**Description**

When the execution condition is OFF, ASRL(065) is not executed. When the execution condition is ON, ASRL(065) shifts a 0 into bit 15 of Wd+1, all bits previously in Wd and Wd+1 are shifted to the right, and bit 00 of Wd is shifted into CY.



**Precautions**

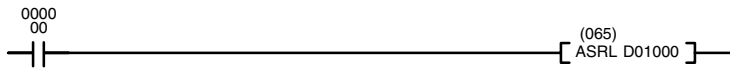
Refer to page 101 for general precautions on operand data areas.

**Flags**

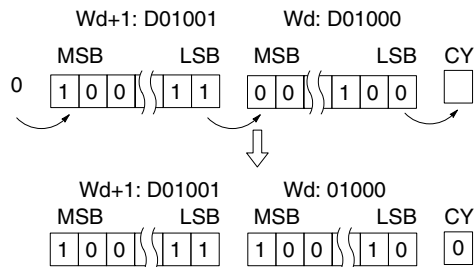
- ER (A50003): Content of \*DM word is not BCD when set for BCD.
- CY (A50004): Receives the status of bit 00 from Wd.
- EQ (A50006): Content of Wd and Wd+1 are 0 after a shift.
- N (A50008): Same status as bit 15 of Wd+1 after shift.

**Example**

When CIO 000000 is ON in the following example, 0 is shifted into bit 15 of D01001, the status of all bits within D01001 are shifted right one position, the status of bit 00 of D01001 is shifted to bit 15 of D01000, the status of all bits within D01000 are shifted right one position, and the status of bit 00 is shifted to CY.



Address	Instruction	Operands
00000	LD	000000
00001	ASRL(065)	
		D01000

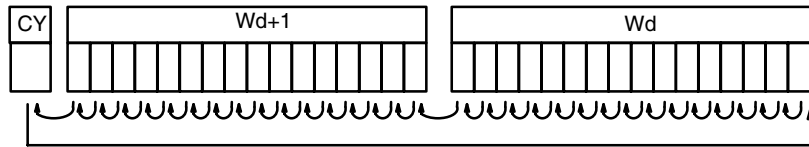


### 5-14-17 DOUBLE ROTATE LEFT: ROLL(066)

<p><b>Ladder Symbol</b></p> <p><b>Variations</b></p> <p>↑ ROLL(066)</p>	<p><b>Operand Data Area</b></p> <p><b>Wd: Word</b>      CIO, G, A, DM</p>
---	---

**Description**

When the execution condition is OFF, ROLL(066) is not executed. When the execution condition is ON, ROLL(066) shifts CY into bit 00 of Wd, all bits previously in Wd and Wd+1 are shifted to the left, and bit 15 of Wd+1 is shifted into CY.



**Precautions**

Use STC(078) to set CY to 1 or CLC(079) to set CY to 0 if necessary before doing a rotate operation to ensure that CY contains the proper status before executing ROLL(066).

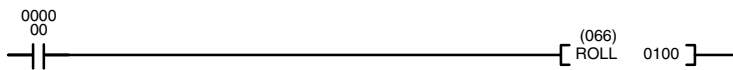
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

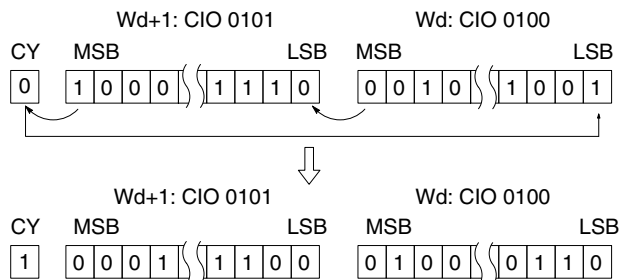
- ER (A50003): Content of \*DM word is not BCD when set for BCD.
- CY (A50004): Receives the status of bit 15 from Wd+1.
- EQ (A50006): Content of Wd and Wd+1 are 0 after execution.
- N (A50008): Same status as bit 15 of Wd+1 after execution.

**Example**

When CIO 000000 is ON in the following example, the status of CY is shifted into bit 00 of CIO 0100, the status of all bits within CIO 0100 are shifted left one position, the status of bit 15 of CIO 0100 is shifted to bit 00 of CIO 0101, the status of all bits within CIO 0101 are shifted left one position, and the status of bit 15 is shifted to CY.



Address	Instruction	Operands
00000	LD	000000
00001	ROLL(066)	
		0100

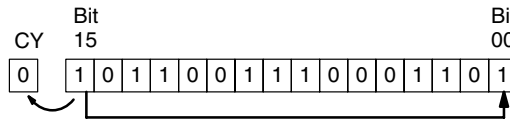


### 5-14-18 ROTATE LEFT WITHOUT CARRY: RLNC(260)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;"> </p> <p><b>Variations</b></p> <p>↑RNLC(260)</p>	<p><b>Operand Data Area</b></p> <p><b>Wd: Word</b>                      CIO, G, A, DM, DR, IR</p>
---	---

**Description**

When the execution condition is OFF, RLNC(260) is not executed. When the execution condition is ON, RLNC(260) shifts all Wd bits one bit to the left, shifting the status of bit 15 of Wd into both bit 00 and CY.

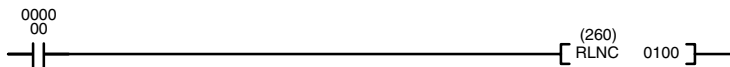


**Flags**

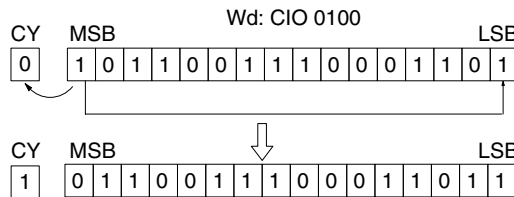
- ER (A50003): Content of \*DM word is not BCD when set for BCD.
- CY (A50004): Receives the status of bit 15 from Wd.
- EQ (A50006): Content of Wd is 0 after execution.
- N (A50008): Same status as bit 15 of Wd after execution.

**Example**

When CIO 000000 is ON in the following example, the status of bit 15 of CIO 0100 is shifted into bit 00 of CIO 0100 and into CR and the status of all bits within CIO 0100 are shifted left one position.



Address	Instruction	Operands
00000	LD	000000
00001	RLNC(260)	
		0100

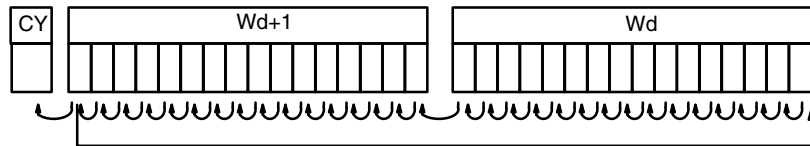


### 5-14-19 DOUBLE ROTATE LEFT WITHOUT CARRY: RLNL(262)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;"> </p> <p><b>Variations</b></p> <p>↑RLNL(262)</p>	<p><b>Operand Data Area</b></p> <p><b>Wd: Word</b>                      CIO, G, A, DM</p>
---	---

**Description**

When the execution condition is OFF, RLNL(262) is not executed. When the execution condition is ON, RLNL(262) shifts all bits previously in Wd and Wd+1 to the left, and bit 15 of Wd+1 is shifted into both bit 00 of Wd and into CY.



**Precautions**

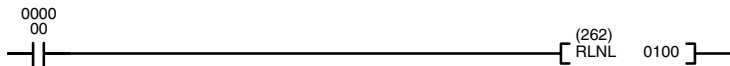
Refer to page 101 for general precautions on operand data areas.

**Flags**

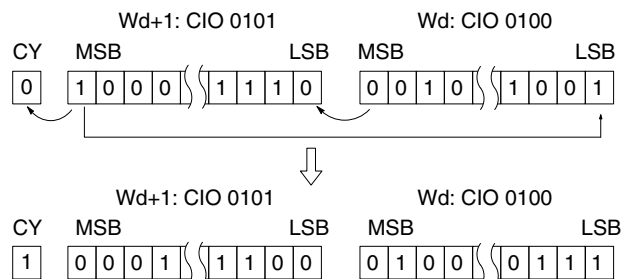
- ER (A50003): Content of \*DM word is not BCD when set for BCD.
- CY (A50004): Receives the status of bit 15 from Wd+1.
- EQ (A50006): Content of Wd and Wd+1 are 0 after execution.
- N (A50008): Same status as bit 15 of Wd+1 after execution.

**Example**

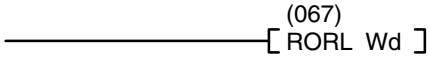
When CIO 000000 is ON in the following example, the status of bit 15 of CIO 0101 is shifted into bit 00 of CIO 0100 and into CY, the status of all bits within CIO 0100 are shifted left one position, the status of bit 15 is shifted to bit 00 of CIO 0101, and the status of all bits within CIO 0101 are shifted left one position.



Address	Instruction	Operands
00000	LD	0000000
00001	RLNL(262)	
		0100

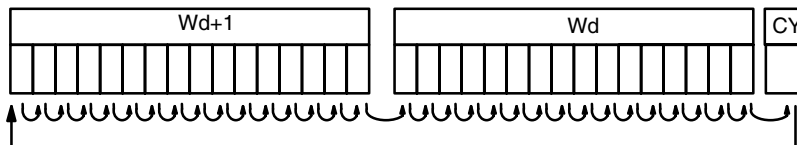


### 5-14-20 DOUBLE ROTATE RIGHT: RORL(067)

<p><b>Ladder Symbol</b></p>  <p><b>Variations</b></p> <p>↑ RORL(067)</p>	<p><b>Operand Data Area</b></p> <p><b>Wd: Word</b>      CIO, G, A, DM</p>
---	---

**Description**

When the execution condition is OFF, RORL(067) is not executed. When the execution condition is ON, RORL(067) shifts CY into bit 15 of Wd+1, all bits previously in Wd and Wd+1 are shifted to the right, and bit 00 of Wd is shifted into CY.



**Precautions**

Use STC(078) to set CY to 1 or CLC(079) to set CY to 0 if necessary before doing a rotate operation to ensure that CY contains the proper status before executing RORL(067).

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

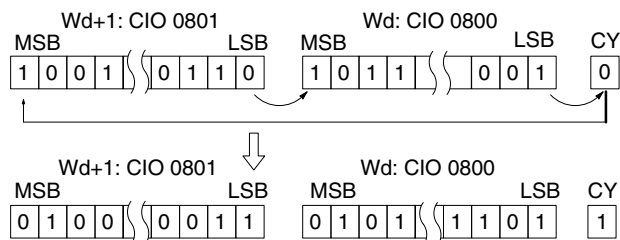
- ER (A50003): Content of \*DM word is not BCD when set for BCD.
- CY (A50004): Receives the status of bit 00 from Wd.
- EQ (A50006): Content of Wd and Wd+1 are 0 after execution.
- N (A50008): Same status as bit 15 of Wd+1 after execution.

**Example**

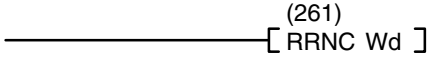
When CIO 000000 is ON in the following example, the status of CY is shifted into bit 15 of CIO 0801, the status of all bits within CIO 0801 are shifted right one position, the status of bit 00 of CIO 0801 is shifted into bit 00 of CIO 0800, the status of all bits within CIO 0800 are shifted right one position, the status of bit 00 of CIO 0800 is shifted into CY.



Address	Instruction	Operands
00000	LD	000000
00001	RORL(067)	
		0800

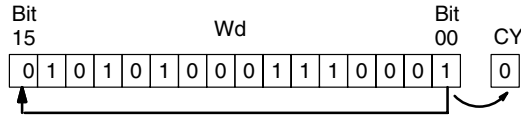


### 5-14-21 ROTATE RIGHT WITHOUT CARRY: RRNC(261)

<p><b>Ladder Symbol</b></p>  <p><b>Variations</b></p> <p>↑RRNC(261)</p>	<p><b>Operand Data Area</b></p> <p><b>Wd: Word</b> CIO, G, A, DM, DR, IR</p>
--	--

**Description**

When the execution condition is OFF, RRNC(261) is not executed. When the execution condition is ON, RRNC(261) shifts all Wd bits one bit to the right, shifting the status of bit 00 into both bit 15 of Wd and into CY.

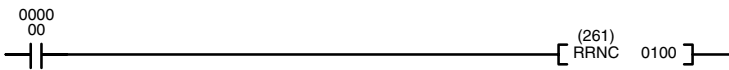


**Flags**

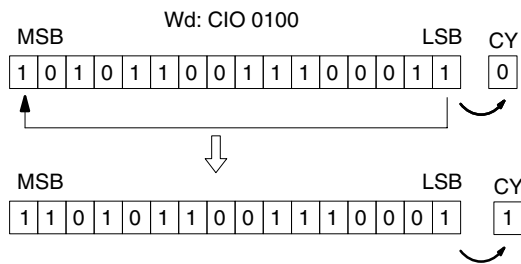
- ER (A50003): Content of \*DM word is not BCD when set for BCD.
- CY (A50004): Receives the status of bit 15 from Wd.
- EQ (A50006): Content of Wd is 0 after execution.
- N (A50008): Same status as bit 15 of Wd after execution.

**Example**

When CIO 000000 is ON in the following example, the status of bit 00 of CIO 0100 is shifted into bit 15 of CIO 0100 and into CR and the status of all bits within CIO 0100 are shifted right one position.



Address	Instruction	Operands
00000	LD	000000
00001	RRNC(261)	
		0100

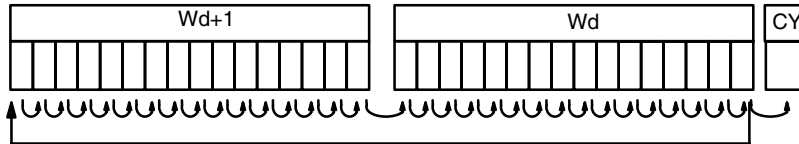


### 5-14-22 DOUBLE ROTATE RIGHT W/O CARRY: RRNL(263)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;"> </p> <p><b>Variations</b></p> <p>↑RRNL(263)</p>	<p><b>Operand Data Area</b></p> <p><b>Wd: Word</b>                      CIO, G, A, DM</p>
---	---

**Description**

When the execution condition is OFF, RRNL(263) is not executed. When the execution condition is ON, RRNL(263) shifts all bits previously in Wd and Wd+1 to the right, and bit 00 of Wd is shifted into both bit 15 of Wd+1 and into CY.



**Precautions**

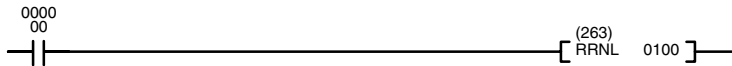
Refer to page 101 for general precautions on operand data areas.

**Flags**

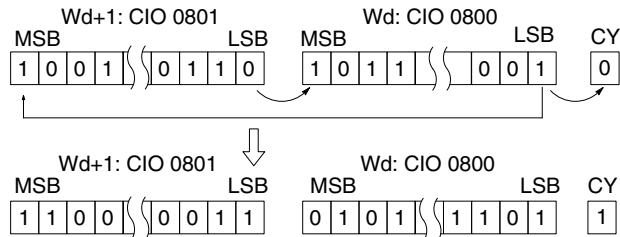
- ER (A50003): Content of \*DM word is not BCD when set for BCD.
- CY (A50004): Receives the status of bit 00 from Wd.
- EQ (A50006): Content of Wd and Wd+1 are 0 after execution.
- N (A50008): Same status as bit 15 of Wd+1 after execution.

**Example**

When CIO 000000 is ON in the following example, the status of bit 00 of CIO 0800 is shifted into bit 15 of CIO 0801 and into CY, the status of all bits within CIO 0801 are shifted right one position, the status of bit 00 is shifted to bit 15 of CIO 0800, and the status of all bits within CIO 0800 are shifted left one position.



Address	Instruction	Operands
00000	LD	000000
00001	RRNL(263)	
		0100

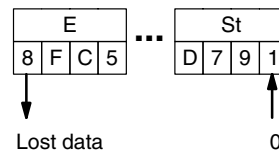


### 5-14-23 ONE DIGIT SHIFT LEFT: SLD(068)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(068)</p> <p>_____ [ SLD    St    E ]</p> <p><b>Variations</b></p> <p>↑ SLD(068)</p>	<p><b>Operand Data Areas</b></p> <p><b>St: Starting word</b>    CIO, G, A, DM</p> <p><b>E: End word</b>        CIO, G, A, DM</p>
---	--

**Description**

When the execution condition is OFF, SLD(068) is not executed. When the execution condition is ON, SLD(068) shifts data between St and E (inclusive) by one digit (four bits) to the left. 0 is written into the rightmost digit of the St, and the content of the leftmost digit of E is lost.



**Precautions**

St must be less than or equal to E. St and E must be in the same data area. The shift operation might not be completed if a power failure occurs during execution of the instruction.

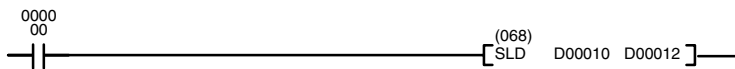
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

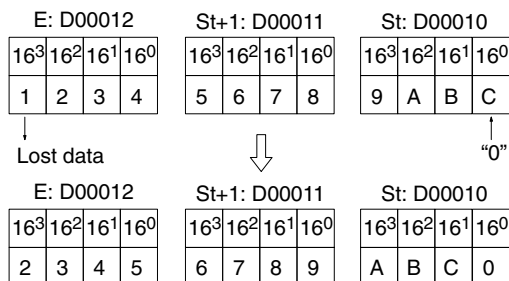
ER (A50003):    Content of \*DM word is not BCD when set for BCD.  
                          St and E are in different areas, or St is greater than E.

**Example**

When CIO 000000 is ON in the following example, 0 is shifted into digit 0 of D00010, the contents of all digits in D00010 are shifted one digit to the left, the content of digit 3 of D00010 is shifted to digit 0 of D00011, the contents of all digits in D00011 are shifted one digit to the left, the content of digit 3 of D00011 is shifted to digit 0 of D00012, the contents of all digits in D00012 are shifted one digit to the left, and the content of digit 3 of D00012 is lost.

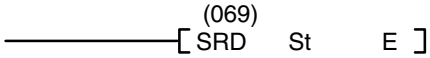


Address	Instruction	Operands
00000	LD	00000
00001	SLD(068)	
		D00010
		D00012



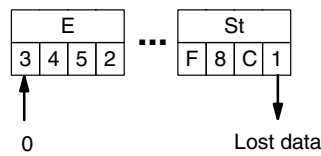


### 5-14-24 ONE DIGIT SHIFT RIGHT: SRD(069)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(069)  </p> <p><b>Variations</b></p> <p>↑ SRD(069)</p>	<p><b>Operand Data Areas</b></p> <p><b>St: Starting word</b> CIO, G, A, DM</p> <p><b>E: End word</b> CIO, G, A, DM</p>
--	--

**Description**

When the execution condition is OFF, SRD(069) is not executed. When the execution condition is ON, SRD(069) shifts data between St and E (inclusive) by one digit (four bits) to the right. 0 is written into the leftmost digit of E and the rightmost digit of St is lost.



**Precautions**

St must be less than or equal to E. St and E must be in the same data area. The shift operation might not be completed if a power interruption occurs during execution of the instruction.

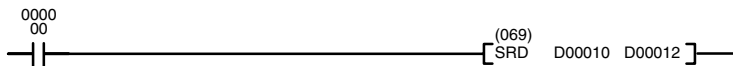
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

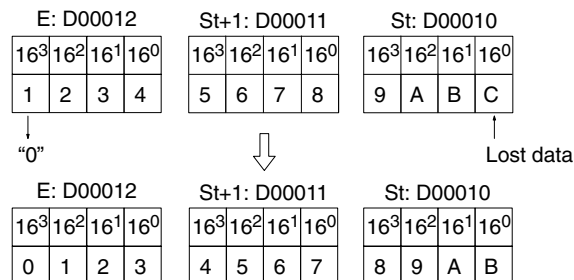
ER (A50003): Content of \*DM word is not BCD when set for BCD. The St and E words are in different areas, or St is greater than E.

**Example**

When CIO 000000 is ON in the following example, 0 is shifted into digit 3 of D00012, the contents of all digits in D00012 are shifted one digit to the right, the content of digit 0 of D00012 is shifted to digit 3 of D00011, the contents of all digits in D00011 are shifted one digit to the right, the content of digit 0 of D00011 is shifted to digit 3 of D00010, the contents of all digits in D00010 are shifted one digit to the right, and the content of digit 0 of D00010 is lost.



Address	Instruction	Operands
00000	LD	000000
00001	SRD(069)	
		D00010
		D00012



## 5-15 Data Movement Instructions

Data Movement Instructions are used for moving data between different addresses in data areas. These movements can be programmed to be within the same data area or between different data areas. Data movement is essential for utilizing all of the data areas of the PC. Effective communications in networks also requires data movement. All of these instructions change only the content of the words to which data is being moved, i.e., the content of source words is the same before and after execution of any of the data movement instructions.

### 5-15-1 MOVE: MOV(030)

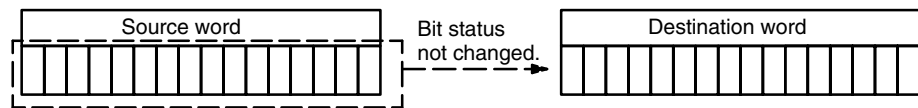
<p><b>Ladder Symbol</b></p> <p><b>Variations</b></p> <p>↑ MOV(030)      ! ↑ MOV(030)</p> <p>! MOV(030)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: Source</b>      CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>D: Destination</b>      CIO, G, A, T, C, DM, DR, IR</p>
--	--

**Description**

When the execution condition is OFF, MOV(030) is not executed. When the execution condition is ON, MOV(030) copies the content of S to D.

If !MOV(030) or !↑MOV(030) is used, input bits used for S will be refreshed just before, and output bits used for D will be refreshed just after execution.

**Note** The immediate-refresh variations of MOV(030) can be used in simplex mode only; they cannot be used in duplex mode.



**Precautions**

Abide by the following guidelines when using MOV(030) to transfer data from the CPU to Special I/O Units.

- Be sure that any require data processing has been completed before executing the move.
- Be sure that the data being transfer remains stable in memory long enough to complete the transfer.
- Be sure to allow enough time between transfers to ensure that data processing is completed.

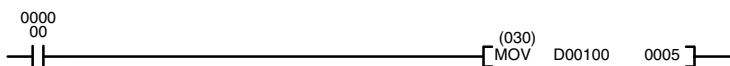
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

- ER (A50003): Content of \*DM word is not BCD when set for BCD.
- EQ (A50006): Content of D is 0 after execution.
- N (A50008): Same status as bit 15 of D after execution.

**Example**

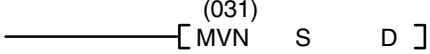
When CIO 000000 is ON in the following example, the content of D00100 is copied into CIO 0005.



Address	Instruction	Operands
00000	LD	000000
00001	MOV(030)	
		D00100
		0005

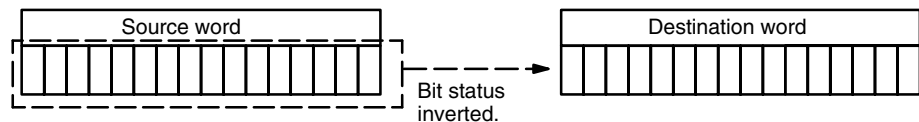


### 5-15-2 MOVE NOT: MVN(031)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(031)  </p> <p><b>Variations</b></p> <p>↑ MVN(031)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: Source</b>      CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>D: Destination</b>      CIO, G, A, T, C, DM, DR, IR</p>
--	--

#### Description

When the execution condition is OFF, MVN(031) is not executed. When the execution condition is ON, MVN(031) transfers the complement of the content of S (specified word or four-digit hexadecimal constant) to D, i.e., for each ON bit in S, the corresponding bit in D is turned OFF, and for each OFF bit in S, the corresponding bit in D is turned ON.

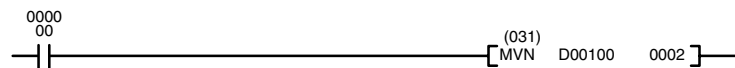


#### Flags

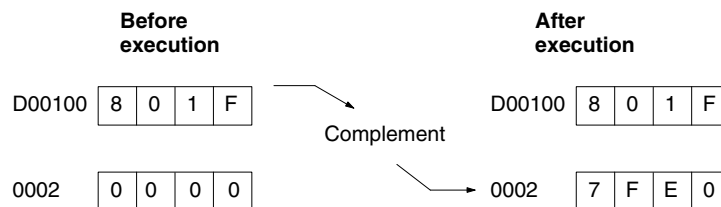
- ER (A50003): Content of \*DM word is not BCD when set for BCD.
- EQ (A50006): Content of D is 0 after execution.
- N (A50008): Same status as bit 15 of D after execution.

#### Example

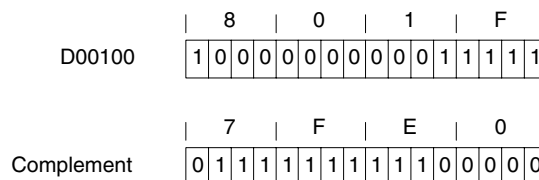
When CIO 000000 is ON in the following example, the complement of the content of D00100 is transferred to the CIO 0002.



Address	Instruction	Operands
00000	LD	000000
00001	MVN(031)	
		D00100
		0002



The bit contents of D00100 and its complement are illustrated below:

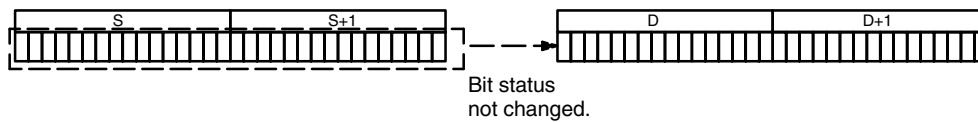


### 5-15-3 DOUBLE MOVE: MOVL(032)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(032)</p> <p style="text-align: center;">— [ MOVL S D ]</p> <p><b>Variations</b></p> <p>↑ MOVL(032)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: Source</b>            CIO, G, A, T, C, #, DM</p> <p><b>D: Destination</b>    CIO, G, A, T, C, DM</p>
--	--

**Description**

When the execution condition is OFF, MOVL(032) is not executed. When the execution condition is ON, MOVL(032) copies the content of S and S+1 to D and D+1.



**Precautions**

Neither D nor S can be the last word in a data area because they designate the first of two words.

Constants are input using eight digits.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

ER (A50003): Content of \*DM word is not BCD when set for BCD.

EQ (A50006): Content of D and D+1 are 0 after execution.

N (A50008): Same status as bit 15 of D+1 after execution.

**Example**

When CIO 000005 is ON in the following example, the contents of A400 and A401 are copied into D00200 and D00201, respectively.



Address	Instruction	Operands
00000	LD	000005
00001	MOVL(032)	
		A400
		D00200

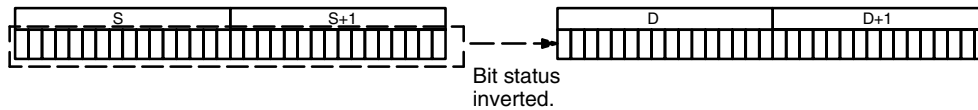
	Before execution	After execution
A400	3 7 0 0	3 7 0 0
A401	0 1 F 9	0 1 F 9
D00200	0 0 0 0	3 7 0 0
D00201	0 0 0 0	0 1 F 9

### 5-15-4 DOUBLE MOVE NOT: MVNL(033)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(033)</p> <p style="text-align: center;">— [ MVNL S D ]</p> <p><b>Variations</b></p> <p>↑ MVNL(033)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: Source</b>      CIO, G, A, T, C, #, DM</p> <p><b>D: Destination</b>    CIO, G, A, T, C, DM</p>
--	--

**Description**

When the execution condition is OFF, MVNL(033) is not executed. When the execution condition is ON, MVNL(033) transfers the complement of the content of S and S+1 (specified words or eight-digit hexadecimal constant) to D and D+1, i.e., for each ON bit in S and S+1, the corresponding bit in D and D+1 is turned OFF, and for each OFF bit in S and S+1, the corresponding bit in D and D+1 is turned ON.



**Precautions**

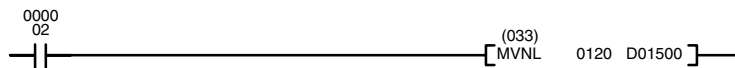
Refer to page 101 for general precautions on operand data areas.

**Flags**

- ER (A50003): Content of \*DM word is not BCD when set for BCD.
- EQ (A50006): Content of D and D+1 are 0 after execution.
- N (A50008): Same status as bit 15 of D+1 after execution.

**Example**

When CIO 000002 is ON in the following example, the complement of the contents of A400 and A401 are copied into D01500 and D01501, respectively.



Address	Instruction	Operands
00000	LD	000002
00001	MVNL(033)	
		0120
		D01500

CIO 0120	7008	→	D01500	8FF7
CIO 0121	AFF0	→	D01501	500F

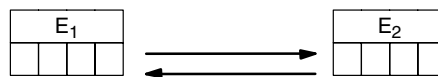
012000	0	} 8	D01501	00	1	} 7
012001	0			01	1	
012002	0			02	1	
012003	1			03	0	
012112	0	} A	D01501	12	1	} 5
012113	1			13	0	
012114	0			14	1	
012115	1			15	0	

### 5-15-5 DATA EXCHANGE: XCHG(034)

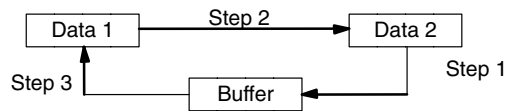
<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(034)</p> <p style="text-align: center;">— [ XCHG E<sub>1</sub> E<sub>2</sub> ]</p> <p><b>Variations</b></p> <p>↑ XCHG(034)</p>	<p><b>Operand Data Areas</b></p> <p><b>E<sub>1</sub>: 1<sup>st</sup> Exchange word</b> CIO, G, A, T, C, DM, DR, IR</p> <p><b>E<sub>2</sub>: 2<sup>nd</sup> Exchange word</b> CIO, G, A, T, C, DM, DR, IR</p>
--	--

**Description**

When the execution condition is OFF, XCHG(034) is not executed. When the execution condition is ON, XCHG(034) exchanges the content of E<sub>1</sub> and E<sub>2</sub>.



If you want to exchange the content of blocks longer than 2 words, use XCGL(035) and/or XCHG(034) and use work words as an intermediate buffer to hold one of the blocks.

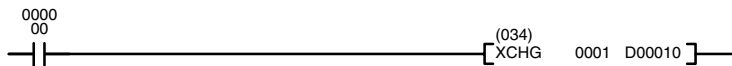


**Flags**

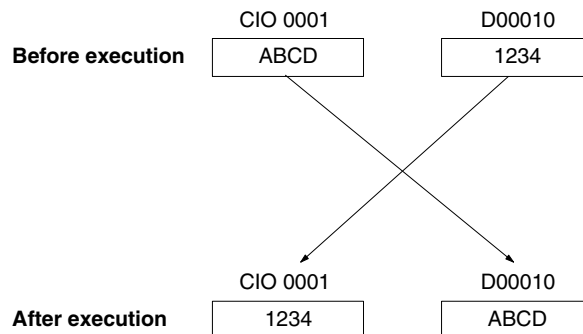
ER (A50003): Content of \*DM word is not BCD when set for BCD.

**Example**

When CIO 00000 is ON in the following example, the content of CIO 0001 is moved to D00010 and the content of D00010 is moved to CIO 0001.



Address	Instruction	Operands
00000	LD	000000
00001	XCHG(034)	
		0001
		D00010

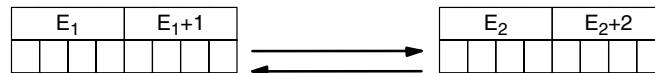


### 5-15-6 DOUBLE DATA EXCHANGE: XCGL(035)

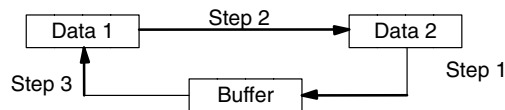
<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(035)</p> <p style="text-align: center;">┌─── XCGL E<sub>1</sub> E<sub>2</sub> ──┐</p> <p><b>Variations</b></p> <p>↑ XCGL(035)</p>	<p><b>Operand Data Areas</b></p> <p><b>E<sub>1</sub>: 1<sup>st</sup> Exchange word</b> CIO, G, A, T, C, DM</p> <p><b>E<sub>2</sub>: 2<sup>nd</sup> Exchange word</b> CIO, G, A, T, C, DM</p>
---	--

**Description**

When the execution condition is OFF, XCGL(035) is not executed. When the execution condition is ON, XCGL(035) exchanges the content of E<sub>1</sub> and E<sub>1</sub>+1 with that of E<sub>2</sub> and E<sub>2</sub>+1.



If you want to exchange the content of blocks longer than 2 words, use XCGL(035) and/or XCHG(034) and use work words as an intermediate buffer to hold one of the blocks.



**Precautions**

Refer to page 101 for general precautions on operand data areas.

**Flags**

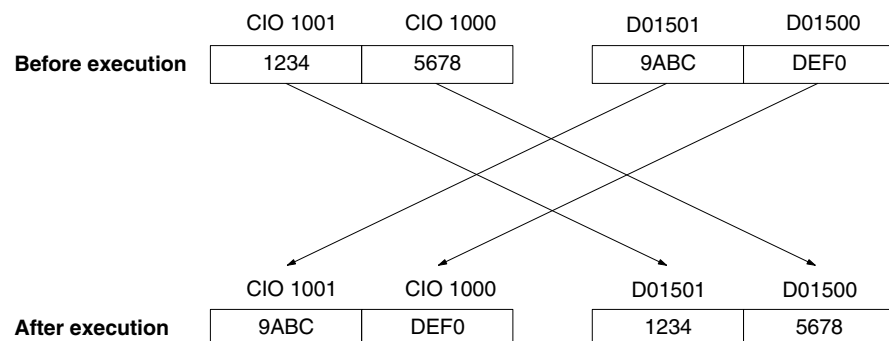
ER (A50003): Content of \*DM word is not BCD when set for BCD.

**Example**

When CIO 00000 is ON in the following example, the contents of CIO 0000 and CIO 0001 are moved to D01500 and D01501, and the contents D01500 and D01501 are moved to CIO 0000 and CIO 0001.



Address	Instruction	Operands
00000	LD	000000
00001	XCGL(035)	
		1000
		D01500



### 5-15-7 MOVE TO REGISTER: MOVR(036)

Ladder Symbol	Operand Data Areas
	<b>S: Source</b> CIO, G, A, T, C, DM <b>D: Destination</b> IR
<b>Variations</b> ↑ MOVR(036)	

**Description**

When the execution condition is OFF, MOVR(036) is not executed. When the execution condition is ON, MOVR(036) copies the PC memory address of word or bit S to the index register designated in D. The index register must be directly addressed.

When S contains a timer or counter number, the PC memory bit address of the timer or counter Completion Flag is copied to the index register. To access the PC memory address of a timer or counter PV with an index register, move the PC memory address of the timer or counter PV (#1000 or #1800) to an index register with MOV(030)

If the index register contains a PC memory address for a timer/counter Completion Flag, a Transition Flag, or a Step Flag, the leftmost three digits indicate the PC memory word address, and the rightmost digit indicates the bit.

**Precautions**

Only direct addresses can be used for IR.

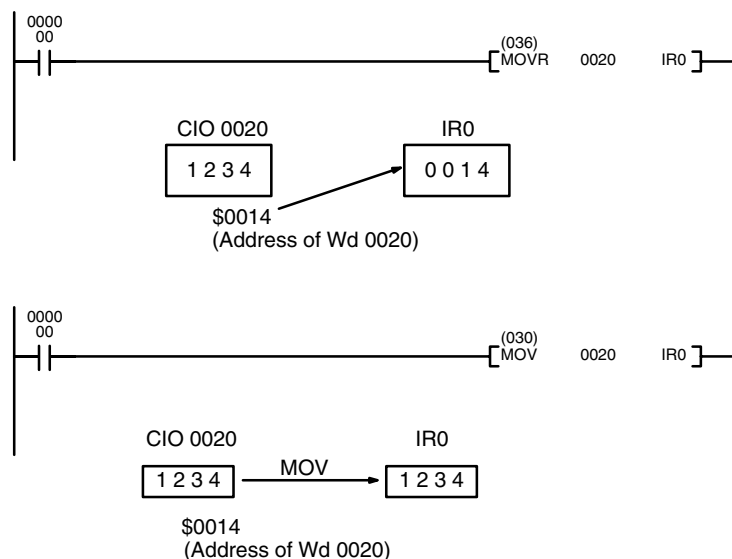
**Flags**

ER (A50003): Content of \*DM word is not BCD when set for BCD.

EQ (A50006): 0000 was placed in the index register.

**Example**

The following example demonstrates the difference between MOV(030) and MOVR(036). In the first instruction line, MOVR(036) copies the PC memory address of CIO 0020 to IR0 when CIO 000000 is ON. In the second instruction line, MOV(030) copies the content of CIO 0020 to IR0 when CIO 000000 is ON.



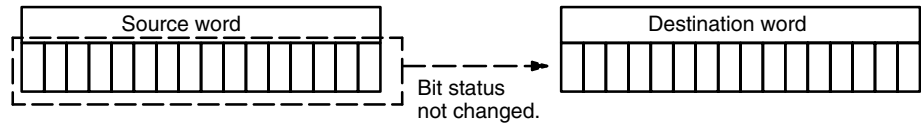


### 5-15-8 MOVE QUICK: MOVQ(037)

Ladder Symbol	Operand Data Areas
	<b>S: Source</b> CIO, G, A, T, C, # <b>D: Destination</b> CIO, G, A, T, C

**Description**

When the execution condition is OFF, MOVQ(037) is not executed. When the execution condition is ON, MOVQ(037) copies the content of S to D at high speed. MOVQ(037) copies the content of S to D at least 10 times faster than MOV(030).

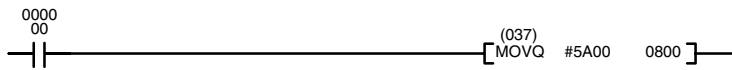


**Flags**

There are no flags affected by MOVQ(037).

**Example**

When CIO 000000 is ON in the following example, 5A00 is copied into CIO 0000.



Address	Instruction	Operands
00000	LD	000000
00001	MOVQ(037)	
		#5A00
		0800

S: #00A5			D: CIO 0800		
"0"	2 <sup>0</sup>	0	→	080000	0
	2 <sup>1</sup>	0		080001	0
	2 <sup>2</sup>	0		080002	0
	2 <sup>3</sup>	0		080003	0
"0"	2 <sup>4</sup>	0		080004	0
	2 <sup>5</sup>	0		080005	0
	2 <sup>6</sup>	0		080006	0
	2 <sup>7</sup>	0		080007	0
"A"	2 <sup>8</sup>	0		080008	0
	2 <sup>9</sup>	1		080009	1
	2 <sup>10</sup>	0		080010	0
	2 <sup>11</sup>	1		080011	1
"5"	2 <sup>12</sup>	1		080012	1
	2 <sup>13</sup>	0		080013	0
	2 <sup>14</sup>	1		080014	1
	2 <sup>15</sup>	0	080015	0	

### 5-15-9 MULTIPLE BIT TRANSFER: XFRB(038)

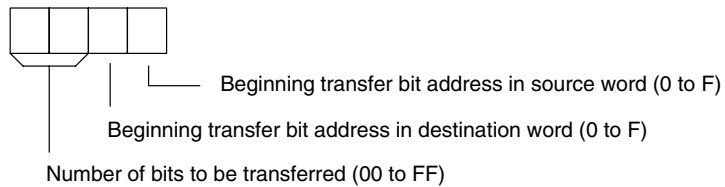
Ladder Symbol	Operand Data Areas
	<p><b>C: Control word</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>S: First source word</b> CIO, G, A, T, C, DM</p> <p><b>D: First destination word</b> CIO, G, A, DM</p>
<p><b>Variations</b></p> <p>↑XFRB(038)</p>	

**Description**

When the execution condition is OFF, XFRB(038) is not executed. When the execution condition is ON, XFRB(038) transfers specified consecutive bits to a destination beginning with a specified bit in a specified word.

The address of the beginning bit to be transferred is designated in hexadecimal (0 to F) in the control word (C). The number of bits to be transferred can be specified within a range of 0 to 255, in hexadecimal (00 to FF). If "0" is specified, no data will be transferred.

**Control Word Contents**



Except for the bits that are transferred, none of the contents of the destination word(s) will be changed.

**Precautions**

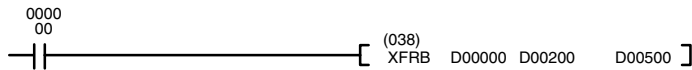
Be sure that the last word in the transfer source or transfer destination does not exceed the data area.

**Flags**

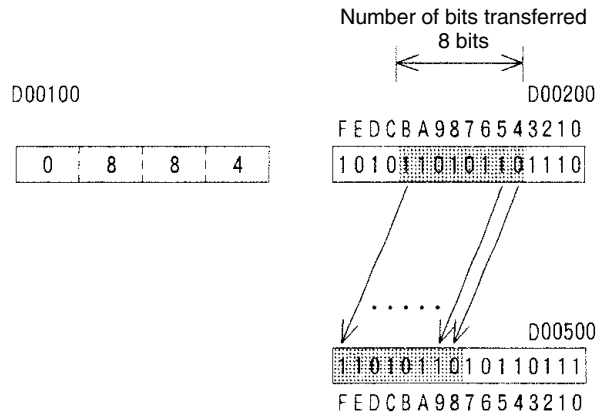
ER (A50003): Content of a\*DM word is not BCD when set for BCD.

**Example 1**

When CIO 000000 is ON in the following example, eight bits from D00200 (beginning with bit 04) will be transferred to D00500 (beginning with bit 08), according to the contents of D00100.

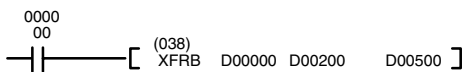


Address	Instruction	Operands
00000	LD	000000
00001	XFRB(038)	
		D00000
		D00200
		D00500

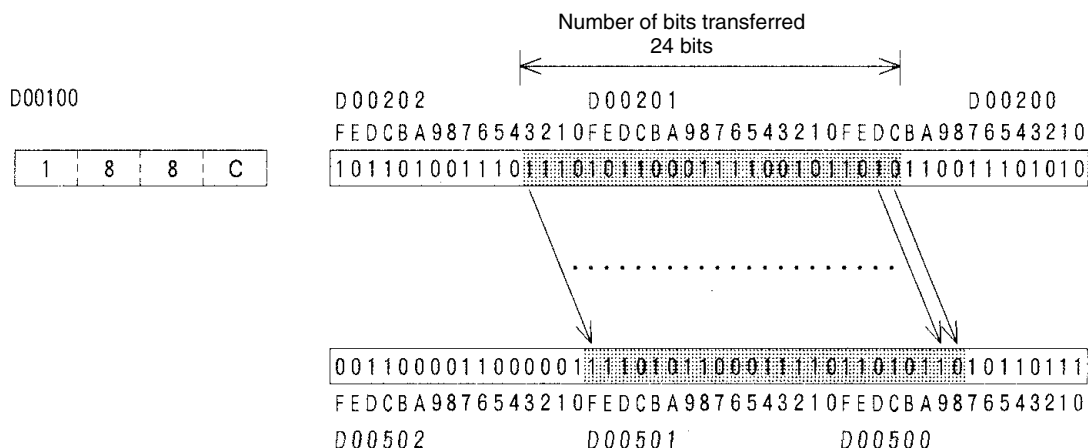


**Example 2**

When CIO 000000 is ON in the following example, 24 bits beginning with bit 12 of D00200 are transferred to D00500 (beginning with bit 08), as specified by the contents of D00100.



Address	Instruction	Operands
00000	LD	000000
00001	XFRB(038)	
		D00000
		D00200
		D00500

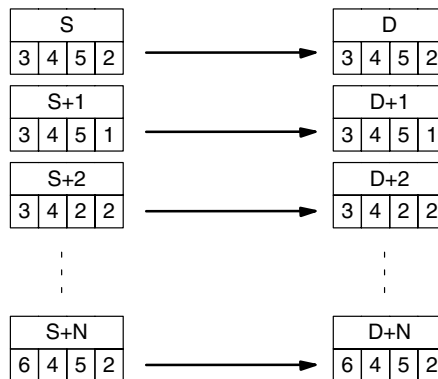


### 5-15-10 BLOCK TRANSFER: XFER(040)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(040)</p> <p>— [ XFER    N        S        D ]</p> <p><b>Variations</b></p> <p>↑ XFER(040)</p>	<p><b>Operand Data Areas</b></p> <p><b>N: Number of words</b>    CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>S: 1<sup>st</sup> source word</b>     CIO, G, A, T, C, DM</p> <p><b>D: 1<sup>st</sup> destination word</b> CIO, G, A, T, C, DM</p>
---	--

**Description**

When the execution condition is OFF, XFER(040) is not executed. When the execution condition is ON, XFER(040) copies the contents of S, S+1, ..., S+N to D, D+1, ..., D+N.



**Precautions**

Both S and D may be in the same data area, but their respective block areas must not overlap. S and S+N must be in the same data area, as must D and D+N. N must be BCD.

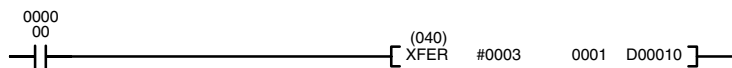
- Note**
1. Transfer source and destination words can overlap, so XFER(040) can be used for word shift operations.
  2. Refer to page 101 for general precautions on operand data areas.

**Flags**

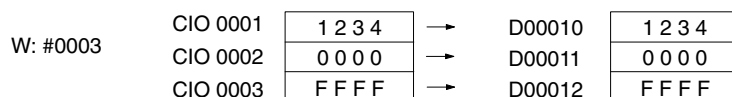
ER (A50003):    N is not BCD.  
 Content of \*DM word is not BCD when set for BCD.

**Example**

When CIO 00000 is ON in the following example, the contents of CIO 0001 through CIO 0003 are copied into D00010 through D00012, as specified by the first operand (#0003).



Address	Instruction	Operands
00000	LD	000000
00001	XFER(040)	
		#0003
		0001
		D00010

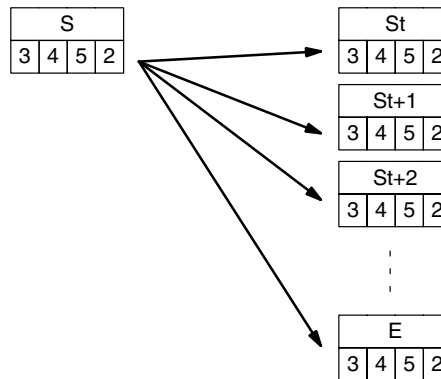


### 5-15-11 BLOCK SET: BSET(041)

Ladder Symbol	Operand Data Areas
$\xrightarrow{(041)}$ $\text{--- [ BSET S St E ]}$	<b>S: Source word</b> CIO, G, A, T, C, #, DM, DR, IR <b>St: Starting word</b> CIO, G, A, T, C, DM <b>E: End word</b> CIO, G, A, T, C, DM
<b>Variations</b> ↑ BSET(041)	

**Description**

When the execution condition is OFF, BSET(041) is not executed. When the execution condition is ON, BSET(041) copies the content of S to all words from St through E.



BSET(041) can be used to change timer/counter PV. BSET(041) can also be used to clear sections of a data area, i.e., the DM area, to prepare for executing other instructions.

**Precautions**

St must be less than or equal to E. St and E must be in the same data area. The BSET(041) operation might not be completed if a power interruption occurs during execution of the instruction.

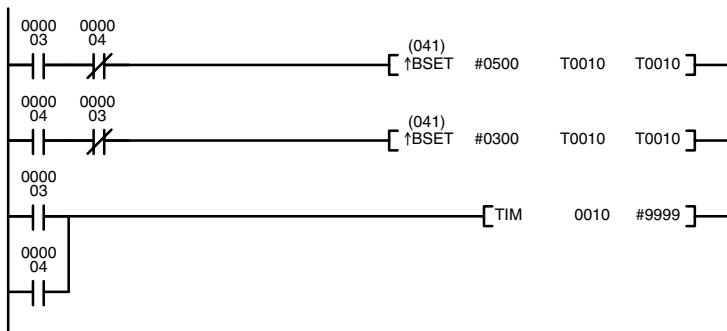
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

ER (A50003): Content of \*DM word is not BCD when set for BCD.  
 St is greater than E.

**Example**

The following example shows how to use BSET(041) to change the PV of a timer depending on the status of CIO 000003 and CIO 000004. When CIO 000003 is ON, TIM 0010 will operate as a 50-second timer; when CIO 000004 is ON, TIM 0010 will operate as a 30-second timer.



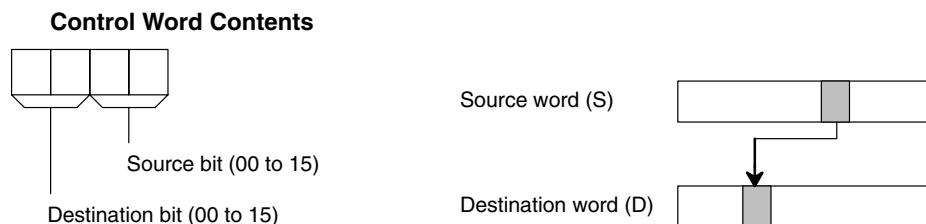
Address	Instruction	Operands
00000	LD	000003
00001	AND NOT	000004
00002	↑BSET(041)	
		#0500
		T0010
		T0010
00003	LD	000004
00004	AND NOT	000003
00005	↑BSET(041)	
		#0300
		T0010
		T0010
00006	LD	000003
00007	OR	000004
00008	TIM	0010
		#9999

### 5-15-12 MOVE BIT: MOVB(042)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;"> </p> <p><b>Variations</b></p> <p>↑MOVB(042)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: Source word or data</b>      CIO, G, A, #, DM, DR, IR</p> <p><b>C: Control word</b>              CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>D: Destination word</b>        CIO, G, A, DM, DR, IR</p>
---	--

**Description**

When the execution condition is OFF, MOV B(042) is not executed. When the execution condition is ON, MOV B(042) transfers a single specified bit from the source word to the specified bit in the destination word. The other bits in the destination word are not changed.



**Precautions**

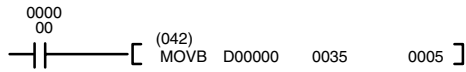
C must be BCD and must be within the values specified above.

**Flags**

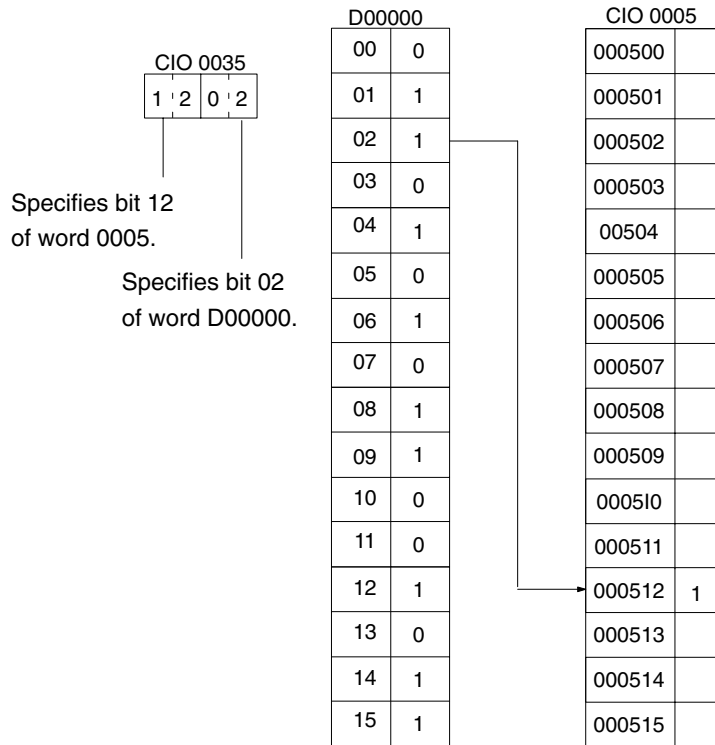
ER (A50003): Control word content is not BCD.  
 Rightmost and leftmost eight bits are not 00 to 15.  
 Content of a\*DM word is not BCD when set for BCD.

**Example**

When CIO 00000 is ON in the following example, the content of bit 02 of the transfer source word (D00000) is copied to bit 12 of the transfer destination word (CIO 0005) as specified by the contents (1202) of control word (CIO 0035).



Address	Instruction	Operands
00000	LD	000000
00001	MOVB(042)	
		D00000
		0035
		0005

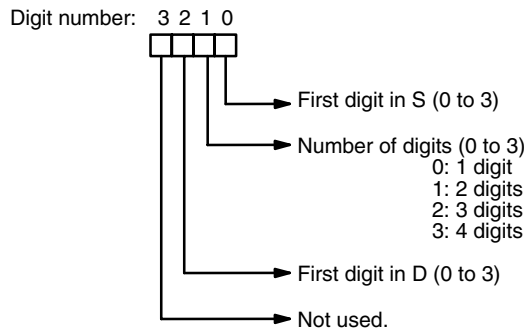


### 5-15-13 MOVE DIGIT: MOVD(043)

<p><b>Ladder Symbol</b></p> <p><b>Variations</b></p> <p>↑ MOVD(043)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: Source word</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>Di: Digit designator</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>D: Destination word</b> CIO, G, A, T, C, DM, DR, IR</p>
---	--

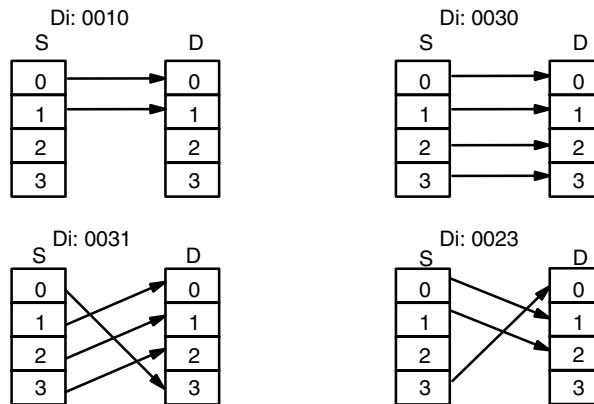
**Description**

When the execution condition is OFF, MOVD(043) is not executed. When the execution condition is ON, MOVD(043) copies the content of the specified digit(s) in S to the specified digit(s) in D. Up to four digits can be transferred at one time. The first digit to be copied, the number of digits to be copied, and the first digit to receive the copy are designated in Di as shown below. Digits from S will be copied to consecutive digits in D starting from the designated first digit and continued for the designated number of digits. If the last digit is reached in either S or D, further digits are used starting back at digit 0.



**Digit Designator**

The following show examples of the data movements for various values of Di.



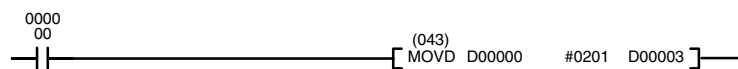
**Precautions**

The rightmost three digits of Di must each be between 0 and 3.

**Note** Refer to page 101 for general precautions on operand data areas.

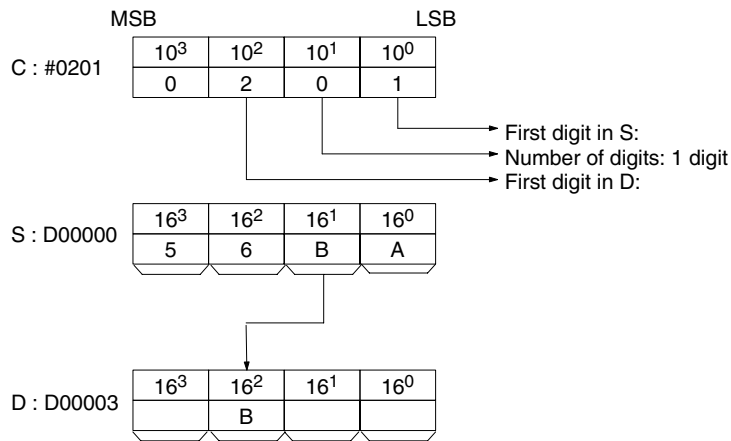
**Flags**

ER (A50003): Content of \*DM word is not BCD when set for BCD.  
At least one of the rightmost three digits of Di is not between 0 and 3.



Address	Instruction	Operands
00000	LD	000000
00001	MOVD(043)	
		D00000
		#0201
		D00003





### 5-15-14 SINGLE WORD DISTRIBUTE: DIST(044)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(044)</p> <p>— [ DIST S DBs Of ]</p> <p><b>Variations</b></p> <p>↑ DIST(044)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: Source data</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>DBs: Destination base</b> CIO, G, A, T, C, DM</p> <p><b>Of: Offset data</b> CIO, G, A, T, C, DM</p>
---	--

**Description**

When the execution condition is OFF, DIST(044) is not executed. When the execution condition is ON, DIST(044) copies the content of S to DBs+Of, i.e., Of is added to DBs to determine the destination word.



**Precautions**

Of must be BCD.

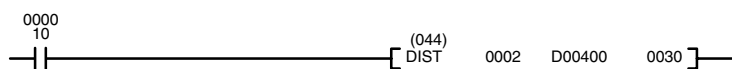
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

- ER (A50003): Of is not BCD.  
Content of \*DM word is not BCD when set for BCD.
- EQ (A50006): Content of S is 0.
- N (A50008): Same status as bit 15 of D+Of after execution.

**Example**

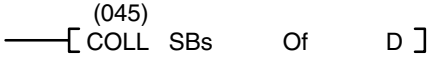
When CIO 000010 is ON in the following example, the content of CIO 0002 is copied into D00410. The destination word D00410 is determined by adding the content of C0030 (i.e., 10) to D00400.



Address	Instruction	Operands
00000	LD	000010
00001	DIST(044)	
		0002
		D00400
		0030

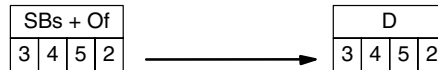
	Before execution		After execution
CIO 0002	4 5 6 7	CIO 0002	4 5 6 7
CIO 0030	0 0 1 0	CIO 0030	0 0 1 0
D00410	0 0 0 0	D00410	4 5 6 7

### 5-15-15 DATA COLLECT: COLL(045)

<p><b>Ladder Symbol</b></p> <p>(045)  </p> <p><b>Variations</b></p> <p>↑ COLL(045)</p>	<p><b>Operand Data Areas</b></p> <p><b>SBs: Source base</b> CIO, G, A, T, C, DM</p> <p><b>Of: Offset data</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>D: Destination word</b> CIO, G, A, T, C, DM, DR, IR</p>
---	--

**Description**

When the execution condition is OFF, COLL(045) is not executed. When the execution condition is ON, COLL(045) copies the content of SBs + Of to D, i.e., Of is added to SBs to determine the source word.



**Precautions**

Of must be BCD.

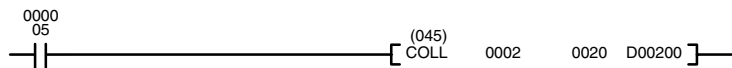
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

- ER (A50003): Of is not BCD.  
Content of \*DM word is not BCD when set for BCD.
- EQ (A50006): Content of S is 0.
- N (A50008): Same status as bit 15 of D after execution.

**Example**

When CIO 000005 is ON in the following example, the source word (CIO 0007) is calculated by adding 5 (the content of CIO 0020) to the source base word (CIO 0002). The content of CIO 0007 is then copied to the destination word (D00200).



Address	Instruction	Operands
00000	LD	000005
00001	COLL(045)	
		0002
		0020
		D00200

	Before execution		After execution
CIO 0020	0 0 0 5	CIO 0020	0 0 0 5
CIO 007	4 0 9 5	CIO 007	4 0 9 5
D00200	0 0 0 0	D00200	4 0 9 5

### 5-15-16 INTERBANK BLOCK TRANSFER: BXFR(046)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(046)</p> <p style="text-align: center;">——— [ BXFR C S D ]</p> <p><b>Variations</b></p> <p>↑BXFR(046)</p>	<p><b>Operand Data Areas</b></p> <p><b>C: First control word</b>      CIO, G, A, T, C, DM</p> <p><b>S: First source word</b>      CIO, G, A, T, C, DM</p> <p><b>D: First destination word</b>    CIO, G, A, T, C, DM</p>
---	--

**Description**

When the execution condition is OFF, BXFR(046) is not executed. When the execution condition is ON, BXFR(046) transfers specified consecutive words from the source bank to a destination beginning with a specified word in a specified bank (D).

The number of words to be transferred and the bank numbers are set as BCD data in two control words (C and C+1). Make sure that the last word in the transfer source or transfer destination does not exceed the data area.

**Control Word Contents**

Dest. bank no.		Source bank no.	
x10 <sup>3</sup>	x10 <sup>2</sup>	x10 <sup>1</sup>	x10 <sup>0</sup>
0	0	0	x10 <sup>4</sup>

} Bank no.: 0 to 7  
Number of words transferred  
1 to 32,766

**Precautions**

The source and destination words can both be in same data area, but they must not overlap.

If the words specified by S and D are not EM words, the specified bank number will not be valid.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

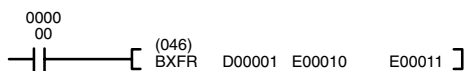
ER (A50003): Bank number is not 00 to 07.

No EM for the specified bank number.

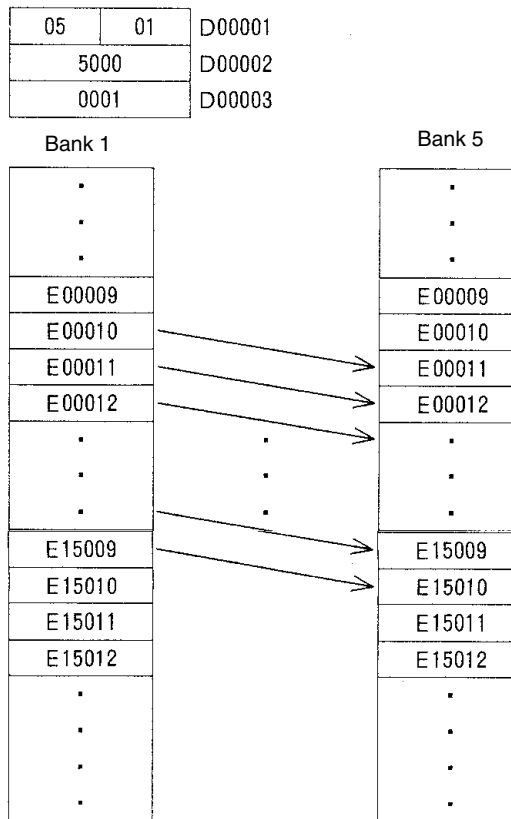
Content of a\*DM word is not BCD when set for BCD.

**Example**

When CIO 000000 is ON in the following example, E00010 through E15000 from EM bank 1 are transferred to EM bank 5 beginning with E00011, according to the contents of D00001 through D00003.



Address	Instruction	Operands
00000	LD	000000
00001	BXFR(046)	
		D00001
		E00010
		E00011



## 5-16 Comparison Instructions

Comparison Instructions are used for comparing data. All comparison instructions affect only the comparison flags and/or results output words. They do not affect the content of the data being compared.

### 5-16-1 COMPARE: CMP(020)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;"> <math>\text{---} \left[ \text{CMP} \begin{matrix} (020) \\ \text{Cp}_1 \end{matrix} \text{ Cp}_2 \right] \text{---}</math> </p> <p><b>Variations</b></p> <p>! CMP(020)</p>	<p><b>Operand Data Areas</b></p> <p><b>Cp<sub>1</sub>: 1<sup>st</sup> compare word</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>Cp<sub>2</sub>: 2<sup>nd</sup> compare word</b> CIO, G, A, T, C, #, DM, DR, IR</p>
--	--

**Description**

When the execution condition is OFF, CMP(020) is not executed. When the execution condition is ON, CMP(020) compares Cp<sub>1</sub> and Cp<sub>2</sub> and outputs the result to the GR, EQ, and LE Flags in the Auxiliary Area.

If !CMP(020) is used, any input bits used for Cp<sub>1</sub> and Cp<sub>2</sub> are refreshed just before execution. (The immediate-refresh variations of CMP(020) can be used in simplex mode only; it cannot be used in duplex mode.)

CMP(020) is an intermediate instruction, like NOT(010), CMPL(021), and EQU(025). Intermediate instructions are entered between conditions or between a condition and a right-hand instruction. Intermediate instructions cannot be placed at the end of an instruction.

CVM1D CPUs are equipped with a similar non-intermediate instruction: CMP(028).

**Precautions**

When comparing a value to the PV of a timer or counter, the value must be in BCD.

Placing other instructions between CMP(020) and the operation which accesses the EQ, LE, and GR Flags may change the status of these flags. Be sure to access them before the desired status is changed.

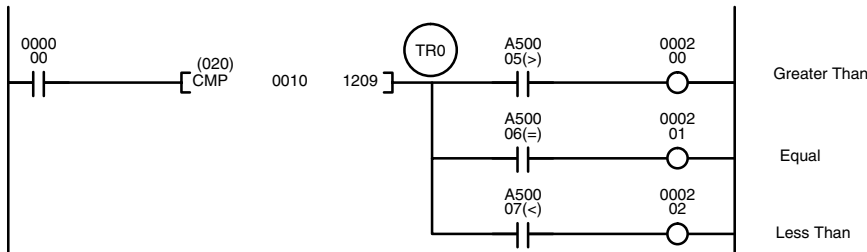
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

- ER (A50003): Content of \*DM word is not BCD when set for BCD.
- GR (A50005): ON if Cp<sub>1</sub> is greater than Cp<sub>2</sub>.
- EQ (A50006): ON if Cp<sub>1</sub> equals Cp<sub>2</sub>.
- LE (A50007): ON if Cp<sub>1</sub> is less than Cp<sub>2</sub>.

**Example 1:  
Saving CMP(020) Results**

The following example shows how to save the comparison result immediately. If the content of word 0010 is greater than that of word 1209, bit 000200 is turned ON; if the two contents are equal, bit 000201 is turned ON; if content of word 0010 is less than that of word 1209, bit 000202 is turned ON. In some applications, only one of the three OUTs would be necessary, making the use of TR 0 unnecessary. With this type of programming, bits 000200, 000201, and 000202 are changed only when CMP(020) is executed.

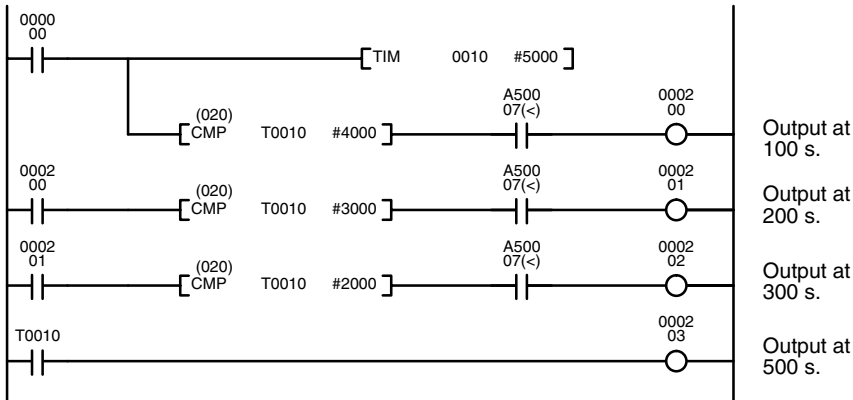


Address	Instruction	Operands
00000	LD	000000
00001	CMP(020)	
		0010
		1209
00002	OUT	TR0
00003	AND	A50005
00004	OUT	000200
00005	LD	TR0
00006	AND	A50006
00007	OUT	000201
00008	LD	TR0
00009	AND	A50007
00010	OUT	000202

**Example 2:  
Obtaining Indications  
during Timer Operation**

The following example uses TIM, CMP(020), and the LE Flag (A50007) to produce outputs at particular times in the timer's countdown. The timer is started by turning ON bit 000000. When bit 000000 is OFF, TIM 0010 is reset and the second two CMP(020)s are not executed (i.e., executed with OFF execution conditions). Output 000200 is produced after 100 seconds; output 000201, after 200 seconds; output 000202, after 300 seconds; and output 000204, after 500 seconds.

The branching structure of this diagram is important in order to ensure that 000200, 000201, and 000202 are controlled properly as the timer counts down. Because all of the comparisons here use the timer's PV as reference, the other operand for each CMP(020) must be in 4-digit BCD.



Address	Instruction	Operands
00000	LD	000000
00001	TIM	0010 #5000
00002	CMP(020)	
		T0010
		#4000
00003	AND	A50007
00004	OUT	000200
00005	LD	000200
00006	CMP(020)	
		T0010
		#3000
00007	AND	A50007
00008	OUT	000201
00009	LD	000201
00010	CMP(020)	
		T0010
		#2000
00011	AND	A50007
00012	OUT	000202
00013	LD	T0010
00014	OUT	000203

### 5-16-2 DOUBLE COMPARE: CMPL(021)

Ladder Symbol	Operand Data Areas
	<p><b>Cp1: 1<sup>st</sup> compare word</b> CIO, G, A, T, C, #, DM</p> <p><b>Cp2: 2<sup>nd</sup> compare word</b> CIO, G, A, T, C, #, DM</p>

#### Description

When the execution condition is OFF, CMPL(021) is not executed. When the execution condition is ON, CMPL(021) compares the eight-digit content of Cp1+1 and Cp1 to the eight-digit content of Cp2+1 and Cp2 and outputs the result to the GR, EQ, and LE Flags in the Auxiliary Area.

CMPL(021) is an intermediate instruction, like CMP(020). Intermediate instructions are entered between conditions or between a condition and a right-hand instruction. Intermediate instructions cannot be placed at the end of an instruction line.

CVM1D CPUs are equipped with a similar non-intermediate instruction: CMPL(029).

Constants are expressed in eight digits.

#### Precautions

When comparing a value to the PVs of timers or counters, the value must be in BCD.

Placing other instructions between CMPL(021) and the operation which accesses the EQ, LE, and GR Flags may change the status of these flags. Be sure to access them before the desired status is changed.

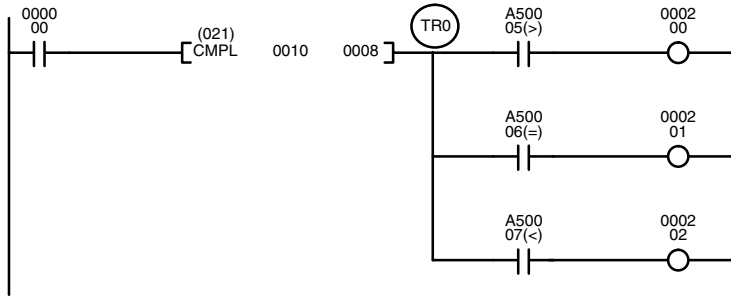
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

- ER (A50003): Content of \*DM word is not BCD when set for BCD.
- GR (A50005): Cp1+1 and Cp1 is greater than Cp2+1 and Cp2.
- EQ (A50006): Cp1+1 and Cp1 equals Cp2+1 and Cp2.
- LE (A50007): Cp1+1 and Cp1 is less than Cp2+1 and Cp2.

**Example**

When CIO 000000 is ON in the following example, the eight-digit content of CIO 0011 and CIO 0010 is compared to the eight-digit content of CIO 0009 and CIO 0008 and the result is output to the GR, EQ, and LE Flags. The results recorded in the GR, EQ, and LE Flags are immediately saved to CIO 000200 (Greater Than), CIO 000201 (Equals), and CIO 000202 (Less Than).



Address	Instruction	Operands
00000	LD	000000
00001	CMPL(021)	
		0010
		0008
00002	OUT	TR0
00003	AND	A50005
00004	OUT	000200
00005	LD	TR0
00006	AND	A50006
00007	OUT	000201
00008	LD	TR0
00009	AND	A50007
00010	OUT	000202

**5-16-3 BLOCK COMPARE: BCMP(022)**

Ladder Symbol	Operand Data Areas
	<p><b>S: Source data</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>CB: 1<sup>st</sup> block word</b> CIO, G, A, T, C, DM</p> <p><b>R: Result word</b> CIO, G, A, T, C, DM, DR, IR</p>
<p><b>Variations</b></p> <p>↑ BCMP(022)</p>	

**Description**

When the execution condition is OFF, BCMP(022) is not executed. When the execution condition is ON, BCMP(022) compares S to the ranges defined by a block consisting of of CB, CB+1, CB+2, ..., CB+32. Each range is defined by two words, the first one providing the lower limit and the second word providing the upper limit. If S is found to be within any of these ranges (inclusive of the upper and lower limits), the corresponding bit in R is set. The comparisons that are made and the corresponding bit in R that is set for each true comparison are shown below. The rest of the bits in R will be turned OFF.

- CB ≤ S ≤ CB+1 Bit 00
- CB+2 ≤ S ≤ CB+3 Bit 01
- CB+4 ≤ S ≤ CB+5 Bit 02
- CB+6 ≤ S ≤ CB+7 Bit 03
- CB+8 ≤ S ≤ CB+9 Bit 04
- CB+10 ≤ S ≤ CB+11 Bit 05
- CB+12 ≤ S ≤ CB+13 Bit 06
- CB+14 ≤ S ≤ CB+15 Bit 07
- CB+16 ≤ S ≤ CB+17 Bit 08
- CB+18 ≤ S ≤ CB+19 Bit 09
- CB+20 ≤ S ≤ CB+21 Bit 10
- CB+22 ≤ S ≤ CB+23 Bit 11

CB+24 ≤ S ≤ CB+25      Bit 12  
 CB+26 ≤ S ≤ CB+27      Bit 13  
 CB+28 ≤ S ≤ CB+29      Bit 14  
 CB+30 ≤ S ≤ CB+31      Bit 15

**Precautions**

Each lower limit word in the comparison block must be less than or equal to the upper limit.

CB cannot be one of the last 31 words in a data area because it designates the first of 32 words.

**Note** Refer to page 101 for general precautions on operand data areas.

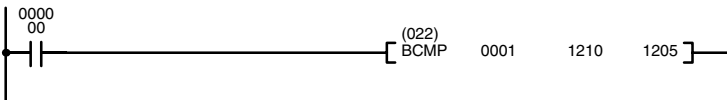
**Flags**

ER (A50003): Content of \*DM word is not BCD when set for BCD.

EQ (A50006): Content of R is zero after execution.

**Example**

The following example shows the comparisons made and the results provided for BCMP(022). Here, the comparison is made during each scan when CIO 000000 is ON.



Address	Instruction	Operands
00000	LD	000000
00001	BCMP(022)	
		0001
		1210
		1205

S: 0001

Lower limits

Upper limits

R: 1205

0001	0210
------	------

Compare data in 0001 (which contains 0210) with the given ranges.

1210	0000
1212	0101
1214	0201
1216	0301
1218	0401
1220	0501
1222	0601
1224	0701
1226	0801
1228	0901
1230	1001
1232	1101
1234	1201
1236	1301
1238	1401
1240	1501

1211	0100
1213	0200
1215	0300
1217	0400
1219	0500
1221	0600
1223	0700
1225	0800
1227	0900
1229	1000
1231	1100
1233	1200
1235	1300
1237	1400
1239	1500
1241	1600

120500	0
120501	0
120502	1
120503	0
120504	0
120505	0
120506	0
120507	0
120508	0
120509	0
120510	0
120511	0
120512	0
120513	0
120514	0
120515	0



### 5-16-4 TABLE COMPARE: TCMP(023)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(023)</p> <p>— [ TCMP S TB R ]</p> <p><b>Variations</b></p> <p>↑ TCMP(023)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: Source data</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>TB: 1<sup>st</sup> table word</b> CIO, G, A, T, C, DM</p> <p><b>R: Result word</b> CIO, G, A, T, C, DM, DR, IR</p>
---	---

**Description**

When the execution condition is OFF, TCMP(023) is not executed. When the execution condition is ON, TCMP(023) compares S to the content of TB, TB+1, TB+2, ..., and TB+15. If S is equal to the content of any of these words, the corresponding bit in R is turned ON, i.e., if S equals the content of TB, bit 00 is turned ON, if it equals the content of TB+1, bit 01 is turned ON, etc. The rest of the bits in R will be turned OFF.

**Precautions**

TB cannot be one of the last 15 words in a data area because it designates the first of 16 words.

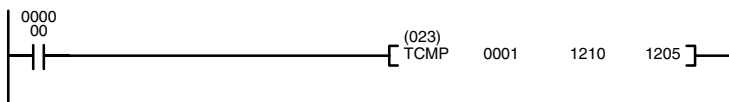
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

ER (A50003): Content of \*DM word is not BCD when set for BCD.  
 EQ (A50006): Content of R is zero after execution.

**Example**

The following example shows the comparisons made and the results provided for TCMP(023). Here, the comparison is made during each scan when CIO 000000 is ON.



Address	Instruction	Operands
00000	LD	000000
00001	TCMP(023)	
		0001
		1210
		1205

S: 0001

0001 0210

Compare the data in CIO 0001 with the given values.

Compare table

1210	0100
1211	0200
1212	0210
1213	0400
1214	0500
1215	0600
1216	0210
1217	0800
1218	0900
1219	1000
1220	0210
1221	1200
1222	1300
1223	1400
1224	0210
1225	1600

R: 1205

120500	0
120501	0
120502	1
120503	0
120504	0
120505	0
120506	1
120507	0
120508	0
120509	0
120510	1
120511	0
120512	0
120513	0
120514	1
120515	0

### 5-16-5 MULTIPLE COMPARE: MCMP(024)

<p><b>Ladder Symbol</b></p> <p><b>Variations</b></p> <p>↑ MCMP(024)</p>	<p><b>Operand Data Areas</b></p> <p><b>TB<sub>1</sub>: 1<sup>st</sup> table word</b> CIO, G, A, T, C, DM</p> <p><b>TB<sub>2</sub>: 2<sup>nd</sup> table word</b> CIO, G, A, T, C, DM</p> <p><b>R: Result word</b> CIO, G, A, DM, DR, IR</p>
---	---

**Description**

When the execution condition is OFF, MCMP(024) is not executed. When the execution condition is ON, MCMP(024) compares the contents of the 16 words TB<sub>1</sub> through TB<sub>1</sub>+15 to the contents of the 16 words TB<sub>2</sub> through TB<sub>2</sub>+15, and turns ON the corresponding bit in word R when the contents are not equal. The content of TB<sub>1</sub> is compared to the content of TB<sub>2</sub>, the content of TB<sub>1</sub>+1 to the content of TB<sub>2</sub>+1, ..., and the content of TB<sub>1</sub>+15 to the content of TB<sub>2</sub>+15. If the content of TB<sub>1</sub>+n is equal to the content of TB<sub>2</sub>+n, bit n of R is turned ON, if the contents are not equal, bit n of R is turned OFF.

**Precautions**

TB<sub>1</sub> and TB<sub>2</sub> cannot be one of the last 15 words in a data area because they designate the first of 16 words.

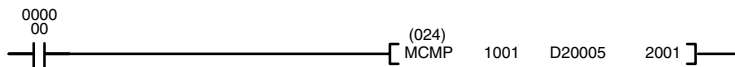
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

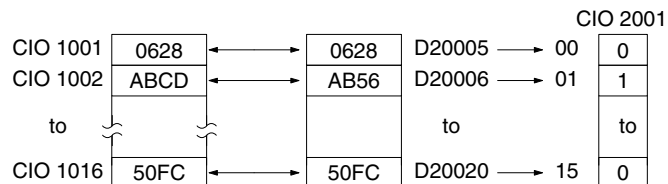
ER (A50003): Content of \*DM word is not BCD when set for BCD.  
 EQ (A50006): Content of R is zero after execution (i.e., if the contents of TB<sub>1</sub> through TB<sub>1</sub>+15 and TB<sub>2</sub> through TB<sub>2</sub>+15 are identical)

**Example**

When CIO 000000 is ON in the following example, words from CIO 1001 through CIO 1016 are compared in order to words from D20005 through D20020 and corresponding bits in CIO 2001 are turn ON when for any pairs of values that are **not** equal.



Address	Instruction	Operands
00000	LD	000000
00001	MCMP(024)	
		1001
		D20005
		2001



### 5-16-6 EQUAL: EQU(025)

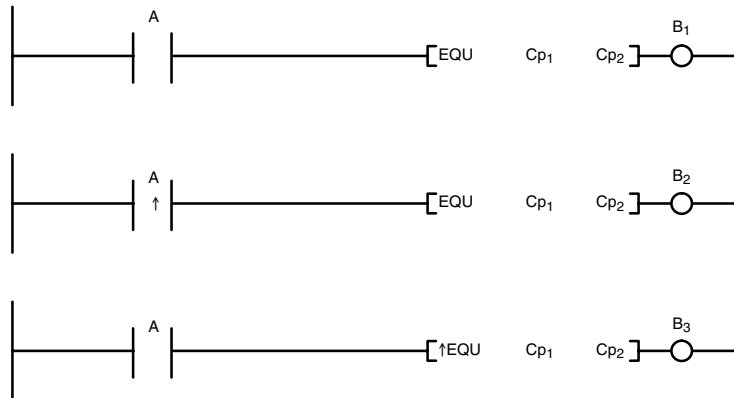
<p><b>Ladder Symbol</b></p> <p><b>Variations</b></p> <p>↑ EQU(025)</p>	<p><b>Operand Data Areas</b></p> <p><b>Cp<sub>1</sub>: 1<sup>st</sup> compare word</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>Cp<sub>2</sub>: 2<sup>nd</sup> compare word</b> CIO, G, A, T, C, #, DM, DR, IR</p>
--	--

**Description**

When the execution condition is OFF, EQU(025) is not executed. When the execution condition is ON, EQU(025) compares the content of Cp<sub>1</sub> to the content of Cp<sub>2</sub> and creates an ON execution condition if the two values are equal or it creates an OFF execution condition if they are not equal.

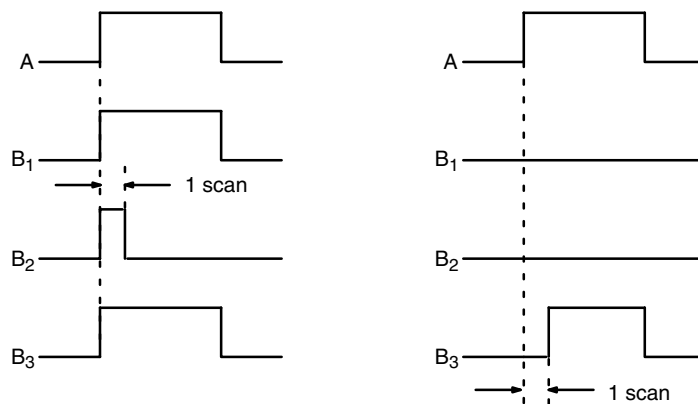
EQU(025) is an intermediate instruction, like NOT(010), CMP(020), and CMPL(021). Intermediate instructions are entered between conditions or between a condition and a right-hand instruction. Intermediate instructions cannot be placed at the end of an instruction line.

The up-differentiated version of EQU(025) behaves like the undifferentiated version when Cp<sub>1</sub> is equal to Cp<sub>2</sub> and will turn OFF for one scan only when Cp<sub>1</sub> is not equal to Cp<sub>2</sub>.



When Cp<sub>1</sub> is equal to Cp<sub>2</sub>.

When Cp<sub>1</sub> is not equal to Cp<sub>2</sub>.

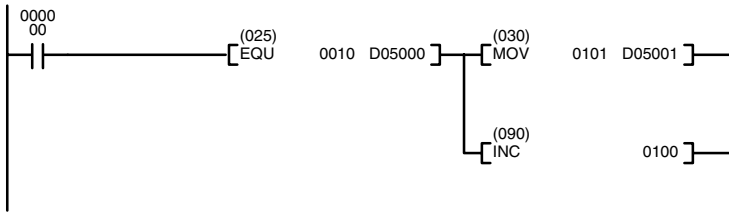


**Flags**

ER (A50003): Content of \*DM word is not BCD when set for BCD.

**Example**

When CIO 00000 is ON in the following example, the content of CIO 0010 is compared to the content of D05000 and MOV(030) and INC(090) are executed only if the contents are the same.



Address	Instruction	Operands
00000	LD	0000000
00001	EQU(025)	
		0010
		D05000
00002	MOV(030)	
		0101
		D05001
00003	INC(090)	
		0100

### 5-16-7 Input Comparison Instructions (300 to 328)

Ladder Symbol	Operand Data Areas
	<p><b>S<sub>1</sub>: Comparison data 1</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>S<sub>2</sub>: Comparison data 2</b> CIO, G, A, T, C, #, DM, DR, IR</p>

**Description**

When the execution condition is OFF, input comparison instructions are not executed and execution continues to the remainder of the instruction line. When the execution is ON, input comparison instructions compare constants and/or the contents of specified words for either signed or unsigned data and will create an ON execution condition when the comparison condition is met. If the comparison condition is not met, the remainder of the instruction line will be skipped and execution will move to the next instruction line.

A total of 24 input comparison instructions are available. These can be input using various combinations of symbols and options. If no options are specified, the comparison will be for one-word unsigned data.

Symbol	Option (data format)	Option (data length)
= (Equal)	S (signed data)	L (double length)
< > (Not equal)		
< (Less than)		
<= (Less than or equal)		
> (Greater than)		
>= (Greater than or equal)		

Unsigned input comparison instructions (i.e., instructions without the S option) can handle unsigned binary or BCD data. Signed input comparison instructions (i.e., instructions with the S option) handle signed binary data. For information concerning signed binary data, refer to 4-12 Data Formats.

When using input comparison instructions, follow each input comparison instruction in the program with another instruction on the same instruction line.

The following table shows the function codes, mnemonics, names, and functions of the input comparison instructions.

Code	Mnemonic	Name	Function
300	=	EQUAL	TRUE WHEN S <sub>1</sub> = S <sub>2</sub>
301	=L	DOUBLE EQUAL	
302	=S	SIGNED EQUAL	
303	=SL	DOUBLE SIGNED EQUAL	

Code	Mnemonic	Name	Function
305	<>	NOT EQUAL	TRUE WHEN $S_1 \neq S_2$
306	<>L	DOUBLE NOT EQUAL	
307	<>S	SIGNED NOT EQUAL	
308	<>SL	DOUBLE SIGNED NOT EQUAL	
310	<	LESS THAN	TRUE WHEN $S_1 < S_2$
311	<L	DOUBLE LESS THAN	
312	<S	SIGNED LESS THAN	
313	<SL	DOUBLE SIGNED LESS THAN	
315	<=	LESS THAN OR EQUAL	TRUE WHEN $S_1 \leq S_2$
316	<=L	DOUBLE LESS THAN OR EQUAL	
317	<=S	SIGNED LESS THAN OR EQUAL	
318	<=SL	DOUBLE SIGNED LESS THAN OR EQUAL	
320	>	GREATER THAN	TRUE WHEN $S_1 > S_2$
321	>L	DOUBLE GREATER THAN	
322	>S	SIGNED GREATER THAN	
323	>SL	DOUBLE SIGNED GREATER THAN	
325	>=	GREATER THAN OR EQUAL	TRUE WHEN $S_1 \geq S_2$
326	>=L	DOUBLE GREATER THAN OR EQUAL	
327	>=S	SIGNED GREATER THAN OR EQUAL	
328	>=SL	DOUBLE SIGNED GREATER THAN OR EQUAL	

**Precautions**

Input comparison instructions cannot be used as right-hand instructions, i.e., another instruction must be used between them and the right bus bar.

**Note** Refer to page 101 for general precautions on operand data areas.

**Example****< (310)**

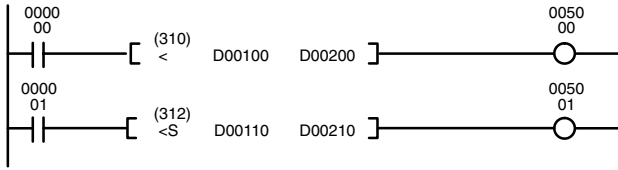
When CIO 000000 is ON in the following example, the contents of D00100 and D00200 are compared in as binary data. If the contents of D00100 is less than that of D00200, CIO 005000 is turned ON and execution proceeds to the next line. If the content of D00100 is not less than that of D00200, the remainder of the instruction line is skipped and execution moves to the next instruction line.

When CIO 000000 is OFF, CIO 005000 is turned OFF.

**<S(312)**

When CIO 000001 is ON in the following example, the contents of D00110 and D00210 are compared as binary data. If the content of D00110 is less than that of D00210, CIO 005001 is turned ON and execution proceeds to the next line. If the content of D00110 is not less than that of D00210, the remainder of the instruction line is skipped and execution moves to the next instruction line.

When CIO 000001 is OFF, CIO 005001 is turned OFF.



Address	Instruction	Operands
00000	LD	000000
00001	<(310)	
		D00100
		D00200
00002	LD	000001
00003	<S(312)	
		D00110
		D00210

**Comparison without sign (<)**

S<sub>1</sub>: D00100

S<sub>2</sub>: D00200

8714

3A1C

Decimal: 34580

Decimal: 14876

34580 > 14876

(Will not proceed to next line.)

**Comparison with sign (<S)**

S<sub>1</sub>: D00110

S<sub>2</sub>: D00210

8714

3A1C

Decimal: -30956

Decimal: 14876

-30956 > 14876

(Will proceed to next line.)

### 5-16-8 SIGNED BINARY COMPARE: CPS(026)

Ladder Symbol	Operand Data Areas
	<p><b>S<sub>1</sub>: Comparison word 1</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>S<sub>2</sub>: Comparison word 2</b> CIO, G, A, T, C, #, DM, DR, IR</p>
<p><b>Variations</b></p> <p>! CPS(026)</p>	

#### Description

When the execution condition is OFF, CPS(026) is not executed. When the execution condition is ON, CPS(026) compares constants and/or the contents of specified words as signed 16-bit binary data and changes the status of comparison flags according to the results.

After CPS(026) execution, the A50005 (GR), A50006 (EQ), and A50007 (LE) Flags will turn ON and OFF as shown in the following table.

Comparison result	A50005 (GR)	A50006 (EQ)	A50007 (LE)
S <sub>1</sub> > S <sub>2</sub>	ON	OFF	OFF
S <sub>1</sub> = S <sub>2</sub>	OFF	ON	OFF
S <sub>1</sub> < S <sub>2</sub>	OFF	OFF	ON

The range that can be specified for comparison is 8000 to 7FFF (i.e., -32,768 to 32,767 in decimal).

**Note** The immediate-refresh variation of CPS(026) can be used in simplex mode only; it cannot be used in duplex mode.

#### Flags

ER (A50003): Content of \*DM word is not BCD when set for BCD.

GR (A40213), EQ (A50006), LE (A50007): (Refer to tables above.)

### 5-16-9 DOUBLE SIGNED BINARY COMPARE: CPSL(027)

Ladder Symbol	Operand Data Areas
	<b>S<sub>1</sub>: First comparison word 1</b> CIO, G, A, T, C, #, DM, <b>S<sub>2</sub>: First comparison word 2</b> CIO, G, A, T, C, #, DM,

**Description**

When the execution condition is OFF, CPSL(027) is not executed. When the execution condition is ON, CPSL(027) compares constants and/or the contents of specified sets of words as signed 32-bit binary data and changes the status of comparison flags according to the results. The content of S<sub>1</sub> and S<sub>1</sub>+1 is compared to that of S<sub>2</sub> and S<sub>2</sub>+1.

After CPSL(027) execution, the A50005 (GR), A50006 (EQ), and A50007 (LE) flags turn ON and OFF as shown in the following table.

Comparison result	A50005 (GR)	A50006 (EQ)	A50007 (LE)
S <sub>1</sub> + 1, S <sub>1</sub> > S <sub>2</sub> + 1, S <sub>2</sub>	ON	OFF	OFF
S <sub>1</sub> + 1, S <sub>1</sub> = S <sub>2</sub> + 1, S <sub>2</sub>	OFF	ON	OFF
S <sub>1</sub> + 1, S <sub>1</sub> < S <sub>2</sub> + 1, S <sub>2</sub>	OFF	OFF	ON

The range that can be specified for comparison is 80000000 to 7FFFFFFF (i.e., -2,147,483,648 to 2,147,483,647 in decimal).

**Flags**

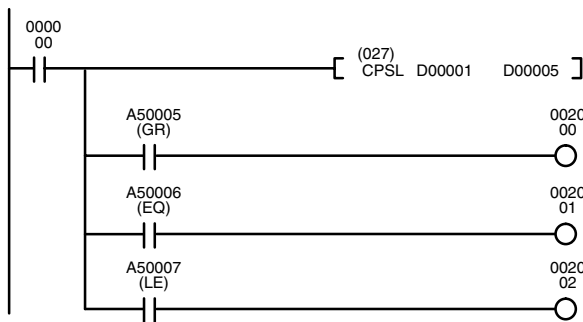
ER (A50003): Content of \*DM word is not BCD when set for BCD.

GR (A40213), EQ (A50006), LE (A50007): (Refer to tables above.)

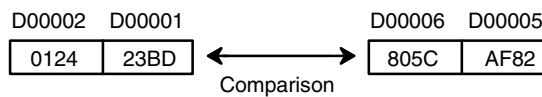
**Example**

When CIO 000000 is ON in the following example, the content of D00002 and D00001 is compared with the content of D00006 and D00005 as signed binary data.

- If the content of D00002 and D00001 is greater than that of D00006 and D00005, then A50005 (GR) will turn ON, causing CIO 002000 to be turned ON.
- If the content of D00002 and D00001 is equal to that of D00006 and D00005, then A50006 (EQ) will turn ON, causing CIO 002001 to be turned ON.
- If the content of D00002 and D00001 is less than that of D00006 and D00005, then A50007 (LE) will turn ON, causing CIO 002002 to be turned ON.



Address	Instruction	Operands
00000	LD	000000
00001	CPSL(027)	
		D00001
		D00005
00002	AND	A50005
00003	OUT	002000
00004	AND	A50006
00005	OUT	002001
00006	AND	A50007
00007	OUT	002002



**Comparison Results**

A50005 (GR)	ON
A50006 (EQ)	OFF
A50007 (LE)	OFF

### 5-16-10 UNSIGNED COMPARE: CMP(028)

<p><b>Ladder Symbol</b></p> <p><b>Variations</b> ! CMP(028)</p>	<p><b>Operand Data Areas</b></p> <p><b>S<sub>1</sub>: Comparison word 1</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>S<sub>2</sub>: Comparison word 2</b> CIO, G, A, T, C, #, DM, DR, IR</p>
---	--

**Description**

When the execution condition is OFF, CMP(028) is not executed. When the execution condition is ON, CMP(028) compares constants and/or the contents of specified words as unsigned 16-bit binary data and changes the status of comparison flags according to the results.

After CMP(028) execution, the A50005 (GR), A50006 (EQ), and A50007 (LE) flags turn ON and OFF as shown in the following table.

Comparison result	A50005 (GR)	A50006 (EQ)	A50007 (LE)
$S_1 > S_2$	ON	OFF	OFF
$S_1 = S_2$	OFF	ON	OFF
$S_1 < S_2$	OFF	OFF	ON

**Note** The immediate-refresh variation of CMP(028) can be used in simplex mode only; it cannot be used in duplex mode.

**Flags**

ER (A50003): Content of \*DM word is not BCD when set for BCD.

GR (A40213), EQ (A50006), LE (A50007): (Refer to tables above.)

### 5-16-11 DOUBLE UNSIGNED COMPARE: CMPL(029)

<p><b>Ladder Symbol</b></p>	<p><b>Operand Data Areas</b></p> <p><b>S<sub>1</sub>: First comparison word 1</b> CIO, G, A, T, C, #, DM,</p> <p><b>S<sub>2</sub>: First comparison word 2</b> CIO, G, A, T, C, #, DM,</p>
-----------------------------	--

**Description**

When the execution condition is OFF, CMPL(029) is not executed. When the execution condition is ON, CMPL(029) compares constants and/or the contents of specified sets of words as unsigned 32-bit data and changes the status of comparison flags according to the results. The content of S<sub>1</sub> and S<sub>1</sub>+1 is compared to that of S<sub>2</sub> and S<sub>2</sub>+1.

After CMPL(029) execution, the A50005 (GR), A50006 (EQ), and A50007 (LE) flags turn ON and OFF as shown in the following table.

Comparison result	A50005 (GR)	A50006 (EQ)	A50007 (LE)
$S_1 + 1, S_1 > S_2 + 1, S_2$	ON	OFF	OFF
$S_1 + 1, S_1 = S_2 + 1, S_2$	OFF	ON	OFF
$S_1 + 1, S_1 < S_2 + 1, S_2$	OFF	OFF	ON

**Flags**

ER (A50003): Content of \*DM word is not BCD when set for BCD.

GR (A40213), EQ (A50006), LE (A50007): (Refer to tables above.)

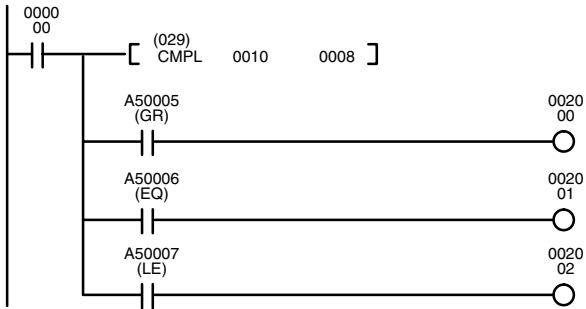
**Example**

When CIO 000000 is ON in the following example, the content of CIO 0011 and CIO 0010 is compared with that of CIO 0009 and CIO 0008 as binary data.

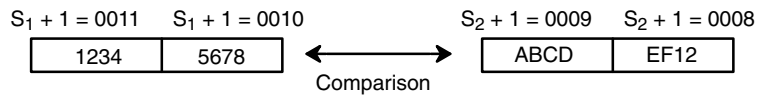
- If the content of CIO 0011 and CIO 0010 is greater than that of CIO 0009 and CIO 0008, then output CIO 002000 will turn ON.



- If the content of CIO 0011 and CIO 0010 is equal to that of CIO 0009 and CIO 0008, then output CIO 002001 will turn ON.
- If the content of CIO 0011 and CIO 0010 is less than that of CIO 0009 and CIO 0008, then output CIO 002002 will turn ON.



Address	Instruction	Operands
00000	LD	000000
00001	CMPL(029)	
		0010
		0008
00002	AND	A50005
00003	OUT	002000
00004	AND	A50006
00005	OUT	002001
00006	AND	A50007
00007	OUT	002002



**Comparison Results**

A50005 (GR)	OFF
A50006 (EQ)	OFF
A50007 (LE)	<b>ON</b>

## 5-17 Conversion Instructions

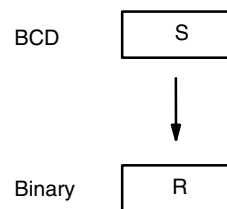
The Conversion Instructions convert word data that is in one format into another format and output the converted data to specified result word(s). All of these instructions change only the content of the words to which converted data is being moved, i.e., the content of source words is the same before and after execution of any of the conversion instructions. Refer to 5-26 *Time Instructions* for instructions that convert between different time formats.

### 5-17-1 BCD-TO-BINARY: BIN(100)

<p><b>Ladder Symbol</b></p> <p><b>Variations</b></p> <p>↑ BIN(100)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: Source word</b> CIO, G, A, T, C, DM, DR, IR</p> <p><b>R: Result word</b> CIO, G, A, DM, DR, IR</p>
--	--

**Description**

When the execution condition is OFF, BIN(100) is not executed. When the execution condition is ON, BIN(100) converts the BCD content of S into the numerically equivalent binary bits, and outputs the binary value to R. Only the content of R is changed; the content of S is left unchanged.



BIN(100) can be used to convert BCD to binary so that displays on Programming Devices will appear in hexadecimal rather than decimal. It can also be used to convert to binary to perform binary arithmetic operations rather than BCD arithmetic operations, e.g., when BCD and binary values must be added.

**Precautions**

S must be BCD.

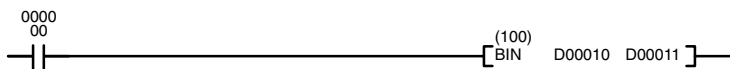
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

- ER (A50003): S or content of \*DM word is not BCD when set for BCD.
- EQ (A50006): 0 has been placed in R
- N (A50008): OFF

**Example**

When CIO 000000 is ON in the following example, the content of D00010 is converted from BCD to binary and stored into D00011. The content of D00010 is left unchanged.



Address	Instruction	Operands
00000	LD	000000
00001	BIN(100)	
		D00010
		D00011

	before execution	after execution
D00010	[ 4   0   9   5 ]	[ 4   0   9   5 ]
D00011	[ 0   0   0   0 ]	[ 0   F   F   F ]

The bit contents of words D00010 and D00011 after execution are:

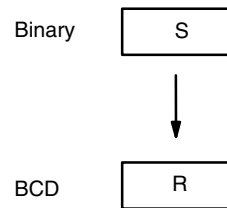
	4	0	9	5
D00010	0	1	0	0
	0	0	0	0
	1	0	0	1
	0	1	0	1
	0	1	0	1
	0	1	0	1
D00011	0	F	F	F
	0	0	0	0
	1	1	1	1
	1	1	1	1
	1	1	1	1
	1	1	1	1
	1	1	1	1
	1	1	1	1

### 5-17-2 BINARY-TO-BCD: BCD(101)

<p><b>Ladder Symbol</b></p> <p><b>Variations</b></p> <p>↑ BCD(101)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: Source word</b> CIO, G, A, T, C, DM, DR, IR</p> <p><b>R: Result word</b> CIO, G, A, DM, DR, IR</p>
--	--

**Description**

When the execution condition is OFF, BCD(101) is not executed. When the execution condition is ON, BCD(101) converts the binary (hexadecimal) content of S into the numerically equivalent BCD digits and outputs the BCD bits to R. Only the content of R is changed; the content of S is left unchanged.



BCD(101) can be used to convert binary to BCD so that displays on a Programming Device will appear in decimal rather than hexadecimal. It can also be used to convert to BCD to perform BCD arithmetic operations rather than binary arithmetic operations, e.g., when BCD and binary values must be added.

**Precautions**

If the content of S exceeds 270F, the converted result will exceed 9999, BCD(101) will not be executed, and the Error Flag (A50003) will be turned ON. When the instruction is not executed, the content of R remains unchanged.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

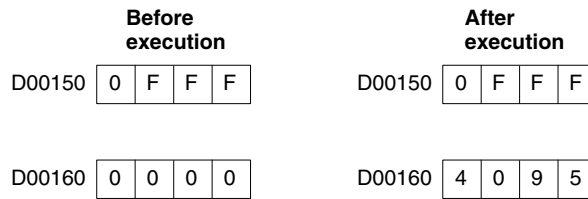
- ER (A50003): Content of \*DM word is not BCD when set for BCD.  
Content of S exceeds 270F.
- EQ (A50006): 0 has been placed in R

**Example**

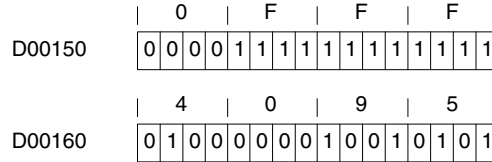
When CIO 000006 is ON in the following example, the content of word D00150 is converted from binary to BCD and stored in D00160. The content of D00150 is left unchanged.



Address	Instruction	Operands
00000	LD	000006
00001	BCD(101)	
		D00150
		D00160



The bit contents of words D00150 and D00160 after execution are:

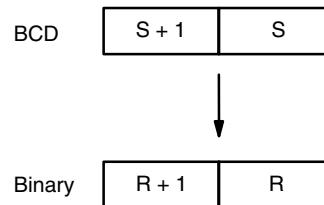


### 5-17-3 DOUBLE BCD-TO-DOUBLE BINARY: BINL(102)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;"> </p> <p><b>Variations</b></p> <p>↑ BINL(102)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: Source word</b> CIO, G, A, T, C, DM</p> <p><b>R: Result word</b> CIO, G, A, DM</p>
--	--

**Description**

When the execution condition is OFF, BINL(102) is not executed. When the execution condition is ON, BINL(102) converts an 8-digit BCD number in S and S+1 into 32-bit binary data, and outputs the converted data to R and R+1.



**Precautions**

S and S+1 must be BCD.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

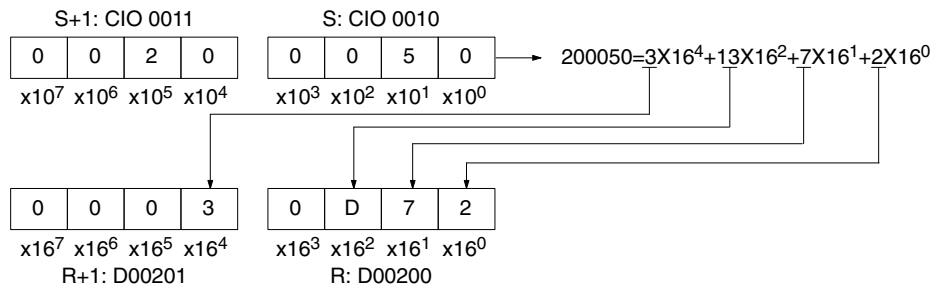
- ER (A50003): Content of S or S+1 is not BCD.  
Content of \*DM word is not BCD when set for BCD.
- EQ (A50006): 0 has been placed in R.
- N (A50008): OFF

**Example**

When CIO 000000 is ON in the following example, the BCD value in CIO 0010 and CIO 0010 is converted to a hexadecimal (binary) value and stored in D00200 and D00201.



Address	Instruction	Operands
00000	LD	000000
00001	BINL(102)	
		0010
		D00200

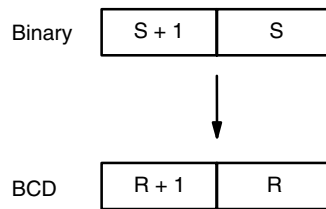


### 5-17-4 DOUBLE BINARY-TO-DOUBLE BCD: BCDL(103)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(103)  </p> <p><b>Variations</b></p> <p>↑ BCDL(103)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: Source word</b>    CIO, G, A, T, C, DM</p> <p><b>R: Result word</b>    CIO, G, A, DM</p>
--	--

**Description**

When the execution condition is OFF, BCDL(103) is not executed. When the execution condition is ON, BCDL(103) converts the 32-bit binary content of S and S+1 into eight digits of BCD data and outputs the converted data to R and R+1.



**Precautions**

If the content of S exceeds 05F5E0FF, the converted result will exceed 99999999, BCDL(103) will not be executed, and the Error Flag (A50003) will be turned ON. When the instruction is not executed, the content of R and R+1 remain unchanged.

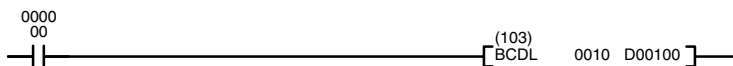
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

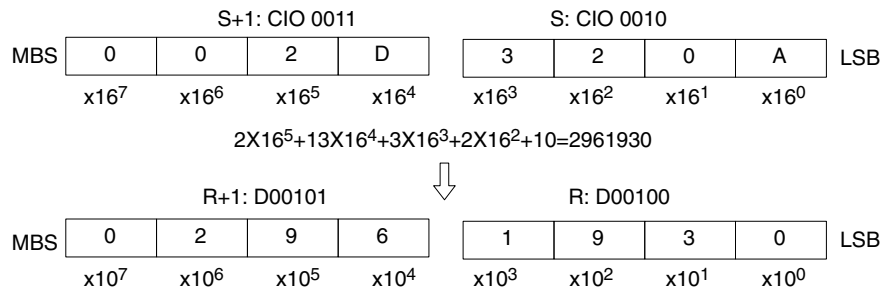
- ER (A50003): Content of \*DM word is not BCD when set for BCD.  
Content of S and S+1 exceeds 05F5E0FF.
- EQ (A50006): 0 has been placed in R

**Example**

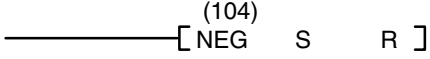
When CIO 000000 is ON in the following example, the hexadecimal value in CIO 0011 and CIO 0010 is converted to a BCD value and stored in D00200 and D00201.



Address	Instruction	Operands
00000	LD	000000
00001	BCDL(103)	
		0010
		D00100



### 5-17-5 2'S COMPLEMENT: NEG(104)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(104)  </p> <p><b>Variations</b></p> <p>↑ NEG(104)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: Source word</b>    CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>R: Result word</b>    CIO, G, A, DM, DR, IR</p>
--	---

**Description**

When the execution condition is OFF, NEG(104) is not executed. When the execution condition is ON, NEG(104) converts the 4-digit hexadecimal content of the source word (S) to its 2's complement and outputs the result to the result word (R). This operation is effectively the same as subtracting S from \$0000 and outputting the result to R.

**Precautions**

Refer to page 101 for general precautions on operand data areas.

**Flags**

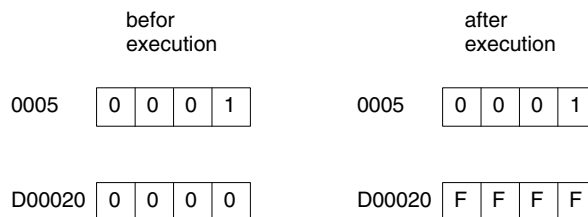
- ER (A50003): Content of \*DM word is not BCD when set for BCD.
- EQ (A50006): Content of S is 0 (the content of R will also be 0 after execution)
- N (A50008): Shows the status of bit 15 of R after execution.

**Example**

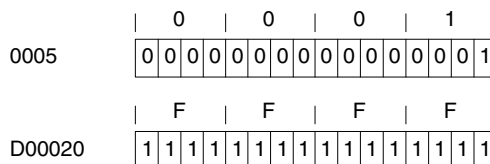
When CIO 000008 is ON in the following example, the 4-digit hexadecimal content of CIO 0005 is converted to its 2's complement equivalent and stored in D00020.



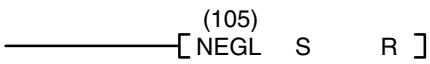
Address	Instruction	Operands
00000	LD	000008
00001	NEG(104)	
		0005
		D00020



The bit contents of word 0005 and word D00020 after execution is as follows.



### 5-17-6 DOUBLE 2'S COMPLEMENT: NEGL(105)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(105)  </p> <p><b>Variations</b></p> <p>↑ NEGL(105)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: 1<sup>st</sup> source word</b> CIO, G, A, T, C, #, DM</p> <p><b>R: 1<sup>st</sup> result word</b> CIO, G, A, DM</p>
---	---

**Description**

When the execution condition is OFF, NEGL(105) is not executed. When the execution condition is ON, NEGL(105) converts the 8-digit hexadecimal content of the source words (S and S+1) to its 2's complement and outputs the result to the result words (R and R+1). This operation is effectively the same as subtracting the 8-digit content S and S+1 from \$0000 0000 and outputting the result to R and R+1.

**Precautions**

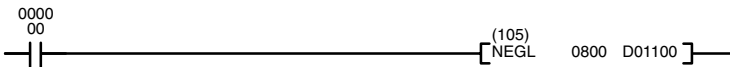
Refer to page 101 for general precautions on operand data areas.

**Flags**

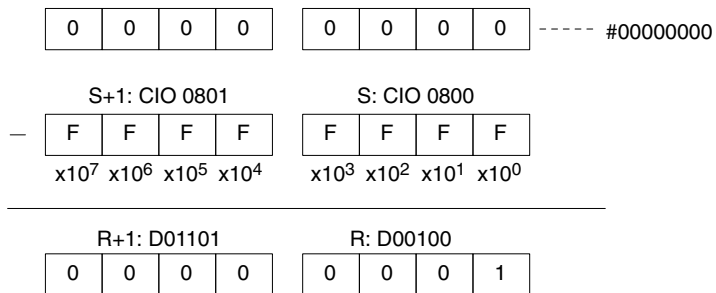
- ER (A50003): Content of \*DM word is not BCD when set for BCD.
- EQ (A50006): Content of S and S+1 is 0 (the content of R and R+1 will also be 0 after execution)
- N (A50008): Shows the status of bit 15 of R+1 after execution.

**Example**

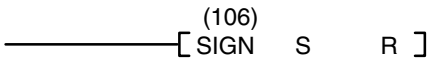
When CIO 000000 is ON in the following example, the 8-digit hexadecimal content of CIO 0000 and CIO 0001 is converted to its 2's complement equivalent and stored in D01100 and D01101.



Address	Instruction	Operands
00000	LD	000000
00001	NEGL(105)	
		0800
		D01100



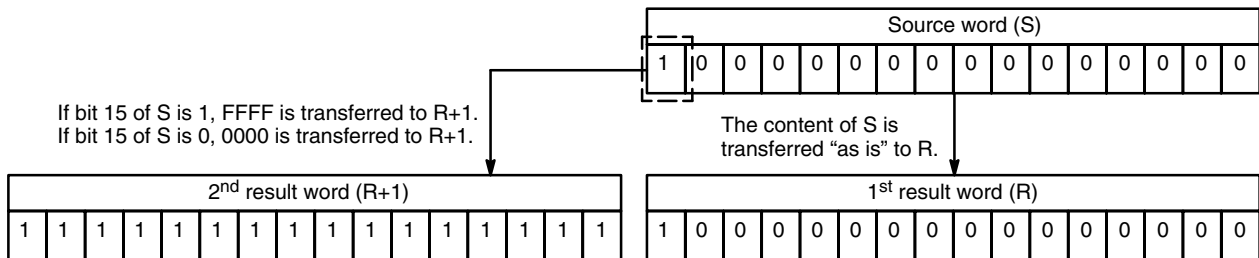
### 5-17-7 SIGN: SIGN(106)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(106)  </p> <p><b>Variations</b></p> <p>↑ SIGN(106)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: Source word</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>R: 1<sup>st</sup> result word</b> CIO, G, A, DM</p>
---	--

**Description**

When the execution condition is OFF, SIGN(106) is not executed. When the execution condition is ON, SIGN(106) copies the 4-digit signed binary source word (S) to R, extracts the sign from bit 15 of S, and outputs the result to R+1. If bit 15 of S is ON, \$FFFF is output to R+1, and if bit 15 of S is OFF, \$0000 is output to R+1. Refer to 3-2 Data Area Structure for details on signed data.

In the following example, the content of S is 8000, so 8000 is transferred to R and FFFF is transferred to R+1 because bit 15 of S is ON.



**Precautions**

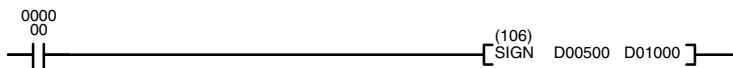
Refer to page 101 for general precautions on operand data areas.

**Flags**

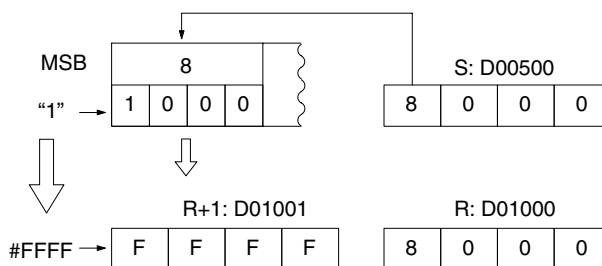
- ER (A50003): Content of \*DM word is not BCD when set for BCD.
- EQ (A50006): Content of R and R+1 is 0 after execution
- N (A50008): Content of R+1 is FFFF after execution.

**Example**

When the CIO 000000 is ON in the following example, the 4-digit signed content of D00500 is copied to D01000 and the sign from bit 15 of D00500 is output to bit 15 of D01001.



Address	Instruction	Operands
00000	LD	000000
00001	SIGN(106)	
		D00500
		D01000





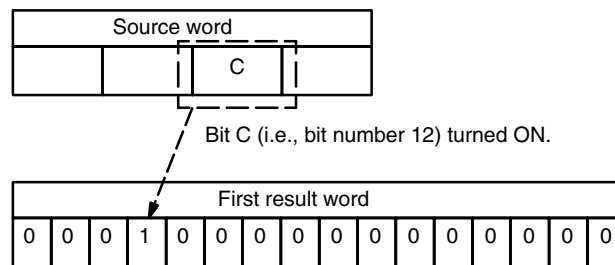
### 5-17-8 DATA DECODER: MLPX(110)

Ladder Symbol	Operand Data Areas
$\text{---} \left[ \begin{array}{c} (110) \\ \text{MLPX} \end{array} \text{ S } \quad \text{Di} \quad \text{R} \right]$	<b>S: Source word</b> CIO, G, A, T, C, DM, DR, IR <b>Di: Digit designator</b> CIO, G, A, T, C, #, DM, DR, IR <b>R: 1<sup>st</sup> result word</b> CIO, G, A, DM
<b>Variations</b> ↑ MLPX(110)	

**Description**

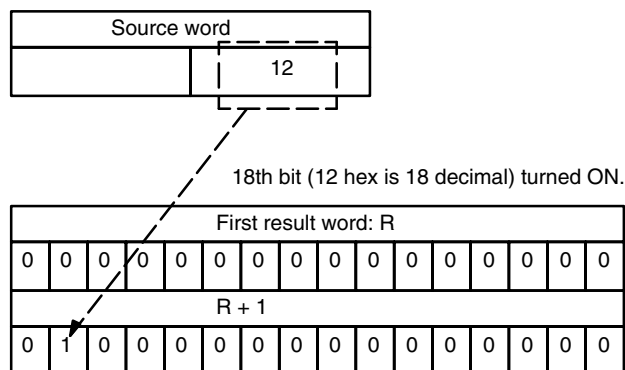
When the execution condition is OFF, MLPX(110) is not executed. When the execution condition is ON, MLPX(110) can be used to convert either 4-bit units or 8-bit units. The type of conversion used is specified in the leftmost bit of Di. For 4-bit conversion, MLPX(110) converts up to four 4-bit hexadecimal digits from S into decimal values from 0 to 15, each of which is used to indicate a bit position. The bit whose number corresponds to each converted value is then turned ON in a result word. If more than one digit is specified, then one bit will be turned ON in each of consecutive words beginning with R. (See examples, below.)

The following is an example of a one-digit decode operation from digit number 1 of S, i.e., here Di would be 0001.



For 8-bit conversion, MLPX(110) converts up to two 8-bit digits from S into decimal values from 0 to 255, each of which is used to indicate a bit position in consecutive result words. The bit corresponding to each converted value counting from the first result word is turned ON in the result words. If more than one digit is specified, then one bit will be turned ON in each set of consecutive words beginning with R. (See examples, below.)

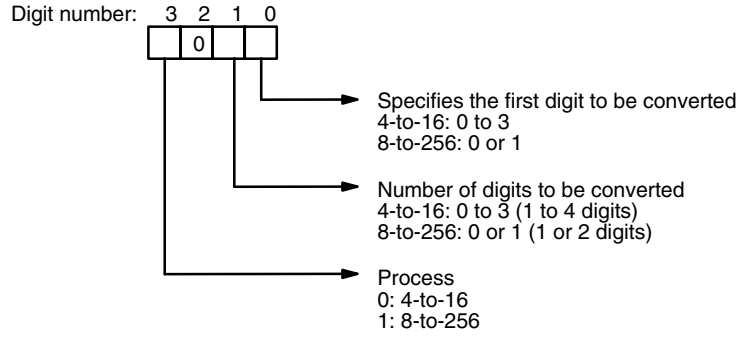
The following is an example of a one-digit decode operation from digit number 1 of S, i.e., here Di would be 1001.



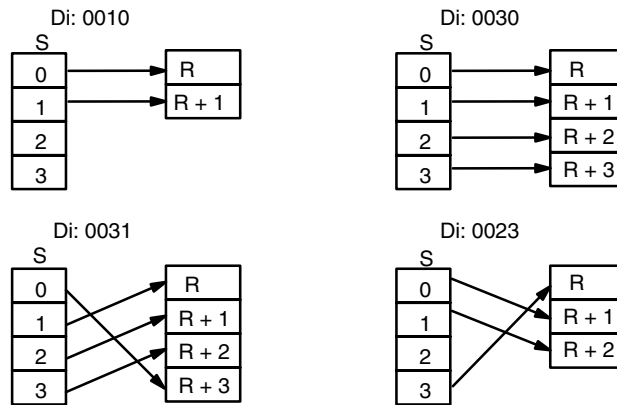
The first digit and the number of digits to be converted are designated in Di. If more digits are designated than remain in S (counting from the designated first digit), the remaining digits will be taken starting back at the beginning of S.

**Digit Designator**

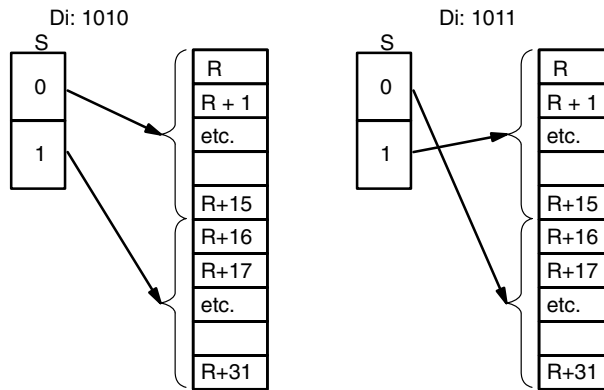
The digits of Di are set as shown below.



Some example Di values and the digit-to-word conversions that they produce are shown below for 4-bit conversion.



Some example Di values and the digit-to-word conversions that they produce are shown below for 8-bit conversion.



**Precautions**

The rightmost two digits of Di must each be between 0 and 3; the leftmost digit must be 0 to 1.

All result words must be in the same data area. MLPX(110) requires either 4 or 32 result words, depending on the type of conversion performed.

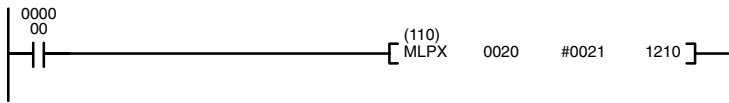
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

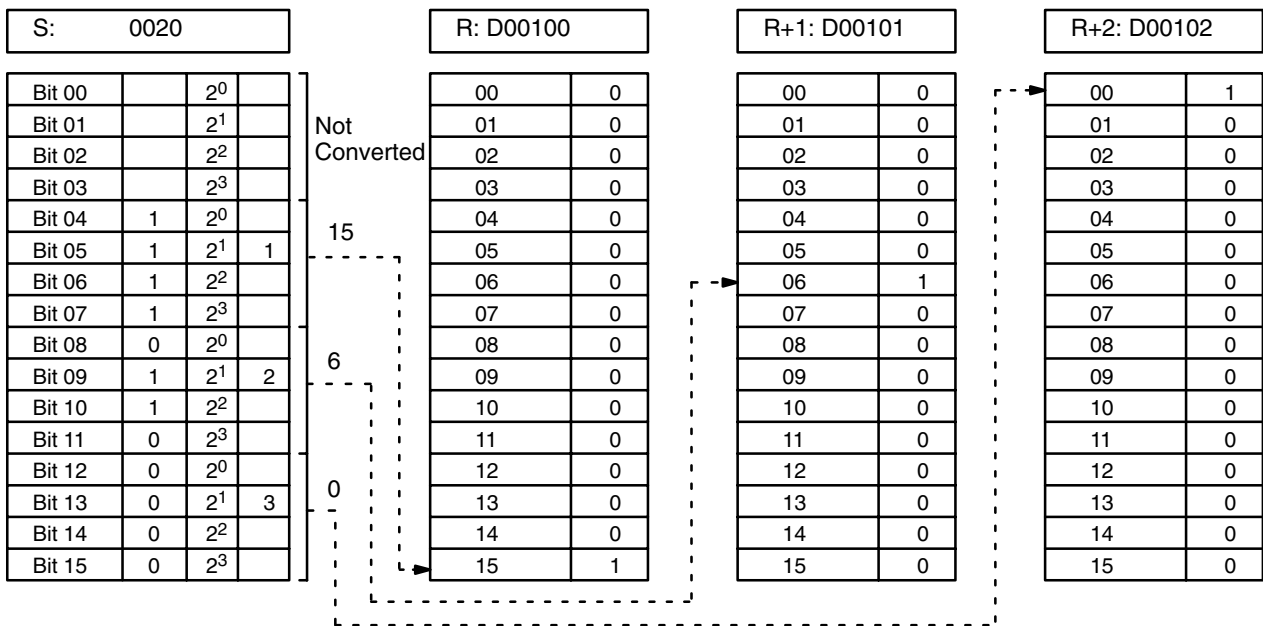
ER (A50003): Content of \*DM word is not BCD when set for BCD.  
Improper digit designator.

**Example**

When CIO 000000 is ON in the following example, three digits of data from CIO 0020 is converted to bit positions and the corresponding bits in three consecutive words starting with D 00100 are turned ON to indicate the position of the ON bits.



Address	Instruction	Operands
00000	LD	000000
00001	MLPX(110)	
		0200
		#0021
		1210



**5-17-9 DATA ENCODER: DMPX(111)**

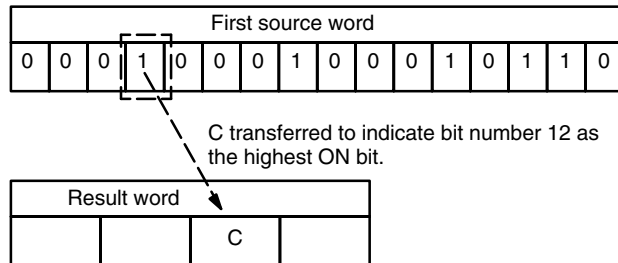
Ladder Symbol	Operand Data Areas
	<b>SB: 1<sup>st</sup> source word</b> CIO, G, A, T, C, DM <b>R: Result word</b> CIO, G, A, DM, DR, IR <b>Di: Digit designator</b> CIO, G, A, T, C, #, DM, DR, IR
<b>Variations</b> ↑ DMPX(111)	

**Description**

When the execution condition is OFF, DMPX(111) is not executed. When the execution condition is ON, DMPX(111) can be used to convert either 16-bit units (2 words) or 256-bit units (32 words). The type of conversion used is specified in the leftmost bit of Di.

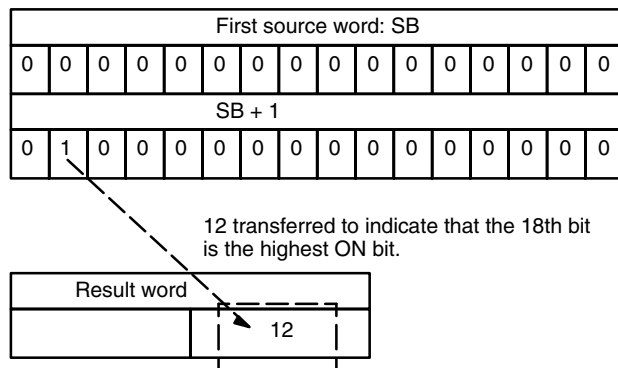
For 16-bit conversion, DMPX(111) determines the position of the highest ON bit in SB, encodes it into one-digit hexadecimal value corresponding to the bit number of the highest ON bit, then transfers the hexadecimal value to the specified 4-bit digit in R. The digits to receive the results are specified in Di, which also specifies the number of digits to be encoded.

The following is an example of a one-digit encode operation to digit number 1 of R, i.e., here Di would be 0001.



For 256-bit conversion, DMPX(111) determines the position of the highest ON bit in SB to SB+15, encodes it into two-digit hexadecimal value corresponding to the bit number of the highest ON bit, then transfers the hexadecimal value to the specified 8-bit digit in R. The first digit to receive the results is specified in Di, which also specifies the number of digits to be encoded.

The following is an example of a one-digit encode operation to digit number 1 of R, i.e., here Di would be 1001.

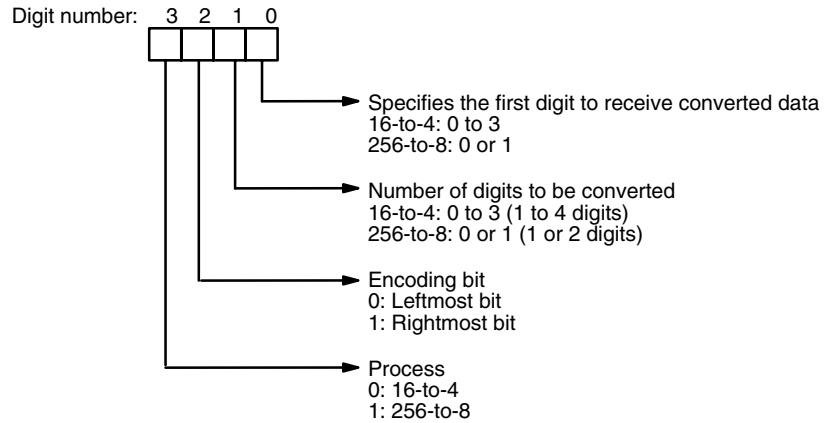


The number of digits (words) to be encoded and the first digit to receive converted data are specified in Di. If more digits are designated than remain in R (counting from the designated first digit), the remaining digits will be placed at digits starting back at the beginning of R.

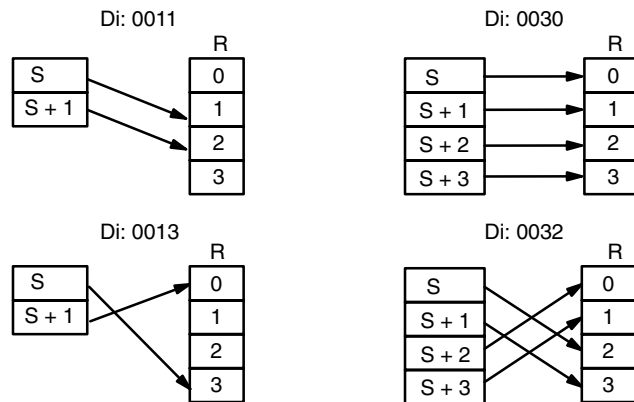
The final word(s) to be converted must be in the same data area as SB.

**Digit Designator**

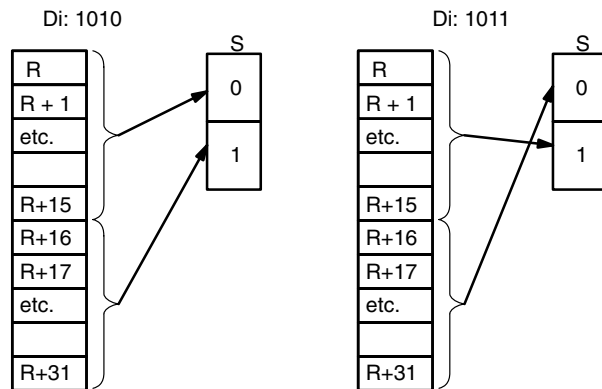
The digits of Di are set as shown below.



Some example Di values and the word-to-digit conversions that they produce are shown below for 16-bit conversion.



Some example Di values and the digit-to-word conversions that they produce are shown below for 256-bit conversion.



**Precautions**

The rightmost two digits of Di must each be between 0 and 3.  
 All source words must be in the same data area. DMPX(111) requires either 4 or 32 source words, depending on the type of conversion performed.

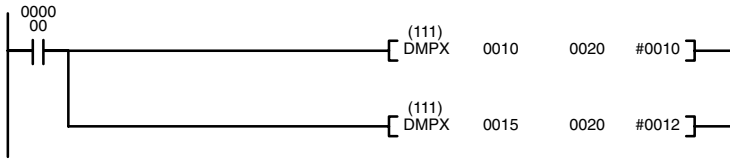
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

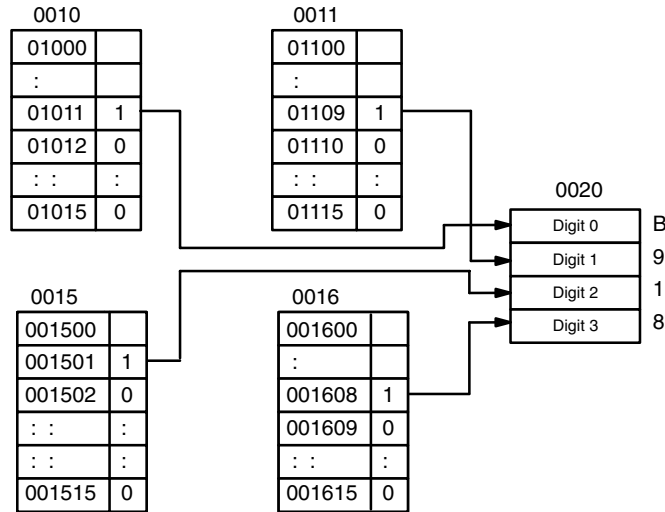
ER (A50003): Content of \*DM word is not BCD when set for BCD.  
 Improper digit designator.  
 Content of a source word is 0.

**Example**

When CIO 000000 is ON in the following example, the bit positions of the highest ON bits in CIO 0010 and 0011 are written to the first two digits of CIO 0020 and the bit positions of the highest ON bits in CIO 0015 and CIO 0016 are written to the last two digits of 0020.



Address	Instruction	Operands
00000	LD	000000
00001	DMPX(111)	
		0010
		0020
		#0010
00002	DMPX(111)	
		0015
		0020
		#0012



**5-17-107-SEGMENT DECODER: SDEC(112)**

Ladder Symbol	Operand Data Areas
	<b>S: Source word</b> CIO, G, A, T, C, DM, DR, IR <b>Di: Digit designator</b> CIO, G, A, T, C, #, DM, DR, IR <b>D: 1<sup>st</sup> destination word</b> CIO, G, A, DM
<b>Variations</b> ↑ SDEC(112)	

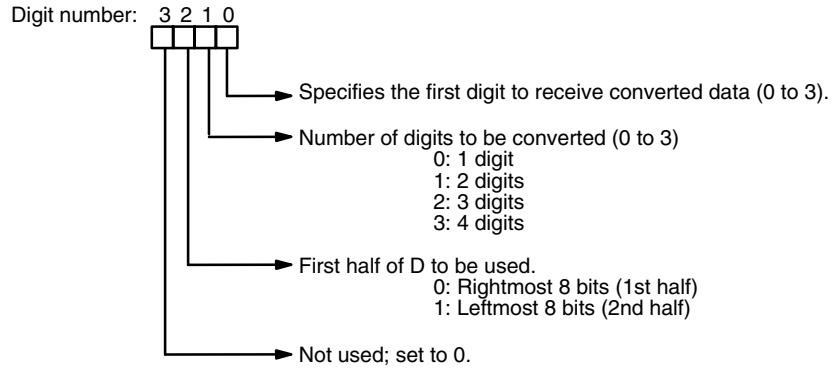
**Description**

When the execution condition is OFF, SDEC(112) is not executed. When the execution condition is ON, SDEC(112) converts the designated digit(s) of S into an 8-bit, 7-segment display code and places it into the destination word(s) beginning with D.

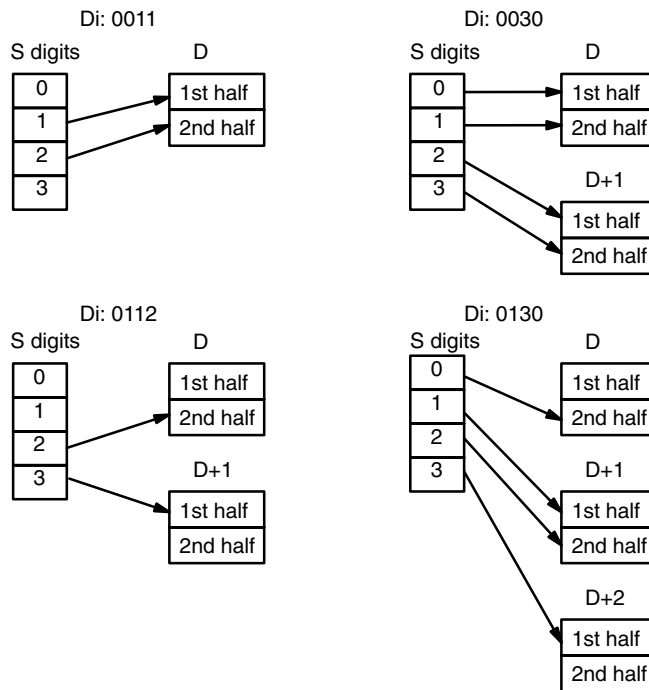
Any or all of the digits in S may be converted in sequence from the designated first digit. The first digit, the number of digits to be converted, and the half of D to receive the first 7-segment display code (rightmost or leftmost 8 bits) are designated in Di. If multiple digits are designated, they will be placed in order starting from the designated half of D, each requiring two digits. If more digits are designated than remain in S (counting from the designated first digit), further digits will be used starting back at the beginning of S.

**Digit Designator**

The digits of Di are set as shown below.



Some example Di values and the 4-bit binary to 7-segment display conversions that they produce are shown below.



**Precautions**

Di must be within the values given below.

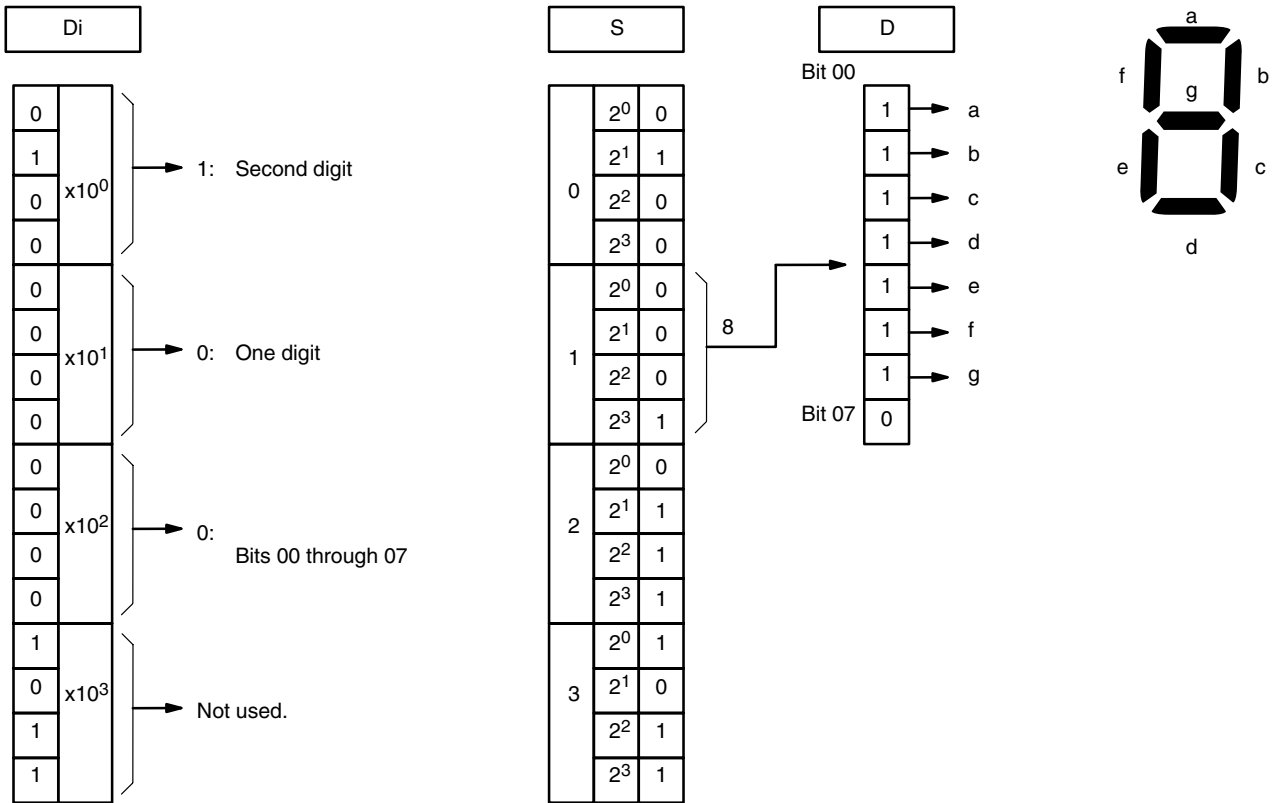
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

ER (A50003): Content of \*DM word is not BCD when set for BCD.  
Improper digit designator.

**Example**

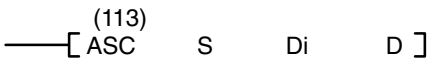
The following example shows the data to produce data for an “8.” The lower case letters show which bits correspond to which segments of the 7-segment display. The table underneath shows the original data and converted code for all hexadecimal digits.



Digit	Original data				Converted code (segments)								Display
	Bits	-	g	f	e	d	c	b	a				
0	0 0 0 0	0	0	1	1	1	1	1	1	1	1	0	
1	0 0 0 1	0	0	0	0	0	1	1	0	1	1	0	
2	0 0 1 0	0	1	0	1	1	0	1	1	0	1	1	
3	0 0 1 1	0	1	0	0	1	1	1	1	1	1	1	
4	0 1 0 0	0	1	1	0	0	1	1	0	1	1	0	
5	0 1 0 1	0	1	1	0	1	1	0	1	1	0	1	
6	0 1 1 0	0	1	1	1	1	1	1	1	1	0	1	
7	0 1 1 1	0	0	1	0	0	1	0	0	1	1	1	
8	1 0 0 0	0	1	1	1	1	1	1	1	1	1	1	
9	1 0 0 1	0	1	1	0	1	1	0	1	1	1	1	
A	1 0 1 0	0	1	1	1	0	1	1	0	1	1	1	
B	1 0 1 1	0	1	1	1	1	1	1	1	1	0	0	
C	1 1 0 0	0	0	1	1	1	0	0	1	0	0	1	
D	1 1 0 1	0	1	0	1	1	1	1	1	1	1	0	
E	1 1 1 0	0	0	1	1	1	1	0	0	0	0	1	
F	1 1 1 1	0	1	1	1	0	0	0	0	0	0	1	



### 5-17-11 ASCII CONVERT: ASC(113)

Ladder Symbol	Operand Data Areas
<p>(113)  </p> <p><b>Variations</b>            ↑ ASC(113)</p>	<p><b>S: Source word</b> CIO, G, A, T, C, DM, DR, IR</p> <p><b>Di: Digit designator</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>D: 1<sup>st</sup> destination word</b> CIO, G, A, DM</p>

**Description**

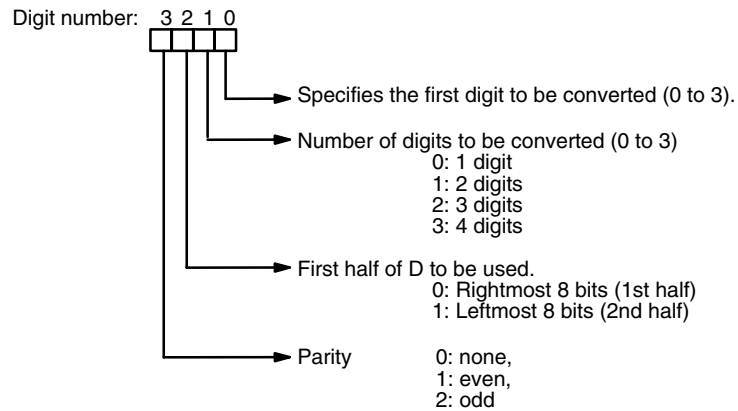
When the execution condition is OFF, ASC(113) is not executed. When the execution condition is ON, ASC(113) converts the designated digit(s) of S into the equivalent 8-bit ASCII code and places it into the destination word(s) beginning with D.

Any or all of the digits in S may be converted in order from the designated first digit. The first digit, the number of digits to be converted, and the half of D to receive the first ASCII code (rightmost or leftmost eight bits) are designated in Di. If multiple digits are designated, they will be placed in order starting from the designated half of D, each requiring two digits. If more digits are designated than remain in S (counting from the designated first digit), further digits will be used starting back at the beginning of S.

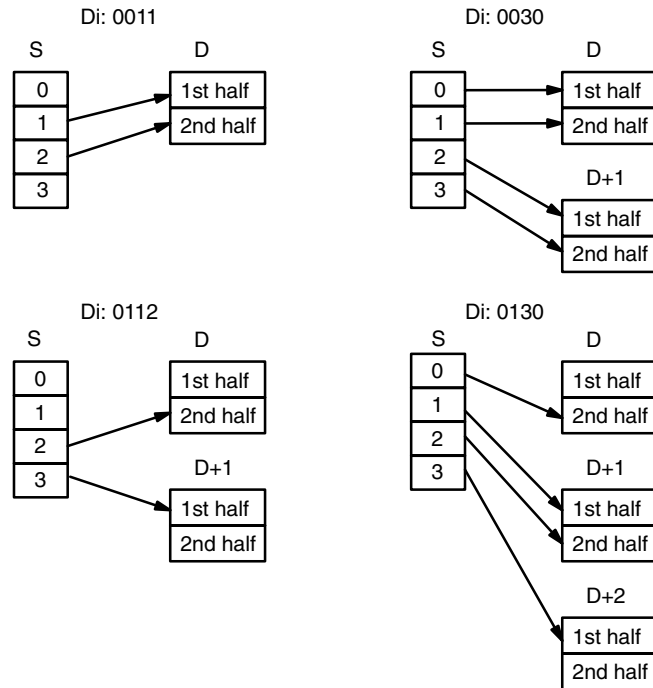
Refer to *Appendix H* for a table of extended ASCII characters.

**Digit Designator**

The digits of Di are set as shown below.



Some examples of Di values and the 4-bit binary to 8-bit ASCII conversions that they produce are shown below.



**Parity**

The leftmost bit of each ASCII character (2 digits) can be automatically adjusted for either even or odd parity. If no parity is designated, the leftmost bit will always be zero.

When even parity is designated, the leftmost bit will be adjusted so that the total number of ON bits is even, e.g., when adjusted for even parity, ASCII “31” (00110001) will be “B1” (10110001: parity bit turned ON to create an even number of ON bits); ASCII “36” (00110110) will be “36” (00110110: parity bit turned OFF because the number of ON bits is already even). The status of the parity bit does not affect the meaning of the ASCII code.

When odd parity is designated, the leftmost bit of each ASCII character will be adjusted so that there is an odd number of ON bits.

**Precautions**

Di must be within the values given below.

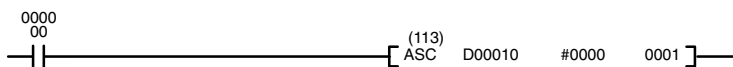
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

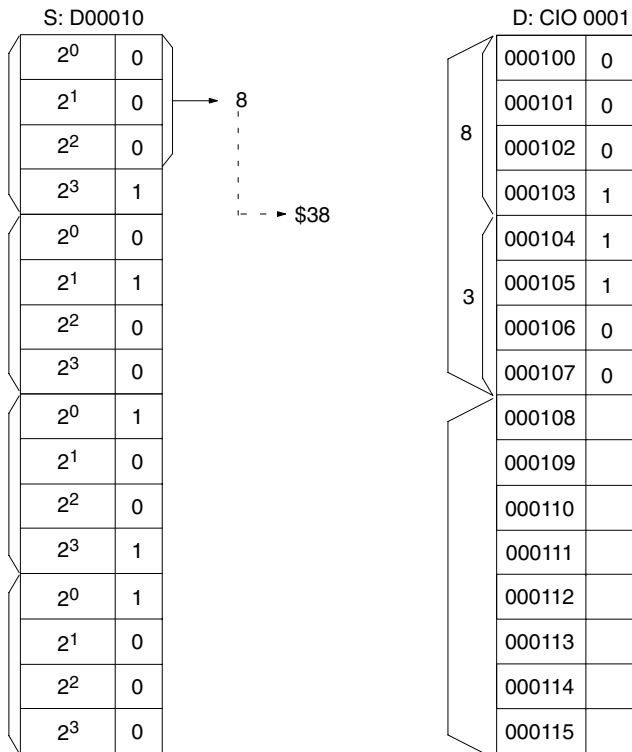
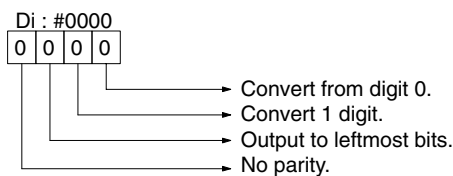
ER (A50003): Content of \*DM word is not BCD when set for BCD.  
Improper digit designator.

**Example**

When CIO 000000 is ON in the following example, the content of digit 0 in D00010 (8) is converted to ASCII(38) and output to the leftmost 8 bits of CIO 0001. “No parity” is specified, so CIO 000107 is set to 0.



Address	Instruction	Operands
00000	LD	000000
00001	ASC(113)	
		D00010
		#0000
		0001



### 5-17-12 BIT COUNTER: BCNT(114)

Ladder Symbol	Operand Data Areas
$\text{---} \overset{(114)}{\text{[ BCNT N S R ]}}$	<b>N: Number of words</b> CIO, G, A, T, C, #, DM, DR, IR <b>S: 1<sup>st</sup> source word</b> CIO, G, A, T, C, DM <b>R: Result word</b> CIO, G, A, T, C, DM, DR, IR
<b>Variations</b> ↑ BCNT(114)	

**Description** When the execution condition is OFF, BCNT(114) is not executed. When the execution condition is ON, BCNT(114) counts the total number of bits that are ON in all words between S and S+(N-1) and places the result in R.

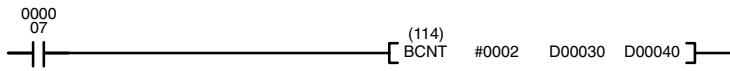
**Precautions** N must be BCD between 0001 and 9999.

**Note** Refer to page 101 for general precautions on operand data areas.

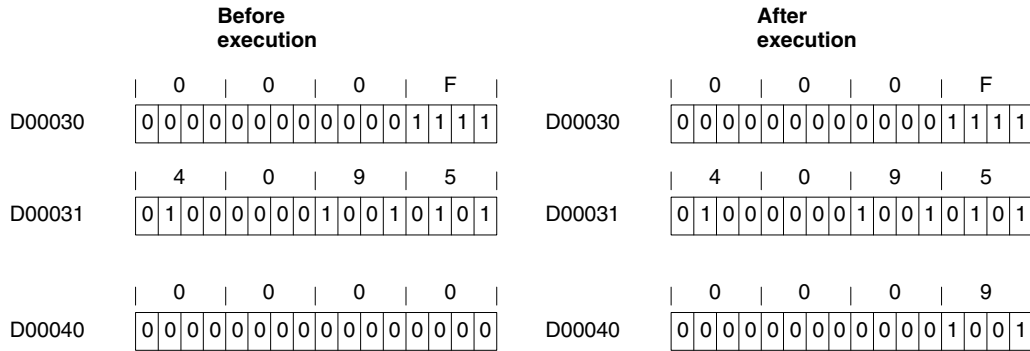
**Flags**  
 ER (A50003): N is not BCD between 0001 and 9999.  
 If the total exceeds 9999.  
 Content of \*DM word is not BCD when set for BCD.  
 EQ (A50006): Result is 0.

**Example**

When CIO 00007 is ON in the following example, all ON bits in D00030 and D00031 are counted and the results is placed in D00040.



Address	Instruction	Operands
00000	LD	000007
00001	BCNT(114)	
		#0002
		D00030
		D00040

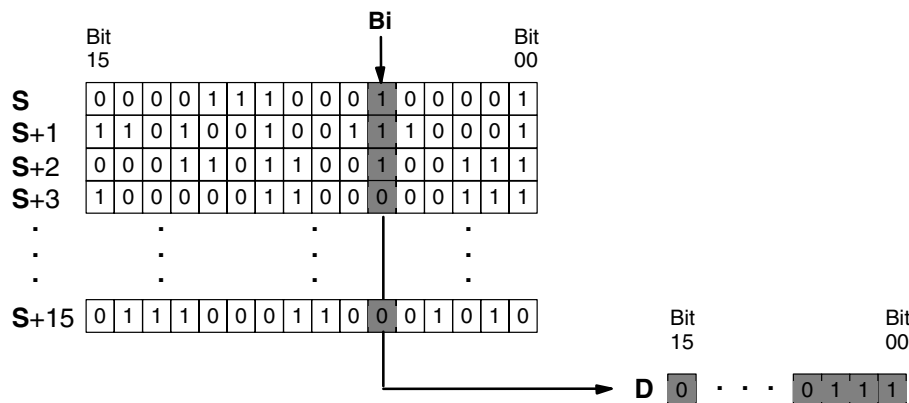


**5-17-13 COLUMN TO LINE: LINE(115)**

Ladder Symbol	Operand Data Areas
	<b>S: 1<sup>st</sup> source word</b> CIO, G, A, T, C, DM <b>Bi: Bit designator</b> CIO, G, A, T, C, #, DM, DR, IR <b>D: Destination word</b> CIO, G, A, DM, DR, IR
<b>Variations</b> ↑ LINE(115)	

**Description**

When the execution condition is OFF, LINE(115) is not executed. When the execution condition is ON, LINE(115) copies bit column Bi from the 16-word set S through S+15 to the 16 bits of word D (00 to 15), i.e., bit Bi of S+n is copied to bit n of D, for n=00 to 15. In the following example, Bi would be 5.



**Precautions**

S cannot be one of the last 15 words in a data area because it designates the first of 16 words.

Bi must be BCD between 0000 and 0015.

**Note** Refer to page 101 for general precautions on operand data areas.

- Flags**
- ER (A50003): Content of \*DM word is not BCD when set for BCD.  
The bit designator Bi is not BCD, or it is specifying a non-existent bit (i.e., bit specification must be between 00 and 15).
  - EQ (A50006): Content of D is 0 after execution

**Example** When CIO 000000 is ON in the following example, the status of bits number 08 in D00100 through D00115 are output in order to D00005, with the status of bit 08 in D00100 being output to bit 00 of D00005.

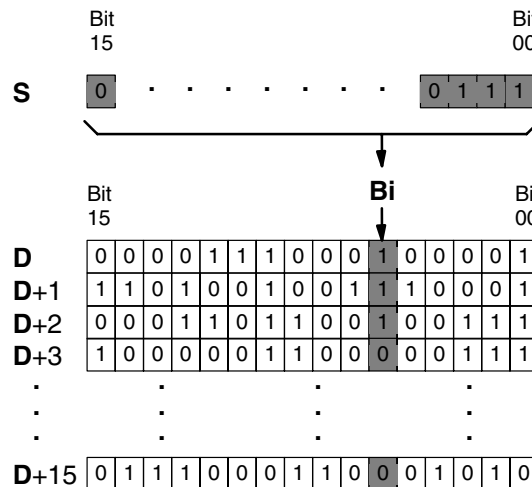


Address	Instruction	Operands
00000	LD	000000
00001	LINE(116)	
		D00100
		#0008
		D00005

### 5-17-14 LINE TO COLUMN: COLM(116)

Ladder Symbol	Operand Data Areas
	<p><b>S: Source word</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>D: 1<sup>st</sup> destination word</b> CIO, G, A, DM</p> <p><b>Bi: Bit designator</b> CIO, G, A, T, C, #(BCD), DM, DR, IR</p>
<p><b>Variations</b></p> <p>↑ COLM(116)</p>	

**Description** When the execution condition is OFF, COLM(116) is not executed. When the execution condition is ON, COLM(116) copies the 16 bits of word S (00 to 15) to bit Bi of the 16-word set D through D+15, i.e., bit n of S is copied to bit Bi of D+n, for n=00 to 15. In the following example, Bi would be 5.



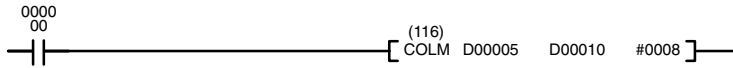
**Precautions** Bi must be BCD between 0000 and 0015.

**Note** Refer to page 101 for general precautions on operand data areas.

- Flags**
- ER (A50003): Content of \*DM word is not BCD when set for BCD.  
The bit designator Bi is not BCD, or it is specifying a non-existent bit (i.e., bit specification must be between 00 and 15).
  - EQ (A50006): Content of S is 0

**Example**

When CIO 000000 is ON in the following example, the status of bits 00 to 15 in D00005 are copied consecutively to bits number 08 of D00010 through D00025, with the status of bit 00 being transferred to bit 08 of D00010.



Address	Instruction	Operands
00000	LD	000000
00001	COLM(116)	
		D00005
		D00010
		#0008

### 5-17-15 ASCII TO HEX: HEX(117)

Ladder Symbol	Operand Data Areas
	<p><b>S: First source word</b> CIO, G, A, T, C, DM</p> <p><b>Di: Digit designator</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>D: Destination word</b> CIO, G, A, DM</p>
<p><b>Variations</b></p> <p>↑HEX(117)</p>	

**Description**

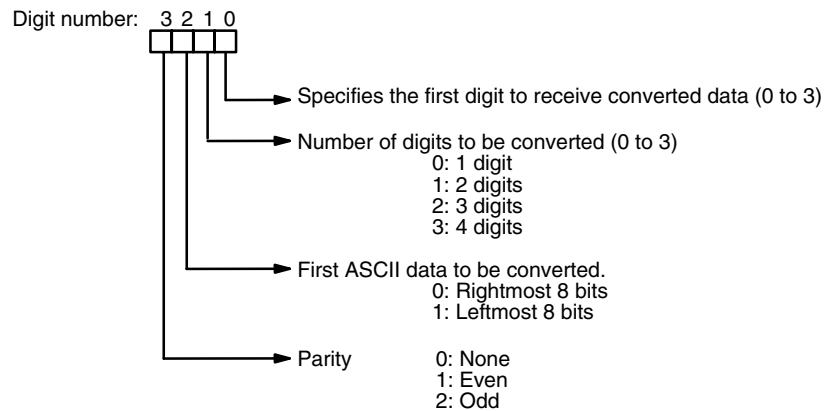
When the execution condition is OFF, HEX(117) is not executed. When the execution condition is ON, HEX(117) converts the data in specified source words from ASCII to hexadecimal data, and outputs the results to a specified destination word.

The ASCII range that can be converted is the numerals 0 through 9 (\$30 through \$39) and the capital letters A through F (\$41 through \$46).

If an attempt is made to convert other data, the Error Flag will turn ON and the instruction will not be executed.

The digit designator, Di, specifies the first digit to receive the converted data, the number of digits to be converted, the first ASCII data to be converted, and the parity (see below).

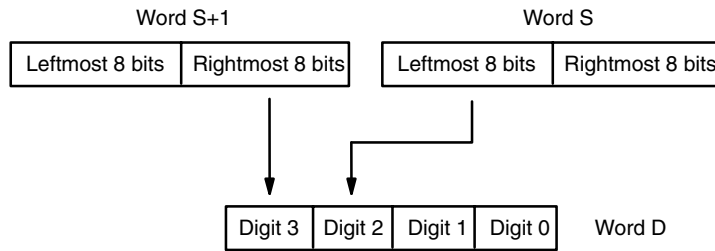
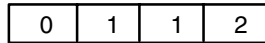
**Digit Designator**



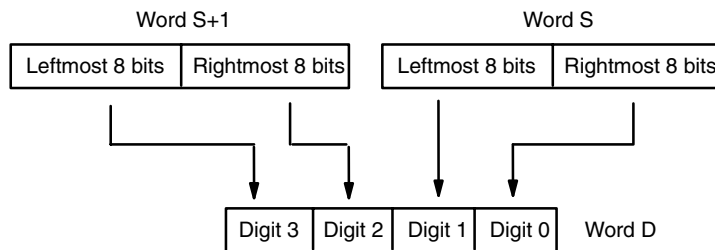
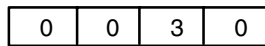
Data in the destination word (D) will not be changed except for the digits that are converted to hexadecimal.

**Digit Designator Examples** The following examples show the digit designators (Di) used to make various multiple-word conversions.

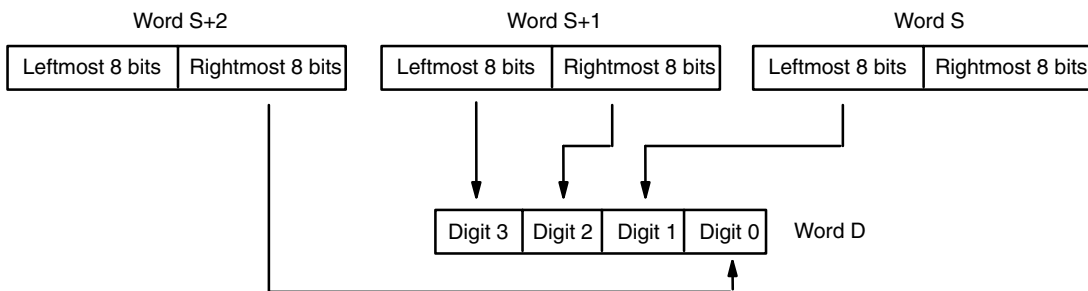
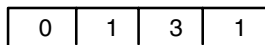
**Example 1** Di Word Contents



**Example 2** Di Word Contents



**Example 3** Di Word Contents



**Parity**

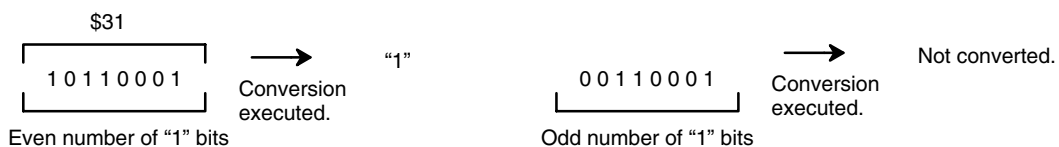
**0: None**

With no parity, data can only be converted when the leftmost bit is zero. If it is not set to zero, the Error Flag will turn ON and the data will not be converted.

**1: Even**

The data (8 bits) can only be converted when the number of "1" bits is even. If the number is odd, the Error Flag will turn ON and the data will not be converted.

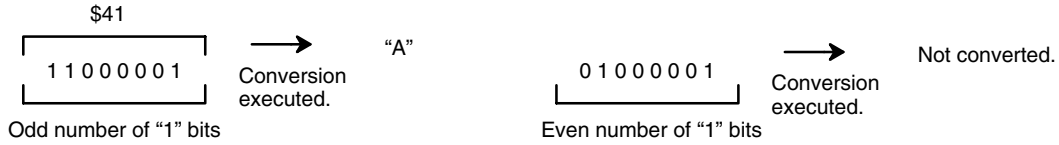
**Example**



**2: Odd**

The data (8 bits) can only be converted when the number of "1" bits is odd. If the number is even, the Error Flag will turn ON and the data will not be converted.

**Example**



**ASCII Code Table**

The following table shows the ASCII codes before conversion and the hexadecimal values after conversion. Refer to *Appendix I* for a table of ASCII characters.

Original data		Converted data											
ASCII Code	Bit status (See note.)								Digit	Bits			
\$30	P	0	1	1	0	0	0	0	0	0	0	0	
\$31	P	0	1	1	0	0	0	1	1	0	0	0	1
\$32	P	0	1	1	0	0	1	0	2	0	0	1	0
\$33	P	0	1	1	0	0	1	1	3	0	0	1	1
\$34	P	0	1	1	0	1	0	0	4	0	1	0	0
\$35	P	0	1	1	0	1	0	1	5	0	1	0	1
\$36	P	0	1	1	0	1	1	0	6	0	1	1	0
\$37	P	0	1	1	0	1	1	1	7	0	1	1	1
\$38	P	0	1	1	1	0	0	0	8	1	0	0	0
\$39	P	0	1	1	1	0	0	1	9	1	0	0	1
\$41	P	1	0	0	0	0	0	1	A	1	0	1	0
\$42	P	1	0	0	0	0	1	0	B	1	0	1	1
\$43	P	1	0	0	0	0	1	1	C	1	1	0	0
\$44	P	1	0	0	0	1	0	0	D	1	1	0	1
\$45	P	1	0	0	0	1	0	1	E	1	1	1	0
\$46	P	1	0	0	0	1	1	0	F	1	1	1	1

**Note** The leftmost bit of each ASCII code is adjusted for parity.

**Precautions**

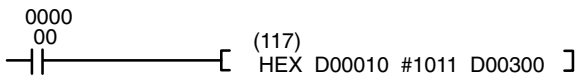
Refer to page 101 for general precautions on operand data areas.

**Flags**

ER (A50003): ASCII in S does not match parity designation.  
 Data in word S is not ASCII that can be converted.  
 Content of \*DM word is not BCD when set for BCD.

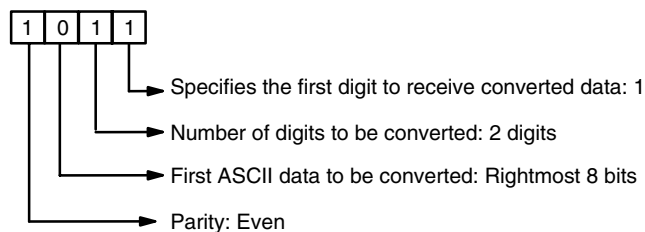
**Programming Example**

When CIO 000000 is ON in the following example, the ASCII data in D00010 is converted to binary data and then output to D00300.

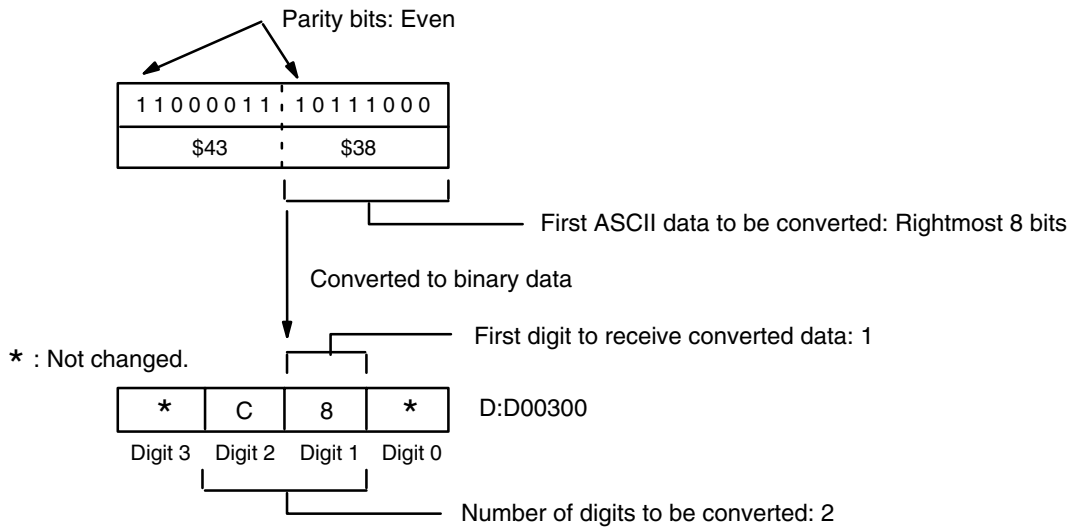


Address	Instruction	Operands
00000	LD	000000
00001	HEX(117)	
		D00010
		#1011
		D00300

**Di Word Contents: #1011**







### 5-17-16 SIGNED BCD-TO-BINARY: BINS(275)

Ladder Symbol	Operand Data Areas
<p>(275) [ BINS C S D ]</p>	<p><b>C: Control word</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>S: Source word</b> CIO, G, A, T, C, DM, DR, IR</p> <p><b>D: Destination word</b> CIO, G, A, DM, DR, IR</p>
<p><b>Variations</b></p> <p>↑BINS(275)</p>	

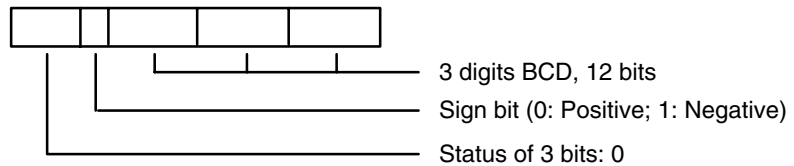
#### Description

When the execution condition is OFF, BINS(275) is not executed. When the execution condition is ON, BINS(275) converts the data in a specified source word (S) from signed BCD to signed binary, and outputs the result to a specified destination word (D). The format of the source word is determined by the contents of the control word (C).

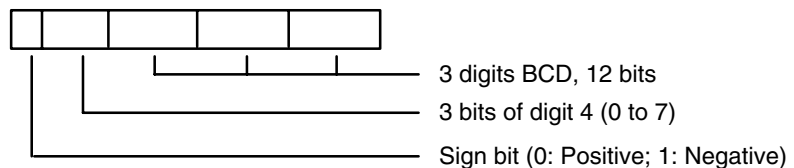
**Note** Special I/O Units sometimes output signed BCD data. Calculations using this data will normally be easier if it is first converted to signed binary data by means of BINS(275) or BISL(277).

The input data format and range designations for the various control word contents are as follows:

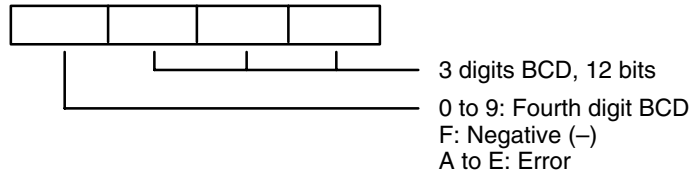
#### When C = 0000 (Input Data Range: -999 to 999 BCD)



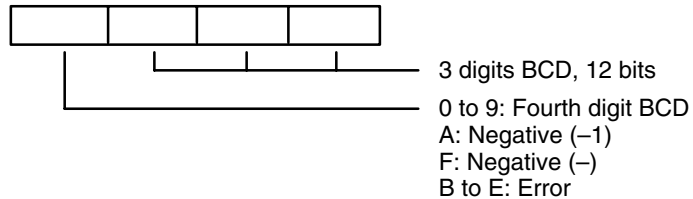
#### When C = 0001 (Input Data Range: -7999 to 7999 BCD)



**When C = 0002 (Input Data Range: -999 to 9999 BCD)**



**When C = 0003 (Input Data Range: -1999 to 9999 BCD)**



First the signed BCD data format and range in word S are checked against the data control word (C). If the check is okay, the signed BCD data in word S is converted to binary and output to the designated word D. If the format and range are not okay, the Error Flag (A50003) will turn ON and the instruction will not be executed.

In signed BCD data, -0 is treated as +0. When the data to be converted is a negative number, it will be output as 2's complement and the Negative Flag (A50008) will turn ON. In order to convert a 2's complement to the true value, it is necessary to subtract it from 0.

**Precautions**

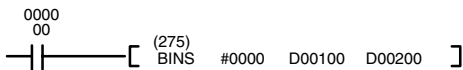
Refer to page 101 for general precautions on operand data areas.

**Flags**

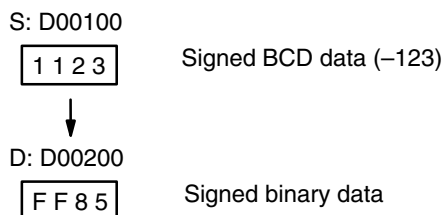
- ER (A50003): Data format is 0002, and leftmost digit is A to E.  
Data format is 0003, and the leftmost is B to E.  
Data to be converted is not BCD.  
Content of \*DM word is not BCD when set for BCD.
- EQ (A50006) Content of the converted data is all zeroes.
- N (A50008) Converted number is negative.

**Example 1**

When CIO 000000 is ON in the following example, first the signed BCD data format and range in D00100 are checked against data control word "0000" (first operand). If the check is okay, the signed BCD data in D00100 is converted to binary and output to D00200.

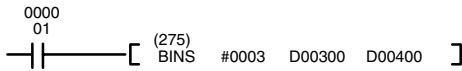


Address	Instruction	Operands
00000	LD	000000
00001	BINS(275)	
		#0000
		D00100
		D00200

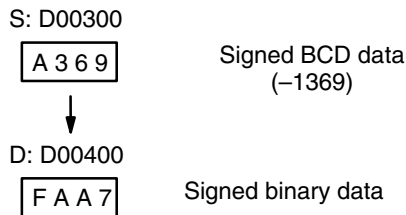


**Example 2**

When CIO 00001 is ON in the following example, first the signed BCD data format and range in D00300 are checked against data control word "0003" (first operand). If the check is okay, the signed BCD data in D00300 is converted to binary and output to D00400.



Address	Instruction	Operands
00000	LD	000001
00001	BINS(275)	
		#0003
		D00300
		D00400



**5-17-17 SIGNED BINARY-TO-BCD: BCDS(276)**

Ladder Symbol	Operand Data Areas
	<p><b>C: Control word</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>S: Source word</b> CIO, G, A, T, C, DM, DR, IR</p> <p><b>D: Destination word</b> CIO, G, A, DM, DR, IR</p>
<p><b>Variations</b></p> <p>↑BCDS(276)</p>	

**Description**

When the execution condition is OFF, BCDS(276) is not executed. When the execution condition is ON, BCDS(276) converts the data in a specified source word (S) from signed binary to signed BCD, and outputs the results to a specified destination word (C). The format of the destination word is determined by the contents of the control word (C).

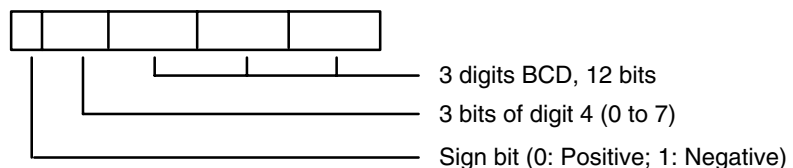
**Note** Special I/O Units sometimes require input of signed BCD data. BCDS(276) or BDSL(278) can be used to easily convert signed binary data to signed BCD data.

The output data format and range designations for the various control word contents are as follows:

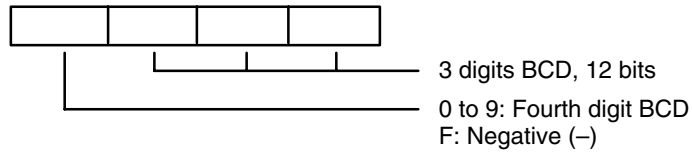
**When C = 0000 (Output Data Range: -999 to 999 BCD)**



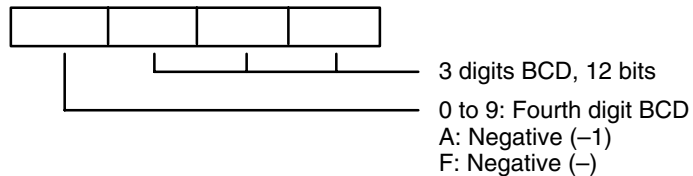
**When C = 0001 (Output Data Range: -7999 to 7999 BCD)**



**When C = 0002 (Output Data Range: -999 to 9999 BCD)**



**When C = 0003 (Output Data Range: -1999 to 9999 BCD)**



**Data Ranges**

The range of data that can be input or output is determined by the control word (0000 to 0003), as shown in the following table.

Data format	Input range (binary)	Output range (BCD)
0000	FFFF to FC19 0000 to 03E7	-999 to 999
0001	FFFF to F0C1 0000 to 1F3F	-7999 to 7999
0002	FFFF to FC19 0000 to 270F	-999 to 9999
0003	FFFF to F831 0000 to 270F	-1999 to 9999

First the signed binary data in word S is checked against the data control word (C). If the check is okay, the signed binary data in word S is converted to BCD and output to the designated word D. If the check is not okay, the Error Flag (A50003) will turn ON and the instruction will not be executed.

In signed BCD data, -0 is treated as +0.

**Precautions**

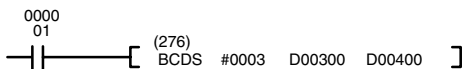
Refer to page 101 for general precautions on operand data areas.

**Flags**

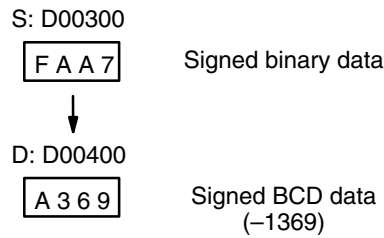
- ER (A50003): Data is not within allowable range for data format.  
Content of\*DM word is not BCD when set for BCD.
- EQ (A50006) Content of the converted data is all zeroes.
- N (A50008) Data to be converted is a negative number.

**Example**

When CIO 000001 is ON in the following example, first the signed binary data in D00300 is checked against data control word "0003" (first operand), and then the signed binary data in D00300 is converted to signed BCD and output to D00400.



Address	Instruction	Operands
00000	LD	000001
00001	BCDS(276)	
		#0003
		D00300
		D00400



### 5-17-18 DOUBLE SIGNED BCD-TO-BINARY: BISL(277)

Ladder Symbol	Operand Data Areas
$\text{---} \left[ \overset{(277)}{\text{BISL C S D}} \right] \text{---}$	<b>C: Control word</b> CIO, G, A, T, C, #, DM, <b>S: 1st source word</b> CIO, G, A, T, C, DM, <b>D: 1st destination word</b> CIO, G, A, DM,
<b>Variations</b> ↑BISL(277)	

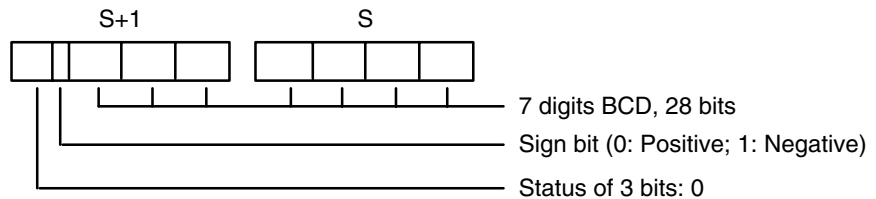
**Description**

When the execution condition is OFF, BISL(277) is not executed. When the execution condition is ON, BISL(277) converts the data in specified source words (S and S+1) from double signed BCD to double signed binary, and outputs the result to specified destination words (D and D+1). The format and data range of the source word is determined by the contents of the control word (C).

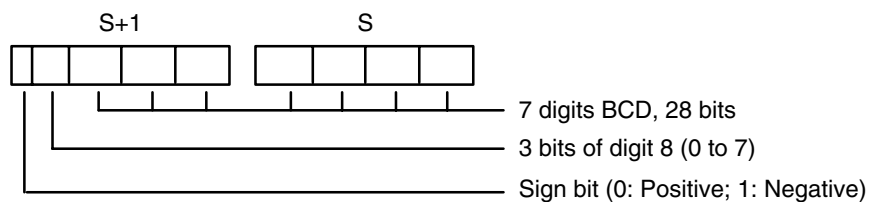
**Note** Special I/O Units sometimes output signed BCD data. Calculations using this data will normally be easier if it is first converted to signed binary data by means of BINS(275) or BISL(277).

The input data format and range designations for the various control word contents are as follows:.

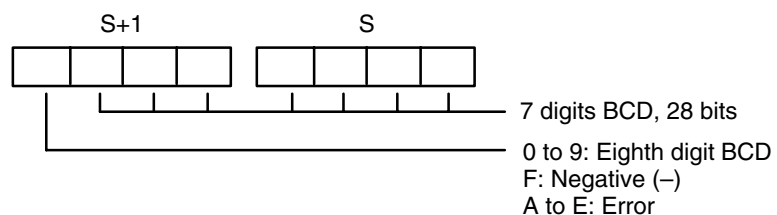
**When C = 0000 (Input Data Range: -999 9999 to 999 9999 BCD)**



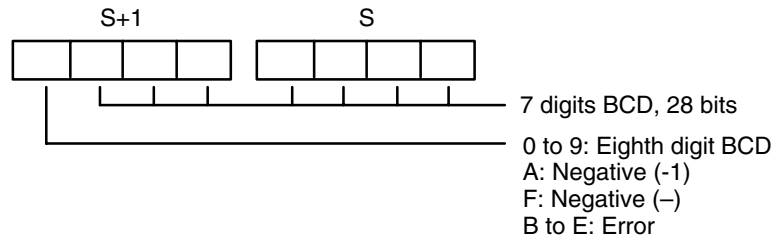
**When C = 0001 (Input Data Range: -7999 9999 to 7999 9999 BCD)**



**When C = 0002 (Input Data Range: -999 9999 to 9999 9999 BCD)**



When C = 0003 (Input Data Range: -1999 9999 to 9999 9999 BCD)



First the signed BCD data format and range in words S+1 and S are checked against the data control word (C). If the check is okay, the signed BCD data in words S+1 and S are converted to binary and output to the designated words D+1 and D. If it is not okay, the Error Flag (A50003) will turn ON and the instruction will not be executed.

In signed BCD data, a -0 is treated as a +0.

When the data to be converted is a negative number, after being converted it will be output as 2's complement and the Negative Flag (A50008) will turn ON. In order to convert a 2's complement to the true value, it is necessary to subtract it from 0.

**Precautions**

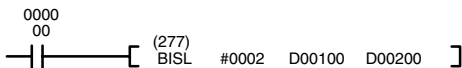
Refer to page 101 for general precautions on operand data areas.

**Flags**

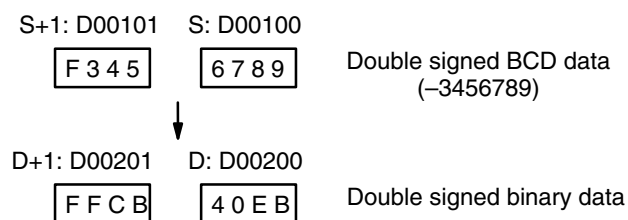
- ER (A50003): Data format is 0002, and the leftmost digit is A to E  
Data format is 0003, and the leftmost digit is B to E.  
Data to be converted is not BCD.  
Content of \*DM word is not BCD when set for BCD.
- EQ (A50006) Content of the converted data is all zeroes.
- N (A50008) Converted number is negative.

**Example**

When CIO 000000 is ON in the following example, first the signed BCD data format and range in D00101 and D00100 are checked against data control word "0002" (first operand). If the check is okay, the double signed BCD data in D00101 and D00100 is converted to binary and output to D00201 and D00200.



Address	Instruction	Operands
00000	LD	000000
00001	BISL(277)	
		#0002
		D00100
		D00200



### 5-17-19 DOUBLE SIGNED BINARY-TO-BCD: BDSL(278)

Ladder Symbol	Operand Data Areas
$(278)$ 	<b>C: Control word</b> CIO, G, A, T, C, #, DM, <b>S: 1st source word</b> CIO, G, A, T, C, DM, <b>D: 1st destination word</b> CIO, G, A, DM,
<b>Variations</b> ↑BDSL(278)	

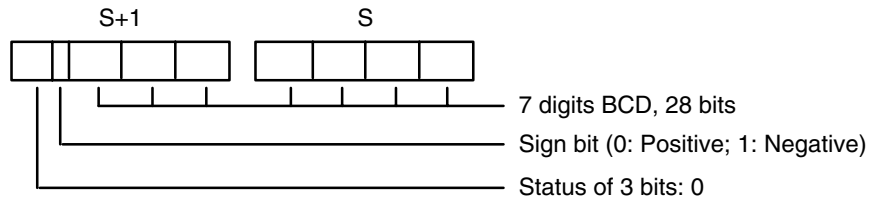
**Description**

When the execution condition is OFF, BDSL(278) is not executed. When the execution condition is ON, BDSL(278) converts the data in specified words (S and S+1) from double signed binary to double signed BCD, and outputs the result to specified destination words (D and D+1). The format of the destination word is determined by the contents of the control word (C).

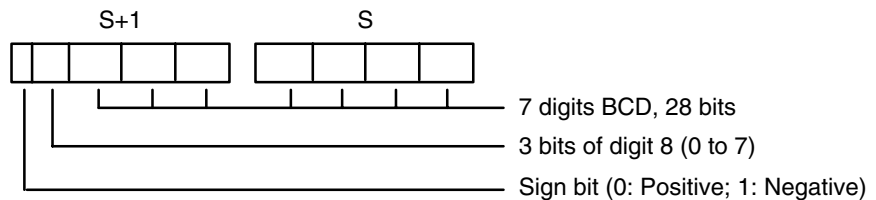
**Note** Special I/O Units sometimes sometimes require input of signed BCD data. BCDS(276) or BDSL(278) can be used to easily convert signed binary data to signed BCD data.

The output data format and range designations for the various control word contents are as follows:

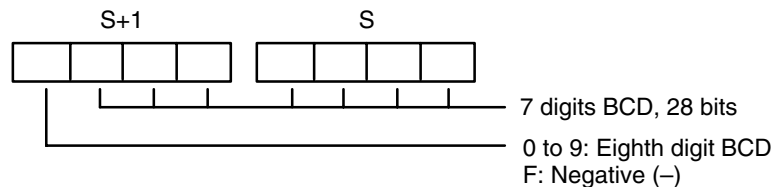
**When C = 0000 (Output Data Range: -999 9999 to 999 9999 BCD)**



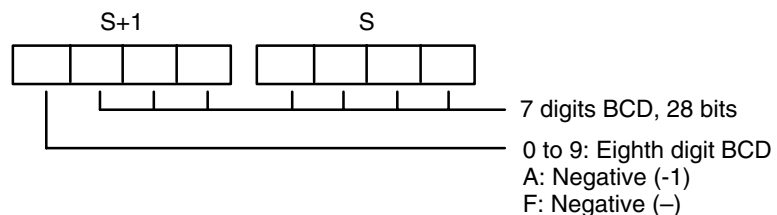
**When C = 0001 (Output Data Range: -7999 9999 to 7999 9999 BCD)**



**When C = 0002 (Output Data Range: -999 9999 to 9999 9999 BCD)**



**When C = 0003 (Output Data Range: -1999 9999 to 9999 9999 BCD)**



First the signed BCD data format and range in words S+1 and S are checked against the data control word (C). If the check is okay, the signed BCD data in words S+1 and S is converted to binary and output to the designated words D+1 and D. If the check is not okay, the Error Flag (A50003) will turn ON and the instruction will not be executed.

In signed BCD data, -0 is treated as +0.

**Precautions**

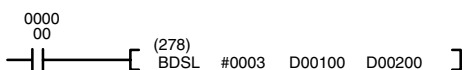
Refer to page 101 for general precautions on operand data areas.

**Flags**

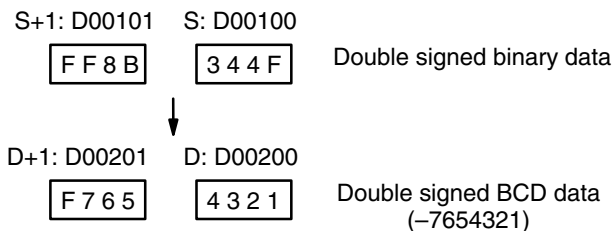
- ER (A50003): Data to be converted is not within range for data format.  
Content of a\*DM word is not BCD when set for BCD.
- EQ (A50006) Content of the converted data is all zeroes.
- N (A50008) Data to be converted is a negative number.

**Example**

When CIO 000000 is ON in the following example, first the data format and range in D00101 and D00100 are checked against data control word "0003" (first operand). If the check is okay, the double signed binary data in D00101 and D00100 is converted to BCD and output to D00201 and D00200.



Address	Instruction	Operands
00000	LD	000000
00001	BDSL(278)	
		#0003
		D00100
		D00200



## 5-18 BCD Calculation Instructions

The BCD Calculation Instructions perform arithmetic operations on BCD data. The CVM1D CPUs also support BCD symbol math instructions. Refer to 5-20 *Symbol Math Instructions* for details.

STC(078) and CLC(079), which set and clear the carry flag, are included in this group because most of the BCD operations make use of the carry flag (CY) in their results. Binary calculations and shift operations also use CY.

The addition and subtraction instructions include CY in the calculation as well as in the result. Be sure to clear CY if its previous status is not required in the calculation, and to use the result placed in CY, if required, before it is changed by execution of any other instruction.



### 5-18-1 SET CARRY: STC(078)

Ladder Symbol	Variations
	↑ STC(078)

When the execution condition is OFF, STC(078) is not executed. When the execution condition is ON, STC(078) turns ON CY (A50004).

### 5-18-2 CLEAR CARRY: CLC(079)

Ladder Symbol	Variations
	↑ CLC(079)

When the execution condition is OFF, CLC(079) is not executed. When the execution condition is ON, CLC(079) turns OFF CY (A50004).

ADD(070), ADDL(074), ADB(080), ADBL(084), SUB(0710), SUBL(075), SBB(081), and SBBL(085) all make use of the carry flag in their calculations. When using any of these instructions, use CLC(079) to clear the carry flag in order to avoid having the calculations affected by previous instructions.

ROL(062), ROLL(066), ROR(063), and RORL(067) make use of the carry flag in their rotation shift operations. When using any of these instructions, use STC(078) and CLC(079) to set and clear the carry flag.

CVM1D CPUs support add, subtract, and rotation shift instructions that do not use the carry flag in their operations. These instructions do not require STC(078) and CLC(079), and reduce the number of program steps that are needed.

### 5-18-3 BCD ADD: ADD(070)

Ladder Symbol	Operand Data Areas
	<b>Au: Augend word</b> CIO, G, A, T, C, #, DM, DR, IR <b>Ad: Addend word</b> CIO, G, A, T, C, #, DM, DR, IR <b>R: Result word</b> CIO, G, A, DM, DR, IR
<b>Variations</b> ↑ ADD(070)	

#### Description

When the execution condition is OFF, ADD(070) is not executed. When the execution condition is ON, ADD(070) adds the contents of Au, Ad, and CY, and places the result in R. CY will be set if the result is greater than 9999.

$$\boxed{\text{Au}} + \boxed{\text{Ad}} + \boxed{\text{CY}} \rightarrow \boxed{\text{CY}} \quad \boxed{\text{R}}$$

**Note** An equivalent symbol math instruction (+BC(406)) is also available.

#### Precautions

Au and Ad must be BCD.

**Note** Refer to page 101 for general precautions on operand data areas.

#### Flags

ER (A50003): Content of Au or Ad is not BCD.

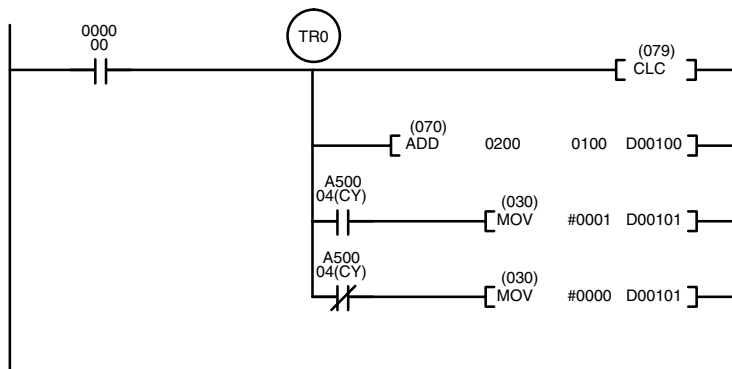
The content of a\*DM word is not BCD when set for BCD.

CY (A50004): There is a carry in the result.

EQ (A50006): The result is 0.

**Example**

When CIO 000000 is ON in the following example, CY is cleared by CLC(079), the content of CIO 0200 is added to the contents of CIO 0100 and the status of CY, the results is placed in D00100, and then either all zeros or 0001 is moved into D00101 depending on the status of CY (A50004). This ensures that any carry from the last digit is preserved in R+1 so that the entire result can be later handled as 8-digit data.



Address	Instruction	Operands
00000	LD	000000
00001	OUT	TR0
00002	CLC(079)	
00003	ADD(070)	
		0200
		0100
		D00100
00004	AND	A50004
00005	MOV(030)	
		#0001
		D00101
00006	LD	TR0
00007	AND NOT	A50004
00008	MOV(030)	
		#0000
		D00101

Although two ADD(070) can be used together to perform 8-digit BCD addition, ADDL(074) is designed specifically for this purpose.

**5-18-4 BCD SUBTRACT: SUB(071)**

Ladder Symbol	Operand Data Areas
$\text{---} \left[ \text{SUB} \quad \text{Mi} \quad \text{Su} \quad \text{R} \right]$ <p>(071)</p>	<p><b>Mi: Minuend word</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>Su: Subtrahend word</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>R: Result word</b> CIO, G, A, DM, DR, IR</p>
<p><b>Variations</b></p> <p>↑ SUB(071)</p>	

**Description**

When the execution condition is OFF, SUB(071) is not executed. When the execution condition is ON, SUB(071) subtracts the contents of Su and CY from Mi, and places the result in R. If the result is negative, CY is set and the 10's complement of the actual result is placed in R. To convert the 10's complement to the true result, subtract the content of R from zero (see example below).

$$\boxed{\text{Mi}} - \boxed{\text{Su}} - \boxed{\text{CY}} \rightarrow \boxed{\text{CY}} \quad \boxed{\text{R}}$$

**Note** An equivalent symbol math instruction (-BC(416)) is also available.

**Precautions**

Mi and Su must be BCD.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

- ER (A50003): Content of Mi or Su is not BCD.  
Content of \*DM word is not BCD when set for BCD.
- CY (A50004): The result is negative, i.e., when Mi is less than Su plus CY.
- EQ (A50006): The result is 0.

**Note** Be sure to clear the carry flag with CLC(079) before executing SUB(071) if its previous status is not required, and check the status of CY after doing a subtraction with SUB(071). If CY is ON as a result of executing SUB(071) (i.e., if the result is negative), the result is output as the 10's complement of the true answer. To convert the output result to the true value, subtract the value in R from 0.

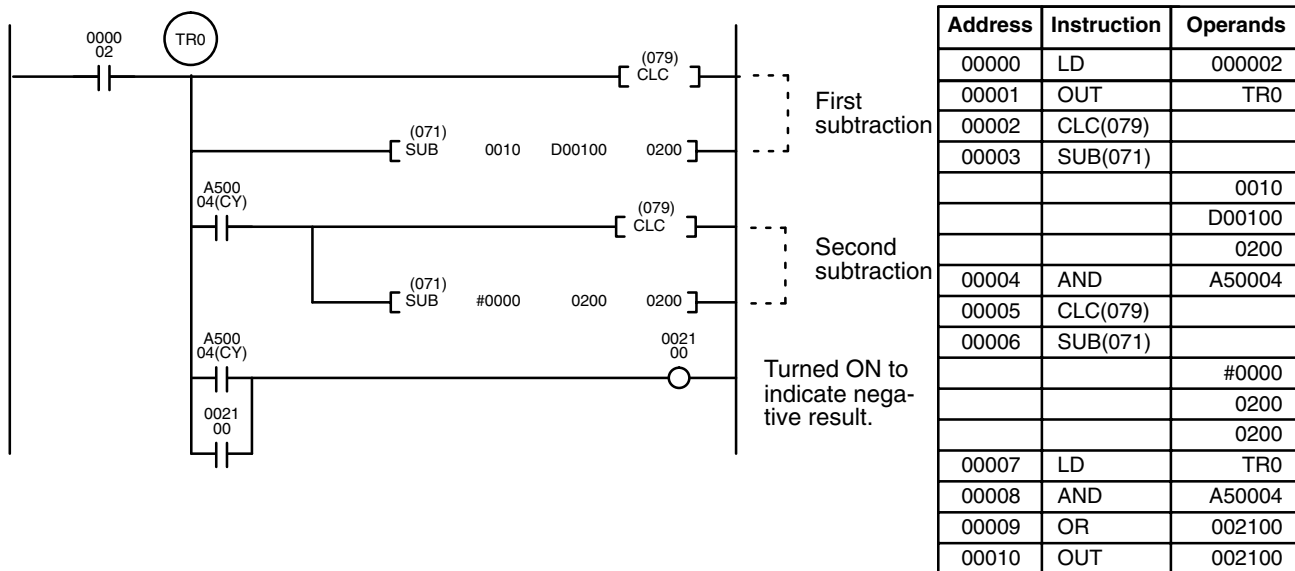
**Example**

When CIO 000002 is ON in the following example, the following ladder program clears CY, subtracts the contents of D00100 and CY from the content of CIO 0010, and places the result in CIO 0200.

If CY is set by executing SUB(071), the result in CIO 0200 is subtracted from zero (note that CLC(079) is again required to obtain an accurate result), the result is placed back in CIO 0200, and CIO 002100 is turned ON to indicate a negative result.

If CY is not set by executing SUB(071), the result is positive, the second subtraction is not performed, and CIO 002100 is not turned ON. CIO 002100 is programmed as a self-maintaining bit so that a change in the status of CY will not turn it OFF when the program is re-scanned.

In this example, differentiated forms of SUB(071) are used so that the subtraction operation is performed only once each time CIO 000002 turns ON. When another subtraction operation is to be performed, CIO 000002 will need to be turned OFF for at least one scan (resetting CIO 002100) and then turned back ON.



The first and second subtractions for this diagram are shown below using example data for CIO 0010 and D00100.

**Note** The actual SUB(071) operation involves subtracting Su and CY from 10,000 plus Mi. For positive results the leftmost digit is truncated. For negative results the 10s complement is obtained. The procedure for establishing the correct answer is given below.

First Subtraction		
CIO 0010	1029	
D00100	- 3452	
CY	- 0	
CIO 0200	7577	(1029 + (10000 - 3452))
CY	1	(negative result)

**Second Subtraction**

	0000	
CIO 0200	-7577	
CY	-0	
CIO 0200	2423	(0000 + (10000 - 7577))
CY	1	(negative result)

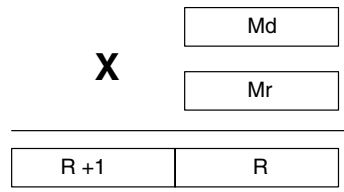
In the above case, the program would turn ON CIO 002100 to indicate that the value held in CIO 0200 is negative.

**5-18-5 BCD MULTIPLY: MUL(072)**

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(072)</p> <p>— [ MUL    Md        Mr        R ]</p> <p><b>Variations</b></p> <p>↑ MUL(072)</p>	<p><b>Operand Data Areas</b></p> <p><b>Md: Multiplicand word</b>    CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>Mr: Multiplier word</b>        CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>R: Result word</b>                CIO, G, A, DM</p>
---	--

**Description**

When the execution condition is OFF, MUL(072) is not executed. When the execution condition is ON, MUL(072) multiplies Md by the content of Mr, and places the result in R and R+1.



**Note** An equivalent symbol math instruction (\*B(424)) is also available.

**Precautions**

Md and Mr must be BCD.

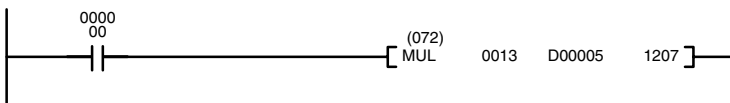
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

ER (A50003): Content of Md or Mr is not BCD.  
 The content of a \*DM word is not BCD when set for BCD.  
 EQ (A50006): The result is 0.

**Example**

When CIO 000000 is ON in the following example, the contents of CIO 0013 and D00005 are multiplied and the results is placed in CIO 1207 and CIO 1208. Example data and calculations are shown below the program.



Address	Instruction	Operands
00000	LD	000000
00001	MUL(072)	
		0013
		D00005
		1207

Md: CIO 0013			
3	3	5	6

X

Mr: D00005			
0	0	2	5

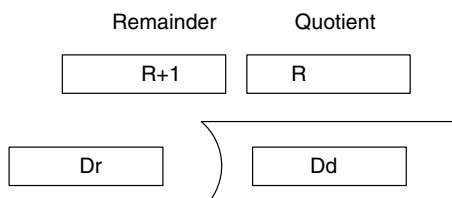
R+1: CIO 1208				R: CIO 1207			
0	0	0	8	3	9	0	0

### 5-18-6 BCD DIVIDE: DIV(073)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(073)</p> <p>— [ DIV    Dd    Dr    R ]</p> <p><b>Variations</b></p> <p>↑ DIV(073)</p>	<p><b>Operand Data Areas</b></p> <p><b>Dd: Dividend word</b>    CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>Dr: Divisor word</b>    CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>R: Result word</b>    CIO, G, A, DM</p>
---	---

**Description**

When the execution condition is OFF, DIV(073) is not executed and the program moves to the next instruction. When the execution condition is ON, Dd is divided by Dr and the result is placed in R and R + 1: the quotient in R and the remainder in R + 1.



**Note** An equivalent symbol math instruction (/B(434)) is also available.

**Precautions**

Dd and Dr must be BCD.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

ER (A50003): Content of Dd or Dr is not BCD.  
 Content of \*DM word is not BCD when set for BCD.  
 EQ (A50006): The result is 0.

**Example**

When CIO 000000 is ON in the following example, the content of CIO 0020 is divided by the content of CIO 1209 and the results is placed in D00017 and D00018. Example data and calculations are shown below the program.

Address	Instruction	Operands
00000	LD	000000
00001	DIV(073)	
		0020
		1209
		D00017

Quotient

R: D00017				R + 1: D00018			
1	1	5	0	0	0	0	2

Remainder

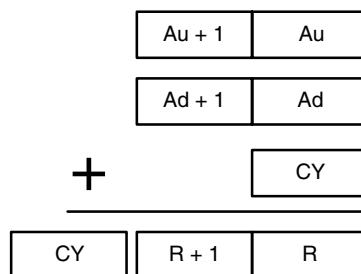
Dd: CIO 1209				Dd: CIO 0020			
0	0	0	3	3	4	5	2

### 5-18-7 DOUBLE BCD ADD: ADDL(074)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(074)</p> <p>— [ ADDL Au Ad R ]</p> <p><b>Variations</b></p> <p>↑ ADDL(074)</p>	<p><b>Operand Data Areas</b></p> <p><b>Au: 1<sup>st</sup> augend word</b> CIO, G, A, T, C, #, DM</p> <p><b>Ad: 1<sup>st</sup> addend word</b> CIO, G, A, T, C, #, DM</p> <p><b>R: 1<sup>st</sup> result word</b> CIO, G, A, DM</p>
--	--

**Description**

When the execution condition is OFF, ADDL(074) is not executed. When the execution condition is ON, ADDL(074) adds the content of CY to the 8-digit value in Au and Au+1 to the 8-digit value in Ad and Ad+1 and places the result in R and R+1. CY will be set if the result is greater than 9999 9999.



**Note** An equivalent symbol math instruction (+BCL(407)) is also available.

**Precautions**

Au and Ad must be BCD.

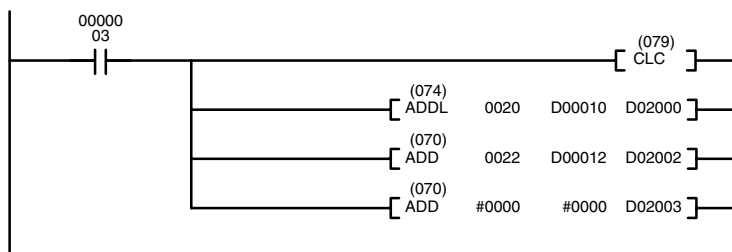
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

- ER (A50003): Content of Au or Ad is not BCD.  
Content of \*DM word is not BCD when set for BCD.
- CY (A50004): There is a carry in the result.
- EQ (A50006): The result is 0.

**Example**

When CIO 000003 is ON, the following program adds two 12-digit numbers, the first contained in CIO 0020 through CIO 0022 and the second in D00010 through D00012. The result is placed in D02000 through D02002. In the second addition (using ADD(070)), any carry from the first addition will be automatically included. The carry from the second addition is placed in D02003 by using another ADD(070) with two all-zero constants.



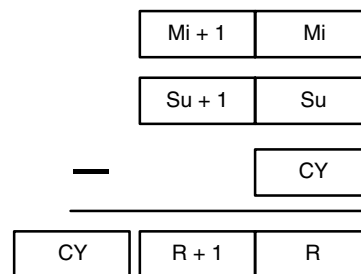
Address	Instruction	Operands
00000	LD	000003
00001	CLC(079)	
00002	ADDL(074)	
		0020
		D00010
		D02000
00003	ADD(070)	
		0022
		D00012
		D02002
00004	ADD(070)	
		#0000
		#0000
		D02003

## 5-18-8 DOUBLE BCD SUBTRACT: SUBL(075)

Ladder Symbol	Operand Data Areas
$\text{---} \begin{matrix} (075) \\ \text{---} \end{matrix} \left[ \text{SUBL} \quad \text{Mi} \quad \text{Su} \quad \text{R} \right]$	<b>Mi: 1<sup>st</sup> minuend word</b> CIO, G, A, T, C, #, DM <b>Su: 1<sup>st</sup> subtrahend word</b> CIO, G, A, T, C, #, DM <b>R: 1<sup>st</sup> result word</b> CIO, G, A, DM
<b>Variations</b> ↑ SUBL(075)	

**Description**

When the execution condition is OFF, SUBL(075) is not executed. When the execution condition is ON, SUBL(075) subtracts CY and the 8-digit content of Su and Su+1 from the 8-digit value in Mi and Mi+1, and places the result in R and R+1. If the result is negative, CY is set and the 10's complement of the actual result is placed in R. To convert the 10's complement to the true result, subtract the content of R from zero.



**Note** An equivalent symbol math instruction (–BCL(417)) is also available.

**Precautions**

Mi and Su must be BCD.

Constants are input using eight digits.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

ER (A50003): Content of Mi, Mi+1, Su or Su+1 is not BCD.

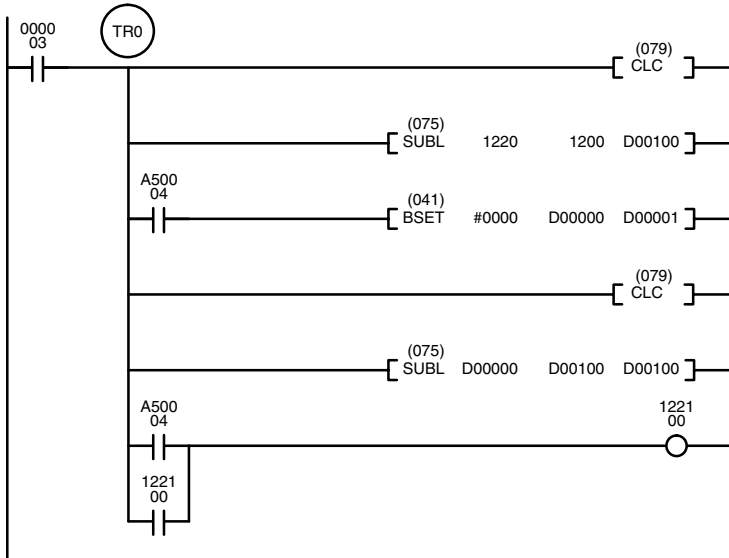
Content of \*DM word is not BCD when set for BCD.

CY (A50004): There is a carry in the result.

EQ (A50006): The result is 0.

**Example**

The following example works much like that for single-word subtraction. In this example, however, the 8-digit number in CIO 0121 and CIO 0120 is subtracted from the 8-digit number in CIO 0201 and CIO 0200 when CIO 000003 is ON, and the result is output to D00101 and D00100. If the result is negative, the complement is then subtracted from 0 to yield the actual number and CIO bit 002100 (a self-holding bit) is turned ON to indicate the negative result.



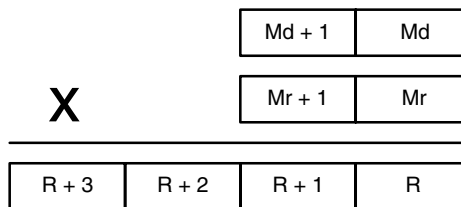
Address	Instruction	Operands
00000	LD	000003
00001	OUT	TR0
00002	CLC(079)	
00003	SUBL(075)	
		1220
		1200
		D00100
00004	AND	A50004
00005	BSET(041)	
		#0000
		D00000
		D00001
00006	CLC(079)	
00007	SUBL(075)	
		D00000
		D00100
		D00100
00008	LD	TR0
00009	AND	A50004
00010	OR	122100
00011	OUT	122100

**5-18-9 DOUBLE BCD MULTIPLY: MULL(076)**

<p><b>Ladder Symbol</b></p> <p>— [ (076) MULL Md Mr R ]</p> <p><b>Variations</b></p> <p>↑ MULL(076)</p>	<p><b>Operand Data Areas</b></p> <p><b>Md: 1<sup>st</sup> multiplicand wd</b> CIO, G, A, T, C, #, DM</p> <p><b>Mr: 1<sup>st</sup> multiplier word</b> CIO, G, A, T, C, #, DM</p> <p><b>R: 1<sup>st</sup> result word</b> CIO, G, A, DM</p>
---	--

**Description**

When the execution condition is OFF, MULL(076) is not executed. When the execution condition is ON, MULL(076) multiplies the 8-digit content of Md and Md+1 by the content of Mr and Mr+1, and places the result in R to R+3.



**Note** An equivalent symbol math instruction (\*BL(425)) is also available.

**Precautions**

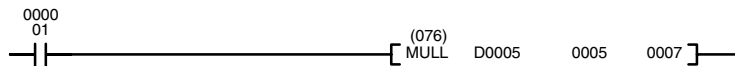
Md, Md+1, Mr, and Mr+1 must be BCD.

**Note** Refer to page 101 for general precautions on operand data areas.



**Flags** ER (A50003): Content of Md, Md+1, Mr, or Mr+1 is not BCD.  
 Content of \*DM word is not BCD when set for BCD.  
 EQ (A50006): The result is 0.

**Example** When CIO 00001 is ON in the following example, the 8-digit content of D00005 and D00006 is multiplied by the content of CIO 0005 and CIO 0006 and places the 16-digit result in CIO 0007, CIO 0008, CIO 0009, and CIO 0010.



Address	Instruction	Operands
00000	LD	000001
00001	MULL(076)	
		D00005
		0005
		0007

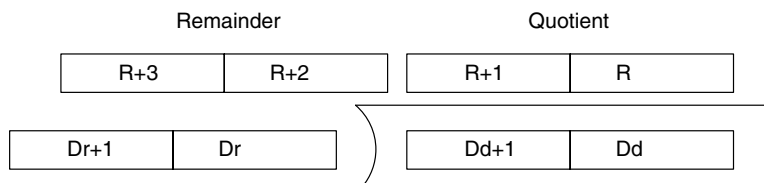
D0006	D0005
8 0 0 1	3 5 9 2
X	
0006	0005
0 0 0 0	0 0 2 5

0010	0009	0008	0007
0 0 0 0	0 0 2 0	0 0 3 3	9 8 0 0

### 5-18-10 DOUBLE BCD DIVIDE: DIVL(077)

<p><b>Ladder Symbol</b></p> <p>— [ DIVL<sup>(077)</sup> Dd Dr R ]</p> <p><b>Variations</b></p> <p>↑ DIVL(077)</p>	<p><b>Operand Data Areas</b></p> <p><b>Dd: 1<sup>st</sup> dividend word</b> CIO, G, A, T, C, #, DM</p> <p><b>Dr: 1<sup>st</sup> divisor word</b> CIO, G, A, T, C, #, DM</p> <p><b>R: 1<sup>st</sup> result word</b> CIO, G, A, DM</p>
---	---

**Description** When the execution condition is OFF, DIVL(077) is not executed. When the execution condition is ON, the 8-digit content of Dd and D+1 is divided by the content of Dr and Dr+1 and the result is placed in R to R+3: the quotient in R and R+1, and the remainder in R+2 and R+3.



**Note** An equivalent symbol math instruction (/BL(435)) is also available.

**Precautions** Dr and Dr+1 must not contain 0 and the content of Dd, Dd+1, Dr or Dr+1 must be BCD.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags** ER (A50003): Dr and Dr+1 contain 0.  
 Content of Dd, Dd+1, Dr or Dr+1 is not BCD.  
 The content of a \*DM word is not BCD when set for BCD.  
 CY (A50004): There is a carry in the result.  
 EQ (A50006): The result is 0.

**Example**

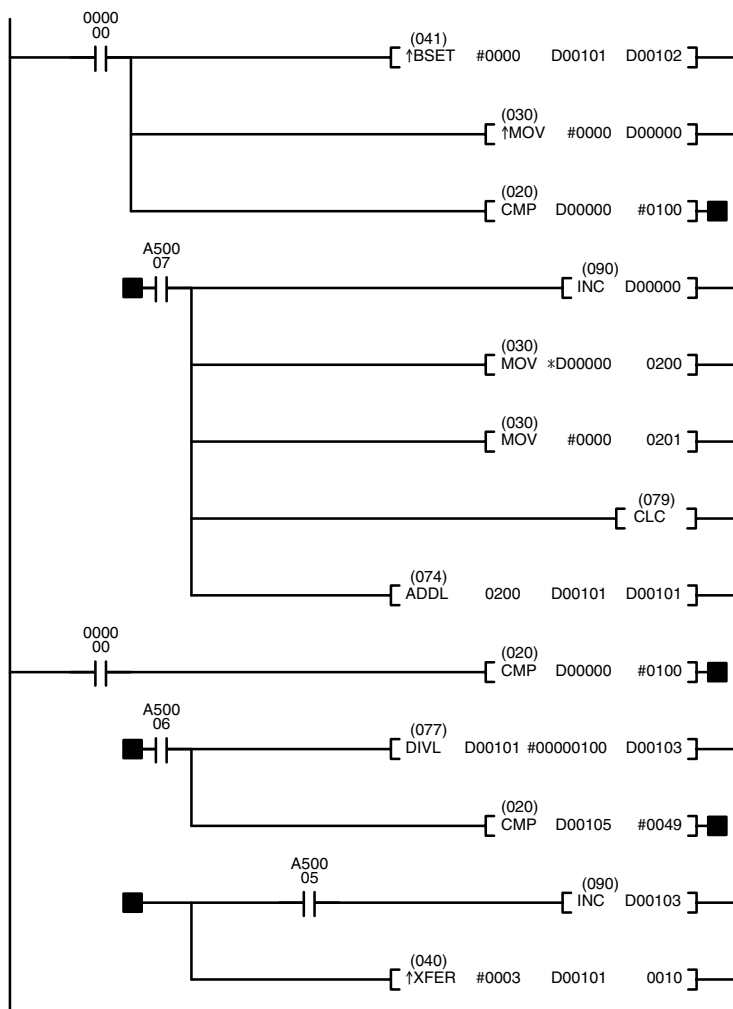
The following example shows how to use DIVL(77) to calculate the average of 100 four-digit numbers. These numbers are added and divided using the long versions of the instructions so that the answer can be rounded to preserve accuracy. This example illustrates not only the use of DIVL(77), but also the use of several other instructions and the use of indirect addressing.

The following words and bits are used in this program.

Bit/word	Application
CIO 000000	Controls execution of the program.
D00101 and D00102	Hold the result of the addition.
D00000	Points to the next word to be added.
A50005 through A50007 (Less Than, Equals, and Greater Than Flags)	Control execution in combination with CMP(020) to end addition and initiate division when the 100th number has been added and to add 1 when rounding up the result is required.
CIO 0200 and 0201	Hold the next values to be added.
D00103 through D00105	Hold the results of division (D00105 is remainder).
CIO 0010 through CIO 0012	CIO 0012 outputs the average and CIO 0010 and 0011 output the sum of the 100 numbers.

When CIO 000000 is ON in the following example, the first two lines in the program clear words used in the remainder of the program. The remainder of the instruction block from CMP(020) adds consecutive numbers indirectly addressed through D00000. The numbers are moved to CIO 0200 so that long addition is possible (CIO 0201 is always zero).

When 100 numbers have been added, the CMP(020) instructions end the addition and start the division, rounding, and output procedure. The result is rounded by incrementing D00103 when the remainder from the division is greater than 0049. Finally, XFER(040) is used to places the results in I/O words for output to an external device, e.g., a display device.



Address	Instruction	Operands
00000	LD	000000
00001	↑BSET(041)	
		#000
		D00101
		D00102
00002	↑MOV(030)	
		#0000
		D00000
00003	CMP(020)	
		D00000
		#0100
00004	AND	A50007
00005	INC(090)	
		D00000
00006	MOV(030)	
		*D00000
		0200
00007	MOV(030)	
		#0000
		0201
00008	CLC(079)	
00009	ADDL(074)	
		0200
		D00101
		D00101
00010	LD	000000
00011	CMP(020)	
		D00000
		#0100
00012	AND	A50006
00013	DIVL(077)	
		D00101
		#00000100
		D00103
00014	CMP(020)	
		D00105
		#0049
00015	OUT	TR1
00016	AND	A50005
00017	INC(090)	
		D00103
00018	LD	TR1
00019	↑XFER(040)	
		#0003
		D00101
		0010

## 5-19 Binary Calculation Instructions

The Binary Calculation Instructions all perform arithmetic operations on binary (hexadecimal) data. The CVM1D CPUs also support binary symbol math instructions. Refer to 5-20 *Symbol Math Instructions* for details.

The addition and subtraction instructions include CY in the calculation as well as in the result. Be sure to clear CY if its previous status is not required in the calculation, and to use the result placed in CY, if required, before it is changed by the execution of any other instruction. STC(078) and CLC(079) can be used to control CY. Refer to 5-18 *BCD Calculation Instructions* for details on STC(078) and CLC(079).

### 5-19-1 BINARY ADD: ADB(080)

<p><b>Ladder Symbol</b></p> <p><b>Variations</b></p> <p>↑ ADB(080)</p>	<p><b>Operand Data Areas</b></p> <p><b>Au: Augend word</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>Ad: Addend word</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>R: Result word</b> CIO, G, A, DM, DR, IR</p>
--	---

**Description**

When the execution condition is OFF, ADB(080) is not executed. When the execution condition is ON, ADB(080) adds the content of Au, Ad, and CY, and places the result in R. CY will be set if the result is greater than FFFF.

$$\boxed{\text{Au}} + \boxed{\text{Ad}} + \boxed{\text{CY}} \rightarrow \boxed{\text{CY}} \boxed{\text{R}}$$

**Note** An equivalent symbol math instruction (+C(402)) is also available, although +C(402) has the additional feature of controlling the Overflow (A50009) and Underflow (A50010) Flags.

**Precautions**

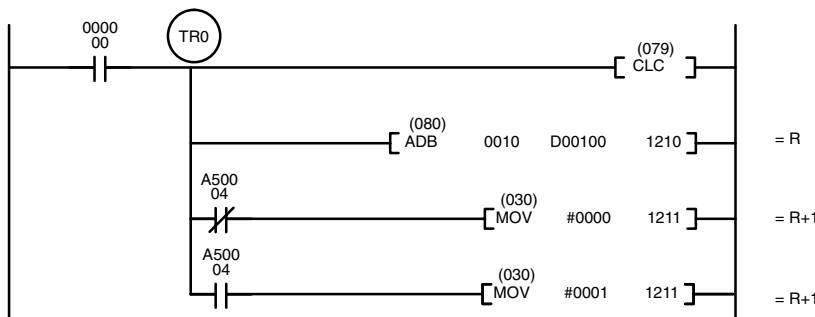
Refer to page 101 for general precautions on operand data areas.

**Flags**

- ER (A50003): Content of \*DM word is not BCD when set for BCD.
- CY (A50004): The result is greater than FFFF.
- EQ (A50006): The result is 0.
- N (A50008): Shows the status of bit 15 of R after execution.

**Examples**

The following example shows a four-digit addition with CY used to place either 0000 or 0001 into R+1 to ensure that any carry is preserved.



Address	Instruction	Operands
00000	LD	000000
00001	OUT	TR0
00002	CLC(079)	
00003	ADB(080)	
		0010
		D00100
		1210
00004	AND NOT	A50004
00005	MOV(030)	
		#0000
		1211
00006	LD	TR0
00007	AND	A50004
00008	MOV(030)	
		#0001
		1211

In the following example,  $A6E2 + 80C5 = 127A7$ . The result is a five-digit number, so  $CY (A50004) = 1$ , and the content of  $R + 1$  becomes 0001.

	Au: CIO 0010
	A 6 E 2
+	
	Ad: D00100
	8 0 C 5
-----	
	R+1: D01001
	0 0 0 1
	R: D01000
	2 7 A 7

Eight-digit binary numbers can be added more quickly and easily using the DOUBLE BINARY ADD: ADBL(084) instruction instead of a combination of ADB(080) instructions.

## 5-19-2 BINARY SUBTRACT: SBB(081)

Ladder Symbol	Operand Data Areas
$\text{---} \overset{(081)}{[ \text{SBB} \quad \text{Mi} \quad \text{Su} \quad \text{R} ]}$	<b>Mi: Minuend word</b> CIO, G, A, T, C, #, DM, DR, IR <b>Su: Subtrahend word</b> CIO, G, A, T, C, #, DM, DR, IR <b>R: Result word</b> CIO, G, A, DM, DR, IR
<b>Variations</b> ↑ SBB(081)	

### Description

When the execution condition is OFF, SBB(081) is not executed. When the execution condition is ON, SBB(081) subtracts the contents of Su and CY from Mi and places the result in R. If the result is negative, CY is set and the 2's complement of the actual result is placed in R. To obtain the true answer when the result is negative, the 2's complement placed in R must be subtracted from 0000.

$$\boxed{\text{Mi}} - \boxed{\text{Su}} - \boxed{\text{CY}} \rightarrow \boxed{\text{CY}} \boxed{\text{R}}$$

**Note** An equivalent symbol math instruction ( $-C(412)$ ) is also available, although  $-C(412)$  has the additional feature of controlling the Overflow (A50009) and Underflow (A50010) Flags.

### Precautions

Refer to page 101 for general precautions on operand data areas.

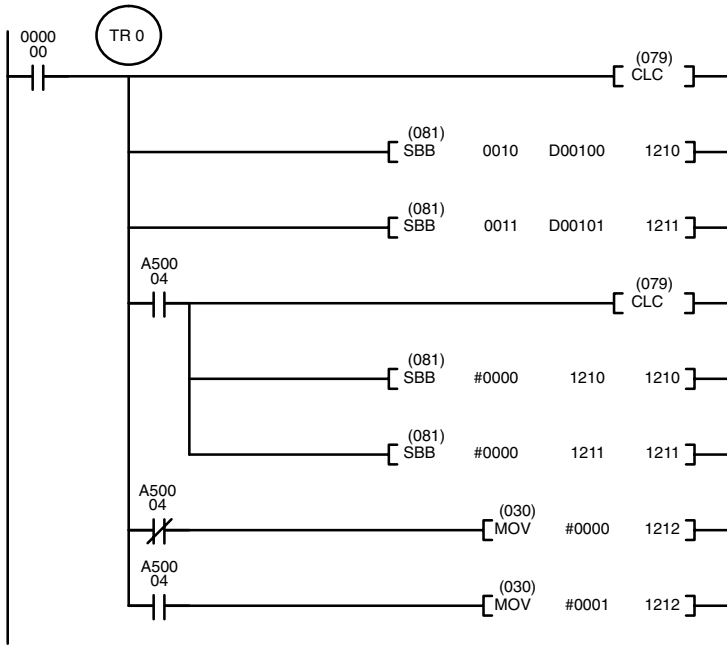
### Flags

ER (A50003): Content of \*DM word is not BCD when set for BCD.  
 CY (A50004): The result is negative, i.e., when Mi is less than Su plus CY.  
 EQ (A50006): The result is 0.  
 N (A50008): Shows the status of bit 15 of R.

**Example**

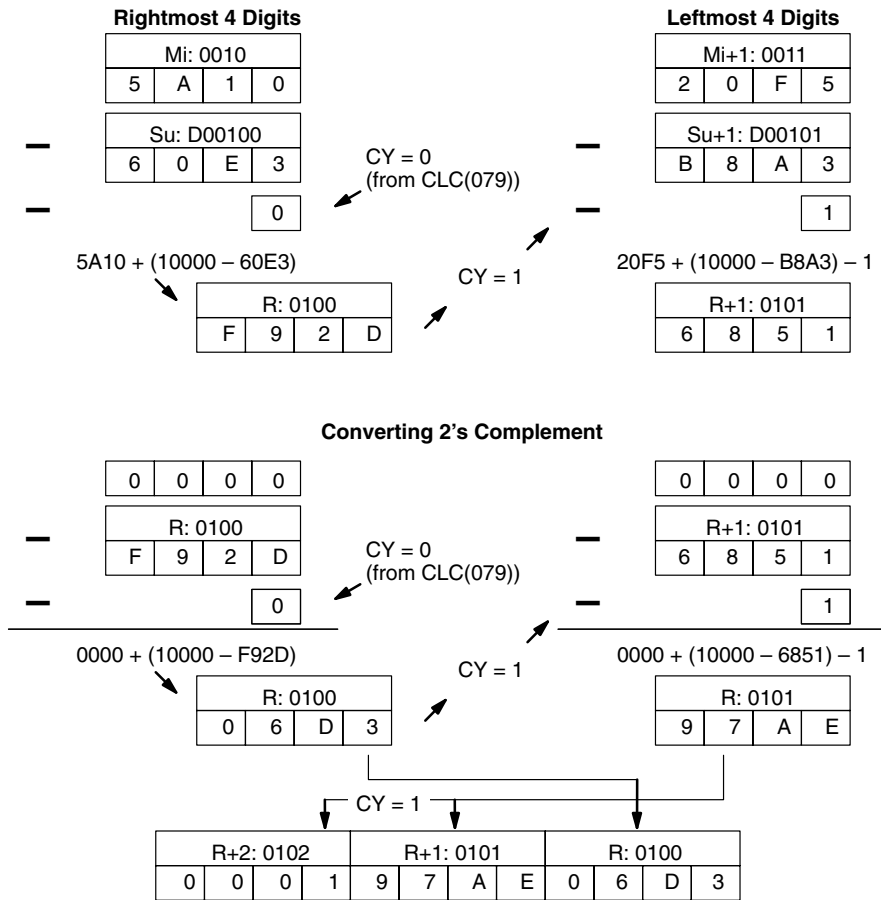
The following example demonstrates the use of SBB(081) in an 8-digit subtraction. In actual practice, 8-digit binary numbers can be subtracted more quickly and easily using the DOUBLE BINARY SUBTRACT: SBBL(085) instruction instead of a combination of SBB(081) instructions.

CY is tested following the first two subtractions to see if the result is negative. If it is, the first result (the complement) is subtracted from zero to obtain the true result, and either 0000 or 0001 is placed in CIO 0102 (0001 indicates a negative result).



Address	Instruction	Operands
00000	LD	000000
00001	OUT	TR0
00002	CLC(079)	
00003	SBB(081)	
		0010
		D00100
		1210
00004	SBB(081)	
		0011
		D00101
		1211
00005	AND	A50004
00006	CLC(079)	
00007	SBB(081)	
		#0000
		1210
		1210
00008	SBB(081)	
		#0000
		1211
		1211
00009	LD	TR0
00010	AND NOT	A50004
00011	MOV(30)	
		#0000
		1212
00012	LD	TR0
00013	AND	A50004
00014	MOV(30)	
		#0001
		1212

In the following example,  $20F55A10 - B8A360E3 = 97AE06D3$ . In the rightmost four-digit subtraction,  $S_u$  is less than  $M_i$ , so  $CY$  (A50004) becomes 1, and the result of the leftmost four-digit subtraction is decremented by 1. In the final calculations,  $0000 - F9D2 = 0000 + (10000 - F9D2) = 06D3$ .  $0000 - 6851 - 1$  (because  $CY$  is 1) =  $0000 + (10000 - 6851 - 1) = 97AE$ . The content of 0102, 0001, indicates a negative result.

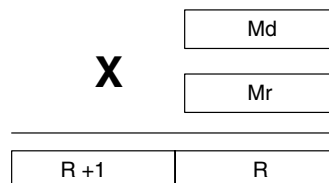


### 5-19-3 BINARY MULTIPLY: MLB(082)

<b>Ladder Symbol</b>	<b>Operand Data Areas</b>
(082) — [ MLB Md Mr R ]	<b>Md: Multiplicand word</b> CIO, G, A, T, C, #, DM, DR, IR
<b>Variations</b>	<b>Mr: Multiplier word</b> CIO, G, A, T, C, #, DM, DR, IR
↑ MLB(082)	<b>R: Result word</b> CIO, G, A, DM

#### Description

When the execution condition is OFF, MLB(082) is not executed. When the execution condition is ON, MLB(082) multiplies the content of Md by the content of Mr, places the rightmost four digits of the result in R, and places the leftmost four digits in R+1.



**Note** An equivalent symbol math instruction (\*U(422)) is also available.

**Precautions**

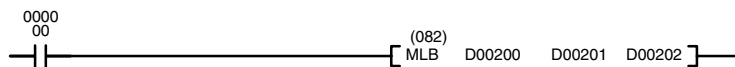
Refer to page 101 for general precautions on operand data areas.

**Flags**

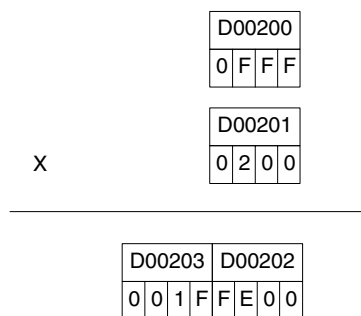
- ER (A50003): Content of \*DM word is not BCD when set for BCD.
- EQ (A50006): The result is 0.
- N (A50008): Shows the status of bit 15 of R+1.

**Example**

When CIO 000000 is ON in the following example, the four-digit hexadecimal content of D00200 is multiplied by the four-digit hexadecimal content of D00201 and the 8-digit hexadecimal result is stored in D00202 and D00203.



Address	Instruction	Operands
00000	LD	000000
00001	MLB(0820	
		D00200
		D00201
		D00202

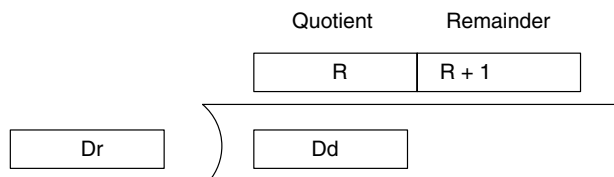


### 5-19-4 BINARY DIVIDE: DVB(083)

Ladder Symbol	Operand Data Areas
$\text{---} \left[ \overset{(083)}{\text{DVB}} \quad \text{Dd} \quad \text{Dr} \quad \text{R} \right]$	<b>Dd: Dividend word</b> CIO, G, A, T, C, #, DM, DR, IR <b>Dr: Divisor word</b> CIO, G, A, T, C, #, DM, DR, IR <b>R: Result word</b> CIO, G, A, DM
<b>Variations</b> ↑ DVB(083)	

**Description**

When the execution condition is OFF, DVB(083) is not executed. When the execution condition is ON, DVB(083) divides the content of Dd by the content of Dr and the result is placed in R and R+1: the quotient in R, the remainder in R+1.



**Note** An equivalent symbol math instruction (/U(432)) is also available.

**Precautions**

Dr must not be 0.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

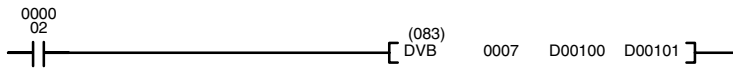
- ER (A50003): Dr contains 0.  
Content of \*DM word is not BCD when set for BCD.
- EQ (A50006): The result is 0.
- N (A50008): Shows the status of bit 15 of R.



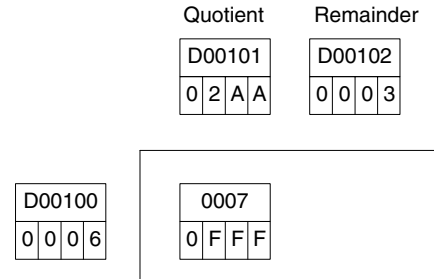
**Example**

When CIO 000002 is ON in the following example, the four-digit hexadecimal content of CIO 0007 is divided by the four-digit hexadecimal content of D00100. The quotient is stored in D00101 with the remainder stored in D00102.

**Note** If the content of the divisor word D00101 is zero, the Error Flag (bit A50003) is set and the instruction is not executed.



Address	Instruction	Operands
00000	LD	000002
00001	DVB(083)	
		0007
		D00100
		D00101

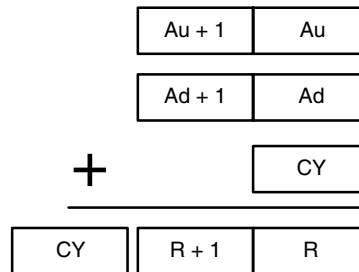


**5-19-5 DOUBLE BINARY ADD: ADBL(084)**

Ladder Symbol	Operand Data Areas
$\text{---} \left[ \overset{(084)}{\text{ADBL}} \quad \text{Au} \quad \text{Ad} \quad \text{R} \right]$	<b>Au: 1<sup>st</sup> augend word</b> CIO, G, A, T, C, #, DM <b>Ad: 1<sup>st</sup> addend word</b> CIO, G, A, T, C, #, DM <b>R: 1<sup>st</sup> result word</b> CIO, G, A, DM
<b>Variations</b> ↑ ADBL(084)	

**Description**

When the execution condition is OFF, ADBL(084) is not executed. When the execution condition is ON, ADBL(084) adds the 8-digit content of Au+1 and Au, the 8-digit content of Ad+1 and Ad, and CY, and places the result in R. CY will be set if the result is greater than FFFF FFFF.



**Note** An equivalent symbol math instruction (+CL(403)) is also available, although +CL(403) has the additional feature of controlling the Overflow (A50009) and Underflow (A50010) Flags.

**Precautions**

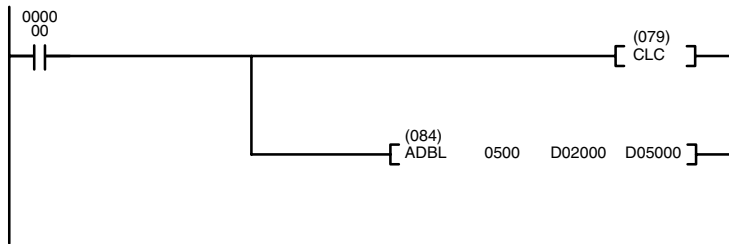
Refer to page 101 for general precautions on operand data areas.

**Flags**

- ER (A50003): Content of \*DM word is not BCD when set for BCD.
- CY (A50004): The result is greater than FFFF FFFF.
- EQ (A50006): The result is 0.
- N (A50008): Shows the status of bit 15 of R+1.

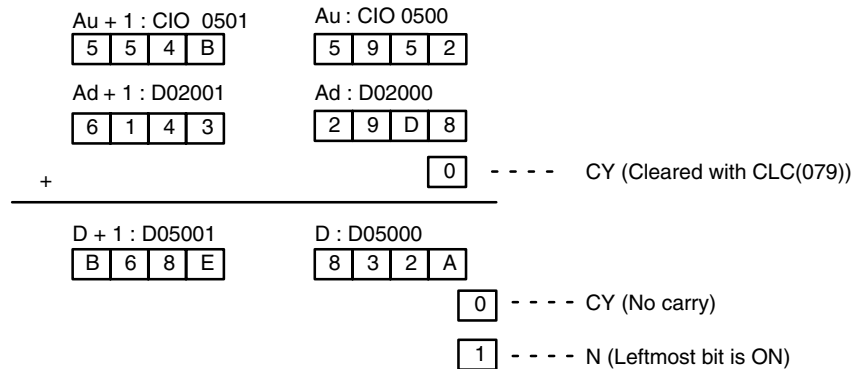
**Example**

The following example shows an 8-digit addition with CY (A50004) used to store the status of the 9<sup>th</sup> digit. The status of CY would need to be stored in another word (normally D05002) before it was affected by execution of another instruction.



Address	Instruction	Operands
00000	LD	000000
00001	CLC(079)	
00002	ADBL(084)	
		0500
		D02000
		D05000

554B5952 + 614329D2 = B68E832A

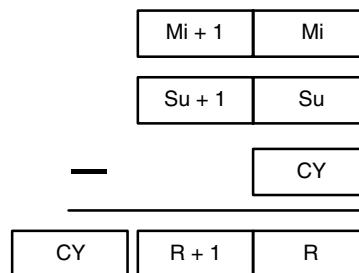


**5-19-6 DOUBLE BINARY SUBTRACT: SBBL(085)**

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(085)</p> <p>— [ SBBL Mi Su R ]</p> <p><b>Variations</b></p> <p>↑ SBBL(085)</p>	<p><b>Operand Data Areas</b></p> <p><b>Mi: 1<sup>st</sup> minuend word</b> CIO, G, A, T, C, #, DM</p> <p><b>Su: 1<sup>st</sup> subtrahend word</b> CIO, G, A, T, C, #, DM</p> <p><b>R: 1<sup>st</sup> result word</b> CIO, G, A, DM</p>
--	---

**Description**

When the execution condition is OFF, SBBL(085) is not executed. When the execution condition is ON, SBBL(085) subtracts CY and the 8-digit value in Su and Su+1 from the 8-digit value in Mi and Mi+1, and places the result in R and R+1. If the result is negative, CY is set and the 2's complement of the actual result is placed in R. To convert the 2's complement to the true result, subtract the content of R from zero.



**Note** An equivalent symbol math instruction (–CL(413)) is also available, although –CL(413) has the additional feature of controlling the Overflow (A50009) and Underflow (A50010) Flags.

**Precautions**

Refer to page 101 for general precautions on operand data areas.

**Flags**

ER (A50003): Content of \*DM word is not BCD when set for BCD.

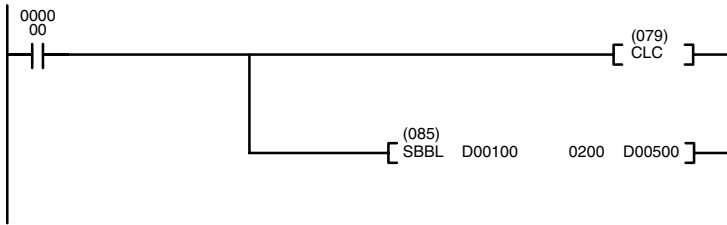
CY (A50004): The result is negative.

EQ (A50006): The result is 0.

N (A50008): Shows the status of bit 15 of R+1.

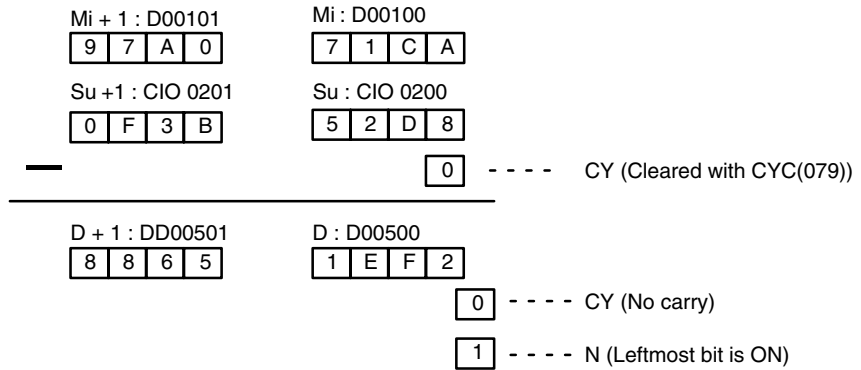
**Example**

In this example, the 8-digit number in CIO 0201 and CIO 0200 is subtracted from the 8-digit number in D00101 and D00100 when CIO 000000 is ON, and the result is output to D00501 and D00500. If the result is negative, CY (A50004) is turned ON and the 2's complement of the result is output to D00501 and D00500. Refer to 5-19-2 BINARY SUBTRACT: SBB(081) for an example of converting a 2's complement.



Address	Instruction	Operands
00000	LD	000000
00001	CLC(079)	
00002	SBBL(085)	
		D00100
		0200
		D00500

$$97A071CA - 0F3B52D8 = 88651DF2$$

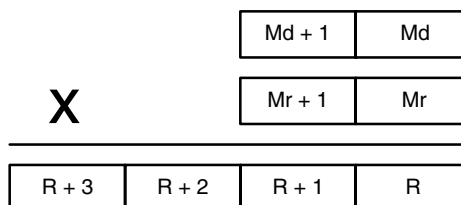


**5-19-7 DOUBLE BINARY MULTIPLY: MLBL(086)**

Ladder Symbol	Operand Data Areas
	<p><b>Md: 1<sup>st</sup> multiplicand wd</b> CIO, G, A, T, C, #, DM</p> <p><b>Mr: 1<sup>st</sup> multiplier word</b> CIO, G, A, T, C, #, DM</p> <p><b>R: 1<sup>st</sup> result word</b> CIO, G, A, DM</p>
<p><b>Variations</b></p> <p>↑ MLBL(086)</p>	

**Description**

When the execution condition is OFF, MLBL(086) is not executed. When the execution condition is ON, MLBL(086) multiplies the 8-digit content of Md and Md+1 by the content of Mr and Mr+1, and places the result in R to R+3.



**Note** An equivalent symbol math instruction (\*UL(423)) is also available.

**Precautions**

Refer to page 101 for general precautions on operand data areas.

**Flags**

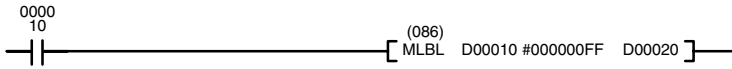
ER (A50003): Content of \*DM word is not BCD when set for BCD.

EQ (A50006): The result is 0.

N (A50008): Shows the status of bit 15 of R+3.

**Example**

When CIO 000010 is ON in the following example, the 8-digit content of D000010 and D00011 is multiplied by 0000 00FF. The 16-digit resultS is stored in D00020, D00021, D00022, and D00023.



Address	Instruction	Operands
00000	LD	000010
00001	MLBL(086)	
		D00010
		#000000FF
		D00020

D00011	D00010
0 9 5 A F 8 A A	

x

0 0 0 0 0 0 F F
-----------------

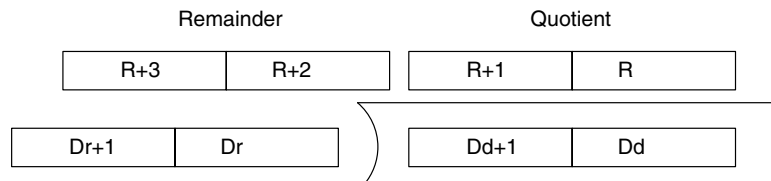
D00023	D00022	D00021	D00020
0 0 0 0	0 0 0 9	5 1 9 D	B 1 5 6

### 5-19-8 DOUBLE BINARY DIVIDE: DVBL(087)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(087)</p> <p>— [ DVBL Dd Dr R ]</p> <p><b>Variations</b></p> <p>↑ DVBL(087)</p>	<p><b>Operand Data Areas</b></p> <p><b>Dd: 1<sup>st</sup> dividend word</b> CIO, G, A, T, C, #, DM</p> <p><b>Dr: 1<sup>st</sup> divisor word</b> CIO, G, A, T, C, #, DM</p> <p><b>R: 1<sup>st</sup> result word</b> CIO, G, A, DM</p>
--	---

**Description**

When the execution condition is OFF, DVBL(087) is not executed. When the execution condition is ON, the 8-digit content of Dd and D+1 is divided by the content of Dr and Dr+1 and the result is placed in R to R+3: the quotient in R and R+1, and the remainder in R+2 and R+3.



**Note** An equivalent symbol math instruction (/UL(433)) is also available.

**Precautions**

Dr and Dr+1 must not contain 0.

Constants are expressed in eight digits.

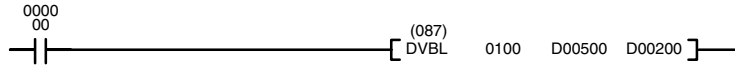
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

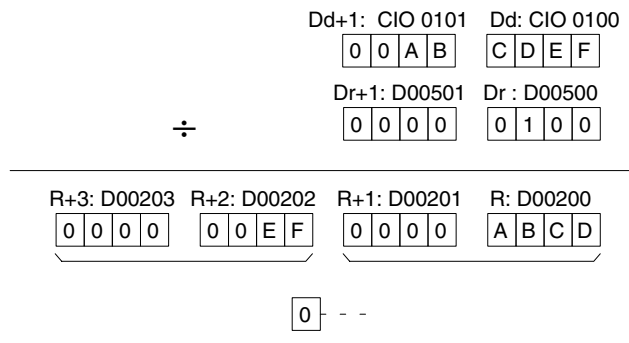
- ER (A50003): Dr and Dr+1 contain 0.  
Content of \*DM word is not BCD when set for BCD.
- EQ (A50006): The result is 0.
- N (A50008): Shows the status of bit 15 of R+1.

**Example**

When CIO 000000 is ON in the following example the content of CIO 0100 and CIO 0101 is divided by the content of D00500 and D00501 and the results is output to D00200 through D00203.



Address	Instruction	Operands
00000	LD	000000
00001	DVBL(087)	
		0100
		D00500
		D00200



## 5-20 Symbol Math Instructions

The Symbol Math Instructions perform arithmetic operations on BCD or binary data.

### 5-20-1 Binary Addition: +(400)/+L(401)/+C(402)/+CL(403)

#### SIGNED BINARY ADD WITHOUT CARRY: +(400)

<p><b>Ladder Symbol</b></p> <pre> (400) ┌───[ +   Au   Ad   R ] </pre> <p><b>Variations</b></p> <p>↑ +(400)</p>	<p><b>Operand Data Areas</b></p> <p><b>Au: Augend word</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>Ad: Addend word</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>R: Result word</b> CIO, G, A, DM, DR, IR</p>
---	---

#### DOUBLE SIGNED BINARY ADD WITHOUT CARRY: +L(401)

<p><b>Ladder Symbol</b></p> <pre> (401) ┌───[ +L   Au   Ad   R ] </pre> <p><b>Variations</b></p> <p>↑ +L(401)</p>	<p><b>Operand Data Areas</b></p> <p><b>Au: 1st augend word</b> CIO, G, A, T, C, #, DM</p> <p><b>Ad: 2nd addend word</b> CIO, G, A, T, C, #, DM</p> <p><b>R: 1st result word</b> CIO, G, A, DM</p>
---	---

#### SIGNED BINARY ADD WITH CARRY: +C(402)

<p><b>Ladder Symbol</b></p> <pre> (402) ┌───[ +C   Au   Ad   R ] </pre> <p><b>Variations</b></p> <p>↑ +C(402)</p>	<p><b>Operand Data Areas</b></p> <p><b>Au: Augend word</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>Ad: Addend word</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>R: Result word</b> CIO, G, A, DM, DR, IR</p>
---	---

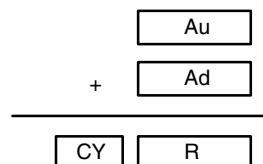
#### DOUBLE SIGNED BINARY ADD WITH CARRY: +CL(403)

<p><b>Ladder Symbol</b></p> <pre> (403) ┌───[ +CL   Au   Ad   R ] </pre> <p><b>Variations</b></p> <p>↑ +CL(403)</p>	<p><b>Operand Data Areas</b></p> <p><b>Au: 1st augend word</b> CIO, G, A, T, C, #, DM</p> <p><b>Ad: 2nd addend word</b> CIO, G, A, T, C, #, DM</p> <p><b>R: 1st result word</b> CIO, G, A, DM</p>
---	---

#### Description

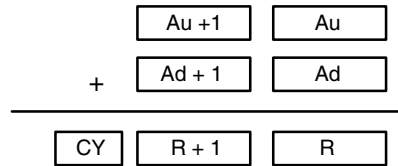
#### SIGNED BINARY ADD WITHOUT CARRY

When the execution condition is OFF, +(400) is not executed. When the execution condition is ON, +(400) adds the contents of Au and Ad and places the result in R. CY will be set if the result is greater than FFFF.

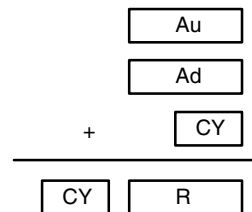


**DOUBLE SIGNED BINARY ADD WITHOUT CARRY**

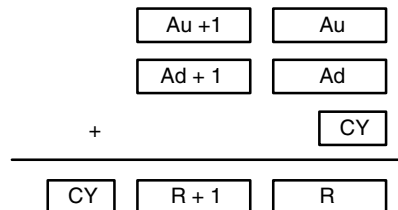
When the execution condition is OFF, +L(401) is not executed. When the execution condition is ON, +L(401) adds the 8-digit contents of Au+1 and Au and the 8-digit contents of Ad+1 and Ad, and places the result in R and R + 1. CY will be set if the result is greater than FFFF FFFF.

**SIGNED BINARY ADD WITH CARRY**

When the execution condition is OFF, +C(402) is not executed. When the execution condition is ON, +C(402) adds the contents of Au, Ad, and CY and places the result in R. CY will be set if the result is greater than FFFF.

**DOUBLE SIGNED BINARY ADD WITH CARRY**

When the execution condition is OFF, +CL(403) is not executed. When the execution condition is ON, +CL(403) adds the 8-digit contents of Au+1, Au, the 8-digit contents of Ad+1 and Ad, and CY, and places the result in R and R + 1. CY will be set if the result is greater than FFFF FFFF.

**Precautions**

Refer to page 101 for general precautions on operand data areas.

**Flags**

ER (A50003): Content of \*DM word is not BCD when set for BCD.

CY (A50004): The result is greater than FFFF or FFFF FFFF.

EQ (A50006): The result is 0.

N (A50008): Shows the status of bit 15 of R or R+1.

OF (A50009) Au (Au +1) and Ad (Ad +1) are both positive numbers and the result is negative.

UF (A50010) Au (Au +1) and Ad (Ad +1) are both negative numbers and the result is positive.

**Using Signed Binary Addition Instructions**

The range for signed data is -32,768 to 32,767 in decimal (-2,147,483,648 to 2,147,483,647 for "double" instructions), and 8000 to FFFF and 0000 to 7FFF in hexadecimal (8000 0000 to FFFF FFFF and 0000 0000 to 7FFF FFFF for "double" instructions).

Negative numbers are expressed as 2's complements. If the result of the addition is within the range of 8000 to FFFF, it represents a signed negative number and the Negative Flag (A50008) turns ON.

When Au and Ag are both positive numbers and the addition result is negative, the Overflow Flag (A50009) turns ON. When Au and Ag are both negative numbers and the addition result is positive, the Underflow Flag (A50010) turns ON. If a addition result in a carry, the Carry Flag turns ON.

The range for unsigned binary data is 0000 to FFFF (0000 0000 to FFFF FFFF for “double” instructions), so the decimal range would be 0 to 65,535 (0 to 4,294,967,295).

## 5-20-2 BCD Addition: +B(404)/ +BL(405)/+BC(406)/+BCL(407)

### BCD ADD WITHOUT CARRY: +B(404)

Ladder Symbol	Operand Data Areas
$\text{---} \overset{(404)}{\text{---}} \left[ \text{+B} \quad \text{Au} \quad \text{Ad} \quad \text{R} \right]$	<b>Au: Augend word</b> CIO, G, A, T, C, #, DM, DR, IR <b>Ad: Addend word</b> CIO, G, A, T, C, #, DM, DR, IR <b>R: Result word</b> CIO, G, A, DM, DR, IR
<b>Variations</b> ↑ +B(404)	

### DOUBLE BCD ADD WITHOUT CARRY: +BL(405)

Ladder Symbol	Operand Data Areas
$\text{---} \overset{(405)}{\text{---}} \left[ \text{+BL} \quad \text{Au} \quad \text{Ad} \quad \text{R} \right]$	<b>Au: 1st augend word</b> CIO, G, A, T, C, #, DM <b>Ad: 2nd addend word</b> CIO, G, A, T, C, #, DM <b>R: 1st result word</b> CIO, G, A, DM
<b>Variations</b> ↑ +BL(405)	

### BCD ADD WITH CARRY: +BC(406)

Ladder Symbol	Operand Data Areas
$\text{---} \overset{(406)}{\text{---}} \left[ \text{+BC} \quad \text{Au} \quad \text{Ad} \quad \text{R} \right]$	<b>Au: Augend word</b> CIO, G, A, T, C, #, DM, DR, IR <b>Ad: Addend word</b> CIO, G, A, T, C, #, DM, DR, IR <b>R: Result word</b> CIO, G, A, DM, DR, IR
<b>Variations</b> ↑ +BC(406)	

### DOUBLE BCD ADD WITH CARRY: +BCL(407)

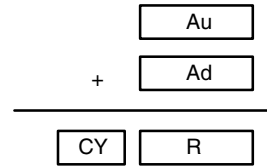
Ladder Symbol	Operand Data Areas
$\text{---} \overset{(407)}{\text{---}} \left[ \text{+BCL} \quad \text{Au} \quad \text{Ad} \quad \text{R} \right]$	<b>Au: 1st augend word</b> CIO, G, A, T, C, #, DM <b>Ad: 2nd addend word</b> CIO, G, A, T, C, #, DM <b>R: 1st result word</b> CIO, G, A, DM
<b>Variations</b> ↑ +BCL(407)	



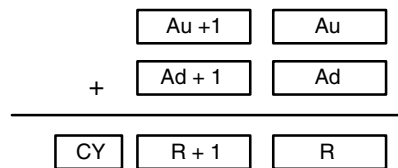
## Description

**BCD ADD WITHOUT CARRY**

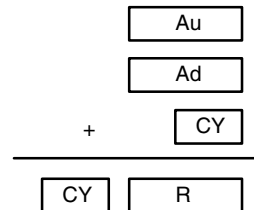
When the execution condition is OFF, +B(404) is not executed. When the execution condition is ON, +B(404) adds the contents of Au and Ad and places the result in R. CY will be set if the result is greater than 9999.

**DOUBLE BCD ADD WITHOUT CARRY**

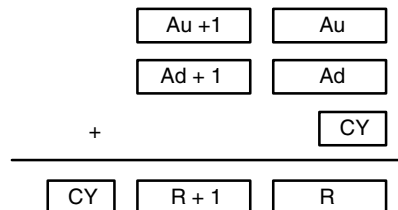
When the execution condition is OFF, +BL(405) is not executed. When the execution condition is ON, +BL(405) adds the 8-digit contents of Au+1 and Au and the 8-digit contents of Ad+1 and Ad, and places the result in R and R + 1. CY will be set if the result is greater than 9999 9999.

**BCD ADD WITH CARRY**

When the execution condition is OFF, +BC(406) is not executed. When the execution condition is ON, +BC(406) adds the contents of Au, Ad, and CY and places the result in R. CY will be set if the result is greater than 9999.

**DOUBLE BCD ADD WITH CARRY**

When the execution condition is OFF, +BCL(407) is not executed. When the execution condition is ON, +BCL(407) adds the 8-digit contents of Au+1, Au, the 8-digit contents of Ad+1 and Ad, and CY, and places the result in R and R + 1. CY will be set if the result is greater than 9999 9999.



## Precautions

Au and Ad (or Au, Au+1, Ad, and Ad+1) must be BCD. If any other data is used, the Error Flag (A50003) will turn ON and the instruction will not be executed.

**Note** Refer to page 101 for general precautions on operand data areas.

## Flags

- ER (A50003): Au and Ad (or Au, Au+1, Ad, and Ad+1) are not BCD.  
 The content of a\*DM word is not BCD when set for BCD.
- CY (A50004): The result exceed the digits.
- EQ(A50006): The result after the addition is all zeros.

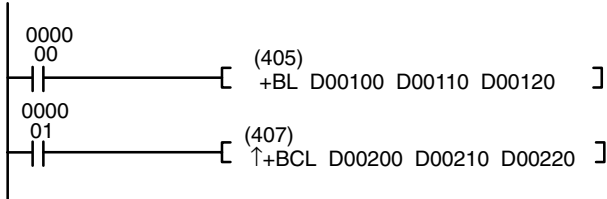
**Example**

**+BL Operation**

When CIO 00000 is ON in the following example, the contents of D00101 and D00100 are added to the content of D00111 and D00110, and the result is output in eight-digit BCD to D00121 and D00120.

**+BCL Operation**

When CIO 000001 is ON in the following example, the contents of D00201 and D00200 are added to the content of D00211 and D00210, and the result including the carry is output in eight-digit BCD to D00221 and D00220.



Address	Instruction	Operands
00000	LD	000000
00001	+BL(405)	
		D00100
		D00110
		D00120
00002	LD	000001
00003	+BCL(407)	
		D00200
		D00210
		D00220

**5-20-3 Binary Subtraction: -(410)/ -L(411)/-C(412)/-CL(413)**

**SIGNED BINARY SUBTRACT WITHOUT CARRY: -(410)**

Ladder Symbol	Operand Data Areas
$\text{---} \left[ \overset{(410)}{-} \text{ Mi Su R } \right]$	<b>Mi: Minuend word</b> CIO, G, A, T, C, #, DM, DR, IR <b>Su: Subtrahend word</b> CIO, G, A, T, C, #, DM, DR, IR <b>R: Result word</b> CIO, G, A, DM, DR, IR
<b>Variations</b> ↑-(410)	

**DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY: -L(411)**

Ladder Symbol	Operand Data Areas
$\text{---} \left[ \overset{(411)}{-L} \text{ Mi Su R } \right]$	<b>Mi: Minuend word</b> CIO, G, A, T, C, #, DM, <b>Su: Subtrahend word</b> CIO, G, A, T, C, #, DM, <b>R: Result word</b> CIO, G, A, DM,
<b>Variations</b> ↑-L(411)	

**SIGNED BINARY SUBTRACT WITH CARRY: -C(412)**

Ladder Symbol	Operand Data Areas
$\text{---} \left[ \overset{(412)}{-C} \text{ Mi Su R } \right]$	<b>Mi: Minuend word</b> CIO, G, A, T, C, #, DM, DR, IR <b>Su: Subtrahend word</b> CIO, G, A, T, C, #, DM, DR, IR <b>R: Result word</b> CIO, G, A, DM, DR, IR
<b>Variations</b> ↑-C(412)	

**DOUBLE SIGNED BINARY SUBTRACT WITH CARRY: -CL(413)**

<p><b>Ladder Symbol</b></p> <p>— [ <sup>(413)</sup> -CL Mi Su R ]</p> <p><b>Variations</b></p> <p>↑ -CL(413)</p>	<p><b>Operand Data Areas</b></p> <p><b>Mi: Minuend word</b> CIO, G, A, T, C, #, DM,</p> <p><b>Su: Subtrahend word</b> CIO, G, A, T, C, #, DM,</p> <p><b>R: Result word</b> CIO, G, A, DM,</p>
--	---

**Description**

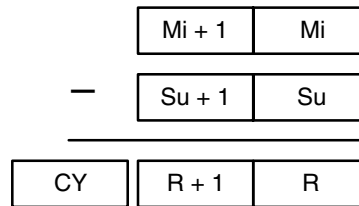
**SIGNED BINARY SUBTRACT WITHOUT CARRY**

When the execution condition is OFF, -(410) is not executed. When the execution condition is ON, -(410) subtracts the contents of Su from Mi and places the result in R. If the subtraction resulted in a borrow, CY is set. To obtain the true answer when the result is negative, the 2's complement placed in R must be subtracted from 0000.



**DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY**

When the execution condition is OFF, -L(411) is not executed. When the execution condition is ON, -L(411) subtracts the 8-digit value in Su and Su+1 from the 8-digit value in Mi and Mi+1, and places the result in R and R+1. If the subtraction resulted in a borrow, CY is set.



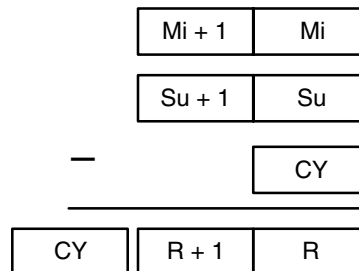
**SIGNED BINARY SUBTRACT WITH CARRY**

When the execution condition is OFF, -C(412) is not executed. When the execution condition is ON, -C(412) subtracts the contents of Su and CY from Mi and places the result in R. If the subtraction resulted in a borrow, CY is set.



**DOUBLE SIGNED BINARY SUBTRACT WITH CARRY**

When the execution condition is OFF, -CL(413) is not executed. When the execution condition is ON, -CL(413) subtracts CY and the 8-digit value in Su and Su+1 from the 8-digit value in Mi and Mi+1, and places the result in R and R+1. If the subtraction resulted in a borrow, CY is set.



**Precautions**

Refer to page 101 for general precautions on operand data areas.

<b>Flags</b>	ER (A50003):	The content of a*DM word is not BCD when set for BCD.
	CY (A50004):	The subtraction resulted in a borrow.
	EQ(A50006)	The contents of word R (or word R and R+1 for “double” instructions) after the subtraction is all zeros
	N (A50008)	The leftmost bit (MSB) of word R (or word R+1 for “double” instructions) after the subtraction is “1.”
	OF (A50009)	Mi is a positive number, Su is negative, and the subtraction result is negative.
	UF (A50010)	Mi is a negative number, Su is positive, and the subtraction result is positive.

**Using SIGNED BINARY SUBTRACT Instructions**

The range for signed data is -32,768 to 32,767 in decimal (-2,147,483,648 to 2,147,483,647 for “double” instructions), and 8000 to FFFF and 0000 to 7FFF in hexadecimal (8000 0000 to FFFF FFFF and 0000 0000 to 7FFF FFFF for “double” instructions).

Negative numbers are expressed as 2’s complements. If the result of the subtraction is within the range of 8000 to FFFF, it represents a signed negative number and the Negative Flag (A50008) turns ON.

When Mi is a positive number, Su is negative and the subtraction result is negative, the Overflow Flag (A50009) turns ON. When Mi is a negative number, Su is positive, and the subtraction result is positive, the Underflow Flag (A50010) turns ON. If a subtraction result in a borrow, the Carry Flag turns ON.

The range for unsigned binary data is 0000 to FFFF (0000 0000 to FFFF FFFF for “double” instructions), so the decimal range would be 0 to 65,535 (0 to 4,294,967,295). When data is unsigned, the Carry Flag turning ON indicates that the subtraction result is negative. The result is expressed as 2’s complement, so in order to find the true answer, the 2’s complement must be subtracted from 0.

**Numeric Example 1**

	Signed data	Unsigned data
FFFF <sub>H</sub> →	-1	65535
-) 0001 <sub>H</sub> →	-) +1	-) 1
FFFE <sub>H</sub> →	-2 (Note 1)	65534 (Note 2)
Negative Flag ON Carry Flag OFF		

**Numeric Example 2**

	Signed data	Unsigned data
FFFD <sub>H</sub> →	-3	65533
-) FFFF <sub>H</sub> →	-) -1	-) 65535
FFFE <sub>H</sub> →	-2 (Note 1)	-2 (Note 3)
Negative Flag ON Carry Flag ON		

- Note**
1. Because the Negative Flag is ON, the result (FFFE) is a negative number (2’s complement) and is expressed as -2.
  2. The Carry Flag is OFF and the result (FFFE) is an unsigned positive number (65,534).
  3. The Carry Flag is ON so the result (FFFE) is an unsigned negative number (2’s complement) and becomes -2 when converted.

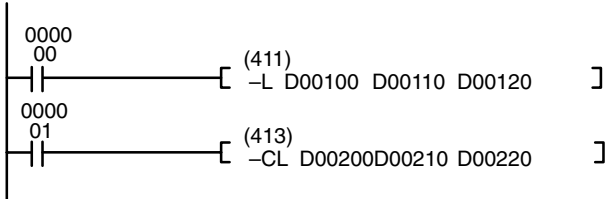
**Programming Example 1**

**-L Operation**

When CIO 00000 is ON in the following example, the content of D00111 and D00110 is subtracted from the content of D00101 and D00100, and the result is output in eight-digit binary to D00121 and D00120. CY is set if the subtraction resulted in a borrow.

**-CL Operation**

When CIO 000001 is ON in the following example, the content of D00211 and D00210 is subtracted from the content of D00201 and D00200, and the result is output in eight-digit binary to D00221 and D00220. CY is set if the subtraction resulted in a borrow.



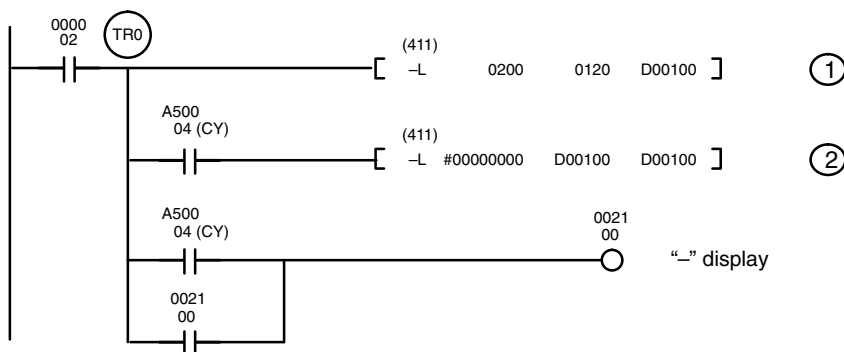
Address	Instruction	Operands
00000	LD	000000
00001	-L(411)	
		D00100
		D00110
		D00120
00002	LD	000001
00003	-CL(413)	
		D00200
		D00210
		D00220

**Program Example 2**

Example (unsigned data): 20F55A10 – B8A360E3 = –97AE06D3.

In this example, the eight-digit binary value in CIO 0121 and CIO 0120 is subtracted from the value in CIO 0201 and CIO 0200, and the result is output to eight-digit binary in D00101 and D00100. If the result is negative, the instruction at (2) will be executed, and the actual result will then be output to D00101 and D00100.

The Carry Flag (A50004) will be turned ON, so the actual number is –97AE06D3. Because the content of D00101 and D00100 is negative, CY is used to turn ON a self-holding bit that turns ON a bit indicating a negative value.



Address	Instruction	Operands
00000	LD	000002
00001	OUT	TR0
00002	-L(411)	
		0200
		0120
		D00100
00003	LD	TR0
00004	AND	A50004
00005	-L(411)	
		#00000000
		D00100
		D00100
00006	LD	TR0
00007	AND	A50004
00008	OR	002100
00009	OUT	002100

**Subtraction at 1**

Mi+1: CIO 0201	Mi: CIO 0200
2 0 F 5	5 A 1 0
Su+1: CIO 0121	Su: CIO 0120
- B 8 A 3	6 0 E 3

---

CY R+1: D00101	R+1: D00100
1 6 8 5 1	F 9 2 D

The Carry Flag (A50004) is ON, so the result is subtracted from 0000 0000 to obtain the actual result.

**Subtraction at 2**

0 0 0 0	0 0 0 0
Su+1: D00101	Su: D00100
- 6 8 5 1	F 9 2 D

---

CY R+1: D00101	R+1: D00100
1 9 7 A E	0 6 D 3

**Final Subtraction Result**

Mi+1: CIO 0201	Mi: CIO 0200
2 0 F 5	5 A 1 0
Su+1: D00101	Su: D00100
- 6 8 5 1	F 9 2 D

---

CY R+1: D00101	R+1: D00100
1 9 7 A E	0 6 D 3

**5-20-4 BCD Subtraction: –B(414)/ –BL(415)/–BC(416)/–BCL(417)****BCD SUBTRACT WITHOUT CARRY: –B(414)**

Ladder Symbol	Operand Data Areas
$\text{---} \overset{(414)}{[ -B \quad \text{Mi} \quad \text{Su} \quad \text{R} ]}$	<b>Mi: Minuend word</b> CIO, G, A, T, C, #, DM, DR, IR <b>Su: Subtrahend word</b> CIO, G, A, T, C, #, DM, DR, IR <b>R: Result word</b> CIO, G, A, DM, DR, IR
<b>Variations</b> ↑ –B(414)	

**DOUBLE BCD SUBTRACT WITHOUT CARRY: –BL(415)**

Ladder Symbol	Operand Data Areas
$\text{---} \overset{(415)}{[ -BL \quad \text{Mi} \quad \text{Su} \quad \text{R} ]}$	<b>Mi: 1<sup>st</sup> minuend word</b> CIO, G, A, T, C, #, DM <b>Su: 1<sup>st</sup> subtrahend word</b> CIO, G, A, T, C, #, DM <b>R: 1<sup>st</sup> result word</b> CIO, G, A, DM
<b>Variations</b> ↑ –BL(415)	

**BCD SUBTRACT WITH CARRY: –BC(416)**

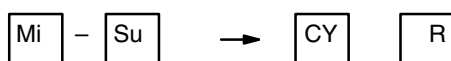
Ladder Symbol	Operand Data Areas
$\text{---} \overset{(416)}{[ -BC \quad \text{Mi} \quad \text{Su} \quad \text{R} ]}$	<b>Mi: Minuend word</b> CIO, G, A, T, C, #, DM, DR, IR <b>Su: Subtrahend word</b> CIO, G, A, T, C, #, DM, DR, IR <b>R: Result word</b> CIO, G, A, DM, DR, IR
<b>Variations</b> ↑ –BC(416)	

**DOUBLE BCD SUBTRACT WITH CARRY: –BCL(417)**

Ladder Symbol	Operand Data Areas
$\text{---} \overset{(417)}{[ -BCL \quad \text{Mi} \quad \text{Su} \quad \text{R} ]}$	<b>Mi: 1<sup>st</sup> minuend word</b> CIO, G, A, T, C, #, DM <b>Su: 1<sup>st</sup> subtrahend word</b> CIO, G, A, T, C, #, DM <b>R: 1<sup>st</sup> result word</b> CIO, G, A, DM
<b>Variations</b> ↑ –BCL(417)	

**Description****BCD SUBTRACT WITHOUT CARRY**

When the execution condition is OFF, –B(414) is not executed. When the execution condition is ON, –B(414) subtracts the BCD contents of Su from Mi, and places the result in R. If the result is negative, CY is set and the 10's complement of the actual result is placed in R. To convert the 10's complement to the true result, subtract the content of R from 0000.



**DOUBLE BCD SUBTRACT WITHOUT CARRY**

When the execution condition is OFF,  $-BL(415)$  is not executed. When the execution condition is ON,  $-BL(415)$  subtracts the 8-digit BCD content of Su and Su+1 from the 8-digit BCD content in Mi and Mi+1, and places the result in R and R+1. If the result is negative, CY is set and the 10's complement of the actual result is placed in R. To convert the 10's complement to the true result, subtract the content of R from 0000 0000.

$$\begin{array}{r}
 \begin{array}{|c|c|} \hline Mi + 1 & Mi \\ \hline \end{array} \\
 - \begin{array}{|c|c|} \hline Su + 1 & Su \\ \hline \end{array} \\
 \hline
 \begin{array}{|c|c|c|} \hline CY & R + 1 & R \\ \hline \end{array}
 \end{array}$$

**BCD SUBTRACT WITH CARRY**

When the execution condition is OFF,  $-BC(416)$  is not executed. When the execution condition is ON,  $-BC(416)$  subtracts the BCD contents of Su and CY from Mi, and places the result in R. If the result is negative, CY is set and the 10's complement of the actual result is placed in R. To convert the 10's complement to the true result, subtract the content of R from 0000.

$$\begin{array}{|c|} \hline Mi \\ \hline \end{array}
 - \begin{array}{|c|} \hline Su \\ \hline \end{array}
 - \begin{array}{|c|} \hline CY \\ \hline \end{array}
 \rightarrow \begin{array}{|c|} \hline CY \\ \hline \end{array}
 \begin{array}{|c|} \hline R \\ \hline \end{array}$$

**DOUBLE BCD SUBTRACT WITH CARRY**

When the execution condition is OFF,  $-BCL(417)$  is not executed. When the execution condition is ON,  $-BCL(417)$  subtracts CY and the 8-digit BCD content of Su and Su+1 from the 8-digit BCD content in Mi and Mi+1, and places the result in R and R+1. If the result is negative, CY is set and the 10's complement of the actual result is placed in R. To convert the 10's complement to the true result, subtract the content of R from 0000 0000.

$$\begin{array}{r}
 \begin{array}{|c|c|} \hline Mi + 1 & Mi \\ \hline \end{array} \\
 \begin{array}{|c|c|} \hline Su + 1 & Su \\ \hline \end{array} \\
 - \begin{array}{|c|} \hline CY \\ \hline \end{array} \\
 \hline
 \begin{array}{|c|c|c|} \hline CY & R + 1 & R \\ \hline \end{array}
 \end{array}$$

**Precautions**

Mi and Su (or Mi, Mi+1, Su, and Su+1) must be BCD. If any other data is used, the Error Flag (A50003) will turn ON and the instruction will not be executed.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

ER (A50003): Mi and Su (or Mi, Mi+1, Su, and Su+1) are not BCD.  
 The content of a\*DM word is not BCD when set for BCD.  
 CY (A50004): Subtraction result exceed the digits.  
 EQ(A50006): The result after the subtraction is all zeros.

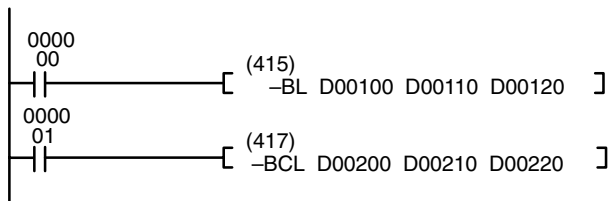
**Example****-BL Operation**

When CIO 000000 is ON in the following example, the content of D00111 and D00110 is subtracted from the content of D00101 and D00100, and the result is output in eight-digit BCD to D00121 and D00120. CY is set if the result is negative



**-BCL Operation**

When CIO 00001 is ON in the following example, the content of D00211 and D00210 are subtracted from the content of D00201 and D00200, and the result including the carry is output in eight-digit BCD to D00221 and D00220. CY is set if the result is negative



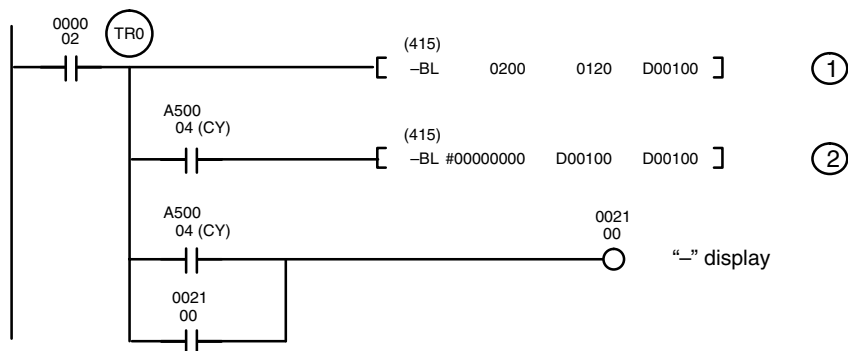
Address	Instruction	Operands
00000	LD	000000
00001	-BL(415)	
		D00100
		D00110
		D00120
00002	LD	000001
00003	-BCL(417)	
		D00200
		D00210
		D00220

**Program Example**

Example:  $9,583,960 - 17,072,641 = -7,488,681$ .

In this example, the eight-digit BCD content of CIO 0121 and CIO 0120 is subtracted from the content of CIO 0201 and CIO 0200, and the result is output in eight-digit BCD to D00101 and D00100. The result is negative, so the instruction at (2) will be executed, and the true value will then be output to D00101 and D00100.

The Carry Flag (A50004) will be turned ON, so the actual number is  $-7,488,681$ . Because the content of D00101 and D00100 is negative, CY is used to turn ON a self-holding bit that turns ON a bit indicating a negative value.



Address	Instruction	Operands
00000	LD	000002
00001	OUT	TR0
00002	-BL(415)	
		0200
		0120
		D00100
00003	LD	TR0
00004	AND	A50004
00005	-BL(415)	
		#00000000
		D00100
		D00100
00006	LD	TR0
00007	AND	A50004
00008	OR	002100
00009	OUT	002100

**Subtraction at 1**

Mi+1: CIO 0201	Mi: CIO 0200								
<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="width: 25%;">0</td><td style="width: 25%;">9</td><td style="width: 25%;">5</td><td style="width: 25%;">8</td></tr> </table>	0	9	5	8	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="width: 25%;">3</td><td style="width: 25%;">9</td><td style="width: 25%;">6</td><td style="width: 25%;">0</td></tr> </table>	3	9	6	0
0	9	5	8						
3	9	6	0						
Su+1: CIO 0121	Su: CIO 0120								
<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="width: 25%;">1</td><td style="width: 25%;">7</td><td style="width: 25%;">0</td><td style="width: 25%;">7</td></tr> </table>	1	7	0	7	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="width: 25%;">2</td><td style="width: 25%;">6</td><td style="width: 25%;">4</td><td style="width: 25%;">1</td></tr> </table>	2	6	4	1
1	7	0	7						
2	6	4	1						

---

09583960 + (100000000 - 17072641)

CY R+1: D00101	R+1: D00100									
<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="width: 25%;">1</td><td style="width: 25%;">9</td><td style="width: 25%;">2</td><td style="width: 25%;">5</td><td style="width: 25%;">1</td></tr> </table>	1	9	2	5	1	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="width: 25%;">1</td><td style="width: 25%;">3</td><td style="width: 25%;">1</td><td style="width: 25%;">9</td></tr> </table>	1	3	1	9
1	9	2	5	1						
1	3	1	9							

The Carry Flag (A50004) is ON, so the result is subtracted from 0000 0000.

**Subtraction at 2**

<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="width: 25%;">0</td><td style="width: 25%;">0</td><td style="width: 25%;">0</td><td style="width: 25%;">0</td></tr> </table>	0	0	0	0	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="width: 25%;">0</td><td style="width: 25%;">0</td><td style="width: 25%;">0</td><td style="width: 25%;">0</td></tr> </table>	0	0	0	0
0	0	0	0						
0	0	0	0						
Su+1: D00101	Su: D00100								
<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="width: 25%;">9</td><td style="width: 25%;">2</td><td style="width: 25%;">5</td><td style="width: 25%;">1</td></tr> </table>	9	2	5	1	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="width: 25%;">1</td><td style="width: 25%;">3</td><td style="width: 25%;">1</td><td style="width: 25%;">9</td></tr> </table>	1	3	1	9
9	2	5	1						
1	3	1	9						

---

00000000 + (100000000 - 92511319)

CY R+1: D00101	R+1: D00100									
<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="width: 25%;">1</td><td style="width: 25%;">0</td><td style="width: 25%;">7</td><td style="width: 25%;">4</td><td style="width: 25%;">8</td></tr> </table>	1	0	7	4	8	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="width: 25%;">8</td><td style="width: 25%;">6</td><td style="width: 25%;">8</td><td style="width: 25%;">1</td></tr> </table>	8	6	8	1
1	0	7	4	8						
8	6	8	1							

**Final Subtraction Result**

Mi+1: CIO 0201	Mi: CIO 0200								
<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="width: 25%;">2</td><td style="width: 25%;">0</td><td style="width: 25%;">F</td><td style="width: 25%;">5</td></tr> </table>	2	0	F	5	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="width: 25%;">5</td><td style="width: 25%;">A</td><td style="width: 25%;">1</td><td style="width: 25%;">0</td></tr> </table>	5	A	1	0
2	0	F	5						
5	A	1	0						
Su+1: D00101	Su: D00100								
<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="width: 25%;">6</td><td style="width: 25%;">8</td><td style="width: 25%;">5</td><td style="width: 25%;">1</td></tr> </table>	6	8	5	1	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="width: 25%;">F</td><td style="width: 25%;">9</td><td style="width: 25%;">2</td><td style="width: 25%;">D</td></tr> </table>	F	9	2	D
6	8	5	1						
F	9	2	D						

---

CY R+1: D00101	R+1: D00100									
<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="width: 25%;">1</td><td style="width: 25%;">0</td><td style="width: 25%;">7</td><td style="width: 25%;">4</td><td style="width: 25%;">8</td></tr> </table>	1	0	7	4	8	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="width: 25%;">8</td><td style="width: 25%;">6</td><td style="width: 25%;">8</td><td style="width: 25%;">1</td></tr> </table>	8	6	8	1
1	0	7	4	8						
8	6	8	1							

### 5-20-5 Binary Multiplication: \*(420)/ \*L(421)/\*U(422)/\*UL(423)

#### SIGNED BINARY MULTIPLY: \*(420)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(420)</p> <p>— [ *      Md      Mr      R ]</p> <p><b>Variations</b></p> <p>↑ *(420)</p>	<p><b>Operand Data Areas</b></p> <p><b>Md: Multiplicand word</b>    CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>Mr: Multiplier word</b>      CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>R: Result word</b>            CIO, G, A, DM</p>
---	--

#### DOUBLE SIGNED BINARY MULTIPLY: \*L(421)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(421)</p> <p>— [ *L      Md      Mr      R ]</p> <p><b>Variations</b></p> <p>↑ *L(421)</p>	<p><b>Operand Data Areas</b></p> <p><b>Md: 1<sup>st</sup> multiplicand wd</b>    CIO, G, A, T, C, #, DM</p> <p><b>Mr: 1<sup>st</sup> multiplier word</b>    CIO, G, A, T, C, #, DM</p> <p><b>R: 1<sup>st</sup> result word</b>        CIO, G, A, DM</p>
---	---

#### UNSIGNED BINARY MULTIPLY: \*U(422)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(422)</p> <p>— [ *U      Md      Mr      R ]</p> <p><b>Variations</b></p> <p>↑ *U(422)</p>	<p><b>Operand Data Areas</b></p> <p><b>Md: Multiplicand word</b>    CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>Mr: Multiplier word</b>      CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>R: Result word</b>            CIO, G, A, DM</p>
---	--

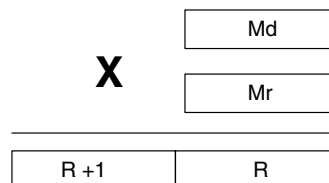
#### DOUBLE UNSIGNED BINARY MULTIPLY: \*UL(423)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(423)</p> <p>— [ *UL      Md      Mr      R ]</p> <p><b>Variations</b></p> <p>↑ *UL(423)</p>	<p><b>Operand Data Areas</b></p> <p><b>Md: 1<sup>st</sup> multiplicand wd</b>    CIO, G, A, T, C, #, DM</p> <p><b>Mr: 1<sup>st</sup> multiplier word</b>    CIO, G, A, T, C, #, DM</p> <p><b>R: 1<sup>st</sup> result word</b>        CIO, G, A, DM</p>
---	---

**Description**

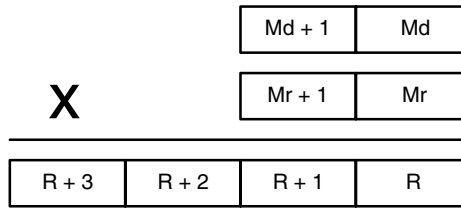
**SIGNED BINARY MULTIPLY**

When the execution condition is OFF, \*(420) is not executed. When the execution condition is ON, \*(420) multiplies the signed content of Md by the signed content of Mr, places the rightmost four digits of the result in R, and places the leftmost four digits in R+1.



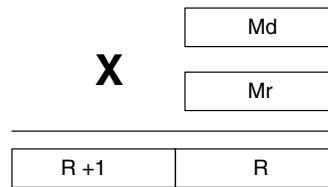
**DOUBLE SIGNED BINARY MULTIPLY**

When the execution condition is OFF, \*L(421) is not executed. When the execution condition is ON, \*L(421) multiplies the signed 8-digit content of Md and Md+1 by the signed content of Mr and Mr+1, and places the result in R to R+3.



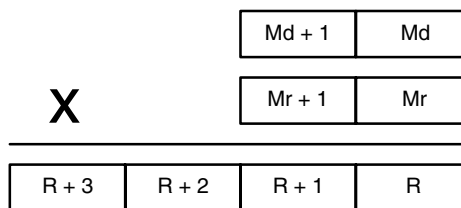
**UNSIGNED BINARY MULTIPLY**

When the execution condition is OFF, \*U(422) is not executed. When the execution condition is ON, \*U(422) multiplies the unsigned content of Md by the unsigned content of Mr, places the rightmost four digits of the result in R, and places the leftmost four digits in R+1.



**DOUBLE UNSIGNED BINARY MULTIPLY**

When the execution condition is OFF, \*UL(423) is not executed. When the execution condition is ON, \*UL(423) multiplies the unsigned 8-digit content of Md and Md+1 by the unsigned content of Mr and Mr+1, and places the result in R to R+3.



**Precautions**

Refer to page 101 for general precautions on operand data areas.

**Flags**

- ER (A50003): The content of a\*DM word is not BCD when set for BCD.
- EQ(A50006) The multiplication result is all zeroes.
- N (A50008) The leftmost bit (MSB) of word R+1 (or word R+3 for “double” instructions) after the multiplication is “1.”

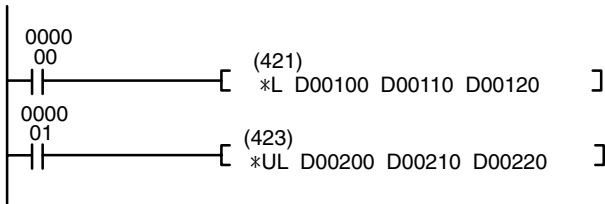
**Example**

**\*L Operation**

When CIO 000000 is ON in the following example, the content of D00101 and D00100 are multiplied by the content of D00111 and D00110, in eight-digit binary with sign, and the result is output to D00123 through D00120.

**\*UL Operation**

When CIO 000001 is ON in the following example, the content of D00201 and D00200 are multiplied by the content of D00211 and D00210, in eight-digit binary without sign, and the result is output to D00223 through D00220.



Address	Instruction	Operands
00000	LD	000000
00001	*L(421)	
		D00100
		D00110
		D00120
00002	LD	000001
00003	*UL(423)	
		D00200
		D00210
		D00220

**5-20-6 BCD Multiplication: \*B(424)/ \*BL(425)**

**BCD MULTIPLY: \*B(424)**

<p><b>Ladder Symbol</b></p> <p>— (424) [ *B Md Mr R ]</p> <p><b>Variations</b></p> <p>↑ *B(424)</p>	<p><b>Operand Data Areas</b></p> <p><b>Md: Multiplicand word</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>Mr: Multiplier word</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>R: Result word</b> CIO, G, A, DM</p>
---	---

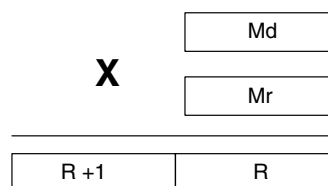
**DOUBLE BCD MULTIPLY: \*BL(425)**

<p><b>Ladder Symbol</b></p> <p>— (425) [ *BL Md Mr R ]</p> <p><b>Variations</b></p> <p>↑ *BL(425)</p>	<p><b>Operand Data Areas</b></p> <p><b>Md: 1<sup>st</sup> multiplicand wd</b> CIO, G, A, T, C, #, DM</p> <p><b>Mr: 1<sup>st</sup> multiplier word</b> CIO, G, A, T, C, #, DM</p> <p><b>R: 1<sup>st</sup> result word</b> CIO, G, A, DM</p>
---	--

**Description**

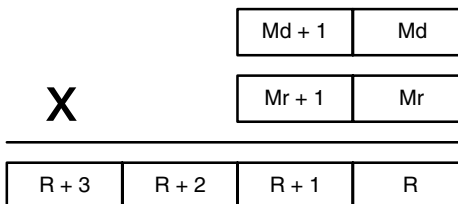
**BCD MULTIPLY**

When the execution condition is OFF, \*B(424) is not executed. When the execution condition is ON, \*B(424) multiplies the BCD content of Md by the BCD content of Mr, and places the result in R and R+1.



**DOUBLE BCD MULTIPLY**

When the execution condition is OFF, \*BL(425) is not executed. When the execution condition is ON, \*BL(425) multiplies the 8-digit BCD content of Md and Md+1 by the BCD content of Mr and Mr+1, and places the result in R to R+3.



**Precautions**

Md (Md+1) and Mr (Mr+1) must be BCD. If any other data is used, the Error Flag (A50003) will turn ON and the instruction will not be executed.

**Note** Refer to page 101 for general precautions on operand data areas.

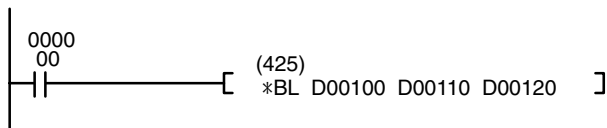
**Flags**

- ER (A50003): Content of Md (Md+1) or Mr (Mr+1) is not BCD.  
The content of a \*DM word is not BCD when set for BCD.
- CY (A50004): There is a carry in the result.
- EQ (A50006): The result is all zeros.

**Example**

**\*BL Operation**

When CIO 000000 is ON in the following example, the content of D00101 and D00100 IS multiplied by the content of D00111 and D00110, in eight-digit BCD, and the result is output to D00123 through D00120.



Address	Instruction	Operands
00000	LD	000000
00001	*BL(425)	
		D00100
		D00110
		D00120

### 5-20-7 Binary Division: /(430)/ /L(431)//U(432)//UL(433)

#### SIGNED BINARY DIVIDE: /(430)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(430)</p> <p>— [ /      Dd      Dr      R ]</p> <p><b>Variations</b></p> <p>↑ /(430)</p>	<p><b>Operand Data Areas</b></p> <p><b>Dd: Dividend word</b>    CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>Dr: Divisor word</b>     CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>R: Result word</b>        CIO, G, A, DM</p>
---	--

#### DOUBLE SIGNED BINARY DIVIDE: /L(431)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(431)</p> <p>— [ /L      Dd      Dr      R ]</p> <p><b>Variations</b></p> <p>↑ /L(431)</p>	<p><b>Operand Data Areas</b></p> <p><b>Dd: 1<sup>st</sup> dividend word</b>    CIO, G, A, T, C, #, DM</p> <p><b>Dr: 1<sup>st</sup> divisor word</b>        CIO, G, A, T, C, #, DM</p> <p><b>R: 1<sup>st</sup> result word</b>         CIO, G, A, DM</p>
---	---

#### UNSIGNED BINARY DIVIDE: /U(432)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(432)</p> <p>— [ /U      Dd      Dr      R ]</p> <p><b>Variations</b></p> <p>↑ /U(432)</p>	<p><b>Operand Data Areas</b></p> <p><b>Dd: Dividend word</b>    CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>Dr: Divisor word</b>     CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>R: Result word</b>        CIO, G, A, DM</p>
---	--

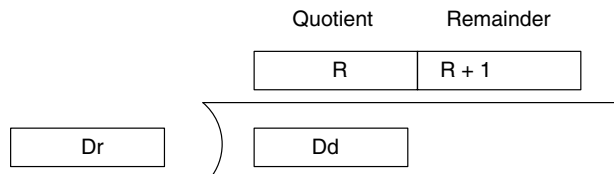
#### DOUBLE UNSIGNED BINARY DIVIDE: /UL(433)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(433)</p> <p>— [ /UL      Dd      Dr      R ]</p> <p><b>Variations</b></p> <p>↑ /UL(433)</p>	<p><b>Operand Data Areas</b></p> <p><b>Dd: 1<sup>st</sup> dividend word</b>    CIO, G, A, T, C, #, DM</p> <p><b>Dr: 1<sup>st</sup> divisor word</b>        CIO, G, A, T, C, #, DM</p> <p><b>R: 1<sup>st</sup> result word</b>         CIO, G, A, DM</p>
---	---

**Description**

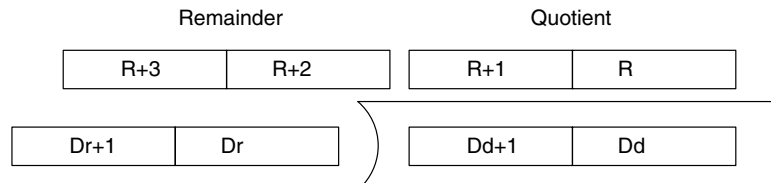
**SIGNED BINARY DIVIDE**

When the execution condition is OFF, /(430) is not executed. When the execution condition is ON, /(430) divides the signed binary content of Dd by the signed binary content of Dr and the result is placed in R and R+1: the quotient in R, the remainder in R+1.



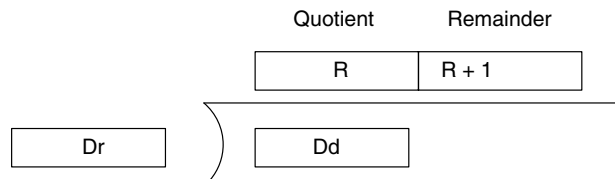
**DOUBLE SIGNED BINARY DIVIDE**

When the execution condition is OFF, /L(431) is not executed. When the execution condition is ON, /L(431) divides the signed 8-digit content of Dd and D+1 by the signed content of Dr and Dr+1 and the result is placed in R to R+3: the quotient in R and R+1, and the remainder in R+2 and R+3.



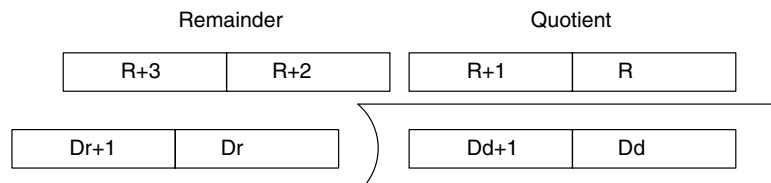
**UNSIGNED BINARY DIVIDE**

When the execution condition is OFF, /U(432) is not executed. When the execution condition is ON, /U(432) divides the unsigned content of Dd by the unsigned content of Dr and the result is placed in R and R+1: the quotient in R, the remainder in R+1.



**DOUBLE UNSIGNED BINARY DIVIDE**

When the execution condition is OFF, /UL(433) is not executed. When the execution condition is ON, /UL(433) divides the 8-digit unsigned content of Dd and D+1 by the unsigned content of Dr and Dr+1 and the result is placed in R to R+3: the quotient in R and R+1, and the remainder in R+2 and R+3.



**Precautions**

S<sub>2</sub> (or S<sub>2</sub>, and S<sub>2+1</sub>) must not be all zeros.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

- ER (A50003): Dr (or Dr and Dr+1) is all zeroes.  
The content of a\*DM word is not BCD when set for BCD.
- EQ(A50006) The division result is all zeroes in the quotient.
- N (A50008) The leftmost bit (MSB) of word R (or word R+1 for “double” instructions) after the division is “1.”

**Example**

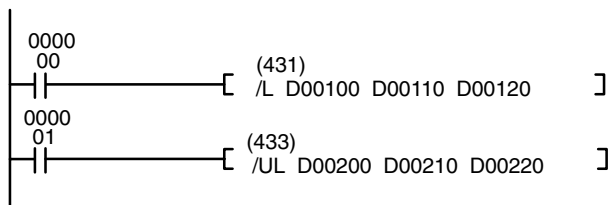
**/L Operation**

When CIO 000000 is ON in the following example, the signed content of D00101 and D00100 is divided by the signed content of D00111 and D00110, in eight-digit binary. When the result is obtained, the quotient is output to D00121 and D00120, and the remainder is output to D00123 and D00122.



**/UL Operation**

When CIO 00001 is ON in the following example, the unsigned content of D00201 and D00200 is divided by the unsigned content of D00211 and D00210, in eight-digit binary. When the result is obtained, the quotient is output to D00221 and D00220, and the remainder is output to D00223 and D00222.



Address	Instruction	Operands
00000	LD	000000
00001	/L(431)	
		D00100
		D00110
		D00120
00002	LD	000001
00003	/UL(433)	
		D00200
		D00210
		D00220

**5-20-8 BCD Division: /B(434)/ /BL(435)**

**BCD DIVIDE: /B(434)**

Ladder Symbol	Operand Data Areas
	<b>Dd: Dividend word</b> CIO, G, A, T, C, #, DM, DR, IR <b>Dr: Divisor word</b> CIO, G, A, T, C, #, DM, DR, IR <b>R: Result word</b> CIO, G, A, DM
<b>Variations</b> ↑ /B(434)	

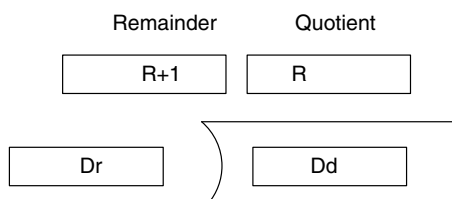
**DOUBLE BCD DIVIDE: /BL(435)**

Ladder Symbol	Operand Data Areas
	<b>Dd: 1<sup>st</sup> dividend word</b> CIO, G, A, T, C, #, DM <b>Dr: 1<sup>st</sup> divisor word</b> CIO, G, A, T, C, #, DM <b>R: 1<sup>st</sup> result word</b> CIO, G, A, DM
<b>Variations</b> ↑ /BL(435)	

**Description**

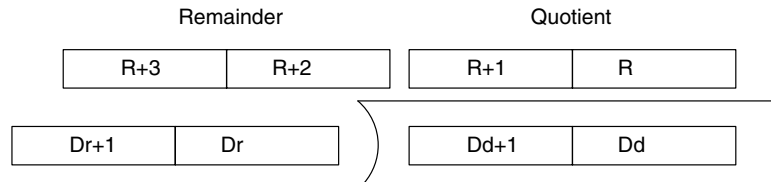
**BCD DIVIDE**

When the execution condition is OFF, /B(434) is not executed and the program moves to the next instruction. When the execution condition is ON, the BCD content of Dd is divided by the BCD content of Dr and the result is placed in R and R + 1: the quotient in R and the remainder in R + 1.



**DOUBLE BCD DIVIDE**

When the execution condition is OFF, /BL(435) is not executed. When the execution condition is ON, the BCD 8-digit content of Dd and D+1 is divided by the BCD content of Dr and Dr+1 and the result is placed in R to R+3: the quotient in R and R+1, and the remainder in R+2 and R+3.



**Precautions**

Dd and Dr (or Dd, Dd+1, Dr, and Dr+1) must be BCD. If any other data is used, the Error Flag (A50003) will turn ON and the instruction will not be executed.

**Note** Refer to page 101 for general precautions on operand data areas.

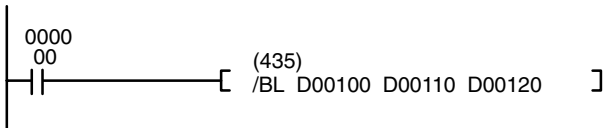
**Flags**

- ER (A50003): Dd and Dr (or Dd, Dd+1, Dr, and Dr+1) are not BCD.  
The content of a\*DM word is not BCD when set for BCD.
- EQ(A50006) The division result is all zeroes.

**Example**

**/BL Operation**

When CIO 000001 is ON in the following example, the content of D00201 and D00200 is divided by the content of D00211 and D00210, in eight-digit BCD. When the result is obtained, the quotient is output to D00221 and D00220, and the remainder is output to D00223 and D00222.



Address	Instruction	Operands
00000	LD	000000
00001	/BL(435)	
		D00100
		D00110
		D00120

## 5-21 Floating-point Math Instructions

The Floating-point Math Instructions convert data and perform floating-point arithmetic operations.

Code	Mnemonic	Name
450	FIX (*)	FLOATING TO 16-BIT
451	FIXL (*)	FLOATING TO 32-BIT
452	FLT (*)	16-BIT TO FLOATING
453	FLTL (*)	32-BIT TO FLOATING
454	+F (*)	FLOATING-POINT ADD
455	-F (*)	FLOATING-POINT SUBTRACT
456	*F (*)	FLOATING-POINT MULTIPLY
457	/F (*)	FLOATING-POINT DIVIDE
458	RAD (*)	DEGREES TO RADIANS
459	DEG (*)	RADIANS-TO-DEGREES
460	SIN (*)	SINE
461	COS (*)	COSINE
462	TAN (*)	TANGENT
463	ASIN (*)	SINE TO ANGLE
464	ACOS (*)	COS TO ANGLE
465	ATAN (*)	TANGENT TO ANGLE
466	SQRT (*)	SQUARE ROOT
467	EXP (*)	EXPONENT
468	LOG (*)	LOGARITHM

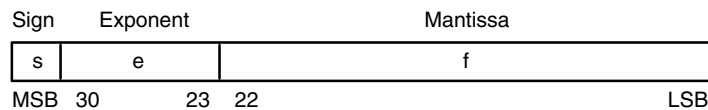
### Data Format

Floating-point data expresses real numbers using a sign, exponent, and mantissa. When data is expressed in floating-point format, the following formula applies.

$$\text{Real number} = (-1)^s 2^{e-127} (1.f)$$

s: Sign  
 e: Exponent  
 f: Mantissa

The floating-point data format conforms to the IEEE754 standards. Data is expressed in 32 bits, as follows:



Data	No. of bits	Contents
s:sign	1	0: positive; 1: negative
e:exponent	8	The exponent (e) value ranges from 0 to 255. The actual exponent is the value remaining after 127 is subtracted from e, resulting in a range of -127 to 128. "e=0" and "e=255" express special numbers.
f: mantissa	23	The mantissa portion of binary floating-point data fits the formal $2.0 > 1.f \geq 1.0$ .

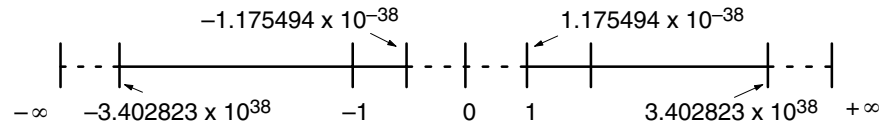
### Number of Digits

The number of effective digits for floating-point data is 24 bits for binary ( approximately seven digits decimal).

**Floating-point Data**

The following data can be expressed by floating-point data:

- $-\infty$
- $-3.402823 \times 10^{38} \leq \text{value} \leq -1.175494 \times 10^{-38}$
- 0
- $1.175494 \times 10^{-38} \leq \text{value} \leq 3.402823 \times 10^{38}$
- $+\infty$
- Not a number (NaN)



**Special Numbers**

The formats for NaN,  $\pm\infty$ , and 0 are as follows:

- NaN\*:  $e = 255, f \neq 0$
- $+\infty$ :  $e = 255, f = 0, s = 0$
- $-\infty$ :  $e = 255, f = 0, s = 1$
- 0:  $e = 0$

\*NaN is a valid floating-point number. Executing instructions involving data conversion or calculations may result in NaN.

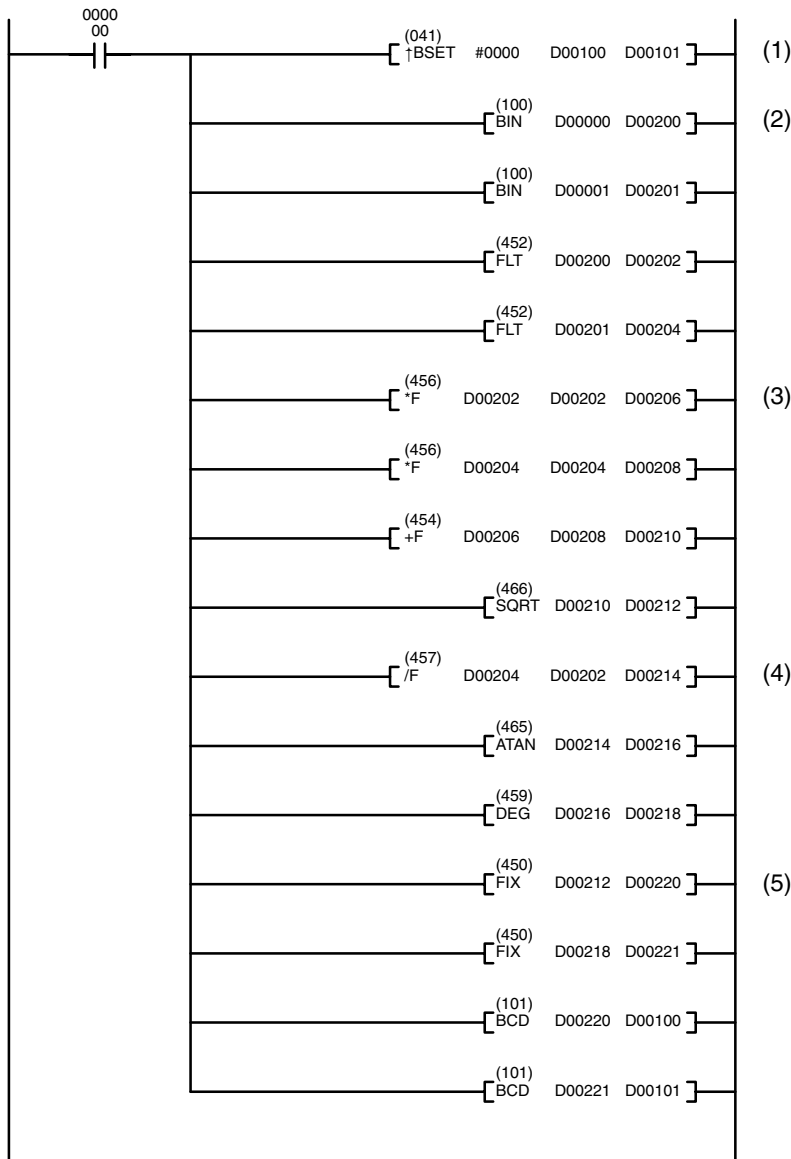
**Floating-point Data Calculation Results**

When the absolute value of the result is greater than the maximum value that can be expressed for floating-point data, the Overflow Flag (A50009) will turn ON and the result will be output as  $\pm\infty$ . If the result is positive, it will be output as  $+\infty$ ; if negative, then  $-\infty$ .

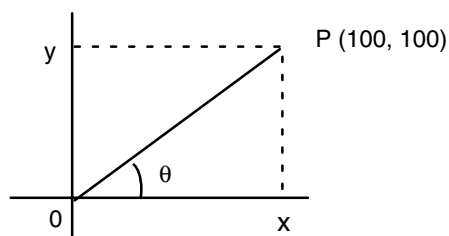
The Equals Flag (A50006) will only turn ON when both the exponent (e) and the mantissa (f) are zero after a calculation. A calculation result will also be output as zero when the absolute value of the result is less than the minimum value that can be expressed for floating-point data. In that case the Underflow Flag (A50010) will turn ON.

**Example**

In this program example, the X-axis and Y-axis coordinates (x, y) are provided by 4-digit BCD content of D00000 and D00001. The distance (r) from the origin and the angle ( $\theta$ , in degrees) are found and output to D00100 and D00101. In the result, everything to the right of the decimal point is truncated.



Address	Instruction	Operands
00000	LD	000000
00001	↑BSET(041)	
		#0000
		D00100
		D00101
00002	BIN(100)	
		D00000
		D00200
00003	BIN(100)	
		D00001
		D00201
00004	FLT(425)	
		D00200
		D00202
00005	FLT(425)	
		D00201
		D00204
00006	*F(456)	
		D00202
		D00202
		D00206
00007	*F(456)	
		D00204
		D00204
		D00208
00008	SQRT(466)	
		D00210
		D00212
00009	/F(457)	
		D00204
		D00202
		D00214
00010	ATAN(465)	
		D00212
		D00216
00011	DEG(459)	
		D00216
		D00218
00012	FIX(450)	
		D00212
		D00220
00013	FIX(450)	
		D00218
		D00221
00014	BCD(101)	
		D00220
		D00100
00015	BCD(101)	
		D00221
		D00101



**Calculations**

$$\text{Distance } r = \sqrt{x^2 + y^2}$$

$$\text{Angle } \theta = \tan^{-1} \left( \frac{y}{x} \right)$$

**Example**

$$\text{Distance } r = \sqrt{100^2 + 100^2} = 141.4214$$

$$\text{Angle } \theta = \tan^{-1} \left( \frac{100}{100} \right) = 45.0$$

**DM Contents**



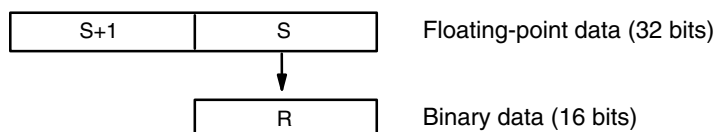
- 1, 2, 3...**
1. This instruction clears (i.e., sets to “0”) D00100 and D00101 so that the distance and angle result can be output to those words.
  2. This section of the program converts the data from BCD to floating-point.
    - a) The data area from D00200 onwards is used as a work area.
    - b) First BIN(100) is used to temporarily convert the BCD data to binary data, and then FLT(452) is used to convert the binary data to floating-point data.
    - c) The value of x that has been converted to floating-point data is output to D00203 and D00202.
    - d) The value of y that has been converted to floating-point data is output to D00205 and D00204.
  3. In order to find the distance r, Floating-point Math Instructions are used to calculate the square root of  $x^2+y^2$ . The result is then output to D00213 and D00212 as floating-point data.
  4. In order to find the angle θ, Floating-point Math Instructions are used to calculate  $\tan^{-1}(y/x)$ . ATAN(465) outputs the result in radians, so DEG(459) is used to convert to degrees. The result is then output to D00219 and D00218 as floating-point data.
  5. The data is converted back from floating-point to BCD.
    - a) First FIX(450) is used to temporarily convert the floating-point data to binary data, and then BCD(101) is used to convert the binary data to BCD data.
    - b) The distance r is output to D00100.
    - c) The angle θ is output to D00101.

### 5-21-1 FLOATING TO 16-BIT: FIX(450)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;"> <math>\text{---} \left[ \overset{(450)}{\text{FIX}} \quad \text{S} \quad \text{R} \right] \text{---}</math> </p> <p><b>Variations</b></p> <p>↑FIX(450)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: First source word</b>    CIO, G, A, T, C, #, DM</p> <p><b>R: Result word</b>        CIO, G, A, DM, DR, IR</p>
--	---

**Description**

When the execution condition OFF, FIX(450) is not executed. When the execution condition is ON, FIX(450) converts the 32-bit floating-point content of S and S+1 to 16-bit binary data, and places the result in R.



Only the integer portion of the floating-point data is converted, and the fraction portion is truncated. For example, “3.5” becomes “3,” and “-3.5” becomes “-3.”

**Precautions**

S must be floating-point data between -32,768 and 32,767.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

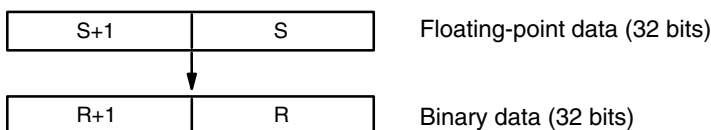
- ER (A50003): S is not floating-point data.  
The floating-point data is not between -32,768 to 32,767.  
The content of a\*DM word is not BCD when set for BCD.
- EQ (A50006): The converted binary data is all zeroes.
- N (A50008): The result of the conversion is a negative number.

### 5-21-2 FLOATING TO 32-BIT: FIXL(451)

<p><b>Ladder Symbol</b></p> <pre style="font-family: monospace;">                 _____(451)                 [  FIXL  S  R  ]-             </pre> <p><b>Variations</b></p> <p>↑FIXL(451)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: First source word</b> CIO, G, A, T, C, #, DM</p> <p><b>R: First result word</b> CIO, G, A, DM,</p>
--	--

**Description**

When the execution condition OFF, FIXL(451) is not executed. When the execution condition is ON, FIXL(451) converts the 32-bit floating-point content of S and S+1 to 32-bit binary data, and places the result in R and R+1.



Only the integer portion of the floating-point data is converted, and the fraction portion is truncated.

**Note** The maximum value for R other than indirect DM and indirect EM is -1. Constants are expressed in eight digits. Data register and index register (direct) cannot be used.

**Precautions**

S must be floating-point data between -2,147,483,648 and 2,147,483,647.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

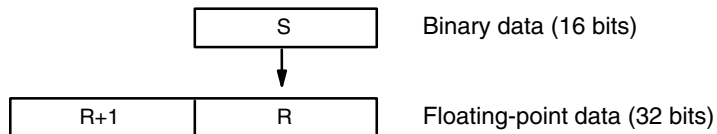
- ER (A50003): S is not floating-point data.  
Floating-point data is not between -2,147,483,648 to 2,147,483,647.  
The content of a\*DM word is not BCD when set for BCD.
- EQ (A50006): The converted binary data is all zeroes.
- N (A50008): The result of the conversion is a negative number.

### 5-21-3 16-BIT TO FLOATING: FLT(452)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(452)</p> <p style="text-align: center;">— [ FLT   S   R ] —</p> <p><b>Variations</b></p> <p>↑FLT(452)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: Source word</b>      CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>R: First result word</b>    CIO, G, A, DM</p>
---	---

**Description**

When the execution condition OFF, FLT(452) is not executed. When the execution condition is ON, FLT(452) converts the 16-bit binary content of S to 32-bit floating-point data, and places the result in R and R+1.



Only binary data within the range of -32,768 to 32,767 can be specified for S. To convert binary data outside of that range, use FLTL(453). Refer to 5-21-4 32-BIT TO FLOATING: FLTL(453).

**Precautions**

Refer to page 101 for general precautions on operand data areas.

**Flags**

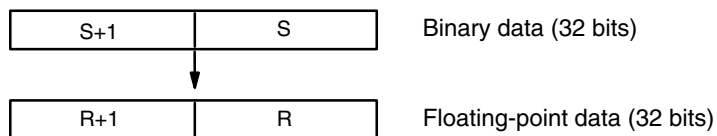
- ER (A50003): The content of a\*DM word is not BCD when set for BCD.
- EQ (A50006): The exponent and mantissa of the result are 0.
- N (A50008): The result of the conversion is a negative number.

### 5-21-4 32-BIT TO FLOATING: FLTL(453)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(453)</p> <p style="text-align: center;">— [ FLTL   S   R ] —</p> <p><b>Variations</b></p> <p>↑FLTL(453)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: First source word</b>    CIO, G, A, T, C, #, DM,</p> <p><b>R: First result word</b>    CIO, G, A, DM</p>
---	--

**Description**

When the execution condition OFF, FLTL(453) is not executed. When the execution condition is ON, FLTL(453) converts specified 32-bit binary data to 32-bit floating-point data, and places the result in specified words.



binary data within the range of -2,147,483,648 to 2,147,483,647 can be specified for S and S+1.

**Note** The maximum value for R other than indirect DM and indirect EM is -1. Constants are expressed in eight digits. Data register and index register (direct) cannot be used.

**Precautions**

Refer to page 101 for general precautions on operand data areas.



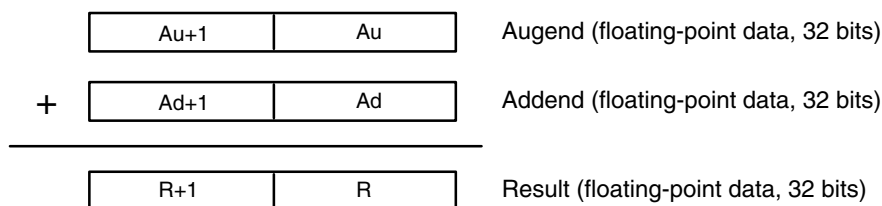
- Flags**
- ER (A50003): The content of a\*DM word is not BCD when set for BCD.
  - EQ (A50006): The exponent and mantissa of the result are 0.
  - N (A50008): The result of the conversion is a negative number.

### 5-21-5 FLOATING-POINT ADD: +F(454)

<p><b>Ladder Symbol</b></p> <pre style="font-family: monospace; font-size: 1.2em;">       _____(454)       [ +F      Au  Ad  R  ]           </pre> <p><b>Variations</b></p> <p>↑+F(454)</p>	<p><b>Operand Data Areas</b></p> <p><b>Au: First augend word</b> CIO, G, A, T, C, #, DM</p> <p><b>Ad: First addend word</b> CIO, G, A, T, C, #, DM</p> <p><b>R: First result word</b> CIO, G, A, DM</p>
---	---

**Description**

When the execution condition OFF, +F(454) is not executed. When the execution condition is ON, +F(454) adds 32-bit floating-point content of Ad and Ad+1 to the 32-bit floating-point content of Au and Au+1 and places the result in R and R+1.



If the absolute value of the result is greater than the maximum value that can be expressed for floating-point data, the Overflow Flag (A50009) will turn ON and the result will be output as  $\pm\infty$ .

If the absolute value of the result is less than the minimum value that can be expressed for floating-point data, the Underflow Flag (A50010) will turn ON and the result will be output as 0.

The various combinations of augend and addend data will produce the results shown in the following table.

Addend	Augend				NaN
	0	Numeral	$+\infty$	$-\infty$	
0	0	Numeral	$+\infty$	$-\infty$	
Numeral	Numeral	See note 1.	$+\infty$	$-\infty$	
$+\infty$	$+\infty$	$+\infty$	$+\infty$	ER	
$-\infty$	$-\infty$	$-\infty$	ER	$-\infty$	
NaN					

- Note**
1. The results could be zero (including underflows), a numeral,  $+\infty$ , or  $-\infty$ .
  2. ER: The Error Flag (A50003) turns ON and the instruction is not executed.

**Precautions**

Au, Au+1, Ad, and Ad+1 must be floating-point data.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

- ER (A50003): Au, Au+1, Ad, and Ad+1 are not floating-point data.  
The content of a\*DM word is not BCD when set for BCD.
- EQ (A50006): The exponent and mantissa of the result are 0.
- N (A50008): The result is a negative number.
- OF (A50009): The absolute value of the result is greater than the maximum value that can be expressed for floating-point data.

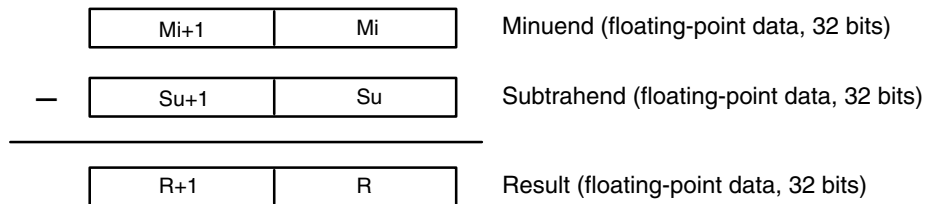
UF (A50010): Absolute value of the result is less than the minimum value that can be expressed for floating-point data.

### 5-21-6 FLOATING-POINT SUBTRACT: -F(455)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;"> <math>\text{---} \left[ \overset{(455)}{-F} \text{ Mi Su R } \right]</math> </p> <p><b>Variations</b></p> <p><math>\uparrow -F(455)</math></p>	<p><b>Operand Data Areas</b></p> <p><b>Mi: First minuend word</b> CIO, G, A, T, C, #, DM</p> <p><b>Su: First subtrahend word</b> CIO, G, A, T, C, #, DM</p> <p><b>R: First result word</b> CIO, G, A, DM</p>
--	--

**Description**

When the execution condition OFF, -F(455) is not executed. When the execution condition is ON, -F(455) subtracts the 32-bit floating-point content of Su and Su+1 from the 32-bit floating-point content of Mi and Mi+1 and places the result in R and R+1.



If the absolute value of the result is greater than the maximum value that can be expressed for floating-point data, the Overflow Flag (A50009) will turn ON and the result will be output as  $\pm \infty$ .

If the absolute value of the result is less than the minimum value that can be expressed for floating-point data, the Underflow Flag (A50010) will turn ON and the result will be output as 0.

The various combinations of minuend and subtrahend data will produce the results shown in the following table.

Subtrahend	Minuend				NaN
	0	Numeral	$+\infty$	$-\infty$	
0	0	Numeral	$+\infty$	$-\infty$	ER
Numeral	Numeral	See note 1.	$+\infty$	$-\infty$	
$+\infty$	$-\infty$	$-\infty$	ER	$-\infty$	
$-\infty$	$+\infty$	$+\infty$	$+\infty$	ER	
NaN					

- Note**
1. The results could be zero (including underflows), a numeral,  $+\infty$ , or  $-\infty$ .
  2. ER: The Error Flag (A50003) turns ON and the instruction is not executed.

**Precautions**

Mi, Mi+1, Su, and Su+1 must be floating-point data.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

- ER (A50003): Mi, Mi+1, Su, and Su+1 are not floating-point data.  
The content of a\*DM word is not BCD when set for BCD.
- EQ (A50006): The exponent and mantissa of the result are 0.
- N (A50008): The result is a negative number.
- OF (A50009): The absolute value of the result is greater than the maximum value that can be expressed for floating-point data.

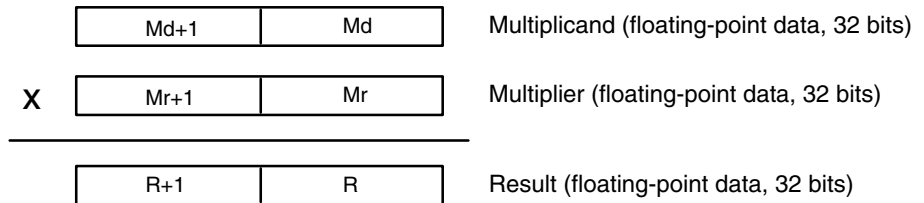
UF (A50010): Absolute value of the result is less than the minimum value that can be expressed for floating-point data.

### 5-21-7 FLOATING-POINT MULTIPLY: \*F(456)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;"> <math>\text{---} \left[ \overset{(456)}{*F} \text{ Md Mr R } \right]</math> </p> <p><b>Variations</b></p> <p>↑*F(456)</p>	<p><b>Operand Data Areas</b></p> <p><b>Md: First multiplicand word</b> CIO, G, A, T, C, #, DM</p> <p><b>Mr: First multiplier word</b> CIO, G, A, T, C, #, DM</p> <p><b>R: First result word</b> CIO, G, A, DM</p>
---	---

**Description**

When the execution condition OFF, \*F(456) is not executed. When the execution condition is ON, \*F(456) multiplies the 32-bit floating-point content of Md and Md + 1 by the 32-bit floating-point content of Mr and Mr + 1 and places the result in R and R+1.



If the absolute value of the result is greater than the maximum value that can be expressed for floating-point data, the Overflow Flag (A50009) will turn ON and the result will be output as  $\pm \infty$ .

If the absolute value of the result is less than the minimum value that can be expressed for floating-point data, the Underflow Flag (A50010) will turn ON and the result will be output as 0.

The various combinations of multiplicand and multiplier data will produce the results shown in the following table.

Multiplier	Multiplicand				NaN
	0	Numeral	$+\infty$	$-\infty$	
0	0	0	ER	ER	
Numeral	0	See note 1.	$+/-\infty$	$+/-\infty$	
$+\infty$	ER	$+/-\infty$	$+\infty$	$-\infty$	
$-\infty$	ER	$+/-\infty$	$-\infty$	$+\infty$	
NaN					

- Note**
1. The results could be zero (including underflows), a numeral,  $+\infty$ , or  $-\infty$ .
  2. ER: The Error Flag (A50003) turns ON and the instruction is not executed.

**Precautions**

Md, Md+1, Mr, and Mr+1 must be floating-point data.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

- ER (A50003): Md, Md+1, Mr, and Mr+1 are not floating-point data.  
The content of a\*DM word is not BCD when set for BCD.
- EQ (A50006): The exponent and mantissa of the result are 0.
- N (A50008): The result is a negative number.
- OF (A50009): The absolute value of the result is greater than the maximum value that can be expressed for floating-point data.

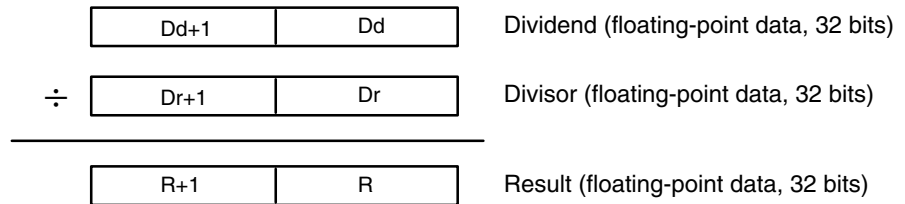
UF (A50010): Absolute value of the result is less than the minimum value that can be expressed for floating-point data.

### 5-21-8 FLOATING-POINT DIVIDE: /F(457)

<p><b>Ladder Symbol</b></p> <p><b>Variations</b></p> <p>↑/F(457)</p>	<p><b>Operand Data Areas</b></p> <p><b>Dd: First dividend word</b> CIO, G, A, T, C, #, DM</p> <p><b>Dr: First divisor word</b> CIO, G, A, T, C, #, DM</p> <p><b>R: First result word</b> CIO, G, A, DM</p>
--	--

**Description**

When the execution condition OFF, /F(457) is not executed. When the execution condition is ON, /F(457) divides the 32-floating-point content of Dd and Dd+1 by the 32-floating-point content of Dr and Dr+1 and places the result in R and R+1.



If the absolute value of the result is greater than the maximum value that can be expressed for floating-point data, the Overflow Flag (A50009) will turn ON and the result will be output as ±∞.

If the absolute value of the result is less than the minimum value that can be expressed for floating-point data, the Underflow Flag (A50010) will turn ON and the result will be output as 0.

The various combinations of dividend and divisor data will produce the results shown in the following table.

Divisor	Dividend				NaN
	0	Numeral	+∞	-∞	
0	ER	+/-∞	+∞	-∞	
Numeral	0	See note 2.	+/-∞	+/-∞	
+∞	0	0 (see note 1)	ER	ER	
-∞	0	0 (see note 1)	ER	ER	
NaN					ER

- Note**
1. The results will be zero for underflows.
  2. The results could be zero (including underflows), a numeral, +∞, or -∞.
  3. ER: The Error Flag (A50003) turns ON and the instruction is not executed.

**Precautions**

Dd, Dd+1, Dr, and Dr+1 must be floating-point data.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

- ER (A50003): Dd, Dd+1, Dr, and Dr+1 are not floating-point data.  
The content of a\*DM word is not BCD when set for BCD.
- EQ (A50006): The exponent and mantissa of the result are 0.
- N (A50008): The result is a negative number.
- OF (A50009): The absolute value of the result is greater than the maximum value that can be expressed for floating-point data.

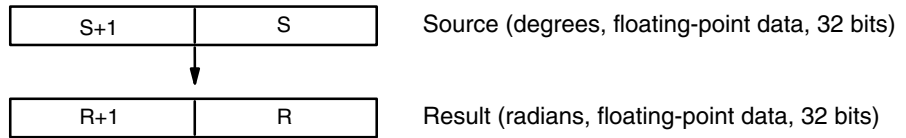
UF (A50010): Absolute value of the result is less than the minimum value that can be expressed for floating-point data.

### 5-21-9 DEGREES TO RADIANS: RAD(458)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(458)</p> <p style="text-align: center;">— [ RAD   S   R ]</p> <p><b>Variations</b></p> <p>↑RAD(458)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: First source word</b>    CIO, G, A, T, C, #, DM</p> <p><b>R: First result word</b>    CIO, G, A, DM</p>
---	---

**Description**

When the execution condition OFF, RAD(458) is not executed. When the execution condition is ON, RAD(458) converts the 32-floating-point content of S and S+1 from degrees to radians, and places the result in R and R+1.



Degrees are converted to radians by means of the following formula:

$$\text{Degrees} \times \pi/180 = \text{radians}$$

If the absolute value of the result is greater than the maximum value that can be expressed for floating-point data, the Overflow Flag (A50009) will turn ON and the result will be output as  $\pm \infty$ .

If the absolute value of the result is less than the minimum value that can be expressed for floating-point data, the Underflow Flag (A50010) will turn ON and the result will be output as 0.

**Precautions**

S and S+1 must be floating-point data.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

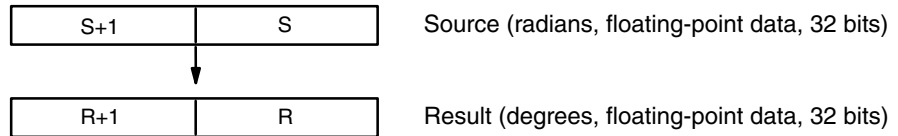
- ER (A50003): S and S+1 is not floating-point data.  
The content of a\*DM word is not BCD when set for BCD.
- EQ (A50006): The exponent and mantissa of the result are 0.
- N (A50008): The result is a negative number.
- OF (A50009): The absolute value of the result is greater than the maximum value that can be expressed for floating-point data.
- UF (A50010): Absolute value of the result is less than the minimum value that can be expressed for floating-point data.

### 5-21-10 RADIANS TO DEGREES: DEG(459)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(459)</p> <p>——— [ DEG S R ]</p> <p><b>Variations</b></p> <p>↑RAD(459)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: First source word</b> CIO, G, A, T, C, #, DM</p> <p><b>R: First result word</b> CIO, G, A, DM</p>
---	---

**Description**

When the execution condition OFF, DEG(459) is not executed. When the execution condition is ON, DEG(459) converts the 32-floating-point content of S and S+1 from degrees to radians, and places the result in R and R+1.



Degrees are converted to radians by means of the following formula:

$$\text{Radians} \times 180/\pi = \text{degrees}$$

If the absolute value of the result is greater than the maximum value that can be expressed for floating-point data, the Overflow Flag (A50009) will turn ON and the result will be output as  $\pm\infty$ .

If the absolute value of the result is less than the minimum value that can be expressed for floating-point data, the Underflow Flag (A50010) will turn ON and the result will be output as 0.

**Precautions**

S and S+1 must be floating-point data.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

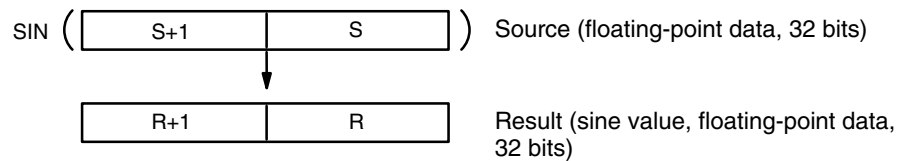
- ER (A50003): S and S+1 is not floating-point data.  
The content of a\*DM word is not BCD when set for BCD.
- EQ (A50006): The exponent and mantissa of the result are 0.
- N (A50008): The result is a negative number.
- OF (A50009): The absolute value of the result is greater than the maximum value that can be expressed for floating-point data.
- UF (A50010): Absolute value of the result is less than the minimum value that can be expressed for floating-point data.

### 5-21-11 SINE: SIN(460)

<p><b>Ladder Symbol</b></p> <pre>           (460)           ┌───┐           │ SIN  S  R  │           └───┘                 </pre> <p><b>Variations</b></p> <p>↑SIN(460)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: First source word</b>    CIO, G, A, T, C, #, DM</p> <p><b>R: First result word</b>    CIO, G, A, DM</p>
---	---

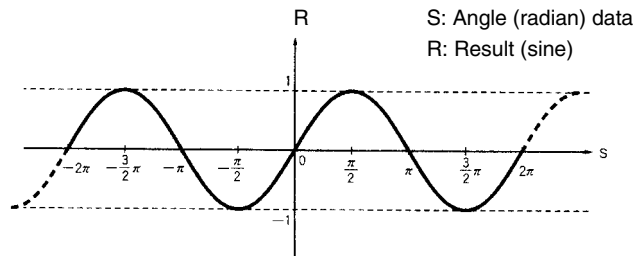
**Description**

When the execution condition OFF, SIN(460) is not executed. When the execution condition is ON, SIN(460) computes the sine of the angle (in radians) expressed as the 32-floating-point content of S and S+1, and places the result in R and R+1.



Specify the angle in radians for S and S+1. For information on converting from degrees to radian, refer to 5-21-9 DEGREES-TO-RADIANS: RAD(458).

**Relation Between Input Data and Result**



**Precautions**

S and S+1 must be floating-point data and its absolute value of must be less than 65,536.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

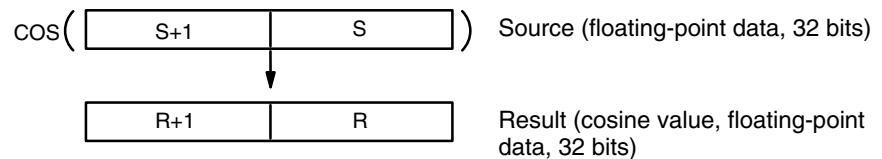
- ER (A50003): The absolute value of S and S+1 is 65,536 or greater.  
S and S+1 is not floating-point data.  
The content of a\*DM word is not BCD when set for BCD.
- EQ (A50006): The exponent and mantissa of the result are 0.
- N (A50008): The result is a negative number.
- OF (A50009): OFF when the computation is executed.
- UF (A50010): OFF when the computation is executed.

### 5-21-12 COSINE: COS(461)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;"> <math>\text{---} \left[ \overset{(461)}{\text{COS}} \quad \text{S} \quad \text{R} \right]</math> </p> <p><b>Variations</b></p> <p>↑COS(461)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: First source word</b>    CIO, G, A, T, C, #, DM</p> <p><b>R: First result word</b>    CIO, G, A, DM</p>
---	---

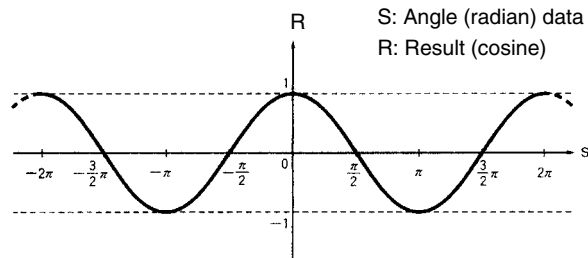
**Description**

When the execution condition OFF, COS(461) is not executed. When the execution condition is ON, COS(461) computes the cosine of the angle (in radians) expressed as the 32-floating-point content of S and S+1, and places the result in R and R+1.



Specify the angle in radians for S and S+1. For information on converting from degrees to radian, refer to 5-21-9 DEGREES-TO-RADIANS: RAD(458).

**Relation Between Input Data and Result**



**Precautions**

S and S+1 must be floating-point data and its absolute value must be less than 65,536.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

- ER (A50003): The absolute value of S and S+1 is 65,536 or greater.  
S and S+1 is not floating-point data.  
The content of a\*DM word is not BCD when set for BCD.
- EQ (A50006): The exponent and mantissa of the result are 0.
- N (A50008): The result is a negative number.
- OF (A50009): OFF when the computation is executed.
- UF (A50010): OFF when the computation is executed.

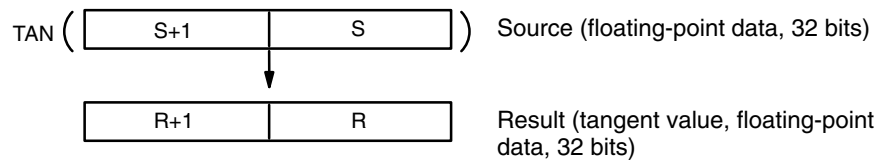


### 5-21-13 TANGENT: TAN(462)

<p><b>Ladder Symbol</b></p> <pre>           (462)           ┌───┴───┐           │ TAN   │ S   R   │           └───┬───┘           </pre> <p><b>Variations</b></p> <p>↑TAN(462)</p>	<p><b>Operand Data Areas</b></p> <p><b>S:</b> First source word    CIO, G, A, T, C, #, DM</p> <p><b>R:</b> First result word    CIO, G, A, DM</p>
--	---

**Description**

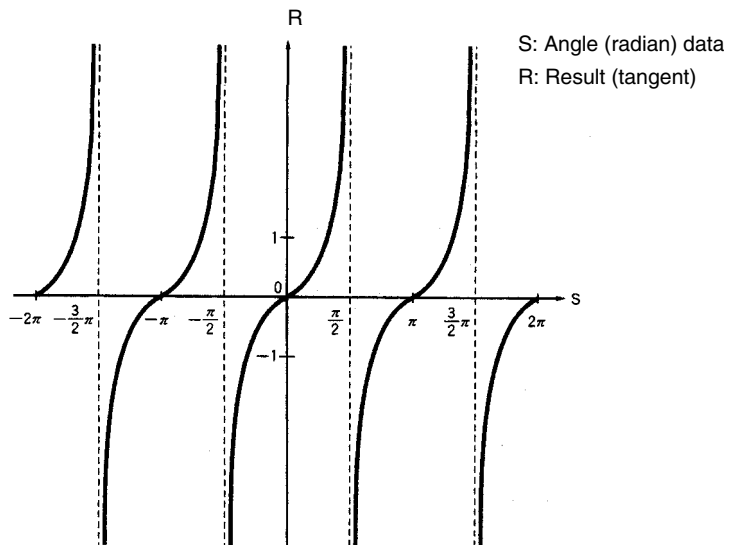
When the execution condition OFF, TAN(462) is not executed. When the execution condition is ON, TAN(462) computes the tangent of the angle (in radians) expressed as the 32-floating-point content of S and S+1, and places the result in R and R+1.



Specify the angle in radians for S and S+1. For information on converting from degrees to radian, refer to 5-21-9 DEGREES-TO-RADIANS: RAD(458).

If the absolute value of the result is greater than the maximum value that can be expressed for floating-point data, the Overflow Flag (A50009) will turn ON and the result will be output as ±∞.

**Relation Between Input Data and Result**



**Precautions**

S and S+1 must be floating-point data and its absolute value must be less than 65,536.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

- ER (A50003): The absolute value of S and S+1 is 65,536 or greater.  
S and S+1 is not floating-point data.  
The content of a\*DM word is not BCD when set for BCD.
- EQ (A50006): The exponent and mantissa of the result are 0.

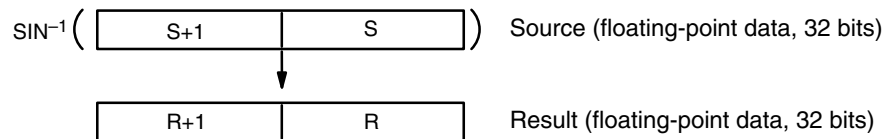
- N (A50008): The result is a negative number.
- OF (A50009): The absolute value of the result is greater than the maximum value that can be expressed for floating-point data.
- UF (A50010): OFF when the computation is executed.

### 5-21-14 SINE TO ANGLE: ASIN(463)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;"> <math>\text{---} \left[ \begin{matrix} (463) \\ \text{ASIN} \end{matrix} \text{ S R } \right]</math> </p> <p><b>Variations</b></p> <p>↑ASIN(463)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: First source word</b> CIO, G, A, T, C, #, DM</p> <p><b>R: First result word</b> CIO, G, A, DM</p>
--	---

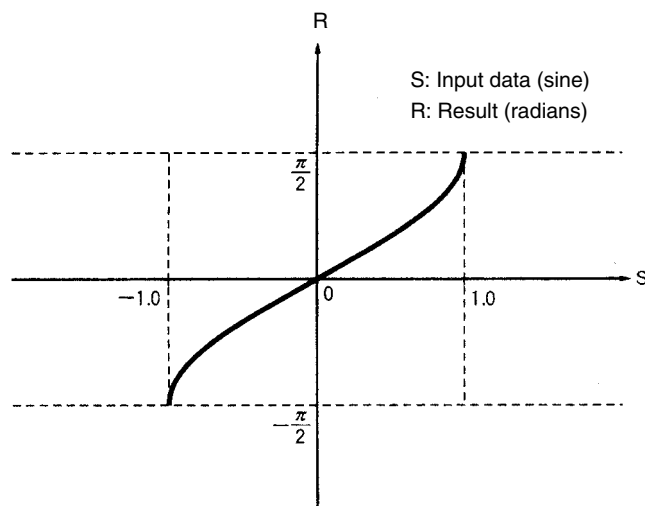
**Description**

When the execution condition OFF, ASIN(463) is not executed. When the execution condition is ON, ASIN(463) computes angle (in radians) for a sine expressed as the 32-floating-point content of S and S+1, and places the result in R and R+1.



The result is output to words R+1 and R as an angle (in radians) within the range of  $-\pi/2$  to  $\pi/2$ .

**Relation Between Input Data and Result**



**Precautions**

The sine must be floating-point data within the range of  $-1.0$  to  $1.0$ .

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

- ER (A50003): The sine data is not within the range of  $-1.0$  to  $1.0$ .  
The sine data is not floating-point data.  
The content of a\*DM word is not BCD when set for BCD.
- EQ (A50006): The exponent and mantissa of the result are 0.
- N (A50008): The result is a negative number.

OF (A50009): OFF when the computation is executed.

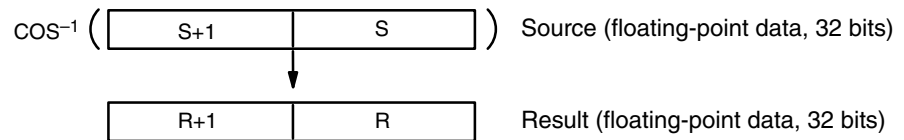
UF (A50010): OFF when the computation is executed.

### 5-21-15 COSINE TO ANGLE: ACOS(464)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(464)</p> <p style="text-align: center;">— [ ACOS S R ]</p> <p><b>Variations</b></p> <p>↑ACOS(464)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: First source word</b> CIO, G, A, T, C, #, DM</p> <p><b>R: First result word</b> CIO, G, A, DM</p>
---	---

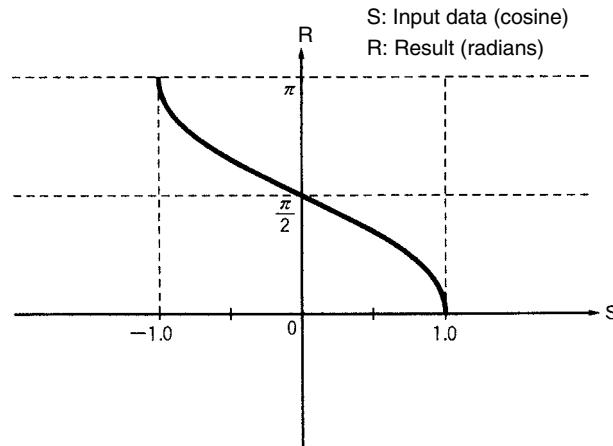
**Description**

When the execution condition OFF, ACOS(464) is not executed. When the execution condition is ON, ACOS(464) computes the angle (in radians) for a cosine expressed as the 32-floating-point content of S and S+1, and places the result in R and R+1.



The result is output to words R+1 and R as an angle (in radians) within the range of 0 to  $\pi$ .

**Relation Between Input Data and Result**



**Precautions**

The cosine must be floating-point data within the range of -1.0 to 1.0.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

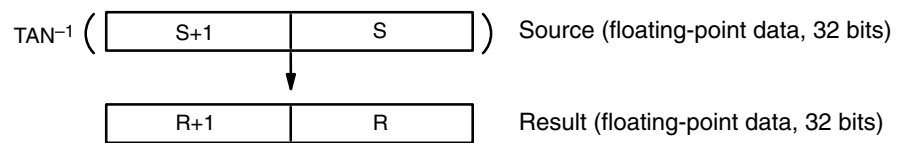
- ER (A50003): The cosine data is not within the range of -1.0 to 1.0.  
The cosine data is not floating-point data.  
The content of a\*DM word is not BCD when set for BCD.
- EQ (A50006): The exponent and mantissa of the result are 0.
- N (A50008): OFF when the computation is executed.
- OF (A50009): OFF when the computation is executed.
- UF (A50010): OFF when the computation is executed.

### 5-21-16 TANGENT TO ANGLE: ATAN(465)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;"> <math>\text{---} \left[ \begin{matrix} (465) \\ \text{ATAN} \end{matrix} \text{ S R } \right]</math> </p> <p><b>Variations</b></p> <p>↑ATAN(465)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: First source word</b> CIO, G, A, T, C, #, DM</p> <p><b>R: First result word</b> CIO, G, A, DM</p>
--	---

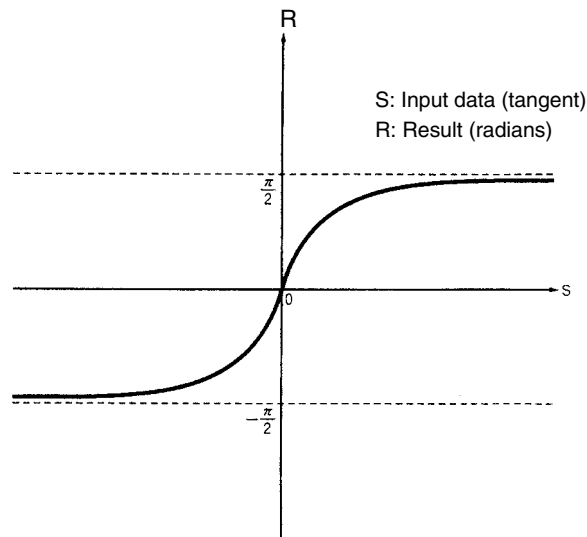
**Description**

When the execution condition OFF, ATAN(465) is not executed. When the execution condition is ON, ATAN(465) computes the angle (in radians) for a tangent expressed as the 32-floating-point content of S and S+1, and places the result in R and R+1.



The result is output to words R+1 and R as an angle (in radians) within the range of  $-\pi/2$  to  $\pi/2$ .

**Relation Between Input Data and Result**



**Precautions**

The tangent must be floating-point data within a range of  $-1.0$  to  $1.0$ .

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

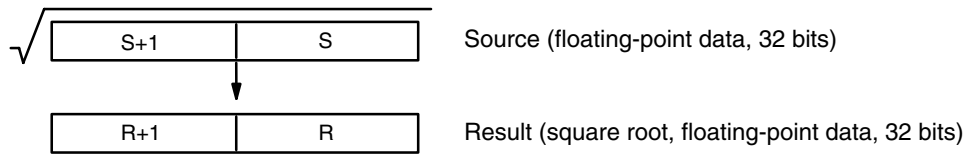
- ER (A50003): The tangent is not within the range of  $-1.0$  to  $1.0$ .  
The tangent is not floating-point data.  
The content of a\*DM word is not BCD when set for BCD.
- EQ (A50006): The exponent and mantissa of the result are 0.
- N (A50008): The result is a negative number.
- OF (A50009): OFF when the computation is executed.
- UF (A50010): OFF when the computation is executed.

### 5-21-17 SQUARE ROOT: SQRT(466)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(466)</p> <p style="text-align: center;">— [ SQRT S R ]</p> <p><b>Variations</b></p> <p>↑SQRT(466)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: First source word</b> CIO, G, A, T, C, #, DM</p> <p><b>R: First result word</b> CIO, G, A, DM</p>
---	---

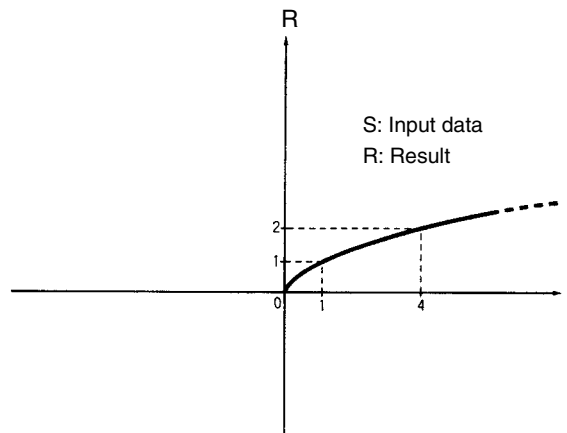
**Description**

When the execution condition OFF, SQRT(466) is not executed. When the execution condition is ON, SQRT(466) computes the square root of the 32-floating-point content of S and S+1, and places the result in R and R+1.



If the absolute value of the result is greater than the maximum value that can be expressed for floating-point data, the Overflow Flag (A50009) will turn ON and the result will be output as  $\pm \infty$ .

**Relation Between Input Data and Result**



**Precautions**

S and S+1 must be non-negative floating-point data.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

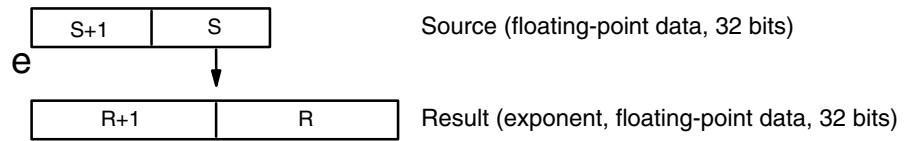
- ER (A50003): S and S+1 is a negative number.  
S and S+1 is not floating-point data.  
The content of a\*DM word is not BCD when set for BCD.
- EQ (A50006): The exponent and mantissa of the result are 0.
- N (A50008): OFF when the computation is executed.
- OF (A50009): The absolute value of the result is greater than the maximum value that can be expressed for floating-point data.
- UF (A50010): OFF when the computation is executed.

### 5-21-18 EXPONENT: EXP(467)

<p><b>Ladder Symbol</b></p> <p><b>Variations</b></p> <p>↑EXP(467)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: First source word</b> CIO, G, A, T, C, #, DM</p> <p><b>R: First result word</b> CIO, G, A, DM</p>
---	---

**Description**

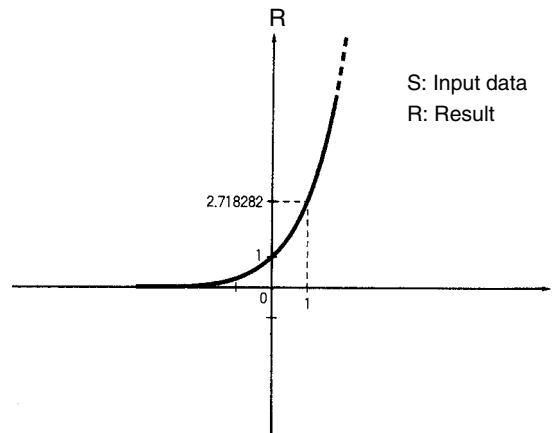
When the execution condition OFF, EXP(467) is not executed. When the execution condition is ON, EXP(467) computes the exponent for the 32-floating-point content of S and S+1, and places the result in R and R+1. The operation is executed with 2.718282 taken as the base (e).



If the absolute value of the result is greater than the maximum value that can be expressed for floating-point data, the Overflow Flag (A50009) will turn ON and the result will be output as  $\pm\infty$ .

If the absolute value of the result is less than the minimum value that can be expressed for floating-point data, the Underflow Flag (A50010) will turn ON and the result will be output as 0.

**Relation Between Input Data and Result**



**Precautions**

S and S+1 must be floating-point data.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

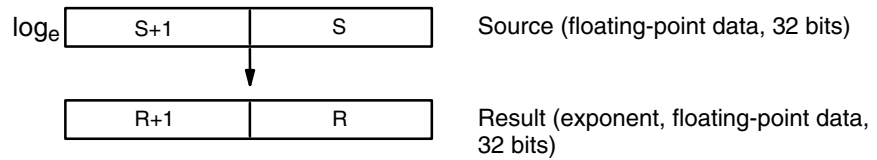
- ER (A50003): S and S+1 is not floating-point data.  
The content of a\*DM word is not BCD when set for BCD.
- EQ (A50006): The exponent and mantissa of the result are 0.
- N (A50008): OFF when the computation is executed.
- OF (A50009): The absolute value of the result is greater than the maximum value that can be expressed for floating-point data.
- UF (A50010): Absolute value of the result is less than the minimum value that can be expressed for floating-point data.

### 5-21-19 LOGARITHM: LOG(468)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(468)</p> <p style="text-align: center;">——— [ LOG   S   R ]</p> <p><b>Variations</b></p> <p>↑LOG(468)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: First source word</b>    CIO, G, A, T, C, #, DM</p> <p><b>R: First result word</b>    CIO, G, A, DM</p>
---	---

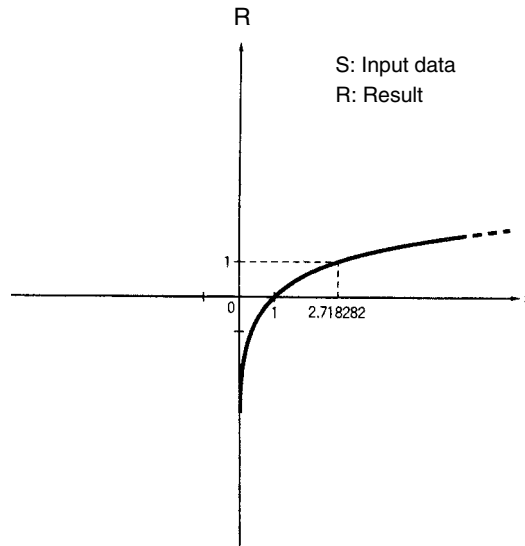
**Description**

When the execution condition OFF, LOG(468) is not executed. When the execution condition is ON, LOG(468) computes the natural logarithm for the 32-floating-point content of S and S+1, and places the result in R and R+1. The operation is executed with 2.718282 taken as the base (e).



If the absolute value of the result is greater than the maximum value that can be expressed for floating-point data, the Overflow Flag (A50009) will turn ON and the result will be output as  $\pm \infty$ .

**Relation Between Input Data and Result**



**Precautions**

S and S+1 must be non-negative floating-point data.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

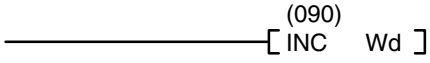
- ER (A50003): S and S+1 is a negative number.  
S and S+1 is not floating-point data.  
The content of a\*DM word is not BCD when set for BCD.
- EQ (A50006): The exponent and mantissa of the result are 0.
- N (A50008): The result is a negative number.
- OF (A50009): The absolute value of the result is greater than the maximum value that can be expressed for floating-point data.
- UF (A50010): OFF when the computation is executed.

## 5-22 Increment/Decrement Instructions

The Increment/Decrement Instructions all either increment or decrement a number by one.

The content of the source word is overwritten with the instruction result for all increment/decrement instructions.

### 5-22-1 INCREMENT BCD: INC(090)

<b>Ladder Symbol</b> 	<b>Operand Data Area</b> <b>Wd: Word</b> CIO, G, A, DM, DR, IR
<b>Variations</b> ↑ INC(090)	

**Description** When the execution condition is OFF, INC(090) is not executed. When the execution condition is ON, INC(090) increments Wd, without affecting carry (CY).

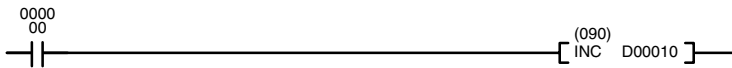
**Precautions** Wd must be BCD.

**Note** Refer to page 101 for general precautions on operand data areas.

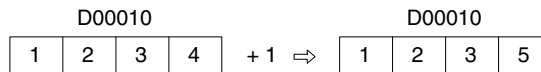
**Flags** ER (A50003): Wd is not BCD  
 Content of \*DM word is not BCD when set for BCD.

EQ (A50006): The result is 0.

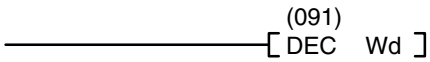
**Example** When CIO 000000 is ON in the following example, the content of D00010 is incremented by 1 as a BCD value.



Address	Instruction	Operands
00000	LD	000000
00001	INC(090)	
		D00010



### 5-22-2 DECREMENT BCD: DEC(091)

<b>Ladder Symbol</b> 	<b>Operand Data Area</b> <b>Wd: Word</b> CIO, G, A, DM, DR, IR
<b>Variations</b> ↑ DEC(091)	

**Description** When the execution condition is OFF, DEC(091) is not executed. When the execution condition is ON, DEC(091) decrements Wd, without affecting CY.

**Precautions** Wd must be BCD.

**Note** Refer to page 101 for general precautions on operand data areas.

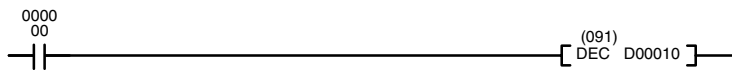
**Flags** ER (A50003): Wd is not BCD  
 Content of \*DM word is not BCD when set for BCD.

EQ (A50006): The result is 0.

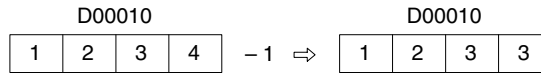


**Example**

When CIO 000000 is ON in the following example, the content of D00010 is decremented by 1 as a BCD value.



Address	Instruction	Operands
00000	LD	000000
00001	DEC(091)	
		D00010



**5-22-3 INCREMENT BINARY: INCB(092)**

Ladder Symbol	Operand Data Area
	<b>Wd: Word</b> CIO, G, A, DM, DR, IR
<b>Variations</b> ↑ INCB(092)	

**Description**

When the execution condition is OFF, INCB(092) is not executed. When the execution condition is ON, INCB(092) increments Wd, without affecting carry (CY). INCB(092) works the same way as INC(090) except that it increments a binary value instead of a BCD value.

**Precautions**

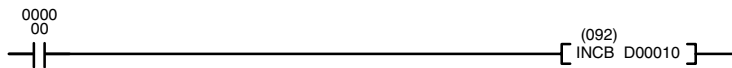
Refer to page 101 for general precautions on operand data areas.

**Flags**

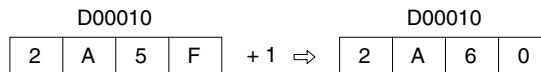
- ER (A50003): Content of \*DM word is not BCD when set for BCD.
- EQ (A50006): The result is 0.
- N (A50008): Shows the status of bit 15 of Wd after execution.

**Example**

When CIO 000000 is ON in the following example, the content of D00010 is incremented by 1 as a binary value.



Address	Instruction	Operands
00000	LD	000000
00001	INCB(092)	
		D00010



0 - - - N



**Example**

When CIO 000000 is ON in the following example, the content of D0100 and D01001 is incremented by 1 as a BCD value.

Address	Instruction	Operands
00000	LD	000000
00001	INCL(094)	
		D01000

Wd+1: D01001    Wd: D01000                  Wd+1: D01001    Wd: D01000

0 0 0 0   
 9 9 9 9
 + 1 ⇒   
 0 0 0 1   
 0 0 0 0

### 5-22-6 DOUBLE DECREMENT BCD: DECL(095)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;"> </p> <p><b>Variations</b></p> <p>↑ DECL(095)</p>	<p><b>Operand Data Area</b></p> <p><b>Wd: Word</b>                  CIO, G, A, DM</p>
--	---

**Description**

When the execution condition is OFF, DECL(095) is not executed. When the execution condition is ON, DECL(095) decrements the 8-digit BCD number contained in Wd+1 and Wd, without affecting carry (CY). Wd+1 contains the 10<sup>4</sup>, 10<sup>5</sup>, 10<sup>6</sup>, and 10<sup>7</sup> digits.

**Precautions**

Wd and Wd+1 must be BCD.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

ER (A50003): Wd or Wd+1 is not BCD  
 Content of \*DM word is not BCD when set for BCD.

EQ (A50006): The result is 0.

**Example**

When CIO 000000 is ON in the following example, the content of D0100 and D01001 is decremented by 1 as a BCD value.

Address	Instruction	Operands
00000	LD	000000
00001	DECL(095)	
		D01000

Wd+1: D01001    Wd: D01000                  Wd+1: D01001    Wd: D01000

0 0 0 1   
 0 0 0 0
 - 1 ⇒   
 0 0 0 0   
 9 9 9 9

### 5-22-7 DOUBLE INCREMENT BINARY: INBL(096)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;"> </p> <p><b>Variations</b></p> <p>↑ INBL(096)</p>	<p><b>Operand Data Area</b></p> <p><b>Wd: Word</b>                  CIO, G, A, DM</p>
--	---

**Description**

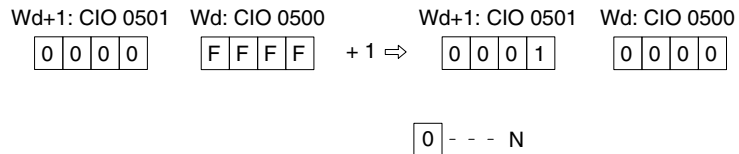
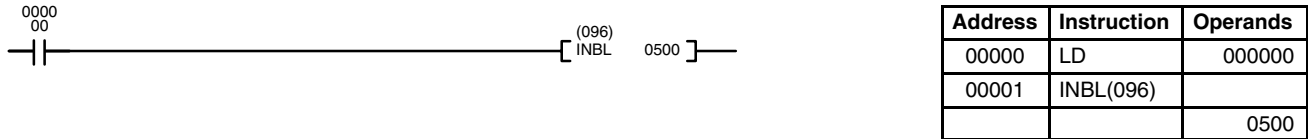
When the execution condition is OFF, INBL(096) is not executed. When the execution condition is ON, INBL(096) increments the 8-digit binary number contained in Wd+1 and Wd, without affecting carry (CY). Wd+1 contains the 16<sup>4</sup>, 16<sup>5</sup>, 16<sup>6</sup>, and 16<sup>7</sup> digits.

**Precautions**

Refer to page 101 for general precautions on operand data areas.

- Flags**
- ER (A50003): Content of \*DM word is not BCD when set for BCD.
  - EQ (A50006): The result is 0.
  - N (A50008): Shows the status of bit 15 of Wd+1 after execution.

**Example** When CIO 000000 is ON in the following example, the content of CIO 0500 and CIO 0501 is incremented by 1 as a binary value.



### 5-22-8 DOUBLE DECREMENT BINARY: DCBL(097)

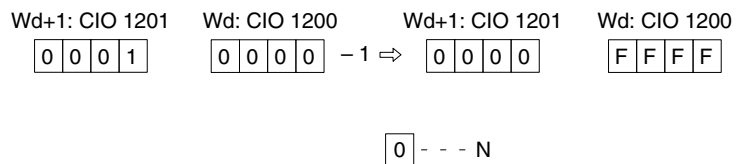
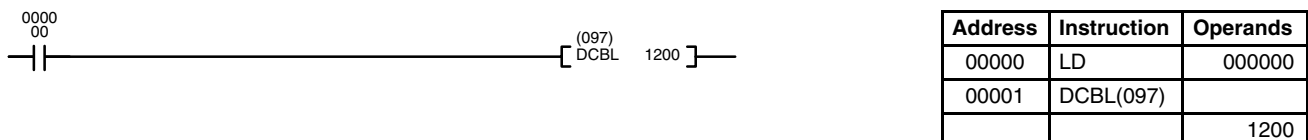
<p><b>Ladder Symbol</b></p> <p><b>Variations</b></p> <p>↑ DCBL(097)</p>	<p><b>Operand Data Area</b></p> <p><b>Wd: Word</b>            CIO, G, A, DM</p>
---	---

**Description** When the execution condition is OFF, DCBL(097) is not executed. When the execution condition is ON, DCBL(097) decrements the 8-digit binary number contained in Wd+1 and Wd, without affecting carry (CY). Wd+1 contains the 16<sup>4</sup>, 16<sup>5</sup>, 16<sup>6</sup>, and 16<sup>7</sup> digits.

**Precautions** Refer to page 101 for general precautions on operand data areas.

- Flags**
- ER (A50003): Content of \*DM word is not BCD when set for BCD.
  - EQ (A50006): The result is 0.
  - N (A50008): Shows the status of bit 15 of Wd+1 after execution.

**Example** When CIO 000000 is ON in the following example, the content of CIO 1200 and CIO 1201 is decremented by 1 as a binary value.



## 5-23 Special Math Instructions

The Special Math Instructions perform special arithmetic operations. MAX(165) searches a range of words for the maximum value. MIN(166) searches a range of words for the minimum value. SUM(167) adds a range of words. ROOT(140) finds the square root of a value. FDIV(141) performs floating-point division. APR(142) finds the sine or cosine of an angle or extrapolates the Y value for a given X value based on a table of coordinates.

### 5-23-1 FIND MAXIMUM: MAX(165)

Ladder Symbol	Operand Data Areas
	<b>C: Control word</b> CIO, G, A, #, DM, DR, IR <b>R<sub>1</sub>: 1<sup>st</sup> word in range</b> CIO, G, A, T, C, DM, <b>D: Destination word</b> CIO, G, A, DM, DR, IR
<b>Variations</b> ↑ MAX(165)	

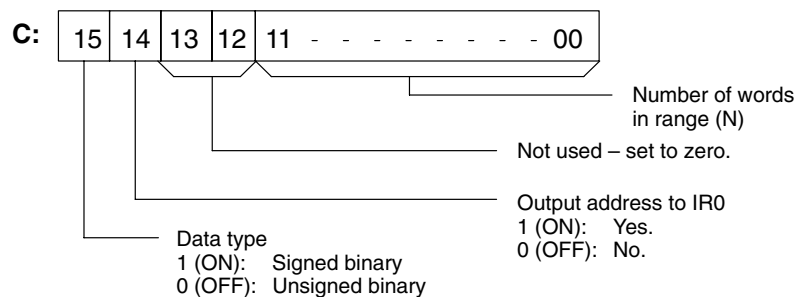
#### Description

When the execution condition is OFF, MAX(165) is not executed. When the execution condition is ON, MAX(165) searches the range of memory from R<sub>1</sub> to R<sub>1</sub>+N-1 for the address that contains the maximum value, outputs the maximum value to the destination word (D) and, if bit 14 of C is ON, outputs the memory address of the word containing the maximum value to IR0.

If bit 14 of C is ON and more than one address contains the same maximum value, the lowest of the addresses will be output to IR0.

The number of words within the range (N) is contained in the 3 rightmost digits of C, which must be BCD between 001 and 999.

When bit 15 of C is OFF, data within the range is treated as unsigned binary and when it is ON the data is treated as signed binary. Refer to 3-2 Data Area Structure for information on signed and unsigned binary data.



#### Precautions

The 3 rightmost digits of C must be BCD between 001 and 999.

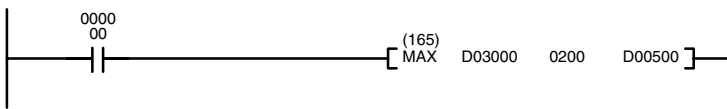
**Note** Refer to page 101 for general precautions on operand data areas.

#### Flags

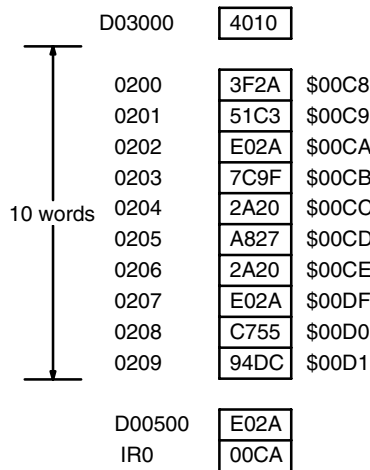
- ER (A50003): The 3 rightmost digits of C are not BCD between 001 and 999. Content of \*DM word is not BCD when set for BCD.
- EQ (A50006): The maximum value is zero.
- N (A50008): Shows the status of bit 15 of the maximum value.

**Example**

When CIO 000000 is ON in the following example, MAX(165) outputs to D00500 the maximum value within the 10-word range from CIO 0200 to CIO 0209. Because bit 14 of C is ON, the lower address of the two addresses within the range that contain the maximum value is output to IR0.



Address	Instruction	Operands
00000	LD	000000
00001	MAX(165)	
		D03000
		0200
		D00500



### 5-23-2 FIND MINIMUM: MIN(166)

Ladder Symbol	Operand Data Areas
	<b>C: Control word</b> CIO, G, A, #, DM, DR, IR <b>R<sub>1</sub>: 1<sup>st</sup> word in range</b> CIO, G, A, T, C, DM <b>D: Destination word</b> CIO, G, A, DM, DR, IR
<b>Variations</b> ↑ MIN(166)a	

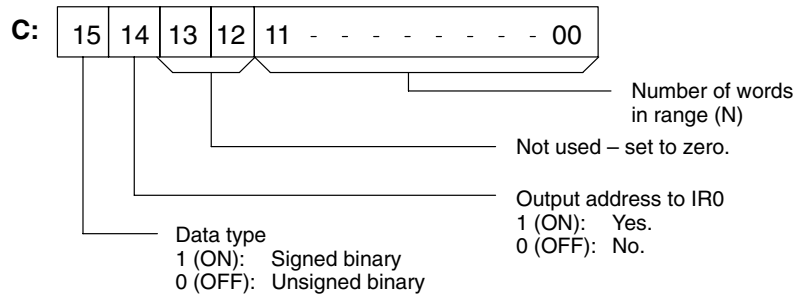
**Description**

When the execution condition is OFF, MIN(166) is not executed. When the execution condition is ON, MIN(166) searches the range of memory from R<sub>1</sub> to R<sub>1</sub>+N-1 for the address that contains the minimum value, outputs that value to the destination word (D) and, if bit 14 of C is ON, outputs the memory address of the word containing the minimum value to IR0.

If bit 14 of C is ON and more than one address contains the same minimum value, the lowest of the addresses will be output to IR0.

The number of words within the range (N) is contained in the 3 rightmost digits of C, which must be BCD between 001 and 999.

When bit 15 of C is OFF, data within the range is treated as unsigned binary and when it is ON the data is treated as signed binary. Refer to 3-2 Data Area Structure for information on signed and unsigned binary data.



**Precautions**

The 3 rightmost digits of C must be BCD between 001 and 999.

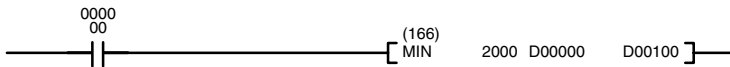
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

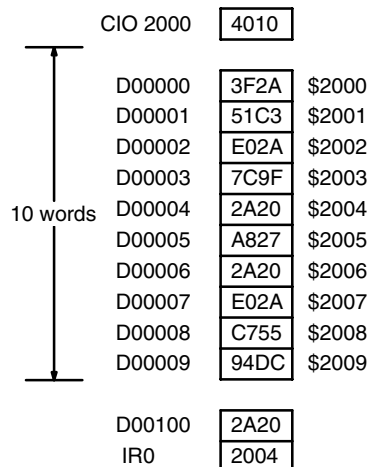
- ER (A50003): The 3 rightmost digits of C are not BCD between 001 and 999.  
Content of \*DM word is not BCD when set for BCD.
- EQ (A50006): The minimum value is zero.
- N (A50008): Shows the status of bit 15 of the minimum value.

**Example**

When CIO 00000 is ON in the following example, MIN(166) outputs to D00100 the minimum value within the 10-word range from D00000 to D00009. Because bit 14 of C is ON, the lower address of the two addresses within the range that contain the minimum value is output to IR0.



Address	Instruction	Operands
00000	LD	000000
00001	MIN(166)	
		2000
		D00000
		D00200



### 5-23-3 SUM: SUM(167)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(167)</p> <p>— [ SUM C R<sub>1</sub> D ]</p> <p><b>Variations</b></p> <p>↑ SUM(167)</p>	<p><b>Operand Data Areas</b></p> <p><b>C: Control word</b> CIO, G, A, #, DM, DR, IR</p> <p><b>R<sub>1</sub>: 1<sup>st</sup> word in range</b> CIO, G, A, T, C, DM</p> <p><b>D: 1<sup>st</sup> destination word</b> CIO, G, A, DM</p>
--	--

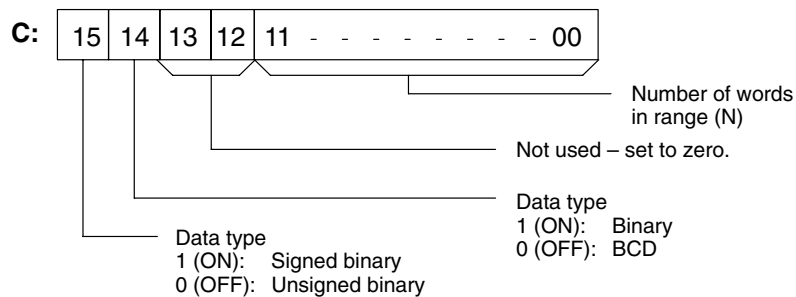
**Description**

When the execution condition is OFF, SUM(167) is not executed. When the execution condition is ON, SUM(167) computes the sum of the contents of words from R<sub>1</sub> to R<sub>1</sub>+N-1 and outputs that value to the destination words (D and D+1).

The number of words within the range (N) is contained in the 3 rightmost digits of C, which must be BCD between 001 and 999.

When bit 15 of C is OFF, data within the range is treated as unsigned binary and when it is ON the data is treated as signed binary. Refer to 3-2 Data Area Structure for information on signed and unsigned binary data.

When bit 14 of C is OFF, data within the range is treated as BCD and when it is ON the data is treated as binary. The data will be treated as BCD when bit 14 is OFF, even if bit 15 is ON, indicating signed binary data.



**Precautions**

The 3 rightmost digits of C must be BCD between 001 and 999.

R<sub>1</sub> and R<sub>1</sub>+N-1 must be in the same data area.

If bit 14 of C is OFF (setting for BCD data), all data within the range R<sub>1</sub> to R<sub>1</sub>+N-1 must be BCD.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

ER (A50003): The 3 rightmost digits of C are not BCD between 001 and 999.  
 Bit 14 of C is OFF, indicating BCD data, but the content of a word within the range is not BCD.  
 Content of \*DM word is not BCD when set for BCD.

EQ (A50006): The result is zero.

N (A50008): Shows the status of bit 15 of D+1.

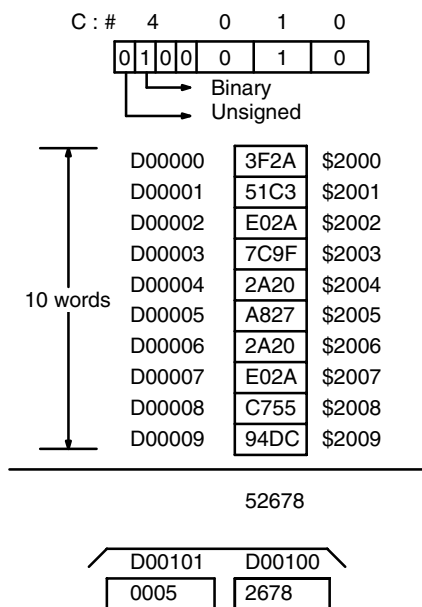
**Example**

When CIO 000000 is ON in the following example, SUM(167) computes the sum of the contents of the words within the 10-word range from D00000 to D00009 and outputs that value to D00100 and D00101. The data is treated as unsigned binary data because the leftmost digit of the control word is 4.



Address	Instruction	Operands
00000	LD	000000
00001	SUM(167)	
		#4010
		D00000
		D00100





### 5-23-4 BCD SQUARE ROOT: ROOT(140)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(140)</p> <p>_____ [ ROOT Sq    R ]</p> <p><b>Variations</b></p> <p>↑ ROOT(140)</p>	<p><b>Operand Data Areas</b></p> <p><b>Sq:</b> 1<sup>st</sup> source word CIO, G, A, T, C, #, DM</p> <p><b>R:</b> Result word    CIO, G, A, DM, DR, IR</p>
--	--

**Description**

When the execution condition is OFF, ROOT(140) is not executed. When the execution condition is ON, ROOT(140) computes the square root of the 8-digit content of Sq and Sq+1 and places the result in R. The fractional portion is truncated.



**Precautions**

Sq must be BCD.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

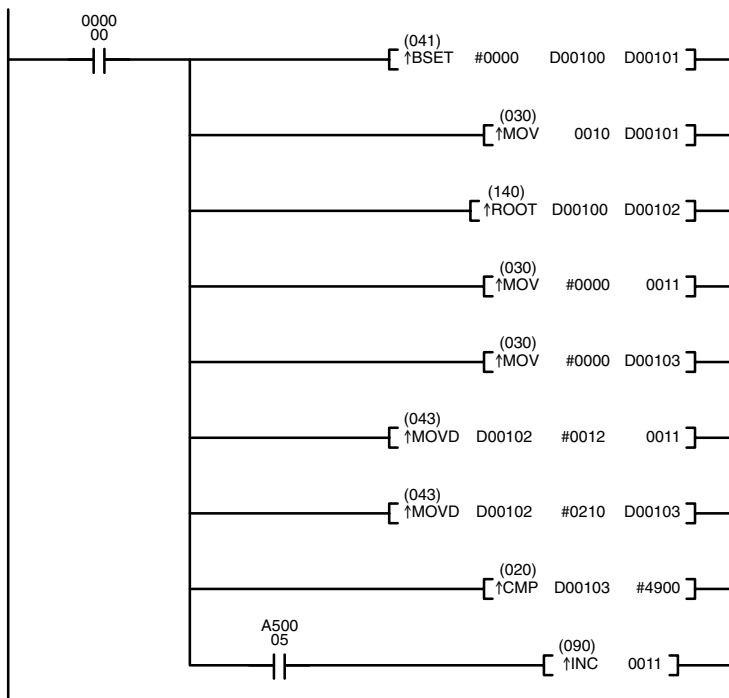
ER (A50003): Sq or the content of \*DM word is not BCD when set for BCD.  
 EQ (A50006): ON when the result is 0.

**Example**

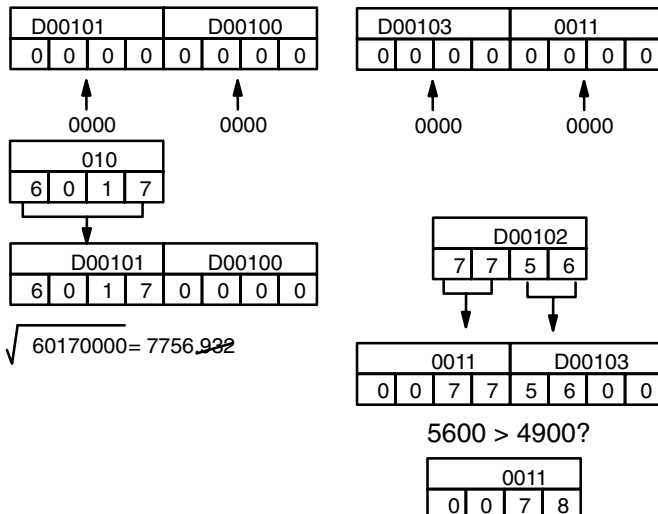
The following example shows how to take the square root of a 4-digit number and then round the result.

When CIO 000000 is ON, first the words to be used are cleared to all zeros and then the value whose square root is to be taken is moved to Sq+1. The result, which has twice the number of digits as the correct answer (because ROOT(140) operates on an 8-digit number and here we are using a 4-digit number), is placed in D00102, and the digits are split into two different words, the leftmost two digits to CIO 0011 for the answer and the rightmost two digits to D00103 so that the answer in CIO 0011 can be rounded. The last step is to compare the value in D00103 so that CIO 0011 can be incremented using the Greater Than Flag (A50005) when the value must be rounded up.

In this example,  $\sqrt{6017} = 77.56$ . The result is rounded off to an integer, according to the digit in the tenths place. Thus, 77.56 is rounded off to 78.



Address	Instruction	Operands
00000	LD	0000000
00001	↑BSET(041)	
		#0000
		D00100
		D00101
00002	↑MOV(030)	
		0010
		D00101
00003	↑ROOT(140)	
		D00100
		D00102
00004	↑MOV(030)	
		#0000
		0011
00005	↑MOV(030)	
		#0000
		D00103
00006	↑MOVD(043)	
		D00102
		#0012
		0011
00007	↑MOVD(043)	
		D00102
		#0210
		D00103
00008	↑CMP(020)	
		D00103
		#4900
00009	LD	A50005
00010	↑INC(090)	
		0011



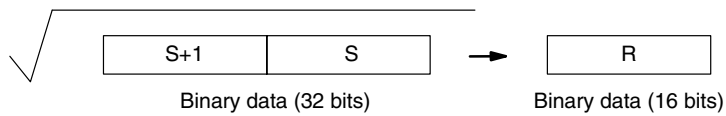
$\sqrt{60170000} = 7756.992$

### 5-23-5 BINARY ROOT: ROTB(274)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(274)</p> <p style="text-align: center;">— [ ROTB S R ] —</p> <p><b>Variations</b></p> <p>↑ROTB(274)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: First source word</b> CIO, G, A, T, C, #, DM</p> <p><b>R: Result word</b> CIO, G, A, DM, DR, IR</p>
---	---

**Description**

When the execution condition is OFF, ROTB(274) is not executed. When the execution condition is ON, ROTB(274) computes the square root of the 32-bit binary content of the specified word (S) and outputs the integer portion of the result to the specified result word (R). The fraction portion is eliminated.



The range of data that can be specified for words S+1 and S is 0000 0000 to 3FFF FFFF. If a number from 4000 0000 to 7FFF FFFF is specified, it will be treated as 3FFF FFFF for the square root computation.

**Precautions**

S, S+1 must be non-negative between 0000 0000 and 3FFF FFFF.

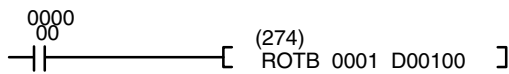
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

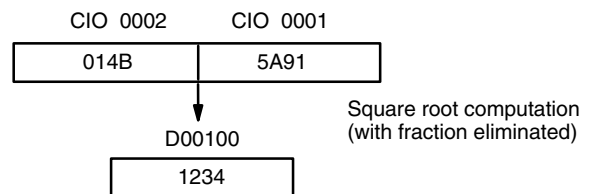
- ER (A50003): S, S+1 is negative (leftmost bit of S+1 is "1").  
The content of a\*DM word is not BCD when set for BCD.
- = (A50006) The output data is all zeroes.
- N (A50008) OFF when ROTB(274) is executed.
- OF (A50009) The input data (S+1, S) is within the range of 4000 0000 to 7FFF FFFF.
- UF (A50010) OFF when ROTB(274) is executed.

**Example**

When CIO 000000 is ON in the following example, the square root of the data in CIO 0002 and CIO 0001 is computed, and the result (integer only) is placed in D00100.



Address	Instruction	Operands
00000	LD	000000
00001	ROTB(274)	
		0001
		D00100

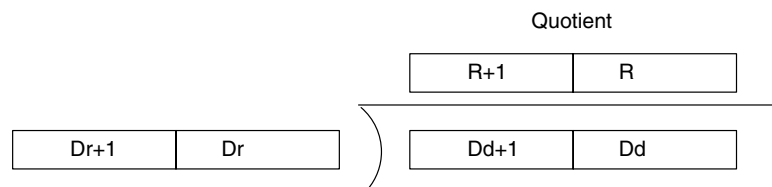


### 5-23-6 FLOATING POINT DIVIDE: FDIV(141)

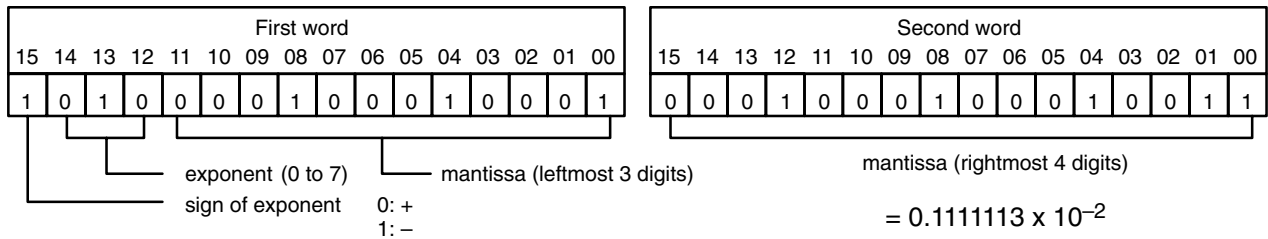
<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(141)</p> <p>— [ FDIV Dd Dr R ]</p> <p><b>Variations</b></p> <p>↑ FDIV(141)</p>	<p><b>Operand Data Areas</b></p> <p><b>Dd: 1<sup>st</sup> dividend word</b> CIO, G, A, T, C, DM</p> <p><b>Dr: 1<sup>st</sup> divisor word</b> CIO, G, A, T, C, DM</p> <p><b>R: 1<sup>st</sup> result word</b> CIO, G, A, DM</p>
--	---

**Description**

When the execution condition is OFF, FDIV(141) is not executed. When the execution condition is ON, FDIV(141) divides the floating-point value in Dd and Dd+1 by that in Dr and Dr+1 and places the result in R and R+1.



To represent the floating point values, the rightmost seven digits are used for the mantissa and the leftmost digit is used for the exponent, as shown in the diagram below. The mantissa is expressed as a value less than one, i.e., to seven decimal places.



**Precautions**

Dd, Dd+1, Dr and Dr+1 must be BCD. Dr and Dr+1 cannot contain zero.

The dividend and divisor must be between  $0.0000001 \times 10^{-7}$  and  $0.9999999 \times 10^7$ . The results must be between  $0.1 \times 10^{-7}$  and  $0.9999999 \times 10^7$ .

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

ER (A50003): Dr and Dr+1 contain 0.

Dd, Dd+1, Dr, or Dr+1 is not BCD.

The result is not between  $0.1 \times 10^{-7}$  and  $0.999999 \times 10^7$ .

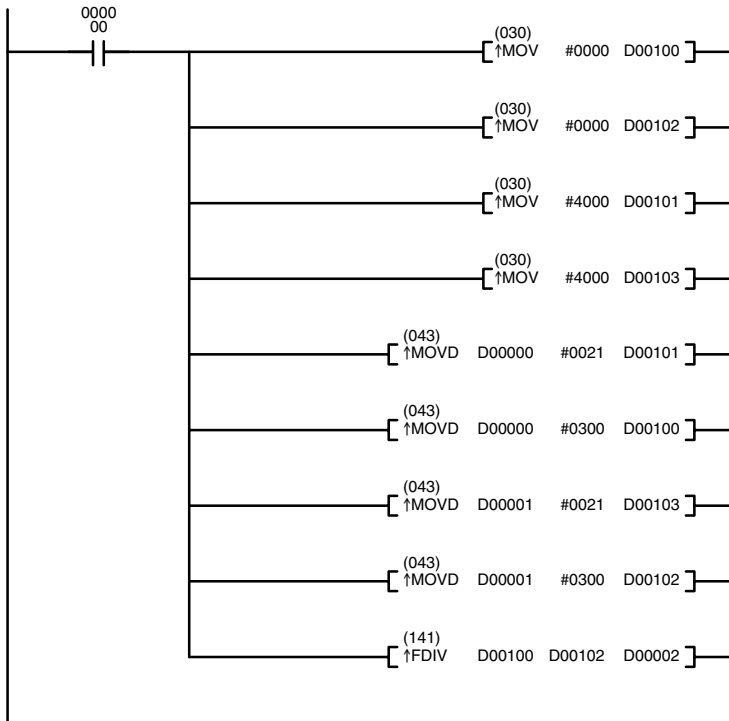
Content of \*DM word is not BCD when set for BCD.

EQ (A50006): ON when the result is 0.

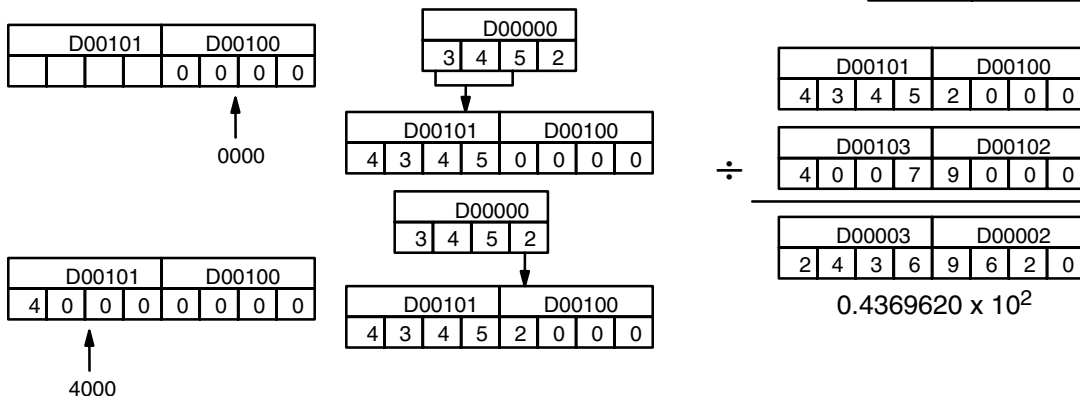
**Example**

The following example shows how to divide two 4-digit whole numbers (i.e., numbers without fractions) so that a floating-point value can be obtained.

First the original numbers must be placed in floating-point form. Because the numbers are originally without decimal points, the exponent will be 4 (e.g., 3452 would equal  $0.3452 \times 10^4$ ). All of the moves are to place the proper data into consecutive words for the final division, including the exponent and zeros. Data movements for Dd and Dd+1 are shown at the right below. Movements for Dr and Dr+1 are basically the same. The original values to be divided are in D00000 and D00001. The final division is also shown.



Address	Instruction	Operands
00000	LD	000000
00001	↑MOV(030)	
		#0000
		D00100
00002	↑MOV(030)	
		#0000
		D00102
00003	↑MOV(030)	
		#4000
		D00101
00004	↑MOV(030)	
		#4000
		D00103
00005	↑MOVD(043)	
		D00000
		#0021
		D00101
00006	↑MOVD(043)	
		D00000
		#0300
		D00100
00007	↑MOVD(043)	
		D00001
		#0021
		D00103
00008	↑MOVD(043)	
		D00001
		#0300
		D00102
00009	↑FDIV(141)	
		D00100
		D00102
		D00002



### 5-23-7 ARITHMETIC PROCESS: APR(142)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(142)</p> <p>— [ APR C S R ]</p> <p><b>Variations</b></p> <p>↑ APR(142)</p>	<p><b>Operand Data Areas</b></p> <p><b>C: Control word</b> CIO, G, A, #, DM, DR, IR</p> <p><b>S: Source data</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>R: Result word</b> CIO, G, A, DM, DR, IR</p>
--	--

**Description**

When the execution condition is OFF, APR(142) is not executed. When the execution condition is ON, the operation of APR(142) depends on the control word C. If C is 0000 or 0001, APR(142) computes the sine or cosine of S. S in units of tenths of degrees.

If C is a word address, APR(142) extrapolates the Y value for the X value in S based on coordinates (forming line segments) entered in advance in a table beginning at C.

**Precautions**

For trigonometric functions, S must be BCD between 0000 and 0900 (between 0° and 90°). For linear extrapolation, S must be BCD when set for BCD. C must be #0000, #0001, or a word address.

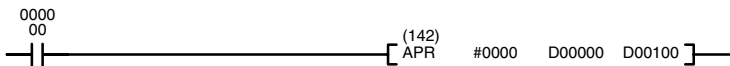
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

- ER (A50003): For trigonometric functions, S is greater than 0900 or not BCD. For linear extrapolation, S is not BCD when set for BCD or the table is not readable. Content of \*DM word is not BCD when set for BCD.
- EQ (A50006): ON when the result is 0.
- N (A50008): Shows the status of bit 15 of the results.

**Sine Function**

The following example shows APR(142) used to calculate the sine of 30°. The sine function is specified because C is #0000.



Address	Instruction	Operands
00000	LD	000000
00001	APR(142)	
		#0000
		D00000
		D00100

**Source data**

S: D00000			
0	10 <sup>1</sup>	10 <sup>0</sup>	10 <sup>-1</sup>
0	3	0	0

Enter input data not exceeding #0900 in BCD form.

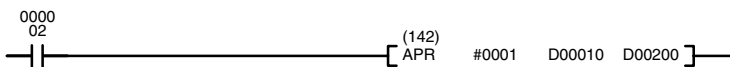
**Result**

R: D00100			
10 <sup>-1</sup>	10 <sup>-2</sup>	10 <sup>-3</sup>	10 <sup>-4</sup>
5	0	0	0

Result data has four significant digits, fifth and higher digits are ignored. The result for sin(90) will be 0.9999, not 1.

**Cosine Function**

The following example shows APR(142) used to calculate the cosine of 30°. The cosine function is specified because C is #0001.



Address	Instruction	Operands
00000	LD	000000
00001	APR(142)	
		#0000
		D00010
		D00200

Source data

S: D00010			
0	10 <sup>1</sup>	10 <sup>0</sup>	10 <sup>-1</sup>
0	3	0	0

Enter input data not exceeding #0900  
in BCD form.

Result

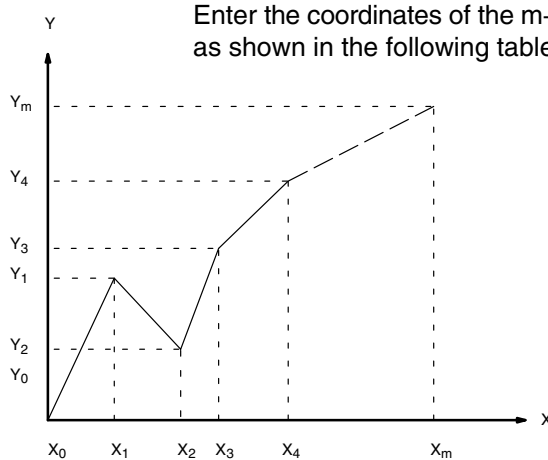
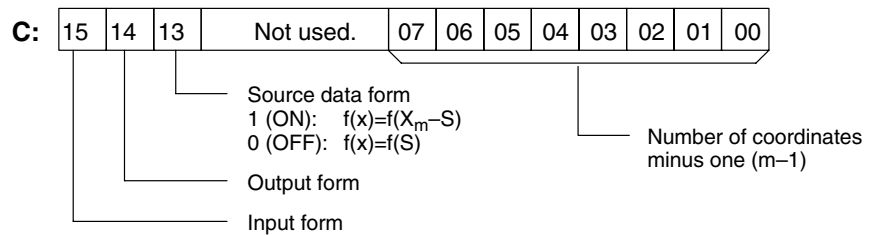
R: D00200			
10 <sup>-1</sup>	10 <sup>-2</sup>	10 <sup>-3</sup>	10 <sup>-4</sup>
8	6	6	0

Result data has four significant  
digits, fifth  
and higher digits are ignored.  
The result for cos(0) will be  
0.9999, not 1.

**Linear Extrapolation**

APR(142) linear extrapolation is specified when C is a word address.

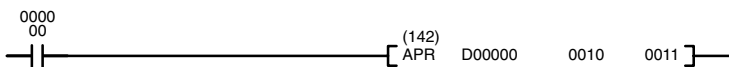
The content of word C specifies the number of coordinates in a data table starting at C+2, the form of the source data, and whether data is BCD or binary. Bits 00 to 07 contain the number (binary) of line coordinates less 1, m-1. Bits 08 to 12 are not used. Bit 13 specifies either f(x)=f(S) or f(x)=f(X<sub>m</sub>-S): OFF specifies f(x)=f(S) and ON specifies f(x)=f(X<sub>m</sub>-S). Bit 14 determines whether the output is BCD or binary: OFF specifies BCD and ON specifies binary. Bit 15 determines whether the input is BCD or binary: OFF specifies BCD and ON specifies binary.



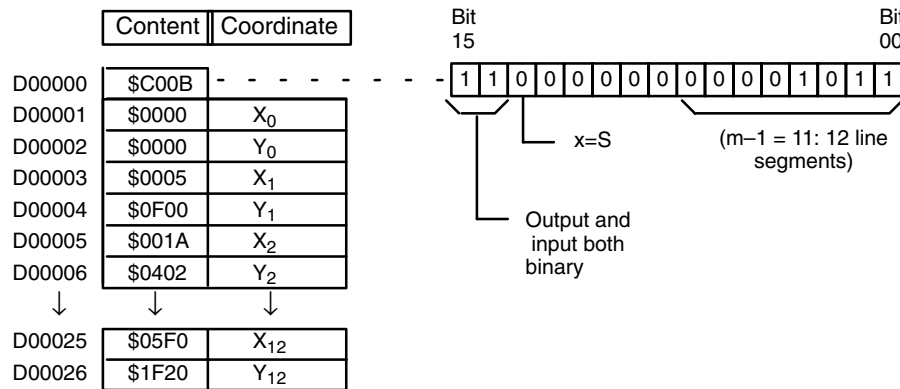
Enter the coordinates of the m+1 end points, which define the m line segments, as shown in the following table. Enter all coordinates in binary form.

Word	Coordinate
C+1	X <sub>0</sub>
C+2	Y <sub>0</sub>
C+3	X <sub>1</sub>
C+4	Y <sub>1</sub>
C+5	X <sub>2</sub>
C+6	Y <sub>2</sub>
↓	↓
C+(2m+1)	X <sub>m</sub> (max. X value)
C+(2m+2)	Y <sub>m</sub>

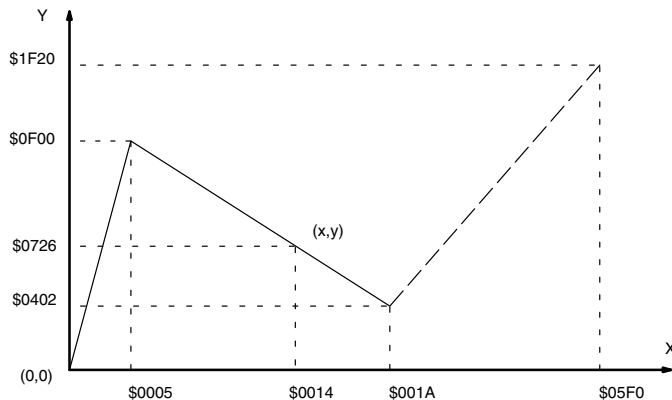
The following example demonstrates the construction of a linear extrapolation with 12 coordinates. The block of data is continuous, as it must be, from D00000 to D00026 (C to C + (2 × 12 + 2)). The input data is taken from CIO 0010, and the result is output to CIO 0011.



Address	Instruction	Operands
00000	LD	000000
00001	APR(142)	
		D00000
		0010
		0011



In this case, the source word, CIO 0010, contains 0014, and f(0014) = 0726 is output to R, CIO 0011.



## 5-24 PID and Related Instructions

### 5-24-1 PID CONTROL: PID(270)

<p><b>Ladder Symbol</b></p> <pre>           (270)           ┌───┴───┐           [ PID  S  C  D ]         </pre>	<p><b>Operand Data Areas</b></p> <p><b>S: Input word</b>            CIO, G, A, DM, DR, IR</p> <p><b>C: First parameter word</b>    CIO, G, A, DM,</p> <p><b>D: Output word</b>            CIO, G, A, DM, DR, IR</p>
---	---

**Caution** A total of 33 continuous words starting with P1 must be provided for PID(270) to operate correctly. Also, PID(270) may not operate dependably in any of the following situations: In subroutines, between IL(02) and ILC(03), between JMP(04) and JME(05), and in step programming (STEP(08)/SNXT(09)). Do not program PID(270) in these situations.

**Description**

When the execution condition is OFF, PID(270) is not executed. When the execution condition is ON, PID(270) carries out PID control according to the designated parameters. It takes the specified input range of binary data from the contents of input word S and carries out the PID action according to the parameters that are set. The result is then stored as the manipulated variable in output word D.

If the settings are not within the range of the PID parameters, the Error Flag (A5003) will turn ON and the PID action will not be executed. The Error Flag will also turn ON if the actual sampling period is two or more times the sampling period that has been set. In this case, however, the PID action will still be executed.



If the manipulated variable after the PID action exceeds the upper limit, the Greater Than (>) Flag (A50005) will turn ON and the result will be output at the upper limit. If the manipulated variable after the PID action is less than the lower limit, the Less Than (<) Flag (A50007) will turn ON and the result will be output at the lower limit.

PID parameter words range from C through C+38. The PID parameters are configured as shown below.

Word	15 to 12	11 to 8	7 to 4	3 to 0
C	Set value (SV)			
C+1	Proportional band (P)			
C+2	Tik = Integral constant (See note.)			
C+3	Tdk = Derivative constant (See note.)			
C+4	Sampling period ( $\tau$ )			
C+5	2-PID parameter ( $\alpha$ )			PID forward/ reverse designation
C+6	Manipulated variable output limit control	Input range	Integral and derivative unit	Output range
C+7	Manipulated variable output lower limit			
C+8	Manipulated variable output upper limit			
C+9 to C+38	Work area (Cannot be accessed directly from program.)			

**Note** The values set for words C+2 and C+3 will vary according to the unit designated in bits 04 to 07 of C+6.

**Parameters**

Item	Contents	Setting range
Set value (SV)	The target value of the process being controlled.	Binary data (of the same number of bits as specified for the input range)
Proportional band	The parameter for P action expressing the proportional control range/total control range.	0001 to 9999 (4-digit BCD); (0.1% to 999.9%, in units of 0.1%)
Tik	A constant expressing the strength of the integral action. As this value increases, the integral strength decreases.	0001 to 8191 (4-digit BCD); (9999 = Integral operation not executed) (See note 1.)
Tdk	A constant expressing the strength of the derivative action. As this value increases, the derivative strength decreases.	0001 to 8191 (4-digit BCD) (0000 = Derivative operation not executed) (See note 1.)
Sampling period ( $\tau$ )	Sets the period for executing the PID action.	0001 to 9999 (4-digit BCD); (0.01 to 99.99 s, in units of 10 ms)
2-PID parameter ( $\alpha$ )	The input filter coefficient. Normally use 0.65 (i.e., a setting of 000). The filter efficiency decreases as the coefficient approaches 0.	000: $\alpha = 0.65$ Setting from 100 to 199 means that the value of the rightmost two digits is set from $\alpha = 0.00$ to $\alpha = 0.99$ . (See note 2.)
PID forward/reverse designation	Determines the direction of the proportional action.	0: Reverse action 1: Forward action
Manipulated variable output limit control	Determines whether or not limit control will apply to the manipulated variable output.	0: Disabled (no limit control) 1: Enabled (limit control)

Item	Contents	Setting range
Input range	The number of input data bits.	0: 8 bits      5: 13 bits 1: 9 bits      6: 14 bits
Output range	The number of output data bits. {The number of output bits is automatically the same as the number of input bits.}	2: 10 bits    7: 15 bits 3: 11 bits    8: 16 bits 4: 12 bits
Integral and derivative unit	Determines the unit for expressing the integral and derivative constants.	1: Sampling period multiple 9: Time (unit: 100 ms)
Manipulated variable output lower limit	The lower limit for when the manipulated variable output limit is enabled.	0000 to FFFF (binary) (See note 3.)
Manipulated variable output upper limit	The upper limit for when the manipulated variable output limit is enabled.	0000 to FFFF (binary) (See note 3.)

- Note**
1. When the unit is designated as 1, the range is from 1 to 8,191 times the period. When the unit is designated as 9, the range is from 0.1 to 819.1 s. When 9 is designated, set the integral and derivative times to within a range of 1 to 8,191 times the sampling period.
  2. Setting the 2-PID parameter ( $\alpha$ ) to 000 yields 0.65, the normal value.
  3. When the manipulated variable output limit control is enabled (i.e., set to "1"), set the values as follows:  
 $0000 \leq \text{lower limit} \leq \text{upper limit} \leq \text{output range maximum value}$

**PID CONTROL Action**

**Execution Condition OFF**

All data that has been set is retained. When the execution condition is OFF, the manipulated variable can be written to the output word (D) to achieve manual control.

**Rising Edge of the Execution Condition**

The work area is initialized based on the PID parameters that have been set and the PID control action is begin. Sudden and radical changes in the manipulated variable output are not made when starting action to avoid adverse affect on the controlled system (bumpless operation).

When PID parameters are changed, they first become valid when the execution condition changes from OFF to ON.

**Execution Condition ON**

The PID action is executed at the intervals based on the sampling period, according to the PID parameters that have been set.

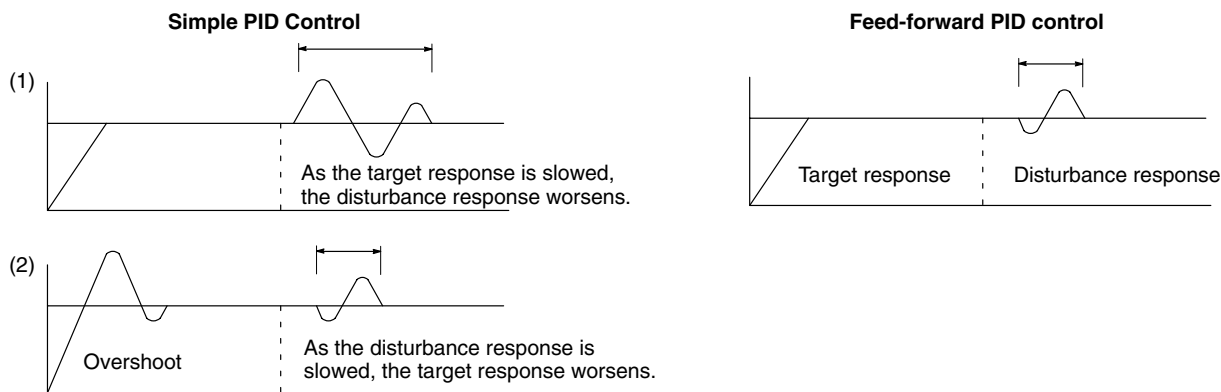
**Sampling Period and PID Execution Timing**

The sampling period is the time interval to retrieve the measurement data for carrying out a PID action. PID(270), however, is executed according to CPU cycle, so there may be cases where the sampling period is exceeded. In such cases, the time interval until the next sampling is reduced.

**PID Control Method**

PID control actions are executed by means of PID control with feed-forward control (two degrees of freedom).

When overshooting is prevented with simple PID control, stabilization of disturbances is slowed (1). If stabilization of disturbances is speeded up, on the other hand, overshooting occurs and response toward the target value is slowed (2). With feed-forward PID control, there is no overshooting, and response toward the target value and stabilization of disturbances can both be speeded up (3).



**Control Actions**

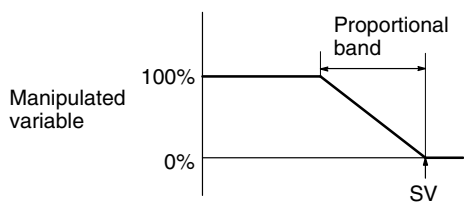
**Proportional Action (P)**

Proportional action is an operation in which a proportional band is established with respect to the set value (SV), and within that band the manipulated variable (MV) is made proportional to the deviation. An example for reverse operation is shown in the following illustration

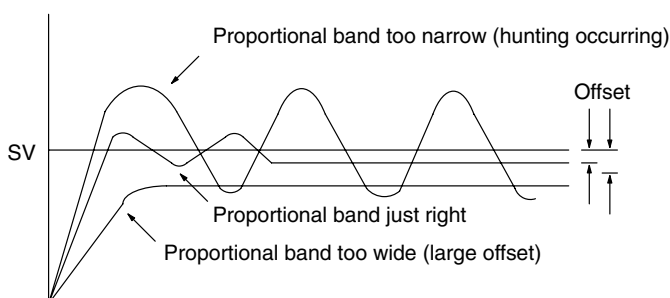
If the proportional action is used and the present value (PV) becomes smaller than the proportional band, the manipulated variable (MV) is 100% (i.e., the maximum value). Within the proportional band, the MV is made proportional to the deviation (the difference between from SV and PV) and gradually decreased until the SV and PV match (i.e., until the deviation is 0), at which time the MV will be 0% (i.e., the minimum value). The MV will also be 0% when the PV is larger than the SV.

The proportional band is expressed as a percentage of the total input range. The smaller the proportional band, the larger the proportional constant and the stronger the corrective action will be. With proportional action an offset (residual deviation) generally occurs, but the offset can be reduced by making the proportional band smaller. If it is made too small, however, hunting will occur.

**Proportional Action (Reverse Action)**

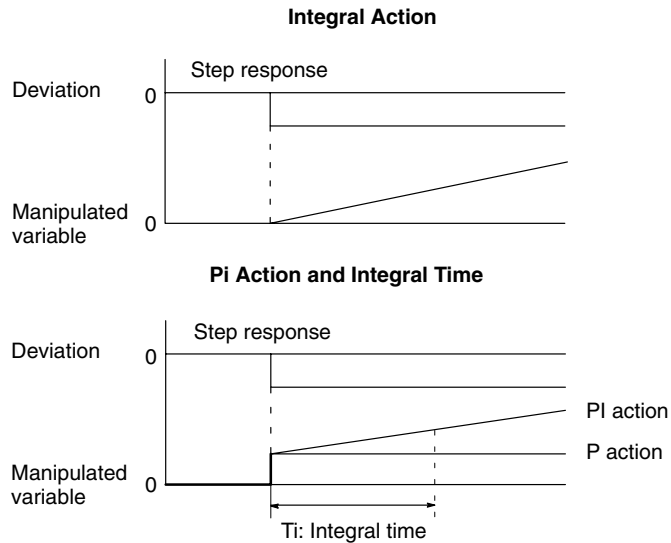


**Adjusting the Proportional Band**



**Integral Action (I)**

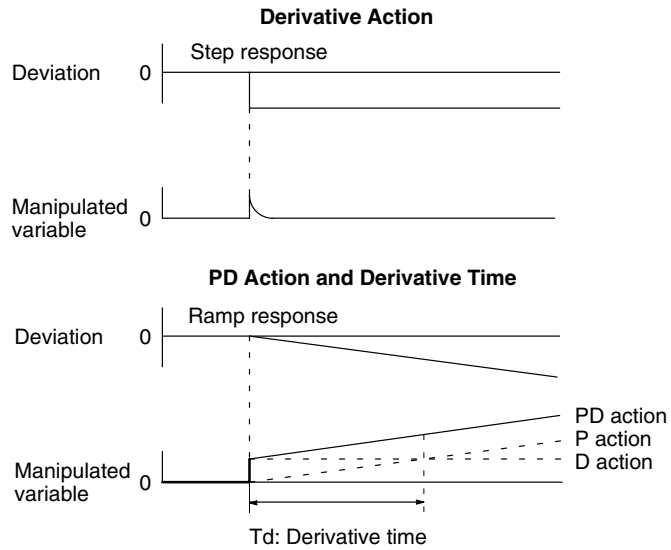
Combining integral action with proportional action reduces the offset according to the time that has passed. The strength of the integral action is indicated by the integral time, which is the time required for the manipulated variable of the integral action to reach the same level as the manipulated variable of the proportional action with respect to the step deviation, as shown in the following illustration. The shorter the integral time, the stronger the correction by the integral action will be. If the integral time is too short, the correction will be too strong and will cause hunting to occur.



**Derivative Action (D)**

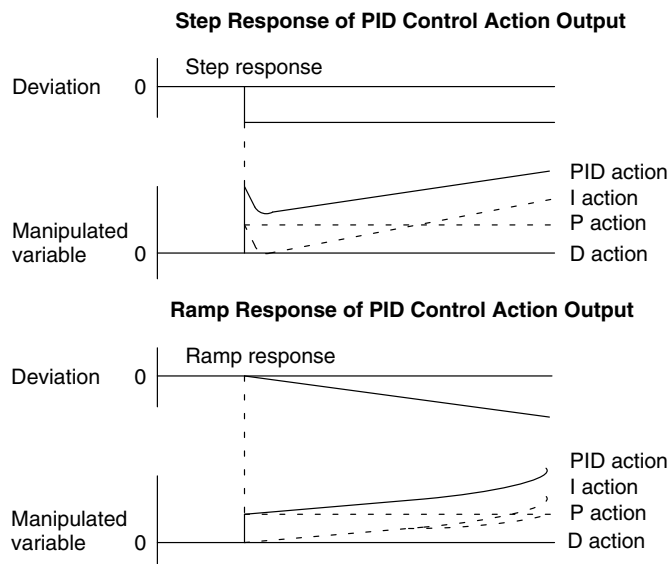
Proportional action and integral action both make corrections with respect to the control results, so there is inevitably a response delay. Derivative action compensates for that drawback. In response to a sudden disturbance it delivers a large manipulated variable and rapidly restores the original status. A correction is executed with the manipulated variable made proportional to the incline (derivative coefficient) caused by the deviation.

The strength of the derivative action is indicated by the derivative time, which is the time required for the manipulated variable of the derivative action to reach the same level as the manipulated variable of the proportional action with respect to the step deviation, as shown in the following illustration. The longer the derivative time, the stronger the correction by the derivative action will be.



**PID Action**

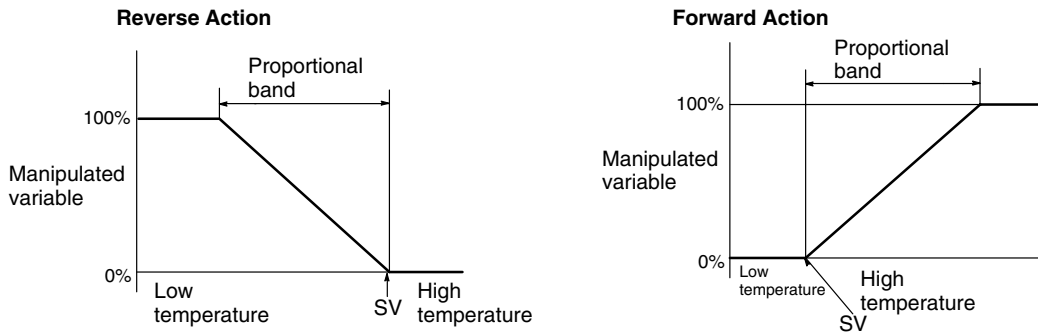
PID action combines proportional action (P), integral action (I), and derivative action (D). It produces superior control results even for control objects with dead time. It employs proportional action to provide smooth control without hunting, integral action to automatically correct any offset, and derivative action to speed up the response to disturbances.



**Direction of Action**

When using PID action, select either of the following two control directions. In either direction, the MV increases as the difference between the SV and the PV increases.

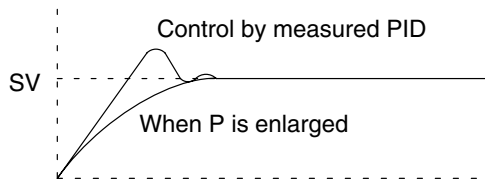
- Forward action: MV is increased when the PV is larger than the SV.
- Reverse action: MV is increased when the PV is smaller than the SV.



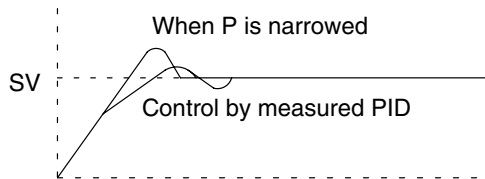
**Adjusting PID Parameters**

The general relationship between PID parameters and control status is shown below.

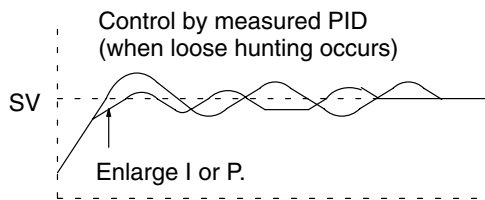
- When it is not a problem if a certain amount of time is required for stabilization (settlement time), but it is important not to cause overshooting, then enlarge the proportional band.



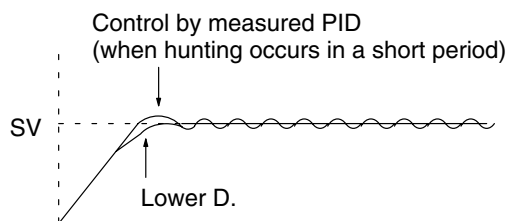
- When overshooting is not a problem but it is desirable to quickly stabilize control, then narrow the proportional band. If the proportional band is narrowed too much, however, then hunting may occur.



- When there is broad hunting, or when operation is tied up by overshooting and undershooting, it is probably because integral action is too strong. The hunting will be reduced if the integral time is increased or the proportional band is enlarged.



- If the period is short and hunting occurs, it may be that the control system response is quick and the derivative action is too strong. In that case, set the derivative action lower.



**Precautions**

PID data must be within prescribed ranges.

**Note** Refer to page 101 for general precautions on operand data areas.

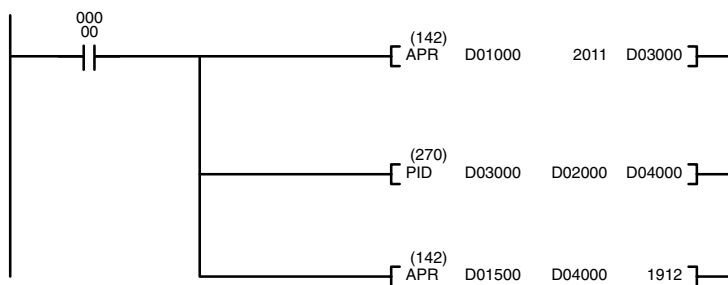
**Flags**

- ER (A50003): PID data is outside of the allowable range.  
The actual sampling period is two or more times the sampling period that has been set.  
The content of a\*DM word is not BCD when set for BCD.
- CY (A50004): The PID action is executed.
- > (A50005): The MV after the PID action exceeds the upper limit.
- < (A50007): The MV after the PID action is less than the lower limit.

**Scaling Data for PID(270)**

Scaling cannot be performed directly with PID(270). To perform scaling, use PID(270) in combination with APR(142) as shown below.

Measurement value data and manipulated variable output data used with PID(270) are between 0000 to FFFF hexadecimal. Therefore, normal operation cannot be achieved if PID(270) is executed for devices that have different data ranges without first converting the data. It is necessary to perform scaling or to take countermeasures before and/or after executing PID(270).



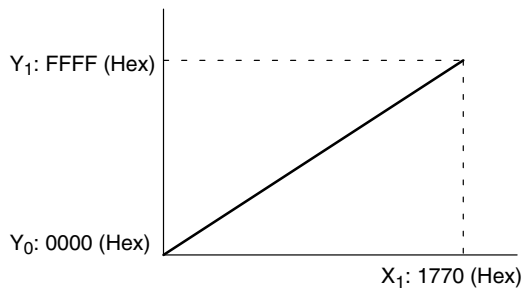
**Outline of Operation**

- 1, 2, 3...**
1. Using APR(142), scale the input value (CIO 2011) so that the value falls within the range applicable to PID(270).
  2. Execute PID(270) using the scaled measurement value as the input (D03000).
  3. Perform scaling again so that the manipulated variable output data (D04000) can be written as the output data (CIO 1912). Refer to the following page for details on the control data for APR(142) and PID(270).

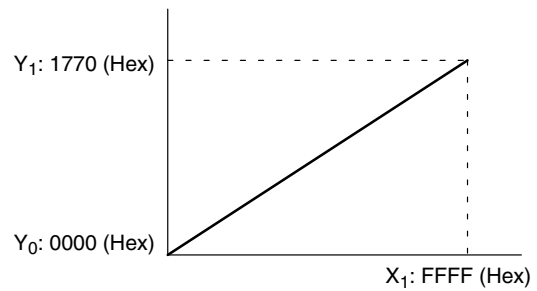
Control Data Settings

	Address	Value	Contents
Parameters for APR instruction 1	D01000	C001	Binary/f(x)=f(S)/ No. of coordinates = 2
	D01001	1770	Max. X value
	D01002	0000	Y <sub>0</sub>
	D01003	1770	X <sub>1</sub>
	D01004	FFFF	Y <sub>1</sub>
Parameters for APR instruction 2	D01500	C001	Binary/f(x)=f(S)/ No. of coordinates = 2
	D01501	FFFF	Max. X value
	D01502	0000	Y <sub>0</sub>
	D01503	FFFF	X <sub>1</sub>
	D01504	1770	Y <sub>1</sub>
Parameters for PID instruction	D02000	8000	Set value
	D02001	0260	Proportional band
	D02002	0290	Integral constant
	D02003	0040	Derivative constant
	D02004	0010	Sampling period
	D02005	0000	2-PID parameter, etc.
	D02006	1898	Range settings, etc.
	D02007	0000	Manipulated variable output lower limit
	D02008	FFFF	Manipulated variable output upper limit
	D02009 to D02038	---	---

APR Instruction 1



APR Instruction 2





### 5-24-2 LIMIT CONTROL: LMT(271)

<p><b>Ladder Symbol</b></p> <p><b>Variations</b></p> <p>↑LMT(271)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: Input word</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>C: First limit word</b> CIO, G, A, T, C, DM</p> <p><b>D: Output word</b> CIO, G, A, T, C, DM, DR, IR</p>
---	--

**Description**

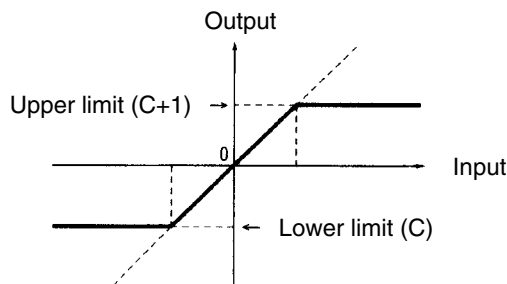
When the execution condition is OFF, LMT(271) is not executed. When the execution condition is ON, LMT(271) controls output data according to whether or not the specified input data (signed 16-bit binary) is within the upper and lower limits. The contents of words C and C+1 are as follows:

C	Lower limit data (minimum output data)
C+1	Upper limit data (maximum output data)

If the input data (S) is less than the lower limit (C), the lower limit data will be output to D and the Less Than Flag (A50007) will turn ON.

If the input data (S) is greater than the upper limit (C+1), the upper limit data will be output to D and the Greater Than Flag (A50005) will turn ON.

If the input data (S) is greater than or equal to the lower limit (C) and less than or equal to the upper limit (C+1), the input data (S) will be output to D.



**Precautions**

The lower limit (C) must be less than or equal to the upper limit (C+1).

**Note** Refer to page 101 for general precautions on operand data areas.

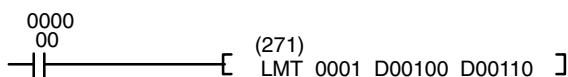
**Flags**

- ER (A50003): The upper limit setting is less than the lower limit.  
The content of a\*DM word is not BCD when set for BCD.
- > (A50005): The input data (S) is greater than the upper limit (C+1).
- EQ (A50006): The output data is all zeros.
- < (A50007): The input data (S) is less than the lower limit (C).
- N (A50008): The output data is a negative number.

**Example**

When CIO 000000 turns ON in the following example, one of the following will occur:

- If the binary content of CIO 0001 is within the range specified by the content of D00100 and D00101, the content of CIO 0001 will be output to D00110.
- If the binary content of CIO 0001 is greater than the content of D00101, the content of D00101 will be output to D00110.
- If the binary content of CIO 0001 is less than the content of D00100, the content of D00100 will be output to D00110.



Address	Instruction	Operands
00000	LD	000000
00001	LMT(271)	
		0001
		D00100
		D00110



### 5-24-3 DEAD-BAND CONTROL: BAND(272)

<p><b>Ladder Symbol</b></p> <p><b>Variations</b> ↑BAND(272)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: Input word</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>C: First limit word</b> CIO, G, A, T, C, DM</p> <p><b>D: Output word</b> CIO, G, A, T, C, DM, DR, IR</p>
---	--

#### Description

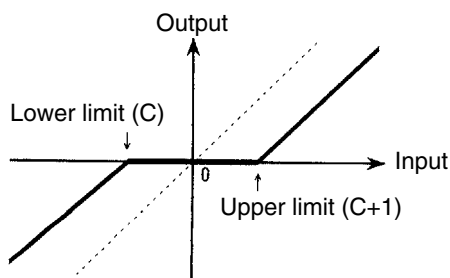
When the execution condition is OFF, BAND(272) is not executed. When the execution condition is ON, BAND(272) controls output data according to whether or not the specified input data (signed 16-bit binary) is within the upper and lower limits (dead band). The contents of words C and C+1 are as follows:

C	Lower limit data (dead band lower limit)
C+1	Upper limit data (dead band upper limit)

If the input data (S) is less than the lower limit (C), the difference between the input data minus the lower limit data will be output to D and the Less Than Flag (A50007) will turn ON.

If the input data (S) is greater than the upper limit (C+1), the difference between the input data minus the upper limit data will be output to D and the Greater Than Flag (A50005) will turn ON.

If the input data (S) is greater than or equal to the lower limit (C) and less than or equal to the upper limit (C+1), 0000 will be output to D and the Equals Flag (A50006) will turn ON.



If the output data is less than 8000 or greater than 7FFF, the sign will reverse. For example, if the lower limit is 0100 and the input data is 8000, the output data will be as follows:

$$8000 - 0100 = 7F00$$

$$(-32,768)_{10} - (256)_{10} = (32,512)_{10}$$

**Precautions**

The lower limit (C) must be less than or equal to the upper limit (C+1).

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

- ER (A50003): The upper limit setting is less than the lower limit.  
The content of a\*DM word is not BCD when set for BCD.
- > (A50005): The input data (S) is greater than the upper limit (C+1).
- EQ (A50006): The output data is all zeros.
- < (A50007): The input data (S) is less than the lower limit (C).
- N (A50008): The output data is a negative number.

**Example**

When CIO 000000 turns ON in the following example, one of the following will occur:

- If the binary content of CIO 0001 is within the range specified by the content of D00100 and D00101, 0000 will be output to D00110.
- If the binary content of CIO 0001 is greater than the content of D00101, the result of (CIO 0001 minus D00101) will be output to D00110.
- If the binary content of CIO 0001 is less than the content of D00100, the result of (CIO 0001 minus D00100) will be output to D00110.



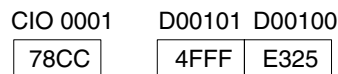
Address	Instruction	Operands
00000	LD	0000000
00001	BAND(272)	
		0001
		D00100
		D00110



$$1000 < 41F5 < 4FFF$$

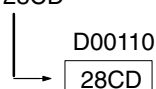


A50006 (=) ON



$$78CC < 4FFF$$

$$78CC - 4FFF = 28CD$$



A50005 (>) ON

### 5-24-4 DEAD-ZONE CONTROL: ZONE(273)

Ladder Symbol	Operand Data Areas
	<p><b>S: Input word</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>C: First bias word</b> CIO, G, A, T, C, DM</p> <p><b>D: Output word</b> CIO, G, A, T, C, DM, DR, IR</p>
<p><b>Variations</b></p> <p>↑ZONE(273)</p>	

**Description**

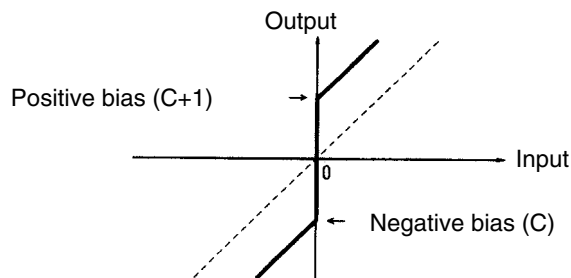
When the execution condition is OFF, ZONE(273) is not executed. When the execution condition is ON, ZONE(273) adds the specified bias to the specified input data (signed 16-bit binary) and places the result in a specified word. The contents of words C and C+1 are as follows:

C	Negative bias
C+1	Positive bias

If the input data (S) is less than zero, the input data plus the negative bias will be output to D and the Less Than Flag (A50007) will turn ON.

If the input data (S) is greater than zero, the input data plus the positive bias will be output to D and the Greater Than Flag (A50005) will turn ON.

If the input data (S) is equal to zero, 0000 will be output to D and the Equals Flag (A50006) will turn ON.



If the output data is less than 8000 or greater than 7FFF, the sign will reverse. For example, if the negative bias is FF00 and the input data is 8000, the output data will be as follows:

$$\begin{array}{rcl} 8000 & +FF00 & = 7F00 \\ (-32,768)_{10} & (-256)_{10} & (32,512)_{10} \end{array}$$

**Precautions**

The negative bias (C) must be less than or equal to the positive bias (C+1).

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

- ER (A50003): C is less than C+1.  
The content of a\*DM word is not BCD when set for BCD.
- > (A50005): The input data (S) is greater than 0000.
- EQ (A50006): The output data is all zeros.
- < (A50007): The input data (S) is less than 0000.
- N (A50008): The output data is a negative number.

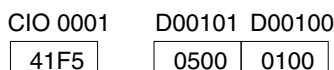
**Example**

When CIO 000000 turns ON in the following example, one of the following will occur:

- If the binary content of CIO 0001 is less than zero, the result of CIO 0001 plus D00100 will be output to D00110.
- If the binary content of CIO 0001 is equal to zero, 0000 will be output to D00110.
- If the binary content of CIO 0001 is greater than the content of D00100, the result of (CIO 0001 plus D00100) will be output to D00110.



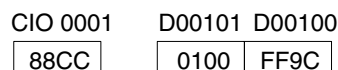
Address	Instruction	Operands
00000	LD	000000
00001	ZONE(273)	
		0001
		D00100
		D00110



41F5 > 0  
41F5 + 0500 = 46F5



A50005 (>) ON



88CC < 0  
88CC + FF9C = 8868



A50007 (<) ON  
A50008 (N) ON

# 5-25 Logic Instructions

The logic instructions perform logic operations on word data.

## 5-25-1 LOGICAL AND: ANDW(130)

Ladder Symbol	Operand Data Areas
	<b>I<sub>1</sub>: Input 1</b> CIO, G, A, T, C, #, DM, DR, IR <b>I<sub>2</sub>: Input 2</b> CIO, G, A, T, C, #, DM, DR, IR <b>R: Result word</b> CIO, G, A, DM, DR, IR
<b>Variations</b> ↑ ANDW(130)	

**Description**

When the execution condition is OFF, ANDW(130) is not executed. When the execution condition is ON, ANDW(130) logically AND's corresponding bits of I<sub>1</sub> and I<sub>2</sub> and places the result in R.

**Precautions**

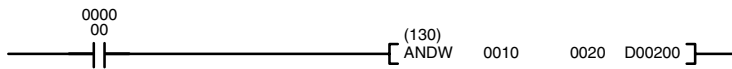
Refer to page 101 for general precautions on operand data areas.

**Flags**

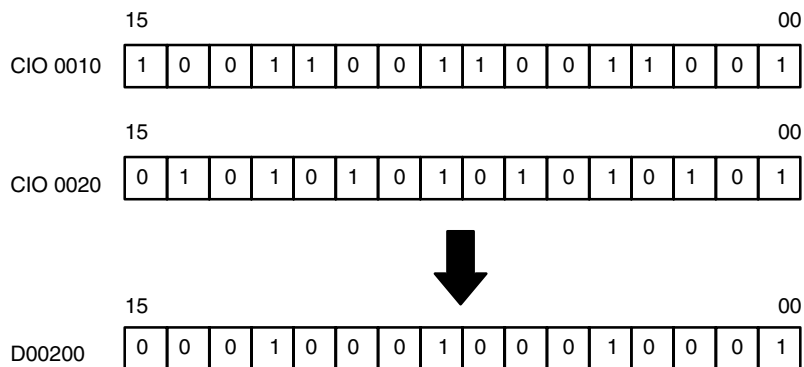
- ER (A50003): Content of \*DM word is not BCD when set for BCD.
- EQ (A50006): The result is 0.
- N (A50008): Shows the status of bit 15 of R after execution.

**Example**

When CIO 000000 is ON in the following example, the logical AND is taken of corresponding bits in CIO 0010 and CIO 0020 and the results is placed in corresponding bits of D00200.



Address	Instruction	Operands
00000	LD	000000
00001	ANDW(130)	
		0010
		0020
		D00200



### 5-25-2 LOGICAL OR: ORW(131)

<p><b>Ladder Symbol</b></p> <p><b>Variations</b></p> <p>↑ ORW(131)</p>	<p><b>Operand Data Areas</b></p> <p><b>I<sub>1</sub>: Input 1</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>I<sub>2</sub>: Input 2</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>R: Result word</b> CIO, G, A, DM, DR, IR</p>
--	---

**Description** When the execution condition is OFF, ORW(131) is not executed. When the execution condition is ON, ORW(131) logically OR's corresponding bits of I<sub>1</sub> and I<sub>2</sub> and places the result in R.

**Precautions** Refer to page 101 for general precautions on operand data areas.

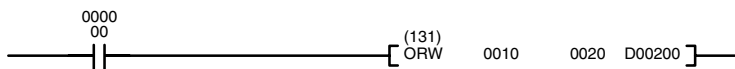
**Flags**

ER (A50003): Content of \*DM word is not BCD when set for BCD.

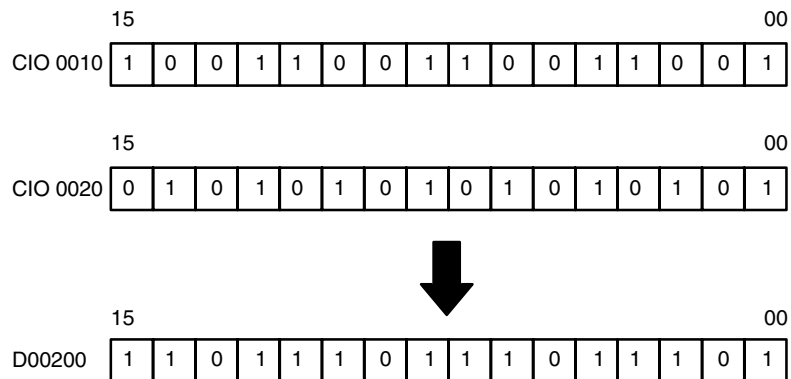
EQ (A50006): The result is 0.

N (A50008): Shows the status of bit 15 of R after execution.

**Example** When CIO 000000 is ON in the following example, the logical OR is taken of corresponding bits in CIO 0010 and CIO 0020 and the results is placed in corresponding bits of D00200.



Address	Instruction	Operands
00000	LD	000000
00001	ORW(131)	
		0010
		0020
		D00200



### 5-25-3 EXCLUSIVE OR: XORW(132)

<p><b>Ladder Symbol</b></p> <p><b>Variations</b></p> <p>↑ XORW(132)</p>	<p><b>Operand Data Areas</b></p> <p><b>I<sub>1</sub>: Input 1</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>I<sub>2</sub>: Input 2</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>R: Result word</b> CIO, G, A, DM, DR, IR</p>
---	---

**Description** When the execution condition is OFF, XORW(132) is not executed. When the execution condition is ON, XORW(132) exclusively OR's corresponding bits of I<sub>1</sub> and I<sub>2</sub> and places the result in R.

**Precautions**

Refer to page 101 for general precautions on operand data areas.

**Flags**

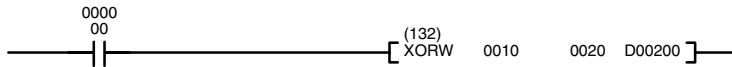
ER (A50003): Content of \*DM word is not BCD when set for BCD.

EQ (A50006): The result is 0.

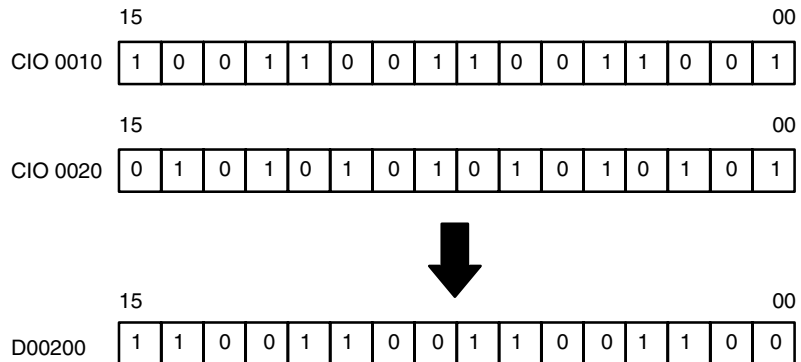
N (A50008): Shows the status of bit 15 of R after execution.

**Example**

When CIO 000000 is ON in the following example, the logical exclusive OR is taken of corresponding bits in CIO 0010 and CIO 0020 and the results is placed in corresponding bits of D00200.



Address	Instruction	Operands
00000	LD	000000
00001	XORW(132)	
		0010
		0020
		D0020



**5-25-4 EXCLUSIVE NOR: XNRW(133)**

Ladder Symbol	Operand Data Areas
	<b>I<sub>1</sub>: Input 1</b> CIO, G, A, T, C, #, DM, DR, IR <b>I<sub>2</sub>: Input 2</b> CIO, G, A, T, C, #, DM, DR, IR <b>R: Result word</b> CIO, G, A, DM, DR, IR
<b>Variations</b> ↑ XNRW(133)	

**Description**

When the execution condition is OFF, XNRW(133) is not executed. When the execution condition is ON, XNRW(133) exclusively NOR's corresponding bits of I<sub>1</sub> and I<sub>2</sub> and places the result in R.

**Precautions**

Refer to page 101 for general precautions on operand data areas.

**Flags**

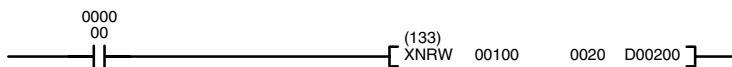
ER (A50003): Content of \*DM word is not BCD when set for BCD.

EQ (A50006): The result is 0.

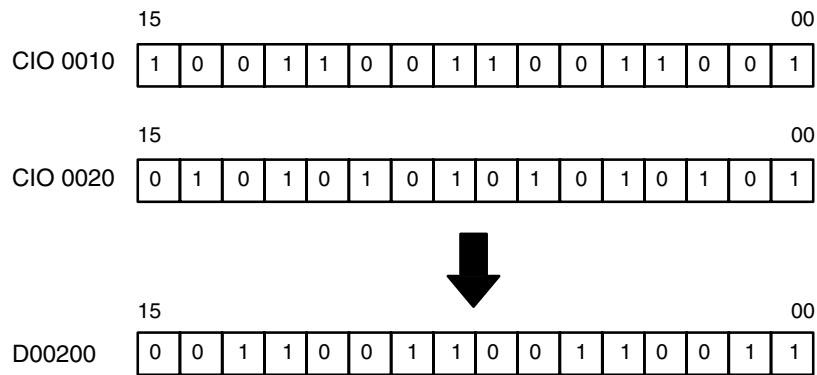
N (A50008): Shows the status of bit 15 of R after execution.

**Example**

When CIO 000000 is ON in the following example, the logical exclusive NOR is taken of corresponding bits in CIO 0010 and CIO 0020 and the results is placed in corresponding bits of D00200.



Address	Instruction	Operands
00000	LD	000000
00001	XNRW(133)	
		0010
		0020
		D0020



### 5-25-5 DOUBLE LOGICAL AND: ANDL(134)

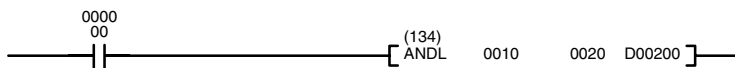
<b>Ladder Symbol</b>	<b>Operand Data Areas</b>
	<b>I<sub>1</sub>: Input 1</b> CIO, G, A, T, C, #, DM <b>I<sub>2</sub>: Input 2</b> CIO, G, A, T, C, #, DM <b>R: Result word</b> CIO, G, A, DM
<b>Variations</b>	
↑ ANDL(134)	

**Description** When the execution condition is OFF, ANDL(134) is not executed. When the execution condition is ON, ANDL(134) logically AND's corresponding bits of I<sub>1</sub> and I<sub>1</sub>+1 with I<sub>2</sub> and I<sub>2</sub>+1 and places the result in R and R+1.

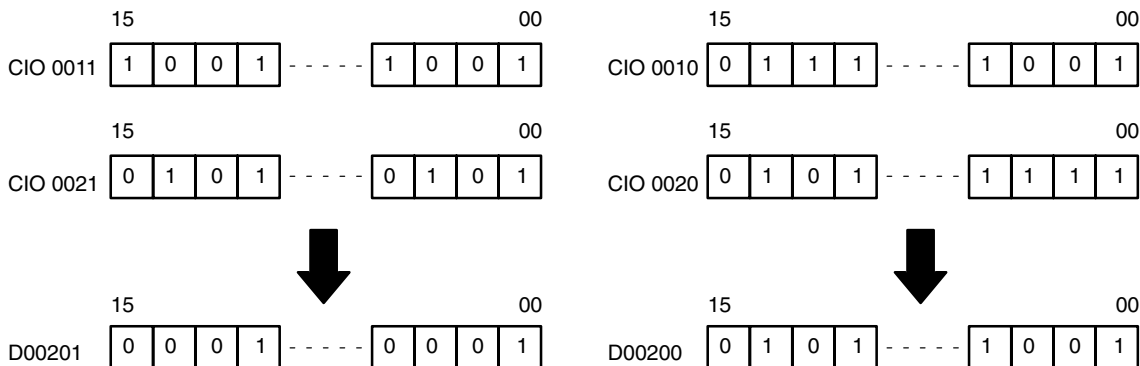
**Precautions** Refer to page 101 for general precautions on operand data areas.

**Flags**  
 ER (A50003): Content of \*DM word is not BCD when set for BCD.  
 EQ (A50006): The result is 0.  
 N (A50008): Shows the status of bit 15 of R+1 after execution.

**Example** When CIO 000000 is ON in the following example, the logical AND is taken of corresponding bits in CIO 0010 to CIO 0011 and CIO 0020 to CIO 0020 and the results is placed in corresponding bits of D00200 and D00201.



Address	Instruction	Operands
00000	LD	000000
00001	ANDL(134)	
		0010
		0020
		D0020





### 5-25-6 DOUBLE LOGICAL OR: ORWL(135)

<p><b>Ladder Symbol</b></p> <p><b>Variations</b></p> <p>↑ ORWL(135)</p>	<p><b>Operand Data Areas</b></p> <p><b>I<sub>1</sub>: Input 1</b> CIO, G, A, T, C, #, DM</p> <p><b>I<sub>2</sub>: Input 2</b> CIO, G, A, T, C, #, DM</p> <p><b>R: Result word</b> CIO, G, A, DM</p>
---	---

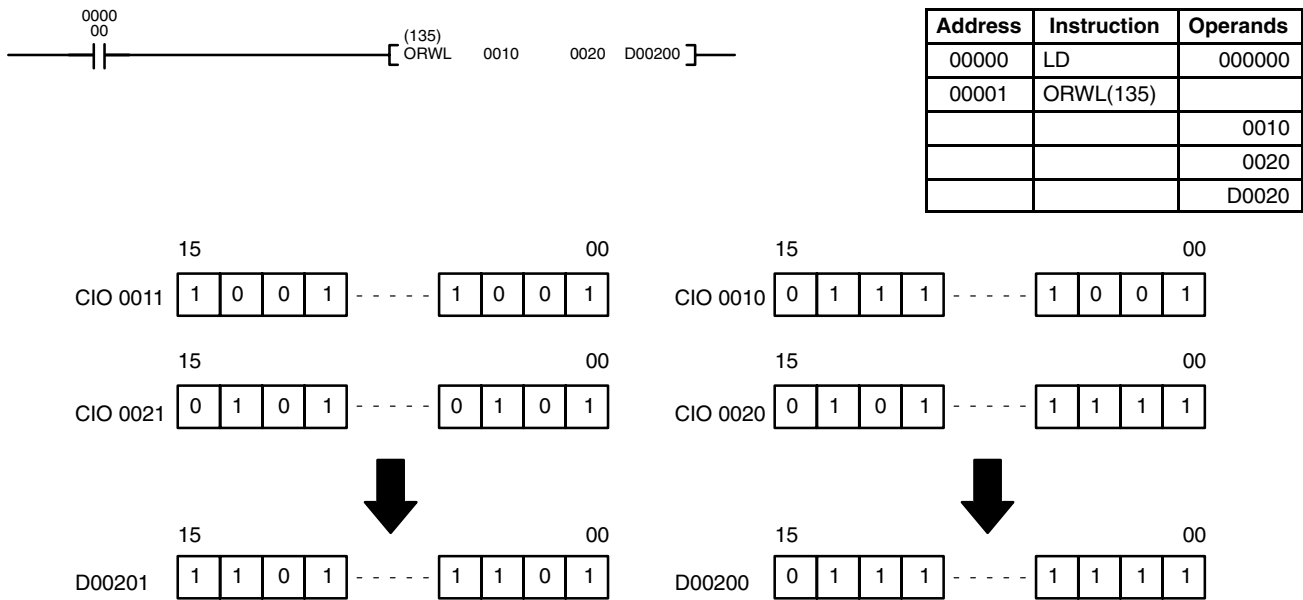
**Description** When the execution condition is OFF, ORWL(135) is not executed. When the execution condition is ON, ORWL(135) logically OR's corresponding bits of I<sub>1</sub> and I<sub>1</sub>+1 with I<sub>2</sub> and I<sub>2</sub>+1 and places the result in R and R+1.

**Precautions** Refer to page 101 for general precautions on operand data areas.

**Flags**

ER (A50003): Content of \*DM word is not BCD when set for BCD.  
 EQ (A50006): The result is 0.  
 N (A50008): Shows the status of bit 15 of R+1 after execution.

**Example** When CIO 000000 is ON in the following example, the logical OR is taken of corresponding bits in CIO 0010 to CIO 0011 and CIO 0020 to CIO 0020 and the results is placed in corresponding bits of D00200 and D00201.



### 5-25-7 DOUBLE EXCLUSIVE OR: XORL(136)

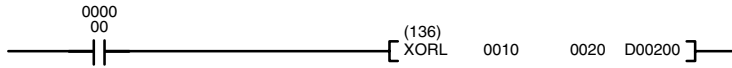
<p><b>Ladder Symbol</b></p> <p><b>Variations</b></p> <p>↑ XORL(136)</p>	<p><b>Operand Data Areas</b></p> <p><b>I<sub>1</sub>: Input 1</b> CIO, G, A, T, C, #, DM</p> <p><b>I<sub>2</sub>: Input 2</b> CIO, G, A, T, C, #, DM</p> <p><b>R: Result word</b> CIO, G, A, DM</p>
---	---

**Description** When the execution condition is OFF, XORL(136) is not executed. When the execution condition is ON, XORL(136) exclusively OR's the contents of I<sub>1</sub> and I<sub>1</sub>+1 with I<sub>2</sub> and I<sub>2</sub>+1 bit-by-bit and places the result in R and R+1.

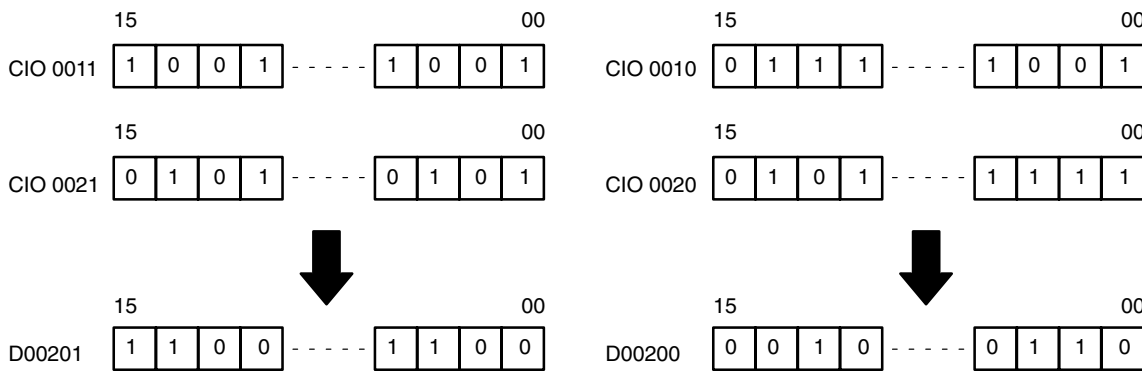
**Precautions** Refer to page 101 for general precautions on operand data areas.

- Flags**
- ER (A50003): Content of \*DM word is not BCD when set for BCD.
  - EQ (A50006): The result is 0.
  - N (A50008): Shows the status of bit 15 of R+1 after execution.

**Example** When CIO 000000 is ON in the following example, the logical exclusive OR is taken of corresponding bits in CIO 0010 to CIO 0011 and CIO 0020 to CIO 0020 and the results is placed in corresponding bits of D00200 and D00201.



Address	Instruction	Operands
00000	LD	000000
00001	XORL(136)	
		0010
		0020
		D00200



### 5-25-8 DOUBLE EXCLUSIVE NOR: XNRL(137)

Ladder Symbol	Operand Data Areas
	<b>I<sub>1</sub>: Input 1</b> CIO, G, A, T, C, #, DM <b>I<sub>2</sub>: Input 2</b> CIO, G, A, T, C, #, DM <b>R: Result word</b> CIO, G, A, DM
<b>Variations</b> ↑ XNRL(137)	

**Description** When the execution condition is OFF, XNRL(137) is not executed. When the execution condition is ON, XNRL(137) exclusively NOR's the contents of I<sub>1</sub> and I<sub>1</sub>+1 with I<sub>2</sub> and I<sub>2</sub>+1 bit-by-bit and places the result in R and R+1.

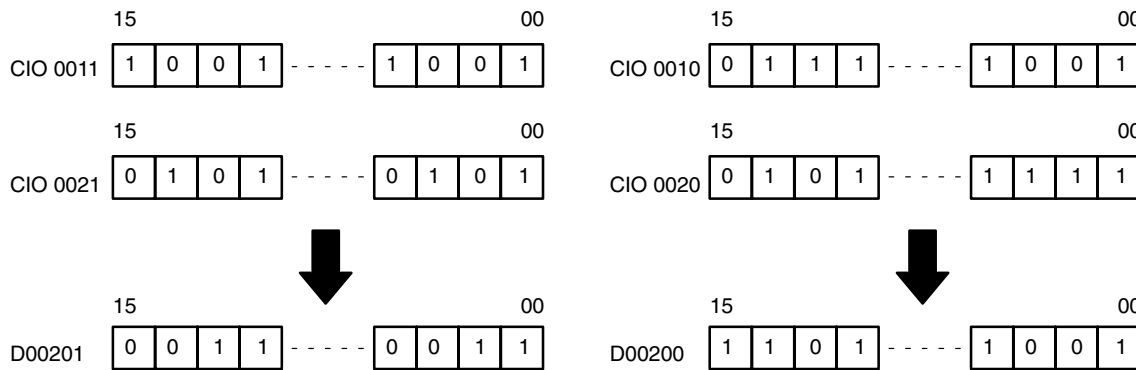
**Precautions** Refer to page 101 for general precautions on operand data areas.

- Flags**
- ER (A50003): Content of \*DM word is not BCD when set for BCD.
  - EQ (A50006): The result is 0.
  - N (A50008): Shows the status of bit 15 of R+1 after execution.

**Example** When CIO 000000 is ON in the following example, the logical exclusive NOR is taken of corresponding bits in CIO 0010 to CIO 0011 and CIO 0020 to CIO 0020 and the results is placed in corresponding bits of D00200 and D00201.



Address	Instruction	Operands
00000	LD	000000
00001	XNRL(137)	
		0010
		0020
		D00200



### 5-25-9 COMPLEMENT: COM(138)

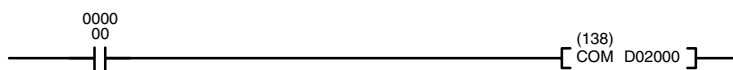
Ladder Symbol	Operand Data Area
	<b>Wd: Word</b> CIO, G, A, DM, DR, IR
<b>Variations</b> ↑ COM(138)	

**Description**      When the execution condition is OFF, COM(138) is not executed. When the execution condition is ON, COM(138) turns OFF all ON bits and turns ON all OFF bits in Wd.

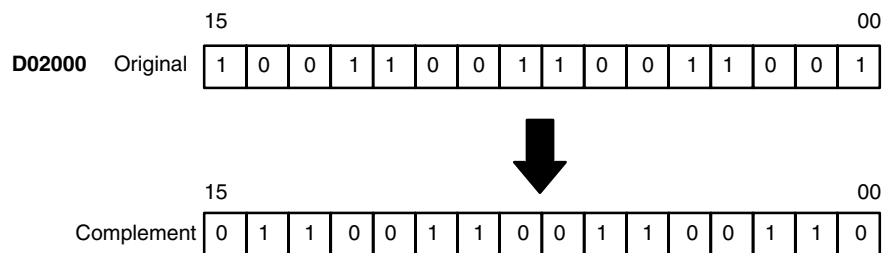
**Precautions**      Refer to page 101 for general precautions on operand data areas.

- Flags**
- ER (A50003): Content of \*DM word is not BCD when set for BCD.
  - EQ (A50006): The result is 0.
  - N (A50008): Shows the status of bit 15 of R after execution.

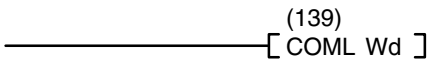
**Example**      When CIO 000000 is ON in the following example, the complement of the status of each bit in D02000 is taken and written back to D02000, i.e., the status of each bit is reversed.



Address	Instruction	Operands
00000	LD	000000
00001	COM(138)	
		D02000



### 5-25-10 DOUBLE COMPLEMENT: COML(139)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(139)  </p> <p><b>Variations</b></p> <p>↑ COML(139)</p>	<p><b>Operand Data Area</b></p> <p><b>Wd: Word</b>      CIO, G, A, DM</p>
---	---

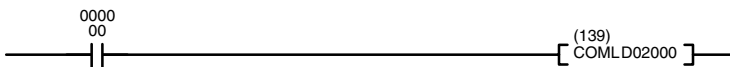
**Description**      When the execution condition is OFF, COML(139) is not executed. When the execution condition is ON, COML(139) turns OFF all ON bits and turns ON all OFF bits in Wd and Wd+1.

**Precautions**      Refer to page 101 for general precautions on operand data areas.

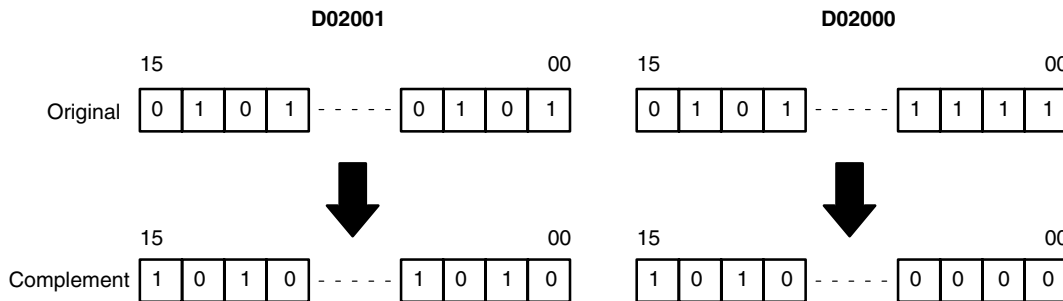
**Flags**

ER (A50003):    Content of \*DM word is not BCD when set for BCD.  
 EQ (A50006):    The result is 0.  
 N (A50008):     Shows the status of bit 15 of R+1 after execution.

**Example**      When CIO 000000 is ON in the following example, the complement of the status of each bit in D02000 and D02001 is taken and written back to D02000 and D02001, i.e., the status of each bit is reversed.



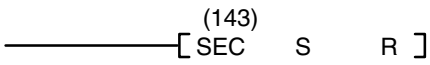
Address	Instruction	Operands
00000	LD	000000
00001	COML(139)	
00002		D02000



## 5-26 Time Instructions

The first two Time Instructions convert time formats. The last two Time Instructions add/subtract time from calendar values.

### 5-26-1 HOURS TO SECONDS: SEC(143)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(143)  </p> <p><b>Variations</b></p> <p>↑ SEC(143)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: 1<sup>st</sup> source word</b>    CIO, G, A, T, C, #, DM</p> <p><b>R: 1<sup>st</sup> result word</b>    CIO, G, A, DM</p>
--	---

**Description**      When the execution condition is OFF, SEC(143) is not executed. When the execution condition is ON, SEC(143) converts time notation in hours/minutes/seconds to an equivalent time in seconds only.

For the source data, the seconds are designated in bits 00 through 07 and the minutes are designated in bits 08 through 15 of S. The hours are designated in S+1. The maximum is thus 9,999 hours, 59 minutes, and 59 seconds.

The results are output to R and R+1. The maximum obtainable value is 35,999,999 seconds.

**Precautions**

S and S+1 must be BCD and must be in the proper hours/minutes/seconds format.

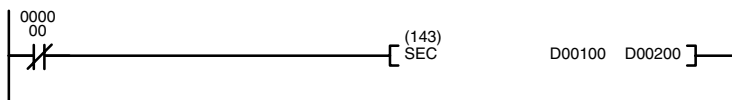
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

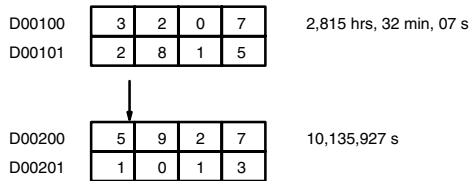
ER (A50003): S or S+1 are not BCD.  
 Number of seconds or minutes exceeds 59.  
 Content of \*DM word is not BCD when set for BCD.  
 EQ (A50006): ON when the result is 0.

**Example**

When 00000 is OFF (i.e., the execution condition is ON), the following instruction would convert the hours, minutes, and seconds given in D00100 and D00101 to seconds and store the results in D00200 and D00201 as shown.



Address	Instruction	Operands
00000	LD NOT	000000
00001	SEC(143)	
		D00100
		D00200



**5-26-2 SECONDS TO HOURS: HMS(144)**

<p><b>Ladder Symbol</b></p> <p><b>Variations</b></p> <p>↑ HMS(144)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: 1<sup>st</sup> source word</b> CIO, G, A, T, C, #, DM</p> <p><b>R: 1<sup>st</sup> result word</b> CIO, G, A, DM</p>
--	---

**Description**

When the execution condition is OFF, HMS(144) is not executed. When the execution condition is ON, HMS(144) converts time notation in seconds to an equivalent time in hours/minutes/seconds.

The number of seconds designated in S and S+1 is converted to hours/minutes/seconds and placed in R and R+1.

For the results, the seconds are placed in bits 00 through 07 and the minutes are placed in bits 08 through 15 of R. The hours are placed in R+1. The maximum will be 9,999 hours, 59 minutes, and 59 seconds.

**Precautions**

S+1 and S must be BCD and less than or equal to 3599 9999.

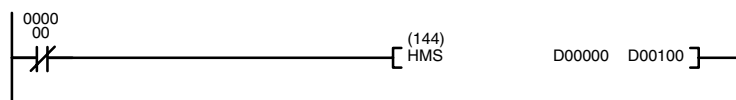
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

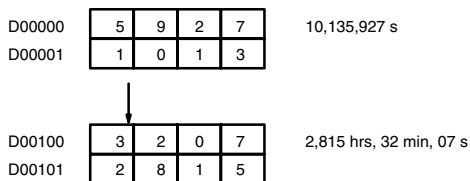
ER (A50003): S and/or S+1 do not contain BCD or exceed 35,999,999 s.  
 Content of \*DM word is not BCD when set for BCD.  
 EQ (A50006): ON when the result is 0.

**Example**

When CIO 000000 is OFF in the following example, the following instruction would convert the seconds given in D00000 and D00001 to hours, minutes, and seconds and store the results in D00100 and D00101 as shown.



Address	Instruction	Operands
00000	LD NOT	000000
00001	HMS(144)	
		D00000
		D00100



**5-26-3 CALENDAR ADD: CADD(145)**

Ladder Symbol	Operand Data Areas
	<b>C: 1<sup>st</sup> calendar word</b> CIO, G, A, T, C, DM <b>T: 1<sup>st</sup> time word</b> CIO, G, A, T, C, #, DM <b>R: 1<sup>st</sup> result word</b> CIO, G, A, DM
<b>Variations</b> ↑ CADD(145)	

**Description**

When the execution condition is OFF, CADD(145) is not executed. When the execution condition is ON, CADD(145) adds the time in words T and T+1 to the calendar data in words C, C+1, and C+2, and outputs the result to words R, R+1, and R+2.

CADD(145) (and the Calendar/Clock Area (G001 to G004)) corrects for leap year.

The following table shows the format of calendar information. The format is the same for the results output to R, R+1, and R+2.

Word	Bits	Contents	Possible values
C	00 to 07	Seconds	00 to 59
	08 to 15	Minutes	00 to 59
C+1	00 to 07	Hours	00 to 23 (24-hour system)
	08 to 15	Day of month	01 to 31 (adjusted by month and for leap year)
C+2	00 to 07	Month	01 to 12
	08 to 15	Year	00 to 99 (Rightmost two digits of year)

The following table shows the format of the time information.

Word	Bits	Contents	Possible values
T	00 to 07	Seconds	00 to 59
	08 to 15	Minutes	00 to 59
T+1	00 to 07	Hours	0000 to 9999

**Precautions**

C, C+1, C+2, T, and T+1 must be BCD and in the proper format.

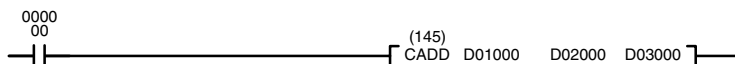
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

- ER (A50003): Time or calendar data is not in the correct format (including impossible dates such as Feb. 30).  
Content of \*DM word is not BCD when set for BCD.
- EQ (A50006): ON when the content of R, R+1, and R+2 is 0 after execution.

**Example**

When CIO 000000 is ON in the following example, the time data in D02000 and D02001 is added to the calendar data in D01000 through D01002 and output as calendar data to D03000 through D03002.



Address	Instruction	Operands
00000	LD	000000
00001	CADD(145)	
		D01000
		D02000
		D03000

C + 2 : D01002	90	07
C + 1 : D01001	28	18
C : D01000	40	30

+

T + 1 : D02001	1532	
T : D02000	27	19

↓

R + 2 : D03002	90	09
R + 1 : D03001	30	15
R : D03000	07	49

**5-26-4 CALENDAR SUBTRACT: CSUB(146)**

Ladder Symbol	Operand Data Areas
	<p><b>C: 1<sup>st</sup> calendar word</b> CIO, G, A, T, C, DM</p> <p><b>T: 1<sup>st</sup> time word</b> CIO, G, A, T, C, #, DM</p> <p><b>R: 1<sup>st</sup> result word</b> CIO, G, A, DM</p>
<p><b>Variations</b></p> <p>↑ CSUB(146)</p>	

**Description**

When the execution condition is OFF, CSUB(146) is not executed. When the execution condition is ON, CSUB(146) subtracts the time in words T and T+1 from the calendar data in words C, C+1, and C+2, and outputs the result to words R, R+1, and R+2.

CSUB(146) (and the Calendar/Clock Area (G001 to G004)) corrects for leap year.

The following table shows the format of calendar information. The format is the same for the results output to R, R+1, and R+2.

Word	Bits	Contents	Possible values
C	00 to 07	Seconds	00 to 59
	08 to 15	Minutes	00 to 59
C+1	00 to 07	Hours	00 to 23 (24-hour system)
	08 to 15	Day of month	01 to 31 (adjusted by month and for leap year)
C+2	00 to 07	Month	01 to 12
	08 to 15	Year	00 to 99 (Rightmost two digits of year)

The following table shows the format of the time information.

Word	Bits	Contents	Possible values
T	00 to 07	Seconds	00 to 59
	08 to 15	Minutes	00 to 59
T+1	00 to 07	Hours	0000 to 9999

**Precautions**

C, C+1, C+2, T, and T+1 must be BCD and in the proper format.

**Note** Refer to page 101 for general precautions on operand data areas.

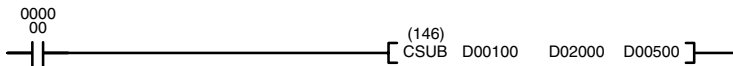
**Flags**

ER (A50003): Time or calendar data is not in the correct format (including impossible dates such as Feb. 30).  
Content of \*DM word is not BCD when set for BCD.

EQ (A50006): ON when the content of R, R+1, and R+2 is 0 after execution.

**Example**

When CIO 000000 is ON in the following example, the time data in D02000 and D02001 is subtracted from the calendar data in D01000 through D01002 and output as calendar data to D00500 through D00502.



Address	Instruction	Operands
00000	LD	000000
00001	CSUB(146)	
		D00100
		D00200
		D00500

C + 2 : D00102  
C + 1 : D00101  
C : D00100

90	04
12	18
40	30

T + 1 : D02001  
T : D02000

1532	
27	19

R + 2 : D00502  
R + 1 : D00501  
R : D00500

90	02
07	22
13	11

**5-26-5 CLOCK COMPENSATION: DATE(179)**

<p><b>Ladder Symbol</b></p> <p><b>Variations</b></p> <p>↑DATE(179)</p>	<p><b>Operand Data Areas</b></p> <p><b>C: Control word</b> CIO, G, A, T, C, DM</p>
--	--

**Description**

When the execution condition OFF, DATE(179) is not executed. When the execution condition is ON, DATE(179) changes the internal clock setting according to the clock data in four consecutive control words (first word: C). The internal clock setting is copied to the clock function area (G001 to G004).



Word	15 to 08	07 to 00
C	Minute (00 to 59)	Second (00 to 59)
C+1	Day (01 to 31)	Hour (00 to 23)
C+2	Year (00 to 99)*	Month (01 to 12)
C+3	---	Day (00 to 06) 00: Sunday 01: Monday 02: Tuesday 03: Wednesday 04: Thursday 05: Friday 06: Saturday

**Note** \*Set the last two digits of the year.

**Precautions**

The lower limit (C) must be less than or equal to the upper limit (C+1).  
An error will not be generated even if the internal clock is set to a non-existent date (such as November 31, for example).

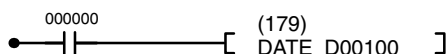
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

ER (A50003): The control word is not within the allowable range.  
The content of a\*DM word is not BCD when set for BCD.

**Example**

When CIO 000000 is ON in the following example, the internal clock setting will be changed according to the content of D00100 through D00103.



Address	Instruction	Operands
00000	LD	000000
00001	DATE(179)	
		D00100

2	6	3	2	D00100
2	5	1	3	D00101
9	4	0	5	D00102
0	0	0	3	D00103

May 25, 1994 (Wednesday)  
1:26:32 PM

## 5-27 Special Instructions

FAL(006) and FALS(007) are used to generate error codes and signals from the program. IORF(184) is used to refresh specified I/O words. IODP(189) is used to output displays on SYSMAC BUS2 Slaves, I/O Control Units, or I/O Interface Units. EMBC(171) is used to change the EM Area bank. SRCH(164) is used to search a specified range of words for a value.

### 5-27-1 FAILURE/SEVERE FAILURE ALARM: FAL(006) and FALS(007)

#### FAILURE ALARM: FAL(006)

<p><b>Ladder Symbol</b></p> <p><b>Variations</b></p> <p>↑ FAL(006)</p>	<p><b>Operand Data Areas</b></p> <p><b>N:</b> FAL number      000 to 511</p> <p><b>M:</b> 1<sup>st</sup> message word CIO, G, A, #, DM</p>
--	--

**SEVERE FAILURE ALARM: FALS(007)**

Ladder Symbol	Operand Data Areas
	<p><b>N:</b> FAL number      001 to 511</p> <p><b>M:</b> 1<sup>st</sup> message word CIO, G, A, #, DM</p>

**Description**

FAL(006) and FALS(007) are provided so that the programmer can output error numbers and messages for use in operation, maintenance, and debugging. When executed with an ON execution condition, the FAL(006) instruction will turn ON the bit in the Auxiliary Area that corresponds to the FAL number. Bits A43001 to A46115 correspond to FAL numbers 001 to 511. FAL number 000 is used to reset FAL errors (see below).

When executed with an ON execution condition, both FAL(006) and FALS(007) cause an error code to be output to A400. The error code identifies the FAL number (FAL(006) and FALS(007) use the same FAL numbers) and whether the error is an FAL or FALS error, as shown in the table. If an error occurs that is more serious than the one recorded in A400 (including errors other than those generated with FAL(006) and FALS(007)), the new error code will replace the previous one. The system also outputs error codes to A400. Refer to *Section 6 Troubleshooting* in the *CVM1D PCs Installation Guide* for details on error code priority.

FAL number	FAL error code	FALS error code
001	4101	C101
002	4102	C102
.	.	.
.	.	.
.	.	.
511	42FF	C2FF

When FAL(006) is executed with an ON execution condition, the FAL Instruction Flag (A40215) will be turned ON, and the ALARM indicator on the front of the CPU will light, but PC operation will continue. When FALS(007) is executed with an ON execution condition, the FALS Instruction Flag (A40106) will be turned ON, the ERROR indicator will light, all outputs will be turned OFF, and PC operation will stop.

Operand M determines whether or not a message will be output to the CVSS when an FAL(006) or FALS(007) instruction is executed. If M is input as a number (#0000 to #FFFF), the message function is disabled. If M is an address, it is the leading address of an 8-word table containing a 16-character ASCII message. Refer to *5-34-3 DISPLAY MESSAGE: MSG(195)*, for information on the format for the message data. If the contents of words M to M+7 are changed after an FAL(006) or FALS(007) instruction is executed, the message will also be changed.

**Resetting Errors**

FAL errors can be cleared by executing the FAL(006) instruction with N equal to 000, or from the CVSS. FALS errors can be cleared from the CVSS or turning the power OFF and ON after first correcting the cause of the error.

When FAL(006) is executed with N=000 and M equal to an FAL number (0001 to 0511), both the error code in A400 and the bit between A43001 and A46115 corresponding to the FAL number in M will be reset.

When FAL(006) is executed with N=000 and M equal to FFFF, all non-fatal errors will be cleared, i.e., all errors that don't stop PC operation will be cleared, and words A400 and A430 to A461 will be reset.

**Precautions**

N must be between 000 and 511 for FAL(006) or between 001 and 511 for FALS(007).

If M designates the first word in a table containing a message, it cannot be one of the last seven words in a data area.

FAL(006) and FALS(007) share FAL numbers. If two instructions use the same FAL number, only the first instruction using the FAL number Will be recognized.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

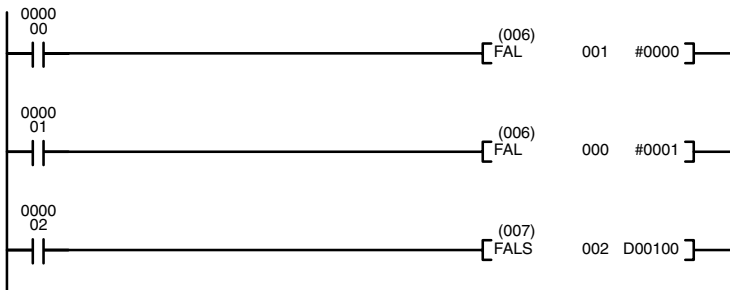
- ER (A50003): N contains improper data.  
Content of \*DM word is not BCD when set for BCD.
- A40215: The FAL Instruction Flag will be turned ON when an FAL(006) instruction is executed.
- A40106: The FALS Instruction Flag will be turned ON when an FALS(007) instruction is executed.
- A43001 to A46115:  
These flags are turned ON to indicate the FAL numbers for which FAL(006) or FALS(007) have been executed.

**Example**

When CIO 000000 is ON in the following example, a non-fatal error is generated as FAL 001, the ALARM indicator on the CPU lights, A43001 turns ON, and 4101 is output to A400. Program execution will continue.

When CIO 000001 turns ON, the FAL 001 error is cleared, A43001 turns OFF, the error code in A400 is cleared (if it is 4101), and the ALARM indicator turns off (assuming no other errors have occurred).

When CIO 000002 turns ON, a fatal error is generated as FAL 002, the ERROR indicator on the CPU lights, all outputs are turned OFF, and C102 is output to A400. Program execution will stop. If a Programming Device is connected and online, the message stored in D00100 through D00107 will also be displayed as an error message.



Address	Instruction	Operands
00000	LD	000000
00001	FAL(006)	001
		#0000
00002	LD	000001
00003	FAL(006)	000
		#0001
00004	LD	000002
00005	FALS(007)	002
		D00100

**5-27-2 FAILURE POINT DETECTION: FPD(177)**

Ladder Symbol	Operand Data Areas
	<b>C: Control data</b> # <b>T: Monitoring time</b> CIO, G, A, #, DM <b>D: First register word</b> CIO, G, A, DM

**Description**

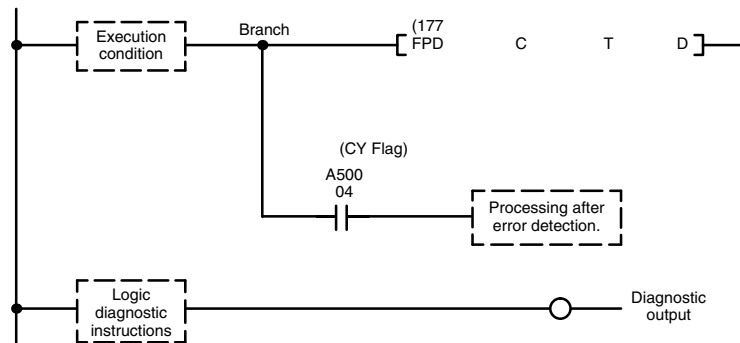
FPD(177) is used to monitor the execution of an instruction block according to specified conditions, detect errors, and determine the input conditions responsible for the errors. FPD(177) is used either to monitor the time between the execution of FPD(177) and the execution of a diagnostic output or to determine

the input in the instruction block that is preventing an output from being turned ON.

FPD(177) can be used in the program as many times as desired, but each must use a different D even if the same message is being output.

The following diagram illustrates the type of program section that can be diagnosed with FPD(177). The instruction block that is diagnosed by FPD(177) starts at the first LD after FPD(177) (excluding LD for TR bits) and ends at the next output or special (right-hand) instruction (except for OUT for TR bits).

The program sections marked by dashed lines in the following diagram can be written according to the needs of the particular program application. The processing programming section triggered by CY is optional and can use any instructions but LD and LD NOT. The logic diagnostic instructions and execution condition can consist of any combination of NC or NO conditions desired.

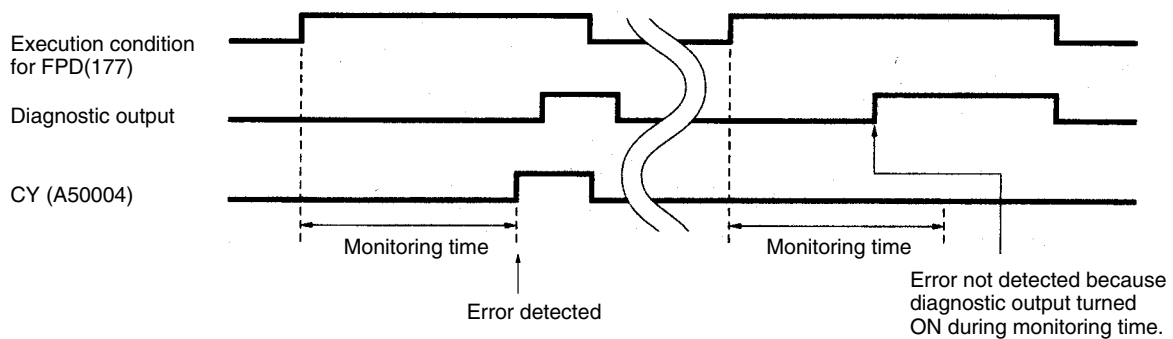


**Time Monitoring**

When the execution condition is OFF, FPD(177) is not executed. When the execution condition is ON, FPD(177) monitors the time until the diagnostic output is executed with an ON execution condition. If this time exceeds T, the following will occur:

- 1, 2, 3... 1. An FAL(06) error is generated with the FAL number specified in the first two digits of C. If 00 is specified, however, an error will not be generated.
2. The CY Flag (A 50004) is turned ON. An error processing program section can be executed using the CY Flag if desired.
3. If bit 15 of C is ON, a preset message with up to 8 ASCII characters will be displayed on the Programming Device along with the bit address mentioned in step 2.

The following chart shows the timing that occurs when the execution condition for FPC(177) turns ON.

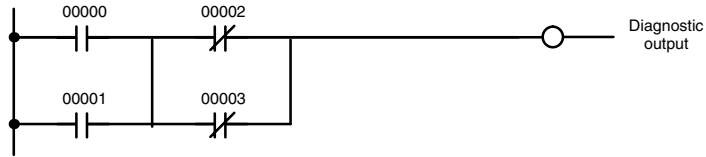


**Logic Diagnosis**

FPD(177) also searches for the input condition that is responsible for the diagnostic output not turning ON and outputs diagnostic results and, if specified, a message. The following instructions are examined: LD (excluding LD for TR bits), LD NOT, AND, AND NOT, OR, and OR NOT. Operands addressed indi-

rectly through index registers, however, are not examined. If more than one input condition is OFF, the input condition on the highest instruction line and nearest the left bus bar is selected.

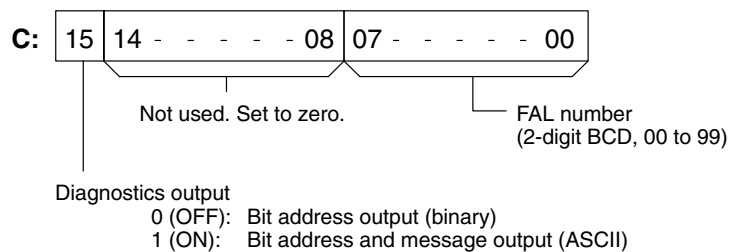
When IR 00000 to IR 00003 are ON in the following example, the normally closed condition IR 00002 would be found as the cause of the diagnostic output not turning ON.



The logic diagnosis operation runs independently from the time monitoring operation. The bit address information bit (bit 15 of D) can be examined to see if information has been output for logic diagnosis.

**Control Data**

The function of the control data bits in C are shown in the following diagram.

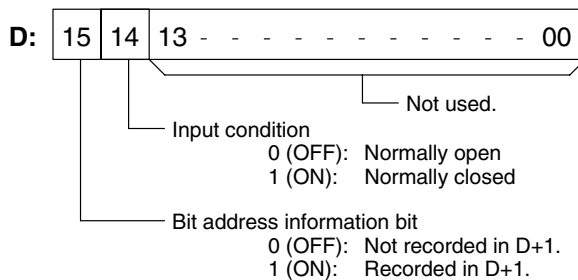


**Diagnostics Output**

There are two ways to output the bit address of the OFF condition detected in the logic diagnosis operation.

- 1, 2, 3...** 1. Bit address output (used when bit 15 of C is OFF).

Bit 15 of D indicates whether or not bit address information is stored in D+1. If there is, bit 14 of D indicates whether the input condition is normally open or closed.



D+1 contains the memory address of the operand of the input condition. The absolute memory address is given in the same form used for indirect addressing. Refer to *Section 3 Memory Areas* and to the discussion of indirect addressing on page 62 for details on absolute memory addresses.

2. Bit address and message output (used when bit 15 of C is ON).

Bit 15 of D indicates whether or not there is bit address information stored in D+1 to D+3. If there is, bit 14 of D indicates whether the input condition is normally open or closed. Refer to the following table.

Words D+1 to D+8 contain information in ASCII displayed on a Programming Device along with the bit address when FPD(177) is executed. Words D+5 to D+8 contain the message preset by the user as shown in the following table.

Word	Bits 15 to 08	Bits 07 to 00
D+1	20 = space	First ASCII character of bit address
D+2	Second ASCII character of bit address	Third ASCII character of bit address
D+3	Fourth ASCII character of bit address	Fifth ASCII character of bit address
D+4	2D = “-”	“30”=normally open, “31”=normally closed
D+5	First ASCII character of message	Second ASCII character of message
D+6	Third ASCII character of message	Fourth ASCII character of message
D+7	Fifth ASCII character of message	Sixth ASCII character of message
D+8	Seventh ASCII character of message	Eighth ASCII character of message

**Note** If 8 characters are not needed in the message, input “0D” after the last character.

**Setting the Monitoring Time** The procedure below can be used to automatically set the monitoring time, T, under actual operating conditions when specifying a word operand for T. This operation cannot be used if a constant is set for T.

- 1, 2, 3...** 1. Connect a Programming Device, such as a Programming Console.
2. Use the Programming Device to turn ON control bit A09800.
3. Execute the program with A09800 turned ON. If the monitoring time currently in T is exceeded, 1.5 times the actual monitoring time will be stored in T.
4. Turn OFF A09800 when an acceptable value has been stored in T.

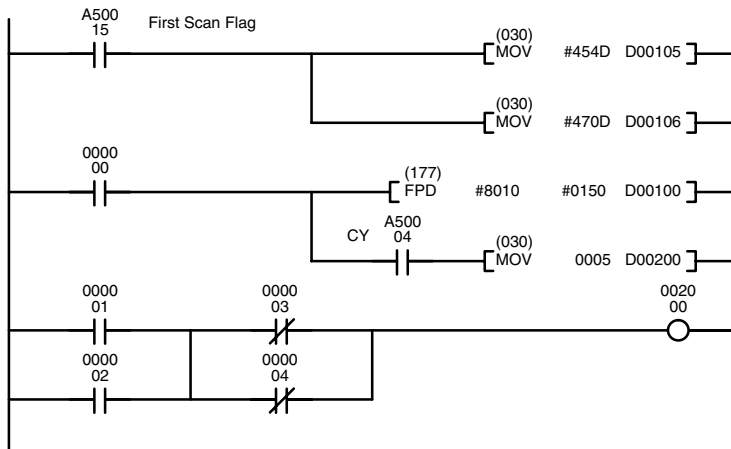
### Example

In the following example, the FPD(177) is set to display the bit address and message (“EMG”) when a monitoring time of 15 s is exceeded.

The first two instruction lines store the message “EMG” in D00105 and D00106. The control data for FPD(177) specifies a FAL number of 010 and message output.

An error will be detected if CIO 002000 does not turn ON within 15 s after CIO 000000 turns ON. When the error is detected, an FAL error number 010 will be generated, the ASCII of the address of the bit responsible for CIO 0020000 not turning ON would be output to D00101 through D00103, and the bit address and message “EMG” would be displayed on a Programming Device. CY would also turn ON, causing the contents of CIO 0005 to be moved to D00200.

In the following example, it is assumed that CIO 000001 through CIO 000004 are all ON, thus CIO 000003 is output as the address of the bit responsible for CIO 002000 not turning ON.



Address	Instruction	Operands
00000	LD	A50015
00001	MOV(030)	
		#454D
		D00105
00002	MOV(030)	
		#047D
		D00106
00003	LD	000000
00004	FPD(177)	
		#8010
		#0150
		D00100
00005	AND	A50004
00006	MOV(030)	
		0005
		D00200
00007	LD	000001
00008	OR	000002
00009	LD NOT	000003
00010	OR NOT	000004
00011	AND LD	
00012	OUT	002000

The contents of D00100 through D00108 would be as follows for the conditions described above. This data would be displayed on the Programming Device as “000003 – 1EMG.

Bit	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	Meaning
D00100	1	1	(Not used.)													Bit information present, NC	
D00101	3		0				3			0			“00”			Bit address (displayed)	
D00102	3		0				3			0			“00”				
D00103	3		0				3			3			“03”				
D00104	2		D				0			1			“-1”			Message (displayed)	
D00105	4		5				4			D			“EM”				
D00106	4		7				0			D			“G” Return				
D00107	0		0				0			0			Spaces			Message (ignored)	
D00108	0		0				0			0			Spaces				

Flags

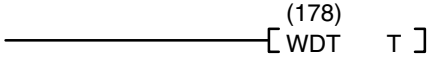
ER (A50003): T is not BCD.

Content of \*DM word is not BCD when set for BCD.

Control word data is incorrect.

CY (A50004): Time between the execution of FPD(177) and the execution of a diagnostic output exceeds T.

### 5-27-3 MAXIMUM CYCLE TIME EXTEND: WDT(178)

<p><b>Ladder Symbol</b></p>  <p><b>Variations</b></p> <p>↑ WDT(178)</p>	<p><b>Operand Data Area</b></p> <p>T: Timer value    # (0000 to 3999)</p>
--	---

Generally, the maximum cycle time is designated in the PC Setup, and if the cycle time exceeds the designated value, a fatal error (Cycle Time Too Long) will occur. WDT(178) allows you to extend the maximum cycle time during program execution without changing the designate value in the PC Setup. WDT(178) is useful when executing a process that requires a long cycle time to avoid causing a fatal error.

**Description**

When the execution condition is OFF, WDT(178) is not executed. When the execution condition is ON, WDT(178) extends the maximum cycle time by 10 ms times T. The value is set by default to 1,000 ms unless changed in PC Setup.

$$\text{Time extension} = 10 \text{ ms} \times T.$$

Specify T in BCD between 0000 and 3999 (i.e., 0 to 39,990 ms).

WDT(178) can be programmed and executed as many times as desired and each will extend the maximum cycle time by the specified amount until the value is set to 40,000 ms. If WDT(178) is executed after the value has reached 40,000 ms, it will remain set to 40,000 ms.

**Flags**

There are no flags affected by this instruction.

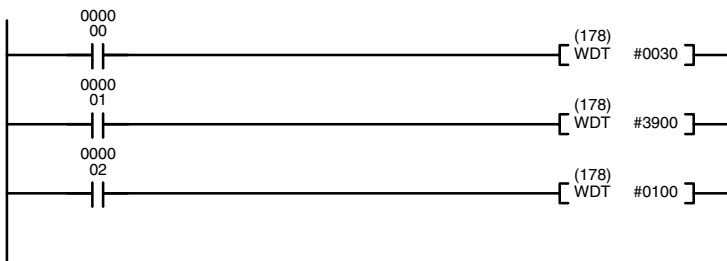
**Example**

This example assumes that the maximum cycle time is set to 1,000 ms when execution of the following instructions is started.

When CIO 000000 turns ON, the maximum cycle time will be extended by 300 ms to 1,300 ms.

When CIO 000001 turns ON, an attempt will be made to extend the maximum cycle time by 39,000 ms, but doing so will exceed 40,000 ms, so the timer will be set to 40,000 ms.

When CIO 000002 turns ON, the maximum cycle time will not be extended because it is already at 40,000 ms, where it will stay.



Address	Instruction	Operands
00000	LD	000000
00001	WDT(178)	
		#0030
00002	LD	000001
00003	WDT(178)	
		#3900
00004	LD	000002
00005	WDT(178)	
		#0100



### 5-27-4 I/O REFRESH: IORF(184)

<p><b>Ladder Symbol</b></p> <p><b>Variations</b> ↑ IORF(184)</p>	<p><b>Operand Data Areas</b></p> <p><b>St: Starting word</b> CIO</p> <p><b>E: Ending word</b> CIO</p>
--	---

**Description** When the execution condition is OFF, IORF(184) is not executed. When the execution condition is ON, all words between St and E will be refreshed. This will be in addition to the normal I/O refresh performed during the CPU's scan.

**Precautions** IORF(184) can be used to refresh I/O words on the CPU, Expansion CPU, or Expansion I/O Racks only. It cannot be used for other I/O words in SYSMAC BUS or SYSMAC BUS/2 Remote I/O Systems.

St must be less than or equal to E. If St is greater than E, the instruction will be treated as NOP(000).

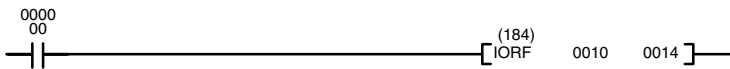
St and E must be in the I/O Area, CIO 0000 to CIO 0511.

The up-differentiation (↑) execution condition might not be recognized when the PC has been switched from duplex mode to simplex mode.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags** There are no flags affected by this instruction.

**Example** When CIO 000000 is ON in the following example, the status of all inputs allocated to bits in words from CIO 0010 through CIO 0014 will be read into memory, refreshing the status of these input bits.



Address	Instruction	Operands
00000	LD	000000
00001	IORF(184)	
		0010
		0014

### 5-27-5 I/O DISPLAY: IODP(189)

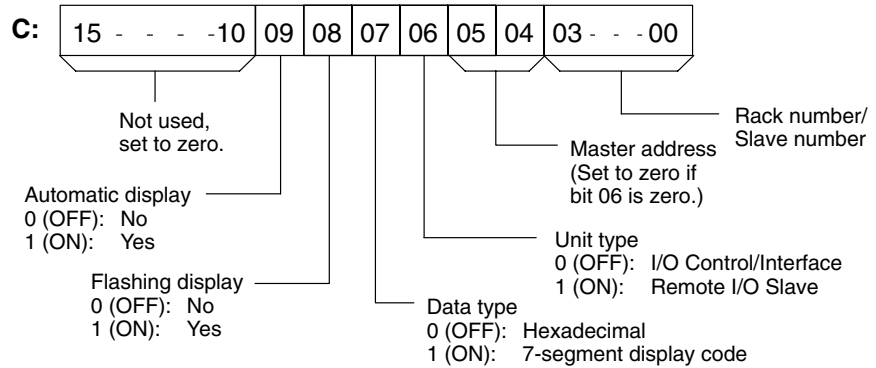
<p><b>Ladder Symbol</b></p> <p><b>Variations</b> ↑ IODP(189)</p>	<p><b>Operand Data Areas</b></p> <p><b>C: Control word</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>S: Source word(s)</b> CIO, G, A, T, C, DM</p>
--	---

**Description** When the execution condition is OFF, IODP(189) is not executed. When the execution condition is ON, IODP(189) outputs four characters to the 7-segment display on a SYSMAC BUS/2 Remote I/O Slave Unit, I/O Control Unit, or I/O Interface Unit. The four characters can be in 7-segment display code in source words S and S+1, or the content of a single hexadecimal source word (S).

**Control Word** The control word contains information defining the Unit to which the characters will be output, whether the source data is hexadecimal or 7-segment display, whether the characters are to flash or not, and whether the display can be controlled from the Unit itself.

Bits 00 to 06 specify the Unit to which the characters will be output (specifics are shown in the following diagram). Bit 07 determines whether the source data is hexadecimal (OFF) or 7-segment display code (ON). Bit 08 determines whether the characters will flash (ON) or not (OFF).

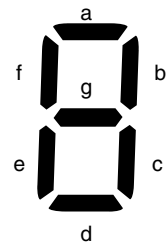
If bit 09 of C is set for automatic display (ON), the characters will be displayed regardless of the Unit's display mode, but if bit 09 of C is OFF, the characters will be displayed only when the Unit is set to display mode 3.



If bit 07 is set for hexadecimal data, the characters are output as they are in the source word. The rightmost digit will be the rightmost on the display.

If bit 07 is set for 7-segment display code, segments a to f of the leftmost digit are contained in S bits 00 to 06, segments a to f of the second digit are contained in S bits 08 to 14. Set bits 07 and 15 to zero. The segments for the third and fourth digits follow the same pattern in word S+1, as shown in the following diagram. The table shows the number of the bit that must be turned ON in S or S+1 to turn ON each segment of each digit in the display.

Segment		g	f	e	d	c	b	a
S	Digit 1	06	05	04	03	02	01	00
	Digit 2	14	13	12	11	10	09	08
S+1	Digit 3	06	05	04	03	02	01	00
	Digit 4	14	13	12	11	10	09	08



**Precautions**

S cannot be the last word in a data area when S and S+1 contain 7-segment display code.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

ER (A50003): Content of \*DM word is not BCD when set for BCD.  
Control word data is incorrect.

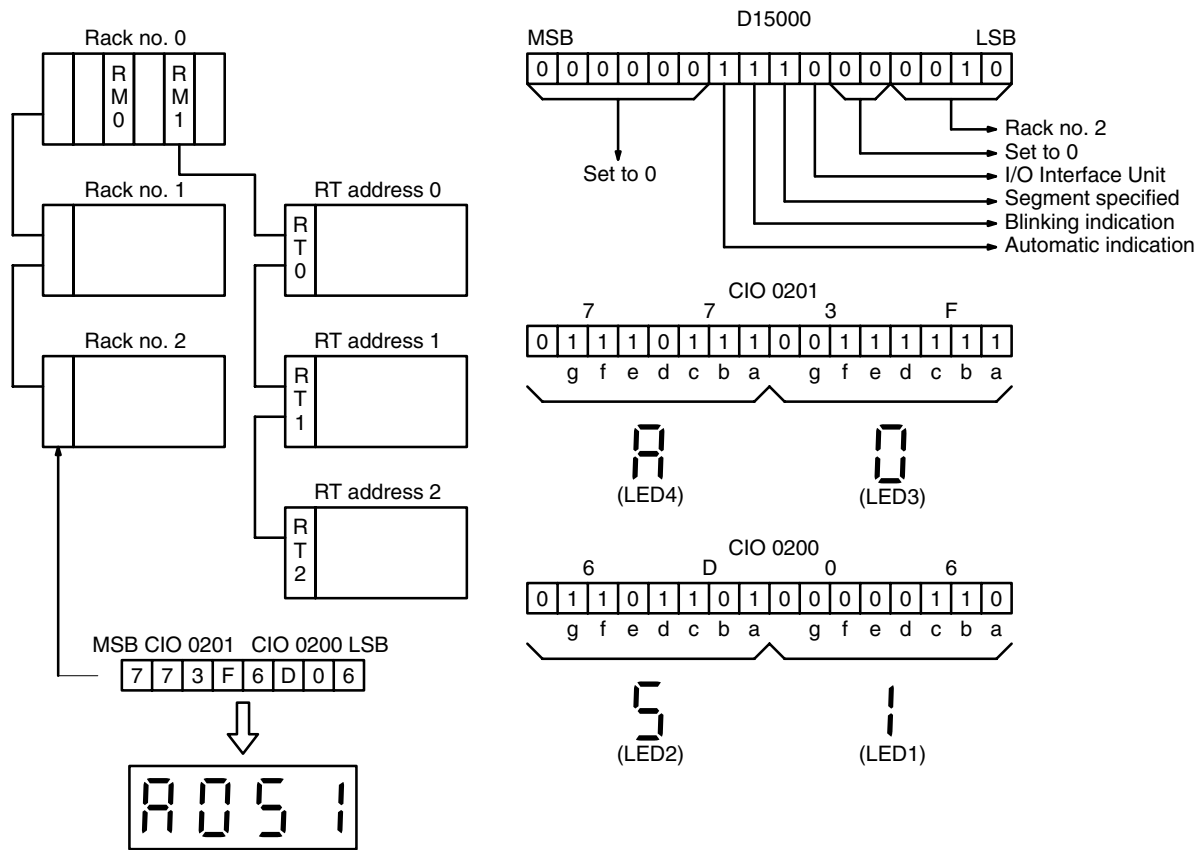
**Example**

The following example show how to produce the display "RDS P" from both a 7-segment display code.

When CIO 000000 is ON, the content of CIO 0200 and CIO 0201 is displayed according to the specifications in D15000.



Address	Instruction	Operands
00000	LD	000000
00001	IODP(189)	
		D15000
		0200



### 5-27-6 SELECT EM BANK: EMBC(171)

Ladder Symbol	Operand Data Area
<p>(171) [ EMBC N ]</p>	<p><b>N:</b> EM bank number CIO, G, A, #, DM, DR, IR</p>
<p><b>Variations</b> ↑ EMBC(171)</p>	

**Description** When the execution condition is OFF, EMBC(171) is not executed. When the execution condition is ON, EMBC(171) changes the current EM bank to the one indicated by the EM bank number (N).

The current EM bank number is recorded in the least significant (rightmost) digit of A511. Bit A51115 is ON when EM is mounted to the CPU.

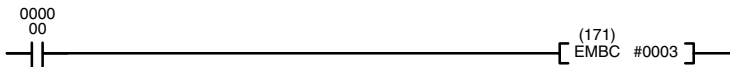
**Precautions** N must be between 0000 and 0007.  
EMBC(171) can be used only with CPUs that support the EM Area.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags** ER (A50003): N is not between 0000 and 0007.  
The EM Unit does not have the EM bank indicated by N, or the indicated bank is being used as file memory.  
Content of \*DM word is not BCD when set for BCD.

**Example**

When CIO 000000 is ON in the following example, the current EM bank is changed to bank 3. The contents of A511 would change to “8003” to indicate that bank 3 is the current bank.



Address	Instruction	Operands
00000	LD	000000
00001	EMBC(171)	
		#0003

**5-27-7 DATA SEARCH: SRCH(164)**

Ladder Symbol	Operand Data Areas
$\text{---} \left[ \overset{(164)}{\text{SRCH}} \quad \text{N} \quad \text{R}_1 \quad \text{Cd} \right] \text{---}$	<b>N: Number of words</b> CIO, G, A, T, C, #, DM, DR, IR <b>R<sub>1</sub>: 1<sup>st</sup> word in range</b> CIO, G, A, T, C, DM <b>Cd: Comparison data</b> CIO, G, A, T, C, #, DM, DR, IR
<b>Variations</b> ↑ SRCH(164)	

**Description**

When the execution condition is OFF, SRCH(164) is not executed. When the execution condition is ON, SRCH(164) searches the range of memory from R<sub>1</sub> to R<sub>1</sub>+N-1 for addresses that contain the comparison data (Cd).

If the content of an address within the range match the comparison data, the EQ Flag (A50006) is turned ON and the address is written to index register IR0. If more than one address contains the comparison data, the EQ Flag (A50006) is turned ON and only the lowest address containing the comparison data is written to IR0.

If none of the addresses within the range contain the comparison data, the EQ Flag (A50006) is turned OFF, and IR0 is left unchanged.

**Precautions**

N must be BCD between 0001 and 9999.

R<sub>1</sub> and R<sub>1</sub>+N-1 must be in the same data area.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

ER (A50003): N is 0000 or is not BCD.

Content of \*DM word is not BCD when set for BCD.

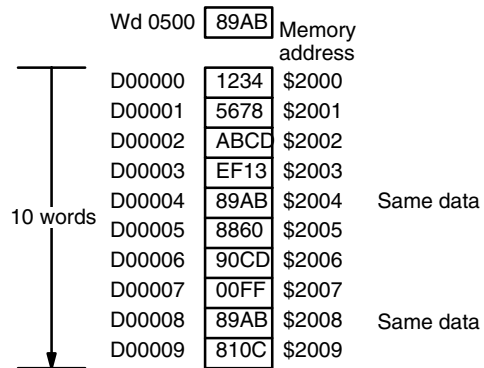
EQ (A50006): An address within the range contains the comparison data.

**Example**

In the following example, the 10-word range from D00000 to D00009 is searched for addresses that contain the same data as CIO 0500. Since two addresses within the range contain the same data as CIO 0500, the EQ Flag (A50006) is turned ON, and the lowest address containing the comparison data is written to IR0.



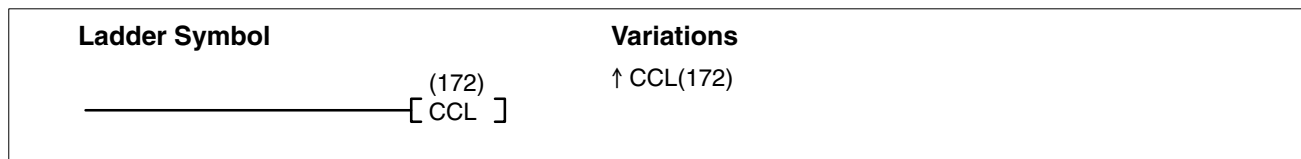
Address	Instruction	Operands
00000	LD	000000
00001	SRCH(164)	
		#0010
		D00000
		0500



## 5-28 Flag/Register Instructions

The Flag/Register Instruction and used to save or reload the contents of the Arithmetic Flags or the index/data registers.

### 5-28-1 LOAD FLAGS: CCL(172)



#### Description

When the execution condition is OFF, CCL(172) is not executed. When the execution condition is ON, CCL(172) changes the Arithmetic Flags to the status recorded by the last CCS(173) instruction.

The following table shows the Arithmetic Flags affected by CCL(172).

Bit	Arithmetic Flag
A50003	Instruction Execution Error Flag
A50004	Carry Flag
A50005	Greater Than Flag
A50006	Equals Flag
A50007	Less Than Flag
A50008	Negative Flag
A50009	Overflow Flag
A50010	Underflow Flag

#### Precautions

CCL(172) should not be executed unless CCS(173) has been executed earlier.

#### Flags

Arithmetic Flags are changed to the status recorded by the last CCS(173) instruction.

#### Example

When CIO 000001 is ON in the following example, the status of the arithmetic flags are all restored to the status they had the last time CCS(173) was executed.



Address	Instruction	Operands
00000	LD	000001
00001	CCL(172)	

### 5-28-2 SAVE FLAGS: CCS(173)

<p><b>Ladder Symbol</b></p>	<p><b>Variations</b></p> <p>↑ CCS(173)</p>
-----------------------------	--

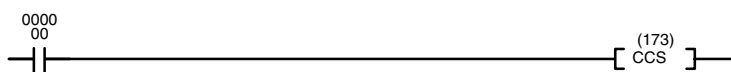
**Description** When the execution condition is OFF, CCS(173) is not executed. When the execution condition is ON, CCS(173) records the current status of the Arithmetic Flags in the CPU for later retrieval by the CCL(172) instruction.

Refer to the table in 5-28-1 LOAD FLAGS: CCL(172) for the Arithmetic Flags recorded by CCS(173).

**Precautions** When CCS(173) is executed, it records over the data recorded by the previous CCS(173) instruction.

**Flags** No flags are affected by CCS(173).

**Example** When CIO 000000 is ON in the following example, the status of all the arithmetic flags is stored in memory for possible later retrieval.



Address	Instruction	Operands
00000	LD	000000
00001	CCS(173)	

### 5-28-3 LOAD REGISTER: REGL(175)

<p><b>Ladder Symbol</b></p>	<p><b>Operand Data Area</b></p> <p>S: 1<sup>st</sup> source word CIO, G, A, DM</p>
<p><b>Variations</b></p> <p>↑ REGL(175)</p>	

**Description** When the execution condition is OFF, REGL(175) is not executed. When the execution condition is ON, REGL(175) copies the data from S, S+1, and S+2 to data registers DR0, DR1, and DR2, and copies the data from S+3, S+4, and S+5 to index registers IR0, IR1, and IR2.

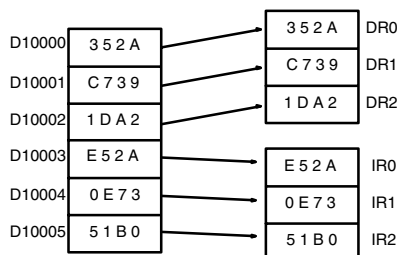
**Precautions** Refer to page 101 for general precautions on operand data areas.

**Flags** ER (A50003): Content of \*DM word is not BCD when set for BCD.

**Example** When CIO 000000 is ON in the following example, the contents of words D10000 through D10005 is copied to the data and index registers.



Address	Instruction	Operands
00000	LD	000000
00001	REGL(175)	
		D10000



### 5-28-4 SAVE REGISTER: REGS(176)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;"> </p> <p><b>Variations</b></p> <p>↑ REGS(176)</p>	<p><b>Operand Data Area</b></p> <p><b>D: 1<sup>st</sup> destination word</b> CIO, G, A, DM</p>
--	--

**Description** When the execution condition is OFF, REGS(176) is not executed. When the execution condition is ON, REGS(176) copies the data from data registers DR0, DR1, and DR2 to D, D+1, and D+2, and copies the data from index registers IR0, IR1, and IR2 to D+3, D+4, and D+5.

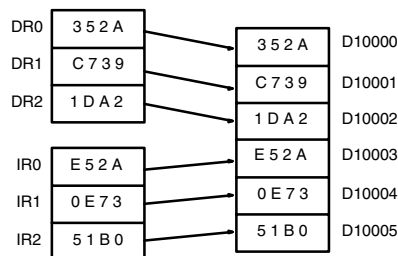
**Precautions** Refer to page 101 for general precautions on operand data areas.

**Flags** ER (A50003): Content of \*DM word is not BCD when set for BCD.

**Example** When CIO 000000 is ON in the following example, the contents of words the data and index registers is copied to D10000 through D10005.



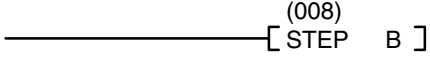
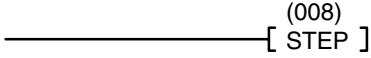
Address	Instruction	Operands
00000	LD	000000
00001	REGS(176)	
		D10000



### 5-29 STEP DEFINE and STEP START: STEP(008)/SNXT(009)

The step instructions STEP(008) and SNXT(009) are used in conjunction to set up break points between sections in a large program so that the sections can be executed as units and reset upon completion. A section of program will usually be defined to correspond to an actual process in the application. (Refer to the application examples later in this section.) A step is written like a normal programming code, except that certain instructions (END(001), IL(002)/ILC(003), JMP(004)/JME(005), and SBN(150)) may not be included.

## STEP DEFINE: STEP(008)

Ladder Symbol	Operand Data Area
 	<b>B: Bit</b> CIO, G, A

## STEP START: SNXT(009)

Ladder Symbol	Operand Data Area
	<b>B: Bit</b> CIO, G, A

**Description**

STEP(008) uses a control bit to define the beginning of a section of the program called a step. STEP(008) does not require an execution condition, i.e., its execution is controlled through the control bit.

To start step execution, SNXT(009) is used with the same control bit as used for the first STEP(008). If SNXT(009) is executed with an ON execution condition, the step with the same control bit is executed. If the execution condition is OFF, the step is not executed. You must use a differentiated execution condition for the SNXT(009) instruction that starts step execution or step execution will last for only one cycle.

The SNXT(009) instruction must be written into the program before the program reaches the step it starts. It can be used at different locations before the step to control the step according to two different execution conditions (see example 2, below). Any step in the program that has not been started with SNXT(009) will not be executed.

Once SNXT(009) is used to start step execution, step execution will continue until STEP(008) is executed without a control bit. STEP(008) executed without a control bit is used to stop step execution and return to normal execution. STEP(008) without a control bit must be preceded by SNXT(009) with a dummy control bit. It cannot be a control bit used in a STEP(008).

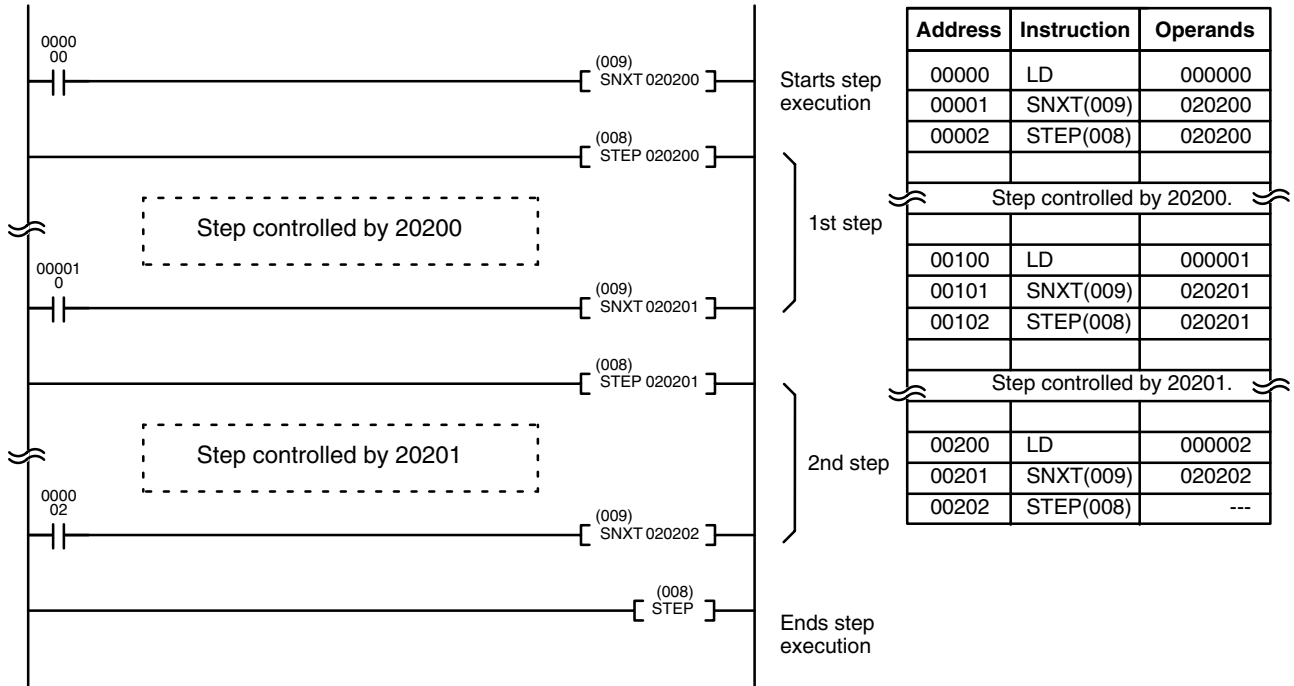
Execution of a step is completed either by execution of the next SNXT(009) or by turning OFF the control bit for the step (see example 3). When the step is completed, all I/O Area output bits, Work Area, Holding Area, and CPU Bus Link Area bits in the step are turned OFF, Timers (except TTIM(120), TIML(121), and MTIM(122)) in the step are reset to their SVs. Counters, shift registers, and bits used in KEEP(011) maintain status.

Steps can be programmed consecutively. Each step must start with STEP(008) and generally ends with SNXT(009) (see example 3, for an exception). All control bits must be in the same word and they must be consecutive.

When steps are programmed in series, three types of execution are possible: sequential, branching, or parallel. The execution conditions for, and the positioning of, SNXT(009) determine how the steps are executed. The three examples given later demonstrate these three types of step execution.



Two simple steps are shown below. In this example, the 1st step would be executed from the time that CIO 00000 goes ON until CIO 000001 goes ON. The 2nd step would be executed for the time the CIO 000001 goes ON until CIO 000002 goes ON. When CIO 000002 goes ON, step execution will be terminated.



**Precautions**

Control bits within one section of step programming must be sequential and from the same word.

STEP(008) and SNXT(009) cannot be used inside of subroutines or block programs.

Only one step programming area can be executed during any one cycle.

Interlocks, jumps, SBN(150), and END(001) cannot be used within step programs.

You must use a differentiated execution condition for the SNXT(009) instruction that starts step execution.

Bits used as control bits must not be used anywhere else in the program unless they are being used to control the operation of the step (see example 3, below). All control bits must be in the same word and must be consecutive.

Control bit status will be lost during any power interruption if it is not in the Holding Area. If it is necessary to maintain status to resume execution at the same step, Holding Area bits must be used.

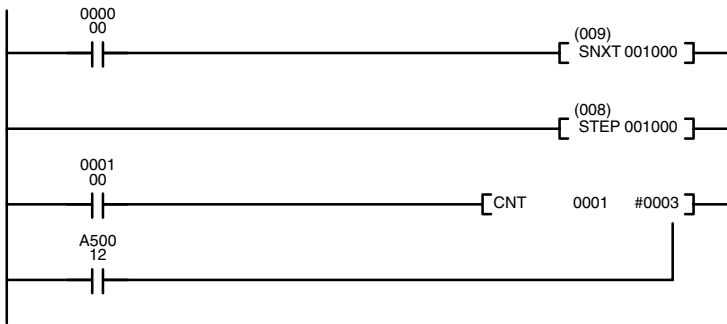
Only one section of step programming can be started during a cycle. Be sure that two steps aren't started during a cycle.

**Note** Refer to page 101 for general precautions on operand data areas.

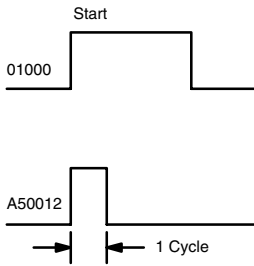
**Flags**

A50012:

The Step Flag is turned ON for one cycle when STEP(008) is executed and if necessary it can be used to reset counters in steps as shown below.



Address	Instruction	Operands
00000	LD	000000
00001	SNXT(009)	001000
00002	STEP(008)	001000
00003	LD	000100
00004	LD	A50012
00005	CNT	001
		#0003

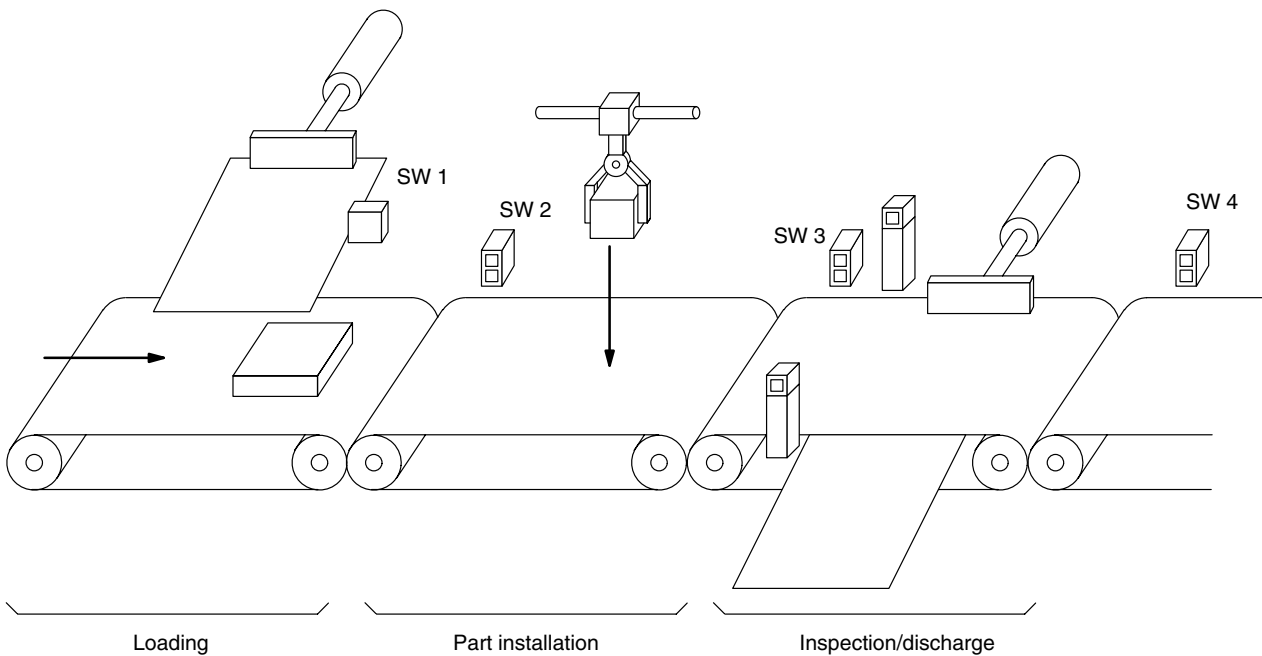


**Examples**

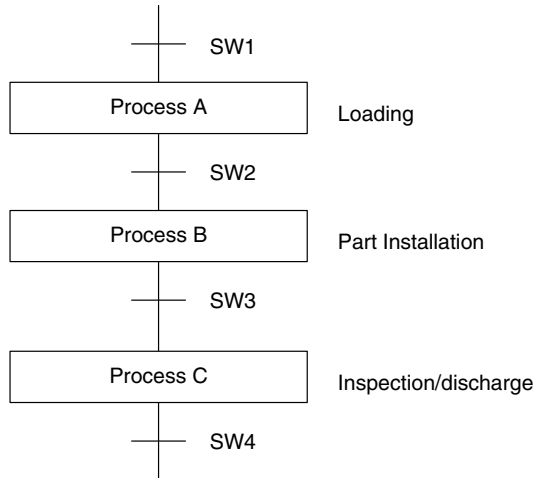
The following three examples demonstrate the three types of execution control possible with step programming. *Example 1* demonstrates sequential execution; *Example 2*, branching execution; and *Example 3*, parallel execution.

**Example 1:  
Sequential Execution**

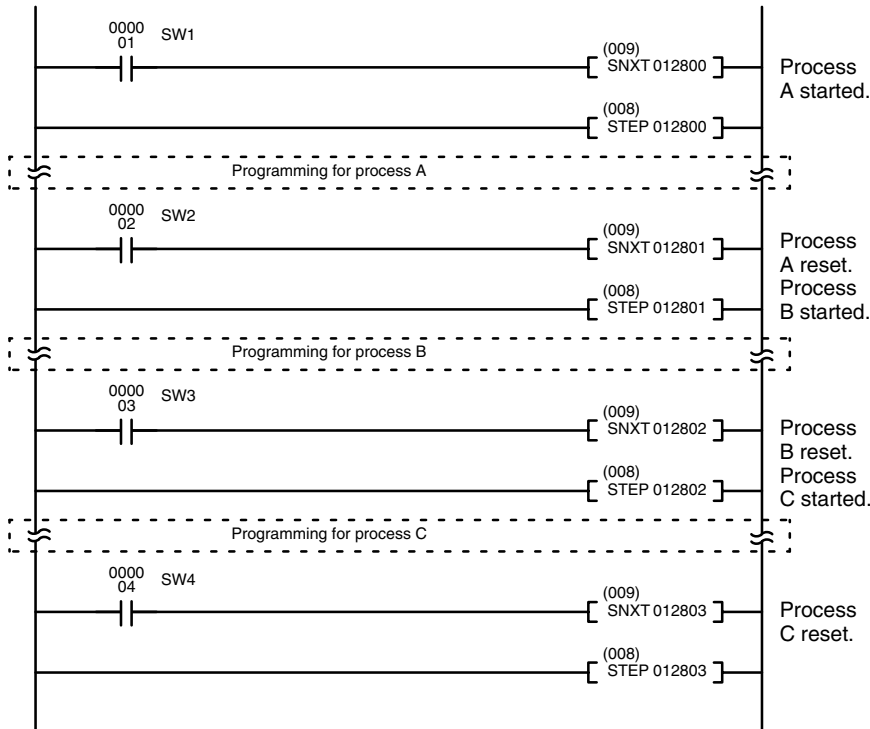
The following process requires that three processes, loading, part installation, and inspection/discharge, be executed in sequence with each process being reset before continuing on the the next process. Various sensors (SW1, SW2, SW3, and SW4) are positioned to signal when processes are to start and end.



The following diagram demonstrates the flow of processing and the switches that are used for execution control.



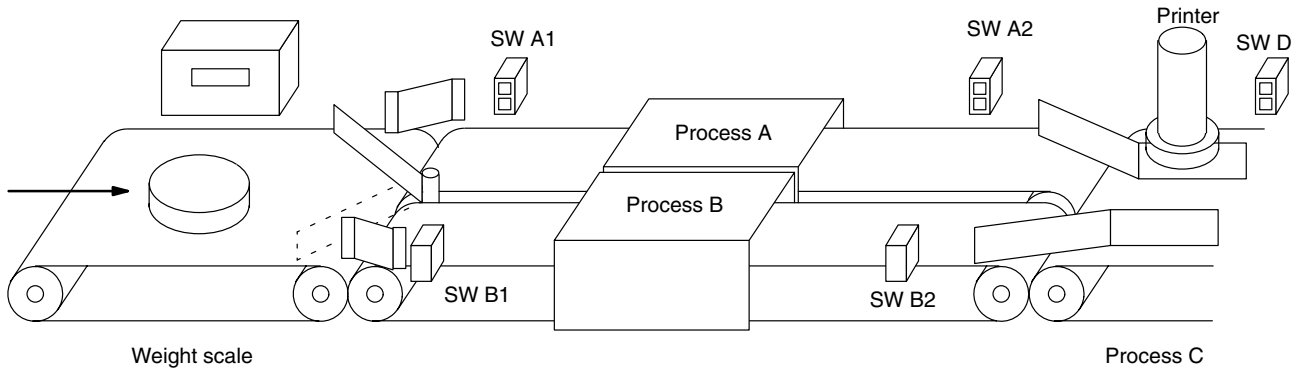
The program for this process, shown below, utilizes the most basic type of step programming: each step is completed by a unique SNXT(009) that starts the next step. Each step starts when the switch that indicates the previous step has been completed turns ON.



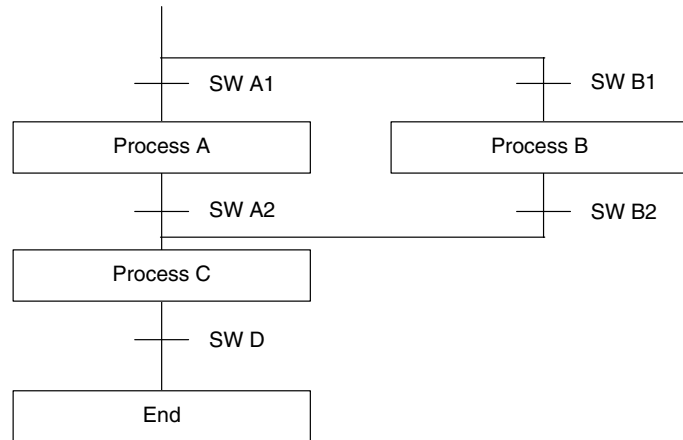
Address	Instruction	Operands
00000	LD	000001
00001	SNXT(009)	012800
00002	STEP(008)	012800
Process A		
00100	LD	000002
00101	SNXT(009)	012801
00102	STEP(008)	012801
Process B		
00100	LD	000003
00101	SNXT(009)	012802
00102	STEP(008)	012802
Process C		
00200	LD	000004
00201	SNXT(009)	012803
00202	STEP(008)	012803

**Example 2:  
Branching Execution**

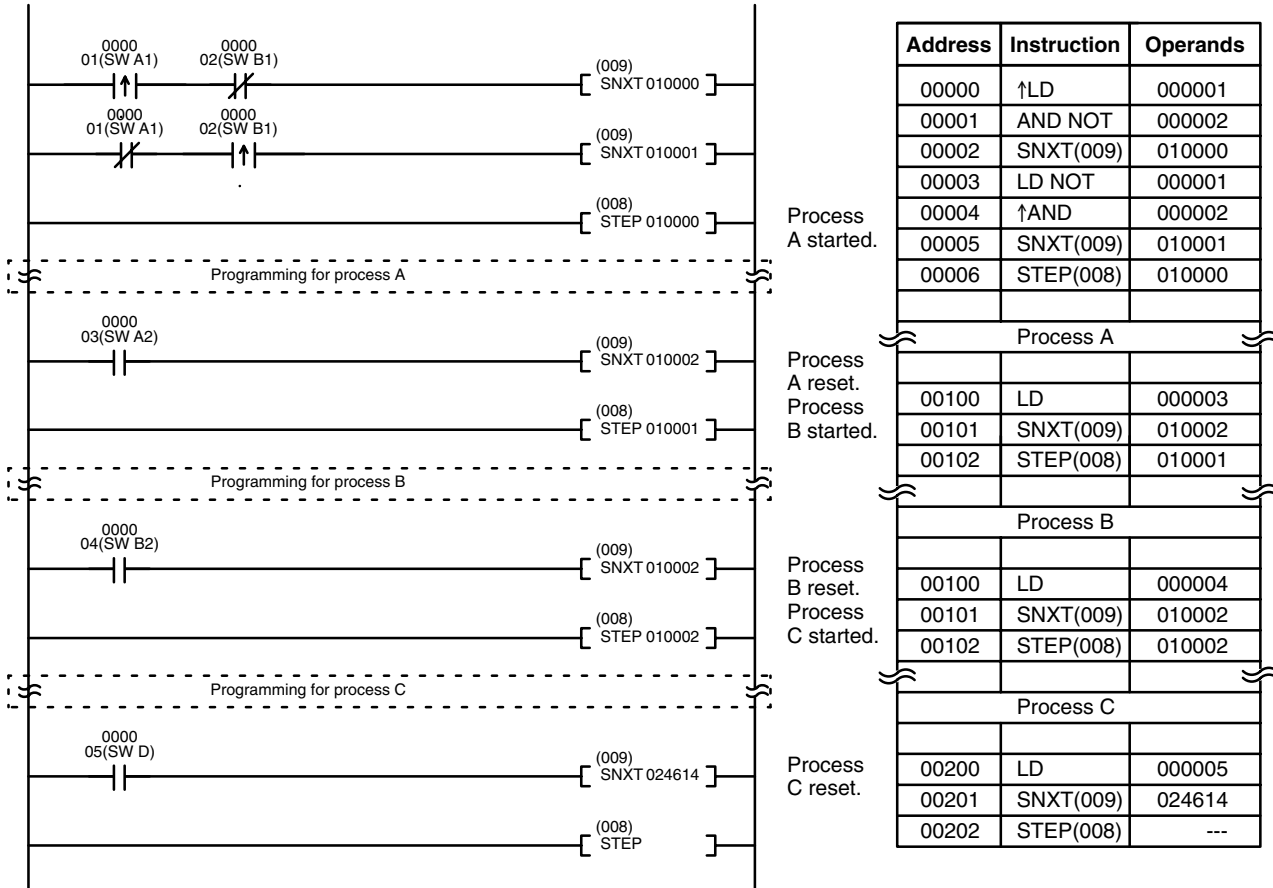
The following process requires that a product is processed in one of two ways, depending on its weight, before it is printed. The printing process is the same regardless of which of the first processes is used. Various sensors are positioned to signal when processes are to start and end.



The following diagram demonstrates the flow of processing and the switches that are used for execution control. Here, either process A or process B is used depending on the status of SW A1 and SW B1.



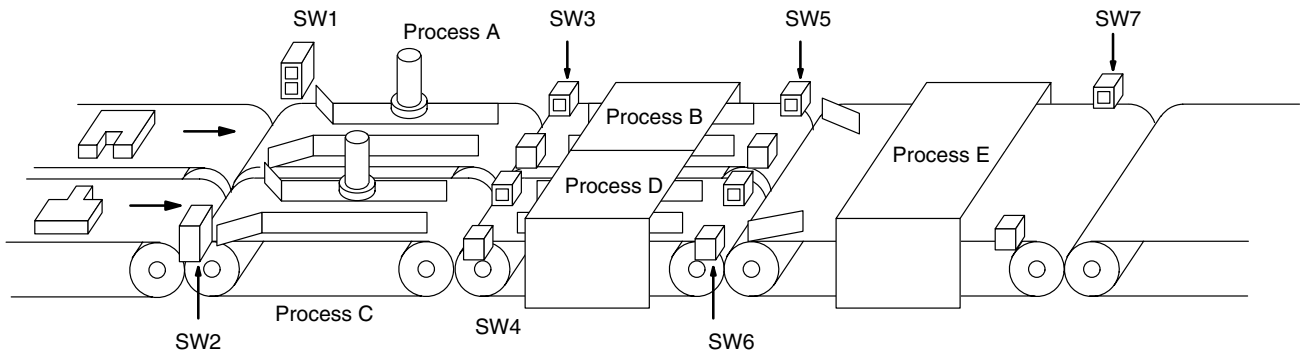
The program for this process, shown below, starts with two SNXT(009) instructions that start processes A and B. Because of the way CIO 000001 (SW A1) and CIO 000002 (SW B1) are programmed, only one of these will be executed with an ON execution condition to start either process A or process B. Both of the steps for these processes end with a SNXT(009) that starts the step (process C).



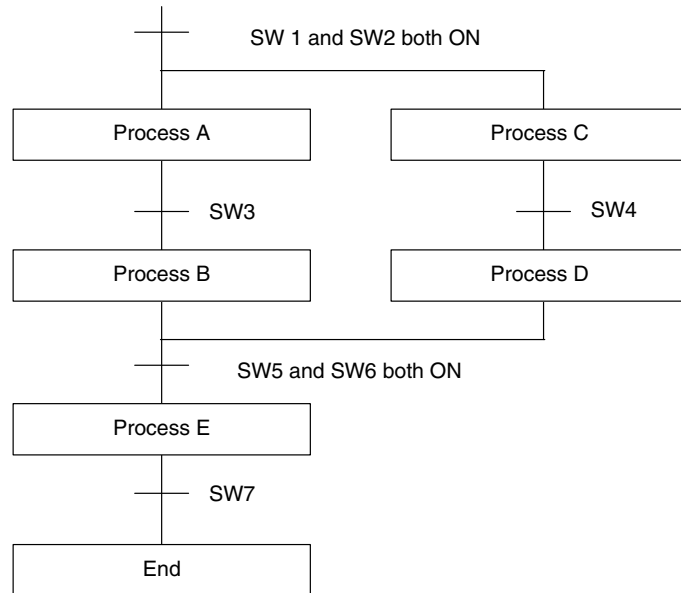
**Note** In the above programming, CIO 010002 is used in two SNXT(0009) instructions. This will not produce a duplication error during the program check.

**Example 3: Parallel Execution**

The following process requires that two parts of a product pass simultaneously through two processes each before they are joined together in a fifth process. Various sensors are positioned to signal when processes are to start and end.

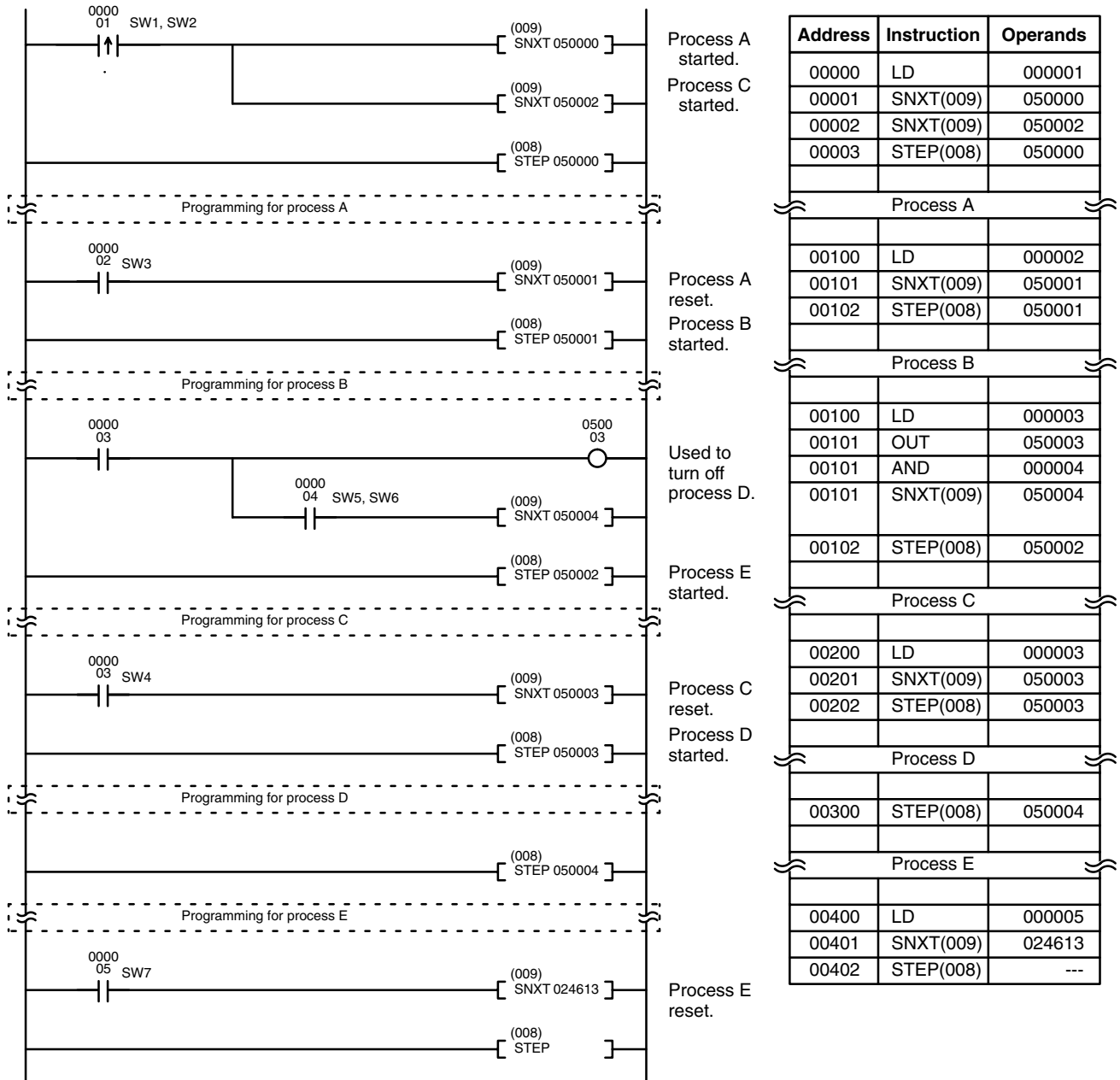


The following diagram demonstrates the flow of processing and the switches that are used for execution control. Here, process A and process C are started together. When process A finishes, process B starts; when process C finishes, process D starts. When both processes B and D have finished, process E starts. When both processes B and D have finished, process E starts.



The program for this operation, shown below, starts with two SNXT(009) instructions that start processes A and C. These instructions branch from the same instruction line and are always executed together, starting steps for both A and C. When the steps for both A and C have finished, the steps for process B and D begin immediately.

When both process B and process D have finished (i.e., when SW5 and SW6 turn ON), processes B and D are reset together by the SNXT(009) at the end of the programming for process B. Although there is no SNXT(009) at the end of process D, the control bit for it is turned OFF by executing SNXT(009) 050004. This is because the OUT for bit 050003 is in the step reset by SNXT(009) 050004, i.e., bit 050003 is turned OFF when SNXT(009) 050004 is executed. Process B is thus reset directly and process D is reset indirectly before executing the step for process E.




## 5-30 Subroutines

Subroutines break large control tasks into smaller ones and enable you to reuse a given set of instructions. When the main program calls a subroutine, control is transferred to the subroutine and the subroutine instructions are executed. The instructions within a subroutine are written in the same way as main program code. When all the subroutine instructions have been executed, control returns to the main program to the point just after the point from which the subroutine was entered (unless otherwise specified in the subroutine).

### 5-30-1 SUBROUTINE ENTRY and RETURN: SBN(150)/RET(152)

#### SUBROUTINE ENTRY: SBN(150)

Ladder Symbol	Operand Data Area
	<b>N: Subroutine number #</b>

#### SUBROUTINE RETURN: RET(152)

Ladder Symbol


#### Description

SBN(150) is used to mark the beginning of a subroutine program; RET(152) is used to mark the end. Each subroutine is identified with a subroutine number, N, that is programmed as the definer for SBN(150). This same subroutine number is used in any SBS(151) that calls the subroutine (see next subsection). No subroutine number is required with RET(152).

All subroutines must be programmed at the end of the main program. When one or more subroutines have been programmed, the main program will be executed up to the first SBN(150) before returning to address 00000 for the next scan; if part of the main program is placed after a SBN(150), it will be executed only as part of a subroutine, if at all. Subroutines will not be executed unless called by SBS(151).

END(001) must be placed at the end of the last subroutine program, i.e., after the last RET(152). END(001) is not required at any other point in the program. (Refer to the next subsection for further details.)

#### Precautions

N must be between 000 and 999 and each subroutine number can be used in SBN(150) once only.

If SBN(150) is mistakenly placed in the main program, it will inhibit program execution past that point, i.e., program execution will return to the beginning when SBN(150) is encountered.

If either DIFU(013) or DIFU(014) is placed within a subroutine, the operand bit will not be turned OFF until the next time the subroutine is executed, i.e., the operand bit may stay ON longer than one cycle.

The step instructions, STEP(008) and SNXT(009), cannot be used within a subroutine.

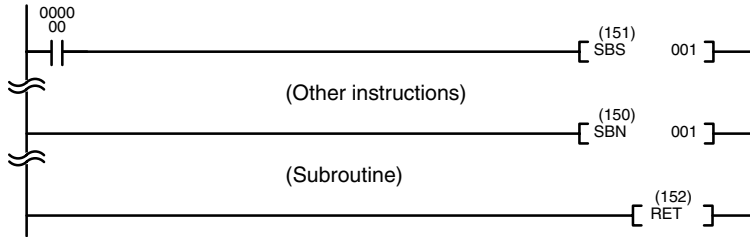
#### Flags

There are no flags directly affected by these instructions.



**Example**

When CIO 000000 is ON in the following example, the instructions between SBN(150) 001 and RET(152) are executed once before returning to execute the next instruction line after SBS(151).



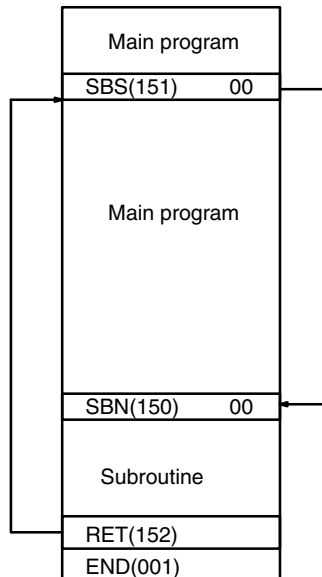
Address	Instruction	Operands
00000	LD	000000
00001	SBS(151)	001
(Other instructions)		
00023	SBN(150)	001
(Subroutine)		
00035	RET(152)	

**5-30-2 SUBROUTINE CALL: SBS(151)**

<p><b>Ladder Symbol</b></p> <p><b>Variations</b> ↑ SBS(151)</p>	<p><b>Operand Data Area</b></p> <p><b>N: Subroutine number #</b></p>
---	--

**Description**

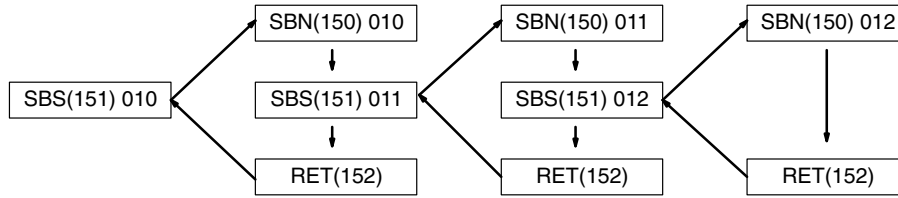
A subroutine can be executed by placing SBS(151) in the main program at the point where the subroutine is desired. The subroutine number used in SBS(151) indicates the desired subroutine. When SBS(151) is executed with an ON execution, the instructions between the SBN(150) with the same subroutine number and the first RET(152) after it are executed before execution returns to the instruction following the SBS(151) that made the call.



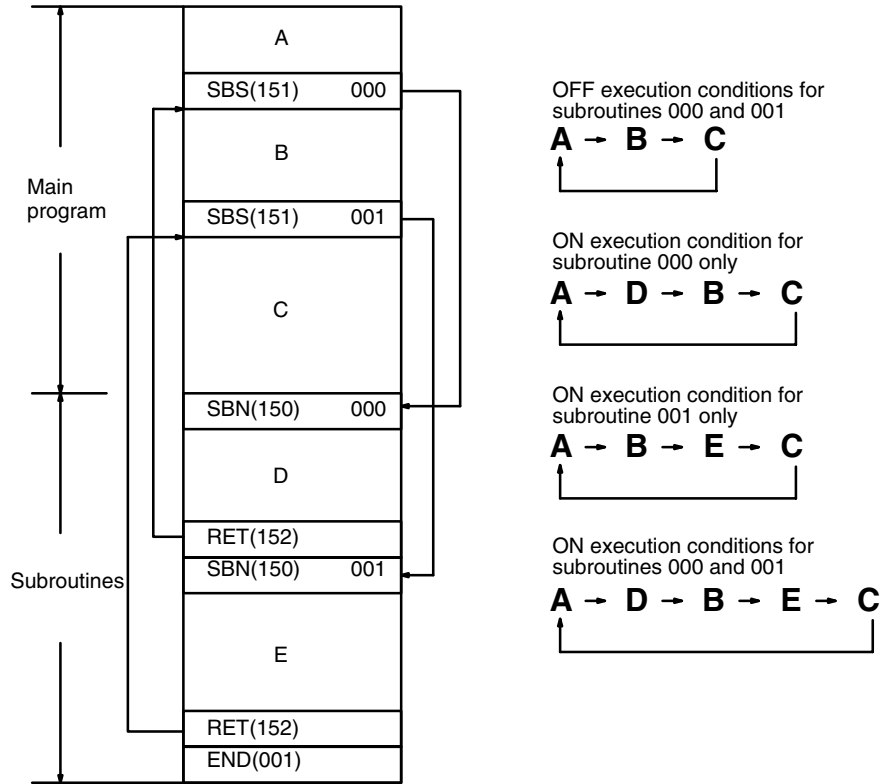
SBS(151) may be used as many times as desired in the program, i.e., the same subroutine may be called from different places in the program.

SBS(151) may also be placed into a subroutine to shift program execution from one subroutine to another, i.e., subroutines may be nested. When the second subroutine has been completed (i.e., RET(152) has been reached), program execution returns to the original subroutine which is then completed before returning to the main program. As many nesting levels as desired can be programmed. A subroutine cannot call itself, (e.g., SBS(151) 00 cannot be programmed within

the subroutine defined with SBN(150) 00). The following diagram illustrates two levels of nesting.



The following diagram illustrates program execution flow for various execution conditions for two SBS(151).



**Precautions**

N must be between 000 and 999.

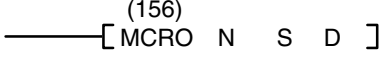
**Flags**

- ER (A50003): No subroutine exists with the specified subroutine number.
- A subroutine has called itself.
- A subroutine being executed has been called.

**Example**

Refer to 5-30-1 *SUBROUTINE ENTRY and RETURN: SBN(150)/RET(152)* for an example.

### 5-30-3 MACRO: MCRO(156)

<p><b>Ladder Symbol</b></p> <p style="text-align: center;">(156)  </p> <p><b>Variations</b></p> <p>↑MCRO(156)</p>	<p><b>Operand Data Areas</b></p> <p><b>N: Subroutine number</b>            000 to 999 or 000 to 099</p> <p><b>S: First input parameter word</b>    CIO, G, A, T, C, DM</p> <p><b>D: First output parameter word</b>   CIO, G, A, T, C, DM</p>
--	---

**Description**

MCRO(156) allows a single subroutine to replace several subroutines that have identical structure but different operands. There are four input words, A200 through A203, and four output words, A204 through A207, allocated to MCRO(156). These eight words are used in the subroutine and take their contents from S through S+3 and then output their contents to D through D+3 after the subroutine is executed.

When the execution condition is OFF, MCRO(156) is not executed. When the execution condition is ON, MCRO(156) copies the contents of S through S+3 to A200 through A203, and then calls and executes the subroutine specified in N. When the subroutine is completed, the contents of A204 through A207 are then transferred back to D through D+3 before MCRO(156) is completed.

Subroutines called by MCRO(156) are defined by SBN(150) and RET(152) just like normal subroutines. For details concerning the use of subroutines, refer to the sections in this manual explaining SBN(150), SBS(151), and RET(152).

**Precautions**

Nesting is possible with MCRO(156), but the data must be saved before calling another subroutine because there is only one processing area (A200 through A207). Recursive calls are not possible, i.e., a subroutine cannot call itself.

N must be between 000 and 999.

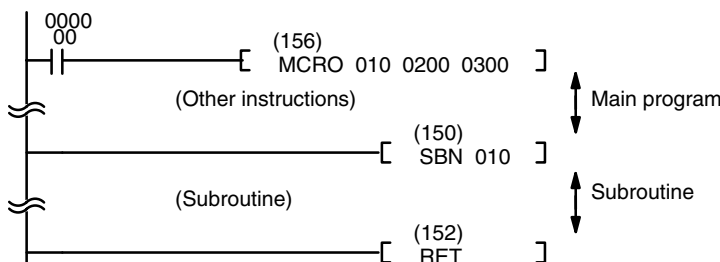
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

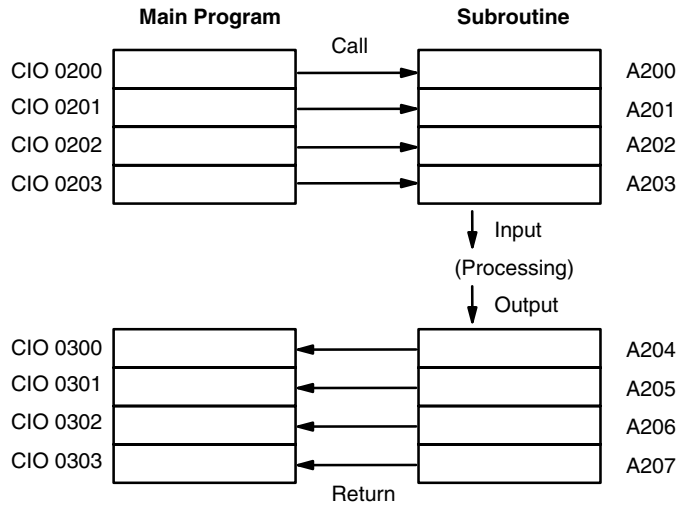
- ER (A50003): Subroutine does not exist for the specified subroutine number.
- A subroutine has called itself.
- An active subroutine has been called.

**Example**

When CIO 000000 is ON in the following example, the contents of CIO 0200 through CIO 0203 are copied to A200 through A203, and subroutine 10 is called and executed. When the subroutine is completed, the contents of A204 through A207 are copied back to CIO 0300 through CIO 0303.

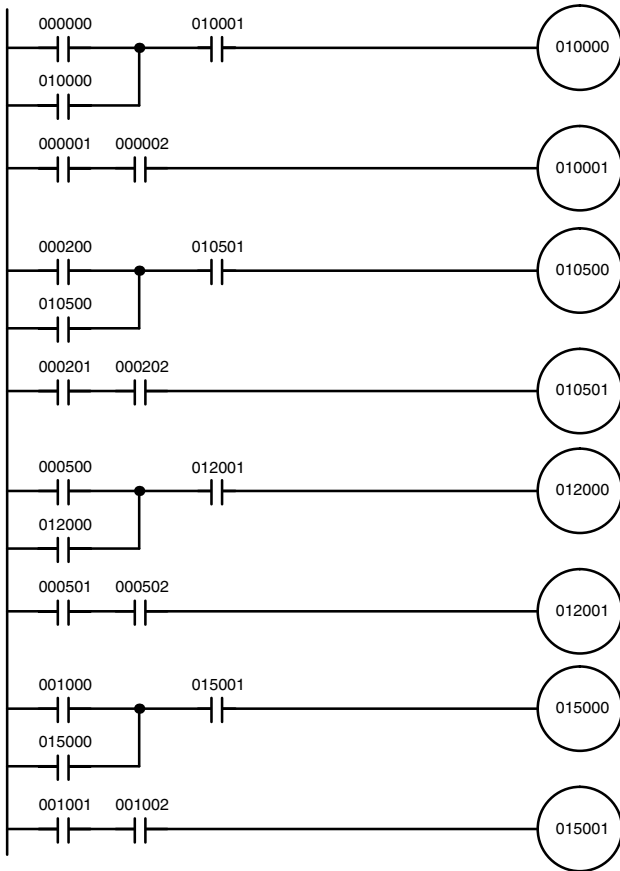


Address	Instruction	Operands
00000	LD	000000
00001	MCRO(156)	010
		0200
		0300
(Other instructions)		
00023	SBN(150)	010
(Subroutine)		
00035	RET(152)	

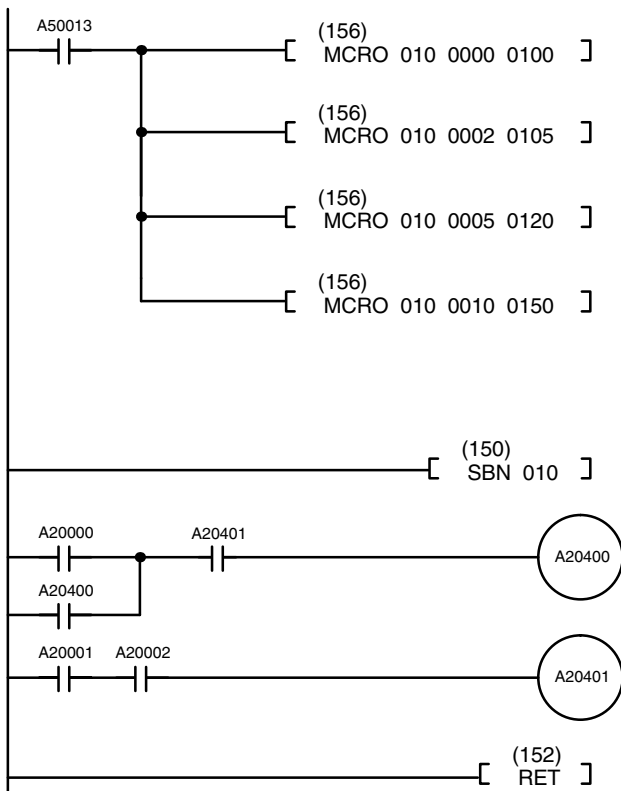


**Program Examples**

The following examples show how MCRO(156) can be used to simplify a program. The second program section uses MCRO(156), whereas the first one does not.



Address	Instruction	Operand
00000	LD	000000
00001	OR	010000
00002	AND	010001
00003	OUT	010000
00004	LD	000001
00005	AND	000002
00006	OUT	010001
00007	LD	000200
00008	OR	010500
00009	AND	010501
00010	OUT	010500
00011	LD	000201
00012	AND	000202
00013	OUT	010501
00014	LD	000500
00015	OR	012000
00016	AND	012001
00017	OUT	012000
00018	LD	000501
00019	AND	000502
00020	OUT	012001
00021	LD	001000
00022	OR	015000
00023	AND	015001
00024	OUT	015000
00025	LD	001001
00026	AND	001002
00027	OUT	015011



Address	Instruction	Operands
00000	LD	A50013
00001	MCRO(156)	010
		0000
		0100
00002	MCRO(156)	010
		0002
		0105
00003	MCRO(156)	010
		0005
		0120
00004	MCRO(156)	010
		0010
		0150
00005	SBN(150)	010
00006	LD	A20000
00007	OR	A20400
00008	AND	A20401
00009	OUT	A20400
00010	LD	A20001
00011	AND	A20002
00012	OUT	A20401
00013	RET(152)	

## 5-31 Stack Instructions

Stack Instructions are used to create and manipulate data tables in memory into which data can be placed and retrieved. Different instructions allow you take data out of the stack in the same order or in the opposite order from which it was placed into the stack. SSET(160) must be used to create a stack before any of the other stack instructions can be used.

### 5-31-1 SET STACK: SSET(160)

<p><b>Ladder Symbol</b></p> <p><b>Variations</b></p> <p>↑ SSET(160)</p>	<p><b>Operand Data Areas</b></p> <p><b>TB1: 1<sup>st</sup> stack address</b> CIO, G, A, DM</p> <p><b>N: Number of words</b> CIO, G, A, T, C, #, DM, DR, IR</p>
---	--

#### Description

When the execution condition is OFF, SSET(160) is not executed. When the execution condition is ON, SSET(160) defines a stack from TB1 to TB1+N-1, and writes zeros to all words from TB1+2 to TB1+N-1.

TB1 contains the memory address of TB1+N-1, and TB1+1 contains the memory address of the word that will be accessed by the next stack instruction. TB1+1 is called the stack pointer, and contains the memory address for TB1+2 after SSET(160) is executed.

SSET(160) must be used to create one or more stacks before any of the other stack instructions can be used.

#### Precautions

N must be BCD between 0003 and 9999.

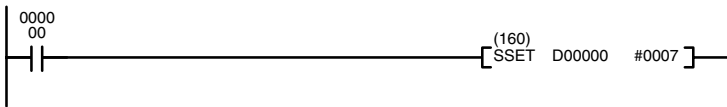
**Note** Refer to page 101 for general precautions on operand data areas.

#### Flags

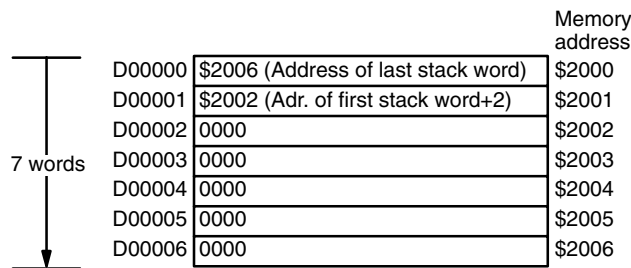
ER (A50003): Content of N is less than 0003, or is not BCD.  
Content of \*DM word is not BCD when set for BCD.

**Example**

When CIO 000000 is ON in the following example, SSET(160) defines a 7-word stack from D00000 to D00006. The memory address of the last word in the stack, \$2006, is written into D00000 and the memory address of TB1+2, \$2002, is written into D00001. The rest of the words in the stack, D00002 to D00006, are reset.



Address	Instruction	Operands
00000	LD	000000
00001	SSET(160)	
		D00000
		#0007



**5-31-2 PUSH ONTO STACK: PUSH(161)**

<p><b>Ladder Symbol</b></p> <p><b>Variations</b></p> <p>↑ PUSH(161)</p>	<p><b>Operand Data Areas</b></p> <p><b>TB1: 1<sup>st</sup> stack address</b> CIO, G, A, DM</p> <p><b>S: Source word</b> CIO, G, A, T, C, #, DM, DR, IR</p>
---	--

**Description**

When the execution condition is OFF, PUSH(161) is not executed. When the execution condition is ON, PUSH(161) copies the data from the source word (S) to the word indicated by the stack pointer (TB1+1). The address in the stack pointer is then incremented by one.

**Precautions**

TB1 must be the first address of a stack defined using SSET(160).  
Do not allow the stack pointer to be incremented higher than the address of the last word in the stack. If the content of the stack pointer is greater than the address in TB1, the ER Flag (A50003) will be turned ON.

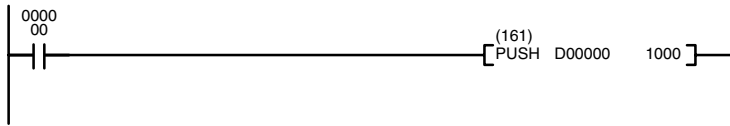
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

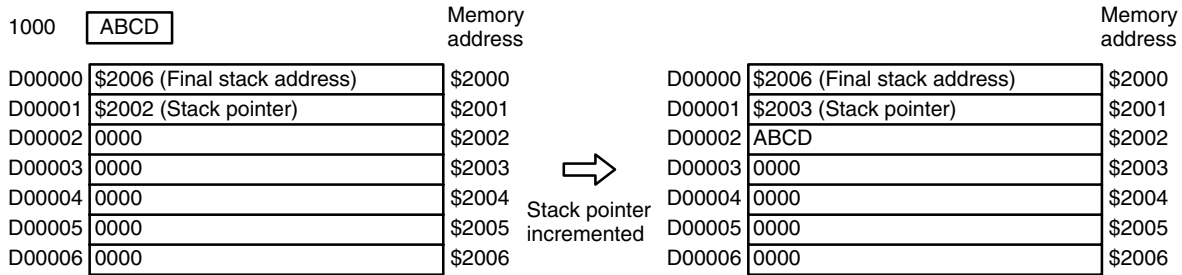
ER (A50003): Content of TB1+1 is greater than the content of TB1.  
Content of \*DM word is not BCD when set for BCD.

**Example**

When CIO 000000 is ON in the following example, PUSH(161) is used to write the data in CIO 1000 to the 7-word stack from D00000 to D00006. The stack pointer contains the memory address of D00002, so the data in CIO 1000 is copied to D00002. The content of the stack pointer is then incremented from \$2002 to \$2003.



Address	Instruction	Operands
00000	LD	000000
00001	PUSH(161)	
		D00000
		1000



**5-31-3 LAST IN FIRST OUT: LIFO(162)**

<p><b>Ladder Symbol</b></p> <p><b>Variations</b></p> <p>↑ LIFO(162)</p>	<p><b>Operand Data Areas</b></p> <p><b>TB1: 1<sup>st</sup> stack address</b> CIO, G, A, DM</p> <p><b>D: Destination word</b> CIO, G, A, DM, DR, IR</p>
---	--

**Description**

When the execution condition is OFF, LIFO(162) is not executed. When the execution condition is ON, LIFO(162) decrements the memory address in the stack pointer (TB1+1) by one, and then copies the data from the word indicated by the stack pointer (the last written to the stack) to the destination word (D). The stack pointer is the only word changed in the stack.

**Precautions**

TB1 must be the first address of a stack defined using SSET(160). Do not allow the stack pointer to be decremented to the memory address of the stack pointer. If the content of the stack pointer is less than or equal to the address of the stack pointer itself, the ER Flag (A50003) will be turned ON.

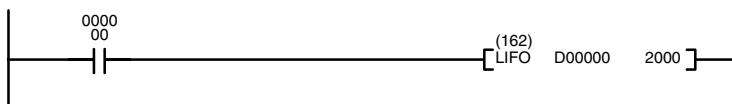
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

ER (A50003): Content of TB1+1 is less than or equal to the address of TB1+1.  
Content of \*DM word is not BCD when set for BCD.

**Example**

When CIO 000000 is ON in the following example, LIFO(162) is used to decrement the content of the stack pointer from \$2004 to \$2003, and copy the last data written to the stack to CIO 2000.



Address	Instruction	Operands
00000	LD	000000
00001	LIFO(162)	
		D00000
		2000

	Memory address		Memory address		
D00000	\$2006 (Final stack address)	\$2000	D00000	\$2006 (Final stack address)	\$2000
D00001	\$2004 (Stack pointer)	\$2001	D00001	\$2003 (Stack pointer)	\$2001
D00002	ABCD	\$2002	D00002	ABCD	\$2002
D00003	37B4	\$2003	D00003	37B4 (Moved to CIO 0200)	\$2003
D00004	0000	\$2004	D00004	0000	\$2004
D00005	0000	\$2005	D00005	0000	\$2005
D00006	0000	\$2006	D00006	0000	\$2006

Stack pointer decremented

2000

37B4

### 5-31-4 FIRST IN FIRST OUT: FIFO(163)

<p><b>Ladder Symbol</b></p> <p><b>Variations</b></p> <p>↑ FIFO(162)</p>	<p><b>Operand Data Areas</b></p> <p><b>TB1: 1<sup>st</sup> stack address</b> CIO, G, A, DM</p> <p><b>D: Destination word</b> CIO, G, A, DM, DR, IR</p>
---	--

**Description**

When the execution condition is OFF, FIFO(163) is not executed. When the execution condition is ON, FIFO(163) writes zeros into the last word of the stack and shifts the contents of each word within the stack down by one address, finally shifting the data from TB1+2 (the first value written to the stack) to the destination word (D). The memory address in the stack pointer (TB1+1) is then decremented by one.

**Precautions**

TB1 must be the first address of a stack defined using SSET(160). Do not allow the stack pointer to be decremented to the memory address of the stack pointer. If the content of the stack pointer is less than or equal to the memory address of the stack pointer itself, the ER Flag (A50003) will be turned ON.

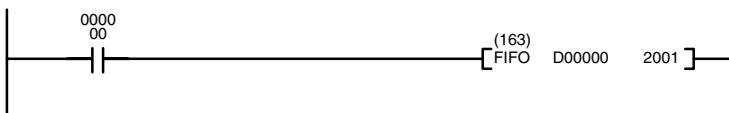
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

ER (A50003): TB1+1 is less than or equal to the address of the stack pointer. Content of \*DM word is not BCD when set for BCD.

**Example**

When CIO 00000 is ON in the following example, the first value in the stack (ABCD at D00002) is moved to CIO 2001 and all values remaining in the stack are moved up on word in the stack.



Address	Instruction	Operands
00000	LD	000000
00001	FIFO(163)	
		D00000
		2001

	Memory address		Memory address		
D00000	\$2006 (Final stack address)	\$2000	D00000	\$2006 (Final stack address)	\$2000
D00001	\$2005 (Stack pointer)	\$2001	D00001	\$2004 (Stack pointer)	\$2001
D00002	ABCD	\$2002	D00002	37B4	\$2002
D00003	37B4	\$2003	D00003	8E19	\$2003
D00004	8E19	\$2004	D00004	8E19	\$2004
D00005	0000	\$2005	D00005	0000	\$2005
D00006	0000	\$2006	D00006	0000	\$2006

CIO 2001

ABCD

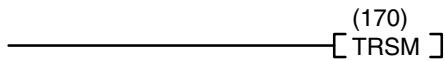


## 5-32 Data Tracing

Data tracing can be used to facilitate debugging programs and is described in detail in the SSS *Operation Manuals*. This section shows the ladder symbols for TRSM(170) and MARK(171) and provides example programs.

### 5-32-1 TRACE MEMORY SAMPLING: TRSM(170)

#### Ladder Symbol



#### Description

TRSM(170) is used in the program to mark locations where specified data is to be stored in Trace Memory. Up to 12 bits and up to 3 words may be designated for tracing.

TRSM(170) is not controlled by an execution condition, but rather by two bits in the Auxiliary Area: A00815 and A00814. A00815 is the Sampling Start Bit. This bit is turned ON to start the sampling processes for tracing. The Sampling Start Bit must not be turned ON from the program, i.e., it must be turned ON only from SSS. A00814 is the Trace Start Bit. When it is set, the specified data is recorded in Trace Memory. The Trace Start Bit can be set either from the program or from SSS. A positive or negative delay can also be set to alter the actual point from which tracing will begin.

Data can be recorded in two ways. In the first method, a timer interval is set from SSS so that the specified data will be traced at a regular interval independent of the cycle time (refer to *Data Tracing* in the *SYSMAC Support Software Operation Manual*). The timer interval can be set to between 5 and 2550 ms, in 5 ms steps. If the timer interval is set to 0 ms, the sampling will take place once each cycle; data will be recorded periodically and TRSM(170) instructions in the program won't trigger sampling.

To disable periodic sampling and enable sampling when TRSM(170) is executed in the program, the timer interval is set to "TRSM." TRSM(170) can be placed at one or more locations in the program to indicate where the specified data is to be traced.

TRSM(170) can be incorporated anywhere in a program, any number of times. The data in the trace memory can be monitored via SSS.

#### Control Bits and Flags

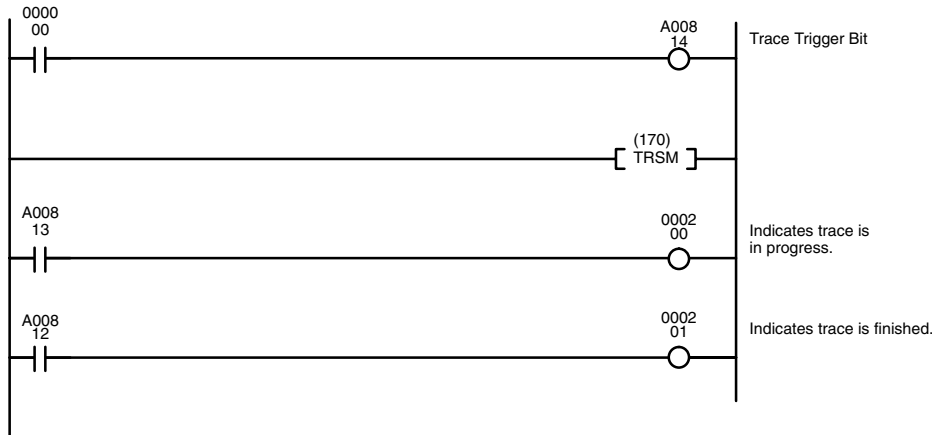
The following control bits and flags are used during data tracing. The Tracing Flag will be ON during tracing operations. The Trace Completed Flag will turn ON when enough data has been traced to fill Trace Memory.

Only A00814 and A00815 are meant to be controlled by the user, and A00815 must not be turned ON from the program, i.e., it must be turned ON only from SSS.

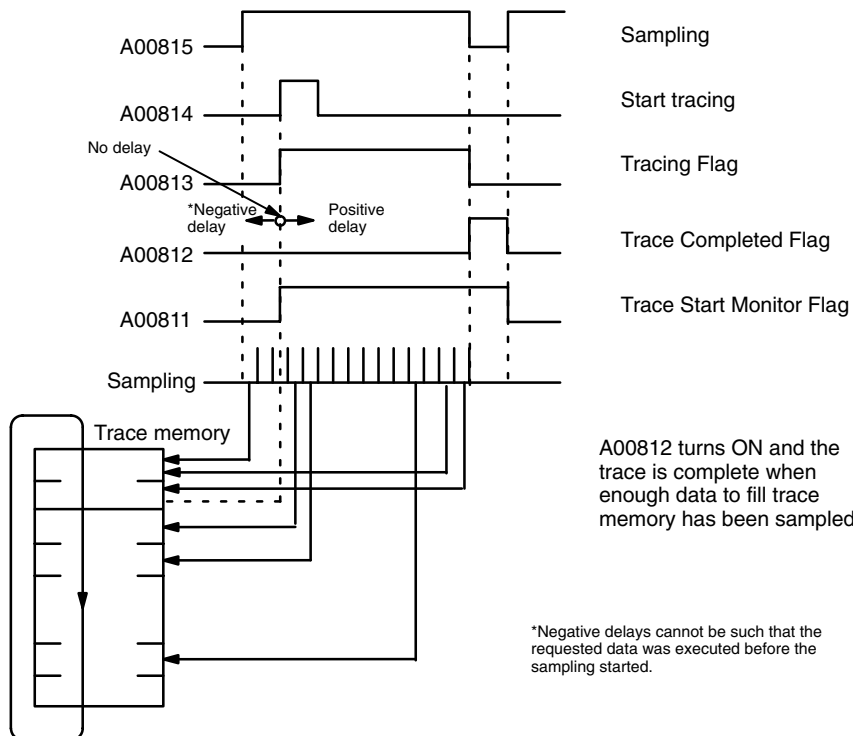
Flag	Function
A00811	Trace Trigger Monitor Flag
A00812	Trace Completed Flag
A00813	Trace Busy Flag
A00814	Trace Start Bit
A00815	Sampling Start Bit

**Example**

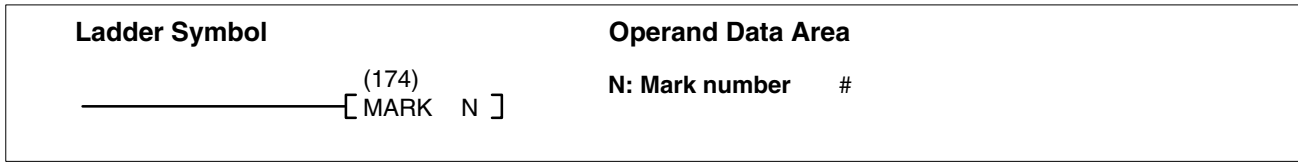
The following shows the basic program and operation for data tracing. The Sampling Start Bit starts the sampling. The data is read and stored into trace memory. When the Trace Start Bit is received, the CPU looks at the delay and marks the trace memory accordingly. This can mean that some of the samples already made will be recorded as the trace memory (negative delay), or that more samples will be made before they are recorded (positive delay). The sampled data is written to trace memory, jumping to the beginning of the memory area once the end has been reached and continuing up to the start marker. This might mean that previously recorded data (i.e., data from this sample that falls before the start marker) is overwritten (this is especially true if the delay is positive). The negative delay cannot be such that the required data was executed before sampling was started.



Address	Instruction	Operands
00000	LD	000000
00001	OUT	A00814
00002	TRSM(170)	
00003	LD	A00813
00004	OUT	000200
00005	LD	A00812
00006	OUT	000201



### 5-32-2 MARK TRACE: MARK(174)



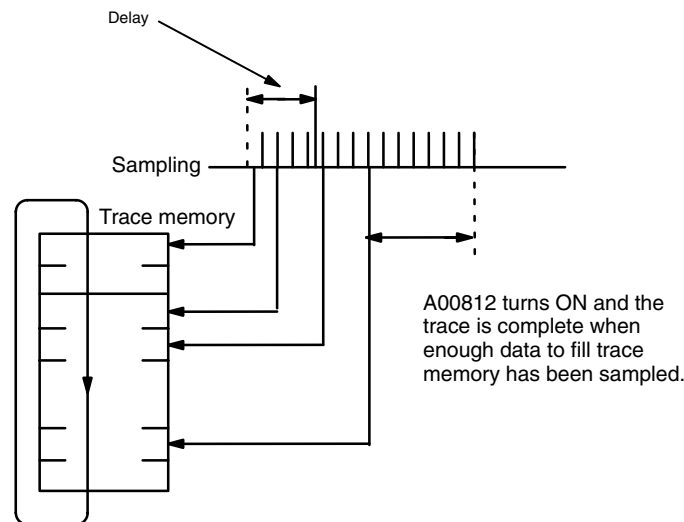
**Description**

Like TRSM(170), MARK(174) is used in the program to mark locations where specified data is to be stored in Trace Memory. Two words may be designated for tracing, and each time the MARK(174) instruction is executed, the word address, content, and mark number are stored in Trace Memory.

MARK(174) can also be used to measure the elapsed time between the execution of two MARK(174) instructions. The Execution Time Measured Flag (A00808) turns ON when the specified execution time has been measured. Refer to the SSS *Operation Manuals* for details.

The word addresses, trigger conditions, and delay must be set beforehand from the SSS. MARK(174) uses the control bits and flags shown in the following table.

Flag	Function
A00808	Execution Time Measured Flag
A00811	Trace Trigger Monitor Flag
A00812	Trace Completed Flag
A00813	Trace Busy Flag
A00814	Trace Start Bit
A00815	Sampling Start Bit



**Precautions**

N must be BCD.

## 5-33 Special I/O Instructions

The Special I/O Instructions are used to write data to or read data from Special I/O Units, such as an ASCII Unit. Refer to the operation manual of the Special I/O Unit for details on the use and content of data transfers.

### 5-33-1 I/O READ: READ(190)

Ladder Symbol	Operand Data Areas
	<b>N: Words to transfer</b> CIO, G, A, T, C, #, DM, DR, IR <b>S: Source word</b> CIO <b>D: 1<sup>st</sup> destination word</b> CIO, G, A, T, C, DM
<b>Variations</b> ↑ READ(190)	

#### Description

When the execution condition is OFF, READ(190) is not executed. When the execution condition is ON, READ(190) reads data from the memory area of the Special I/O Unit allocated word S and transfers it to D through D+N-1. N indicates the number of words to be read. S indicates the rightmost (first) of the two words allocated for the Special I/O Unit (i.e., the input word).

This instruction cannot be used to read data from Special I/O Units mounted to Slave Racks in SYSMAC BUS Remote I/O Systems. It can be used for certain Special I/O Units mounted to newer version of SYSMAC BUS/2 Remote I/O Systems. Refer to page 34 for details.

#### Precautions

N must be BCD. (Special I/O Units can transfer up to 255 words of data.)

S must be in the I/O Area and allocated to a Special I/O Unit.

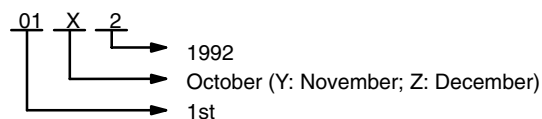
If the ER Flag (A50003) or CY Flag (A50004) is turned ON, the instruction will not be executed.

If the data cannot be sent or the Special I/O Unit is busy, the data will be transferred during the next cycle. To make sure that READ(190) execution has been completed, check EQ (A50006).

A maximum of two READ(190)/WRIT(191) instructions can be used for Slaves connected to the same Master. If a third instruction is attempted when two instructions are already being executed, the third instruction will not be executed and the Carry Flag (A50004) will turn ON. Be sure to control execution of these instructions so that no more than two are being executed simultaneously for Units connected under the same Master.

This instruction cannot be used to read data from Special I/O Units mounted to Slave Racks in SYSMAC BUS Remote I/O Systems. It can be used for Units mounted to Slave Racks in SYSMAC BUS/2 Systems as long as the the following conditions are met.

- 1, 2, 3... 1. The lot number of the Remote I/O Master Unit and Remote I/O Slave Unit must be the same as or latter than the following.



2. The DIP switch on the Remote I/O Slave Unit must be set to "54MH." (Only four Slaves can be connected to each Master when this setting is used.)
3. The Special I/O Unit must be one of the following: AD101, CT012, CT041, ASC04, IDS01-V1, IDS02, IDS21, IDS22, LDP01-V1, or NC222.

4. No more than one READ(190) and/or WRIT(191) cannot be executed for the same Special I/O Unit at the same time. Be sure the first instruction has completed execution before starting execution of another READ(190)/WRIT(191) instruction.

**Note** Refer to page 101 for general precautions on operand data areas.

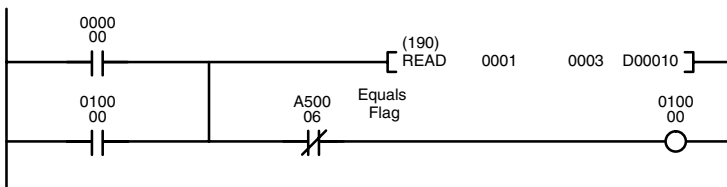
**Flags**

- ER (A50003): Content of \*DM word is not BCD when set for BCD.  
 S is not allocated to a Special I/O Unit.  
 N is not BCD.  
 Instruction executed reading more than 255 words from a Special I/O Unit on a SYSMAC BUS/2 Slave Rack.
- CY (A50004): This Flag operates only for Special I/O Units mounted to SYSMAC BUS/2 Slave Racks and turns on for the following:  
 The Special I/O Unit or Slave is busy.  
 An error has occurred in the Special I/O Unit.  
 Word operands have been designated for a Special I/O Unit that requires a constant.  
 The Slave is not set to 54MH.  
 The communications path is not normal.
- EQ (A50006): OFF while data is being read; ON when reading has been completed.

**Example**

When CIO 000000 is ON in the following example, the number of words specified in CIO 0001 is read through CIO 0003 (the I/O word allocated to the Special I/O Unit) and stored at consecutive words starting at D00010.

The following program section uses a self-holding bit to ensure that the read operation is executed even if the Special I/O Unit (or Slave Rack) is busy when READ(190) is executed. The type of programming is also effective when the read operation requires more than one cycle. The Equals Flag, which turns ON when the reading operation is completed, is used to end the execution.



Address	Instruction	Operands
00000	LD	000000
00001	OR	010000
00002	READ(190)	0001 0003 D00010
00003	AND NOT	A50006
00004	OUT	010000

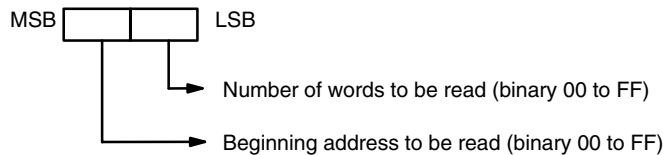
## 5-33-2 I/O READ 2: RD2(280)

Ladder Symbol	Operand Data Areas
	<p><b>C: Control word</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>S: Source word</b> CIO</p> <p><b>D: First destination word</b> CIO, G, A, T, C, DM</p>

**Description**

When the execution condition is OFF, RD2(280) is not executed. When the execution condition is ON, RD2(280) reads the memory contents of a Special I/O Unit to specified words (beginning with D) in the Programmable Controller through the source word ( S ). The word specified for S is the third of the four words (i.e., the first input word) that serve as the interface with the Special I/O Unit.

The words that are to be transferred are specified by the control word. The contents of the control word are as follows:

**Control Word Contents**

The beginning address to be read specifies the rightmost (lowest) word of the range of data that is to be read from the memory area in the Special I/O Unit.

If 00 is specified as the number of words to be read, the instruction will not be executed. If the sum of the beginning address to be read plus the number of words to be read is more than 100 (binary), the Error Flag (A5003) will turn ON.

If an error occurs at the Special I/O Unit (such as, for example, a non-readable area being specified), the Carry Flag (A50004) will turn ON.

The Equals Flag (A50006) can be used to check whether or not RD2(280) execution has been completed.

**Precautions**

Several cycles may be required before RD2(280) execution is completed. During that time, the execution condition for RD2(28) must remain ON.

RD2(280) carries out data exchange with the Special I/O Unit via the I/O area, so the time required to complete execution depends on the I/O refresh interval (i.e., the cycle time).

Be sure that there is a Special I/O Unit mounted. If no Special I/O Unit is mounted, RD2(280) execution will continue without stopping.

As of this printing, the RD2(280) can only be used with the C500-CT021 High-speed Counter Unit is set to four-word operation (normally, when used with SYS-MAC BUS).

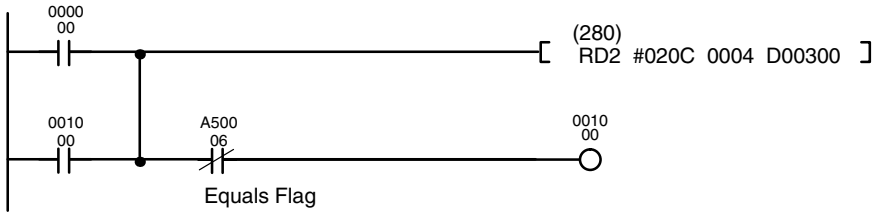
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

- ER (A50003): Beginning address plus number of words exceeds 100 binary.  
Content of \*DM word is not BCD when set for BCD.
- CY (A50004): An error has occurred at the Special I/O Unit.
- EQ (A50006): The read operation has been completed.

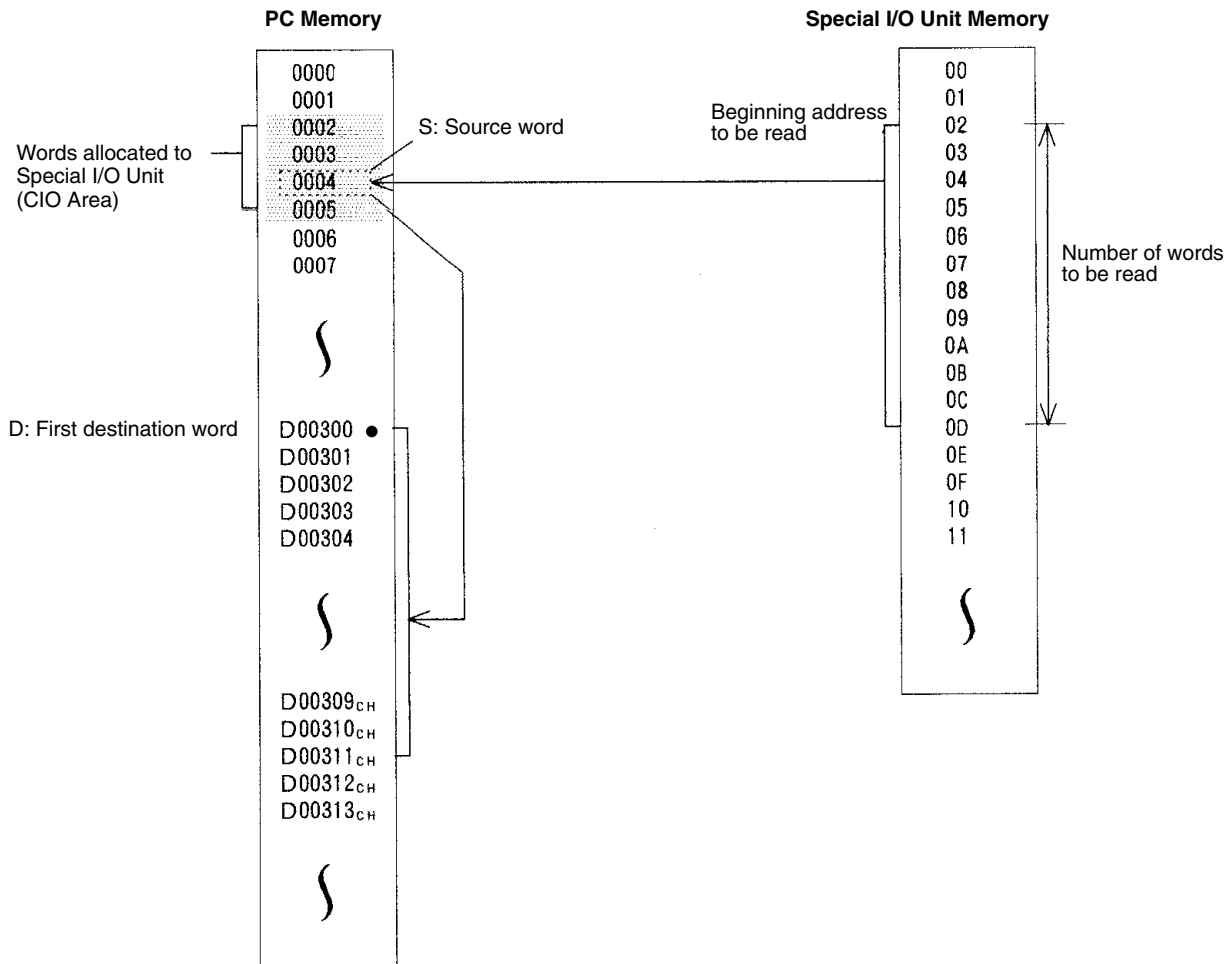
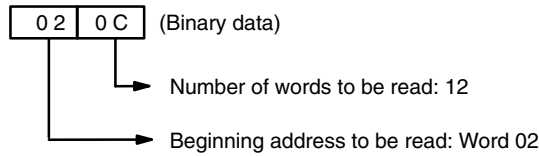
**Example**

When CIO 000000 is ON in the following example, the contents of words 02 through 12 in the Special I/O Unit's memory area are read in order, one word at a time, through CIO 0004, and the contents that are read are transferred in order to D00300 through D00311.



Address	Instruction	Operands
00000	LD	000000
00001	OR	001000
00002	RD2(280)	
		#020C
		0004
		D00300
00003	AND NOT	A50006
00004	OUT	001000

**C: Control Word Contents**



### 5-33-3 I/O WRITE: WRIT(191)

Ladder Symbol	Operand Data Areas
<p>(191) — [ WRIT    N    S    D ]</p>	<p><b>N: Words to transfer</b>    CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>S: 1<sup>st</sup> source word</b>    CIO, G, A, T, C, DM</p> <p><b>D: Destination word</b>    CIO</p>
<p><b>Variations</b></p> <p>↑ WRIT(191)</p>	

**Description**

When the execution condition is OFF, WRIT(191) is not executed. When the execution condition is ON, WRIT(191) transfers the contents of S through S+(N-1) to the Special I/O Unit allocated word D. N indicates the number of words to be read. D indicates the rightmost (first) of the two words allocated for the Special I/O Unit (i.e., the output word).

**Precautions**

N must be BCD. (Special I/O Units can transfer up to 255 words of data.)

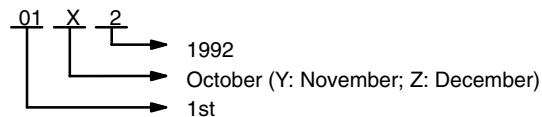
D must be in the I/O Area and allocated to a Special I/O Unit.

If a Special I/O Unit is busy and unable to receive data, the data will be written during the next cycle. To make sure that WRIT(191) execution has been completed, check EQ (A50006).

A maximum of two READ(190)/WRIT(191) instructions can be used for Slaves connected to the same Master. If a third instruction is attempted when two instructions are already being executed, the third instruction will not be executed and the Carry Flag (A50004) will turn ON. Be sure to control execution of these instructions so that no more than two are being executed simultaneously for Units connected under the same Master.

This instruction cannot be used to write data to Special I/O Units mounted to Slave Racks in SYSMAC BUS Remote I/O Systems. It can be used for Units mounted to Slave Racks in SYSMAC BUS/2 Systems as long as the the following conditions are met.

- 1, 2, 3...**
1. The lot number of the Remote I/O Master Unit and Remote I/O Slave Unit must be the same as or latter than the following.



2. The DIP switch on the Remote I/O Slave Unit must be set to “54MH.” (Only four Slaves can be connected to each Master when this setting is used.)
3. The Special I/O Unit must be one of the following: AD101, CT012, CT041, ASC04, IDS01-V1, IDS02, IDS21, IDS22, LDP01-V1, or NC222.
4. No more than one READ(190) and/or WRIT(191) cannot be executed for the same Special I/O Unit at the same time. Be sure the first instruction has completed execution before starting execution of another READ(190)/WRIT(191) instruction.

**Note** Refer to page 101 for general precautions on operand data areas.



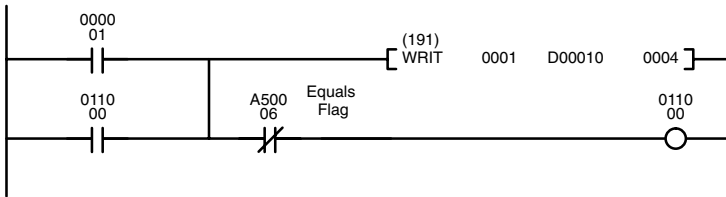
**Flags**

- ER (A50003): Content of \*DM word is not BCD when set for BCD.  
D is not allocated to a Special I/O Unit.  
N is not BCD.  
Instruction executed transferring more than 255 words to a Special I/O Unit on a SYSMAC BUS/2 Slave Rack.
- CY (A50004): This Flag operates only for Special I/O Units mounted to SYSMAC BUS/2 Slave Racks and turns on for the following:  
The Special I/O Unit or Slave is busy.  
An error has occurred in the Special I/O Unit.  
Word operands have been designated for a Special I/O Unit that requires a constant.  
The Slave is not set to 54MH.  
The communications path is not normal.
- EQ (A50006): OFF while data is being written; ON when writing has been completed.

**Example 1**

When CIO 000001 is ON in the following example, the number of words specified in CIO 0001 is read consecutive from words starting at D00010 and transferred to the Special I/O Unit through CIO 0004 (the I/O word allocated to the Special I/O Unit).

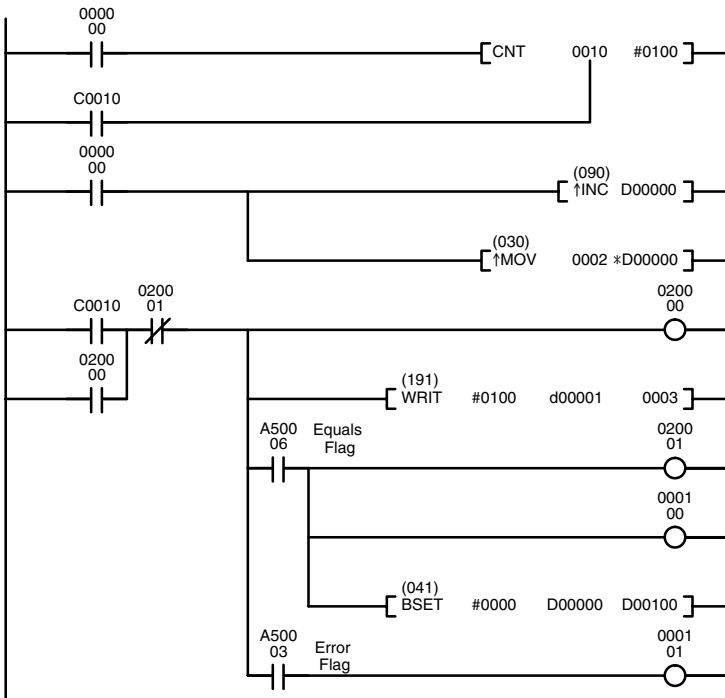
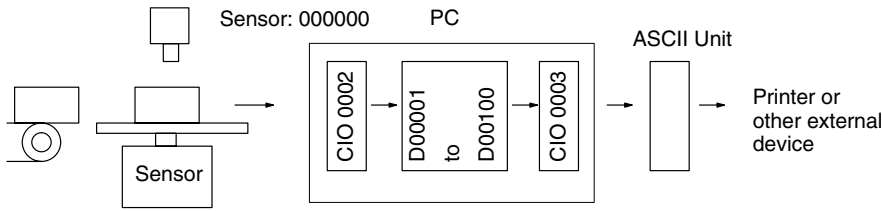
The following program section uses a self-holding bit to ensure that the write operation is executed even if the Special I/O Unit (or Slave Rack) is busy when WRIT(191) is executed. The type of programming is also effective when the read operation requires more than one cycle. The Equals Flag, which turns ON when the reading operation is completed, is used to end execution.



Address	Instruction	Operands
00000	LD	000000
00001	OR	011000
00002	WRIT(191)	
		0001
		D00010
		0004
00003	AND NOT	A50006
00004	OUT	011000

**Example 2**

The following program section shows one way to pass data from a weighing station on a conveyor line through an ASCII Unit to a printer or other external device. Data is input via CIO 0002, stored in D00001 through D00100, and output via CIO 0003.



Address	Instruction	Operands
0000	LD	000000
00001	LD	C0010
00002	CNT	0010
		#0100
00003	LD	000000
00004	↑INC(090)	
		D00000
00005	↑MOV(030)	
		0002
		*D00000
00006	LD	C0010
00007	OR	020000
00008	AND NOT	020001
00009	OUT	020000
00010	WRIT(191)	
		#0100
		D00001
		0003
00011	AND	A50006
00012	OUT	020001
00013	OUT	000100
00014	BSET(041)	
		#0000
		D00000
		D00100
00015	AND	A50003
00016	OUT	000101

**5-33-4 I/O WRITE 2: WR2(281)**

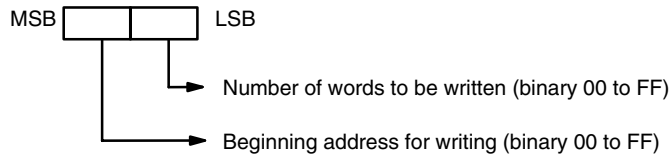
Ladder Symbol	Operand Data Areas
	<p><b>C: Control word</b> CIO, G, A, T, C, #, DM, DR, IR</p> <p><b>S: First source word</b> CIO, G, A, T, C, DM</p> <p><b>D: Destination word</b> CIO</p>

**Description**

When the execution condition is OFF, WR2(281) is not executed. When the execution condition is ON, WR2(281) writes the specified number of words to the specified address in a Special I/O Unit via the destination word (D). The word specified for D is the first of the four words (i.e., the first output word) that serve as the interface with the Special I/O Unit.

The words that are to be transferred are specified by the control word. The contents of the control word are as follows:

**Control Word Contents**



The beginning address for writing specifies the rightmost (lowest) word of the range of in the Special I/O Unit into which the data is to be written.

If 00 is specified as the number of words to be written, the instruction will not be executed. If the sum of the beginning address for writing plus the number of words to be written is more than 100 (binary), the Error Flag (A5003) will turn ON. If an error occurs at the Special I/O Unit (such as, for example, an area for in writing is not possible being specified), the Carry Flag (A50004) will turn ON. The Equals Flag (A50006) can be used to check whether or not WR2(281) execution has been completed.

**Precautions**

Several cycles may be required before WR2(281) execution is completed. During that time, the execution condition for RD2(28) must remain ON.

WR2(281) carries out data exchange with the Special I/O Unit via the I/O area, so the time required to complete execution depends on the I/O refresh interval (i.e., the cycle time).

Be sure that there is a Special I/O Unit mounted. If no Special I/O Unit is mounted, WR2(210) execution will continue without stopping.

As of this printing, WR2(281) can only be used for the C500-CT021 High-speed Counter Unit is set for four-word operation (normally, when used with SYSMAC BUS).

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

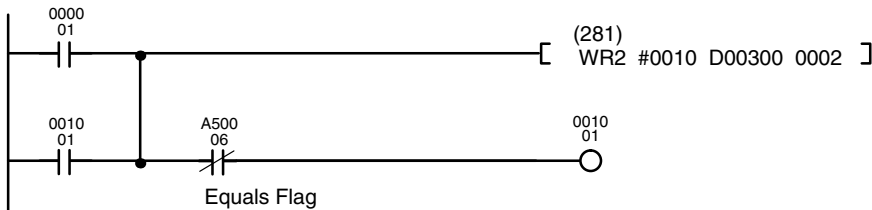
ER (A50003): Beginning address plus number of words exceeds 100 binary. Content of \*DM word is not BCD when set for BCD.

CY (A50004): An error has occurred at the Special I/O Unit.

EQ (A50006): The write operation has been completed.

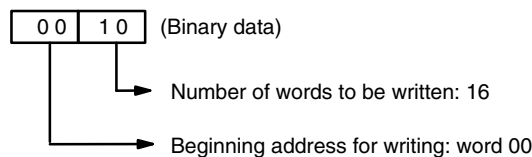
**Example**

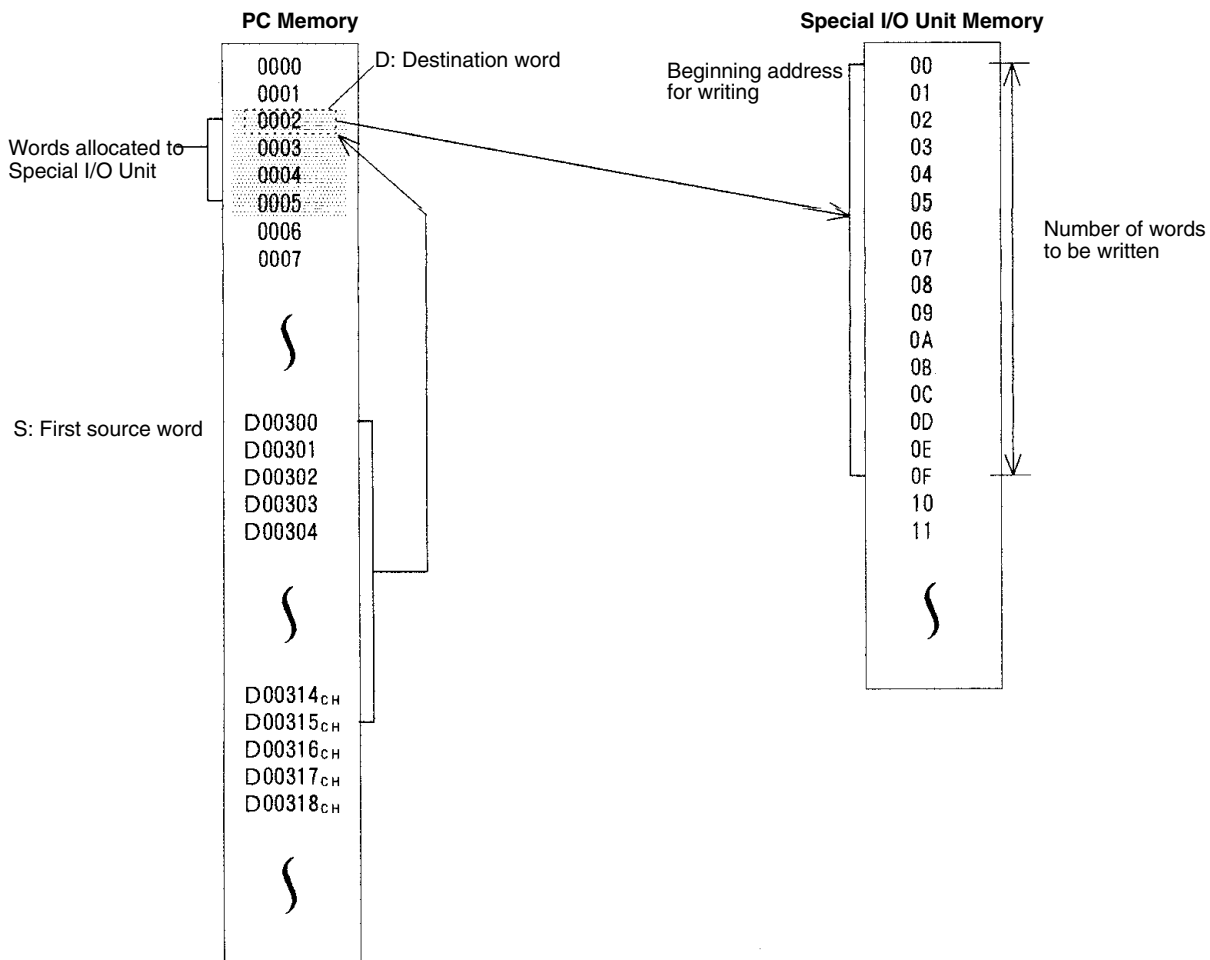
When CIO 000001 is ON in the following example, the 16 words beginning with D00300 are transferred in order through CIO 0002 and are written in order to words 00 through 0F in the Special I/O Unit's memory area.



Address	Instruction	Operands
00000	LD	000001
00001	OR	001001
00002	WR2(281)	
		#0010
		D00300
		0002
00003	AND NOT	A50006
00004	OUT	001001

**C: Control Word Contents**





## 5-34 Network Instructions

The Network Instructions are used for communicating with other control PCs or other Units linked through the SYSMAC NET Link System, SYSMAC LINK System, or Controller Link System. The first two Network Instructions are used to control the access right to the PC from both local and remote Programming Devices.

### 5-34-1 DISABLE ACCESS: IOSP(187)

Ladder Symbol	Variations
	↑ IOSP(187)

#### Description

When the execution condition is OFF, IOSP(187) is not executed. When the execution condition is ON, both read and write access to PC memory is disabled from Programming Devices connected through the Programming Device Connector or Host Link Connector, CPU Bus Units, and Communications Units (other than SYSMAC BUS Units). Access to memory is disabled until either END(001) or IORS(188) is executed or PC operation is stopped.

If PC memory is being accessed when IOSP(187) is executed, access will not be disabled until the current accessing has been completed.

IOSP(187) is designed to temporarily disable access, e.g., during read/write operations. Servicing of CPU Bus Units, the host link interface, and Programming Devices can be disabled for longer intervals by turning ON the bits shown in the following table. These bits can be turned ON at the beginning of the user program to more permanently disable memory access than is possible with IOSP(187), which is effective only until the next IORS(188) or END(01) instruction (or until power is turned OFF).

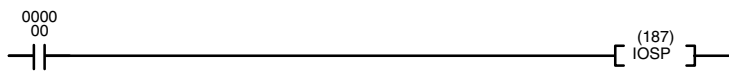
Bit(s)	Name	Function
A01500 to A01515	CPU Service Disable Bits	Turned ON to stop service to CPU Bus Units numbered #0 through #15, respectively.
A01703	Host Link Service Disable Bit	Turned ON to stop Host Link and NT Link System servicing.
A01704	Programming Device Service Disable Bit	Turned ON to stop service to Programming Devices.

**Flags**

There are no flags affected by this instruction.

**Example**

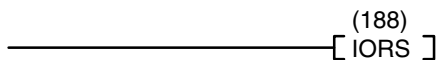
When CIO 000000 is ON in the following example, memory access is disabled from Programming Devices, through communications systems, and from other specified Units.



Address	Instruction	Operands
00000	LD	000000
00001	IOSP(187)	

### 5-34-2 ENABLE ACCESS: IORS(188)

**Ladder Symbol**



**Description**

When the execution condition is OFF, IORS(188) is not executed. When the execution condition is ON, both read and write access to PC memory from Programming Devices, CPU Bus Units, and Communications Units is enabled.

**Flags**

There are no flags affected by this instruction.

**Example**

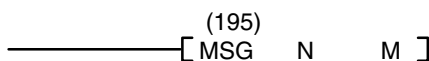
When CIO 000001 is ON in the following example, memory access is enabled.



Address	Instruction	Operands
00000	LD	000000
00001	IORS(188)	

### 5-34-3 DISPLAY MESSAGE: MSG(195)

**Ladder Symbol**



**Operand Data Areas**

**N: Message number** CIO, G, A, T, C, #, DM, DR, IR

**M: 1<sup>st</sup> message word** CIO, G, A, T, C, #, DM

**Variations**

↑ MSG(195)

**Description**

When the execution condition is OFF, MSG(195) is not executed. When executed with an ON execution condition, MSG(195) reads sixteen words of extended ASCII from M to M+15 and displays the message on the SSS screen or other Programming Device. The displayed message can be up to 32 characters long, i.e., each ASCII character code requires eight bits (two hexadecimal digits). Refer to *Appendix H* for the extended ASCII codes. Japanese katakana characters are included in this code.

If the message data changes while the message is being displayed, the display will also change.

If not all sixteen words are required for the message, it can be stopped at any point by inputting "OD." When OD is encountered in a message, no more words will be read and the words that normally would be used for the message can be used for other purposes.

Only the messages whose message number has been preset in the Programming Device will be displayed. If two or more MSG(195) instructions have the same message number, the earlier message will be replaced when another MSG(195) instruction with the same message number is executed later.

When a message instruction is executed, its corresponding Message Flag will be turned ON (bits A09900 to A09907 correspond to messages 0 to 7).

**Clearing Messages**

A message instruction can be cleared by executing the instruction with a constant (#0000 to #FFFF) entered for M.

**Precautions**

N must be BCD between 0000 and 0007.

**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

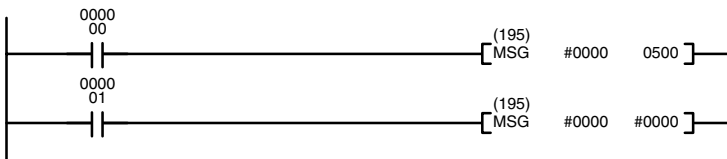
ER (A50003): N is not between 0 and 7.

Content of \*DM word is not BCD when set for BCD.

**Example**

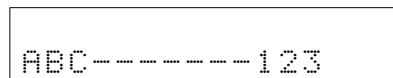
The following example shows the display that would be produced for the instruction and data given when CIO 000000 is ON. If CIO 000001 goes ON, message 0 will be cleared.

The display message number must be set to 0 in the Programming Device before executing the instruction.



Address	Instruction	Operands
00000	LD	000000
00001	MSG(195)	
		#0000
		0500
00002	LD	000001
00003	MSG(195)	
		#0000
		#0000

DM contents					ASCII equivalent	
0500	4	1	4	2	A	B
0501	4	3	4	4	C	D
0502	4	5	4	6	E	F
	.	.	.	.	.	.
	.	.	.	.	.	.
	.	.	.	.	.	.
0513	3	1	3	2	1	2
0514	3	3	0	D	3	(end)
0515	3	4	3	5	5	6



The last character displayed is just before "OD" in the message data.

### 5-34-4 NETWORK SEND: SEND(192)

<p><b>Ladder Symbol</b></p> <p><b>Variations</b></p> <p>↑ SEND(192)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: 1<sup>st</sup> source word</b> CIO, G, A, T, C, DM</p> <p><b>D: 1<sup>st</sup> destination word</b> CIO, G, A, T, C, DM</p> <p><b>C: 1<sup>st</sup> control word</b> CIO, G, A, T, C, DM</p>
---	--

**Description**

When the execution condition is OFF, SEND(192) is not executed. When the execution condition is ON, SEND(192) transfers data beginning at word S, to addresses beginning at D in the designated PC, BASIC Unit, Personal Computer Unit, or computer in the designated node on the designated SYSMAC NET Link, SYSMAC LINK, or Controller Link System.

The possible values for D depend on the destination Unit. Check the settings for the Unit. If D is in the EM Area, data will be transferred to the current EM bank in destination Unit.

The control words, beginning with C, specify the number of words to be sent, the destination node, and other parameters. Some control data parameters depend on whether a transmission is being sent through a SYSMAC NET Link, SYSMAC LINK, or Controller Link System.

SEND(192) only starts the transmission. Verify that the transmission has been completed with the Network Status Flags in A502.

- Note**
1. The node number for the SYSMAC LINK, SYSMAC NET Link, or Controller Link Unit corresponds to the unit number for the host link interface in the PC Setup.
  2. The execution condition for the up-differentiated variation (↑SEND(192)) might not be recognized when the mode has been switched from duplex mode to simplex mode.

**Control Data**

**SYSMAC NET Link Systems**

Set the destination node number to \$FF to send the data to all nodes in the designated network or to \$00 to send to a destination within the node of the PC executing the send. Refer to the *SYSMAC NET Link System Manual* for details.

Word	Bits 00 to 07	Bits 08 to 15
C	Number of words (1 to 0990 in 4-digit hexadecimal, i.e., \$0001 to \$03DE)	
C+1	Destination network address (0 to 127, i.e., \$00 to \$7F) <sup>1</sup>	Bits 08 to 11: (\$0 to \$F) <sup>2</sup> Bits 12 to 15: Set to 0.
C+2	Destination unit address <sup>3</sup>	Destination node number <sup>4</sup>
C+3	Bits 00 to 03: No. of retries (0 to 15 in hexadecimal, i.e., \$0 to \$F) Bits 04 to 07: Set to 0.	Bits 08 to 11: Transmission port number (\$0 to \$7) Bit 12 to 14: Set to 0. Bit 15: ON: No response. OFF: Response returned.
C+4	Response monitoring time ( \$0001 to \$FFFF = 0.1 to 6553.5 seconds) <sup>5</sup>	

- Note**
1. Set the destination network address to \$00 when transmitting within the local network. In this case, the network of the Link Unit with the lowest unit number will be selected if the PC belongs to more than one network.
  2. Set the BASIC Unit interrupt number when a BASIC Unit is designated.

3. Indicates a Unit as shown in the following table.

Unit	Setting
PC	00
SYSMAC NET Link, SYSMAC LINK, or Controller Link Unit	\$10 to \$1F: Unit numbers 0 to F \$FE: The local Unit
SYSMAC BUS/2 Master, BASIC Unit, or Personal Computer Unit	\$10 to \$1F: Unit numbers 0 to F
SYSMAC BUS/2 Group-2 Slave	\$90 to \$CF: Unit number+90+10×Master address

- Values of \$01 to \$7E indicate nodes 1 to 126. Set to \$FF to send to all nodes, \$00 to send data within the local PC.
- Designates the length of time that the PC retries transmission when bit 15 of C+3 is OFF and no response is received. The default value is \$0000, which indicates 2 seconds. The response function is not used when the destination node number is set to \$FF, broadcasting to all nodes in the network.
- Transmissions cannot be sent to the PC executing the send.

**SYSMAC LINK and Controller Link Systems**

Set the destination node number to \$FF to send the data to all nodes in the designated network or to \$00 to send to a destination within the node of the PC executing the send. Refer to the *SYSMAC LINK System Manual* or *Controller Link System Manual* for details.

Word	Bits 00 to 07	Bits 08 to 15
C	Number of words (1 to 256 in 4-digit hexadecimal, i.e., \$0001 to \$0100)	
C+1	Destination network address (0 to 127, i.e., \$00 to \$7F) <sup>1</sup>	Bits 08 to 11: (\$0 to \$F) <sup>2</sup> Bits 12 to 15: Set to 0.
C+2	Destination unit address <sup>3</sup>	Destination node number <sup>4</sup>
C+3	Bits 00 to 03: No. of retries (0 to 15 in hexadecimal, i.e., \$0 to \$F) Bits 04 to 07: Set to 0.	Bits 08 to 11: Transmission port number (\$0 to \$7) Bit 12 to 14: Set to 0. Bit 15: ON: No response. OFF: Response returned.
C+4	Response monitoring time ( \$0001 to \$FFFF = 0.1 to 6553.5 seconds) <sup>5</sup>	

- Note**
- Set the destination network address to \$00 when transmitting within the local network. In this case, the network of the Link Unit with the lowest unit number will be selected if the PC belongs to more than one network.
  - Set the BASIC Unit interrupt number when a BASIC Unit is designated.
  - Same as for SYSMAC NET Link Systems. See note 3 above.
  - Values of \$01 to \$3E indicate nodes 1 to 62. Set to \$FF to send to all nodes, \$00 to send data within the local PC. In a Controller Link System, the destination node number can be set to values of \$00 to \$20 or \$FF.
  - Designates the length of time that the PC retries transmission when bit 15 of C+3 is OFF and no response is received. The default value is \$0000, which indicates 2 seconds. The response function is not used when the destination node number is set to \$FF, broadcasting to all nodes in the network.
  - Transmissions cannot be sent to the PC executing the send.

**Precautions**

C through C+4 must be within the values specified above. To be able use of SEND(192), the PC must have a SYSMAC NET Link Unit, SYSMAC LINK Unit, or Controller Link Unit mounted.

Do not change the control data during a transmission (i.e., while the corresponding Port Enabled Flag in A502 is OFF).

**Note** Refer to page 101 for general precautions on operand data areas.

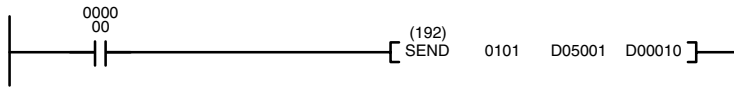


**Flags**

ER (A50003): Content of \*DM word is not BCD when set for BCD.

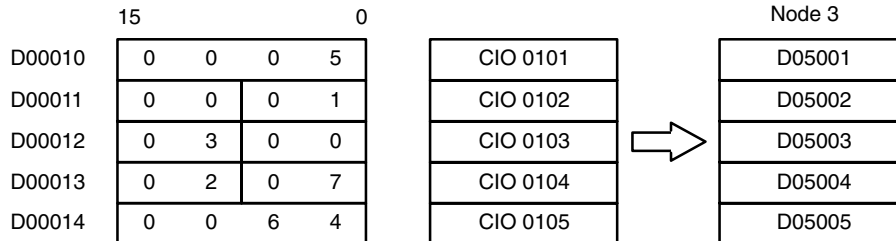
**Example**

The following example is for transmission to a PC through a SYSMAC NET Link System. When CIO 00000 is ON, the SEND(192) transfers the content of CIO 0100 through CIO 0104 to D05001 through D05005 of the PC on node 3 of network 1. The control words also specify that a response is required, use of port 2, 7 retries, and a 10-second response monitoring time.



Address	Instruction	Operands
00000	LD	000000
00001	SEND(192)	
		0101
		D05001
		D00010

**Control Words**



**5-34-5 NETWORK RECEIVE: RECV(193)**

<p><b>Ladder Symbol</b></p> <p>— [ (193) RECV S D C ]</p> <p><b>Variations</b></p> <p>↑ RECV(193)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: 1<sup>st</sup> source word</b> CIO, G, A, T, C, DM</p> <p><b>D: 1<sup>st</sup> destination word</b> CIO, G, A, T, C, DM</p> <p><b>C: 1<sup>st</sup> control word</b> CIO, G, A, T, C, DM</p>
---	--

**Description**

When the execution condition is OFF, RECV(193) is not executed. When the execution condition is ON, RECV(193) transfers data beginning at word S from the designated PC, BASIC Unit, Personal Computer Unit, or host computer in the designated node on the SYSMAC NET Link/SYSMAC LINK/Controller Link System to addresses beginning at D in the PC executing the instruction.

The possible values for S depend on the source Unit. Check the settings for the Unit. If S is in the EM Area, data will be transferred from the current EM bank in the source Unit.

The control words, beginning with C, specify the number of words to be received, the source node, and other parameters. Some control data parameters depend on whether a transmission is being received in a SYSMAC NET Link, SYSMAC LINK, or Controller Link System.

Normally a response is required with RECV(193), so set C+3 bit 15 to OFF.

RECV(193) only starts the transmission. Verify that the transmission has been completed with the Network Status Flags in A502.

- Note**
1. The node number for the SYSMAC LINK, SYSMAC NET Link, or Controller Link Unit corresponds to the unit number for the host link interface in the PC Setup.
  2. The execution condition for the up-differentiated variation (↑RECV(193)) might not be recognized when the mode has been switched from duplex mode to simplex mode.

**Control Data**

**SYSMAC NET Link Systems** Set the source node number to \$00 to send data within the PC executing the instruction. Refer to the *SYSMAC NET Link System Manual* for details.

Word	Bits 00 to 07	Bits 08 to 15
C	Number of words (1 to 0990 in 4-digit hexadecimal, i.e., \$0001 to \$03DE)	
C+1	Source network address (0 to 127, i.e., \$00 to \$7F) <sup>1</sup>	Bits 08 to 11: (\$0 to \$F) <sup>2</sup> Bits 12 to 15: Set to 0.
C+2	Source unit address <sup>3</sup>	Source node number <sup>4</sup>
C+3	Bits 00 to 03: No. of retries (0 to 15 in hexadecimal, i.e., \$0 to \$F) Bits 04 to 07: Set to 0.	Bits 08 to 11: Transmission port number (\$0 to \$7) Bit 12 to 14: Set to 0. Bit 15: ON: No response. OFF: Response returned.
C+4	Response monitoring time ( \$0001 to \$FFFF = 0.1 to 6553.5 seconds) <sup>5</sup>	

- Note**
1. Set the source network address to \$00 when transmitting within the same network. In this case, the network of the Link Unit with the lowest unit number will be selected if the PC belongs to more than one network.
  2. Set the BASIC Unit interrupt number when a BASIC Unit is designated.
  3. Indicates a Unit as shown in the following table.

Unit	Setting
PC	00
SYSMAC NET Link, SYSMAC LINK, or Controller Link Unit	\$10 to \$1F: Unit numbers 0 to F \$FE: The local Unit
SYSMAC BUS/2 Master, BASIC Unit, or Personal Computer Unit	\$10 to \$1F: Unit numbers 0 to F
SYSMAC BUS/2 Group-2 Slave	\$90 to \$CF: Unit number+\$90+\$10×Master address

4. Values of \$01 to \$7E indicate nodes 1 to 126. Set to \$00 to receive data from within the local PC.
5. Designates the length of time that the PC retries transmission when bit 15 of C+3 is OFF and no response is received. The default value is \$0000, which indicates 2 seconds.
6. Transmissions cannot be received from the PC executing RECV(193).

**SYSMAC LINK and Controller Link Systems**

Set the source node number to \$00 to receive from a source within the node of the PC executing the instruction. Refer to the *SYSMAC LINK System Manual* or *Controller Link System Manual* for details.

Word	Bits 00 to 07	Bits 08 to 15
C	Number of words (1 to 256 in 4-digit hexadecimal, i.e., \$0001 to \$0100)	
C+1	Source network address (0 to 127, i.e., \$00 to \$7F) <sup>1</sup>	Bits 08 to 11: (\$0 to \$F) <sup>2</sup> Bits 12 to 15: Set to 0.
C+2	Source unit address <sup>3</sup>	Source node number <sup>4</sup>
C+3	Bits 00 to 03: No. of retries (0 to 15 in hexadecimal, i.e., \$0 to \$F) Bits 04 to 07: Set to 0.	Bits 08 to 11: Transmission port number (\$0 to \$7) Bit 12 to 14: Set to 0. Bit 15: ON: No response. OFF: Response returned.
C+4	Response monitoring time ( \$0001 to \$FFFF = 0.1 to 6553.5 seconds) <sup>5</sup>	

- Note**
1. Set the source network address to \$00 when transmitting within the same network. In this case, the network of the Link Unit with the lowest unit number will be selected if the PC belongs to more than one network.

2. Set the BASIC Unit interrupt number when a BASIC Unit is designated.
3. Same as for SYSMAC NET Link Systems. See note 3 above.
4. Values of \$01 to \$3E indicate nodes 1 to 62. Set to \$00 to receive data from within the local PC. In a Controller Link System, the destination node number can be set to values of \$00 to \$20 or \$FF.
5. Designates the length of time that the PC retries transmission when bit 15 of C+3 is OFF and no response is received. The default value is \$0000, which indicates 2 seconds.
6. Transmissions cannot be received from the PC executing RECV(193).

**Precautions**

C through C+4 must be within the values specified above. To be able use of RECV(193), the PC must have a SYSMAC NET Link, SYSMAC LINK, or Controller Link Unit mounted.

Do not change the control data during a transmission (i.e., while the corresponding Port Enabled Flag in A502 is OFF).

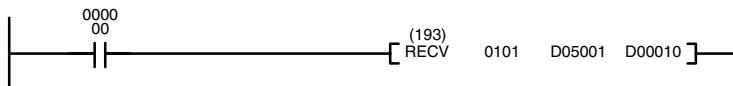
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

ER (A50003): Content of \*DM word is not BCD when set for BCD.

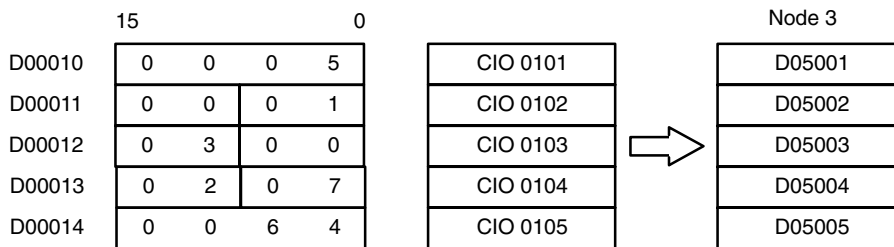
**Example**

The following example is for receiving from a PC through a SYSMAC NET Link System. When CIO 000000 is ON, the RECV(193) transfers the content of CIO 0101 through CIO 0105 of the PC on node 3 of network 1, to D05001 through D05005 of the PC executing RECV(193). The control words also specify that a response is required, use of port 2, 7 retries, and a 10-second response monitoring time.



Address	Instruction	Operands
00000	LD	000000
00001	RECV(193)	
		0101
		D05001
		D00010

**Control Words**



**5-34-6 DELIVER COMMAND: CMND(194)**

<p><b>Ladder Symbol</b></p> <p>— [ <sup>(194)</sup> CMND S D C ]</p> <p><b>Variations</b></p> <p>↑ CMND(194)</p>	<p><b>Operand Data Areas</b></p> <p><b>S: 1<sup>st</sup> command word</b> CIO, G, A, T, C, DM</p> <p><b>D: 1<sup>st</sup> response word</b> CIO, G, A, T, C, DM</p> <p><b>C: 1<sup>st</sup> control word</b> CIO, G, A, T, C, DM</p>
--	--

**Description**

When the execution condition is OFF, CMND(194) is not executed. When the execution condition is ON, CMND(194) transmits the command beginning at word S to the designated Unit at the destination node number in the designated network of the SYSMAC NET Link/SYSMAC LINK/Controller Link System, and receives the response beginning at word D.

If the destination node number is \$FF, the command will be broadcast to all nodes in the designated network. Normally a response is required with CMND(194) and C+3 bit 15 is turned OFF. The response function is disabled when the command is sent to all nodes.

Any of the CV-mode (FINS) commands supported by the CV-series PCs can be sent with certain changes (see below). Refer to the *CV-series PC Operation Manual: Host Interface* for details on the CV-mode commands.

The following table shows differences between the CV-mode commands used through the Host Link System and those used with CMND(194).

Host Link FINS command	CMND(194)
ASCII	Binary
The node number, header, FCS, and terminator are required in the command frame. (See note 1.)	Only the command part is required. The header code, FCS, etc. are not required.

- Note**
1. The node number for the SYSMAC LINK, SYSMAC NET Link, or Controller Link Unit corresponds to the unit number for the host link interface in the PC Setup.
  2. The execution condition for the up-differentiated variation (↑CMND(194)) might not be recognized when the mode has been switched from duplex mode to simplex mode.

**Control Data**

The control words, beginning with C, specify the number of bytes of control data to be sent, the number of bytes of response data to be received, the destination node, and other parameters. Some control data parameters depend on whether a transmission is being received in a SYSMAC NET Link, SYSMAC LINK, or Controller Link System.

Word	Bits 00 to 07	Bits 08 to 15
C	Number of bytes to send (0 to 1990, i.e., \$0000 to \$07C6) <sup>1</sup>	
C+1	Number of bytes to receive (0 to 1990, i.e., \$0000 to \$07C6) <sup>1</sup>	
C+2	Destination network address (0 to 127, i.e., \$00 to \$7F) <sup>2</sup>	Bits 08 to 11: (\$0 to \$F) <sup>3</sup> Bits 12 to 15: Set to 0.
C+3	Destination unit address <sup>4</sup>	Destination node number <sup>5</sup>
C+4	Bits 00 to 03: No. of retries (0 to 15 in hexadecimal, i.e., \$0 to \$F) Bits 04 to 07: Set to 0.	Bits 08 to 11: Transmission port number (\$0 to \$7) Bit 12 to 14: Set to 0. Bit 15: ON: No response. OFF: Response returned.
C+5	Response monitoring time ( \$0001 to \$FFFF = 0.1 to 6553.5 seconds) <sup>6</sup>	

- Note**
1. Maximum number of bytes that can be sent or received:

System	Max. number of bytes
SYSMAC NET Link	\$07C6 (1,990)
Controller Link	
SYSMAC LINK	\$021E (542)
SYSMAC BUS/2	

2. Set the destination network address to \$00 when transmitting within the same network.
3. Set the BASIC Unit interrupt number when a BASIC Unit is designated.

4. Indicates a Unit as shown in the following table.

Unit	Setting
PC	\$00
SYSMAC NET Link, SYSMAC LINK, or Controller Link Unit	\$10 to \$1F: Unit numbers 0 to F \$FE: The local Unit
SYSMAC BUS/2 Master, BASIC Unit, or Personal Computer Unit	\$10 to \$1F: Unit numbers 0 to F
SYSMAC BUS/2 Group 2 Slave	\$90 to \$CF: Unit number+90+10×Master address

5. The destination node number can have the following values:

System	Possible values
SYSMAC NET Link System	\$01 to \$7E (nodes 1 to 126), \$00, or \$FF
SYSMAC LINK System	\$01 to \$3E (nodes 1 to 62), \$00, or \$FF
Controller Link System	\$01 to \$20 (nodes 1 to 32), \$00, or \$FF

Set a value of \$00 to transmit within the local PC (to/from CPU Bus Units).  
Set a value of \$FF to broadcast to all nodes in the network.

- 6. Designates the length of time that the PC retries transmission when bit 15 of C+3 is OFF and no response is received. The default value is \$0000, which indicates 2 seconds.
- 7. Transmissions cannot be sent to the PC executing CMND(194) (i.e., the Unit cannot be specified as \$00 and the destination node number set to \$00).

**Precautions**

C through C+5 must be within the values specified below. To be able use of CMND(194), the PC must have a SYSMAC NET Link, SYSMAC LINK Unit, Controller Link Unit, or SYSMAC BUS/2 Remote I/O Master Unit mounted.

Do not change the control data during a transmission (i.e., while the corresponding Port Enabled Flag in A502 is OFF).

The Execute Error Flag for the designated port will be turned ON if no response is received in the response monitoring time (C+5), the amount of command data transmitted exceeds the maximum for the system, or the response data exceeds the number of bytes specified in C+1.

The amount of response data can be less than the “number of bytes to receive” in C+1 without causing an error, but the amount of the command data must agree with the “number of bytes to send” in C or a command length error will occur. Be sure that the command data and response data do not go over the end of a data area.

CMND(194) only starts the transmission of command data. Verify that the transmission has been completed with the Network Status Flags in A502. If data is changed before transmission is completed, the data might be transmitted while data is being changed.

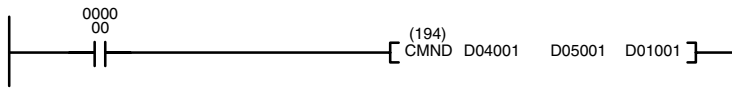
**Note** Refer to page 101 for general precautions on operand data areas.

**Flags**

ER (A50003): Content of \*DM word is not BCD when set for BCD.

**Example**

This example shows CMND(194) used to transmit a command to the PC on node 3 of network 1 to change the PC to MONITOR mode when CIO 00000 is ON. D05001 is the first word to receive the response, and D04001 through D04003 contain the command data.



Address	Instruction	Operands
00000	LD	000000
00001	CMND(194)	
		D04001
		D05001
		D01001

Word	Content
D04001	0401
D04002	0000
D04003	0002

D01001 to D01006 contain the control data, as shown below.

Word	Content	Function
D01001	0006	Number of bytes to send = 6
D01002	0100	Number of bytes to receive = 256
D01003	0001	Destination network address = 01
D01004	0300	Destination node number = 03; unit = PC
D01005	0207	Response returned; transmission port = 2; retries = 7
D01006	0064	Response monitoring time = 10 seconds

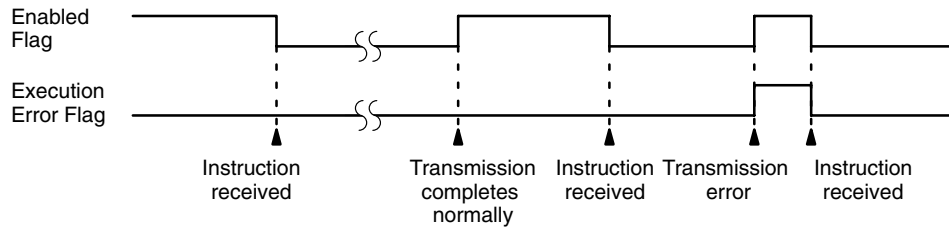
### 5-34-7 About Network Instruction Operations

SEND(192), RECV(193), and CMND(194) are based on command/response processing. That is, the transmission is not complete until the sending node receives and acknowledges a response from the destination node, unless the response function is disabled in the control word or data is being broadcast to all nodes in a network. Refer to the *SYSMAC NET Link System Manual*, *SYSMAC LINK System Manual*, or *Controller Link System Manual* for details about command/response operations.

If more than one network instruction (SEND(192)/RECV(193)/CMND(194)) is used through one port, the following flags must be used to ensure that any previous operation has completed before attempting further network instructions.

Flag	Functions
Port #0 to #7 Enabled Flags (A50200 to A50207)	Enabled Flags A50200 to A50207 are OFF during network instruction execution for ports #0 to #7, respectively. Do not start a network instruction for a port unless the corresponding Enabled Flag is ON.
Port #0 to #7 Execution Error Flags (A50208 to A50215)	OFF following normal completion of a network instruction (i.e., after reception of response signal) ON after an unsuccessful network instruction attempt. Error status is maintained until the next network instruction.  Error types: Time-out error (command/response time greater than the response monitoring time set in the control words) Transmission data errors
Port #0 to #7 Completion Codes (A503 to A510)	A503 to A510 contain the completion codes for errors that occur during network instruction execution for ports #0 to #7, respectively. The code for normal completion of a network instruction is 0000. Refer to the <i>SYSMAC NET Link System Manual</i> , <i>SYSMAC LINK System Manual</i> , or <i>Controller Link System Manual</i> for details on error codes.

**Timing**



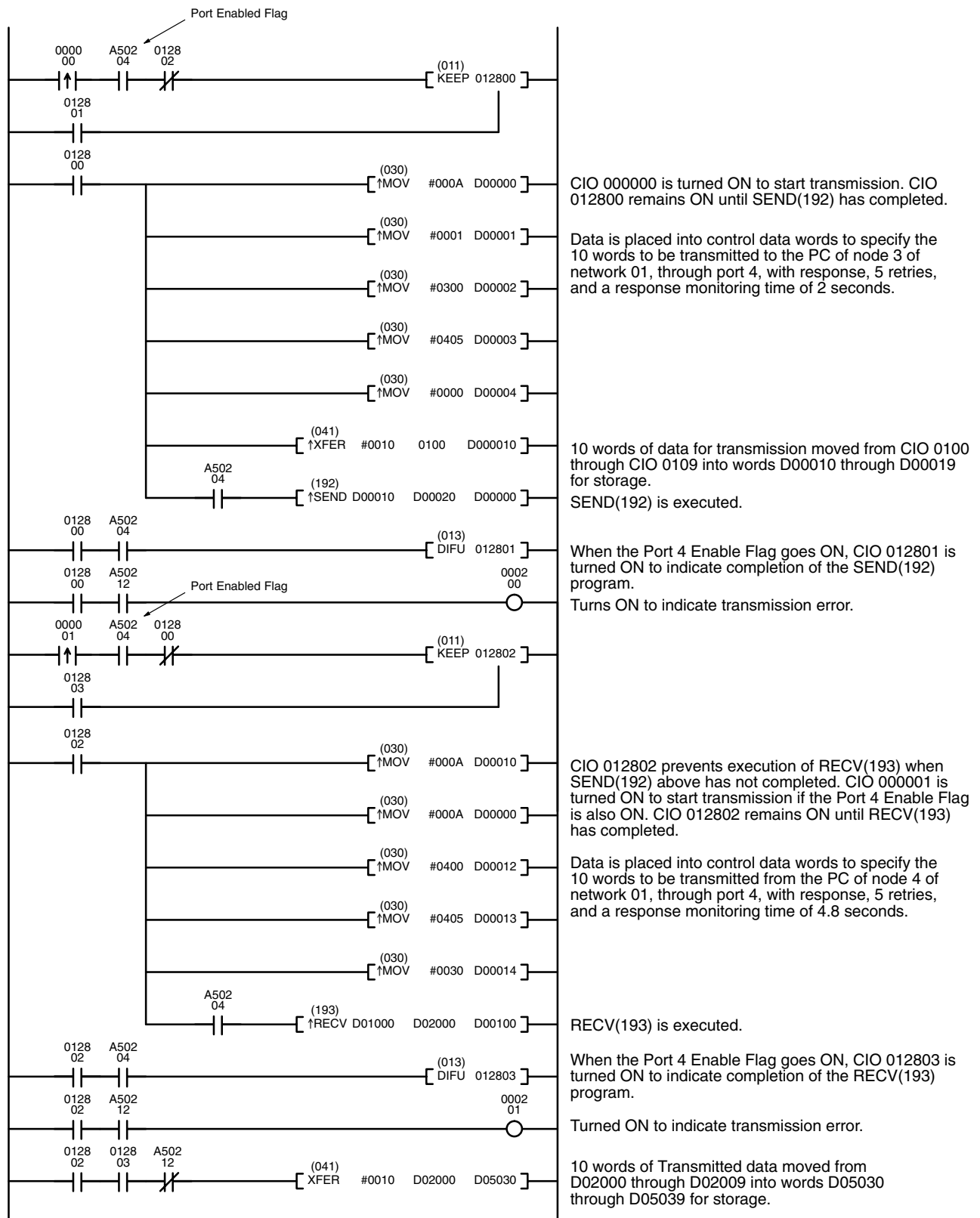
**Data Processing for Network Instructions**

Data is transmitted on a network when SEND(192), RECV(193), or CMND(194) is executed. Final processing for transmissions/receptions is performed during servicing of Link Units.

The CVM1D PCs always operate in synchronous operation, so it isn't necessary to take steps to synchronize data processing as in the CVM1 and CV-series PCs.

**Programming Example: Multiple Instructions**

To ensure successful SEND(192)/RECV(193) operations with more than one instruction for a single port, your program must use the Enabled Flag and Error Flag to confirm that execution is possible. The following program shows one example of how to do this for port 4 in a SYSMAC NET Link System. The program is effective when both bit 000000 and the Port 4 Enable Flag (A50204) are ON.



CIO 000000 is turned ON to start transmission. CIO 012800 remains ON until SEND(192) has completed.

Data is placed into control data words to specify the 10 words to be transmitted to the PC of node 3 of network 01, through port 4, with response, 5 retries, and a response monitoring time of 2 seconds.

10 words of data for transmission moved from CIO 0100 through CIO 0109 into words D00010 through D00019 for storage.  
SEND(192) is executed.

When the Port 4 Enable Flag goes ON, CIO 012801 is turned ON to indicate completion of the SEND(192) program.

Turns ON to indicate transmission error.

CIO 012802 prevents execution of RCV(193) when SEND(192) above has not completed. CIO 000001 is turned ON to start transmission if the Port 4 Enable Flag is also ON. CIO 012802 remains ON until RCV(193) has completed.

Data is placed into control data words to specify the 10 words to be transmitted from the PC of node 4 of network 01, through port 4, with response, 5 retries, and a response monitoring time of 4.8 seconds.

RCV(193) is executed.

When the Port 4 Enable Flag goes ON, CIO 012803 is turned ON to indicate completion of the RCV(193) program.

Turned ON to indicate transmission error.

10 words of Transmitted data moved from D02000 through D02009 into words D05030 through D05039 for storage.



Address	Instruction	Operands
00000	↑LD	000000
00001	AND	A50204
00002	AND NOT	012802
00003	LD	012801
00004	KEEP(011)	012800
00005	LD	012800
00006	↑MOV(030)	
		#000A
		D00000
00007	↑MOV(030)	
		#0001
		D00001
00008	↑MOV(030)	
		#0300
		D00002
00009	↑MOV(030)	
		#0405
		D00003
00010	↑MOV(030)	
		#0000
		D00004
00011	↑XFER(041)	
		#0010
		0100
		D00010
00012	↑SEND(192)	
		D00010
		D00020
		D00000
00013	LD	012800
00014	AND	A50204
00015	DIFU(13)	012801
00016	LD	012800
00017	AND	A50204
00018	OUT	000200
00019	↑LD	000001

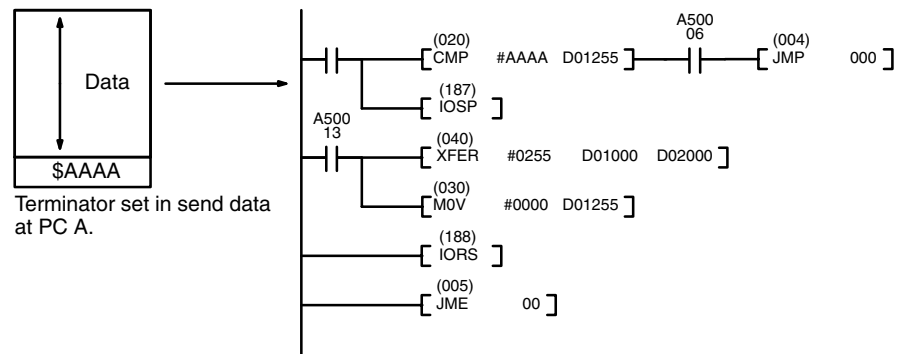
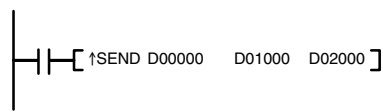
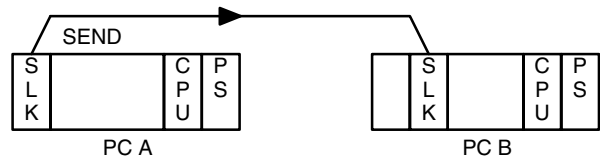
Address	Instruction	Operands
00020	AND	A50204
00021	AND NOT	012800
00022	LD	012803
00023	KEEP(011)	012802
00024	LD	012802
00025	↑MOV(030)	
		#000A
		D00010
00026	↑MOV(030)	
		#000A
		D00000
00027	↑MOV(030)	
		#0400
		D00012
00028	↑MOV(030)	
		#0405
		D00013
00029	↑MOV(030)	
		#0030
		D00014
00030	↑RECV(193)	
		D01000
		D02000
		D00100
00031	LD	012802
00032	AND	A50204
00033	DIFU(13)	012803
00034	LD	012802
00035	AND	A50204
00036	OUT	000201
00037	LD	012802
00038	AND	012803
00039	AND NOT	A50212
00040	XFER(041)	
		#0010
		D02000
		D05030

**Programming Example:  
Synchronizing Data**

The following program shows how to synchronize data transmission during asynchronous operation using IOSP(187) and IORS(188).

In the program in PC A (the sending PC), the data is set in memory while the Enabled Flag is ON, i.e., when SEND(192) is not being executed, and a code is added in the last word of data to verify that the data has been transmitted successfully.

In the program in PC B (the receiving PC), the code in the last word of the transmitted data is used to prevent more data from being transferred until the transmitted data is copied to another section of Data Memory for storage. The code is erased from the original block of data after it is copied.



## 5-35 Block Programming Instructions

Block programming can be used to program operations that are difficult to program with ladder diagrams, such as certain data computations. Effective block programming can be used to reduce the number of programming steps required for certain operations, thus reducing the cycle time and increasing overall processing speed.

A maximum of 100 block programs can be used.

### 5-35-1 Overview

#### Instructions

The following block programming instructions are available.

Name	Mnemonic	Function
BLOCK PROGRAM BEGIN	BPRG(250)	Changes to block programming (BPRG(250) is actually a ladder diagram instruction.)
BLOCK PROGRAM END	BEND<001>	Ends a block program.
IF (NOT)	IF<002> (NOT)	Starts conditional branching.
ELSE	ELSE<003>	Marks the alternate route for conditional branching.
IF END	IEND<004>	Ends conditional branching.
ONE CYCLE AND WAIT (NOT)	WAIT<005> (NOT)	Creates a conditional one-cycle wait.
CONDITIONAL BLOCK EXIT (NOT)	EXIT <006> (NOT)	Conditionally ends block program execution.
LOOP	LOOP<009>	Starts a loop.
LOOP END (NOT)	LEND<010>(NOT)	Ends a loop.
BLOCK PROGRAM PAUSE	BPPS<011>	Temporarily stops block program execution.
BLOCK PROGRAM RESTART	BPRS<012>	Restarts block program execution.
TIMER WAIT	TIMW<013>	Creates a timed wait.
COUNTER WAIT	CNTW<014>	Creates a wait based on a counter.
HIGH-SPEED TIMER WAIT	TMHW<015>	Creates a high-speed timed wait.

#### Restricted Instructions

Although most ladder-diagram instructions can also be used in mnemonic form within a block program, there are restrictions for the following instructions.

LD, LD NOT, AND, AND NOT, AND LD, OR LD, UP(018), DOWN(19), EQU(025), symbol comparison instructions (300 to 328), TST(350), and TSTN(351).

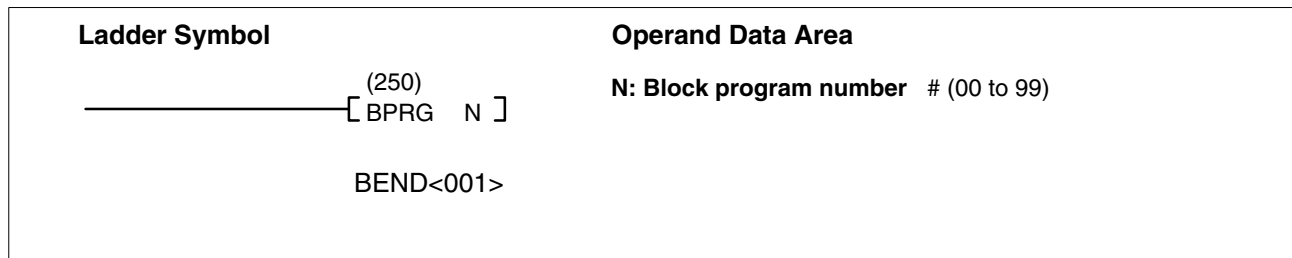
The instructions must be combined with either the IF<002>, WAIT<005>, EXIT<006>, or LEND<010> instructions, or with ladder-diagram jump instructions.

**Note** The operation of the above instructions will not be dependable if they are not used in combination as described or if they are used in combination with other instructions.

The following instructions cannot be used in block programs.

Group	Mnemonic	Remarks
Bit control instructions	DIFU(013)	---
	DIFD(014)	
	KEEP(011)	
	OUT	Use SET(016) and RSET(017). (There are not block SET and RSET instructions for CV-series PCs.)
	OUT NOT	
Interlock and jump instructions	IL(002)	---
	ILC(003)	
	JMP(004) 0000	These instruction can be used if the PC Setup is set so that multiple jump with jump number 0000 cannot be used.
	CJP(221) 0000	
	CJPN(222) 0000	
	JME(005) 0000	
END	END(001)	Use BEND<001>.
Timer and counter instructions	TIM	Use the block programming timer and counter instructions.
	CNT	
	TIMH(015)	
	TTIM(120)	---
	TIML(121)	
	MTIM(122)	
	CNTR(012)	
Step instructions	STEP(008)	---
	SNXT(009)	
Shift instructions	SFT(050)	---
Subroutine instructions	SBN(150)	---
	RET(152)	
Diagnostic instructions	FPD(177)	---
PID and related instructions	PID(270)	---
Special I/O Unit instructions	RD2(280)	---
	WR2(281)	
Differentiated instructions	---	No input differentiated instructions (†) can be used in block programs.

### 5-35-2 BLOCK PROGRAM BEGIN/END: BPRG(250) / BEND<001>



**Description**

BPRG(250) is used to switch to block programming and BEND<001> is used to switch back to ladder-diagram programming. For every BPRG(250) there must be a corresponding BEND<001>. The corresponding block program will be executed with the execution condition for BPRG(250) is ON and it will be skipped (not executed) when the execution condition is OFF.

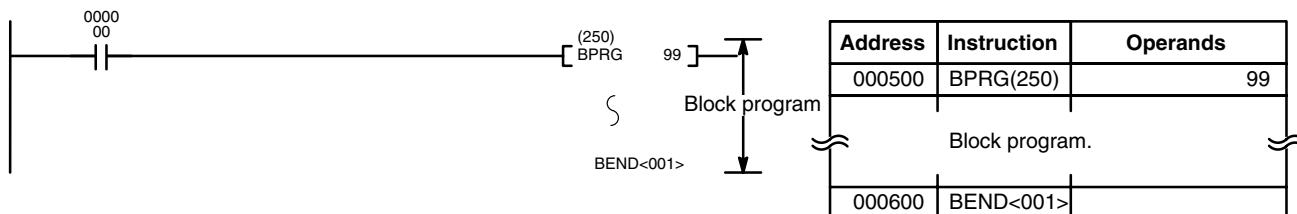
**Precautions**

The same block number cannot be used more than once.

Block programs cannot be nested.

**Example**

When CIO 000000 is ON in the following diagram, the block program between program addresses 000501 and 000600 will be executed.



**5-35-3 Branching—IF<002>, ELSE<003>, and IEND<004>**

Ladder Symbol	Operand Data Area
IF<002>	B
IF<002> NOT	B
ELSE<003>	
IEND<004>	

**B: Bit** CIO, G, A, T, C

**Description**

Branching instructions are used to branch according to either the current execution condition or the status of a designated bit. IF<002> and IF<002> NOT must be used in combination with IEND<004>. ELSE<003> may be used in between them, but is optional.

Branching is initiated with any of the following: IF<002> with a bit operand, IF<002> without a bit operand, or IF<002> NOT with a bit operand.

If the IF condition is YES, the instructions immediately following the IF<002> or IF<002> NOT will be executed. A YES execution condition is produced by an ON bit or ON execution condition for IF<002> or an OFF bit for IF<002>NOT.

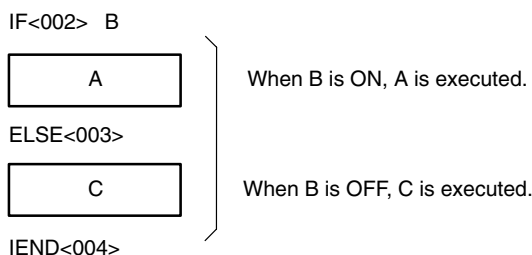
If ELSE<003> is encountered following IF<002> or IF<002>NOT, execution will jump to IEND<003> without executing any instruction in between. If ELSE<003> is not encountered, execution will continue as normal.

If the IF condition is NO, execution will jump to ELSE<003> or to IEND<004>, whichever appears first after the IF<002> or IF<002> NOT.

LD, possible in combination with AND or OR, must be used to establish the execution condition for IF<002> without an operand or IF<002> NOT without an operand.

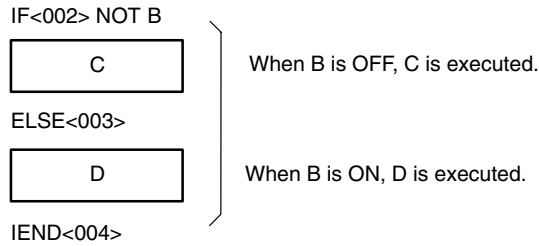
**Execution Flow Examples**

**IF<002> with an Operand** IF<002> to ELSE to IEND



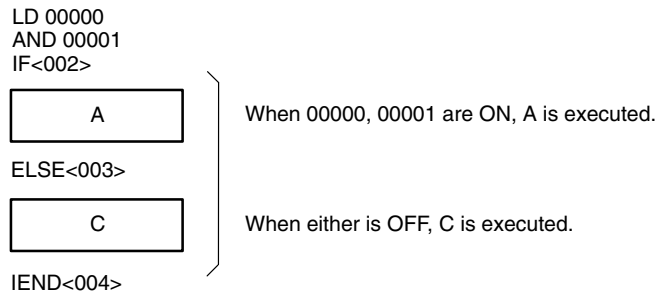
**IF<002> NOT with an Operand**

IF<002> NOT to ELSE to IEND



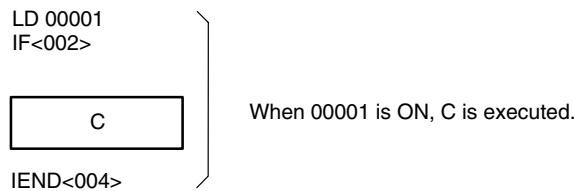
**IF<002> without an Operand**

IF<002> to ELSE to IEND



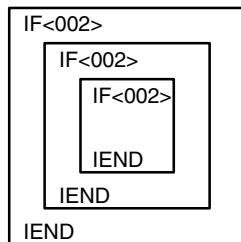
**IF<002> without ELSE**

IF<002> to IEND



**Nesting**

IF<002> blocks can be nested up to a maximum of 253 levels. Each IF<002> or IF<002> NOT will be effective through the next ELSE<003> and/or IEND<004>.



**Flags**

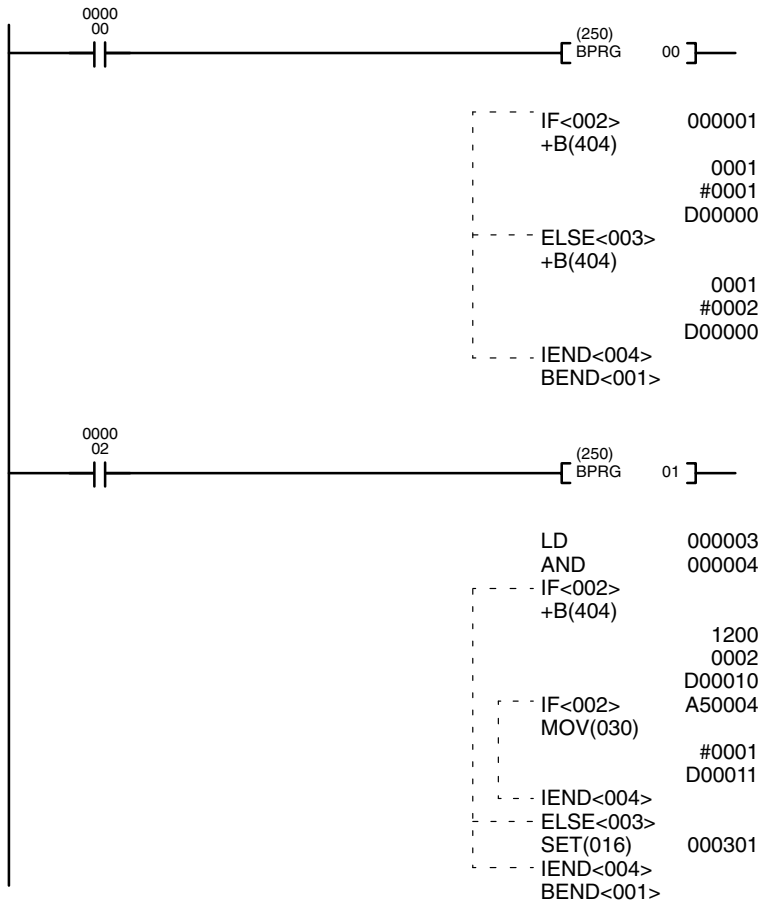
No flags are affected by these instructions.

**Example**

The following example shows two different block programs controlled by CIO 000000 and CIO 000002.

The first block executes one of two additions depending on the status of CIO 000001. This block is executed when CIO 000000 is ON. If CIO 000001 is ON, 0001 is added to the contents of CIO 0001. If CIO 000001 is OFF, 0002 is added to the contents of CIO 0001. In either case the result is placed in D00000.

The second block is executed when CIO 000002 is ON and shows nesting two levels. If CIO 000003 and CIO 000004 are both ON, the contents of CIO 1200 and CIO 0002 are added and the result is placed in D00010 and then 0001 is moved into D00011 based on the status of CY. If either CIO 000003 or CIO 000004 is OFF, then the entire addition operation is skipped and CIO 000301 is turned ON.



Address	Instruction	Operands
00000	LD	000000
00001	BPRG(250)	00
00002	IF<002>	000001
00003	+B(404)	
		0001
		#0001
		D00000
00004	ELSE<003>	
00005	+B(404)	
		0001
		#0002
		D00000
00006	IEND<004>	
00007	BEND<001>	
00008	LD	000002
00009	BPRG(250)	01
00010	LD	000003
00011	AND	000004
00012	IF<002>	
00013	+B(404)	
		1200
		0002
		D00010
		A50004
00014	IF<002>	A50004
00015	MOV(030)	
		#0001
		D00011
00016	IEND<004>	
00017	ELSE<003>	
00018	SET(016)	000301
00019	IEND<004>	
00020	BEND<001>	

### 5-35-4 ONE CYCLE AND WAIT: WAIT<005>

Ladder Symbol	Operand Data Area	
WAIT<005>	<b>B: Bit</b>	CIO, G, A, T, C
WAIT<005>	B	
WAIT<005> NOT	B	

#### Description

WAIT<005> and WAIT<005> NOT allow you to inhibit execution of the portion of block program from WAIT<005> to BEND<001> until B turns ON or if a bit is not specified, until the execution condition turns ON.

As long as the execution condition or operand bit of WAIT<005> is ON, or the operand bit of WAIT<005> NOT is OFF, the block program will be executed as normal. If the execution condition or operand bit of WAIT<005> is OFF or the operand bit of WAIT<005> NOT is ON, only the part of the block program up to the WAIT<005> or WAIT<005> NOT instruction will be executed during the first cycle. During following cycles, none of the block program will be executed until the operand bit or execution condition changes, at which point the remainder of the block program will be executed. Once the entire block program has been executed, the process is repeated.

#### Precautions

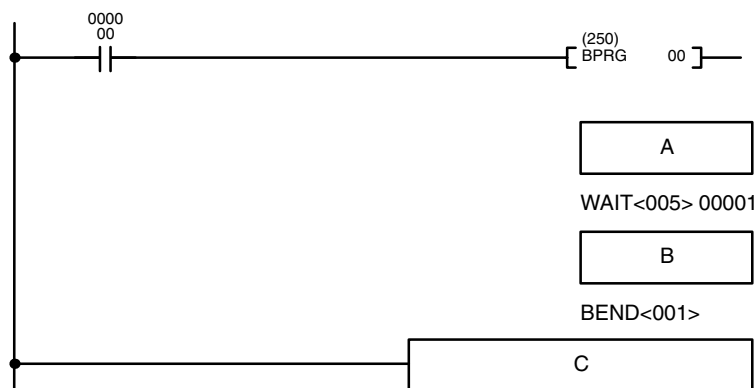
WAIT<005> NOT cannot be used without an operand bit.

If programs are edited online from a Programming Device, the wait status that is normally saved until the wait condition goes ON will be cleared and the block program will be executed from the beginning again.

When using WAIT<005> without an bit operand, the instructions used to create the execution condition for WAIT<05> must begin with LD.

#### Execution Flow Examples

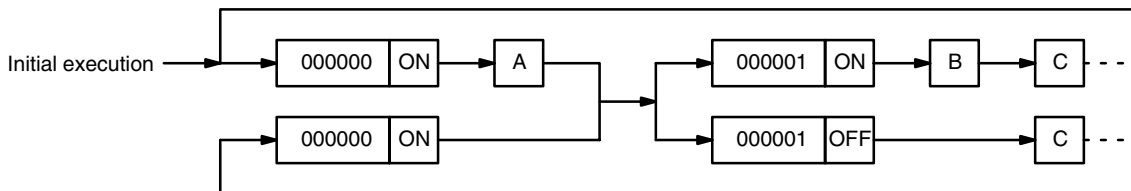
When CIO 000000 is ON, the block program is executed as normal. If CIO 000001 is OFF, however, A is executed and then B is skipped and program control jumps to BEND<001>. During the following cycles, no instructions within block 00 will be executed (except WAIT <005>) until CIO 000001 turns ON, at which point B will be executed.



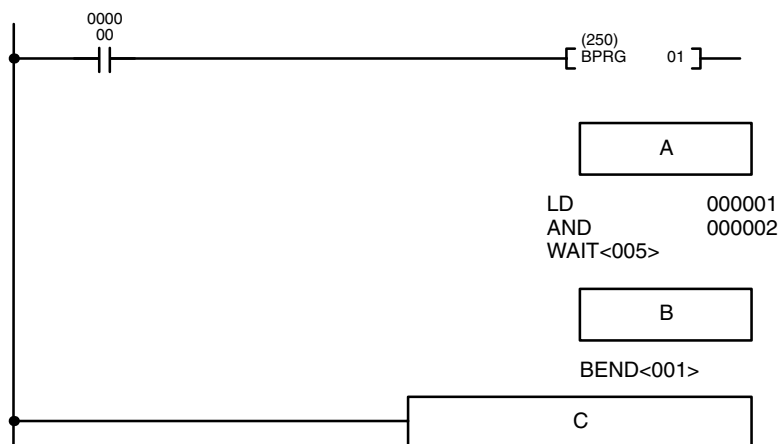
Address	Instruction	Operands
000000	LD	000000
000001	BPRG(250)	00
A		
000100	WAIT<005>	000001
B		
000200	BEND<001>	
C		



The execution flow for this example would be as shown below:



The following example would work similarly, except that execution of WAIT<005> would be based on an AND between the status of CIO 000001 and CIO 000002.



Address	Instruction	Operands
000000	LD	000000
000001	BPRG(250)	01
A		
000200	LD	000001
000201	AND	000002
000202	WAIT<005>	
B		
000300	BEND<001>	
C		

### 5-35-5 CONDITIONAL BLOCK EXIT: EXIT<006>

Ladder Symbol	Operand Data Area	
EXIT<006>	<b>B: Bit</b>	CIO, G, A, T, C
EXIT<006>	B	
EXIT<006> NOT	B	

**Description**

EXIT<006> and EXIT<006> NOT allow you to skip the portion of block program from EXIT<006> to BEND<001> while B is ON or if a bit is not specified, while the execution condition is ON.

As long of the execution condition or operand bit of EXIT<006> is OFF, or the operand bit of EXIT<006> NOT is ON, the block program will be executed as normal. If the execution condition or operand bit of EXIT<006> is ON or the operand bit of EXIT<006> NOT is OFF, only the part of the block program up to the EXIT<006> or EXIT<006> NOT instruction will be executed and the rest of the block program through BEND<001> will be skipped. If a bit is not programmed for EXIT<006>, then the same operation will occur, but it will be based on the status of the execution condition for EXIT<006>.

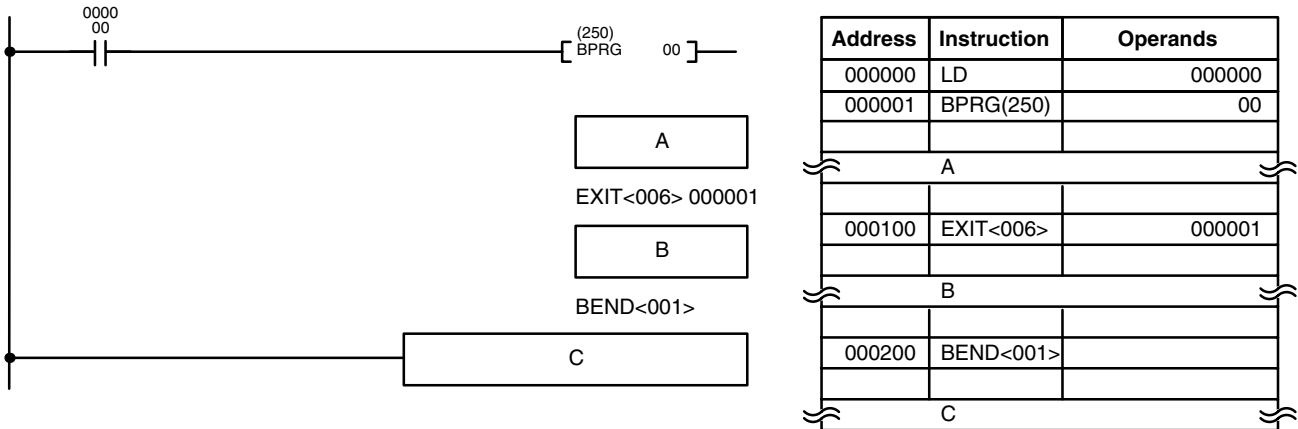
**Precautions**

EXIT<006> NOT cannot be used without an operand bit.

When using EXIT<006> without an bit operand, the instructions used to create the execution condition for EXIT<006> must begin with LD.

**Execution Flow Examples**

When CIO 000000 is OFF, the block program is executed as normal. If CIO 000001 turns ON, however, A is executed and then B is skipped and program control jumps to BEND<001>. Section B of the program will continue to be skipped until CIO 000001 turns OFF again.



**5-35-6 Loop Control—LOOP<009>/LEND<010>**

Ladder Symbol	Operand Data Area	
LOOP<009>	<b>B: Bit</b>	CIO, G, A, T/C
LEND<010>		
LEND<010>	B	
LEND<010> NOT	B	

**Description**

LOOP<009> and LEND<010> are used to create a loop that is repeatedly executed until the LOOP END condition becomes YES. LOOP<009> designates the beginning of the loop program, and a LEND<010> or LEND<010> NOT instruction specifies the end of the loop. When LEND<010> or LEND<010> NOT is reached, program execution will loop back to the next previous LOOP<009> an exit condition is attained.

A YES LOOP END condition is produced by an ON execution condition for LEND<010> without an operand, by an ON bit for LEND<010> with an operand bit, or by an OFF bit for LEND<010> NOT with an operand bit.

LD, possibly in combination with AND or OR, must be used to create an execution condition for LEND<010> when used without an operand bit.

**Note** Execution inside a loop does not refresh I/O data. If I/O data must be refreshed during the loop, use IORF(184).

**Precautions**

- Conditional block branching can be used within a loop, but the entire branch operation must be within the loop.

**Correct:**

LOOP<009>  
IF<002>  
IF<002>

**Incorrect:**

LOOP<009>  
IF<002>  
IF<002>

IEND<004> IEND<004>  
 IEND<004> LEND<010>  
 LEND<010>

- Loops cannot be nested within loops.

**Incorrect:**

LOOP<009>  
 LOOP<009>  
 LEND<010>  
 LEND<010>

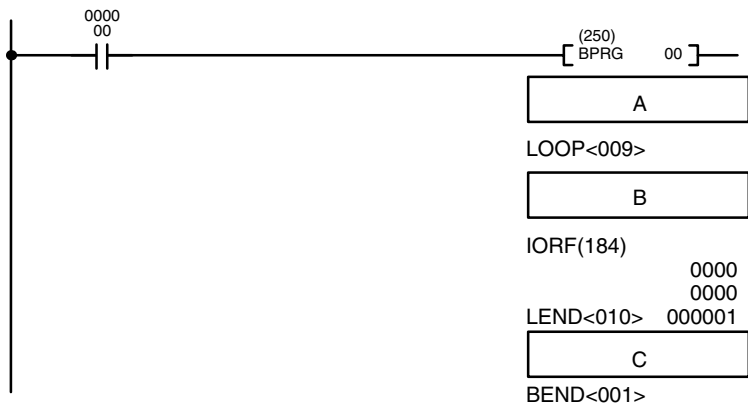
- Do not reverse the order of LOOP and LEND.

**Incorrect:**

LEND<010>  
 :  
 :  
 LOOP<009>

**Execution Flow Examples**

When CIO 000000 is ON, the block program is executed. After A is executed, B and the IORF(184) after it will be executed repeatedly until CIO 000001 is ON, at which time C will be executed and the block program will end.



Address	Instruction	Operands
00000	LD	000000
00001	BPRG(250)	00
A		
00015	LOOP<009>	
B		
00024	IORF(184)	
		0000
		0000
00025	LEND<010>	000001
C		
00033	BEND<001>	

**5-35-7 BLOCK PROGRAM PAUSE/RESTART : BPPS<011>/BPRS<012>**

Ladder Symbol	Operand Data Area
BPPS<011>	N
BPRS<012>	N

**N: Block program number # (00 to 99)**

**Description**

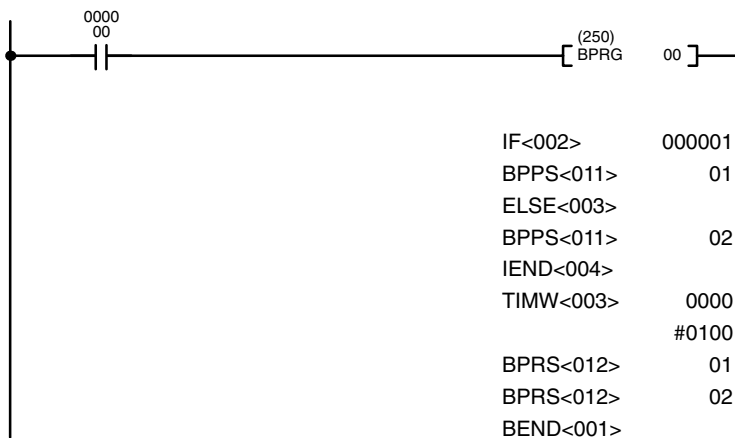
BPPS<011> is used inside one block program to suspend the execution of another block program. BPRS<012> restarts the specified block program. These instructions are effective whenever executed, i.e., they do not rely on operand bit status or execution condition.

**Precautions**

Even if the execution of a block program is suspended, PVs of all operating TIMW<013> timers and TMHW<015> high-speed timers with timer numbers T0000 through T0255 in the block program are updated continuously.

**Example**

If CIO 000000 is ON, the following program suspends execution of either block program 01 or block program 02 depending on the status of CIO 000001. The block program that was suspended is then restarted after 10 seconds.



Address	Instruction	Operands
000000	LD	000000
000001	BPRG(250)	00
000002	IF<002>	000001
000003	BPPS<011>	01
000004	ELSE<003>	
000005	BPPS<011>	02
000006	IEND<004>	
000007	TIMW<003>	0000 #0100
		# 0100
000008	BPRS<012>	01
000009	BPRS<012>	02
000010	BEND<001>	

**5-35-8 HIGH-SPEED TIMER/TIMER WAIT: TIMW<013>/TMHW<015>**

Ladder Symbol		Operand Data Areas	
TIMW<013>	N	<b>N:</b> Timer number	T
	SV	<b>SV:</b> Set value	CIO, G, A, T, C, #, DM, DR, IR
TMHW<015>	N		
	SV		

**Description**

TIMW<013> and TMHW<015> allow you to create a specified time lag (SV) between execution of the program part preceding it and the part following. The first part will be executed the first time the block program is entered. When the block timer instruction is reached, execution of the block program will halt until SV has expired, at which time the second part of the block program will be executed. Once the entire block program has been executed, the process is repeated.

SV is between 000.0 and 999.9 s for TIMW<013>, and between 00.00 and 99.99 s for TMHW<015>. The decimal point is not entered.

**Timer Precision**

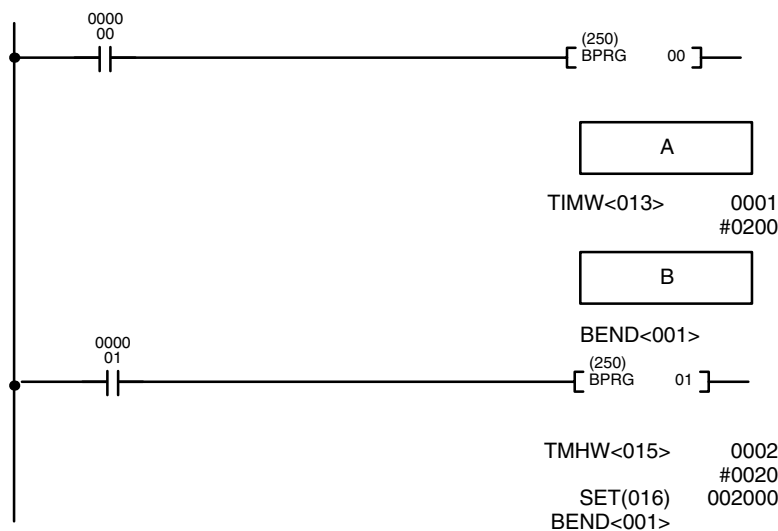
The operation of TIMW<013> is the same as in C/CVM1/CV-series PCs, but the Completion Flag can be turned ON about one cycle time (up to two cycle times) before or after the intended timer time-out. The error range is ± (10 ms + cycle time).

**Precautions**

Each timer number can be used as the definer in only one timer instruction, including those used in normal ladder-diagram timers.

**Example**

In the following example, B will be executed 20 seconds after A whenever CIO 000000 is ON, and CIO 002000 will be set 0.2 seconds after CIO 000001 goes ON.



Address	Instruction	Operands
000000	LD	000000
000001	BPRG(250)	00
A		
000200	TIMW<013>	0001 # 0200
B		
000300	BEND<001>	
000301	LD	000001
000302	BPRG(250)	01
000303	TMHW<015>	0002 # 0020
000304	SET(016)	002000
000305	BEND<001>	

**Flags**

ER (A 50003): SV data is not BCD.  
Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)

**5-35-9 COUNTER WAIT: CNTW<014>**

Ladder Symbol	Operand Data Areas
CNTW<014>	<b>N:</b> Counter number C
	<b>SV:</b> Set value CIO, G, A, T, C, #, DM, DR, IR
	<b>I:</b> Count input CIO, G, A, T, C

**Description**

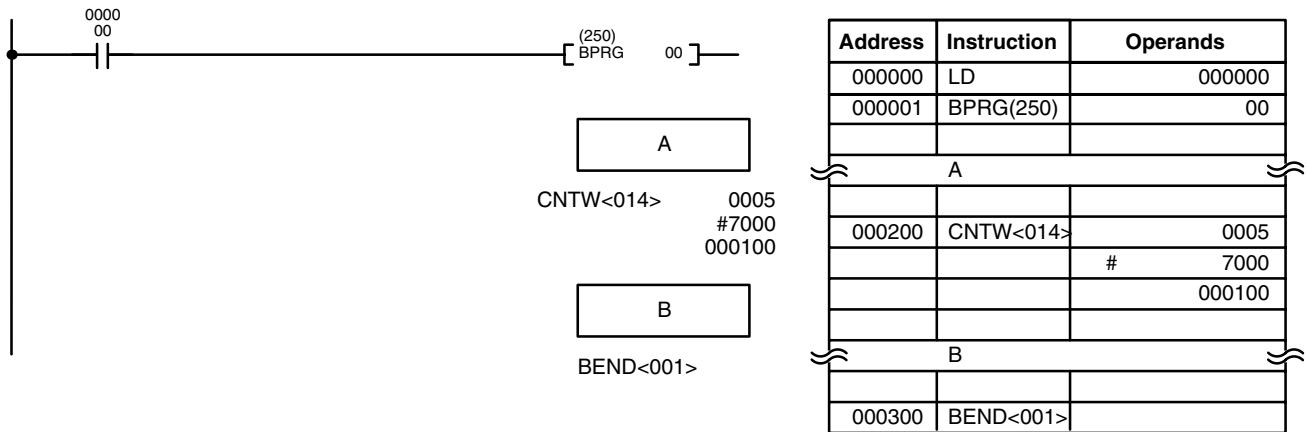
CNTW<014> allows you to create a 'count' lag (SV) between execution of the program part preceding the CNTW<014> (i.e., between BPRG(250) and CNTW<014> ) and the part following it (i.e., between CNTW<014> and BEND<001>). The first part will be executed the first time the block program is entered. When CNTW<014> is reached, the execution of the block program will stop until SV has been reached, at which time the second part of the block program will be executed. Once the entire block program has been executed, the process is repeated.

**Precautions**

Each counter number can be used as the definer in only one timer or counter instruction, including the normal ladder-diagram timers and counters.

**Example**

In the following example, B will be executed after the execution of A and after 7,000 counts of CIO 000100 while CIO 000000 is ON.



**Flags**

ER (A50003): SV data is not BCD.  
Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)

# SECTION 6

## Program Execution Timing

This section explains the execution cycle of the PC and shows how to calculate the cycle time and I/O response times. I/O response times in Link Systems are described in the individual System Manuals.

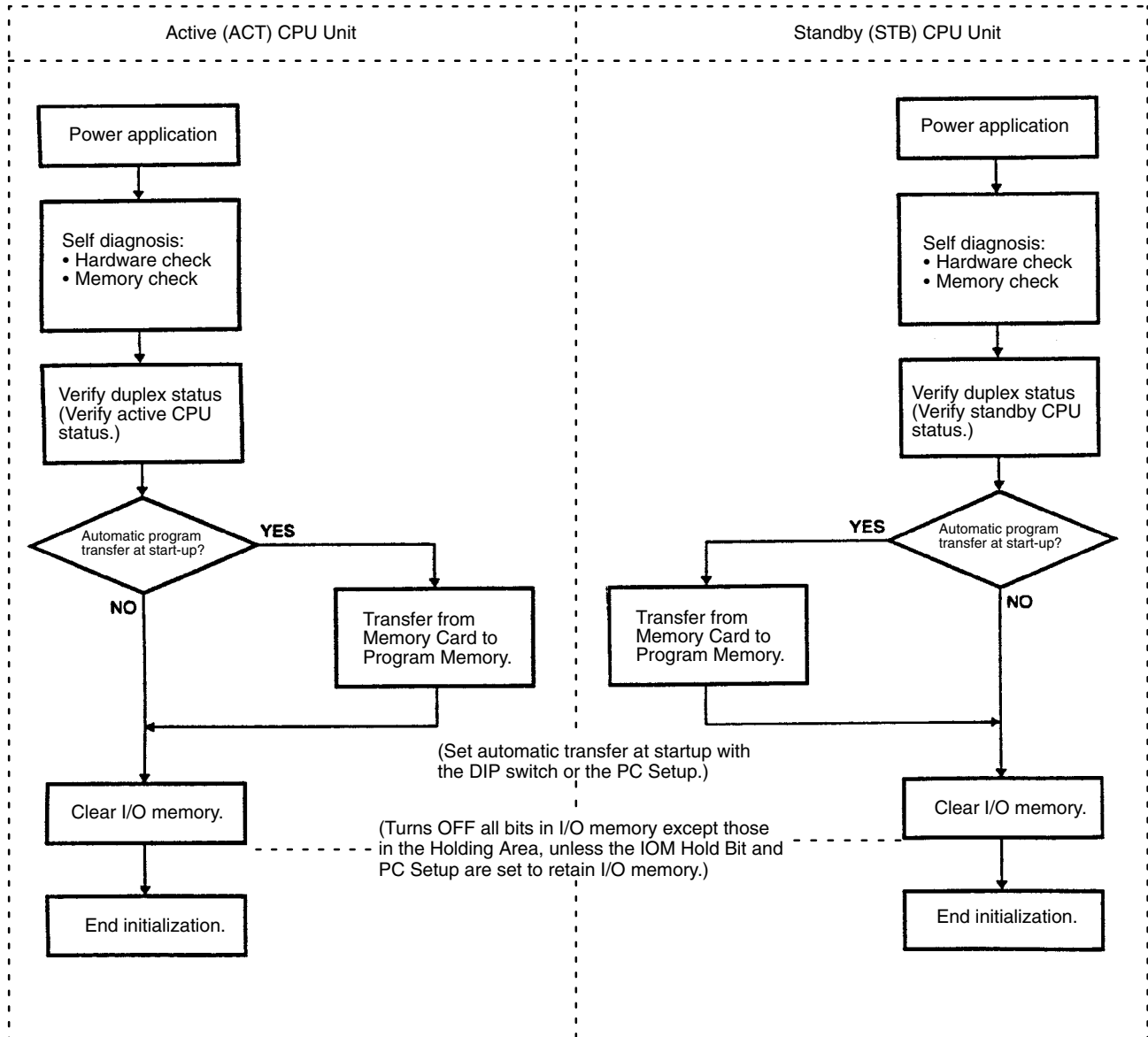
6-1	PC Operation .....	414
6-1-1	Initialization .....	414
6-1-2	Synchronous Operation in the Duplex System .....	415
6-1-3	I/O Refreshing Limitations .....	416
6-1-4	Memory Card Usage Limitations .....	416
6-1-5	CPU Processing after Power Interruptions .....	416
6-1-6	I/O Refreshing in SYSMAC BUS/2 and SYSMAC BUS Systems .....	417
6-1-7	Power OFF Operation .....	417
6-2	Cycle Time .....	418
6-2-1	Synchronous Operation .....	419
6-2-2	Duplex Initialization .....	420
6-2-3	Switching from Duplex to Simplex Operation .....	421
6-2-4	Example of Error Detection Program .....	422
6-2-5	Operations Significantly Increasing Cycle Time .....	422
6-2-6	Potential Problems with Event Processing .....	423
6-3	Calculating Cycle Time .....	423
6-4	Instruction Execution Times .....	425
6-5	I/O Response Time .....	437
6-5-1	I/O Units Only .....	437
6-5-2	I/O Response Times in a SYSMAC BUS System .....	438
6-5-3	I/O Response Times in a SYSMAC BUS/2 System .....	439

## 6-1 PC Operation

This section details basic CPU operation of CVM1D PCs.

### 6-1-1 Initialization

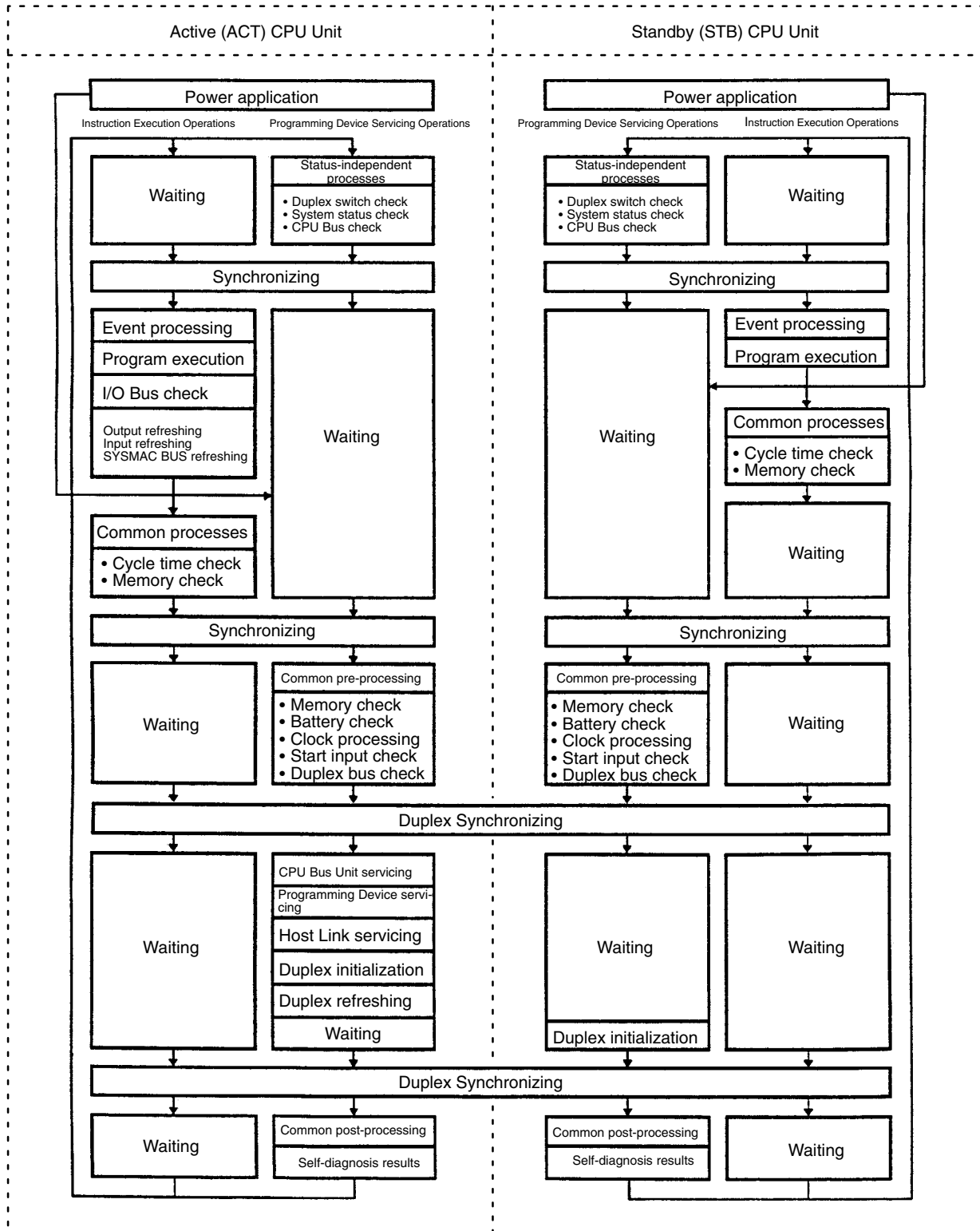
The following diagram shows initialization in a duplex CVM1D PC on power-up:





### 6-1-2 Synchronous Operation in the Duplex System

The active CPU and standby CPU perform synchronous processing as shown in the following diagram. The CVM1D PCs always operate in synchronous operation, asynchronous operation **cannot** be set in the PC Setup.



### 6-1-3 I/O Refreshing Limitations

In CVM1D PCs, cyclic refreshing is the only method used to refresh the entire I/O area. The I/O REFRESH instruction (IORF(184)) can be used to refresh specific ranges of I/O words.

- The refreshing method cannot be changed to “scheduled refreshing” or “zero-cross refreshing” in the PC Setup. It is not possible to change these settings from Programming Devices.
- Immediate-refreshing variations (with the “!” prefix) cannot be used in duplex mode. Operation in the active and standby CPUs might differ in duplex mode. Immediate-refreshing variations can be used in simplex mode.
- Interrupt Input Units cannot be used. If an Interrupt Input Unit is mounted, it will be treated as a normal Input Unit.  
The interrupt control instructions (MSKS(153), CLI(154), and MSKR(155)) cannot be used in either simplex or duplex mode. These instructions will be processed as NOP(000) if they are included in the program.

### 6-1-4 Memory Card Usage Limitations

When Memory Cards are being used in duplex mode, only the Memory Card mounted in the active CPU Unit is used during operation. A Memory Card mounted in the standby CPU Unit will be ignored during operation, but the contents of the Memory Card will be downloaded to the CPU Unit at startup if pin 5 of the standby CPU Unit’s DIP switch is set to ON.

Memory Card read/write operations cannot be performed from the program and the Memory Card instructions (FILR(180), FILW(181), FILP(182), and FLSP(183)) cannot be used in the CVM1D. These instructions will be processed as NOP(000) if they are included in the program.

### 6-1-5 CPU Processing after Power Interruptions

In CVM1D PCs, a power interruption signal is output when the CPU detects a power supply voltage below 85% of full power. The momentary power interruption time is fixed at 0 ms, so power interruptions cannot be processed as momentary power interruptions. When the power supply voltage returns to the minimum value, PC operation will restart with initialization.

Power OFF Interrupts are not processed.

The following settings cannot be changed in the PC Setup. Changed settings will be ignored.

Parameter		Setting	Operation
F:Execute control 1	S:Error on power off	Fixed at “Do not treat as an error.”	Determines whether to stop operation based solely on the size of the DC Power Supply’s voltage drop when a power interruption or voltage drop occurs, regardless of the length of the interruption.
P:Power break		Fixed at “Momentary power interruption time = 0 ms.” (No momentary power interruptions)	
G:Execute control 2	D:Power OFF interrupt	Fixed at “Do not execute Power OFF interrupt.”	When the power supply voltage returns to minimum values, operation starts as if power was turned on.
B:Startup hold		Fixed at “Do not hold.”	The I/O memory will be refreshed once the PC has stopped.

### 6-1-6 I/O Refreshing in SYSMAC BUS/2 and SYSMAC BUS Systems

#### SYSMAC BUS/2

I/O refreshing takes place once each PC cycle, but I/O points in the SYSMAC BUS/2 System may not be refreshed every cycle if the SYSMAC BUS/2 communications cycle is longer than the PC cycle.

I/O refreshing of Units in a SYSMAC BUS/2 System can be disabled by turning ON the corresponding CPU Bus Service Disable Bits in A015. Bits 00 to 15 correspond to Units #0 to #15, respectively. Turn the bits OFF again to enable service and resume I/O refreshing.

#### SYSMAC BUS

I/O refreshing takes place once each PC cycle.

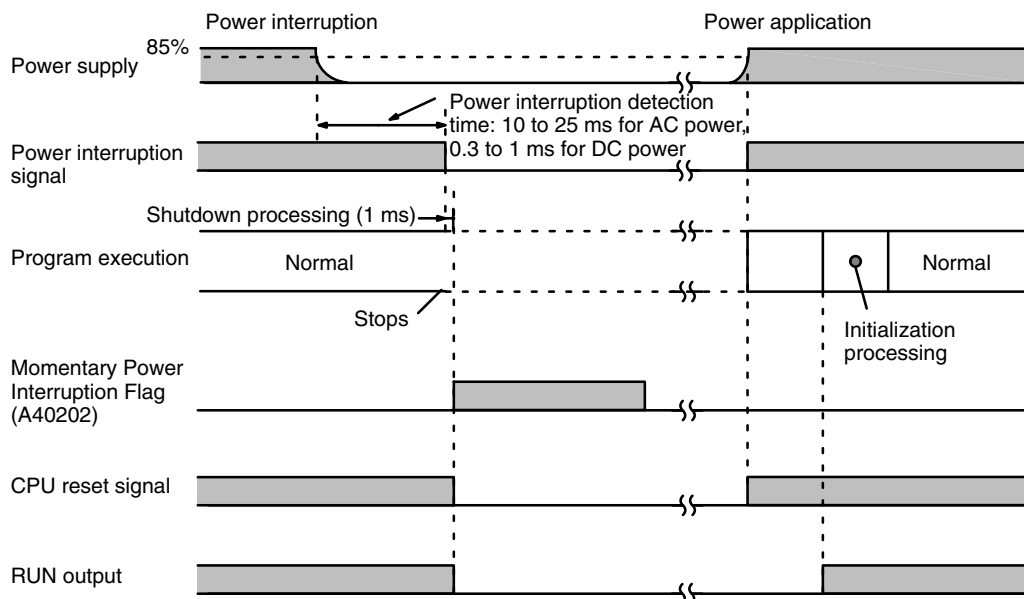
I/O refreshing of Units in a SYSMAC BUS System can be disabled by turning ON bit A01705 (the I/O Refresh Disable Bit). Turn A01705 OFF again to resume I/O refreshing.

### 6-1-7 Power OFF Operation

This section details CPU operation when power is turned OFF and ON, and during a momentary power interruption.

#### Power OFF/ON

The following diagram and explanation show the CPU operation when the power goes off and when power is turned on.



#### Power OFF Operation

The following list shows CPU operation when power is interrupted.

- 1, 2, 3... 1. A power interruption signal is output when the CPU detects a power supply voltage below 85% of full power. It takes the CPU between 10 ms and 25 ms to detect the power interruption with an AC power supply and between 0.3 ms and 1 ms to detect the power interruption with a DC power supply.
2. When the power interruption signal is output, program execution is stopped and the system shutdown procedure (1 ms) takes place. At this point, outputs are reset and timer PVs are maintained.  
Next, the CPU reset signal is generated, the CPU is stopped, and all outputs are turned OFF.

#### Power ON Procedure

The following list shows CPU procedures when power is returned to the PC.

- 1, 2, 3... 1. When the CPU detects a power supply voltage above 85% of full power, both the power interruption signal and the CPU reset signal will be turned OFF.

2. The CPU will begin initialization when the CPU reset signal is turned OFF.
3. When CPU initialization is completed, the program will be executed. The new power interruption time is written to A012 and A013 and the new number of power interruptions is written to A014.

Momentary power interruptions are not treated as errors.

- Note**
1. The momentary power interruption time is fixed at 0 ms.
  2. The actual time that power is interrupted is the power interruption detection time (10 ms to 25 ms) + the shutdown processing time (1 ms).

**Related Auxiliary Area Words** Auxiliary Area words A012 and A013 contain the time at which power was interrupted, as shown in the following table.

Word	Bits	Contents	Possible values
A012	00 to 07	Seconds	00 to 59
	08 to 15	Minutes	00 to 59
A013	00 to 07	Hours	00 to 23 (24-hour system)
	08 to 15	Day of month	01 to 31 (adjusted by month and for leap year)
A014	00 to 15	Number of power interruptions (0 to 9,999)	

## 6-2 Cycle Time

Understanding the operations that occur during the cycle and the elements that affect cycle time is essential to effective programming and PC operations. The major factors in determining program timing are the cycle time and the I/O response time. One cycle of CPU operation is called a cycle; the time required for one cycle is called the cycle time. The time required to produce a control output signal following reception of an input signal is called the I/O response time.

To aid in PC operation, the present and maximum cycle times are recorded in the read-only section of the Auxiliary Area. The present cycle time is held in A462 and A463 and the maximum cycle time is held in A464 and A465.

## 6-2-1 Synchronous Operation

CVM1D PCs always operate in synchronous operation; the instruction execution processing and Programming Device servicing are synchronized in a single cycle. The cycle time is thus the sum of the time required for instruction execution and that required for Programming Device servicing, and the cycle time will lengthen as more Communications Units and Special I/O Units are connected that are serviced during Programming Device servicing.

### Instruction Execution Cycle Time

Process	Actions		Processing time
Basic processes	Checks the cycle time and PC memory.	Checks the CPU bus, PC memory, battery, start input, duplex bus, and updates the clock.	Duplex mode: approx. 8.5 ms Simplex mode: approx. 5.0 ms
Event processing	Execution of non-scheduled requests for data processing from Programming Devices.		0 ms if there is no processing request; 20 ms max. (12% of previous cycle's cycle time is for servicing.)
Program execution	Program executed.		For ladder programs, total execution time for all instructions varies with program size, the instructions used, and execution conditions. Refer to 6-4 <i>Instruction Execution Times</i> for details.
Output refreshing	Output terminals are updated according to status of output bits in memory (including Special I/O Units).		7 $\mu$ s per word (per 16 points)
Input refreshing	Input bits are updated according to status of input signals (including Special I/O Units).		8 $\mu$ s per word (per 16 points)
SYSMAC BUS refreshing	Input bits allocated to Units connected to Slaves are updated according to status of input signals. Output signals sent to Units connected to Slaves are updated according to status of output bits in memory.		1 ms per Master plus 0.05 ms per word on Remote I/O Slaves (0.12 ms per word on Optical I/O Units)
CPU Bus Unit servicing	Refreshing and event processing are performed in unit-number order starting with Unit #0. (Refreshing involves only CPU Bus Link refreshing of BASIC Units and Personal Computer Units.)		SYSMAC NET Link/SYSMAC LINK refreshing: Approx. 1.4 ms + 1 $\mu$ s per data link word + 0.3 ms if both DM and CIO are used for data links.  SYSMAC BUS/2 refreshing: Approx. 2.0 ms + 1 $\mu$ s per word refreshed <ul style="list-style-type: none"> <li>• Approx. 0.8 ms for BASIC Unit refreshing</li> <li>• Approx. 0.8 ms for Personal Computer</li> <li>• Approx. 1 ms for event processing</li> </ul>
Programming Device servicing	If a Programming Device is connected, commands from it are processed.		1.2 ms
Host interface servicing (Pin 3 of DIP switch OFF)	Host link servicing is processed if a host link is connected to the DPL Unit.		1.2 ms
NT link servicing (Pin 3 of DIP switch ON)	The NT link is serviced when a PT is connected to the CPU and the NT link is being used.  NT link servicing is processed if a PT is connected to the DPL Unit and an NT link is established.		Fixed at 2.2 ms
I/O bus check	Performs an I/O bus check of basic Racks.		1.2 ms
Duplex refreshing (Duplex mode only)	Transfers input refreshing data to the standby CPU Unit.		10 $\mu$ s per Input Unit word

### Service Disable Bits

The Service Disable Bits shown in the table below are primarily used to reduce the cycle time; they are only effective in RUN and MONITOR modes. The bits are OFF when the PC is first turned on, and are normally turned ON from the program.

Do not leave Service Disable Bits ON for longer than is necessary; service between the PC and the designated Unit will be stopped completely as long as the corresponding Service Disable Bit is ON.

Word(s)	Bit(s)	Function
A015	00 to 15	CPU Bus Service Disable Bits (Bits 00 to 15 correspond to units #0 to #15.)
A017	03	Host Link/NT Link Service Disable Bit
	04	Programming Device Service Disable Bit
	05	I/O Refresh Disable Bit

### 6-2-2 Duplex Initialization

When the PC is operating in duplex mode, duplex initialization is performed just after host link servicing to coordinate data values in the active and standby CPU Units. An active/standby CPU changeover cannot occur during duplex initialization because the PC is in simplex status, not duplex status, during duplex initialization.

PC operation will not continue if a changeover error (WDT error, memory error, or CPU Usage Switch setting of "NO USE") occurs in the active CPU Unit during duplex initialization.

The following table shows the execution timing and processing that occur during duplex initialization. (UM is the Program Area and IOM is I/O memory.)

Execution Timing		Processing						
		UM Verification	UM Transfer	PC Setup Verification	PC Setup Transfer	EM Size Verification	EM Transfer	IOM Transfer
Both CPUs are turned ON with DPL setting.		YES	---	YES	---	YES	YES	YES
Initialize Button is pressed with DPL setting. (Unless the PC is already operating in duplex mode.)		YES	YES	YES	YES	YES	YES	YES
Start of operation in duplex mode		---	---	---	---	YES	YES	YES
FINS commands	0202: MEMORY AREA WRITE	---	---	YES	YES	YES	YES	YES
	0203: MEMORY AREA FILL	---	---	YES	YES	YES	YES	YES
	0221: UNIT SETUP WRITE	---	---	YES	YES	YES	YES	YES
	0222: I/O TABLE CREATE	---	---	YES	YES	YES	YES	YES
	0304: PROGRAM AREA PROTECT	YES	YES	---	---	YES	YES	YES
	0305: PROGRAM AREA PROTECT CLEAR	YES	YES	---	---	YES	YES	YES
	0307: PROGRAM AREA WRITE	YES	YES	---	---	YES	YES	YES
	0321: PROGRAM INSERT/DELETE	YES	YES	---	---	YES	YES	YES
	0323: OPERAND CHANGE	YES	YES	---	---	YES	YES	YES
	0325: TIM/CNT SV CHANGE	YES	YES	---	---	YES	YES	YES
	2104: ONLINE I/O REPLACE START/END	---	---	---	---	YES	YES	YES
	220B: PARAMETER AREA FILE TRANSFER	---	---	YES	YES	YES	YES	YES
	220C: PROGRAM AREA FILE TRANSFER	YES	YES	---	---	YES	YES	YES

The configuration of the PC can greatly affect the cycle time. The following table shows how duplex initialization will increase the cycle time for a PC configuration with a normal cycle time of 20 ms.

Number of EM banks	Program size	Maximum cycle time
0	Minimum	127 ms
	Maximum	200.2 ms
2	Minimum	283 ms
	Maximum	356.2 ms
4	Minimum	439 ms
	Maximum	512.2 ms
8	Minimum	751 ms
	Maximum	824.2 ms

The maximum cycle time (T ms) can be calculated with the following equation:

$$T = (\text{UM size (KW)} \times 1.22) + (\text{EM banks} \times 78.0) + \text{normal cycle time}$$

- Note**
1. The standby CPU's WAIT indicator will be lit and its DPL RUN indicator will be flashing during duplex initialization.
  2. If the standby CPU's Usage Switch is set to "NO USE" during duplex initialization, the CPU's program and PC Setup will be incorrect and the CPU will not operate properly when it is started.

### 6-2-3 Switching from Duplex to Simplex Operation

During duplex operation, the active CPU and standby CPU monitor the other CPU's errors. Also, the synchronous transmission buses are also monitored since it is a duplex system and the PC will switch from duplex operation to simplex operation when an error is detected.

The following table lists the cases in which operation will be switched from duplex to simplex operation.

CPU Status	Cause for Switching	Precautions during Switching
Active CPU → Standby CPU, Standby CPU → Active CPU	<p>The active and standby CPU Units will be interchanged in the following cases:</p> <ul style="list-style-type: none"> <li>• Memory error</li> <li>• CPU error</li> <li>• Active CPU's Usage Switch setting changed from "USE" to "NO USE."</li> </ul>	<p>Commands issued from Programming Devices or host computers may be corrupted whether the device is connected to the PC through the Programming Device Connector, Host Link Connector, or the Programming Device Connector on an I/O Interface Unit.</p> <p>A communications error may occur in a Programming Device. Program a host computer to retry failed transmissions.</p> <p>The execution conditions for up-differentiated (↑) variations of SEND(192), RECV(193), CMND(194), and IORF(184) might not be recognized.</p> <p>The cycle time will be shortened.</p>
Active CPU remains active	<p>The active CPU Unit will remain active in the following cases:</p> <ul style="list-style-type: none"> <li>• Duplex bus error</li> <li>• Duplex verification error</li> <li>• A memory error or CPU error occurs in the standby CPU.</li> <li>• Standby CPU's Usage Switch setting changed from "USE" to "NO USE."</li> <li>• Memory or CPU errors occur in both the active and standby CPUs.</li> <li>• The DPL Unit's Mode Switch setting is set to "SPL" and its Initialize Button is pressed.</li> </ul>	<p>The active CPU will continue operating.</p> <p>The cycle time will be shortened.</p>

To switch from simplex operation back to duplex operation, eliminate the “cause for switching” described in the table above, set the DPL Unit’s Mode Switch to “DPL,” and press the Initialize Button.

**Note** The cycle time will be shortened when operation is switched from duplex to simplex operation because the time required to synchronize the active and standby CPUs is eliminated. That difference will be substantial when instructions that have much longer instruction execution times in duplex mode (READ(190), WRIT(191), SEND(192), RECV(193), CMND(194), and IORF(184)) have been used. Be sure to check system operation in simplex and duplex modes in advance.

### 6-2-4 Example of Error Detection Program

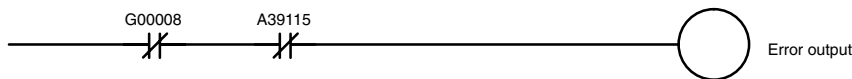
Use either of the program sections shown below to detect a switch from duplex operation to simplex operation.

- 1, 2, 3... 1. Use Auxiliary Area bits A39409 to A39411 to indicate the status of the other CPU Unit during duplex mode operation:



A39409: ON when the other CPU Unit can be mounted or dismantled.  
 A39410: ON when a CPU error is generated at the other CPU Unit.  
 A39411: ON when a memory error is generated at the other CPU Unit.

2. Use G00008 and A39115:



G00008: SPL/DPL mode setting output  
 OFF in simplex mode  
 ON in duplex mode  
 A39115: ON during duplex initializing processing

### 6-2-5 Operations Significantly Increasing Cycle Time

The operations described below can significantly affect the cycle time when performed from SSS.

Operation	Effect on cycle time	Precautions
Instruction trace	Cycle time increased by a factor of 3 to 5 times.	The cycle time will change during an instruction trace and high-speed input signals might not be detected. Set the maximum cycle time in the PC Setup at least 5 times higher than its usual value.
Memory Card operations <sup>1</sup>	Duplex initialization time will be increased.	High-speed input signals might not be detected.

**Note** 1. Memory Card operations include: transfers from the Memory Card to the PC, online editing, transferring customization information from the computer to the PC, making Communications Unit settings, transferring routing tables, and transferring data link tables from the computer to the PC.  
 2. In the CVM1D, PC operation is stopped for the times shown in the following table but there is no effect on cycle time (i.e., those times are not added to the cycle time).



Online edit operation	Stopped time
Instruction block inserted or deleted at the beginning of 1 62K-word program.	Approx. 0.5 s
Instruction block including JME(005) deleted at the beginning of a 62K-word program.	Approx. 0.2 s

While PC operation is stopped, output status is retained and inputs are not accepted and communications with SYSMAC NET, SYSMAC LINK, SYSMAC BUS/2, Host Link, and Programming Devices are stopped.

## 6-2-6 Potential Problems with Event Processing

It is possible for two or more events (unscheduled requests for data processing) from Programming Devices, the host interface, or CPU Bus Units to occur at the same time.

### Multiple Writing Events

When two or more writing events (writing the program, writing parameters, registering the I/O table, changing the mode) occur simultaneously, they are processed in the order in which they were received. The later arriving events will not be executed until the earlier events have been completed.

The following SSS operations are writing events:

Data modification, set, reset, online edit, DM edit, mode change, transfer block (SSS to PC), save block (SSS to PC), I/O table transfer, I/O table register, PC Setup, PC Setup information transfer, data trace execution, program trace execution, setup for BASIC Units and Personal Computer Units, software switch settings for BASIC Units and Personal Computer Units, data link table transfer, routing table transfer, clearing errors, clearing the error log, setting the clock, setting program memory protect, clearing program memory protect, debug transfer, reading cycle time

The following host interface operations are writing events:

Data area block write, data area transfer, parameter area write, parameter area block write, start program area protect, clear protect area, program area write, program area clear, start execution, stop execution, write clock information, clear message, allow access, force allow access, open access, clearing errors, clearing the error log, variable area to file transfer, parameter area to file transfer, program area to file transfer, force set/reset, force set/reset for all bits

### Multiple Instruction Execution Events

Like writing events, when two or more instruction execution events (reading/writing the program, etc.) occur simultaneously, they are processed in the order in which they were received. The later arriving events will not be executed until the earlier events have been completed.

The following SSS operations are instruction execution events:

Monitoring, data modification, set, reset, online edit, DM edit, search, transfer block (SSS to PC), save block (SSS to PC), I/O table change, PC Setup, PC Setup information transfer, data trace execution, program trace execution, setting program memory protect, clearing program memory protect

The following host interface operations are writing events:

Data area block write, multiple data area read, data area transfer, parameter area write, parameter area block write, start program area protect, clear protect area, program area read, program area write, program area clear, cycle time read, program area to file transfer, force set/reset, force set/reset for all bits

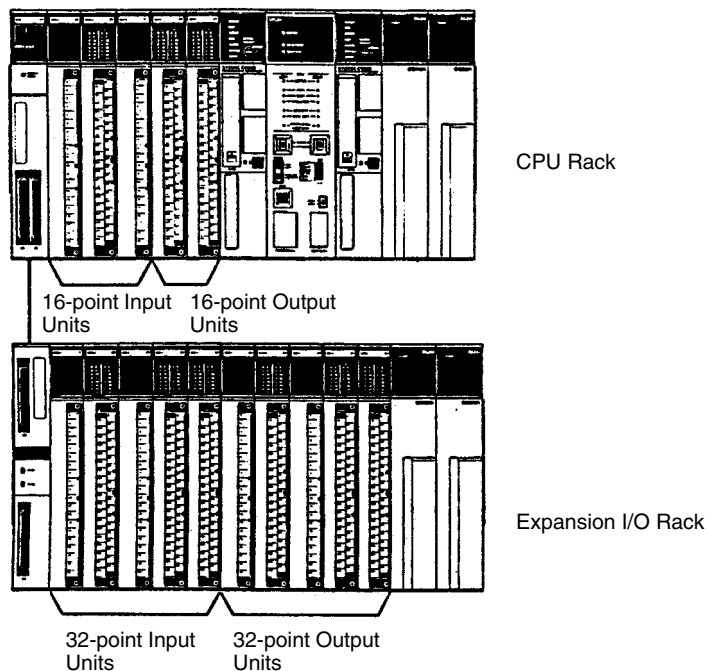
## 6-3 Calculating Cycle Time

The PC configuration, the program, and program execution conditions must be taken into consideration when calculating the cycle time. This means tak-

ing into account such things as the number of I/O points, the programming instructions used, and whether or not Programming Devices are being used. This section shows a basic example of cycle time calculation. Operating times are given in the tables in 6-2 Cycle Time.

Here, we'll compute the cycle time for a CVM1D controlling only I/O Units, five on the CPU Rack and ten on an Expansion I/O Rack. The PC configuration for this is shown below. It is assumed that the program contains 20,000 instructions requiring an average of 0.3 μs each to execute.

Refer to the next section for instruction execution times. Using the cycle time in calculating the I/O response time is described in the last part of Section 6.



**Calculations**

The following table shows the time required for each process in the cycle.

	Process	Calculation	Time (ms)
1	Overseeing time (basic processes)	---	8.5
2	Program execution	20,000 steps × 0.3 μs	6.0
3	Output refreshing	$[(16 \text{ points} \times 2 + 32 \text{ points} \times 5) \div 16] \times 7 \mu\text{s}$	0.084
4	Input refreshing	$[(16 \text{ points} \times 3 + 32 \text{ points} \times 5) \div 16] \times 8 \mu\text{s}$	0.104
5	I/O bus check	---	1.2
6	Duplex refreshing	$[(16 \text{ points} \times 3 + 32 \text{ points} \times 5) \div 16] \times 10 \mu\text{s}$	0.11
Approximate cycle time (processes 1 through 5)			16.00

## 6-4 Instruction Execution Times

This following table lists the execution times for CVM1D instructions. The maximum and minimum execution times and the conditions which cause them are given where relevant. When “word” is referred to in the Conditions column, it implies the content of any word except for indirectly addressed DM words. Indirectly addressed DM words, which create longer execution times when used, are indicated by “\*DM.”

Execution times for most instructions depend on whether they are executed with an ON or an OFF execution condition. Exceptions are the ladder diagram instructions OUT and OUT NOT, which require the same time regardless of the execution condition. The OFF execution time for an instruction can also vary depending on the circumstances, i.e., whether it is in an interlocked program section and the execution condition for IL is OFF, whether it is between JMP(004) and JME(005) and the execution condition for JMP(004) is OFF, or whether it is reset by an OFF execution condition. “R,” “IL,” and “JMP” are used to indicate these three times.

The *Words* column provides the number of words required by the instruction in program memory. Instructions can require between one and eight words in memory. The length of an instruction depends not only on the instruction, but also on the operands used for the instruction. If an index register is addressed directly or a data register is used as an operand, the instruction will require one word less than when specifying a word address for the operand. If a constant is designated for an instruction that uses 2-word operands, the instruction will require one word more than when specifying a word address for the operand.

**Table: Instruction Execution Times**

Instruction	Words	Conditions	ON execution time (μs)	OFF execution time (μs)
LD	1	CIO 000000 to CIO 051115 for operand	0.13	0.13
	2	CIO 051200 to CIO 255515 for operand	0.25 ↑/↓: +0.375 !: +2.0	0.25 ↑/↓: +0.625 !: +2.0
LD NOT	1	CIO 000000 to CIO 051115 for operand	0.13	0.13
	2	CIO 051200 to CIO 255515 for operand	0.25 ↑/↓: +0.375 !: +2.0	0.25 ↑/↓: +0.375 !: +2.0
AND	1	CIO 000000 to CIO 051115 for operand	0.13	0.13
	2	CIO 051200 to CIO 255515 for operand	0.25 ↑/↓: +0.375 !: +2.0	0.25 ↑/↓: +0.375 !: +2.0
AND NOT	1	CIO 000000 to CIO 051115 for operand	0.13	0.13
	2	CIO 051200 to CIO 255515 for operand	0.25 ↑/↓: +0.375 !: +2.0	0.25 ↑/↓: +0.375 !: +2.0
OR	1	CIO 000000 to CIO 051115 for operand	0.13	0.13
	2	CIO 051200 to CIO 255515 for operand	0.25 ↑/↓: +0.375 !: +2.0	0.25 ↑/↓: +0.375 !: +2.0
OR NOT	1	CIO 000000 to CIO 051115 for operand	0.13	0.13
	2	CIO 051200 to CIO 255515 for operand	0.25 ↑/↓: +0.375 !: +2.0	0.25 ↑/↓: +0.375 !: +2.0
AND LD	1	---	0.13	0.13
OR LD	1	---	0.13	0.13
OUT	2	---	0.38	0.38
OUT NOT	2	---	!: +1.75	!: +1.75

Instruction	Words	Conditions	ON execution time (μs)	OFF execution time (μs)
TIM	3	Constant for SV	3.32	R or IL: 7.8
		*DM for SV		R or IL: 10.8
CNT	3	Constant for SV	1.25	R or IL: 1.25
		*DM for SV		R: 5.75 IL: 1.25
NOP(000)	1	---	0.13	---
END(001)	1		3.75	
IL(002)	2	---	1.5	1.0
ILC(003)	2		1.0	1.0
JMP(004)	3	---	11.0	0.88
JME(005)	2	---	0.25	0.25
FAL(006)	4	---	457	1.0
FAL(006) 00			611	1.0
FALS(007)			825	1.0
STEP(008)	3	---	218	207
SNXT(009)	3	---	225	0.88
NOT(010)	1	---	0.13	0.13
KEEP(011)	2	---	0.38	0.38
CNTR(012)	4	Constant for SV	6.0	R: 7.5 IL: 5.3
		*DM for SV	8.0	R: 8.9 IL: 5.3
DIFU(013)	2	---	0.63	0.63
DIFD(014)	2	---	0.63	0.63
TIMH(015)	3	Constant for SV	3.19	R or IL: 7.7
		*DM for SV		R or IL: 10.7
SET(016)	2	---	0.375	0.38
RSET(017)	2			
UP(018)	2	---	4.0	4.0
DOWN(019)	2			
CMP(020)	4	When comparing a constant to a word	3.3	0.88
		When comparing two *DM	5.9	
!CMP(020)	4	Additional time over CMP(020) for each input word being compared	+5.0	
		Additional time over CMP(020) for each output word being compared	+4.0	
		Additional time over CMP(020) for words other than I/O words	+0	
CMPL(021)	4	When comparing two words	4.3	1.0
		When comparing two *DM	6.8	
BCMP(022)	5	Comparing constant to table with results to a word	16.9	1.13
		Comparing *DM to table with results to *DM	20.3	
TCMP(023)	5	When comparing a word to a word table	14.9	
		When comparing a *DM to *DM table	18.3	
MCMP(024)	5	When comparing two words	17.3	
		When comparing two *DM	20.3	
EQU(025)	4	Comparing two words	3.1	1.0
		Comparing two *DM	5.8	

Instruction	Words	Conditions	ON execution time (μs)	OFF execution time (μs)	
CPS(026)	4	Comparing constants and words	3.5	1.0	
		Comparing two *DM	6.25		
ICPS(026)	4	Amount added per input word at time of comparison	+5.0		
		Amount added per output word at time of comparison	+4.0		
		Other areas at time of comparison	+0		
CPSL(027)	4	Comparing constants and words	4.5		
		Comparing two *DM	6.88		
CMP(028)	4	Comparing constants and words	3.3		0.88
		Comparing two *DM	5.9		
ICMP(028)	4	Amount added per input word at time of comparison	+5.0		
		Amount added per output word at time of comparison	+4.0		
		Other areas at time of comparison	+0		
CMPL(029)	4	Comparing two words	4.3	1.0	
		Comparing two *DM	6.8		
MOV(030)	4	When transferring a constant to a word	4.3		
		When transferring *DM to *DM	5.3		
IMOV(030)		Additional time over MOV(020) for each input word being used	+5.0		
		Additional time over MOV(020) for each output word being used	+4.0		
		Additional time over MOV(020) for words other than I/O words	+0		
MVN(031)	4	When transferring a constant to a word	4.4		
		When transferring *DM to *DM	5.4		
MOVL(032)	4	When transferring a word to a word	5.4		
		When transferring *DM to *DM	6.3		
MVNL(033)	4	When transferring a word to a word	5.4		
		When transferring *DM to *DM	6.3		
XCHG(034)	4	Word → word	6.4		
		*DM → *DM	6.9		
XCGL(035)	4	Word → word	8.5		
		*DM → *DM	9.0		
MOVR(036)	3	Word → IR	4.1		0.88
		*DM → IR	4.4		
MOVQ(037)	3	When transferring a word to a word	0.5	0.38	
		When transferring a constant to a word	0.63		
XFRB(038)	3	When transferring 1 bit from word to word	12.5	1.13	
		When transferring 255 bits from *DM to *DM	48.9		
XFER(040)	5	When transferring 1 word	12.3		
		When transferring 1000 words by *DM	1.51 ms		
BSET(041)	5	When setting a constant to one word	11.4		
		When setting 1000 DM words using *DM	1.16 ms		
MOVB(042)	5	When transferring a word to a word	8.1		
		When transferring *DM to *DM	11.4		
MOVD(043)	5	When transferring a word data to a word	7.4		
		When transferring *DM to *DM	10.8		

Instruction	Words	Conditions	ON execution time (μs)	OFF execution time (μs)		
DIST(044)	5	Constant → (word + word)	5.8	1.13		
		*DM → (*DM + *DM)	7.8			
COLL(045)	5	(word + word) → word	6.3			
		(*DM + *DM) → *DM	9.4			
BXFR(046)	5	Transferring 1 word from word to word	16.1			
		*DM → *DM, *DM (1000 words) transfer	1.52 ms			
SETA(047)	5	Setting 1 bit in a word	13.8			
		*DM → *DM (1000 bits) set	99.5			
RSTA(048)	5	Resetting 1 bit in a word	13.9			
		*DM → *DM, *DM (1000 bits) reset	99.8			
SFT(050)	5	With 1 word shift register	14.0		R: 12.0 IL: 1.0	
		With 1000 words shift register	1.28 ms		R: 1.15 ms IL: 1.0	
SFTR(051)	5	When shifting 1 word	13.5		1.13	
		When shifting DM 1000 words using *DM	1.30 ms			
ASFT(052)	5	When shifting 1 word	366			
		When shifting DM 1000 words using *DM	13.3 ms			
WSFT(053)	5	When shifting 1 word	11.5			
		When shifting 1000 DM words using *DM	1.17 ms			
NSFL(054)	5	Shifting 1 bit in a word	15.8			
		*DM → *DM (15 bits) shift	32.4			
NSFR(055)	5	Shifting 1 bit in a word	15.6			
		*DM → *DM (15 bits) shift	32.4			
NASL(056)	4	1-bit word shift	14.1	1.0		
		*DM (16 bits) shift	19.4			
NASR(057)	4	1-bit word shift	14.1			
		*DM (16 bits) shift	19.4			
NSLL(058)	4	1-bit word shift	15.0			
		*DM (32 bits) shift	35.4			
NSRL(059)	4	1-bit word shift	14.8			
		*DM (32 bits) shift	35.1			
ASL(060)	3	When shifting a word to left	5.1			0.88
		When shifting *DM to left	6.1			
ASR(061)	3	When shifting a word to right	5.3			
		When shifting *DM to right	6.3			
ROL(062)	3	When rotating a word to counterclockwise	5.3			
		When rotating *DM to counterclockwise	6.3			
ROR(063)	3	When rotating a word to clockwise	5.4			
		When rotating *DM to clockwise	6.4			
ASLL(064)	3	When shifting two words to left	6.3			
		When shifting two *DM to left	7.3			
ASRL(065)	3	When shifting two words to right	6.3			
		When shifting two *DM to right	7.3			
ROLL(066)	3	When rotating two words to counterclockwise	6.4			
		When rotating two *DM to counterclockwise	7.4			

Instruction	Words	Conditions	ON execution time (μs)	OFF execution time (μs)	
RORL(067)	3	When rotating two words to clockwise	6.4	0.88	
		When rotating two *DM to clockwise	7.4		
SLD(068)	4	When shifting 1 word	11.4	1.0	
		When shifting 1000 DM words using *DM	151		
SRD(069)	4	When shifting 1 word	11.4	1.13	
		When shifting 1000 DM words using *DM	151		
ADD(070)	5	Constant + word → word	7.5	1.13	
		*DM + *DM → *DM	9.0		
SUB(071)	5	Constant – word → word	7.4		
		*DM – *DM → *DM	8.9		
MUL(072)	5	Constant x word → word	18.4		
		*DM x *DM → word	19.8		
DIV(073)	5	Word ÷ constant → word	17.5		
		*DM ÷ *DM → *DM	21.1		
ADDL(074)	5	Constant + word → word	18.0		
		*DM + *DM → *DM	21.5		
SUBL(075)	5	Constant – word → word	17.5		
		*DM – *DM → *DM	21.0		
MULL(076)	5	Constant x word → word	53.8		
		*DM x *DM → word	57.3		
DIVL(077)	5	Word ÷ constant → word	62.1		
		*DM ÷ *DM → *DM	65.5		
STC(078)	2	---	1.38		0.75
CLC(079)	2	---	1.38		
ADB(080)	5	Constant + word → word	6.0		1.13
		*DM + *DM → *DM	9.6		
SBB(081)	5	Constant – word → word	6.1		
		*DM – *DM → *DM	9.8		
MLB(082)	5	Constant x word → word	13.1		
		*DM x *DM → word	16.8		
DVB(083)	5	Word ÷ constant → word	15.9		
		*DM ÷ *DM → *DM	19.5		
ADBL(084)	5	Constant + word → word	7.6		
		*DM + *DM → *DM	11.1		
SBBL(085)	5	Constant – word → word	7.8		
		*DM – *DM → *DM	11.3		
MLBL(086)	5	Constant x word → word	38.4		
		*DM x *DM → word	41.9		
DVBL(087)	5	Word ÷ constant → word	47.8		
		*DM ÷ *DM → *DM	51.3		
INC(090)	3	Word increment	5.3	0.88	
		*DM increment	6.3		
DEC(091)	3	Word decrement	5.4		
		*DM decrement	6.4		
INCB(092)	3	Word increment	4.6		
		*DM increment	5.6		
DECB(093)	3	Word decrement	4.9		
		*DM decrement	5.9		

Instruction	Words	Conditions	ON execution time (μs)	OFF execution time (μs)	
INCL(094)	3	Word increment	12.9	0.88	
		*DM increment	13.9		
DECL(095)	3	Word decrement	12.8		
		*DM decrement	13.8		
INBL(096)	3	Word increment	5.6		
		*DM increment	6.6		
DCBL(097)	3	Word decrement	5.8		
		*DM decrement	6.6		
BIN(100)	4	When converting a word to a word	5.1		1.0
		When converting *DM to *DM	7.1		
BCD(101)	4	When converting a word to a word	5.0		
		When converting *DM to *DM	7.0		
BINL(102)	4	When converting a word to a word	8.5		
		When converting *DM to *DM	10.5		
BCDL(103)	4	When converting a word to a word	9.6		
		When converting *DM to *DM	11.6		
NEG(104)	4	When converting a word to a word	4.8		
		When converting *DM to *DM	6.9		
NEGL(105)	4	When converting a word to a word	5.9		
		When converting *DM to *DM	7.9		
SIGN(106)	4	When expanding a word to a word	5.5		
		When expanding *DM to *DM	7.5		
MLPX(110)	5	When decoding a word to a word	6.0	1.13	
		When decoding *DM to *DM	13.0		
DMPX(111)	5	When encoding a word to a word	8.1		
		When encoding *DM to *DM	13.5		
SDEC(112)	5	When decoding a word to a word	264		
		When decoding *DM to *DM	353		
ASC(113)	5	Word → word	270		
		*DM → *DM	359		
BCNT(114)	5	When counting 1 word	344		
		When counting 1000 words with *DM	17.1 ms		
LINE(115)	5	When converting word to word using constant specification	11.6		
		When converting *DM to *DM with DM specification	15.0		
COLM(116)	5	When converting word to word using constant specification	20.9		
		When converting *DM to *DM with DM specification	24.5		
HEX(117)	5	Converting 1 digit, word → word	17.1		
		Converting 4 digits, *DM → *DM	40.5		
TTIM(120)	4	Constant for SV	10.3	R: 8.1 IL: 2.5	
		*DM for SV	11.7	R: 9.5 IL: 2.5	
TIML(121)	5	---	1.42 ms	R or IL: 0.96	
MTIM(122)	5	---	1.88 ms	1.113	
ANDW(130)	5	Constant AND word → word	5.6	1.13	
		*DM AND *DM → *DM	9.1		



Instruction	Words	Conditions	ON execution time (μs)	OFF execution time (μs)	
ORW(131)	5	Constant OR word → word	5.6	1.13	
		*DM OR *DM → *DM	9.3		
XORW(132)	5	Constant XOR word → word	5.6		
		*DM XOR *DM → *DM	9.3		
XNRW(133)	5	Constant XNOR word → word	5.6		
		*DM XNOR *DM → *DM	9.3		
ANDL(134)	5	Constant AND word → word	7.3		
		*DM AND *DM → *DM	10.8		
ORWL(135)	5	Constant OR word → word	7.3		
		*DM OR *DM → *DM	10.8		
XORL(136)	5	Constant XOR word → word	7.3		
		*DM XOR *DM → *DM	10.8		
XNRL(137)	5	Constant XNOR word → word	7.3		
		*DM XNOR *DM → *DM	10.8		
COM(138)	3	When inverting a word	4.6		0.88
		When inverting *DM	5.6		
COML(139)	3	When inverting a word	5.5		
		When inverting *DM	6.5		
ROOT(140)	4	---	647	1.0	
FDIV(141)	5	Word ÷ word → word (equals 0)	305	1.13	
		Word ÷ word → word (not 0)	1.43 ms		
		*DM ÷ *DM → *DM	1.45 ms		
APR(142)	5	When specifying sine or cosine	338	1.13	
		When specifying a word with 256-word table	2.35 ms		
SEC(143)	4	When converting a word to a word	342	1.0	
		When converting *DM to *DM	477		
HMS(144)	4	---	740	1.13	
CADD(145)	5	---	0.94 ms		
CSUB(146)	5	---	0.94 ms	---	
SBN(150)	2	---	---	---	
SBS(151)	3	---	3.1	0.88	
RET(152)	2	---	11.5	7.1	
MCRO(156)	5	Parameter word designation	29.5	1.13	
		Parameter *DM designation	31.4		
SSET(160)	4	When setting a 3-word stack	12.6	1.0	
		When setting a 999-word stack	644		
PUSH(161)	4	When designating stack through word	3.3		
		When designating stack through *DM	4.8		
LIFO(162)	4	When designating stack through word	3.4		
		When designating stack through *DM	4.4		
FIFO(163)	4	When using 3-word stack	12.3		
		When using 999-word stack	1.04 ms		
SRCH(164)	5	When searching 1 word	289		1.13
		When searching 1000 words for *DM	9.54 ms		
MAX(165)	5	When searching 1 word	386	1.13	
		When searching 1000 words for *DM	56.9 ms		
MIN(166)	5	When searching 1 word	388	1.13	
		When searching 1000 words for *DM	57.1 ms		

Instruction	Words	Conditions	ON execution time (μs)	OFF execution time (μs)
SUM(167)	5	When adding 1 word	632	1.13
		When adding 1000 words via *DM	144 ms	
TRSM(170)	2	When sampling 1 point + 0 word	18.0	
		When sampling 12 points + 3 words	42.0	
EMBC(171)	3	When setting a constant	3.0	0.88
		When setting *DM	4.5	
CCL(172)	2	---	2.1	0.75
CCS(173)	2	---	1.6	
MARK(174)	3	When sampling 0 words	14.8	2.25
		When sampling 2 words	20.6	
REGL(175)	3	When setting a word	4.9	0.88
		When setting *DM	5.9	
REGS(176)	3	When setting a word	8.1	
		When setting *DM	9.1	
FPD(177)	5	Without message output: execution	775	219
		Without message output: first time	809	
		With message output: execution	835	
		With message output: first time	874	
WDT(178)	3	---	7.18	0.88
DATE(179)	3	Word designation	259	
		*DM designation	262	
IORF(184)	4	When refreshing 1 word	Simplex: 23.6 Duplex: 925	1.0
		When refreshing 32 words	Simplex: 312 Duplex: 1,386	
IOSP(187)	2	---	228	0.75
IORS(188)	2	---	2.0	IL: 0.75
IODP(189)	4	---	308	1.0
READ(190)	5	When reading 1 word	Simplex: 530 Duplex: 1.53 ms	1.13
		When reading 255 words	Simplex: 3.87 ms Duplex: 5.58 ms	
WRIT(191)	5	When writing 1 word	Simplex: 551 Duplex: 1.47 ms	
		When writing 255 words	Simplex: 4.1 ms Duplex: 4.55 ms	
SEND(192)	5	---	Simplex: 304 Duplex: 1.10 ms	
RECV(193)	5	---	Simplex: 279 Duplex: 1.16 ms	
CMND(194)	5	---	Simplex: 278 Duplex: 1.14 ms	
MSG(195)	4	When specifying constant and a word	5.1	1.0
		When specifying two *DM	7.6	
CJP(221)	3	Jumping to designated word	8.75	2.38
		Jumping to designated *DM	9.88	
CJPN(222)	3	Jumping to designated word	8.75	2.25
		Jumping to designated *DM	9.88	
CNR(236)	4	When resetting 1 word	16.9	1.0
		When resetting 1000 words via *DM	519	
BPRG(250)	2	---	4.88	2.50

Instruction	Words	Conditions	ON execution time (μs)	OFF execution time (μs)	
RLNC(260)	3	Rotating word	9.63	0.88	
		Rotating *DM	11.8		
RRNC(261)	3	Rotating word	9.75		
		Rotating *DM	11.9		
RLNL(262)	3	Rotating word	10.8		
		Rotating *DM	12.9		
RLNL(263)	3	Rotating word	10.8		
		Rotating *DM	12.9		
PID(270)	5	When not sampling	Simplex: 19.1 Duplex: 828		4.50
		When sampling	Simplex: 306 Duplex: 1.12 ms		
		When executing first time	Simplex: 673 Duplex: 1.47 ms		
LMT(271)	5	Input, output word designation	14.3		1.13
		Input, output *DM designation	20.9		
BAND(272)	5	Input, output word designation	14.4		
		Input, output *DM designation	21.0		
ZONE(273)	5	Input, output word designation	14.6		
		Input, output *DM designation	21.3		
ROTB(274)	3	Word → word	34.6	1.00	
		*DM → *DM	39.0		
BINS(275)	5	Word → word	13.4	1.13	
		*DM → *DM	20.9		
BCDS(276)	5	Word → word	13.5		
		*DM → *DM	20.3		
BISL(277)	5	Word → word	17.8		
		*DM → *DM	25.4		
BDSL(278)	5	Word → word	19.4		
		*DM → *DM	26.1		
RD2(280)	5	Reading 1 word to word	16.9		4.38
		Reading 255 words *DM to *DM	21.6		
WR2(281)	5	Writing 1 word to word	17.8		4.50
		Writing 255 words *DM to *DM	22.5		
=(300)	5	Comparing constant and word	12.3	5.13	
		Comparing *DM and *DM	16.8		
=L(301)	5	Comparing constant and word	13.0		
		Comparing *DM and *DM	17.5		
=S(302)	5	Comparing constant and word	12.5		
		Comparing *DM and *DM	17.0		
=SL(303)	5	Comparing constant and word	13.3		
		Comparing *DM and *DM	17.8		
<>(305)	5	Comparing constant and word	12.3		
		Comparing *DM and *DM	16.8		
<>L(306)	5	Comparing constant and word	13.0		
		Comparing *DM and *DM	17.5		
<>S(307)	5	Comparing constant and word	12.5		
		Comparing *DM and *DM	17.0		
<>SL(308)	5	Comparing constant and word	13.3		
		Comparing *DM and *DM	17.8		

Instruction	Words	Conditions	ON execution time (μs)	OFF execution time (μs)
<(310)	5	Comparing constant and word	12.3	5.13
		Comparing *DM and *DM	16.8	
<L(311)	5	Comparing constant and word	13.0	
		Comparing *DM and *DM	17.5	
<S(312)	5	Comparing constant and word	12.5	
		Comparing *DM and *DM	17.0	
<SL(313)	5	Comparing constant and word	13.3	
		Comparing *DM and *DM	17.8	
<=(315)	5	Comparing constant and word	12.3	
		Comparing *DM and *DM	16.8	
<=L(316)	5	Comparing constant and word	13.0	
		Comparing *DM and *DM	17.5	
<=S(317)	5	Comparing constant and word	12.5	
		Comparing *DM and *DM	17.0	
<=SL(318)	5	Comparing constant and word	13.3	
		Comparing *DM and *DM	17.8	
>(320)	5	Comparing constant and word	12.3	
		Comparing *DM and *DM	16.8	
>L(321)	5	Comparing constant and word	13.0	
		Comparing *DM and *DM	17.5	
>S(322)	5	Comparing constant and word	12.5	
		Comparing *DM and *DM	17.0	
>SL(323)	5	Comparing constant and word	13.3	
		Comparing *DM and *DM	17.8	
>=(325)	5	Comparing constant and word	12.3	
		Comparing *DM and *DM	16.8	
>=(326)	5	Comparing constant and word	13.0	
		Comparing *DM and *DM	17.5	
>=S(327)	5	Comparing constant and word	12.5	
		Comparing *DM and *DM	17.0	
>=SL(328)	5	Comparing constant and word	13.3	
		Comparing *DM and *DM	17.8	
TST(350)	5	Designating constant for word bit	12.1	
		Designating constant for *DM bit	17.0	
TSTN(351)	5	Designating constant for word bit	12.0	
		Designating constant for *DM bit	16.9	
+(400)	5	Constant + word → word	13.6	1.13
		*DM + *DM → *DM	20.0	
+L(401)	5	Constant + word → word	15.4	
		*DM + *DM → *DM	22.0	
+C(402)	5	Constant + word → word	7.00	
		*DM + *DM → *DM	10.5	
+CL(403)	5	Constant + word → word	8.63	
		*DM + *DM → *DM	12.4	
+B(404)	5	Constant + word → word	14.0	
		*DM + *DM → *DM	21.4	
+BL(405)	5	Constant + word → word	24.4	
		*DM + *DM → *DM	31.6	

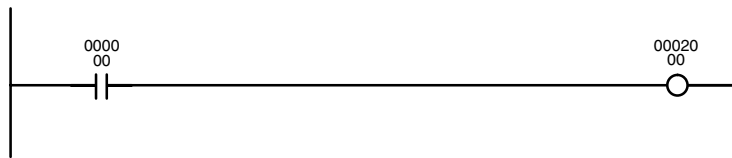
Instruction	Words	Conditions	ON execution time (μs)	OFF execution time (μs)	
+BC(406)	5	Constant + word → word	7.50	1.13	
		*DM + *DM → *DM	11.5		
+BCL(407)	5	Constant + word → word	17.9		
		*DM + *DM → *DM	21.8		
-(410)	5	Constant – word → word	13.9		
		*DM – *DM → *DM	20.5		
-L(411)	5	Constant – word → word	15.3		
		*DM – *DM → *DM	21.9		
-C(412)	5	Constant – word → word	6.88		
		*DM – *DM → *DM	11.1		
-CL(413)	5	Constant – word → word	8.63		
		*DM – *DM → *DM	12.1		
-B(414)	5	Constant – word → word	13.8		
		*DM – *DM → *DM	21.1		
-BL(415)	5	Constant – word → word	23.9		
		*DM – *DM → *DM	30.8		
-BC(416)	5	Constant – word → word	7.38		
		*DM – *DM → *DM	11.4		
-BCL(417)	5	Constant – word → word	17.5		
		*DM – *DM → *DM	21.0		
*(420)	5	Constant x word → word	20.8		
		*DM x *DM → *DM	28.6		
*L(421)	5	Constant x word → word	46.1		
		*DM x *DM → *DM	55.5		
*U(422)	5	Constant x word → word	13.5		
		*DM x *DM → *DM	17.4		
*UL(423)	5	Constant x word → word	39.0		
		*DM x *DM → *DM	45.0		
*B(424)	5	Constant x word → word	18.4		
		*DM x *DM → *DM	22.5		
*BL(425)	5	Constant x word → word	53.5		
		*DM x *DM → *DM	60.1		
/(430)	5	Constant ÷ word → word	23.1		
		*DM ÷ *DM → *DM	29.9		
/L(431)	5	Constant ÷ word → word	58.0		
		*DM ÷ *DM → *DM	64.6		
/U(432)	5	Constant ÷ word → word	15.8		
		*DM ÷ *DM → *DM	18.8		
/UL(433)	5	Constant ÷ word → word	49.8		
		*DM ÷ *DM → *DM	53.0		
/B(434)	5	Constant ÷ word → word	17.3		
		*DM ÷ *DM → *DM	20.8		
/BL(435)	5	Constant ÷ word → word	62.6		
		*DM ÷ *DM → *DM	65.9		
FIX(450)	5	Word → word	279		1.0
		*DM → *DM	280		
FIXL(451)	5	Word → word	344		
		*DM → *DM	344		

Instruction	Words	Conditions	ON execution time (μs)	OFF execution time (μs)	
FLT(452)	4	Word → word	272	1.0	
		*DM → *DM	277		
FLTL(453)	5	Word → word	335	1.13	
		*DM → *DM	340		
+F(454)	5	Constant + word → word	351	1.13	
		*DM + *DM → *DM	358		
-F(455)	5	Constant - word → word	374	1.13	
		*DM - *DM → *DM	381		
*F(456)	5	Constant x word → word	363	1.13	
		*DM x *DM → *DM	370		
/F(457)	5	Constant ÷ word → word	401	1.13	
		*DM ÷ *DM → *DM	408		
RAD(458)	5	Word → word	365	1.0	
DEG(459)	5	Word → word	365		
SIN(460)	5	Word → word	2.34 ms		
COS(461)	5	Word → word	2.23 ms		
TAN(462)	5	Word → word	3.59 ms		
ASIN(463)	5	Word → word	3.76 ms		
ACOS(464)	5	Word → word	3.85 ms		
ATAN(465)	5	Word → word	1.82 ms		
SQRT(466)	5	Word → word	742		
EXP(467)	5	Word → word	3.72 ms		
LOG(468)	5	Word → word	1.87 ms		
BEND<001>	2	---	3.88		2.38
IF<002>	3	Without operand	3.50		2.25
		With operand	6.75		
ELSE<003>	2	---	3.63	2.25	
IEND<004>	2	---	3.75		
WAIT<005>	3	Without operand	3.63	2.25	
		With operand	7.25		
EXIT<006>	3	Without operand	4.13	2.25	
		With operand	7.38		
LOOP<009>	2	---	3.50	2.13	
LEND<010>	3	Without operand	3.38	2.25	
		With operand	6.63		
BPPS<011>	3	---	4.13	2.50	
BPRS<012>	3	---	3.38	2.88	
TIMW<013>	4	Initial startup	20.5		
		Normal execution	16.6		
CNTW<014>	5	Initial startup	17.9	2.75	
		Normal execution	17.6		
TMHW<015>	4	Initial startup	20.6	2.88	
		Normal execution	17.5		

## 6-5 I/O Response Time

The I/O response time is the time it takes for the PC to output a control signal after it has received an input signal. The time it takes to respond depends on the cycle time and when the CPU receives the input signal relative to the input refresh period.

The minimum and maximum I/O response time calculations described below are for where bit 000000 is the input bit that receives the signal and bit 000200 is the output bit corresponding to the desired output point.



### 6-5-1 I/O Units Only

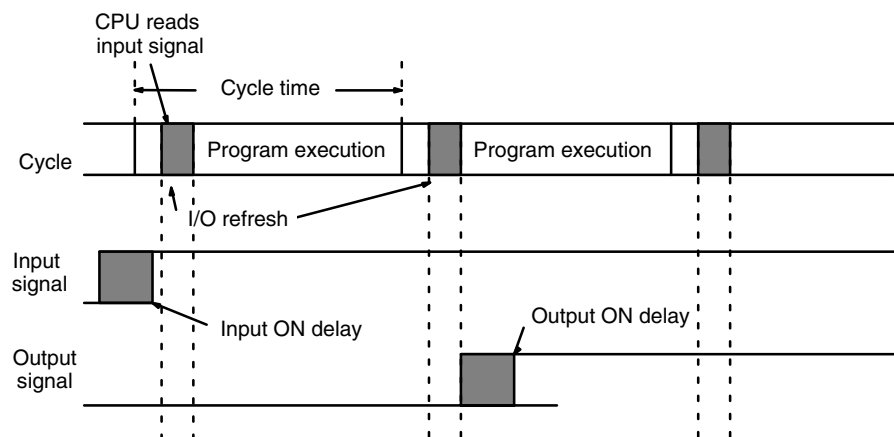
Here, we'll compute the minimum and maximum I/O response times for a PC controls only I/O Units, mounted on the CPU Rack, Expansion CPU Rack, or Expansion I/O Rack.

The data in the following table is used to produce the minimum and maximum cycle times shown calculated below.

Input ON delay	1.5 ms
Output ON delay	15 ms
Cycle time	20 ms

#### Minimum I/O Response Time

The PC responds most quickly when it receives an input signal just prior to the input refresh period in the cycle. Once the input bit corresponding to the signal has been turned ON, the program will have to be executed once to turn ON the output bit for the desired output signal and then the output refresh operation refreshes the output bit. The I/O response time in this case is thus found by adding the input ON delay time, the cycle time, and the output ON delay time. This situation is illustrated below.

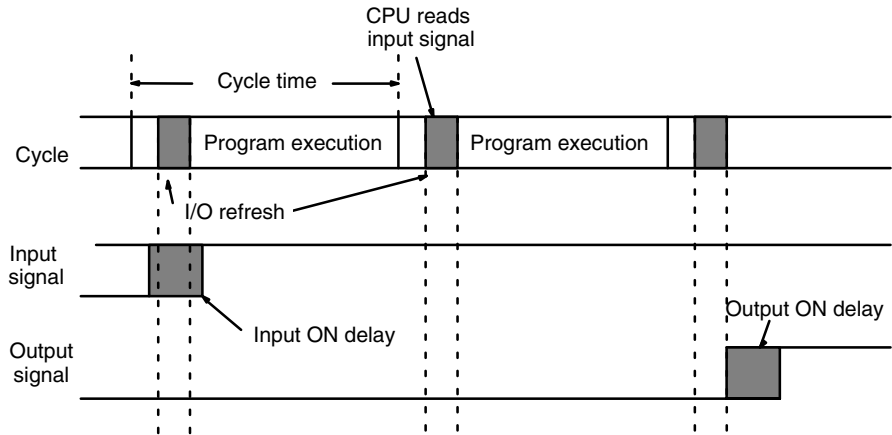


Minimum I/O response time = input ON delay + cycle time + output ON delay

Minimum I/O response time = 1.5 + 20 + 15 = 36.5 ms

**Maximum I/O Response Time**

The PC takes longest to respond when it receives the input signal just after the input refresh phase of the cycle. In this case the CPU does not recognize the input signal until the end of the next cycle. The maximum response time is thus one cycle longer than the minimum I/O response time.



Maximum I/O response time =  
input ON delay + (cycle time x 2) + output ON delay  
Maximum I/O response time = 1.5 + (20 x 2) + 15 = 56.5 ms

**6-5-2 I/O Response Times in a SYSMAC BUS System**

Here, we'll compute the minimum and maximum I/O response times for a PC controlling a SYSMAC BUS System. Both the input and output are on I/O Units connected to Slave Racks. SYSMAC BUS refreshing is carried out just after I/O refreshing as one phase of the PC cycle.

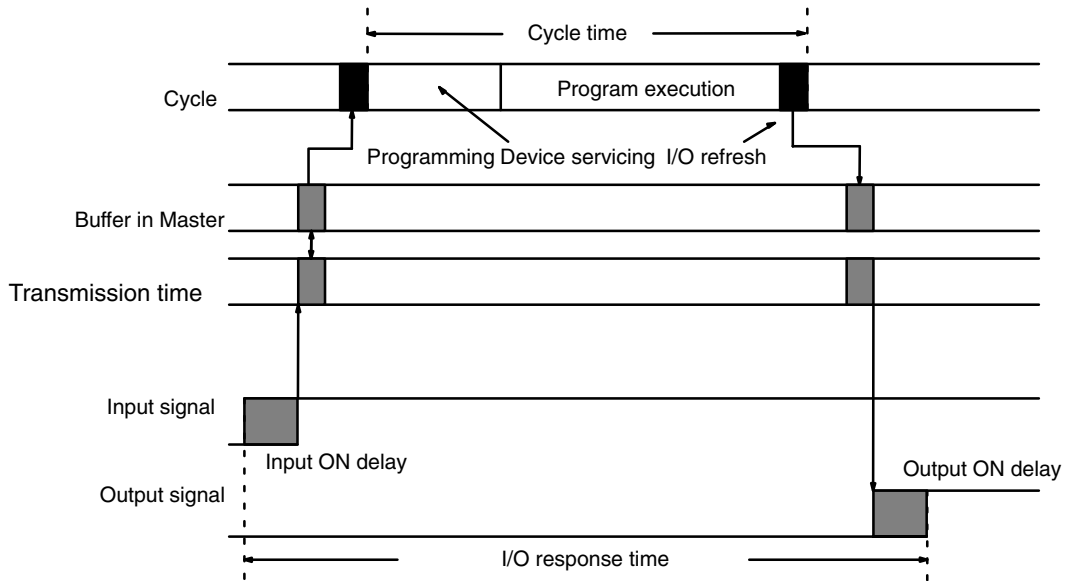
The transmission time for a Master is the sum total of the transmission times for all Slaves connected to it. The transmission time for Slave Racks is  $1.4 + (0.2 \times a)$  ms, where  $a$  is the number of I/O words on the Slave. The transmission time for I/O Terminals is  $2 \times b$  ms, where  $b$  is the number of I/O words on the I/O Terminals. The data in the following table is used to produce the minimum and maximum cycle times shown calculated below.

Input ON delay	1.5 ms
Cycle time	20 ms
Output ON delay	15 ms
Slave Rack transmission time	$1.4 + (0.2 \times 4) = 2.2$ ms
I/O Terminal transmission time	$2 \times 3 = 6$ ms
Master transmission time ( $T_{RM}$ )	$2.2$ ms + $6$ ms = $8.2$ ms



**Minimum I/O Response Time**

The PC responds most quickly when the Master receives the input signal just prior to I/O refreshing. This situation is illustrated below.

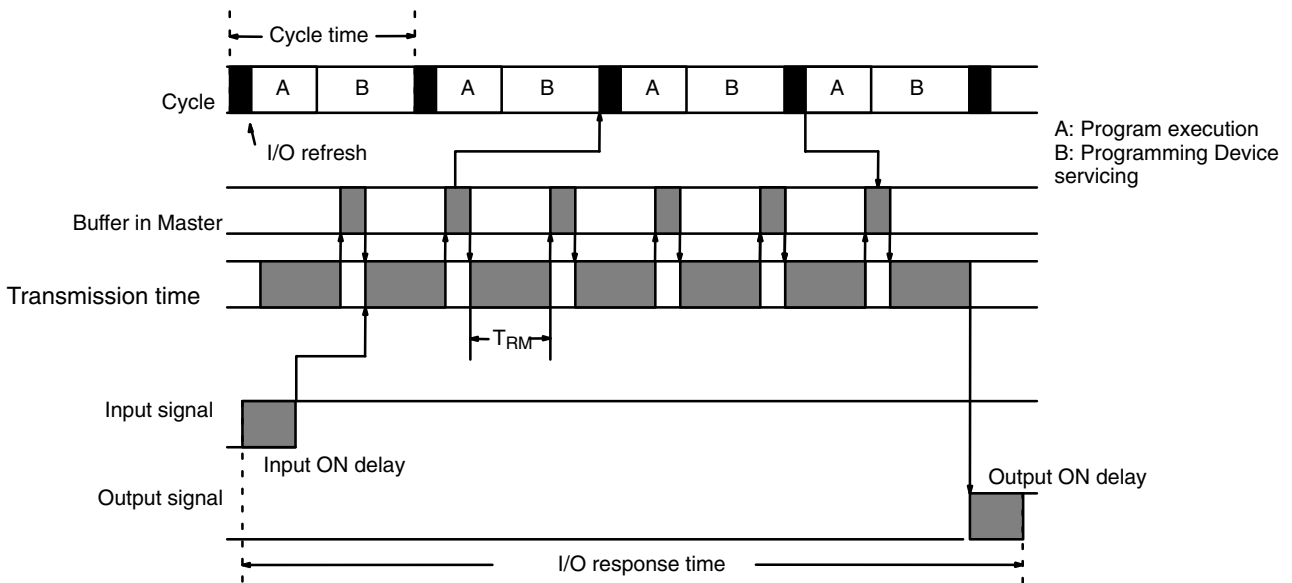


$$\text{Minimum I/O response time} = \text{input ON delay} + \text{cycle time} + \text{Slave transmission time} \times 2 + \text{output ON delay}$$

$$\text{Minimum I/O response time} = 1.5 + 20 + 2.2 \times 2 + 15 = 40.9 \text{ ms}$$

**Maximum I/O Response Time**

The PC takes longest to respond when the Master receives the input signal just after I/O refreshing. This situation is illustrated below.



$$\text{Maximum I/O response time} = \text{input ON delay} + \text{cycle time} \times 2 + \text{Master transmission time} \times 2 + \text{Slave transmission time} \times 2 + \text{output ON delay}$$

$$\text{Maximum I/O response time} = 1.5 + (20 \times 2) + (8.2 \times 2) + (2.2 \times 2) + 15 = 77.3 \text{ ms}$$

**6-5-3 I/O Response Times in a SYSMAC BUS/2 System**

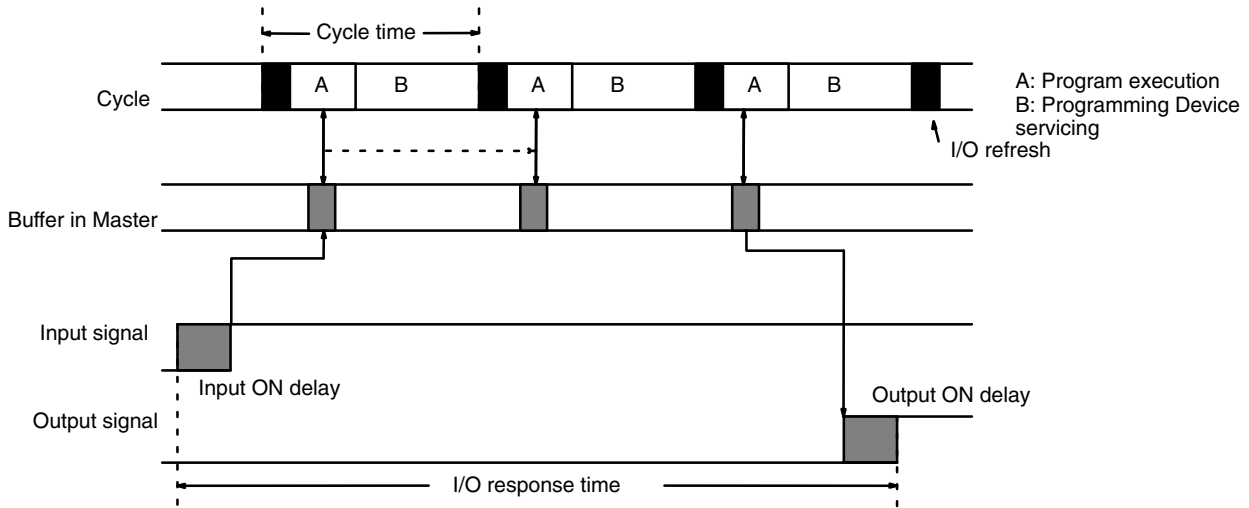
Here, we'll compute the minimum and maximum I/O response times for PC controlling a SYSMAC BUS/2 System. Both the input and output are on Units connected to Slave Racks. The PC receives data from the Master once each cycle. The Master waits to transfer to the PC after refreshing.

The data in the following table is used to produce the minimum and maximum cycle times shown calculated below.

Input ON delay	1.5 ms
Cycle time	20 ms
Output ON delay	15 ms
Communications cycle time	5 ms (one group 3, 58M Slave)

**Minimum I/O Response Time**

The PC responds most quickly when it receives an input signal just prior to SYS-MAC BUS/2 refreshing.

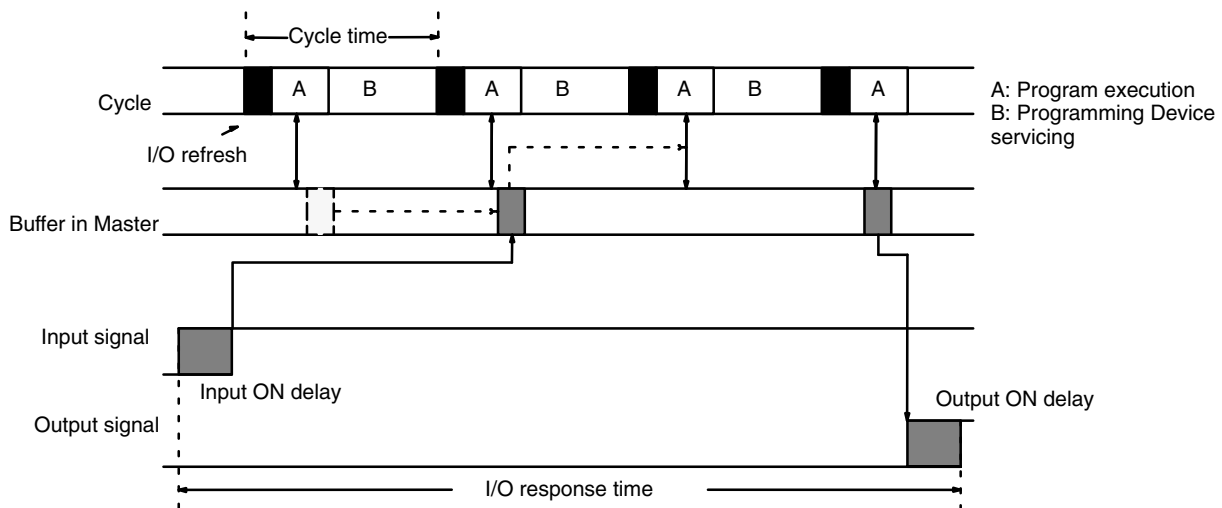


$$\text{Minimum I/O response time} = \text{input ON delay} + \text{communications cycle time} \times 5 + \text{cycle time} \times 2 + \text{output ON delay}$$

$$\text{Minimum I/O response time} = 1.5 + (5 \times 5) + (20 \times 2) + 15 = 81.5 \text{ ms}$$

**Maximum I/O Response Time**

The PC takes longest to respond when it receives the input signal just after SYS-MAC BUS/2 refreshing. In this case the CPU does not recognize the input signal until the next cycle. This situation is illustrated below.



$$\text{Maximum I/O response time} = \text{input ON delay} + \text{communications cycle time} \times 7 + \text{cycle time} \times 3 + 10 \text{ ms} + \text{output ON delay}$$

$$\text{Maximum I/O response time} = 1.5 + (5 \times 7) + (20 \times 3) + 10 + 15 = 121.5 \text{ ms}$$

# SECTION 7

## PC Setup

The tables in this section list the parameters in the PC Setup, provide examples of normal application, and provide the default values. The PC Setup can be changed from the SSS. Refer to *SSS Operation Manuals* for details changing settings. The use of each parameter in the PC Setup is described where relevant in this manual and in other CVM1D manuals.

7-1	PC Setup Overview .....	442
7-2	PC Setup Details .....	443
7-3	PC Setup Default Settings .....	446

## 7-1 PC Setup Overview

Parameter		Function	Normal application(s)
A:Hold areas	H:Hold areas	Specifies which bits are to maintain status when power is turned off.	To extend the Holding Area beyond CIO 0300.
	R:Hold bits	Specifies Racks or Masters (Remote I/O Subsystems) that are to maintain status when operation is stopped or modes are changed.	To maintain output status for specific Racks or Remote I/O Subsystems.
B:Startup hold	K:Forced Status (A00013)	Maintains the status of the Forced Status Hold Bit when power is turned off and on.	To maintain the status of bits forced ON or OFF.
	I:I/O bits (A00012)	Maintains the status of the IOM Hold Bit when power is turned off and on.	To prevent I/O status from being cleared when power is turned on.
	D:Power on flag (A00011)	Fixed at "Do not hold" in CVM1D PCs.	---
C:Startup mode		Specifies the initial PC operating mode.	To automatically start the PC when power is turned ON. Set the mode to MONITOR or RUN when using restart continuation.
D:Startup processing		Specifies whether the user program is loaded from the Memory Card when power is turned on.	To enable using a ROM Memory Card without a backup battery.
E:I/O refresh		Fixed at "cyclic refreshing" in CVM1D PCs.	---
F:Execute control 1	B:Detect low battery	Specifies detection of CPU battery errors.	To disable detection when batteries are not being used.
	S:Error on power off	Fixed at "Not an error" in CVM1D PCs.	---
	T:CPU standby	Specifies whether the CPU is to go on standby or start operation while initializing the system or detecting terminators in SYSMAC BUS/2 Systems.	
	K:Measure CPU SIOU cycle	Specifies whether or not the CPU Bus Unit servicing cycle is to be measured.	
G:Execute control 2	C:Execute process	Fixed at "synchronous operation" in CVM1D PCs.	---
	I:I/O interrupt	Not applicable in CVM1D PCs.	
	D:Power OFF interrupt	Fixed at "No power OFF interrupt" in CVM1D PCs.	---
	A:Dup action process	Not applicable in CVM1D PCs.	
	T:Step timer	Not applicable in CVM1D PCs.	
	J:Startup trace	Specifies whether a trace is to be automatically executed when power is turned on.	
	B:*DM BIN/BCD	Specifies whether indirect addresses are treated as binary (memory addresses) or BCD (data area addresses).	To enable indirectly addresses for the entire DM and EM areas by using binary addresses.
	P:Multiple use of JMP000	Specifies whether or not multiple JMP000 instructions can be programmed.	
E:Comp error process	Specifies whether I/O verification errors are to be fatal or non-fatal.		
H:Host link		Sets communications parameters for the host link interface.	These settings must be made when using the host link interface.
I:CPU bus link		Specifies whether or not CPU bus links are to be created.	To enable linking of two or more BASIC Units.
J:Scheduled interrupt		Not applicable in CVM1D PCs.	---
K:1st Rack addr		Sets the first word for each of the CPU, Expansion CPU, and Expansion I/O Racks.	To simplify word allocations, to prevent changes in allocations, or to allow for expansion without changes in allocations.

Parameter	Function	Normal application(s)
L:Group 1,2 1st addr	Sets the first word for group-1 and group-2 Slaves for each Master.	To prevent overlapping of word allocations when group-1 and group-2 Slaves require more than 50 words per Master.
M:Trans I/O addr	Sets the first word for I/O Terminals for each Master.	To separate I/O Terminal allocations from those for other Slaves.
N:Group 3, RT 1st addr	Sets the first word for each Slave Rack.	To simplify word allocations, to prevent changes in allocations, or to allow for expansion without changes in allocations.
O:CV-SIOU 1st addr	Not used at present.	---
P:Power break	Fixed at 0 (no momentary power interruptions) in CVM1D PCs.	---
Q:Cycle time	Sets a minimum cycle time.	To eliminate irregular I/O delays.
R:Watch cycle time	Sets a maximum cycle time.	To stop operation when a specified cycle time is exceeded or to enable longer cycle times by setting a high maximum.
S>Error log	Sets the number of records recorded and the words in which they are recorded.	To increase the number of error records that are maintained.
T:IOIF, RT display	Sets the startup display mode for the 7-segment displays on I/O Control Units, I/O Interface Units, and SYSMAC BUS/2 Slave Racks.	

## 7-2 PC Setup Details

Name		Operation
A:Hold areas	Hold area	The status of bits specified here will be maintained when power is turned off and on. The holding bits can be set in any continuous range between CIO 1000 to CIO 2399. Be sure to perform the Create I/O Table operation or turn the power off and on after changing this parameter. Do not specify any bits allocated to I/O points on Remote I/O Units. Outputs on Remote I/O Units will remain on after program execution stops if they are in the hold area. (Default: CIO 1200 to CIO 1499)
	Hold bits	The I/O status on Racks specified here or in all Slaves connected to Masters specified here will be maintained when operation is stopped or when PC operating modes are changed. Status will not be maintained for these outputs when power is turned off. Be sure to perform the Create I/O Table operation or turn the power off and on after changing this parameter. (Default: nothing held)
B:Startup hold	Forced Status Hold Bit status (A00013) (Forced status)	Specify whether the status of the Forced Status Hold Bit is to be maintained or reset to OFF when power is turned on. If A00013 is reset, the forced ON/OFF status of all force-set and force-reset bits will be cleared when power is turned on. Changes to this setting are effective the next time the power is turned ON. (Default: A00013 turned OFF)
	IOM Hold Bit status (A00012) (I/O bits)	Specify whether the status of the IOM Hold Bit is to be maintained or reset to OFF when power is turned on. If A00012 is reset, the CIO Area, Transition Flags, Timer Flags, Timer PVs, index registers, data register, and the Current EM bank number will be cleared when power is turned on. Changes to this setting are effective the next time the power is turned ON. (Default: A00012 turned OFF)
	Restart Continuation Bit status (A00011) (Power on flag)	This setting is fixed at "Do not hold" (A00011 turned OFF) in CVM1D PCs.
C:Startup mode		Designate the PC operating mode to be set when PC power is turned ON. Changes to this setting are valid the next time the power is turned ON. (Default: PROGRAM)

Name		Operation
D:Startup processing		Designate whether the user program (AUTOEXEC.OBJ) is automatically transferred from the Memory Card to PC memory when the power is turned ON. If this parameter is set to transfer the program, the program will be transferred regardless of the PC's startup mode setting. Changes to this setting are effective the next time the power is turned ON. DIP switch pin #5 on the CPU can be turned ON to transfer both the user program (AUTOEXEC.OBJ) and the PC setup (AUTOEXEC.STD). Refer to information on the Memory Card for details. (Default: Don't transfer)
E:I/O refresh		This setting is fixed at "cyclic refreshing" in CVM1D PCs.
F:Execute control 1	Detect low battery	Designate whether battery errors are detected. Changes to this setting are effective immediately. (Default: Detect)  The following bits will be turned ON when a battery error is detected. A40204 Battery Low Flag (PC or Memory Card) A42614 Memory Card Battery Low Flag A42615 PC Battery Low Flag
	Error on power off	This setting is fixed at "Not an error" in CVM1D PCs.
	CPU standby	Designate whether PC operation will begin or the CPU will standby during initialization and until SYSMAC BUS/2 terminators are properly detected. If this parameter is set for operation, PC operation will continue even if SYSMAC BUS/2 terminators aren't detected. Changes to this setting are effective immediately. (Default: CPU standby)
	Measure CPU-bus Unit (CPU SIOU) cycle	Designate whether or not the time between CPU-bus Unit services is to be measured. If measured, the cycle is stored starting at A310. Changes to this setting are effective immediately. (Default: Don't measure cycle)
G:Execute control 2	Execute process	This setting is fixed at "synchronous operation" in CVM1D PCs. (This setting will be changed to "asynchronous" when the PC Setup is initialized with SSS, but the setting is ignored in CVM1D operation.)
	I/O interrupts	Not applicable in CVM1D PCs.
	Power OFF interrupt	This setting is fixed at "No power OFF interrupt" in CVM1D PCs.
	Dup action process	SFC instructions are not used in CVM1D PCs.
	Step timer	SFC instructions are not used in CVM1D PCs.
	Startup trace	Designate whether a trace is executed automatically according to the preset conditions when the power is turned on or the operating mode is changed. Changes to this setting are effective the next time power is turned on. (Default: Don't start trace.)
	Indirect DM binary/BCD (*DM BIN/BCD)	Designate whether indirect DM and EM addresses are binary (PC memory addresses) or BCD (DM and EM area addresses). Changes to this setting are effective immediately. (Default: BCD)
	Multiple use of JMP000	Specify whether multiple JMP000 instructions can be programmed. Changes to this setting are effective immediately. (Default: Multiple use of JMP000 enabled)
	Comparison error process	Designate whether or not to start operation even though an I/O verification error has occurred. This setting affects only the start of PC operation. The I/O verification error is non-fatal, so PC operation will continue if an I/O verification error occurs. Changes to this setting are effective immediately. (Default: Run after error)

Name		Operation
H:Host link	Baud rate	Designate 1200, 2400, 4800, 9600, or 19200 bps. (Default: 9600 bps)
	Stop bits	Designate either 1 stop bit or 2 stop bits. (Default: 2 stop bits)
	Parity	Designate even, odd, or no parity. (Default: Even parity)
	Data length (Data bits)	Designate either 7-bit or 8-bit data. (Default: 7-bit data)
	Unit number	Designate the unit number between 00 and 31. The unit number must not be the same as the unit number of another node in an RS-422 host link. Changes to this setting are effective immediately. (Default: 00)
I:CPU bus link		Designate whether or not CPU bus links are used. CPU bus links are used between BASIC Units only. The CPU bus link service interval is comparable to the cycle time. Changes to this setting are effective immediately. (Default: Don't use CPU bus link)
J:Scheduled interrupt interval		Scheduled interrupts cannot be processed in CVM1D PCs.
K:1st Rack addr		Designate the first CIO words allocated to the CPU, Expansion CPU, and Expansion I/O Rack. The first word can be set between 0 and 511. Do not allow word allocations to overlap. Racks without a designated first word will be allocated words automatically beginning from CIO 0000. Perform the Create I/O Table or Change I/O Table operation or turn the power off and on after changing this setting. (Default: Automatic allocation by rack number beginning from CIO 0000)
L:Group 1,2 1st addr		Designate the first words between CIO 0000 and CIO 0999 for each SYSMAC BUS/2 group-1 and group-2 Masters. The default first word will be used for Masters without a designated first word. Perform the Create I/O Table or Change I/O Table operation or turn the power off and on after changing this setting. (Default: See the table on page 7-3 for details.)
M:Trans I/O addr		Designate the first word between CIO 0000 and CIO 2555 for each Master for SYSMAC BUS I/O Terminals. Do not designate any bits that are in the hold area. Outputs on I/O Terminals will remain on after program execution stops if they are in the hold area.  The default first word will be used for Masters without a designated first word. This setting does not change the Slave address. Perform the Create I/O Table or Change I/O Table operation or turn the power off and on after changing this setting. (Default: 32 words per I/O Terminal starting from CIO 2300)
N:Group 3, RT 1st addr		Designate the first word for each SYSMAC BUS/2 group-3 Slave between CIO 0000 and CIO 0999 and for each SYSMAC BUS Slave Rack between CIO 0000 and CIO 2555. Do not designate any bits that are in the hold area. Outputs on Slaves will remain on after program execution stops if they are in the hold area.  The default first word will be used for Slaves without a designated first word. Perform the Create I/O Table or Change I/O Table operation or turn the power off and on after changing this setting. (Default: See the table on page 7-3 for details.)
O:CV-SIOU 1st addr		Not used.
P:Power break		This setting is fixed at 0 ms (no momentary power interruptions) in CVM1D PCs.
Q:Cycle time		Set the minimum cycle time to between 0 and 32,000 ms. If the actual cycle time is less than the set cycle time, execution will be halted until the set cycle time elapses before the next cycle is executed. If the actual cycle time exceeds the set cycle time, the setting is ignored and the next cycle is executed when the current cycle is complete. Changes to this setting are effective immediately.  The actual cycle time might vary 3 to 4 ms from the set cycle time. (Default: Variable cycle)

Name	Operation
R:Watch cycle time	Designate the maximum cycle time between 10 and 40,000 ms. If the cycle time exceeds the designated value, a fatal error will occur and A40108 will be turned ON (Cycle Time Too Long Flag). The actual maximum cycle time might vary about 5 ms from the designated value. Changes to this setting are effective immediately. (Default: 1,000 ms)
S>Error log	Designate the size and range of the error log area. When a error occurs, information about the error is saved in this memory area together with the time that the error occurred. The error log can be allocated in the DM or EM Area. Up to 2,047 errors can be recorded. Changes to this setting are effective the next time the power is turned ON. (Default: 20 records of 5 words each in A100 to A199)
T:IOIF, RT display	Designate the display mode to be used for the 7-segment displays on I/O Interface Units, the I/O Control Unit, and SYSMAC BUS/2 Remote I/O Slave Units when the power is turned ON. Changes to this setting are effective the next time the power is turned ON. (Default: Mode 1)

### 7-3 PC Setup Default Settings

Parameter		Default value
A:Hold areas	H:Hold areas	CIO 1200 to CIO 1499
	R:Hold bits	Nothing held.
B:Startup hold	K:Forced Status	Reset at startup.
	I:I/O bits	Reset at startup.
	D:Power on flag	Reset at startup. (Fixed)
C:Startup mode		PROGRAM
D:Startup processing		Don't transfer program.
E:I/O refresh		Cyclic refreshing (Fixed)
F:Execute control 1	B:Detect low battery	Detect
	S>Error on power off	Not an error (Fixed)
	T:CPU standby	CPU waits
	K:Measure CPU SIOU cycle	Don't measure cycle.
G:Execute control 2	C:Execute process	Synchronous (Fixed) (This setting is changed to "asynchronous" when the PC Setup is initialized with SSS, but the setting is ignored in CVM1D operation.)
	I:I/O interrupt	Nesting
	D:Power OFF interrupt	Disable (Fixed)
	A:Dup action process	Error
	T:Step timer	Set to 0.1 s
	J:Startup trace	Don't start trace.
	B:*DM BIN/BCD	BCD
	P:Multiple use of JMP000	Enabled
E:Compare error process	Run after error	
H:Host link	B:Baud rate	9600 bps
	S:Stop bit	2 bits
	P:Parity	Even
	D>Data bits	7 bits
	G:Unit #	Unit number 0
I:CPU bus link		Don't use CPU Bus Link.
J:Scheduled interrupt		Scheduled interrupts are not used in CVM1D PCs.
K:1st Rack addr (First words for local racks)		0 for CPU Rack



Parameter	Default value																				
L:Group 1,2 1st addr (First words for SYSMAC BUS/2 Slaves)	<table border="0" style="width:100%"> <tr> <td style="text-align:center"><u>RM0</u></td> <td style="text-align:center"><u>RM1</u></td> <td style="text-align:center"><u>RM2</u></td> <td style="text-align:center"><u>RM3</u></td> </tr> <tr> <td colspan="4">Group 1: CIO 0200 CIO 0400 CIO 0600 CIO 0800</td> </tr> <tr> <td colspan="4">Group 2: CIO 0250 CIO 0450 CIO 0650 CIO 0850</td> </tr> </table>	<u>RM0</u>	<u>RM1</u>	<u>RM2</u>	<u>RM3</u>	Group 1: CIO 0200 CIO 0400 CIO 0600 CIO 0800				Group 2: CIO 0250 CIO 0450 CIO 0650 CIO 0850											
<u>RM0</u>	<u>RM1</u>	<u>RM2</u>	<u>RM3</u>																		
Group 1: CIO 0200 CIO 0400 CIO 0600 CIO 0800																					
Group 2: CIO 0250 CIO 0450 CIO 0650 CIO 0850																					
M:Trans I/O addr (First words for I/O Terminals)	<table border="0" style="width:100%"> <tr> <td style="text-align:center"><u>RM0</u></td> <td style="text-align:center"><u>RM1</u></td> <td style="text-align:center"><u>RM2</u></td> <td style="text-align:center"><u>RM3</u></td> </tr> <tr> <td>CIO 2300</td> <td>CIO2332</td> <td>CIO 2364</td> <td>CIO2396</td> </tr> <tr> <td colspan="4"> </td> </tr> <tr> <td style="text-align:center"><u>RM4</u></td> <td style="text-align:center"><u>RM5</u></td> <td style="text-align:center"><u>RM6</u></td> <td style="text-align:center"><u>RM7</u></td> </tr> <tr> <td>CIO 2428</td> <td>CIO 2460</td> <td>CIO 2492</td> <td>CIO 2524</td> </tr> </table>	<u>RM0</u>	<u>RM1</u>	<u>RM2</u>	<u>RM3</u>	CIO 2300	CIO2332	CIO 2364	CIO2396					<u>RM4</u>	<u>RM5</u>	<u>RM6</u>	<u>RM7</u>	CIO 2428	CIO 2460	CIO 2492	CIO 2524
<u>RM0</u>	<u>RM1</u>	<u>RM2</u>	<u>RM3</u>																		
CIO 2300	CIO2332	CIO 2364	CIO2396																		
<u>RM4</u>	<u>RM5</u>	<u>RM6</u>	<u>RM7</u>																		
CIO 2428	CIO 2460	CIO 2492	CIO 2524																		
N:Group 3, RT 1st addr (First words for group-3 Slave Racks)	<p>Group 3 (SYSMAC BUS/2):</p> <table border="0" style="width:100%"> <tr> <td style="text-align:center"><u>RM0</u></td> <td style="text-align:center"><u>RM1</u></td> <td style="text-align:center"><u>RM2</u></td> <td style="text-align:center"><u>RM3</u></td> </tr> <tr> <td>CIO 0300</td> <td>CIO 0500</td> <td>CIO 0700</td> <td>CIO 0900</td> </tr> </table> <p>Words allocated to Units in order under each Master.</p> <p>RT (SYSMAC BUS): Defaults for SYSMAC BUS Slaves are the same as for I/O Terminals (see above). Words allocated to Units in order under each Master.</p>	<u>RM0</u>	<u>RM1</u>	<u>RM2</u>	<u>RM3</u>	CIO 0300	CIO 0500	CIO 0700	CIO 0900												
<u>RM0</u>	<u>RM1</u>	<u>RM2</u>	<u>RM3</u>																		
CIO 0300	CIO 0500	CIO 0700	CIO 0900																		
O:CV-SIOU 1st addr	Not used at present.																				
P:Power break (Momentary power interruption time)	0 ms (Fixed)																				
Q:Cycle time	Cycle variable																				
R:Watch cycle time (Cycle time monitoring time)	1,000 ms																				
S>Error log	20 records in A100 through A199																				
T:IOIF, RT display (Slave display modes at startup)	Mode 1																				

# Index

## A

acronym, definition, 27  
address tracing. *See* tracing  
addresses  
  data area, description, 27  
  memory, description, 27  
addressing, indirect. *See* indirect addressing  
arithmetic flags, 57, 102  
  recording status, 355  
  retrieving status, 354  
ASCII, converting data, 222  
auto-increments, with Index and Data Registers, 63  
auto-decrements, with Index and Data Registers, 63  
Auxiliary Area, 41–59

## B

BASIC Units  
  disabling read/write access, 385  
  enabling read/write access, 386  
  I/O allocations, 41  
  servicing, 419  
BCD  
  calculations, 261, 268, 274, 278  
  version-1/2 CPUs, 237–248  
  converting, 28  
  definition, 28  
binary  
  calculations, 249, 259, 263, 272, 276  
  definition, 28  
  signed binary, 30  
  unsigned binary, 29  
bits, controlling, 112  
blackout, power. *See* power interruptions  
block programming, 68  
block programming instructions, 400  
branching  
  *See also* ladder diagrams  
  block programs, 402  
brownout, power. *See* power interruptions  
bus bar, definition, 67

## C

Calendar/Clock Area, 40  
CIO Area, 31–39  
clock, 40

  adding to clock time, 339  
  compensation, 341  
  setting, 9  
  subtracting from clock time, 340  
clock pulse bits, 58  
commands, delivering commands through a network, 392  
compatible devices, 7  
complements, calculating, 336  
CompoBus/D, memory areas, 38  
conditions, definition, 67  
connecting  
  cables, 7  
  Programming Devices, 8  
constants, operands, 101  
control bits  
  CPU Bus Unit Restart Bits, 47  
  CPU Service Disable Bits, 48, 420  
  definition, 27  
  Error Log Reset Bit, 47  
  Forced Status Hold Bit, 46  
  Host Link Service Disable Bit, 49, 420  
  I/O Refresh Disable Bit, 420  
  IOM Hold Bit, 46  
  Output OFF Bit, 47  
  Programming Device Service Disable Bit, 49, 420  
  Sampling Start Bit, 48, 374, 376  
  Service Disable Bits, 48, 419  
  SYSMAC BUS Error Check Bits, 47, 55  
  SYSMAC BUS Refresh Disable Bit, 49  
  Trace Start Bit, 48, 374, 376  
control systems, 3  
  design, 5  
controlled systems, 3  
Controller Link System  
  communications, 395–400  
  data links, 37  
converting. *See* data, converting  
Counter Area, 60  
counters, 60, 127  
  block programs, 410  
  conditions when reset, 144  
  Counter Completion Flags, 60  
  counter numbers, 60, 127  
  counter present values, 60  
  creating extended timers, 142  
  extended counters, 142  
  instructions, 127  
  PV and SV, 127  
  resetting with CNR(236), 145  
  reversible counter, 144  
CPU Bus Link Area, 39  
CPU Bus Unit Area, 38  
CPU Bus Units

*See also* Units  
definition, 38  
disabling service, 48, 420  
Duplication Error Flags, 54  
Error Flags, 53, 56  
I/O allocation, 38  
Initializing Flags, 50  
Initializing Wait Flag, 50  
service interval, 51  
servicing, 419  
Setting Error Flag, 54  
Unit Number Setting Error Flag, 57

CPU Unit  
memory, 21  
operation, during power interruption, 417  
operational flow, 414  
processing after power interruptions, 416  
switches, 12  
synchronous operation, 415, 419

CY. *See* flags, CY

cycle, First Cycle Flag, 58

cycle times, 418  
calculating, 418  
examples, 423–424  
Cycle Time Too Long Flag, 53  
instruction execution times, 425  
maximum since start-up, 57  
operations significantly increasing, 422  
present, 57  
Programming Device servicing, 51

## D

data  
comparison instructions, 193–206  
converting, 29, 207–237  
floating-point data, 280  
radians and degrees, 290, 291  
decrementing, 301–305  
formats, 90  
incrementing, 301–305  
moving, 175–193  
retention  
in DM Area, 60  
in EM Area, 60  
searching, 353  
shifting, 147–174  
tracing. *See* tracing  
transfers, 21

data areas  
definition, 27  
structure, 27

Data Link Area, 41

Data Registers, 62  
copying current contents, 356  
loading data, 355

data tracing. *See* tracing

dead-band control, 327

dead-zone control, 328

DEBUG mode, description, 6

decimal points, 29

decrementing, 301–305

definers, definition, 100

differentiated instructions, 103  
function codes, 100

digit numbers, 28

DIP switch, 13

display, I/O Control and I/O Interface Units, 22  
outputting characters, 350

DM Area, 60–62

DR. *See* Data Registers

duplex, initialization, 420

Duplex Startup Error Flags, 51

Duplex System  
Duplex Alternate CPU Waiting At Startup Flag, 51  
Duplex Bus Error At Startup Flag, 51  
Duplex Bus Error Flag, 54  
Duplex CPU Error Flag, 51  
Duplex CPU Memory Error Flag, 52  
Duplex CPU Replacement Enabled Flag, 51  
Duplex Power Supply Error Flag, 56  
Duplex Verification Error At Startup Flag, 51  
Duplex Verification Error Flag, 52  
EM Size Mismatch Flag, 52  
PC Setup Mismatch Flag, 52  
PC Setup Transfer Error Flag, 52  
Program Mismatch Flag, 52

Duplex Unit, switches, 14

## E

EC Directives, xv

EM Area, 60–62

EM Unit, 17  
current bank number, 59  
main components, 17  
selecting bank number, 352

EQ. *See* flags, EQ

ER. *See* flags, Instruction Execution Error

Error Log, clearing, 9

Error Log Area, 49

errors  
codes, 53  
Error Log Area, 49  
programming, 342  
failure point detection, 344  
Instruction Execution Error Flag, 57  
messages, programming, 386  
programming messages, 386

event processing, potential problems, 423

execution condition, definition, 68

execution time, instructions, 425–436

Expansion Data Memory Unit. *See* EM Unit

exponents, 299

Extended PC Setup, definition, 19

## F

failure point detection, 344

FAL, area, 343

FAL number, 56

FALS Flag, 53

files

file memory, 18

ladder program, automatic transfer at start-up, 20

specifying, 21

transferring to/from Memory Card, 19

types of Memory Card files, 19

flags

Accessing Memory Card Flag, 51

arithmetic, 57

programming example, 194

Battery Low Flags, 54

Counter Completion Flags, 60, 127

CPU Bus Error Flag, 53

CPU Bus Link Error Flag, 57

CPU Bus Unit Error Flag, 56

CPU Bus Unit Initializing Flags, 50

CPU Bus Unit Initializing Wait Flag, 50

CPU Bus Unit Number Setting Error Flag, 57

CPU Bus Unit Setting Error Flag, 54

CPU Rack Left PSU Error Flag, 52

CPU Rack Left PSU OFF Flag, 52

CPU Rack Right PSU Error Flag, 52

CPU Rack Right PSU OFF Flag, 52

CY, 58

clearing, 238

setting, 238

Cycle Time Too Long Flag, 53

definition, 27

Differentiate Monitor Completed Flag, 47

Duplex Alternate CPU Waiting At Startup Flag, 51

Duplex Bus Error At Startup Flag, 51

Duplex Bus Error Flag, 54

Duplex CPU Error Flag, 51

Duplex CPU Memory Error Flag, 52

Duplex CPU Replacement Enabled Flag, 51

Duplex EM Size Mismatch Flag, 52

Duplex PC Setup Mismatch Flag, 52

Duplex PC Setup Transfer Error Flag, 52

Duplex Power Supply Error Flag, 56

Duplex Verification Error At Startup Flag, 51

Duplex Verification Error Flag, 52

Duplication Error Flag, 54

EM Installed Flag, 59

EM Status Flags, 59

EQ, 58

Execution Time Measured Flag, 47, 376

FAL Flag, 56

FALS Flag, 53

First Cycle Flag, 58

GR, 58

I/O Bus Error Flag, 54

I/O Setting Error Flag, 53

I/O Verification Error Flag, 56

I/O Verification Error Wait Flag, 50

Indirect DM BCD Error Flag, 56

Instruction Execution Error, 57

Jump Error Flag, 56

LE, 58

Memory Card flags, 51

Memory Card Format Error Flag, 51

Memory Card Protected Flag, 51

Memory Card Start-up Transfer Error Flag, 57

Memory Error Flag, 54

Message Flags, 49

N, 58

Network Status Flags, 59

Of, 58

Online I/O Replacement In Progress Flag, 53

Online I/O Replacement Status Retention Flag, 53

Overflow Flag, 46

Power Interruption Flag, 54

Power Supply Unit Error Flags for Racks #2 through #7, 52

Program Error Flag, 53

Program Mismatch Flag, 52

Programming Device Connected Flag, 51

Programming Device flags, 51

Rack #1 Left PSU Error Flag, 52

Rack #1 Right PSU Error Flag, 52

SSS Flags, 47

Start-up Wait Flag, 50

Step Flag, 58

Stop Monitor Completed Flag, 47

Stop Monitor Flag, 47

SYSMAC BUS Error Flag, 55

SYSMAC BUS Terminator Wait Flag, 50

SYSMAC BUS/2 Error Flag, 55

SYSMAC BUS/2 Programming Device Flags, 51

Timer Completion Flags, 59

Too Many I/O Points Flag, 53

Trace Busy Flag, 48, 374, 376

Trace Completed Flag, 47, 374, 376

Trace Trigger Monitor Flag, 47, 374, 376

Uf, 58

Underflow Flag, 46

floating-point data, 90

*See also* mathematics

division, 313

exponents, 299

logarithms, 300

square roots, 298

floating-point instructions, 280

function codes, 100

## G-H

GR. *See* flags, GR

hexadecimal, definition, 28

Hold Area, 37

Host Link System

disabling read/write access, 385

disabling service, 49, 420

enabling read/write access, 386

**I**

- I/O allocations
  - displaying the first I/O word on a Rack, 22
  - example, 34
- I/O Area, 32–36
- I/O bits
  - definition, 32
  - limits, 32
- I/O Control Unit, display. *See* display
- I/O Interface Unit, display. *See* display
- I/O points, determining requirements, 5
- I/O refreshing, 416
  - immediate refreshing, 104
  - IORF(184), 350
  - synchronous operation, 415
- I/O response time, 418, 437
- I/O tables
  - changing, 36
  - creating, 32
- I/O Units. *See* Units
- I/O words
  - allocation, 32
  - definition, 32
  - limits, 32
  - reserving in I/O table, 36
  - Unit requirements, 33
- immediate refreshing, 104
- incrementing, 301–305
- Index Registers, 62
  - copying current contents, 356
  - loading data, 355
- indirect addressing
  - BCD (in DM or EM), 102
  - binary (in DM or EM), 103
  - DM and EM Areas, 61
  - with Index and Data Registers, 62
- initialization, duplex, 420
- input bits
  - application, 32
  - definition, 3
- input comparison instructions, 201
- input device, definition, 3
- input point, definition, 3
- input signal, definition, 3
- instruction lines, definition, 67
- instruction sets
  - +F(454), 286
  - F(455), 287
  - \*F(456), 288
  - /F(457), 289
  - ACOS(464), 296
  - ADB(080), 249
  - ADBL(084), 254
  - ADD(070), 238
  - ADDL(074), 243
  - AND, 70, 107
    - combining with OR, 71
  - AND LD, 72, 111
    - combining with OR LD, 75
    - using in logic blocks, 73
  - AND NOT, 70, 107
  - ANDL(134), 333
  - ANDW(130), 330
  - APR(142), 315
  - ASC(113), 222
  - ASFT(052), 151
  - ASIN(463), 295
  - ASL(060), 161
  - ASLL(064), 165
  - ASR(061), 162
  - ASRL(065), 166
  - ATAN(465), 297
  - BAND(272), 327
  - BCD(101), 208
  - BCDL(103), 210
  - BCDS(276), 232
  - BCMP(022), 196
  - BCNT(114), 224
  - BDSL(278), 236
  - BEND<001>, 401
  - BIN(100), 207
  - BINL(102), 209
  - BINS(275), 230
  - BISL(277), 234
  - BPPS<011>, 408
  - BPRG(250), 401
  - BPRS<012>, 408
  - BSET(041), 186
  - BXFR(046), 192
  - CADD(145), 339
  - CCL(172), 354
  - CCS(173), 355
  - CJP(221), 125
  - CJPN(222), 125
  - CLC(079), 238
  - CMND(194), 392, 395
  - CMP(020), 193
  - CMP(028), 205
  - CMPL(021), 195
  - CMPL(029), 205
  - CNR(236), 145
  - CNT, 140
  - CNTR(012), 144
  - CNTW<014>, 410
  - COLL(045), 191
  - COLM(116), 226
  - COM(138), 336
  - COML(139), 337
  - COS(461), 293
  - CPS(026), 203
  - CPSL(027), 204
  - CSUB(146), 340
  - DATE(179), 341
  - DCBL(097), 305
  - DEC(091), 301
  - DECB(093), 303
  - DECL(095), 304
  - DEG(459), 291
  - DIFD(014), 85, 113–114

- using in interlocks, 121
- using in jumps, 123
- DIFU(013), 85, 113–114
  - using in interlocks, 121
  - using in jumps, 123
- DIST(044), 190
- DIV(073), 242
- DIVL(077), 246
- DMPX(111), 216
- DOWN(019), 109
- DVB(083), 253
- DVBL(087), 257
- ELSE<003>, 402
- EMBC(171), 352
- END(001), 72, 106, 126
- EQU(025), 200
- EXIT<006>, 406
- EXP(467), 299
- FAL(006), 342
- FALS(007), 342
- FDIV(141), 313
- FIFO(163), 373
- FIX(450), 283
- FIXL(451), 284
- FLT(452), 285
- FLTL(453), 285
- FPD(177), 344
- HEX(117), 227
- HMS(144), 338
- IEND<004>, 402
- IF<002>, 402
- IF<002> NOT, 402
- IL(002), 82, 120–122
- ILC(003), 82, 120–122
- INBL(096), 304
- INC(090), 301
- INCB(092), 302
- INCL(094), 303
- input comparison instructions (300 to 328), 201
- IODP(189), 350
- IORF(184), 350
- IORS(188), 386
  - example, 399
- IOSP(187), 385
  - example, 399
- JME(005), 122
- JMP(004), 122
- JMP(004) and JME(005), 83
- KEEP(011), 118
  - in controlling bit status, 86
- ladder instructions, 69
- LD, 69, 107
- LD NOT, 69, 107
- LEND<010>, 407
- LIFO(162), 372
- LINE(115), 225
- LMT(271), 326
- LOG(468), 300
- LOOP<009>, 407
- MARK(174), 376
- MAX(165), 306
- MCMP(024), 199
- MCRO(156), 368
- MIN(166), 307
- MLB(082), 252
- MLBL(086), 256
- MLPX(110), 214
- MOV(030), 175
- MOVB(042), 187
- MOVD(043), 189
- MOVL(032), 177
- MOVQ(037), 182
- MOVR(036), 181
- MSG(195), 386
- MTIM(122), 138
- MUL(072), 241
- MULL(076), 245
- MVN(031), 176
- MVNL(033), 178
- NASL(056), 156
- NASR(057), 157
- NEG(104), 211
- NEGL(105), 212
- NOP(000), 126
- NOT, 67
- NOT(010), 111
- NSFL(054), 154
- NSFR(055), 155
- NSLL(058), 158
- NSRL(059), 160
- OR, 70, 108
  - combining with AND, 71
- OR LD, 73, 111
  - combining with AND LD, 75
  - use in logic blocks, 74
- OR NOT, 70, 108
- ORW(131), 331
- ORWL(135), 334
- OUT, 71, 112
- OUT NOT, 71, 112
- PID(270), 317
- PUSH(161), 371
- RAD(458), 290
- RD2(280), 379
- READ(190), 377
- RECV(193), 390, 395
- REGL(175), 355
- REGS(176), 356
- RET(152), 365, 367
- RLNC(260), 168
- ROL(062), 163
- ROLL(066), 167, 169
- ROOT(140), 310
- ROR(063), 164, 171
- RORL(067), 170
- ROTB(274), 312
- RRNL(263), 172
- RSET(017), 85, 115–116
- RSTA(048), 116–118
- SBB(081), 250
- SBBL(085), 255
- SBN(150), 365, 367
- SBS(151), 366
- SDEC(112), 219
- SEC(143), 337
- SEND(192), 388, 395
- SET(016), 85, 115–116
- SETA(047), 116–118
- SFT(050), 147
- SFTR(051), 150
- SIGN(106), 213

SIN(460), 292  
SLD(068), 173  
SNXT(009), 356  
SQRT(466), 298  
SRCH(164), 353  
SRD(069), 174  
SSET(160), 370  
STC(078), 238  
STEP(008), 356  
SUB(071), 239  
SUBL(075), 244  
SUM(167), 309  
TAN(462), 294  
TCMP(023), 198  
testing bit status, 110  
TIM, 129  
TIMH(015), 133  
TIML(121), 137  
TIMW<013>, 409  
TMHW<015>, 409  
TRSM(170), 374  
TST(350), 110  
TSTN(351), 110  
TTIM(120), 135  
UP(018), 109  
WAIT<005>, 405  
WDT(178), 349  
WR2(281), 383  
WRIT(191), 381  
WSFT(053), 153  
XCGL(035), 180  
XCHG(034), 179  
XFER(040), 185  
XFRB(038), 183  
XNRL(137), 335  
XNRW(133), 332  
XORL(136), 334  
XORW(132), 331  
ZONE(273), 328

instructions  
  block programming instructions, 400  
  combining, AND LD and OR LD, 75  
  controlling bit status  
    using KEEP (011), 86  
    using OUT and OUT NOT, 112  
  controlling execution conditions, UP(018) and DOWN(019), 109  
  differentiated instructions, 103  
  execution times, 425–436  
  floating-point data, 280  
  format, 100  
  immediate refresh instructions, 104  
  intermediate instructions, 87  
  ladder diagram instructions, 107  
  ladder instructions, 69, 107  
  logic block instructions, 107  
  logic instructions, 330  
  mnemonic code, 105  
  operands, 66  
  right-hand instructions, 67  
  symbol math instructions, 259  
  terminology, 66

interlocks, 120–122  
  converting to mnemonic code, 121

  using self-maintaining bits, 86  
intermediate instructions, 67, 87  
Interrupt Input Units, 416  
IR. *See* Index Registers

## J–L

jump numbers, 123  
jumps, 122–125  
  CJP(221) and CJPN(222), 125  
ladder diagrams  
  *See also* programming  
  automatic transfer at start-up, 20  
  branching, 79  
    IL(002) and ILC(003), 82  
    using TR bits, 80  
  complicated, 76  
  controlling bit status  
    using DIFU(013) and DIFD(014), 85, 113–114  
    using KEEP(011), 118–129  
    using OUT and OUT NOT, 71  
    using SET(016) and RSET(017), 85, 115–116  
    using SETA(047) and RSTA(048), 116–118  
  converting to mnemonic code, 68–72  
  instructions, 107–111  
  notation, 100  
  structure, 67  
  using logic blocks, 72  
ladder symbol, 101  
latching relays, using KEEP(011), 118  
LE. *See* flags, LE  
leftmost, definition, 28  
limit control, 326  
Link Area, 37  
Link Units  
  *See also* Units  
  data links, 37  
logarithm, 300  
logic blocks, 68  
  *See also* ladder diagram  
  instructions, converting to mnemonic code, 72–78  
logic instructions, 330–337

## M

MACRO, input/output areas, 50  
macros, 368  
manuals, related, 9  
mathematics  
  *See also* trigonometric functions  
  adding a range of words, 309  
  BCD calculation instructions, 261, 268, 274, 278  
    version-1/2 CPUs, 237  
  binary calculation instructions, 249, 259, 263, 272, 276  
  exponents, 299

- finding the maximum in a range, 306
- finding the minimum in a range, 307
- floating-point addition, 286
- floating-point data, 280
- floating-point division, 289, 313
- floating-point multiplication, 288
- floating-point subtraction, 287
- linear extrapolation, 316
- logarithm, 300
- square root, 298, 310, 312
- maximum cycle time, extending, 349
- memory areas, definition, 27
- memory backup battery, 14
- Memory Cards, 18–21
  - backing up data, 21
  - flags, 51
  - limitations, 416
  - mounting and removing, 18
  - power switch, 14
  - transferring files, 19
  - types, 51
  - types of Memory Card files, 19
- Memory Error, Area Location, 56
- Message Flags, 49
- messages, programming, 386
- mnemonic code
  - converting, 68–72
  - right-hand instructions, 105
- MONITOR mode, description, 6
- multiple instruction lines, 106
- multiple right-hand instructions, coding, 79

## N–O

- N. *See* flags, N
- nesting, subroutines, 367
- normally closed condition, definition, 67
- normally open condition, definition, 67
- NOT, definition, 67
- offset indirect addressing, 63
- online I/O replacement, 53
- operands, 100
  - allowable designations, 101
  - bits, 68
    - definition, 66
  - definition, 66
  - word, definition, 66
- operating modes, PC, definition, 5
- output bits
  - application, 32
  - controlling ON/OFF time, 112
  - controlling status, 85, 86
  - definition, 3
- output device, definition, 3

- output point, definition, 3
- output signal, definition, 3
- OV. *See* flags, overflow

## P

- parameters, PC Setup, 442
  - See also* PC Setup
- PC
  - definition, 3
  - modes, 5
    - flags indicating, 40
  - operation, overview, 4
- PC Setup, 9, 442
  - automatic transfer at start-up, 20
  - checking, 8
  - default settings, 446
  - Extended PC Setup, 19
- PC Status Area, 40
- periodic refreshing, disabling, 420
- PID control, 317
- PLC. *See* PC
- power
  - OFF, 417
  - ON, 417
- power interruption, 417
  - number since start-up, 48
  - time of occurrence, 48, 418
- precautions
  - general, xi
  - operand data areas, 101
  - programming, 89
- Program Area, 9
- Program Memory, 9
  - structure, 68
- PROGRAM mode, description, 6
- programming
  - basic steps, 4
  - example, using shift register, 148
  - jumps, 83
  - ladder diagrams, 67
  - pausing/restarting block programs, 408
  - precautions, 89
  - preparing data in data areas, 186
  - program capacity, 9
  - Program Error Flag, 53
  - sequencing control operation, 5
  - simplification with differentiated instructions, 114
  - using work bits, 87
  - writing, 66
- Programming Console, 7
- Programming Devices, 7
  - connecting, 8
  - disabling service, 49, 420
  - flags, 51
  - number allowed, 7
  - operations, 8



Programming Device Cycle Time, 51  
servicing, in asynchronous operation, 419

programs  
capacity, 9  
execution, 90

Protect Keyswitch, 13

PV, timers and counters, 127

## R

rack numbers  
CPU-recognized rack numbers, 57  
Duplication Error Flags, 54  
setting, 22

Racks, changing, 35

response times, I/O, 437–440

right-hand instruction, definition, 67

rightmost, definition, 28

RUN mode, description, 6

## S

scan times. *See* cycle times

self-maintaining bits, using KEEP(011), 119

seven-segment displays, converting data, 219

shift registers, 147–174  
controlling individual bits, 148

signed BCD, 90

signed binary, 90

signed binary data, 30

signed data, removing sign, 213

simplified backup function, 21

Special I/O Units. *See* Units

special math instructions. *See* mathematics

square root. *See* mathematics

SSS, 7

SSS flags, 47

stack instructions, 370

start-up  
processing, 20  
time, 48

steps  
step executions, Step Flag, 58  
step instructions, 356–364

subroutines, 365–370  
number, 365

SV, timers and counters, 127

switches  
DIP. *See* DIP switch

Memory Card power, 14  
switching, from duplex to simplex operation, 421

synchronous operation, 415, 419

SYSMAC BUS Remote I/O System  
disabling read/write access, 385  
disabling refreshing, 49, 420  
enabling read/write access, 386  
Error Flags and Check Bits, 55  
I/O allocation, 38  
I/O refreshing, 417  
I/O response time, example, 438  
reading error codes, 47  
SYSMAC BUS Area, 38

SYSMAC BUS/2 Remote I/O System  
disabling read/write access, 385  
enabling read/write access, 386  
Error Flags, 55  
I/O allocation, 37  
I/O refreshing, 417  
I/O response time, example, 439  
Slaves, outputting to the display. *See* display  
SYSMAC BUS/2 Area, 37

SYSMAC LINK System  
communications, 395–400  
data links, 37  
disabling read/write access, 385  
enabling read/write access, 386  
flags, 59  
instructions, 385

SYSMAC NET Link System  
communications, 395–400  
data links, 37  
disabling read/write access, 385  
enabling read/write access, 386  
flags, 59  
instructions, 385

SYSMAC Support Software, 7

## T

terminology, 2

time

*See also* clock  
converting time notation, 337, 338  
time instructions, 337–342

timers, 59, 127  
block programs, 409  
example using CMP(020), 194  
extended timers, 131  
flicker bits, 133  
instructions, 127  
ON/OFF delays, 131  
one-shot bits, 132  
PV and SV, 127  
resetting with CNR(236), 145  
Timer Area, 59  
Timer Completion Flags, 59  
timer numbers, 59, 127  
timer present values, 59

TR bits, 79

TR (Temporary Relay) Area, 39  
use in branching, 80

tracing, 374–375  
effect of instruction trace on cycle time, 422  
flags and control bits, 374, 376

trigonometric functions  
converting to angles, 295, 296, 297  
cosine, 293  
sine, 292  
tangent, 294

troubleshooting, failure point detection. *See* errors

## U–W

UN. *See* flags, underflow

### Units

*See also* CPU Unit; Duplex Unit; EM Units

changing configuration, 35  
CPU Bus Units, definition, 4  
definition, 3  
determining requirements, 5

I/O Control Units. *See* display

I/O Interface Units. *See* display

I/O Units, definition, 3

Interrupt Input Units, 416

Link Units, definition, 4

Power Supply Unit, 52

Special I/O Units

definition, 4

READ(190) and WRIT(191), 377

unit numbers, Duplication Error Flags, 54

words required, 33

unsigned BCD, 90

unsigned binary, 90

unsigned binary data, 29

Wait Flags, 50

watchdog timer, extending, 349

words, definition, 28

Work Area, 36

work bit, 87

definition, 36

work word, 87

definition, 36

## Revision History

A manual revision code appears as a suffix to the catalog number on the front cover of the manual.

Cat. No. W351-E1-03

↑  
— Revision code

The following table outlines the changes made to the manual during each revision. Page numbers refer to the previous version.

Revision code	Date	Revised content
1	January 1999	Original production
2	August 2001	<b>Page xiv:</b> Wording of information on mounting and dismounting changed. <b>Pages 43, 50:</b> Information on A39115 added. <b>Page 324:</b> Information on PID Instruction added. <b>Page 420:</b> Information on detecting errors added.
03	April 2003	<b>Page 40:</b> Information on G00008 (SPL/DPL Mode Flag) added. <b>Page 43:</b> Information on A39115 added. <b>Page 423:</b> Stopped time for online edit operation changed from 2.0 to 0.2 s.