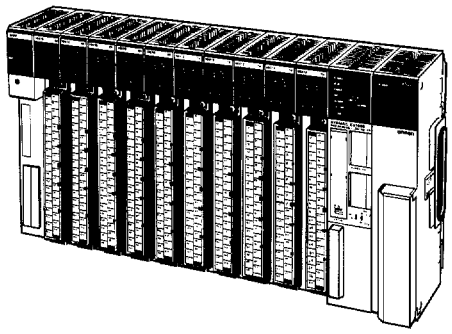
SYSMAC CV-series CV500/CV1000/CV2000/CVM1 Programmable Controllers

Operation Manual: Ladder Diagrams

Revised August 1998



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.

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

About this Manual:

This manual describes ladder diagram programming and memory allocation in the SYSMAC CV-series Programmable Controllers (PCs) (CV500, CV1000, CV2000, and CVM1). This manual is designed to be used together with two other CV-series PC operation manuals and an installation guide. The entire set of CV-series PC manuals is listed below. Only the basic portions of the catalog numbers are given; be sure you have the most recent version for your area.

Manual	Cat. No.
CV-series PC Installation Guide	W195
CV-series PC Operation Manual: SFC	W194
CV-series PC Operation Manual: Ladder Diagrams	W202
CV-series PC Operation Manual: Host Interface	W205

Programming and operating CV-series PCs are performed with the CV Support Software (CVSS), the SYSMAC Support Software (SSS), and the CV-series Programming Console for which the following manuals are available.

Product	Manuals
CVSS	The CV Series Getting Started Guidebook (W203) and the CV Support Software Operation Manuals: Basics (W196), Offline (W201), and Online (W200).
SSS	SYSMAC Support Software Operation Manuals: Basics (W247), C-series PC Operations (W248), and CVM1 Operations (W249)
CV-series Programming Console	CVM1-PRS21-E Programming Console Operation Manual (W222)

Note The CVSS does not support new instructions added for version-2 CVM1 PCs. The SSS does not support SFC programming (CV500, CV1000, or CV2000).

Please read this manual completely together with the other CV-series manuals and be sure you understand the information provide before attempting to install, program, or operate a CV-series PC. The basic content of each section of this manual is outlined below.

Section 1 gives a brief overview of the history of Programmable Controllers and explains terms commonly used in ladder-diagram programming. It also provides an overview of the process of programming and operating a PC. A list of the manuals available to use with this manual is also provided.

Section 2 provides information on hardware aspects of the CV-series PCs relevant to programming and software operation. This information is covered in more detail in the *CV-series PC Installation Guide*.

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

Section 4 explains the basic steps and concepts involved in writing a basic ladder diagram program. The entire set of instructions used in programming is described in *Section 5 Instruction Set*.

Section 5 explains each instruction in the CV-series PC instruction sets and provides the ladder diagram symbols, data areas, and flags used with each. The instructions provided by the CV-series PCs are described in following subsections by instruction group.

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. These manuals are listed at the end of *Section 1 Introduction*.

Section 7 provides tables that list the parameters in the PC Setup, provide examples of normal application, and provides the default values. The use of each parameter in the PC Setup is described where relevant in this manual and in other CV-series manuals.

Section 8 provides information on hardware and software errors that may occur during PC operation. Although described mainly in *Section 3 Memory Areas*, flags and other error information provided in the Auxiliary Area are listed in *8-5 Error Flags*.

Various appendices are also provided for convenience (see table of contents for a list).

⚠WARNING 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

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.

⚠ 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 connect to a class-3 ground (to 100 Ω or less) when installing the Units. Not connecting to a class-3 ground may result in 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 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 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.
- Resume operation only after transferring to the new CPU Unit the contents of the DM and HR Areas required for resuming operation. Not doing so may result in an unexpected operation.
- 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.

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 also provides an overview of the process of programming and operating a PC and explains basic terminology used with OMRON PCs. A list of the manuals available to use with this manual for special PC applications and products is also provided.

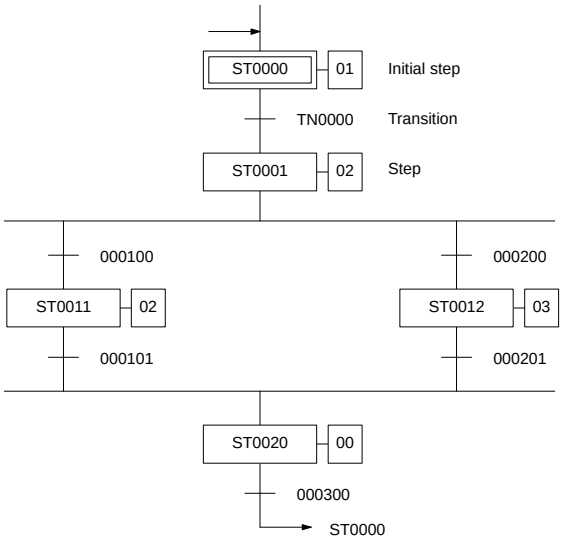
1-1 Overview

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, CV-series PCs use a form of PC logic called ladder-diagram programming. A single ladder-diagram program can be used, as in C-series PCs, but CV-series PCs are also support sequential function chart, or SFC, programming. SFC programming breaks the program into sections based on processes, greatly reducing program development and maintenance times, and allowing program sections to be easily used in other programs. The following diagram shows a simple SFC program, which consists of steps connected by lines representing the flow of execution.



The transitions between the steps control when execution moves between the steps and actions contained within the steps specify the actual executable elements of the program. Programming the actions and transitions within SFC programming are generally achieved using ladder diagrams. There are also some ladder diagram instructions that can be used to control the SFC program.

This manual is written to explain ladder-diagram programming and to prepare the reader to program and operate the CV-series PCs. SFC programming is explained in the *CV-series PCs Operation Manual: SFC*.

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 CV-series PCs are Rack PCs, there is no single product that is a CV-series 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 a 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**, **Peripheral 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 *CV-series PCs 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. Divide the required control actions into processes that need to be treated as individual sections and create an SFC program to control the flow of execution of the processes. Refer to the *CV-series PCs Operation Manual: SFC* for details on the SFC program. If desired, you can also program the PC without using an SFC program by setting the PC for ladder-only operation from the CVSS/SSS.
5. Using relay ladder symbols, write a program that represents the sequence of required operations within each process and their inter-relationships. If you are using an SFC program, you will actually be writing transition programs and action programs within the SFC program. 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*)
6. Write the program in the CVSS/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 *CVSS/SSS Operation Manual* for details.
7. Generate the I/O table with I/O Units installed. The I/O table can be generated either online from the CVSS/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 *CVSS/SSS Operation Manual* for details.
8. The PC Setup controls a variety of basic options in PC operation (such as the method of I/O refreshing and PC mode at start-up). The operating parameters in the PC Setup can be left in their default settings or changed with the CVSS/SSS as required. Refer to *Section 7 PC Setup* for details.
9. Debug the program, first to eliminate any syntax errors, and then to find execution errors. Refer to the three CVSS/SSS operation manuals for details on debugging operations. (*Section 8 Error Processing*)
10. 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.
11. Test the program in an actual control situation and carry out fine tuning as required. Refer to the CVSS/SSS operation manuals for details on debugging operations. (*Section 8 Error Processing*)
12. Record two copies of the finished program on masters and store them safely in different locations. Refer to the CVSS/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 *CVSS/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).

Control System Design

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.

Input/Output Requirements

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.

Sequence, Timing, and Relationships

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.

Unit Requirements

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 *Operation Manuals* and *System Manuals*.

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

CV-series 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 Peripheral 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.

DEBUG 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, a single step can be checked for errors from a Peripheral Device using the DEBUG operation. Output points at Output Units will remain OFF, even when the corresponding output bit is ON.

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.

1-7 Peripheral Devices

The CV Support Software (CVSS) and the SYSMAC Support Software (SSS) are the main Peripheral Device used to program and monitor CV-series PCs. You must have the CVSS/SSS to program and operate these PCs. The following Peripheral Devices are available for basic programming/monitoring.

Note The CVSS does not support new instructions added for version-2 CVM1 PCs. The SSS does not support SFC programming (CV500, CV1000, and CV2000). New instructions added for version-2 CVM1 PCs are also supported by version-1 CV-series Programming Consoles.

Graphic Programming Console

The Graphics Programming Console (GPC) can be used for monitoring and programming of PCs, but does not support SFC programming.

Programming Console

The Programming Console can be used for onsite monitoring and programming of PC, but does not support SFC programming and other advanced programming/debugging operations.

1-8 CV-series Manuals

The following manuals are available for CV-series products. Manuals are also available for compatible C-series products (see next section). Catalog number suffixes have been omitted; be sure you have the current version for your region.

Product	Manual	Cat. No.
CV-series PCs	CV-series PCs Installation Guide	W195
	CV-series PCs Operation Manual: SFC	W194
	CV-series PCs Operation Manual: Ladder Diagrams	W202
	CV-series PCs Operation Manual: Host Link System, CV500-LK201 Host Link Unit	W205
CV Support Software (CVSS)	The CV Series Getting Started Guidebook	W203
	CV Support Software Operation Manual: Basics	W196
	CV Support Software Operation Manual: Offline	W201
	CV Support Software Operation Manual: Online	W200
SYSMAC Support Software Operation Manual: Basics	SSS installation procedures, hardware information for the SSS, and general basic operating procedures (including data conversion between C-series and CVM1 PCs).	W247
SYSMAC Support Software Operation Manual: C-series PC Operations	Detailed operating procedures for the C-series PCs.	W248
SYSMAC Support Software Operation Manual: CVM1 Operations	Detailed operating procedures for CVM1 PCs.	W249
Graphic Programming Console (GPC)	CV500-MP311-E Graphic Programming Console Operation Manual	W216
Programming Console	CVM1-PRS21-E Programming Console Operation Manual	W222
SYSMAC NET Link System	SYSMAC NET Link System Manual	W213
SYSMAC LINK System	SYSMAC LINK System Manual	W212
SYSMAC BUS/2 Remote I/O System	SYSMAC BUS/2 Remote I/O System Manual	W204
CompoBus/D Device Network	CompoBus/D (DeviceNet) Operation Manual)	W267
CV-series Ethernet Unit	CV-series Ethernet System Manual	W242
BASIC Unit	BASIC Unit Reference Manual	W207
	BASIC Unit Operation Manual	W206
Personal Computer Unit	Personal Computer Unit Operation Manual	W251
	Personal Computer Unit Technical Manual	W252
Motion Control Unit	Motion Control Unit Operation Manual: Introduction	W254
	Motion Control Unit Operation Manual: Details	W255
Temperature Controller Data Link Unit	CV500-TDL21 Temperature Controller Data Link Unit Operation Manual	W244
Memory Card Writer	CV500-MCW01-E Memory Card Writer Operation Manual	W214
Optical Fiber Cable	Optical Fiber Cable Installation Guide	W156

1-9 C-series–CV-series System Compatibility

The following table shows when C-series Units can be used and when CV-series Units must be used. Any C-series Unit or Peripheral Device not listed in this table cannot be used with the CV-series PCs.

Unit		C Series	CV Series	Remarks
CPU Rack	CPU	No	**Yes**	CV500-CPU01-EV1, CV1000-CPU01-EV1, CV2000-CPU01-EV1, CVM1-CPU01-EV2, CVM1-CPU11-EV2, and CVM1-CPU21-EV2
	Power Supply	No	**Yes**	CV500-PS221, CV500-PS211, and CVM1-PA208
	CPU Backplane	No	**Yes**	CV500-BC031, CV500-BC051, CV500-BC101, CVM1-BC103, and CVM1-BC053
	I/O Control Unit	No	**Yes**	CV500-IC□01
Expansion CPU Backplane		No	**Yes**	CV500-BI111
Expansion I/O Backplane		No	**Yes**	CV500-BI042, CV500-BI062, CV500-BI112, CVM1-BI114, and CVM1-BI064 (C500 Expansion I/O Racks can be used with certain limitations.)
16-/32-/64-point I/O Units		**Yes**	**Yes**	---
Special I/O Units		**Yes**	**Yes**	Applicable Units include Analog Input, Analog Output, High-speed Counter, PID, Position Control, Magnetic Card, ASCII, ID Sensor, and Ladder Program I/O Units (The C500-ASC03 cannot be used.)
BASIC Unit		No	**Yes**	CV500-BSC□1
Personal Computer Unit		No	**Yes**	CV500-VP213-E/217-E/223-E/227-E
Temperature Control Data Link Unit		No	**Yes**	CV500-TDL21
Link Systems	SYSMAC NET	No	**Yes**	CV500-SNT31
	SYSMAC LINK	No	**Yes**	CV500-SLK11 and CV500-SLK21
	Host Link Unit	No	**Yes**	CV500-LK201
	Ethernet Unit	No	**Yes**	CV500-ETN01
Remote I/O Systems	SYSMAC BUS Units	**Yes**	**Yes**	---
	SYSMAC BUS/2	No	**Yes**	CV500-RM211/221 and CV500-RT211/221
Peripheral Devices	CV Support Software	No	**Yes** (See note.)	CV500-ZS3AT1-EV2 (3 1/2” floppy disks) and CV500-ZS5AT1-EV2 (5 1/4” floppy disks) for IBM PC/AT compatible
	SYSMAC Support Software (SSS)	**Yes**	**Yes** (See note.)	C500-ZL3AT1-E (3.5” floppy disks) for IBM PC/AT compatible
	Graphic Programming Console	**Yes (Main unit only)**	**Yes (System Cassette)** (See note.)	GPC: 3G2C5-GPC03-E System Cassette: CV500-MP311-E
	Programming Console	No	**Yes** (See note.)	CVM1-PRS21-EV1 (set)

Note The CVSS does not support new instructions added for version-2 CVM1 PCs. The SSS does not support SFC programming (CV500, CV1000, and CV2000). New instructions added for version-2 CVM1 PCs are also supported by version-1 CV-series Programming Consoles.

1-10 Networks and Remote I/O Systems

Systems that can be used to create networks and enable remote I/O are introduced in this section. Refer to the operation manuals for the Systems for details.

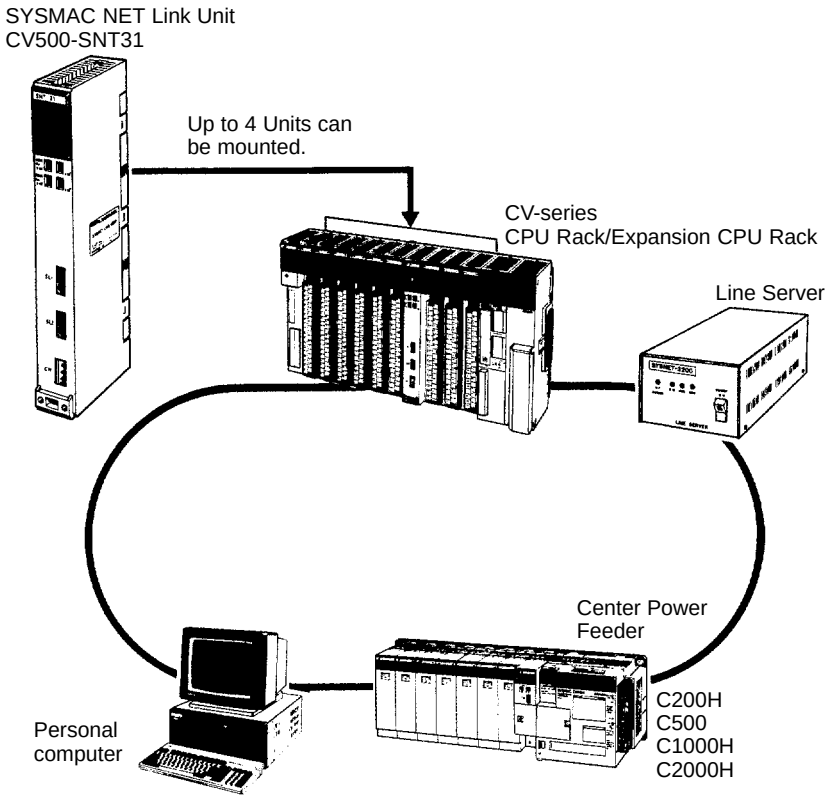
SYSMAC NET Link System

The SYSMAC NET Link System is a LAN (local area network) for use in factory automation systems. The SYSMAC NET Link System can consist of up to 128 nodes among which communications may be accomplished via datagrams, data transfers, or automatic data links.

Datagrams transmit and receive data using a command/response format. Commands can be issued from the user program by the DELIVER COMMAND instruction (CMND(194)).

Data can also be transmitted and received using the NETWORK SEND and NETWORK RECEIVE (SEND(192)/RECV(193)) instructions in the user program. Up to 256 words of data can be transferred for each instruction.

Automatic data links allow PCs and computers to create common data areas.



Note Up to four SYSMAC NET Link Units (CV500-SNT31) can be mounted to the CPU Rack and/or Expansion CPU Rack of each CV-series PC.

SYSMAC LINK System

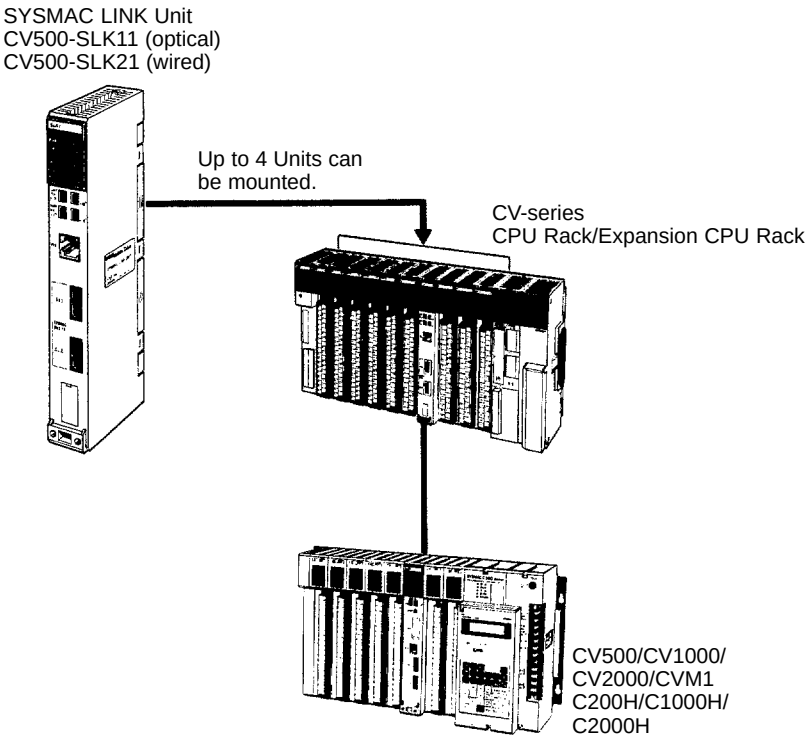
Networks can also be created using SYSMAC LINK Systems. A SYSMAC LINK System can consist of up to 62 PCs, including the CV500, CV1000, CV2000, CVM1, C200H, C1000H, and C2000H. Communications between the PCs is accomplished via datagrams, data transfers, or automatic data links in ways similar to the SYSMAC NET Link System.

The main differences between SYSMAC NET Link and SYSMAC LINK Systems is in the structure of automatic data links and in the system configuration, e.g., only PCs can be linked in SYSMAC LINK Systems, whereas other devices can form nodes in SYSMAC NET Link Systems.

Datagrams transmit and receive data using a command/response format. Commands can be issued from the user program by the DELIVER COMMAND instruction (CMND(194)).

Data can also be transmitted and received using the NETWORK SEND and NETWORK RECEIVE (SEND(192)/RECV(193)) instructions in the user program. Up to 256 words of data can be transferred for each instruction.

Automatic data links allow PCs and computers to create common data areas.



Note Up to four SYSMAC LINK Units (CV500-SLK11/21) can be mounted to the CPU Rack and/or Expansion CPU Rack of each CV-series PC.

SYSMAC BUS/2 Remote I/O System

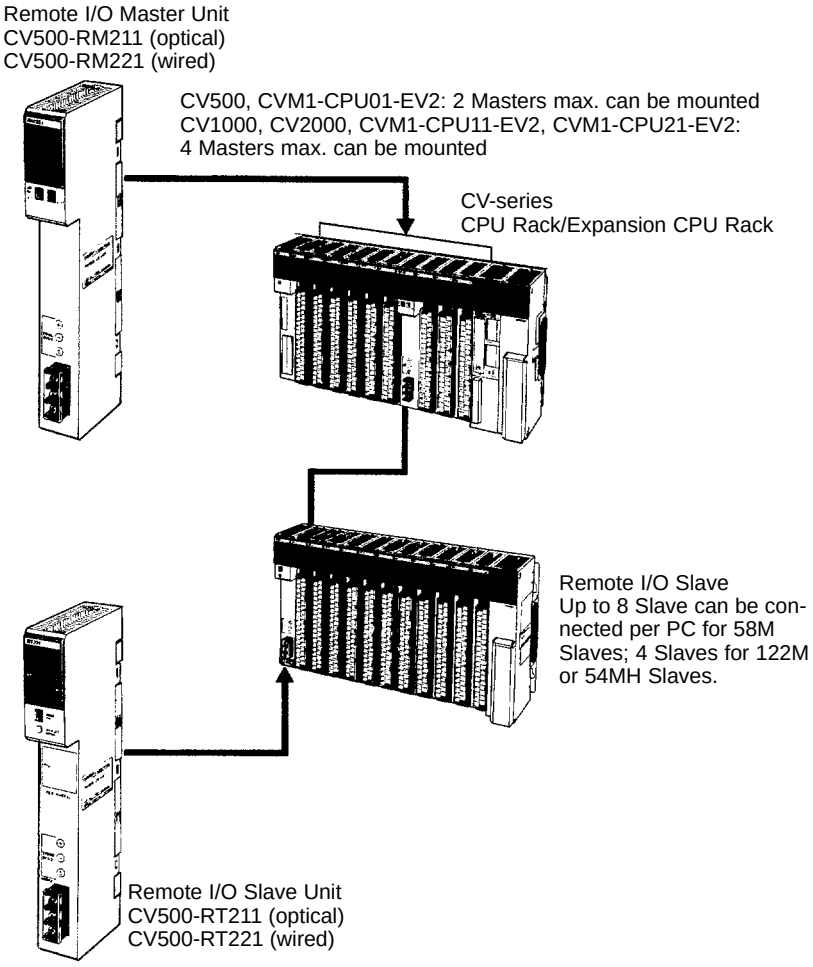
Remote I/O can be enabled by adding a SYSMAC BUS/2 Remote I/O System to the PC. The SYSMAC BUS/2 Remote I/O System is available in two types: optical and wired.

Two Remote I/O Master Units, optical or wired, can be mounted to the CV500 or CVM1-CPU01-EV2 CPU Rack or Expansion CPU Rack. Four Remote I/O Master Units can be mounted to the CV1000, CV2000, CVM1-CPU11-EV2, or CVM1-CPU21-EV2 CPU Rack or Expansion CPU Rack.

Up to eight Remote I/O Slave Racks can be connected per PC.

Slaves can be used to provide up to 1,024 remote I/O points for the CV500 or CVM1-CPU01-EV2, and up to 2,048 remote I/O points for the CV1000, CV2000, CVM1-CPU11-EV2, or CVM1-CPU21-EV2.

A Programming Device (such as the CVSS/SSS) can be connected to up to two Remote I/O Slave Units for each Remote I/O Master Unit as long as a total of no more than four Programming Devices are connected per PC.



SYSMAC BUS Remote I/O System

Remote I/O can also be enabled by using the C-series SYSMAC BUS Remote I/O System with CV-series PC.

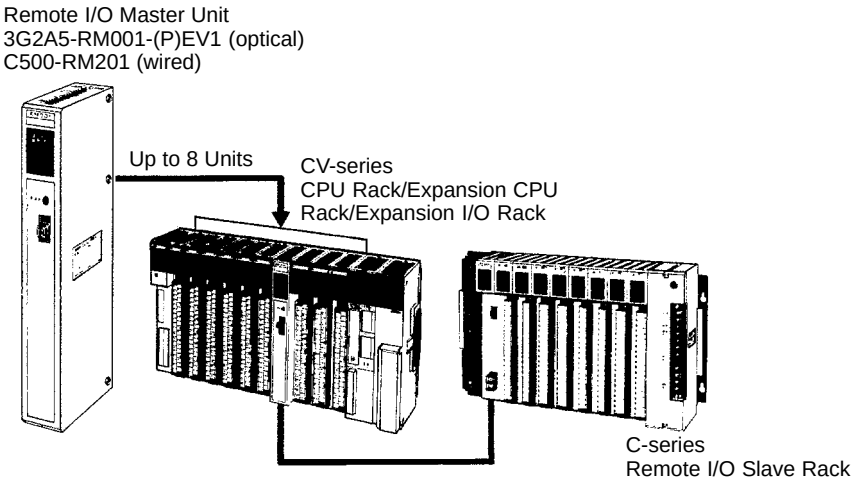
Remote I/O Master Units can be mounted on any slot of the CPU Rack, Expansion CPU Rack, or Expansion I/O Rack. Up to four Masters can be mounted for the CV500 or CVM1-CPU01-EV2, up to eight Masters for the CV1000, CV2000, CVM1-CPU11-EV2, or CVM1-CPU21-EV2.

For each Master, up to two Slave Racks can be connected for the CV500 or CVM1-CPU01-EV2; up to eight Slave Racks for the CV1000, CV2000, CVM1-CPU11-EV2, or CVM1-CPU21-EV2. No more than 16 Slave Racks can be connected per PC.

Slaves can be used to provide up to 512 remote I/O points for the CV500 or CVM1-CPU01-EV2; up to 1,024 remote I/O points for the CV1000, CV2000, or CVM1-CPU11-EV2, and up to 2,048 remote I/O points for the CVM1-CPU21-EV2.

Programming Devices cannot be connected to SYSMAC BUS Slave Racks.

When a C200H 10-slot Backplane is used is used as a SYSMAC BUS Slave Rack, only the eight leftmost slots can be used.

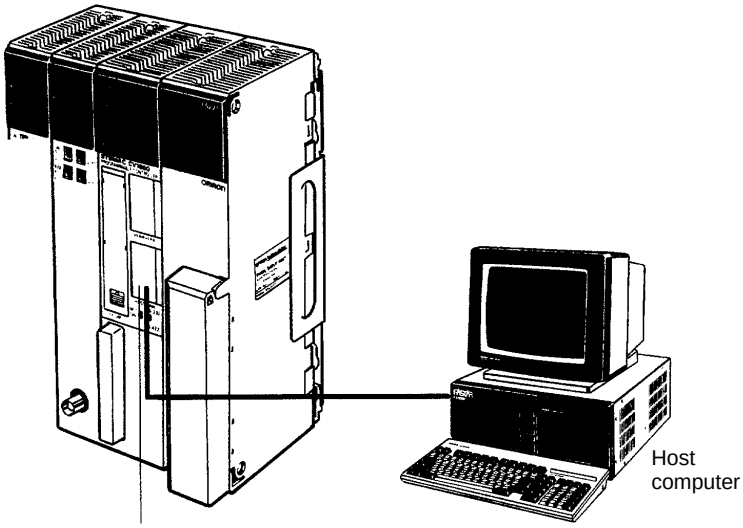


Host Link System (SYSMAC WAY)

The CV-series PCs can be connected to a host computer with the host link connector via the CPU or a CV500-LK201 Host Link Unit mounted to a Rack.

RS-232C or RS-422 communications can be used depending on the switch setting. When RS-422 is selected, up to 32 PCs can be connected to a single host.

Data is transmitted and received by commands and responses.

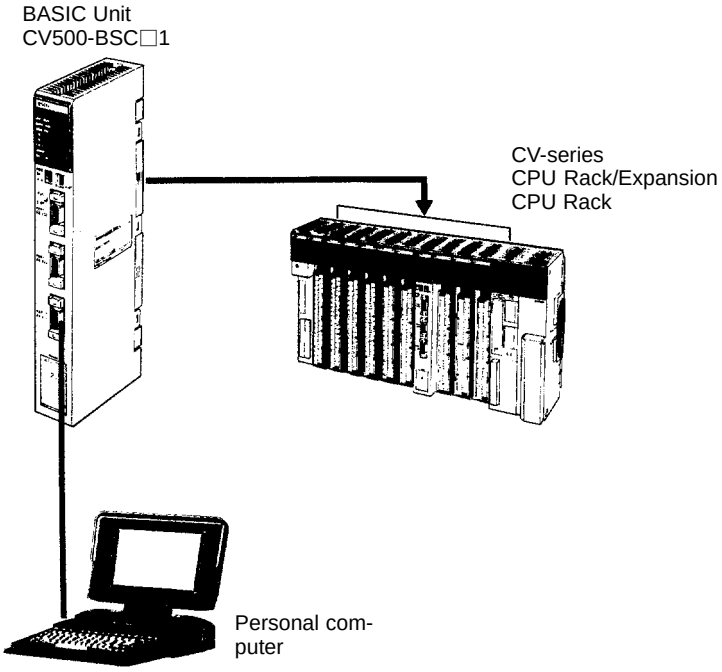


BASIC Unit

The BASIC Unit can be connected to a personal computer to enable communications with the PC using the BASIC programming language. Up to 512 bytes (256 words) of data can be transferred between the BASIC Unit and the CPU by the PC READ/WRITE command without using the PC program.

Up to 256 words of data can also be transferred between the BASIC Unit and the PC's CPU by using the NETWORK SEND and NETWORK RECEIVE (SEND(192)/RECV(193)) instructions in the PC program.

Data can also be transferred to other BASIC Units mounted on the same PC, or to BASIC Units mounted to other PCs connected by networks formed using a SYSMAC NET Link or SYSMAC LINK System. RS-232C, RS-422, Centronics, and GPIB interfaces are available.



Personal Computer Unit

The Personal Computer Unit is a full-fledged IBM PC/AT compatible that can be used to run independent programming directly on a Rack to eliminate the need for separate installation space. It can run along or connected to any of the normal peripherals supported by IBM PC/AT compatibles (mice, keyboards, monitors, data storage devices, etc.), and as a CPU Bus Unit, the Personal Computer Unit interfaces directly to the PC's CPU though the CPU bus to eliminate the need for special interface hardware, protocols, or programming.

1-11 New CPUs and Related Units

The following new CV-series CPUs and related Units are included in this version of the manual for the first time. Refer to relevant sections of this manual or the *CV-series PC Operation Manual: Ladder Diagrams* for further details.

Unit	Model number	Main specifications
CPU	CVM1-CPU01-EV2	I/O capacity: 512 pts; Ladder diagrams only
	CVM1-CPU11-EV2	I/O capacity: 1,024 pts; Ladder diagrams only
	CVM1-CPU21-EV2	I/O capacity: 2,048 pts; Ladder diagrams only
	CV500-CPU01-EV1	I/O capacity: 512 pts; Ladder diagrams or SFC + ladder diagrams
	CV1000-CPU01-EV1	I/O capacity: 1,024 pts; Ladder diagrams or SFC + ladder diagrams
	CV2000-CPU01-EV1	I/O capacity: 2,048 pts; Ladder diagrams or SFC + ladder diagrams
Temperature Controller Data Link Unit	CV500-TDL21	Connects up to 64 temperature controllers via 2 ports.

1-12 CPU Comparison

The following table shows differences between the various CV-series CPUs.

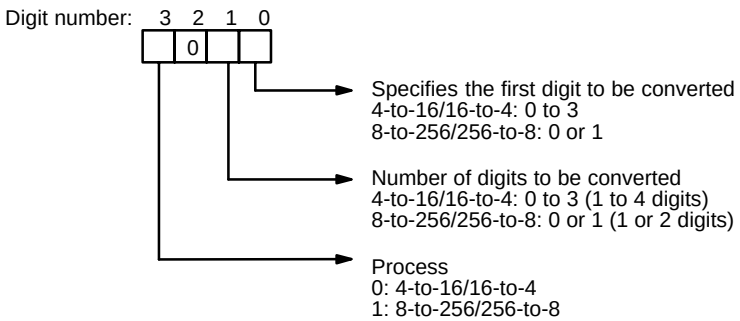
CPU		CVM1-CPU01-EV2	CVM1-CPU11-EV2	CVM1-CPU21-EV2	CV500-CPU01-EV1	CV1000-CPU01-EV1	CV2000-CPU01-EV1
Program-ming	Ladder diagrams	Supported	Supported	Supported	Supported	Supported	Supported
	SFC	Not supported	Not supported	Not supported	Supported	Supported	Supported
	Instructions	284	284	285	169	170	170
Speed	Basic instructions (ms)	0.15 to 0.45	0.125 to 0.375	0.125 to 0.375	0.15 to 0.45	0.125 to 0.375	0.125 to 0.375
	Other instructions (ms)	0.6 to 9.9	0.5 to 8.25	0.5 to 8.25	0.6 to 9.9	0.5 to 8.25	0.5 to 8.25
Program capacity		30K words	30K words	62K words	30K words	62K words	62K words
Local I/O capacity		512 pts	1,024 pts	2,048 pts	512 pts	1,024 pts	2,048 pts
Remote I/O capacity	SYSMAC BUS/2	1,024 pts	2,048 pts	2,048 pts	1,024 pts	2,048 pts	2,048 pts
	SYSMAC BUS	512 pts	1,024 pts	2,048 pts	512 pts	1,024 pts	1,024 pts
DM Area		8K words	24K words	24K words	8K words	24K words	24K words
Expansion DM Area		Not supported	Not supported	32K words each for 8 banks	Not supported	32K words each for 8 banks	32K words each for 8 banks
Timers		512	1,024	1,024	512	1,024	1,024
Counters		512	1,024	1,024	512	1,024	1,024
SFC steps		None	None	None	512	1,024	1,024
Step Flags		None	None	None	512	1,024	1,024
Transition Flags		None	None	None	512	1,024	1,024

1-13 Improved Specifications

1-13-1 Upgraded Specifications

The following improvements were made December 1992 and are applicable to all CV500-CPU01-E and CV1000-CPU01-E CPUs with lot numbers in which the rightmost digit is 3 (□□□3) or higher.

1, 2, 3... 1. The MLPX(110) (4-TO-16 DECODER) instruction has been improved to also function as a 8-to-256 decoder and the DMPX(111) (16-TO-4 ENCODER) instruction has been improved to also function as a 256-to-8 encoder. To enable this improvement, the digit designator (Di) has been changed as shown below. Refer to *5-17-8 DATA DECODER – MLPX(110)* and *5-17-9 DATA ENCODER – DMPX(111)* for details on these instructions.



2. The following operating parameter has been added to the PC Setup. Refer to *Section 7 PC Setup* for details on the PC Setup.

   JMP(004) 0000 Processing
   Y: Enable multiple usage (default)
   N: Disable multiple usage

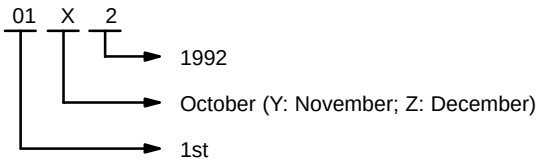
3. The operation of Completion Flags for timers has been changed so that the Completion Flag for a timer turns ON only when the timer instruction is executed with a PV of 0000 and not when the timer's PV is refreshed to a PV value of 0000, as was previously done.

   Only the timing of the activation of the Completion Flag has been changed, and the timer's PV is still refreshed at the same times (i.e., when the timer instruction is executed, at the end of user program execution, and every 80 ms if the cycle time exceeds 80 ms).

   Refer to *5-3 Data Areas, Definers, and Flags* for details on timer and counter instructions.

4. The READ(190) (I/O READ) and WRIT(191) (I/O WRITE) instructions have been improved so that they can be used for Special I/O Units on Slave Racks 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, CT021, CT041, ASC04, IDS01-V1, IDS02, IDS21, IDS22, or LDP01-V1. (The NC221-E, NC222, CP131, and FZ001 cannot be mounted to Slave Racks.)

   Refer to *5-35-1 I/O READ – READ(190)* and *5-35-3 I/O WRITE – WRIT(191)* for details on these instructions.

1-13-2 Version-1 CPUs

CV-series CPUs were changed to version 1 from December 1993. The new model numbers are as follows: CVM1-CPU01-EV1, CVM1-CPU11-EV1, CV500-CPU-EV1, CV1000-CPU-EV1, and CV2000-CPU-EV1. (Of these, all CVM1 CPUs were changed to version 2 from December 1994; refer to the next sections for details.)

The following additions and improvements were made to create the version-1 CPUs.

PT Link Function

The host link interface on the CPU can be used to connect directly to Programmable Terminals (PTs) to create high-speed data links. To use the PT links, turn ON pin 3 of the DIP switch on the CPU. Pin 3 must be turned OFF for host link connections.

EEPROM Writes

With the new CPUs, you can write to EEPROM Memory Cards mounted to the CPU by using the file write operation from a Peripheral Device. A Memory Card Writer is no longer required for this write operation. Writing is possible in PROGRAM mode only.

New Command

A new I/O REGISTER command (QQ) has been added so that words from different data areas can be read at the same time.

Faster Host Links

The communications response time for the built-in host link interface on the CPU has been improved by a factor of approximately 1.2.

Faster Searches

The search speed from Peripheral Devices for instructions and operands has been nearly doubled.

1-13-3 Version-2 CVM1 CPUs

CVM1 CPUs were changed to version 2 and a new CPU was added from December 1994. The new model numbers are as follows: CVM1-CPU01-EV2, CVM1-CPU11-EV2, and CVM1-CPU21-EV2.

The following additions and improvements were made to create the version-2 CPUs.

CMP/CMPL

New versions of the CMP(020) and CMPL(021) have been added that are not intermediate instructions. The new instructions are CMP(028) and CMPL(029) and are programs as right-hand (final) instructions. A total of 24 other new comparison instructions have also been added with symbol mnemonics (e.g., >, +, and <).

XFER(040)

This instruction has been upgraded so that source and destination areas can overlap.

DMPX(111)

This instruction has been upgraded so that either the MSB or the LSB can be specified for use as the end code. Previously only the the MSB could be used.

New Flags

Underflow and Overflow Flags have been added at A50009 and A50010, respectively. These flags can be turned ON or OFF when executing ADB, ADBL, SBB, and SBBL and can be saved or loaded using CCL and CCS.

New Instructions

A total of 125 new instructions have been added. These instructions are supported by version-2 CPUs only.

Faster Online Editing

The time that operation is stopped for online editing has been reduced and is no longer added to the cycle time. The following are just a couple of examples.

Edit	Time operation is stopped
Adding or deleting one instruction block at the beginning of a 62K-word program	Approx. 0.5 s
Deleting an instruction block containing JME from the beginning of a 62K-word program	Approx. 2.0 s

The above speed increase also applies to all V1 CPUs with lot numbers in which the rightmost digit is 5 (□□□5) or higher.

New Host Link Commands

New C-mode commands have been added and the functionality of existing commands has been improved as follows:

New Commands

- RL/WL: Read and write commands for the CIO Area.
- RH/WH: Read and write commands for the CIO Area.
- CR: Read command for the DM Area.
- R#/R$/R%: SV read commands.
- W#/W$/W%: SV change commands.
- *: Initialization command.

Improved Commands

- The Link Area (CIO 1000 to CIO 1063) and Holding Area (CIO 1200 to CIO 1299) can now be specified for the KS, KR, KC, and QQ commands.
- CVM1-CPU21-EV1 can now be read for the MM command.

The above new and improved commands can also be used with all V1 CPUs with lot numbers in which the rightmost digit is 5 (□□□5) or higher.

Note Only the following Programming Devices support version-2 CPUs: SSS (C500-ZL3AT-E) and the CVM1-PRS21-V1 Programming Console (CVM1-MP201-V1). Of these, the SSS does not support SFC and thus cannot be used for the CV500, CV1000, and CV2000. Use the CVSS for these PCs.

1-13-4 Upgraded Specifications

The following improvements were made December 1995 and are applicable to all CV500/CV1000/CV2000-CPU01-EV1 and CVM1-CPU01/CPU11/CPU21-EV2 CPUs with lot numbers in which the rightmost digit is 6 (□□□6) or higher.

Simplified Backup Function Added

Specifications have been changed so that the user program, Extended PC Setup, and IOM/DM data can be backed up from memory in the CPU Unit to a Memory Card without using a Programming Device, and so that the data backed up in the Memory Card can be transferred back to memory in the CPU Unit without using a Programming Device. (This method is provided as an easy way to backup and restore data. We still recommend that a Programming Device be used to confirm all essential backup and restore operations.)

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 DIP switch on the CPU 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 DIP switch on the CPU Unit.

Specifying Files

Pins 1 and 2 on the DIP switch are used to specify the files to be transferred. These pins are normally used to specify the baud rate for a Programming Device, so be sure to return them to their original settings when you finish backing up or restoring data. 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 about 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.

Note Approximately 17 s will be required to backup all data except the EM files for the CV1000 using a 1-Mbyte Memory Card. Approximately 2 s will be required to restore the same data to the CPU Unit's memory.

Application of Commercial 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.

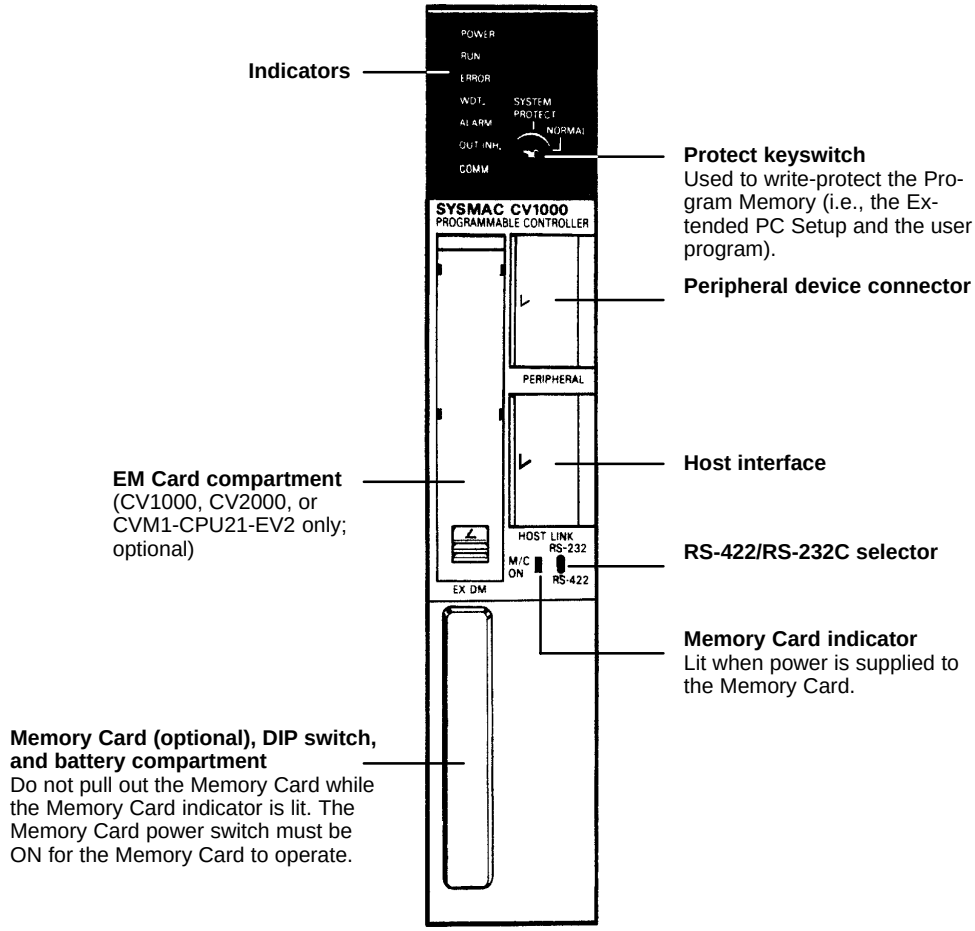
Note The 2-Mbyte Memory Cards cannot be used in the CV500-MCW01 Memory Card Writer.

SECTION 2
Hardware Considerations

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

2-1 CPU Components

The following diagram shows the basic components of the CPU that are used in general operation of the PC.



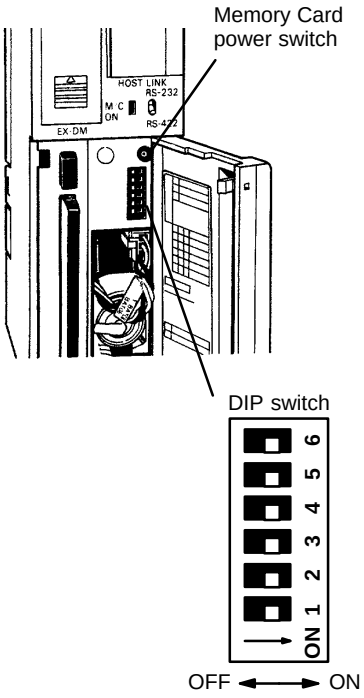
2-1-1 Indicators

CPU indicators provide visual information on the general operation of the PC. Although not substitutes for proper error programming using the flags and other error indicators provided in the data areas of memory, these indicators provide ready confirmation of proper operation. CPU indicators are shown below and are described in the following table. Indicators are the same for all CV-series PCs.

Indicator	Function
POWER (green)	Lights when power is supplied to the CPU.
RUN (green)	Lights when the CPU is operating normally.
ERROR (red)	Lights when an error is discovered in diagnostic operations. When this indicator lights, the RUN indicator will go off, CPU operation will be stopped, and all outputs from the PC will be turned OFF.
WDT (red)	Lights when a CPU error (watchdog timer error) has been detected. When this indicator lights, the RUN indicator will go off, CPU operation will be stopped, and all outputs from the PC will be turned OFF.
ALARM (red)	Lights when an error is discovered in diagnostic operations. PC operation will continue.
OUT INH (orange)	Lights when the Output OFF Bit, A00015, is turned ON. All PC outputs will be turned OFF.
COMM (orange)	Lights when the host link interface is transmitting or receiving data.
M/C ON (orange)	Lights when power is supplied to the Memory Card. Press the Memory Card power switch once to turn the power OFF or ON. Do not remove the Memory Card while the power is ON. It may flicker when the simplified backup function operates. Refer to *1-13-4 Upgraded Specifications* for details.

2-1-2 Switches

The DIP switch and memory card power switch are shown below and the setting of these and the other CPU switches are described in the following table. Switches are the same for all CV-series PCs.



Switch		Position	Function
Protect keyswitch		Vertical	Program Memory (i.e., the Extended PC Setup and the user program) is write-protected. (See note 1)
		Horizontal	Program Memory is not write-protected.
RS-422/RS-232C selector		Up	Host link communications set for RS-232. (Also see CPU DIP switch pins 4 and 6 below.)
		Down	Host link communications set for RS-422. (Also see CPU DIP switch pins 4 and 6 below.)
Memory Card power switch (See note 4.)		Not applicable	Press and release to turn the power on or off. (The M/C ON indicator lights when power is on.)
CPU DIP	Pins 1, 2 (See notes 3 and 4.)	OFF, OFF	Peripheral device communications: 50,000 bps
		ON, OFF	Peripheral device communications: 19,200 bps
		OFF, ON	Peripheral device communications: 9,600 bps
		ON, ON	Peripheral device communications: 4,800 bps
	Pin 3	OFF	Communicate via Host Link communications
		ON	Communicate with PT via NT Link communications.
	Pin 4	OFF	Host link communications governed by PC Setup. (See note 2)
		ON	Following settings used for host link communications, regardless of PC Setup: 9,600 bps, unit number 00, even parity, 7-bit data, 2 stop bits. **Note:** The above settings apply to CPUs manufactured from July 1995 (lot number **75 for July 1995). For CPUs manufactured before July 1995 (lot number **65 for June 1995), only 1 stop bit will be set and the baud rate will be 2,400 bps.
	Pin 5 (See note 4.)	OFF	Files are not transferred from the Memory Card automatically at start-up.
		ON	The program file (AUTOEXEC.OBJ) and PC Setup file (AUTOEXEC.STD) will be transferred from the Memory Card to the CPU automatically at start-up.
	Pin 6	OFF	The termination resistance is off.
		ON	The termination resistance is on. (This setting is used for the last Unit in a RS-422 Host Link System only; intermediate Units must be set to OFF.)

Note

1. The user program can also be protected from a Peripheral Device.
2. Factory settings are 9,600 bps, 7-bit data, even parity, and 2 stop bits.
3. The baud rate must be set to 50,000 bps when the Graphic Programming Console or Programming Console is connected to the PC, and to 9,600 bps when a computer running the CV Support Software is connected.
4. The following switches and pins are also used for the simplified backup function. Pins 1 and 2 are used to specify files, pin 5 is used to specify the direction of the transfer, and the Memory Card power switch is used to start data transfers. Refer to *1-13-4 Upgraded Specifications* for details.

2-2 Program Memory

Program Memory is contained in the CPU and is divided into two areas, the PC Setup and the Program Area. There are 32K words of Program Memory available in the CV500, CVM1-CPU01-EV2, or CVM1-CPU11-EV2, and 64K words available in the CV1000, CV2000, or CVM1-CPU21-EV2. The first 2K words in both groups of PCs is taken up by the PC Setup, leaving 30K words in the CV500, CVM1-CPU01-EV2, or CVM1-CPU11-EV2 Program Area, and 62K words in the CV1000, CV2000, or CVM1-CPU21-EV2 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 CV-series 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 SFC and/or ladder program.

The following table shows the maximum program size (combined total of the SFC and ladder programs) when SFC programming is used and the maximum number of steps, transitions, and actions in the SFC program.

PC	Program capacity	SFC steps	SFC transitions	SFC actions
CV500	30K words	512	512	1,024
CV1000	62K words	1,024	1,024	2,048
CV2000	62K words	1,024	1,024	2,048
CVM1-CPU01/11-EV2	30K words	None (SFC programming is not supported.)		
CVM1-CPU21-EV2	62K words			

Note When ladder programming is used, the program capacity includes 1.85K words reserved for system use.

2-3 Memory Cards

File memory (used to store programs and other data) is attached to the CPU in the form of Memory Cards. The portable, high-capacity Cards allow large quantities of data to be handled by simply switching Memory Cards. Because Memory Cards are not provided with the PC, they must be selected and installed in the CPU. Three types of Memory Card are available: RAM, EPROM, and EEPROM. Each of these comes in various capacities. Some of the memory is used for file management and directories.

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	
	1 Mbyte	1016K bytes	HMC-EP161	

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

The following commercially available memory cards can be used for all CV500/CV1000/CV2000-CPU01-EV1 and CVM1-CPU01/CPU11/CPU21-EV2 CPUs with lot numbers in which the rightmost digit is 6 (□□□6) or higher. 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 CVSS/SSS or the CV500-MCW□□ Memory Card Writer; EPROM Memory Cards can be formatted with the CV500-MCW□□ Memory Card Writer only.

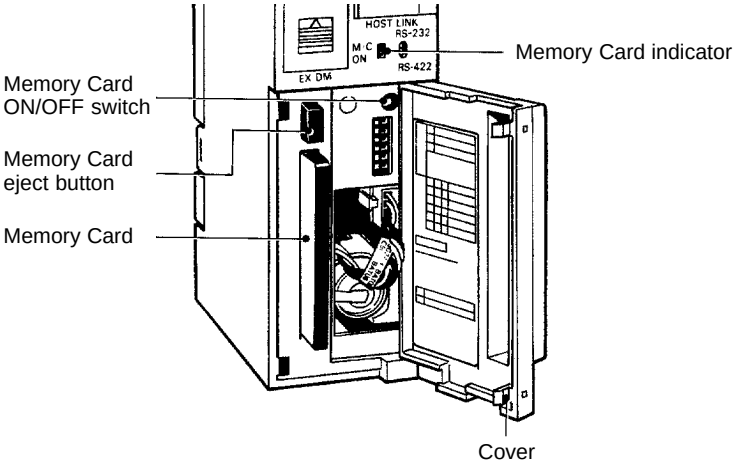
⚠ Caution Memory Cards can be damaged by twisting, shock, or exposure to high temperature, humidity, or direct sunlight. Handle them with care.

2-3-1 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 ON/OFF 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 ON/OFF 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 Memory Card indicator is lit; doing so may result in data errors in the memory.

2-3-2 File Transfer between the CPU and Memory Card

Data files can be transferred between the Memory Card and PC data areas with the FILR(180) and FILW(181) instructions. A program file can be transferred from the Memory Card with FILP(182) or FLSP(183) to change the program during operation. Refer to details on these instructions later in the manual.

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[1]	*filename*.STD
Data files	*filename*.IOM
Ladder program files (files saved with the partial save operation)	*filename*.LDP
SFC program files (one step)	*filename*.SFC
Program file (complete program)	*filename*.OBJ
Extended PC Setup[1] (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 and SYSMAC NET 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 *1-13-4 Upgraded Specifications* for details.

File Transfer at Start-up

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

1, 2, 3...

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 CVSS/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 CVSS/SSS. The Extended PC Setup file (AUTOEXEC.STD) can be transferred directly from the PC to the Memory Card in online operations from the CVSS/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 CVSS/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:

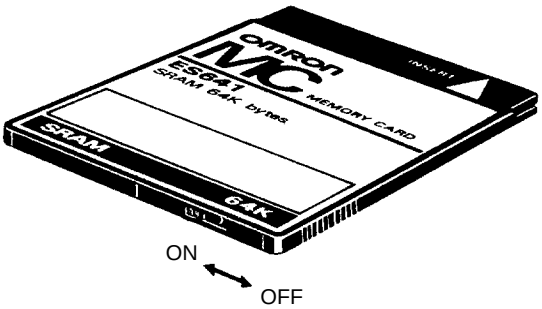
- The write-protect switch is ON.
- One or both AUTOEXEC files are missing.
- The Memory Card power is OFF. (If the M/C ON indicator is not lit when the PC power is ON, press the Memory Card power switch.)
- The Memory Card is not installed.

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 Peripheral Device, e.g., the CVSS/SSS.
2. Reading and writing can be performed by a command from a host computer.
3. Reading and writing can be performed by instructions in the ladder diagram program. The four instructions are described in the following table. Refer to *Section 5 Instruction Set* for details.

Instruction	Function	Filename
FILR(180) (READ DATA FILE)	Reads the specified data file from the Memory Card and writes it to a specified data area.	*filename*.IOM
FILW(181) (WRITE DATA FILE)	Reads a specified amount of data file from a specified data area and writes it to (or creates) the specified data file in the Memory Card.	*filename*.IOM
FILP(182) (READ PROGRAM FILE)	Reads the specified ladder program file (either one action program or one transition program if SFC programming is being used) from the Memory Card and writes it in Program Memory.	*filename*.LDP
FLSP(183) (CHANGE STEP PROGRAM)	Reads the specified SFC program file (one step) from the Memory Card and writes it in Program Memory.	*filename*.SFC

4. Reading and writing can be performed by using the simplified backup function. Refer to *1-13-4 Upgraded Specifications* for details.

2-4 Data Memory and Expansion Data Memory Unit

The size of the Data Memory Area for the CV-series PCs is shown in the following table.

PC	DM Area capacity	Addresses
CV500 or CVM1-CPU01-EV2	8K words	D00000 to D08191
CV1000, CV2000, CVM1-CPU11-EV2 or CVM1-CPU21-EV2	24K words	D00000 to D24575

If the above capacities are insufficient, an Expansion Data Memory Unit can be added to create an EM (Expansion Data Memory) Area with the CV1000, CV2000, or CVM1-CPU21-EV2. This Unit must be purchased separately as an option and is not available for other PCs. 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.

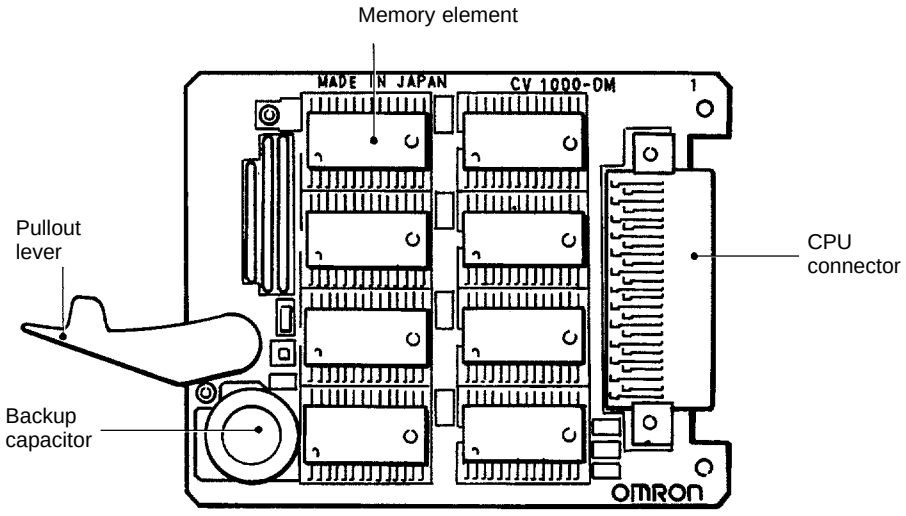
EM Area memory is divided into banks of 32K words each. Words E00000 to E32765 of the current bank can be accessed. The current bank number is contained in the least significant digit of A511. A511 is in a read-only area, but the

current bank number can be changed with the EMBC(171) instruction. Refer to *Section 5 Instruction Set* for details.

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)

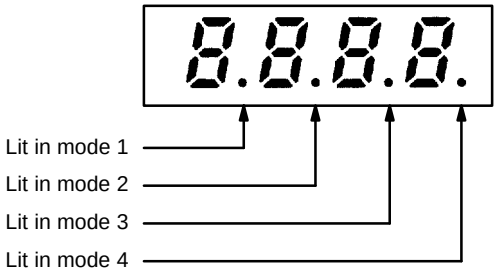
The following diagram shows the structure of the EM Unit and identifies its main components.



Expansion Data Memory Unit

2-5 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.

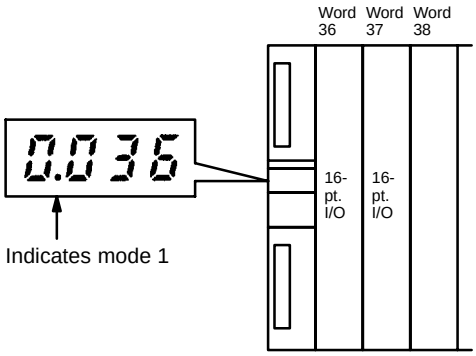


Pressing the mode selector switch changes the display to the next mode. The Unit will automatically enter the mode specified in the PC Setup (default setting: mode 1). Refer to *Section 7 PC Setup* for details.

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.



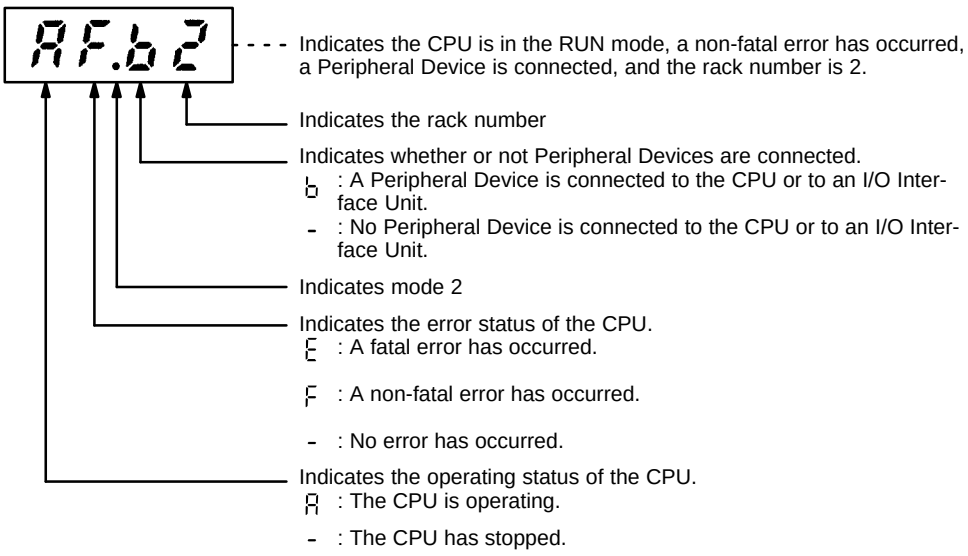
Display Mode 2

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...

1. The leftmost digit indicates whether or not the CPU is operating.
   "A" indicates it is operating.
   "-" indicates it is stopped.
2. 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.
3. The third digit from the left indicates whether or not a peripheral device is connected to the CPU or Expansion CPU Rack. If a peripheral device is already connected, another cannot be connected.
   "b" indicates that a peripheral device is connected.
   "-" indicates that a peripheral device is not connected.

   Note Only one Peripheral Device can be connected to the CPU and I/O Interface Units for each PC, but three additional Peripheral Devices can be connected to the SYSMAC BUS/2 Slave Racks.
4. 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 des-

tination 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

Mode 4 is not being used currently. In mode 4, the display will show only the decimal point indicating it is in mode 4.

2-6 Peripheral Devices

A total of four Peripheral Devices can be connected to a CV-series PC, as shown in the following table. Only one Peripheral Device can be connected to the CPU or an I/O Interface Unit.

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

Connecting Unit	Max. connection combinations		
CPU	1	0	0
I/O Interface Unit	0	1	0
SYSMAC BUS/2 Remote I/O Slave Units	3	3	4

Connecting Peripheral Devices

Peripheral Devices can be connected even when the PC is ON. Insert the cable connector until it locks. Using pins 1 and 2 on the CPU DIP switch, set the baud rate to 50,000 bps for the Graphic Programming Console or Programming Console or to 9,600 bps for a computer running the CV Support Software.

If the ERROR indicator lights when the PC is turned ON, find the source of the error by displaying error messages at the terminal. For a memory error, perform the memory clear or program transfer operation online from the CVSS/SSS and then clear the error. If a memory error cannot be cleared, there might be a hardware problem in the CPU.

Note

1. I/O tables cannot be created or edited and broadcast testing is not possible for SYSMAC LINK Systems if the Peripheral Device is connected to a Slave in a SYSMAC BUS/2 Remote I/O System.
2. Refer to *Appendix A Standard Models* in the *CV-series PC Installation Guide* for a list of available Peripheral Devices.

2-7 PC Configuration

The following is an overview of the PC configuration. Refer to the *CV-series PC Installation Guide* for details.

The basic PC configuration consists of three types of Rack: a CPU Rack, an Expansion CPU Rack, and one or more Expansion I/O Racks. The Expansion CPU Rack and Expansion I/O Racks are not a required part of the basic system.

An Expansion CPU Rack is used when the CPU Rack cannot accommodate the required number of CPU Bus Units (SYSMAC BUS/2 Remote I/O Master Units, BASIC Units, SYSMAC NET Link Units, and SYSMAC LINK Units). Expansion I/O Racks are used to increase the number of I/O points, but do not support CPU Bus Units. An illustration of these Racks is provided in *3-3-1 I/O Area*.

An Expansion CPU Rack cannot be connected to a CVM1-BC103/053 Backplane.

A fourth type of Rack, called a Slave Rack, can be used when the PC is provided with a SYSMAC BUS or SYSMAC BUS/2 Remote I/O System.

CPU Racks

A CPU Rack consists of four components: (1) The CPU Backplane, to which the CPU, the Power Supply, and other Units are mounted. (2) The CPU, which

executes the program and controls the PC. (3) Other Units, such as I/O Units, Special I/O Units, and Link Units, which provide the physical I/O terminals corresponding to I/O points. (4) The I/O Control Unit which provides connections to an Expansion CPU Rack and Expansion I/O Racks. The I/O Control Unit is not required if an Expansion CPU Rack and Expansion I/O Racks are not connected and connect be mounted to CVM1-BC103/053 Backplanes. (5) The Power Supply, which provides power to the CPU Rack.

A CPU Rack can be used alone or it can be connected to other Racks to provide additional I/O points. The CPU Backplane provides slots to which other Units can be mounted. Depending on the model of Backplane used, either three, five, or ten slots are available for other Units.

Expansion CPU Racks

An Expansion CPU Rack Consists of an Expansion CPU Backplane, a Power Supply, and an I/O Interface Unit to connect to the CPU Rack. Eleven slots are available for other Units. Up to 16 CPU Bus Units can be connected to the CPU Rack and Expansion CPU Rack. Expansion I/O Racks can be connected to the Expansion CPU Rack.

An Expansion CPU Rack cannot be connected to a CVM1-BC103/053 Backplane.

Expansion I/O Racks

An Expansion I/O Rack can be thought of as an extension of the PC because it provides additional slots to which other Units (except CPU Bus Units) can be mounted. It is built onto an Expansion I/O Backplane to which a Power Supply and other Units are mounted. Depending on the model of Backplane used, either four, six, or 11 slots are available for other Units.

An I/O Interface Unit is also mounted to any Expansion I/O Rack to interface the Rack to the CPU or Expansion CPU Rack. Also, an I/O Control Unit must be mounted to any CPU Rack to which more than one Expansion I/O Rack is mounted. If only one Expansion I/O Rack and no Expansion CPU Rack is connected, the I/O Interface and I/O Control Units are not required and the Expansion I/O Rack can be connected directly to the CPU Rack.

An Expansion I/O Rack is always connected to the CPU via the connectors on the Backplanes, allowing communication between the two Racks. With C-series Expansion I/O Racks, up to seven Expansion I/O Racks can be connected in series to the CPU Rack. With CV-series Expansion I/O Racks, up to seven Expansion I/O Racks can be connected to the CPU Rack in two series. If an Expansion CPU Rack is used, only six Expansion I/O Racks can be connected.

Only one Expansion I/O Rack cannot be connected to a CVM1-BC103/053 Backplane.

Setting Rack Numbers

I/O words are allocated to Units mounted on the CPU, Expansion CPU, and 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. Never set the rack number of an Expansion CPU or Expansion I/O Rack to 0.

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.
3. If a single Expansion I/O Rack is connected to a CPU Rack, an I/O Interface Unit is not required and the rack number of Expansion I/O Rack is fixed at 1.
4. When mounting an Interrupt Input Unit to an Expansion CPU Rack, always set the rack number of the Expansion CPU Rack to 1.

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

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 PC memory addresses are shown in parentheses. These memory address are used for indirect addressing. Refer to *3-11 DM and EM Areas* and to *5-3 Data Areas, Definers, and Flags* for details on indirect addressing.

Area	PC	Range	Function
CIO Area (Core I/O)	All	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 *3-3 CIO (Core I/O) Area* for details.
Temporary Relay Area	All	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	All	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	All	Words: A000 to A511 Bits: A00000 to A51115 ($0B00 to $0CFF)	Contains flags and bits with special functions.
Transition Area	CV500	TN0000 to TN0511 ($0D00 to $0D1F)	Transition Flags for the transitions in the SFC program.
	CV1000/CV2000	TN0000 to TN1023 ($0D00 to $0D3F)	
Step Area	CV500	ST0000 to ST0511 ($0E00 to $0E1F)	Step Flags for steps in the SFC program. A step is active when its flag is ON.
	CV1000/CV2000	ST0000 to ST1023 ($0E00 to $0E3F)	
Timer Area	CV500/ CVM1-CPU01-EV2	T0000 to T0511 (Completion Flags: $0F00 to $0F1F Present Values: $1000 to $11FF)	Used to define timers (normal, high-speed, and totalizing) and to access Completion Flags, PV, and SV.
	CV1000/CV2000/ CVM1-CPU11-EV2 CVM1-CPU21-EV2	T0000 to T1023 (Completion Flags: $0F00 to $0F3F Present Values: $1000 to $13FF)	
Counter Area	CV500/ CVM1-CPU01-EV2	C0000 to C0511 (Completion Flags: $0F80 to $0F9F Present Values: $1800 to $19FF)	Used to define counters (normal, reversible, and transition) and to access Completion Flags, PV, and SV.
	CV1000/CV2000/ CVM1-CPU11-EV2 CVM1-CPU21-EV2	C0000 to C1023 (Completion Flags: $0F80 to $0FBF Present Values: $1800 to $1BFF)	
DM Area	CV500/ CVM1-CPU01-EV2	D00000 to D08191 ($2000 to $3FFF)	Used for internal data storage and manipulation.
	CV1000/CV2000/ CVM1-CPU11-EV2 CVM1-CPU21-EV2	D00000 to D24575 ($2000 to $7FFF)	
EM Area	CV1000/CV2000 CVM1-CPU21-EV2	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	All	IR0 to IR2	Used for indirect addressing.
Data registers	All	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 possible also to access any memory location through its hexadecimal PC memory address with indirect addressing. Refer to *3-11 DM and EM Areas*, and *3-12 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.

Bit number	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CIO word 0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CIO word 0001	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 *3-9 Timer Area* and *3-10 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 PC 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 *3-11 DM and EM Areas* and *3-12 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.

Digit number	3				2				1				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

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

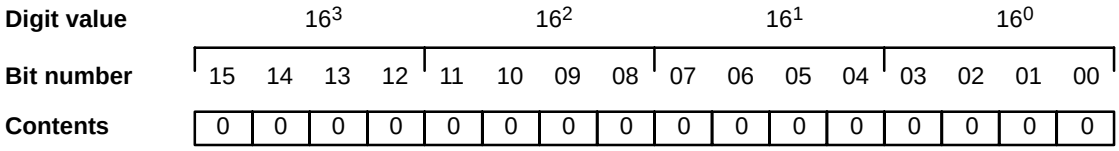
Decimal points are used in timers only. The least significant digit represents tenths of a second. All arithmetic instructions operate on integers only. When inputting data for use by Special I/O Units or other special applications, be sure to check on the type of data required for the application.

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.

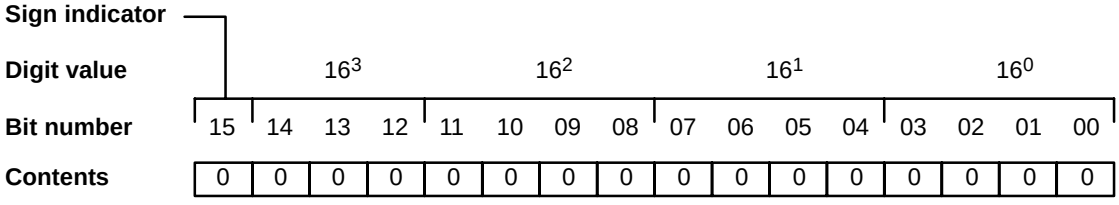
Unsigned binary

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).



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).



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. PC memory addresses are in parentheses.

Area	PC	Range	Function
I/O Area	CV500 CVM1-CPU01-EV2	Words: CIO 0000 to CIO 0031 Bits: CIO 000000 to CIO 003115 ($0000 to $001F)	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 CVSS/SSS with an I; output bits, with a Q.
	CV1000 CVM1-CPU11-EV2	Words: CIO 0000 to CIO 0063 Bits: CIO 000000 to CIO 006315 ($0000 to $003F)	
	CV2000 CVM1-CPU21-EV2	Words: CIO 0000 to CIO 0127 Bits: CIO 000000 to CIO 012715 ($0000 to $007F)	
Work Area	CV500 CVM1-CPU01-EV2	Words: CIO 0032 to CIO 0199 Bits: CIO 003200 to CIO 019915 ($0020 to $00C7)	These bits are used in the program to manipulate or to temporarily store data.
	CV1000 CVM1-CPU11-EV2	Words: CIO 0064 to CIO 0199 Bits: CIO 006400 to CIO 019915 ($0040 to $00C7)	
	CV2000 CVM1-CPU21-EV2	Words: CIO 0128 to CIO 0199 Bits: CIO 012800 to CIO 019915 ($0080 to $00C7)	
SYSMAC BUS/2 Area	CV500/ CVM1-CPU01-EV2	Words: CIO 0200 to CIO 0599 Bits: CIO 020000 to CIO 059915 ($00C8 to $0257)	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.
	CV1000/CV2000/ CVM1-CPU11-EV2	Words: CIO 0200 to CIO 0999 Bits: CIO 020000 to CIO 099915 ($00C8 to $03E7)	
Link Area	All	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 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	All	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	All	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	All	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.

Area	PC	Range	Function
Work Areas	All	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	CV500 CVM1-CPU01-EV2	Words: CIO 2300 to CIO 2427 Bits: CIO 230000 to CIO 242715 ($08FC to $097B)	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.
	CV1000/CV2000 CVM1-CPU11-EV2 CVM1-CPU21-EV2	Words: CIO 2300 to CIO 2555 Bits: CIO 230000 to CIO 255515 ($08FC to $09FB)	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. The number of I/O words varies between the PCs as shown in the following table.

PC	I/O words	I/O bits
CV500/ CVM1-CPU01-EV2	CIO 0000 to CIO 0031	CIO 000000 to CIO 003115
CV1000/ CVM1-CPU11-EV2	CIO 0000 to CIO 0063	CIO 000000 to CIO 006315
CV2000 CVM1-CPU21-EV2	CIO 0000 to CIO 0127	CIO 000000 to 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., 512 bits for the CV500 or CVM1-CPU01-EV2; 1,024 for the CV1000 or CVM1-CPU11-EV2; and 2,048 for the CV2000 or CVM1-CPU21-EV2. 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 Peripheral 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 CVSS/SSS (CV Support Software/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 *CVSS/SSS Operation Manuals*.

The first word allocated to each Rack can be set with the CVSS/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 CVSS/SSS to avoid overlapping allocations.

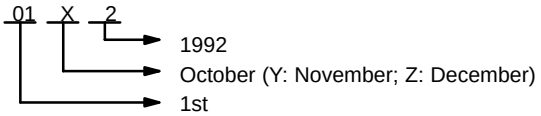
There are no specific words associated with any particular slot because different Units can require a different number of words. Rather, each Unit is assigned the next word(s) following the word(s) assigned to the previous Unit. If there are any empty slots, no words will be assigned to those slots. 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	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

Unit	Words required
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)
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 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.



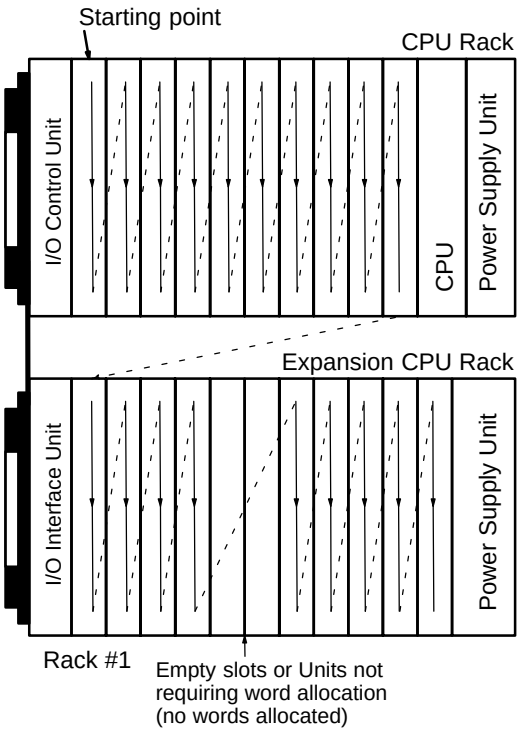
   b) The DIP switch on the Remote I/O Slave Unit must be set to “54MH.”
   c) The Special I/O Unit must be one of the following: AD101, CT012, CT041, ASC04, IDS01-V1, IDS02, IDS21, IDS22, LDP01-V1, or NC222.
7. Refer to the *CV-series PC Installation Guide* or to the operation manuals for individual Units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

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.

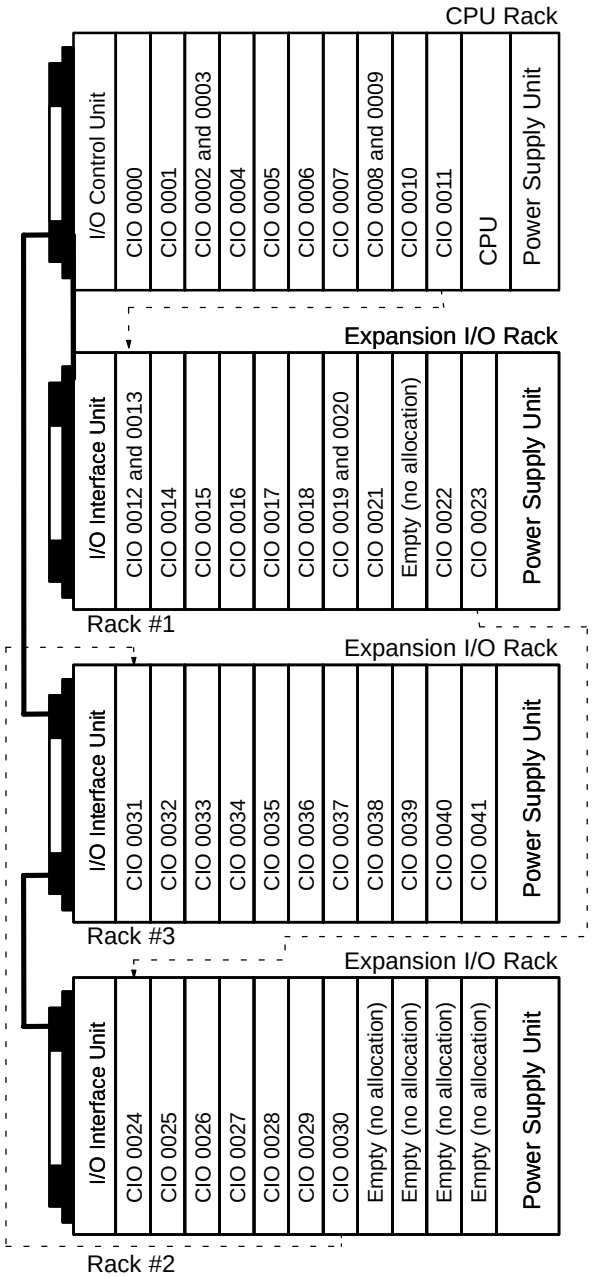
With I/O Units, bits within a word are assigned to terminals starting at the top of the I/O Unit with bit 00 and going sequentially to the bottom. If the first Unit

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.

Basic I/O Allocation



I/O Word Allocation Example



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 gener-

ating 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 *CVSS/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 two 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.

PC	Work words	Work bits
CV500 CVM1-CPU01-EV2	CIO 0032 to CIO 0199	CIO 003200 to CIO 019915
CV1000 CVM1-CPU11-EV2	CIO 0064 to CIO 0199	CIO 006400 to CIO 019915
CV2000 CVM1-CPU21-EV2	CIO 0128 to CIO 0199	CIO 012800 to CIO 019915
All	CIO 1964 to CIO 1999 CIO 2064 to CIO 2299	CIO 196400 to CIO 199915 CIO 206400 to CIO 229915

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 to the CV1000, CV2000, CVM1-CPU11-EV2, or CVM1-CPU21-EV2 (RM/2 #0 to RM/2 #3), and up to two Masters can be connected to the CV500 or CVM1-CPU01-EV2 (RM/2 #0 and RM/2 #1). 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) for the CV1000, CV2000, CVM1-CPU11-EV2, or CVM1-CPU21-EV2, and 1,024 (64 words) for the CV500 or CVM1-CPU01-EV2.

SYSMAC BUS/2 Area address allocation can be customized with the PC Setup using the CVSS/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 CVSS/SSS displays. Refer to the *CVSS/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 or a SYSMAC NET 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 unless specific link words are designated. Allocations can be designated from the CVSS/SSS. Refer to the *CVSS/SSS Operation Manuals*, and the *SYSMAC LINK System Manual*, or *SYSMAC NET Link System 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 CVSS/SSS. If the Holding Area is increased, it will overlap other areas. An “H” will appear before Holding Area bit addresses on the CVSS/SSS screen. Refer to the *CVSS/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 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, 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 their unit number settings as shown in the following tables.

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

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

An additional1600 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.

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 CV1000, CV2000, CVM1-CPU11-EV2, or CVM1-CPU21-EV2, and up to 4 Masters can be connected to the CV500 or CVM1-CPU01-EV2. The total number of I/O points in the SYSMAC BUS System must not exceed 2,048 (128 words) for the CVM1-CPU21-EV2, 1024 (64 words) for the CV1000, CV2000, or CVM1-CPU11-EV2, and 512 (32 words) for the CV500 or CVM1-CPU01-EV2.

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 CVSS/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
CIO words	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 CVSS/SSS displays. Refer to the *CVSS/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 CVSS/SSS enters TR bits automatically, although the TR bits are not shown on the CVSS/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 occured. (PC operation continues.)
07	ON when a fatal error has occured. (PC stops.)
08 to 10	Not used.
11	UM Protect Bit. Prevents both reading out and writing to Program Memory when turned ON. Set with the CVSS/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 CVSS/SSS. Refer to the *CVSS/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 CVSS/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
Words	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
Words	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 CVSS/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 10	Not used.
	11	Restart Continuation Bit
	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.
A005	00 to 07	SYSMAC BUS Error Check Bits
	08 to 15	Not used.

Word(s)	Bit(s)	Function
A006	00 to 15	Not used.
A007	00 to 15	Momentary Power Interruption Time (BCD)
A008	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	00 to 15	Startup Time (BCD)
A012 to A013	00 to 15	Power Interruption Time (BCD)
A014	00 to 15	Number of Power Interruptions (BCD)
A015	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	Peripheral Service Disable Bit
	05	I/O 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.
A20 to A299	00 to 15	Not used.
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.
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 to 07	Not used.
	08 to 11	Connected Device Code 2: GPC 3: Programming Console
	12 to 14	Not used.
	15	Peripheral Connected Flag
A307	00 to 07	Peripheral Connected Flags for RT #0 to RT #7 of RM/2 #0

Word(s)	Bit(s)	Function
	08 to 15	Peripheral Connected Flags for RT #0 to RT #7 of RM/2 #1
A308	00 to 07	Peripheral Connected Flags for RT #0 to RT #7 of RM/2 #2
	08 to 15	Peripheral Connected Flags for RT #0 to RT #7 of RM/2 #3
A309	00 to 15	Peripheral 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	00 to 02	Memory Card Type
	03 to 06	Not used.
	07	Memory Card Format Error Flag
	08	Memory Card Transfer Error Flag
	09	Memory Card Write Error Flag
	10	Memory Card Read Error Flag
	11	FIle Missing Flag
	12	Memory Card Write Flag
	13	Memory Card Instruction Flag
	14	Accessing Memory Card Flag
	15	Memory Card Protected Flag
A344 to 345	00 to 15	Not used.
A346	00 to 15	Number of Words Remaining to transfer to memory card for a file read/write instruction (BCD)
A347 to A399	00 to 15	Not used.
A400	00 to 15	Error Code
A401	00 to 04	Not used.
	06	FALS Error Flag
	07	SFC Fatal Error Flag
	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 to 01	Not used.
	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	Not used.
	11	SFC Non-fatal Error Flag
	12	Indirect DM Error Flag
	13	Jump Error Flag
	14	Not used.
	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 A413	00 to 15	Not used.
A414	00 to 15	SFC Fatal Error Code
A415 to A417	00 to 15	Not used.
A418	00 to 15	SFC Non-fatal Error Code
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 13	Not used
	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	0.1-s Clock Pulse
	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 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 Restart Continuation Bit

Bit A00011 can be turned ON to make the PC automatically resume operation from the point that operation stopped due to a power interruption. If bit A00011 is OFF, the PC will enter the start-up mode set in the PC Setup and will begin operation from the first step if the start-up mode is RUN or MONITOR mode.

When the Restart Continuation Bit is turned ON, several parameters in the PC Setup must also be made for the PC to restart properly. Refer to *Section 7 PC Setup* for more details.

3-6-2 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.

3-6-3 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.

3-6-4 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.

3-6-5 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. The Output OFF Bit is turned ON automatically when Restart Continuation (bit A00011) has taken place. It is therefore necessary to include a step in the program to turn this bit OFF to continue operation after a power interruption. Refer to *6-1 PC Operation* for details.

3-6-6 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 CVSS/SSS.

3-6-7 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 *3-6-35 SYSMAC BUS Error Flag* for more details.

3-6-8 Momentary Power Interruption Time

Word A007 contains the duration of the most recent power interruption. The time is recorded in 4-digit BCD in milliseconds (0000 ms to 9999 ms), as shown in the following table.

Bits			
15 to 12	**11 to 08**	**07 to 04**	**03 to 00**
10^3	10^2	10^1	10^0

The power interruption time is output to words A012 and A013, and the number of power interruptions is output to word A014.

3-6-9 CVSS/SSS Flags

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

Stop Monitor Flag (A00807) Bit A00807 is turned ON when the Stop Monitor is used from the CVSS/SSS, and is turned OFF when it is completed.

Execution Time Measured Flag (A00808) Bit A00808 is turned ON when the execution time has been measured with MARK(174) instructions with the CVSS/SSS.

Differentiate Monitor Completed Flag (A00809) Bit A00809 is turned ON when the differentiate monitor condition has been established with the CVSS/SSS.

Stop Monitor Completed Flag (A00810) Bit A00810 is turned ON when the Stop Monitor operation has been completed with the CVSS/SSS.

Trace Trigger Monitor Flag (A00811) Bit A00811 is turned ON when one of the trigger conditions has been established during execution of a Data or Program Trace with the CVSS/SSS.

Trace Completed Flag (A00812) 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 CVSS/SSS.

Trace Busy Flag (A00813) Bit A00813 is turned ON when a Data or Program Trace is executed with the CVSS/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 CVSS/SSS.

Sampling Start Bit (A00815) Bit A00815 is turned ON to start a Data Trace.

3-6-10 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 99
	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)

3-6-11 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 99
	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)

3-6-12 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 BCD, and can be reset by writing #0000 to word A014.

3-6-13 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.

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.

Peripheral Service Disable Bit

Bit A01704 can be turned ON to stop service to Peripheral Devices. Turn OFF again to resume service to Peripheral Devices.

I/O Refresh Disable Bit

Bit A01705 can be turned ON to stop periodic and SYSMAC BUS refreshing. Turn OFF again to resume periodic and SYSMAC BUS refreshing.

3-6-14 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.

3-6-15 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 8 Error Processing* 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

Record	Addresses	Pointer value*
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 contents, 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.

3-6-16 CPU Bus Unit Initializing Flags

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

3-6-17 Wait Flags

Start-up Wait Flag (A30600)

Bit A30600 is ON when the CPU Rack Power Supply Unit start input terminals are OFF.

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, and the PC Setup are set so the PC does not run when an I/O Verification Error occurs. The PC Setup can be changed to enable operation during I/O Verification Errors. Refer to “Comparison error process” in the PC Setup.

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, or a terminator missing in the SYSMAC BUS/2 System.

3-6-18 Peripheral Device Flags

Connected Device Code (A30608 to A30611)

Bits A30608 through A30611 contain a binary code that identifies the type of Peripheral Device (0: FIT10; 1: FIT20; 2: GPC; 3: Programming Console) connected to the CPU, the Expansion CPU, or an Expansion I/O Rack.

Peripheral Connected Flag (A30615)

Bit A30615 is ON when a Peripheral Device is connected to the CPU, the Expansion CPU, or an Expansion I/O Rack.

SYSMAC BUS/2 Peripheral Flags (A307 and A308)

Bits A30700 through A30815 are turned ON when a Peripheral 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

Peripheral Device Cycle Time (A309)

Word A309 contains the cycle time in ms (in binary) required to service Peripheral 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 is not formatted or a formatting error has occurred.

Memory Card Transfer Error Flag (A34308)

Bit A34308 is turned ON when an error occurs while writing to the Memory Card.

Memory Card Write Error Flag (A34309)

Bit A34309 is turned ON when the Memory Card cannot be written to because the card is write-protected, EPROM, EEPROM, data is beyond the capacity of the card, or there are too many files.

Memory Card Read Error Flag (A34310)

Bit A34310 is turned ON when the specified file is damaged and cannot be read.

File Missing Flag (A34311) Bit A34311 is turned ON when the specified file is not on the installed card or no card is installed.

Memory Card Write Flag (A34312) Bit A34312 is turned ON when the Memory Card is being written to from the program (FILW(181)).

Memory Card Instruction Flag (A34313) Bit A34313 is turned ON when an instruction affecting Memory Card files (FILR(180), FILW(181), FILP(182), or FLSP(183)) is being executed.

Accessing Memory Card Flag (A34314) Bit A34314 is turned ON when the Memory Card is being accessed.

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.

Number of Words to Transfer (A346) Word A346 contains the number of words left to transfer to or from the Memory Card (FILW(181) or FILR(180)). When either instruction is first executed, the number of words in the file is placed in word A346 and as data is transferred, the number of words transferred is subtracted from this number.

The number of words to transfer is recorded in 4-digit BCD.

A346 bits			
15 to 12	**11 to 08**	**07 to 04**	**03 to 00**
10^3	10^2	10^1	10^0

3-6-21 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 8 Error Processing* for details on error codes.

3-6-22 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.

3-6-23 SFC Fatal Error Flag and Error Code

Bit A40107 is turned ON if an error that stops operation occurs while the SFC program is being executed. The SFC Fatal Error Code is written in BCD to word A414. Refer to *Section 8 Error Processing* for details on the error codes.

3-6-24 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.

3-6-25 Program Error Flag

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

3-6-26 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.

3-6-27 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 Ex-

pansion Racks is written to word A407; in the SYSMAC BUS/2 system, to word A408; and in the SYSMAC BUS system, to word A478.

3-6-28 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.

3-6-29 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.

3-6-30 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).

3-6-31 Memory Error Flag

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

3-6-32 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.

3-6-33 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 a error occurs, the bit corresponding to the unit number of the CPU Bus Unit involved is turned ON.

3-6-34 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.

3-6-35 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.

3-6-36 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
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

3-6-37 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.

3-6-38 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.

3-6-39 SFC Non-fatal Error Flag and Error Code

Bit A40211 is turned ON if an error that does not stop operation occurs while the SFC program is being executed. The error code is written to word A418. Refer to *Section 8 Error Processing* for details on the error codes.

3-6-40 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 PC memory addresses. BCD will access other DM words according to DM Area addresses. If binary addresses are used, this flag will not operate.

3-6-41 Jump Error Flag

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

3-6-42 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

3-6-43 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
01: Memory Card
02: I/O Memory
03: EM
04: PC Setup
05: I/O Table
06: System Memory
07: Routing Tables
08: CPU Bus Unit Software Switches

3-6-44 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.

3-6-45 CPU-recognized Rack Numbers

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

3-6-46 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).

3-6-47 CPU Bus Link Error Flag

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

3-6-48 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	10^6	10^5	10^4	10^3
A462	10^2	10^1	10^0	10^{-1}

3-6-49 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	10^6	10^5	10^4	10^3
A464	10^2	10^1	10^0	10^{-1}

3-6-50 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.**

3-6-51 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 Peripheral 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 405 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.

3-6-52 Step Flag

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

3-6-53 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. When SFC programming is used, A50015 turns ON for one cycle at the beginning of action program execution. A50015 also turns ON at the beginning of scheduled interrupt execution. 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*.

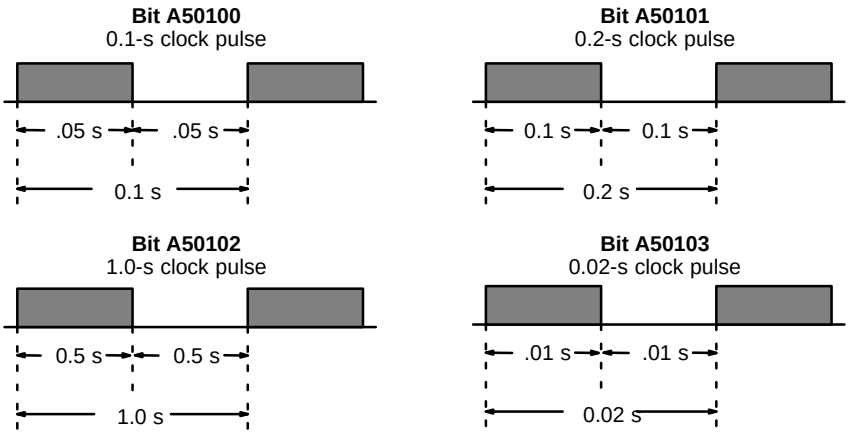
Note Do not use A50015 to control execution of differentiated instructions. The instructions will never be executed.

3-6-54 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%.

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.

Pulse width	0.1 s	0.2 s	1.0 s	0.02 s
Bit	A50100	A50101	A50102	A50103



Caution Because the 0.1-second and 0.02-second clock pulse bits have ON times of 50 and 10 ms, respectively, the CPU may not be able to accurately read the pulses if program execution time is too long.

3-6-55 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) in either a SYSMAC NET Link or SYSMAC LINK System. 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 *SYSMAC NET Link System Manual* or *SYSMAC LINK System Manual* for details on completion codes.

3-6-56 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 Transition Area

A transition is a condition which moves the active status from one step to the next in the SFC program. Flags in the Transition Area are turned ON when a TOUT(202) instruction is executed with an ON execution condition, or a TCNT(123) counter times out.

The CV500 has 512 Transition Flags, numbered TN0000 to TN0511, and the CV1000 or CV2000 has 1,024 Transition Flags, numbered TN0000 to TN1023.

Input the transition number as a bit operand when designating Transition Flags in instructions.

The CVM1 does not support SFC programming and is not equipped with a Transition Area.

3-8 Step Area

A step in the program represents a single process. All SFC programs are executed by step. Each step must have its own unique step number. Flags in the Step Area are turned ON to indicate that a step is active.

The CV500 has 512 Step Flags, numbered ST0000 to ST0511, and the CV1000 or CV2000 has 1,024 Step Flags, numbered ST0000 to ST1023. Input the step number as a bit operand when designating Step Flags in instructions.

The CVM1 does not support SFC programming and is not equipped with a Step Area.

3-9 Timer Area

Timer Completion Flags and present values (PV) are accessed through timer numbers ranging from T0000 through T0511 for the CV500 or CVM1-CPU01-EV2 and from T0000 through T1023 for the CV1000, CV2000, CVM1-CPU11-EV2, or CVM1-CPU21-EV2. 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 require 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.

When the cycle time exceeds 10 ms, define TIMH(015) instructions with timer numbers T0000 through T0127 for the CV500 or CVM1-CPU01-EV2, and T0000 through T0255 for the CV1000, CV2000, CVM1-CPU11-EV2, or CVM1-CPU21-EV2, to ensure accuracy.

TIM timers are not affected by the cycle time, but TTIM(120) timers time slowly if the cycle time exceeds 100 ms.

3-10 Counter Area

Counter Completion Flags and present values (PV) are accessed through counter numbers ranging from C0000 through C0511 for the CV500 or CVM1-CPU01-EV2 and from C0000 through C1023 for the CV1000, CV2000, CVM1-CPU11-EV2, or CVM1-CPU21-EV2. 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.

3-11 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 D00000 through D08191 for the CV500 or CVM1-CPU01-EV2; from D00000 through D24575 for the CV1000, CV2000, CVM1-CPU11-EV2, or CVM1-CPU21-EV2.

The EM (Extended Data Memory) Area is available with the CV1000, CV2000, or CVM1-CPU21-EV2 only and only if purchased and installed as an option. EM Area and DM Area functions are identical. The main difference between EM and DM is that DM is internal, while EM is contained on an EM Unit, a card that fits into a slot on the CV1000, CV2000, or CVM1-CPU21-EV2 CPU. There are three models of Memory Units available, with 64K words (E00000 to E32765 $\times$ 2 banks), 128K words (E00000 to E32765 $\times$ 4 banks), and 256K words (E00000 to E32765 $\times$ 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 CVSS/SSS. The 1,l600 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.

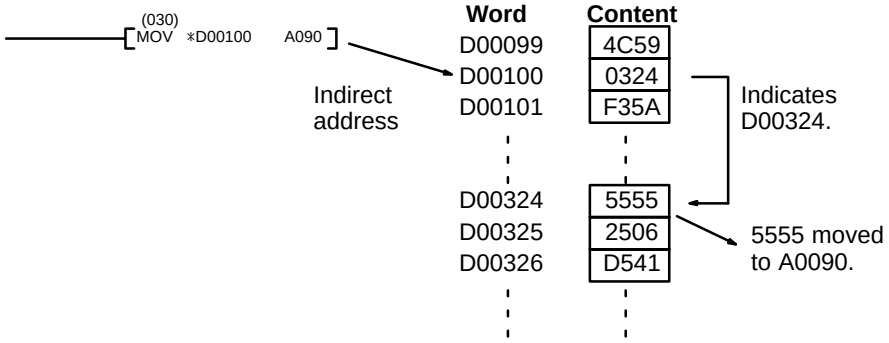
Indirect Addressing

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.

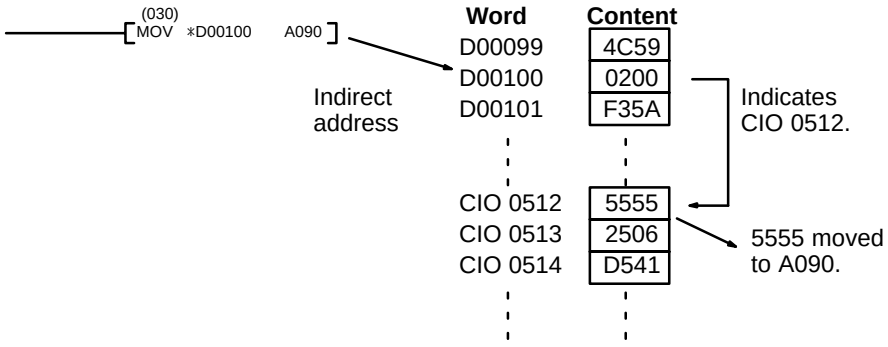
If the PC Setup define the content of a *DM (or *EM) address as BCD, the number indicates another DM (or EM) address. If the contents of the *DM address are not BCD, a *DM BCD error will occur, and an error flag, A50003, will be turned ON. Because only the last four digits of the final address can be 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 PC Setup define the content of a *DM address as binary, the number indicates a PC memory address. The range of possible binary numbers, $0000 to $FFFF, allows all memory areas, including EM, to be indirectly addressed.

wIf, in this case, the content of D00100 is $0324, then *D00100 indicates PC 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.



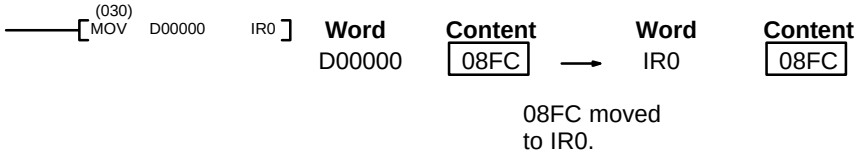
Indirect addressing can also be used in instructions that require bit operands for bits in the Core I/O Area ($0000 to $0FFF). These bits are designated by using the rightmost digits of the memory address as the leftmost three digits of the hexadecimal address and adding the bit number as the rightmost 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.

3-12 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.

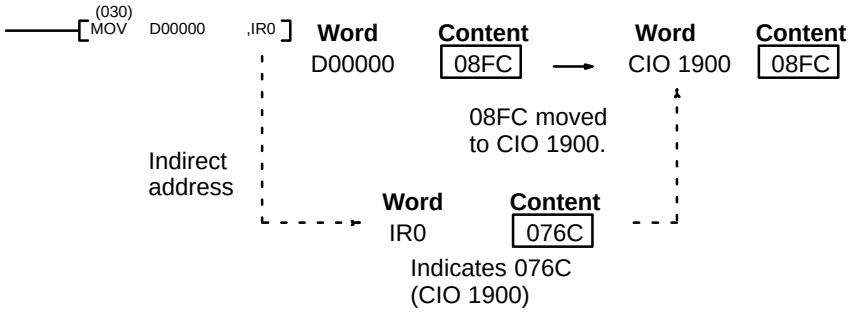
Direct Addressing

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 PC memory address indicated by that Index Register, as in the following example.

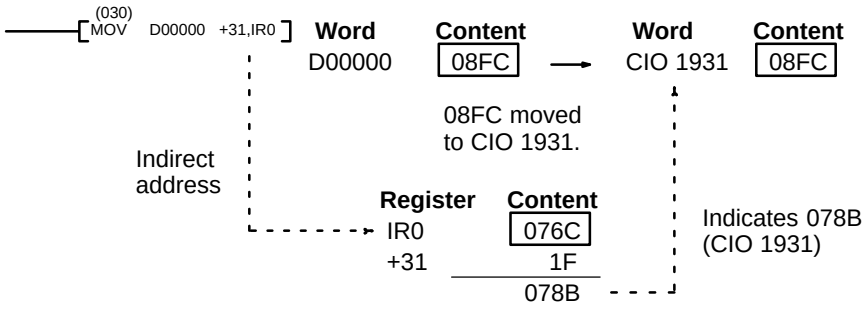


Indirect addressing can also be used in instructions that require bit operands for bits in the Core I/O Area ($0000 to $0FFF). These bits are designated by using the rightmost digits of the memory address as the leftmost three digits

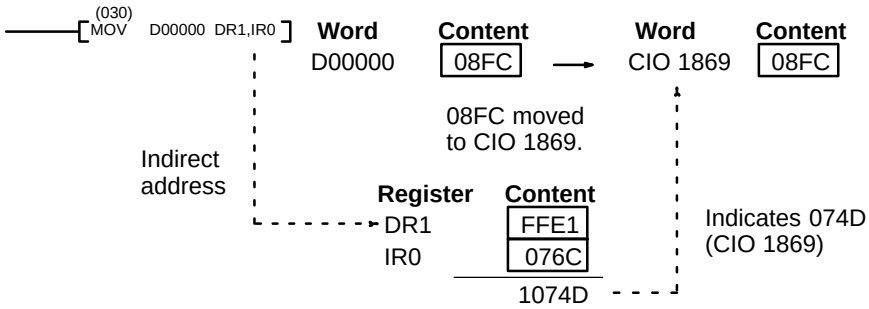
of the hexadecimal address and adding the bit number as the rightmost digit. For example, the CIO bit 190000 is designated by $76CA where 76C is the rightmost three digits of the memory address (CIO word 1900 is $076C) and A is bit 10.

Offset Indirect Addressing

The PC 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 PC 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

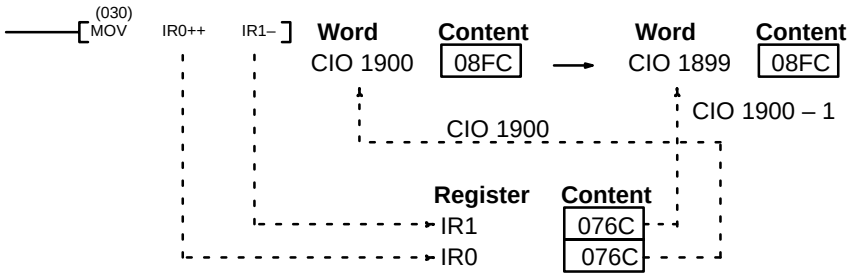
An auto-increment increases the contents of an Index Register by 1 or 2 after executing the instruction. A "+" suffix indicates an auto-increment of 1, and a "++" suffix indicates an auto-increment of 2.

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. It also introduces the new version-2 CVM1 CPUs instructions and explains the data formats that they can utilize.

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

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. If SFC programming is being used, you will need to write a ladder diagram for each action program and each transition program. You will also need to write interrupt programs if they are required.

   Note The CVM1 does not support SFC programming.
6. Input the program into the CPU. Actual input is done from the CVSS 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 8 Error Processing* provides information required for debugging. Refer to the *CVSS Operation Manuals* for program input, debugging, and monitoring procedures.

4-2 Instruction Terminology

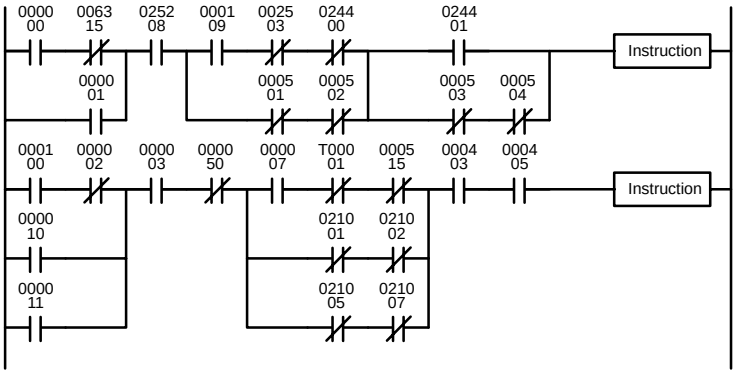
There are basically two types of instructions used in ladder-diagram programming: instructions that correspond to the conditions on the rungs of the ladder diagram and are used in instruction form only when converting a program to mnemonic code, and instructions that are used on the right side of the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



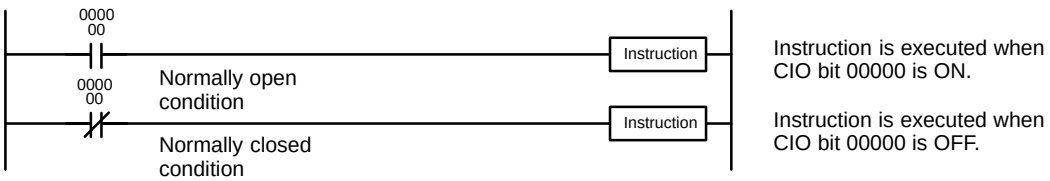
As shown in the diagram above, instruction lines can branch apart and they can join back together. The short vertical pairs of lines are called **conditions**. Conditions without diagonal lines through them are called **normally open conditions** and correspond to a LOAD, AND, or OR instruction. The 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.



Execution Conditions

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.

Operand Bits

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 *4-5-1 TR Bits* for details.

Logic Blocks

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.

Block Programming

Version-2 CVM1 CPUs support the block programming instructions of the C1000H and C2000H. 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 Peripheral 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.

Program Memory Structure

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.

With a CV-series PC, 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 CVSS, most operands are separated only by spaces. Refer to the *CVSS 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.

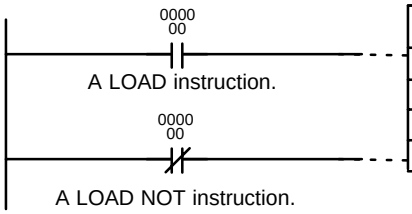
When programming, addresses are automatically displayed and do not have to be input unless for some reason a different location is desired for the instruction. When converting to mnemonic code, it is best to start at Program Memory address 00000 unless there is a specific reason for starting elsewhere.

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.

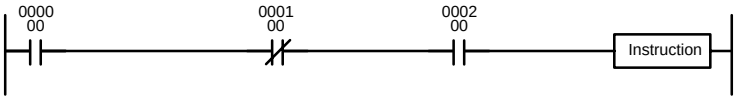


Address	Instruction	Operands
00000	LD	000000
00001	Instruction	
00002	LD NOT	000000
00003	Instruction	

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.



Address	Instruction	Operands
00000	LD	000000
00001	AND NOT	000100
00002	AND	000200
00003	Instruction	

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.

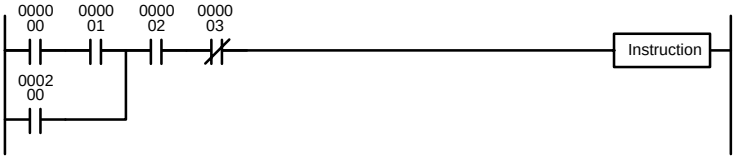
Address	Instruction	Operands
00000	LD NOT	000000
00001	OR NOT	000100
00002	OR	000200
00003	Instruction	

The instruction at the right would have an ON execution condition when any one of the three conditions was ON, i.e., when CIO 00000 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.



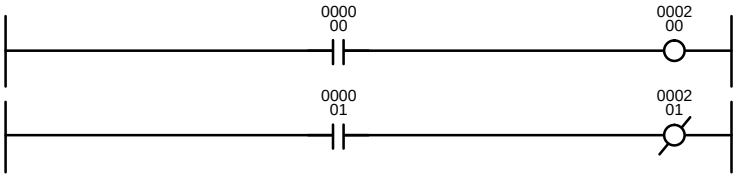
Address	Instruction	Operands
00000	LD	000000
00001	AND	000001
00002	OR	000200
00003	AND	000002
00004	AND NOT	000003
00005	Instruction	

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.



Address	Instruction	Operands
00000	LD	000000
00001	OUT	000200

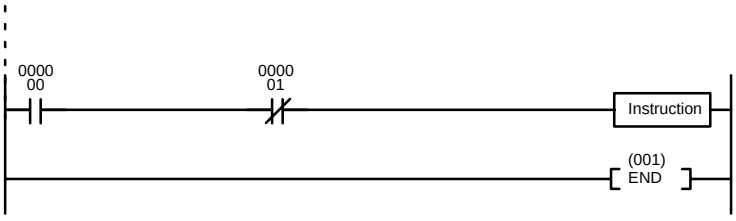
Address	Instruction	Operands
00000	LD	000001
00001	OUT NOT	000201

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.



Address	Instruction	Operands
00500	LD	000000
00501	AND NOT	000001
00502	Instruction	
00503	END(001)	---

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

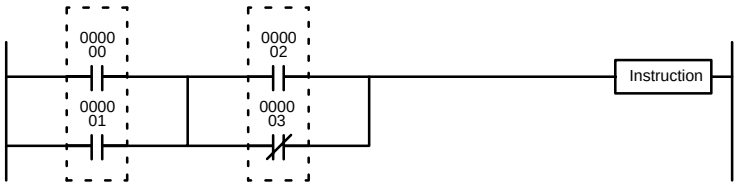
4-4 Mnemonic Code

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.



Address	Instruction	Operands
00000	LD	000000
00001	OR	000001
00002	LD	000002
00003	OR NOT	000003
00004	AND LD	---

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).

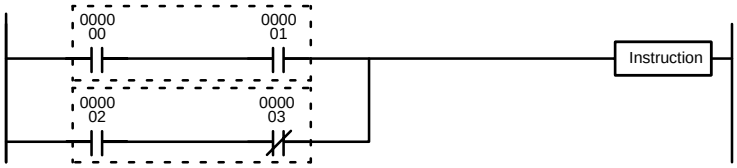
The above ladder diagram cannot be converted to mnemonic code using AND and OR instructions alone. If an AND between CIO 000002 and the results of an OR between CIO 000000 and CIO 000001 is attempted, the OR NOT between CIO 000002 and CIO 000003 is lost and the OR NOT ends up being an OR NOT between 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.



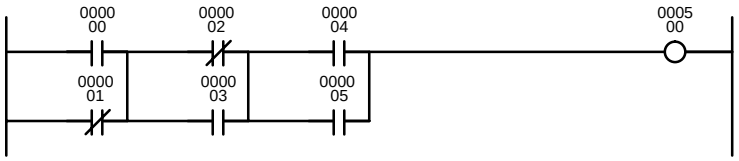
Address	Instruction	Operands
00000	LD	000000
00001	AND	000001
00002	LD	000002
00003	AND NOT	000003
00004	OR LD	---

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

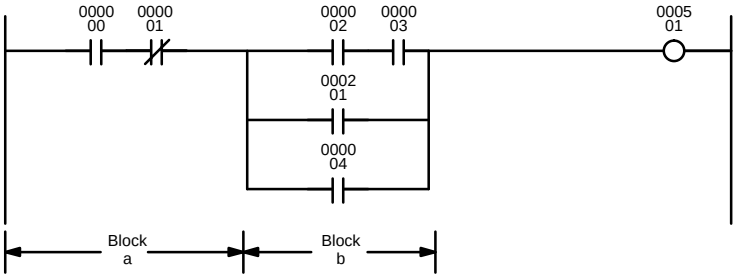
The first of each pair of conditions is converted to LOAD with the assigned bit operand and then ANDed with the other condition. The first two blocks can be coded first, followed by OR LOAD, the last block, and another OR LOAD, or the three blocks can be coded first followed by two OR LOADs. The mnemonic 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.

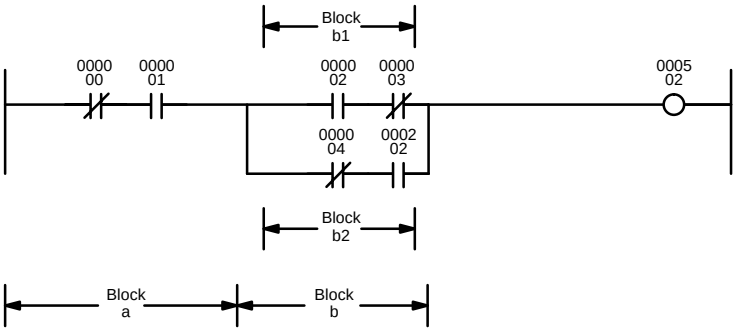
The following diagram contains only two logic blocks as shown. It is not necessary to further separate block b components, because it can be coded directly using only AND and OR.



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.



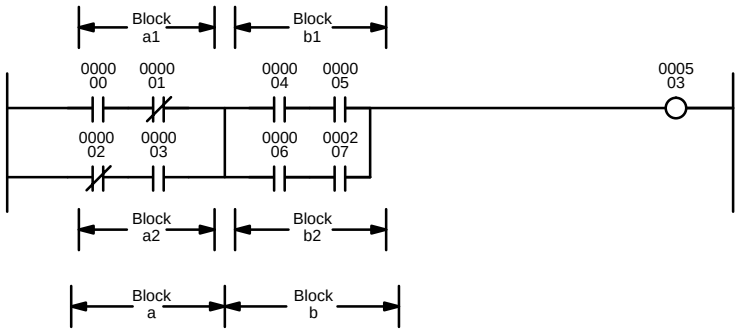
Address	Instruction	Operands
00000	LD NOT	000000
00001	AND	000001
00002	LD	000002
00003	AND NOT	000003
00004	LD NOT	000004
00005	AND	000202
00006	OR LD	—
00007	AND LD	—
00008	OUT	000502

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.

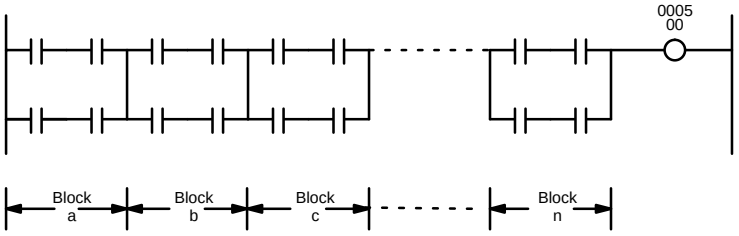
When working with complicated diagrams, blocks will ultimately be coded starting at the top left and moving down before moving across. This will generally mean that, when there might be a choice, OR LOAD will be coded before AND LOAD.

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.

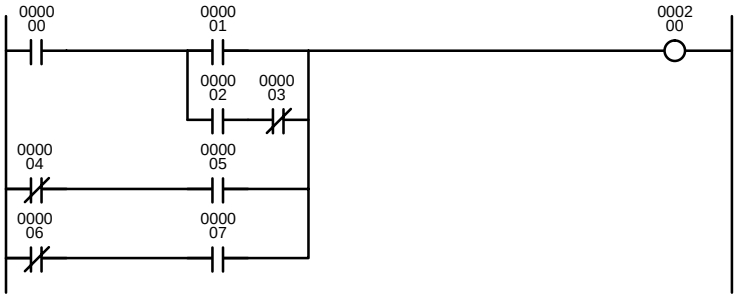


Address	Instruction	Operands	
00000	LD	000000	
00001	AND NOT	000001	
00002	LD NOT	000002	
00003	AND	000003	
00004	OR LD	—	Blocks a1 and a2
00005	LD	000004	
00006	AND	000005	
00007	LD	000006	
00008	AND	000007	
00009	OR LD	—	Blocks b1 and b2
00010	AND LD	—	Blocks a and b
00011	OUT	000503	

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.

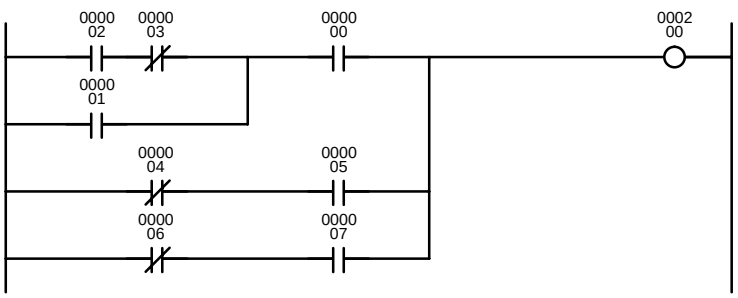


The following diagram requires an OR LOAD followed by an AND LOAD to code the top of the three blocks, and then two more OR LOADs to complete the mnemonic code.



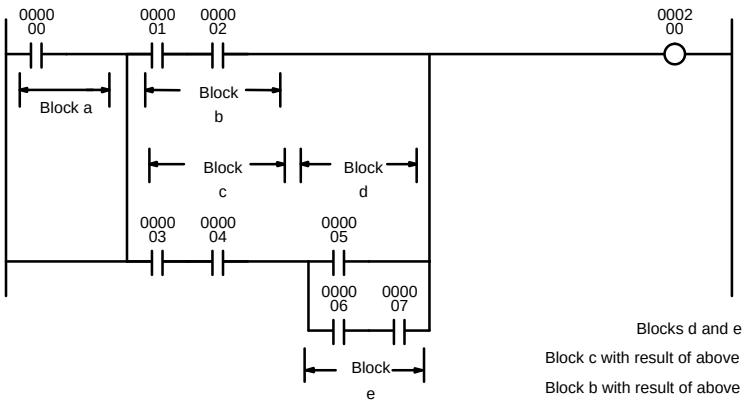
Address	Instruction	Operands
00000	LD	000000
00001	LD	000001
00002	LD	000002
00003	AND NOT	000003
00004	OR LD	—
00005	AND LD	—
00006	LD NOT	000004
00007	AND	000005
00008	OR LD	—
00009	LD NOT	000006
00010	AND	000007
00011	OR LD	—
00012	OUT	000200

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.



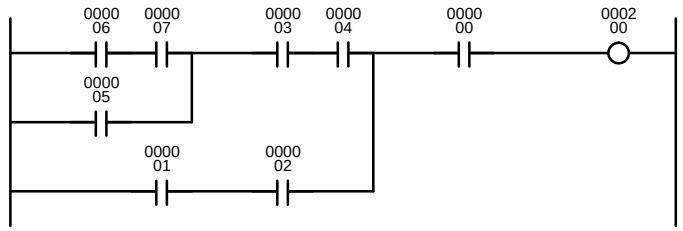
Address	Instruction	Operands
00000	LD	000002
00001	AND NOT	000003
00002	OR	000001
00003	AND	000000
00004	LD NOT	000004
00005	AND	000005
00006	OR LD	—
00007	LD NOT	000006
00008	AND	000007
00009	OR LD	—
00010	OUT	000200

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.



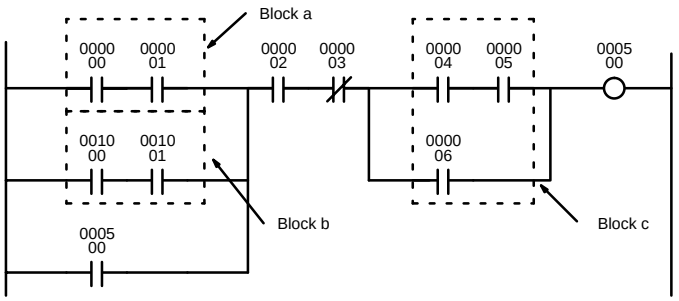
Address	Instruction	Operands
00000	LD	000000
00001	LD	000001
00002	AND	000002
00003	LD	000003
00004	AND	000004
00005	LD	000005
00006	LD	000006
00007	AND	000007
00008	OR LD	—
00009	AND LD	—
00010	OR LD	—
00011	AND LD	—
00012	OUT	000200

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

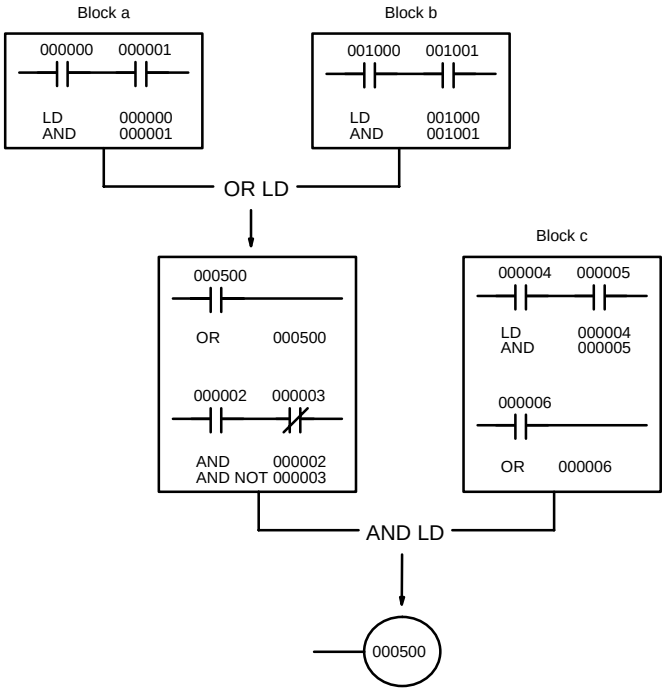


Address	Instruction	Operands
00000	LD	000006
00001	AND	000007
00002	OR	000005
00003	AND	000003
00004	AND	000004
00005	LD	000001
00006	AND	000002
00007	OR LD	—
00008	AND	000000
00009	OUT	000200

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:



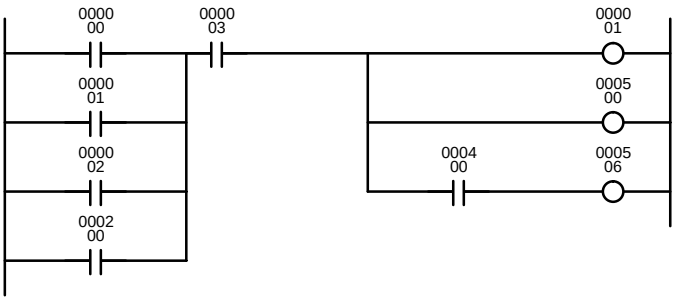
The first logic block instruction is used to combine the execution conditions resulting from blocks a and b, and the second one is to combine the execution condition of block c with the execution condition resulting from the normally closed condition assigned CIO 000003. The rest of the diagram can be 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.

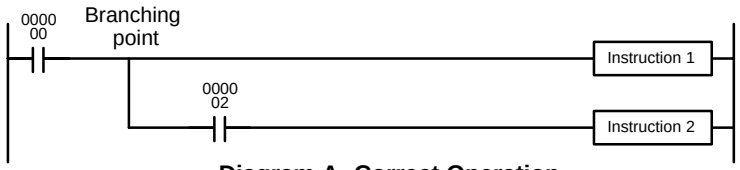


Diagram A: Correct Operation

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

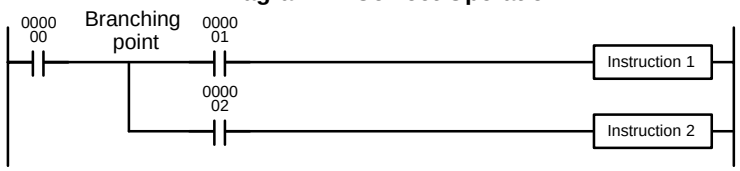


Diagram B: Incorrect Operation

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 CVSS, it is not necessary to input TR bits and none will appear on the screen. The CVSS 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

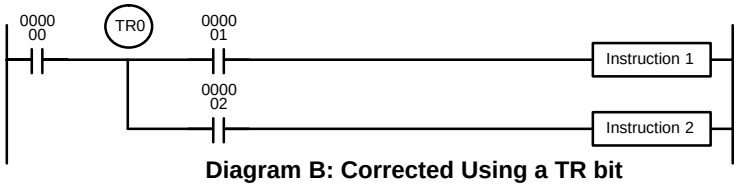
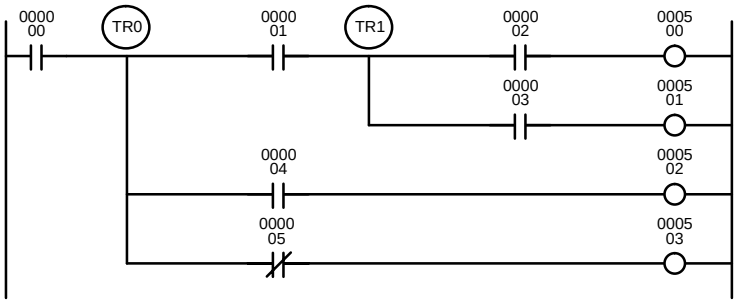


Diagram B: Corrected Using a TR bit

Address	Instruction	Operands
00000	LD	000000
00001	OUT	TR0
00002	AND	000001
00003	Instruction 1	
00004	LD	TR0
00005	AND	000002
00006	Instruction 2	

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.



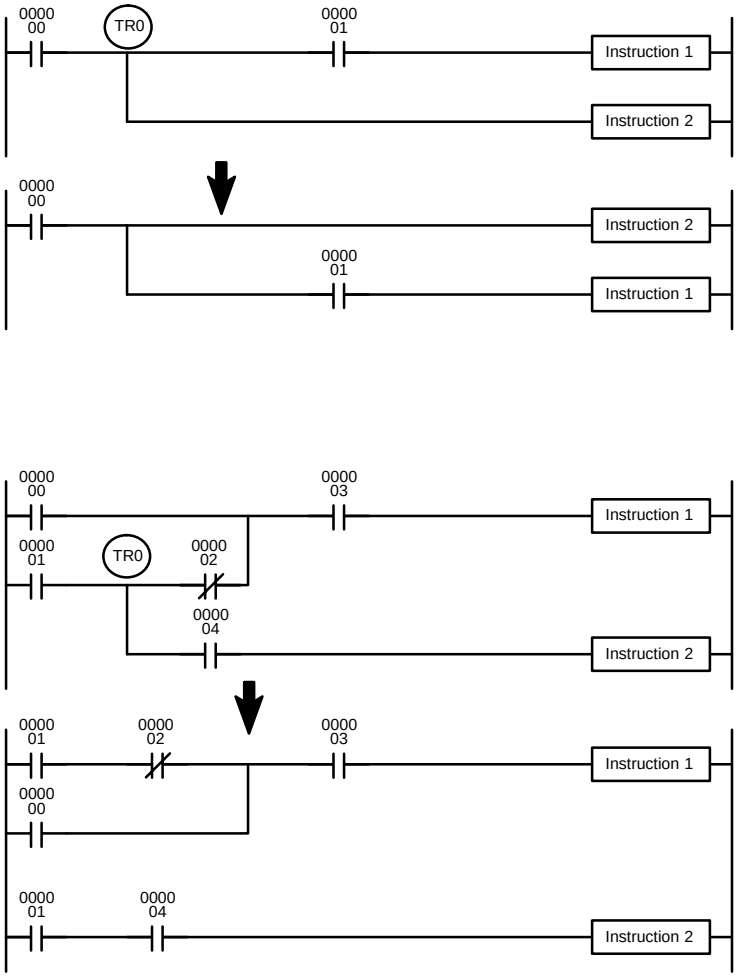
Address	Instruction	Operands
00000	LD	000000
00001	OUT	TR0
00002	AND	000001
00003	OUT	TR1
00004	AND	000002
00005	OUT	000500
00006	LD	TR1
00007	AND	000003
00008	OUT	000501
00009	LD	TR0
00010	AND	000004
00011	OUT	000502
00012	LD	TR0
00013	AND NOT	000005
00014	OUT	000503

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 required 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.

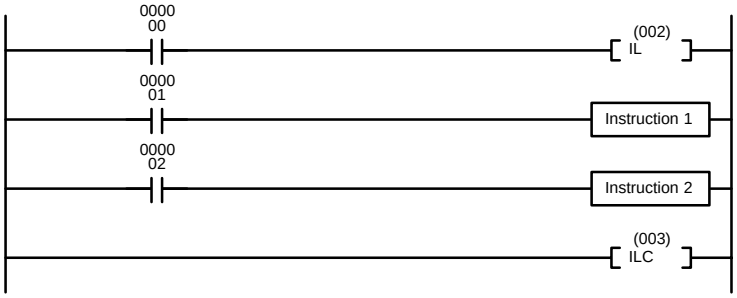


4-5-2 Interlocks

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)*.

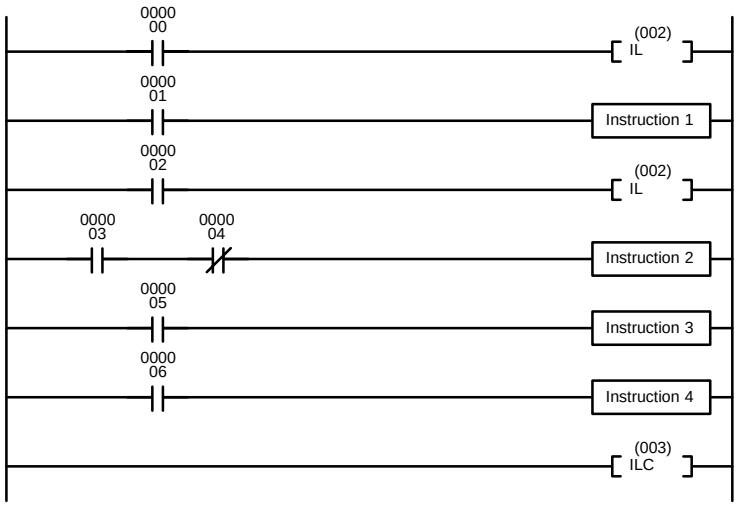
Diagram B can also be corrected with an interlock. Here, the conditions leading up to the branching point are placed on an instruction line for the INTERLOCK instruction, all of lines leading from the branching point are written as separate instruction lines, and another instruction line is added for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



Address	Instruction	Operands
00000	LD	000000
00001	IL(002)	---
00002	LD	000001
00003	Instruction 1	
00004	LD	000002
00005	Instruction 2	
00006	ILC(003)	---

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.



Address	Instruction	Operands
00000	LD	000000
00001	IL(002)	---
00002	LD	000001
00003	Instruction 1	
00004	LD	000002
00005	IL(002)	---
00006	LD	000003
00007	AND NOT	000004
00008	Instruction 2	
00009	LD	000005
00010	Instruction 3	
00011	LD	000006
00012	Instruction 4	
00013	ILC(003)	---

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 required 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.

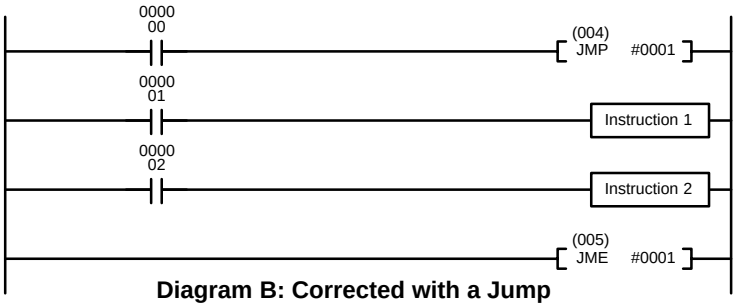


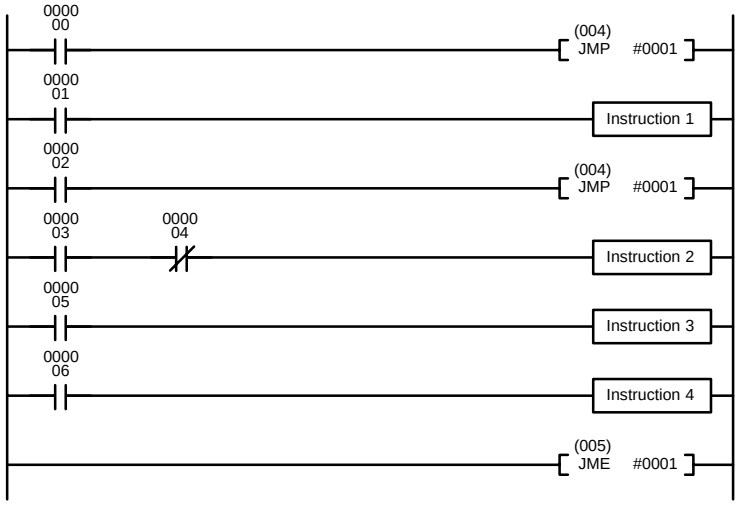
Diagram B: Corrected with a Jump

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

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. If SFC programming is being used, the JUMP END instruction must be contained within the same action or transition 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 499). 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.

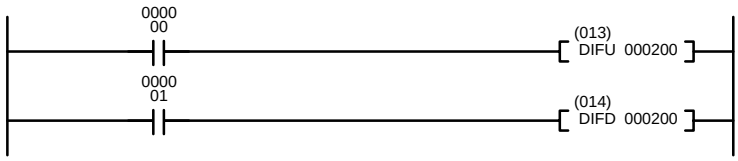
Note Version-2 CVM1 CPUs also support the CJP(221) and CJPN(222) jump instructions that can also be used to create jumps in programs. Refer to *Section 5 Instruction Set* for details.

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.



Address	Instruction	Operands
00000	LD	000000
00001	DIFU(013)	000200
00002	LD	000001
00003	DIFD(014)	000200

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.

Note Version-2 CVM1 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.

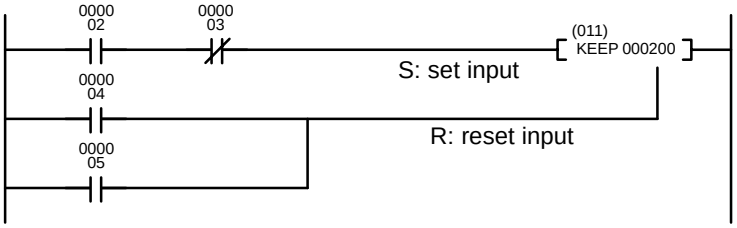
Address	Instruction	Operands
00000	LD	000000
00001	SET(015)	000200
00002	LD	000001
00003	RSET(016)	000201

In the above example, CIO 000200 will be turned ON when CIO 000000 goes ON and will remain ON even after CIO 00000 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 00000 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.



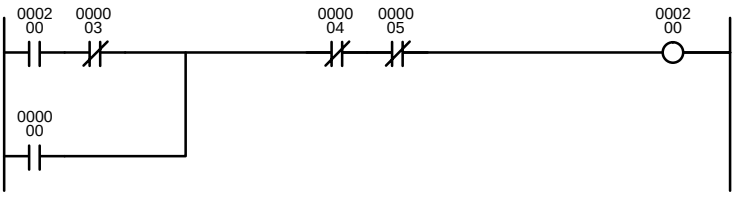
Address	Instruction	Operands
00000	LD	000002
00001	AND NOT	000003
00002	LD	000004
00003	OR	000005
00004	KEEP(011)	000200

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.

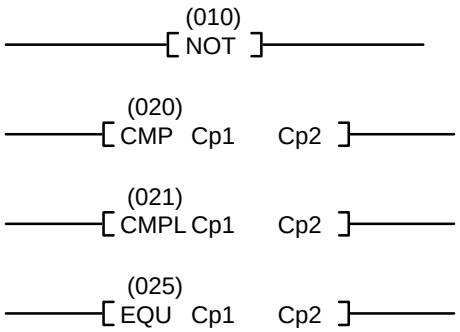
The above diagram for the KEEP instruction can be rewritten as shown below. The only difference in these diagrams would be their operation in an interlocked program section when the execution condition for the INTERLOCK instruction was ON. Here, just as in the same diagram using the KEEP instruction,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



Address	Instruction	Operands
00000	LD	000200
00001	AND NOT	000003
00002	OR	000000
00003	AND NOT	000004
00004	AND NOT	000005
00005	OUT	000200

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 *4-12-1 Input Comparison Instructions* 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 works 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

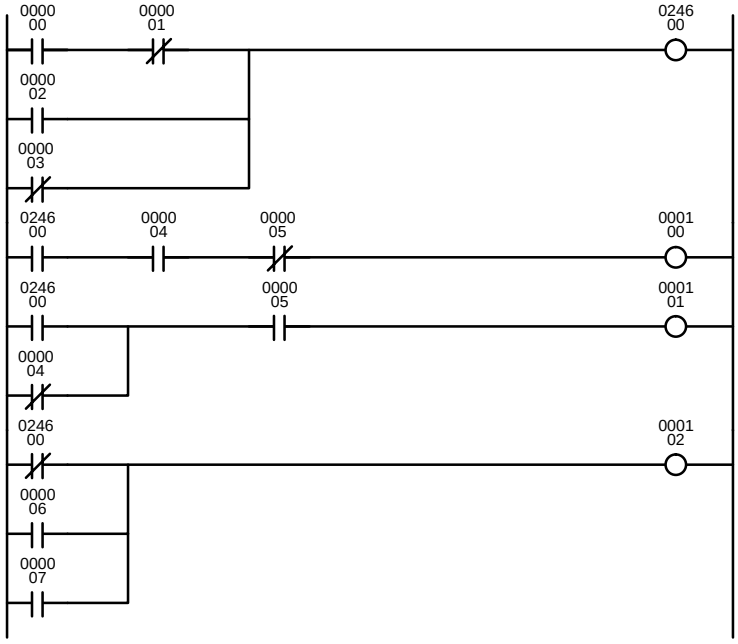
Examples given later in this subsection show two of the most common ways to employ work bits. These should act as a guide to the almost limitless number of ways in which the work bits can be used. Whenever difficulties arise in programming a control action, consideration should be given to work bits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

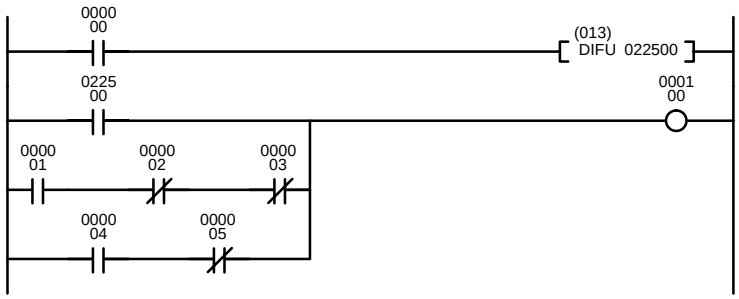


Address	Instruction	Operands
00000	LD	000000
00001	AND NOT	000001
00002	OR	000002
00003	OR NOT	000003
00004	OUT	024600
00005	LD	024600
00006	AND	000004
00007	AND NOT	000005
00008	OUT	000100
00009	LD	024600
00010	OR NOT	000004
00011	AND	000005
00012	OUT	000101
00013	LD NOT	024600
00014	OR	000006
00015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

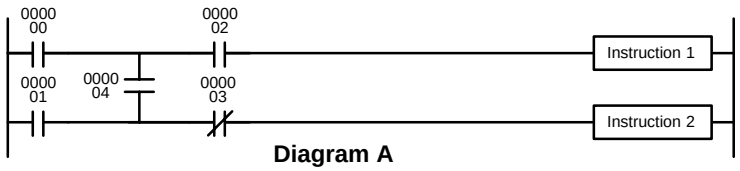


Diagram A

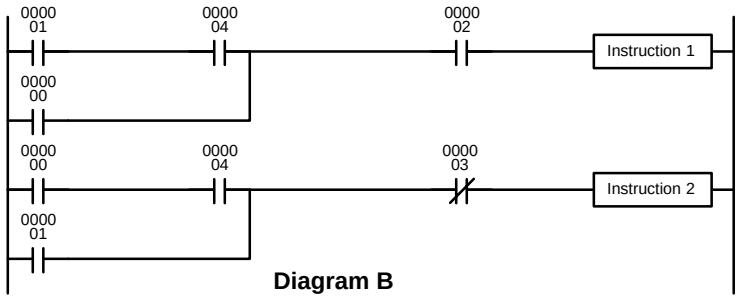


Diagram B

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.

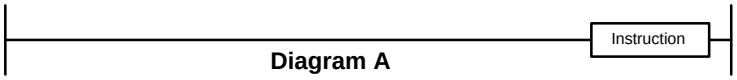


Diagram A

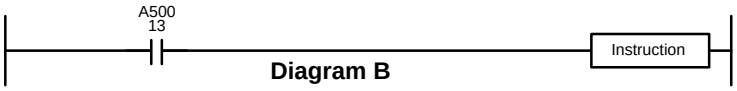
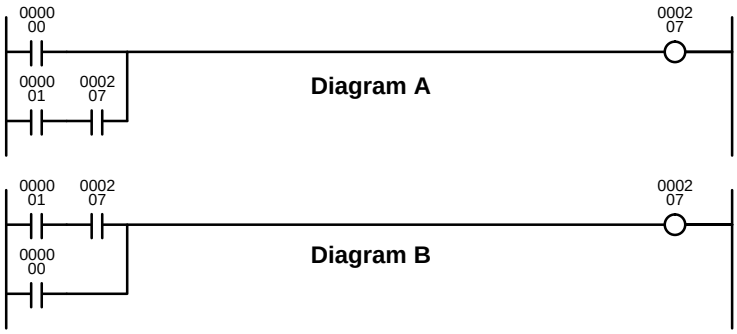


Diagram B

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.



Address	Instruction	Operands
00000	LD	000000
00001	LD	000001
00002	AND	000207
00003	OR LD	---
00004	OUT	000207

Address	Instruction	Operands
00000	LD	000001
00001	AND	000207
00002	OR	000000
00003	OUT	000207

4-11 Program Execution

When execution or 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 scan time. Refer to *Section 6 Program Execution Timing* for details.

4-12 Using Version-2 CVM1 CPUs

The most significant improvement that the version-2 CVM1 CPUs offers in comparison with version-1 CPUs is a greatly enhanced instruction set. This section explains the basics that the user should be familiar with before attempting to use the new instructions. All of these instructions can be used when the SYSMAC Support Software and the CVM1-PRS21-EV1 Programming Console are used. They are not supported by the CVSS or other Programming Devices.

4-12-1 Input Comparison Instructions

The version-2 CVM1 CPUs provide 24 new comparison instructions. The functions of these instructions are shown as symbols, making them easy to understand at a glance.

Most of these instructions are shown with a symbol and options. When the options are not included, the instructions will handle unsigned one-word data.

Symbol		Option (data format)	Option (data length)
=	(Equal)	S (signed)	L (double)
< >	(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) can handle signed binary data. For information concerning signed binary data, refer to *4-13 Data Formats*.

The following table shows the function codes, mnemonics, and names of all of the input comparison instructions. For details, refer to *5-16-7 Input Comparison Instructions (300 to 328)*.

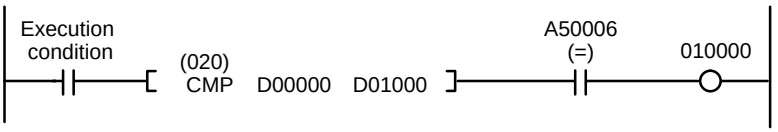
Code	Mnemonic	Name
300	=	EQUAL
301	=L	DOUBLE EQUAL
302	=S	SIGNED EQUAL
303	=SL	DOUBLE SIGNED EQUAL
305	<>	NOT EQUAL
306	<>L	DOUBLE NOT EQUAL
307	<>S	SIGNED NOT EQUAL
308	<>SL	DOUBLE SIGNED NOT EQUAL
310	<	LESS THAN
311	<L	DOUBLE LESS THAN
312	<S	SIGNED LESS THAN
313	<SL	DOUBLE SIGNED LESS THAN
315	<=	LESS THAN OR EQUAL
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
321	>L	DOUBLE GREATER THAN
322	>S	SIGNED GREATER THAN
323	>SL	DOUBLE SIGNED GREATER THAN
325	>=	GREATER THAN OR EQUAL
326	>=L	DOUBLE GREATER THAN OR EQUAL
327	>=S	SIGNED GREATER THAN OR EQUAL
328	>=SL	DOUBLE SIGNED GREATER THAN OR EQUAL

Features

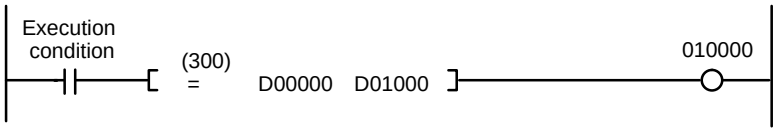
With the earlier comparison instructions, CMP(020) and CMPL(021), the comparison result was output to the Greater Than Flag (A50005), Equals Flag (A50006), and Less Than Flag (A50007), and those flags then had to serve as the input condition for subsequent processing in accordance with the comparison result.

With the input comparison instructions, however, the comparison results are directly reflected as the input condition for the next instruction. This simplifies programming requirements by eliminating the need to use flags for that purpose.

CMP(020) Example



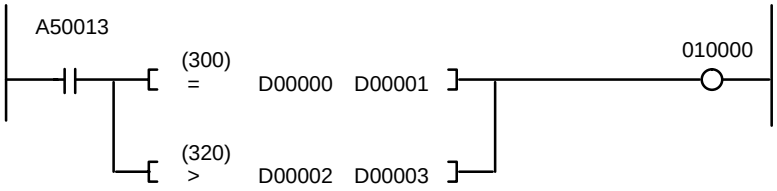
Input Comparison Instruction Example



Precautions

Input comparison instructions must have an execution condition preceding them on the instruction line; they cannot be directly connected to the left bus line. In addition, because they are intermediate instructions, they must have another instruction following them on the same instruction line. As shown in the example above, place the execution condition, the instruction, and the output (or other right-hand instruction) in order.

Multiple input comparison instructions can be used together, as shown in the following example.



To input this instruction block using the Programming Console, input the following mnemonics.

LD	A50013	
OUT	TR0	
= (300)	D00000	D00001
LD	TR0	
>(320)	D00002	D00003
OR LD		
OUT	010000	

Instruction Input Methods

There are two ways to input input comparison instructions. The first is to input the symbol directly and the second is to input the function code.

Direct Symbol Input

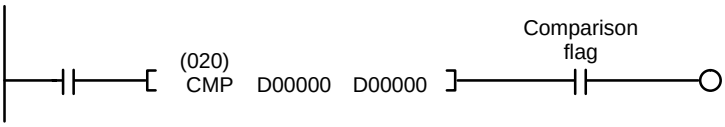
Direct string inputs are possible using the SYSMAC Support Software. Input the symbol and the options in order. For example, “>=L(326)” can be entered by simply inputting “>=L.”

Function Code Input

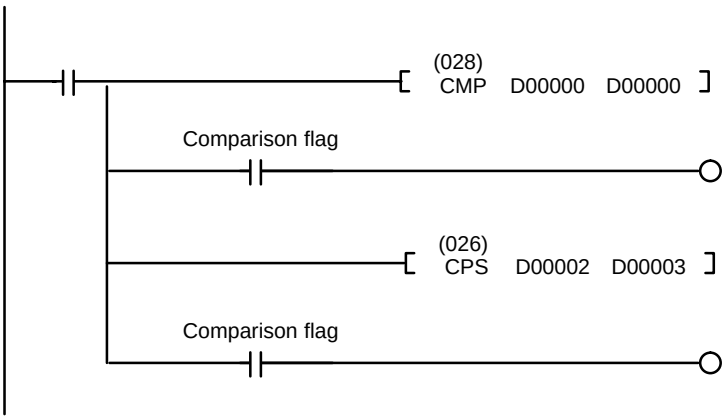
Function codes can be input using the SYSMAC Support Software or the CVM1-PRS21-EV1 Programming Console. Simply input the instruction’s function code.

4-12-2 CMP and CMPL

CMP(020) and CMPL(021) are the same in the CV-series and CVM1-series as in the C-series in that they all output the comparison results to comparison flags. There are differences, however, in the way that are depicted in ladder diagrams.



In the version-2 CVM1 CPUs, CMP(028) and CMPL(029) operate the same as comparison instructions in C-series PC in that they output results to comparison flags, but they are programmed as right-hand instructions. The signed binary comparison instructions CPS(026) and CPSL(027) also output results to the comparison flags, but are programmed as right-hand instructions in the same was as for comparison instructions in the C-series PCs, as shown in the following program section.



In the version-2 CVM1 CPUs, CMP(028), CMPL(029), CPS(026), and CPSL(027) are not intermediate instructions, and no other instructions can be programmed on the same instruction line between them and the right-hand bus bar.

Precautions when Programming

Input methods and instruction sets vary according to which support software is used.

CV Support Software

Entering “CMP” or “CMPL” by means of string input specifies CMP(020) or CMPL(021).

SYSMAC Support Software

Entering “CMP” or “CMPL” by means of string input specifies CMP(028) or CMPL(029). In order to input CMP(020) or CMPL(021), it is necessary to input the function code.

Converting Programs

When a C-series ladder program is converted from C to CV using the SYSMAC Support Software, CMP and CMPL instructions in the program are converted to CMP(028) and CMPL(029), respectively.

4-12-3 Enhanced Math Instructions

The version-2 CVM1 CPUs provides symbol math instructions as an improvement over the earlier BCD and binary math instructions. The basic data format for these instructions is signed binary, although unsigned, BCD, and floating-point data options can be specified. The functions of these instructions are shown as symbols, making them easy to understand at a glance.

Most of these instructions are shown with a symbol and options. When the options are not included, the instruction will handle data as signed one-word binary data without carry.

Symbol		Data format options	Carry option	Data length option
+	(Add)	B (BCD)	C (with carry)	L (double)
–	(Subtract)	U (Unsigned binary)		
*	(Multiply)	F (Floating point)		
/	(Divide)			

The following table shows the function codes, mnemonics, and names of all of the symbol math instructions. For details, refer to *5-20 Symbol Math Instructions*.

Code	Mnemonic	Name
400	+	SIGNED BINARY ADD WITHOUT CARRY
401	+L	DOUBLE SIGNED BINARY ADD WITHOUT CARRY
402	+C	SIGNED BINARY ADD WITH CARRY
403	+CL	DOUBLE SIGNED BINARY ADD WITH CARRY
404	+B	BCD ADD WITHOUT CARRY
405	+BL	DOUBLE BCD ADD WITHOUT CARRY
406	+BC	BCD ADD WITH CARRY
407	+BCL	DOUBLE BCD ADD WITH CARRY
410	–	SIGNED BINARY SUBTRACT WITHOUT CARRY
411	–L	DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY
412	–C	SIGNED BINARY SUBTRACT WITH CARRY
413	–CL	DOUBLE SIGNED BINARY SUBTRACT WITH CARRY
414	–B	BCD SUBTRACT WITHOUT CARRY
415	–BL	DOUBLE BCD SUBTRACT WITHOUT CARRY
416	–BC	BCD SUBTRACT WITH CARRY
417	–BCL	DOUBLE BCD SUBTRACT WITH CARRY
420	*	SIGNED BINARY MULTIPLY
421	*L	DOUBLE SIGNED BINARY MULTIPLY
422	*U	UNSIGNED BINARY MULTIPLY
423	*UL	DOUBLE UNSIGNED BINARY MULTIPLY
424	*B	BCD MULTIPLY
425	*BL	DOUBLE BCD MULTIPLY
430	/	SIGNED BINARY DIVIDE
431	/L	DOUBLE SIGNED BINARY DIVIDE
432	/U	UNSIGNED BINARY DIVIDE
433	/UL	DOUBLE UNSIGNED BINARY DIVIDE
434	/B	BCD DIVIDE
435	/BL	DOUBLE BCD DIVIDE

Correspondence with Existing Instructions

The following table shows the correspondence between the symbol math instructions and the existing BCD and binary calculation instructions.

Existing instructions		Version-2 instructions
BCD ADD	ADD(070)	+BC (406)
BCD SUBTRACT	SUB(071)	–BC(416)
BCD MULTIPLY	MUL(072)	*B(424)
BCD DIVIDE	DIV(073)	/B(434)
DOUBLE BCD ADD	ADDL(074)	+BCL (407)
DOUBLE BCD SUBTRACT	SUBL(075)	–BCL(417)
DOUBLE BCD MULTIPLY	MULL(076)	*BL(425)
DOUBLE BCD DIVIDE	DIVL(077)	/BL(435)
BINARY ADD	ADB(080)	+C (402)
BINARY SUBTRACT	SBB(081)	–C(412)
BINARY MULTIPLY	MLB(082)	*U(422)
BINARY DIVIDE	DVB(083)	/U(432)
DOUBLE BINARY ADD	ADBL(084)	+CL (403)
DOUBLE BINARY SUBTRACT	SBBL(085)	–CL(413)
DOUBLE BINARY MULTIPLY	MLBL(086)	*UL(423)
DOUBLE BINARY DIVIDE	DVBL(087)	/UL(433)

Instruction Input Methods

There are two ways to input symbol math instructions. The first is to input the symbol directly and the second is to input the function code.

Direct Symbol Input

Direct string inputs are possible using the SYSMAC Support Software. Input the symbol and the options in order. For example, “+BL(405)” can be entered by simply inputting “+BL.”

Function Code Input

Function codes can be input using the SYSMAC Support Software or the CVM1-PRS21-EV1 Programming Console. Simply input the instruction’s function code.

4-13 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

4-13-1 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^{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	16^{3}				16^{2}				16^{1}				16^{0}			

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.

4-13-2 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:
$-[(3 \times 16^3) + (0 \times 16^2) + (3 \times 16^1) + (9 \times 16^0)] = -12,345$

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	
8000	+32,768	
7FFF	+32,767	7FFF
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

4-13-3 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.

4-13-4 Signed BCD Data

Signed BCD data is a 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 version-2 CVM1 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*.

4-13-5 Floating-point Data

Floating-point data is stored as 2-word (32-bit) data in a format defined in IEEE754. The version-2 CVM1 CPUs provides 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 CV-series PC instruction sets and provides the ladder diagram symbols, data areas, and flags used with each. The instructions provided by the CV-series 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

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 version of that instruction. Differentiated and immediate refresh instructions are explained on page 117.

5-2 Instruction Format

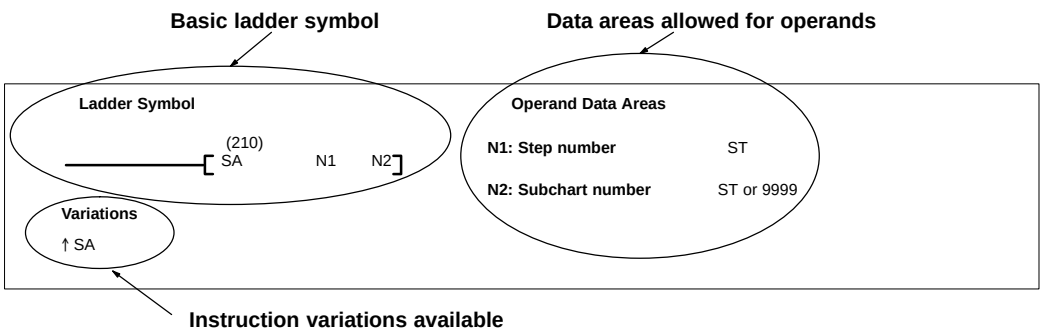
Most instructions have at least one or more operands associated with them. Operands indicate or provide the data on which an instruction is to be performed. These are sometimes input as the actual numeric values (i.e., as constants), but are usually the addresses of data area words or bits that contain 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, 210) is provided above the mnemonic (SA) and the operands are provided to the right (here, N_1 and N_2). 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.

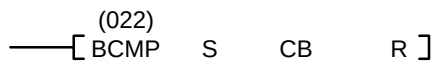
Operand Data Area Precautions

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



Variations

↑ BCMP(022)

Operand Data Areas

S: Source data	CIO, G, A, T, C, #, DM, DR, IR
CB: 1st block word	CIO, G, A, T, C, DM
R: Result word	CIO, G, A, T, C, DM, DR, IR

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

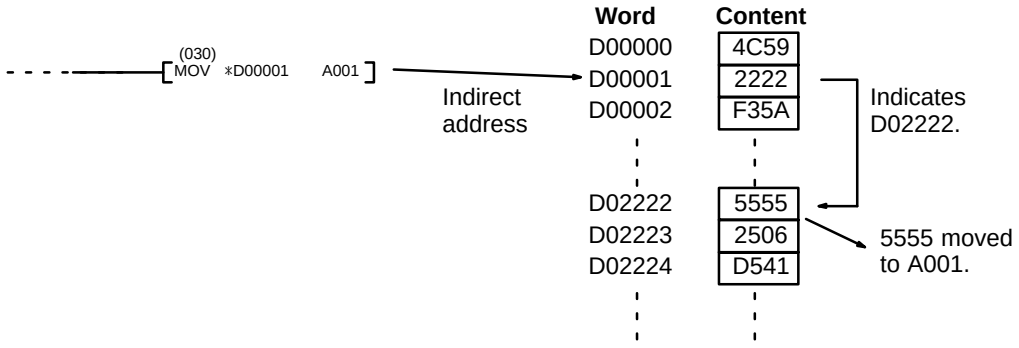
The DM or EM Area can be used to indirectly address an operand. Indirect DM or EM addressing is specified by placing an asterisk before the D or E: ∗D or ∗E. (EM Area is available as an option for the CV1000, CV2000, or CVM1-CPU21-EV2 only.)

The operation of indirect addressing is affected by the PC Setup specified from the CVSS. 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 8191 for the CV500 or CVM1-CPU01-EV2 and between 0000 and 9999 for the CV1000, CV2000, CVM1-CPU11-EV2, or CVM1-CPU21-EV2. Although CV1000, CV2000, and CVM1-CPU21-EV2 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.

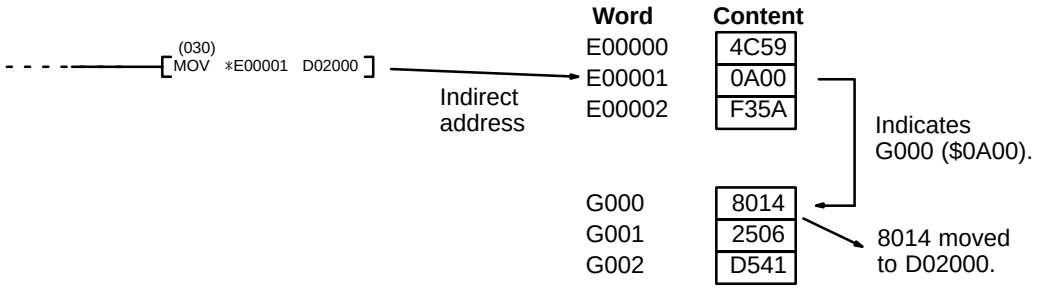
When an indirect DM or EM address is specified in BCD, the DM or EM word specified for the operand will contain the address of the DM or EM word that contains the data that will be used as the operand of the instruction. If, for example, ∗D00001 was designated as the first operand of MOV(030), the contents 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 *3-12 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 step in an SFC program, 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

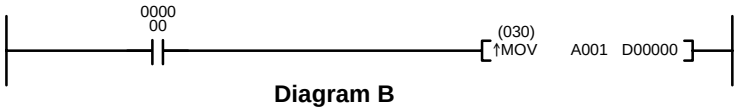


Diagram B

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* and *5-31 Interrupt Control* for details.

CV-series 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 version, 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.

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.

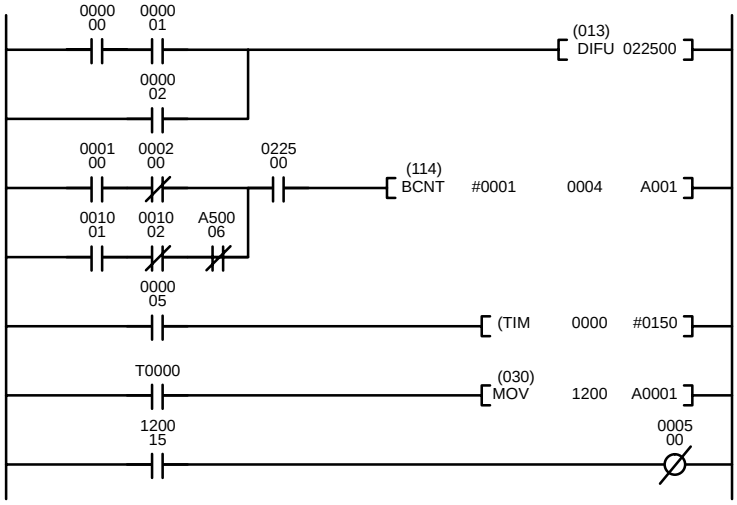
The first word of any instruction defines the instruction and provides any definers. The bit operand is also placed on the same line as the mnemonic for some instructions with certain operands. All other operands are placed on lines after the instruction line, one operand per line and in the same order 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 Peripheral Device. Also be sure to designate differentiated instructions with the ↑ or ↓ symbol.

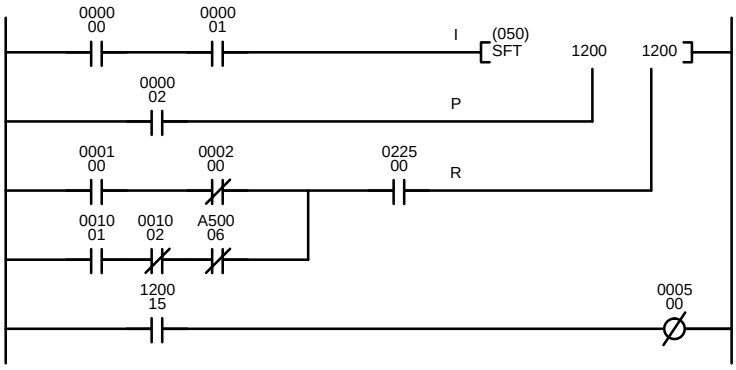
The following diagram and corresponding mnemonic code illustrates the points described above.



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.

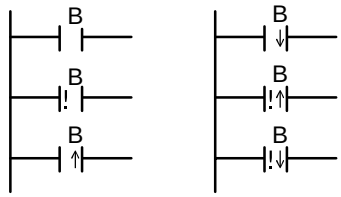
5-6 Ladder Diagram Instructions

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

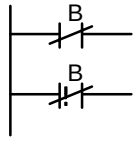
B: Bit CIO, G, A, T, C, ST, TN

Mnemonics

LD	! ↑ LD	! ↓ LD
↑ LD	↓ LD	! LD

LOAD NOT: LD NOT

Ladder Symbols



Operand Data Area

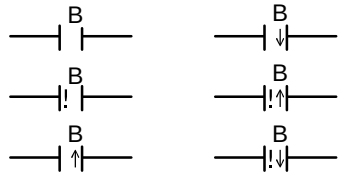
B: Bit CIO, G, A, T, C, ST, TN

Mnemonics

LD NOT	! LD NOT

AND: AND

Ladder Symbols



Operand Data Area

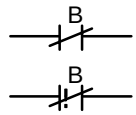
B: Bit CIO, G, A, T, C, ST, TN

Mnemonics

AND	! ↑ AND	! ↓ AND
↑ AND	↓ AND	! AND

AND NOT: AND NOT

Ladder Symbols



Operand Data Area

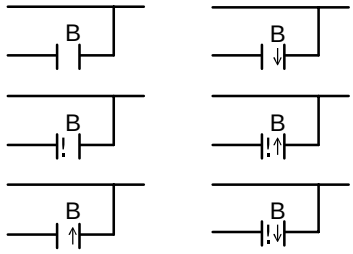
B: Bit CIO, G, A, T, C, ST, TN

Mnemonics

AND NOT	! AND NOT

OR: OR

Ladder Symbols



Operand Data Area

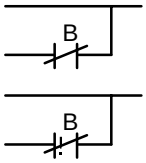
B: Bit CIO, G, A, T, C, ST, TN

Mnemonics

OR	! ↑ OR	! ↓ OR
↑ OR	↓ OR	! OR

OR NOT: OR NOT

Ladder Symbols



Operand Data Area

B: Bit CIO, G, A, T, C, ST, TN

Mnemonics

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 CVSS. 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.

Precautions

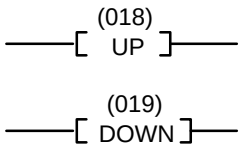
There is no limit to the number of any of these instructions, or restrictions in the order in which they must be used, as long as the program memory capacity of the PC is not exceeded.

Flags

There are no flags affected by these instructions.

5-6-2 CONDITION ON/OFF: UP(018) and DOWN(019) (CVM1 V2)

Ladder Symbols



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* and *5-31 Interrupt Control* for details.

UP(018) and DOWN(019) can only be used with CVM1 version 2 or later CPUs. They cannot be used with version 1 or earlier CPUs; use DIFU(013) and DIFD(014). Refer to *5-7-2 DIFFERENTIATE UP/DOWN: DIFU(013) and DIFD(014)*.

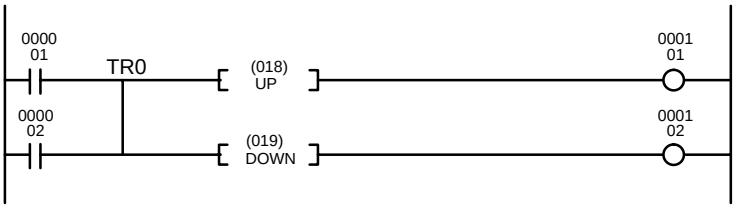
The DIFU(013) and DIFD(014) instructions can also be used for the same purpose, but they require work bits. UP(018) and DOWN(019) simplify programming by reducing the number of work bits and program addresses needed.

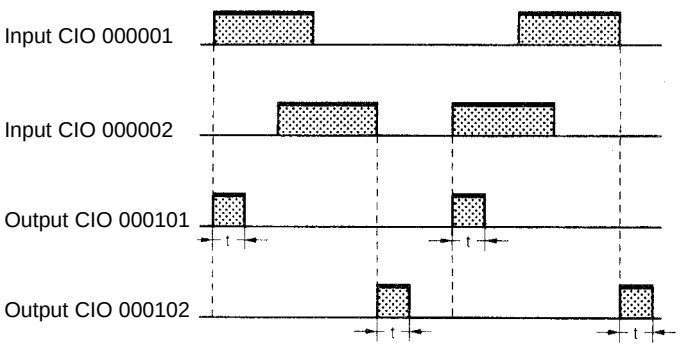
Flags

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.





t: Cycle time

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

5-6-3 BIT TEST: TST(350) and TSTN(351) (CVM1 V2)

Ladder Symbol

(350)
TST S N

(351)
TSTN S N

Operand Data Areas

S: Source word	CIO, G, A, DM, DR, IR
N: Bit number	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 115 for general precautions on operand data areas.

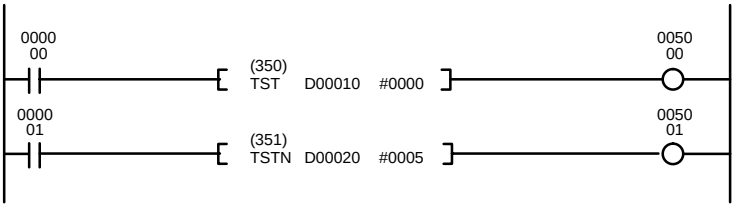
Flags

ER (A50003): N is not 0000 to 0015 BCD.
Content of *DM word is not BCD.

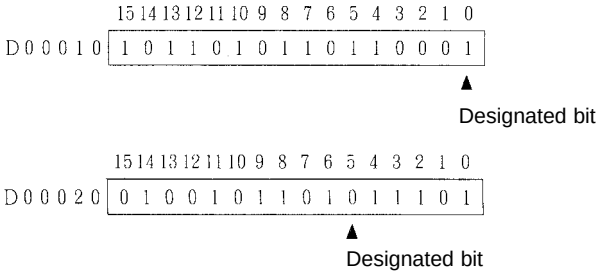
Example

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.

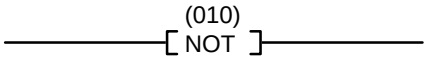


Address	Instruction	Operands
00000	LD	000000
00001	TST(350)	
		D00010
		#0000
00002	OUT	005000
00003	LD	000001
00004	TSTN(351)	
		D00020
		#0005
00005	OUT	005001



5-6-4 NOT: NOT(010)

Ladder Symbol



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

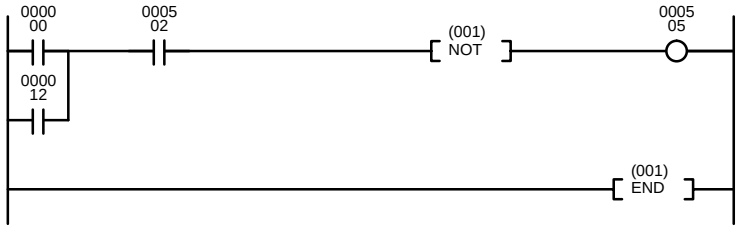
NOT(010) cannot be used as right-hand instructions, i.e., another instruction must appear between them and the right bus bar.

Flags

There are no flags affected by NOT(010).

Example

The following example and bit status table show the operation of NOT(010).



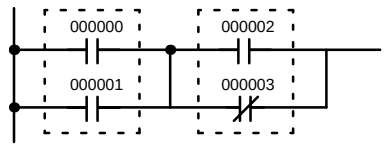
Address	Instruction	Operands
00000	LD	000000
00001	OR	000012
00002	AND	000502
00003	NOT(010)	---
00004	OUT	000505
00005	END(001)	---

Bit	Bit status							
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

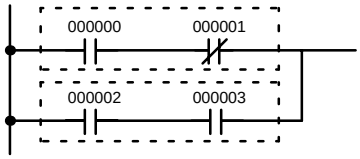
AND LOAD: AND LD

Ladder Symbol



OR LOAD: OR LD

Ladder Symbol



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: Bit**	CIO, G, A, TR
	**Mnemonics**	
	OUT	! OUT

OUTPUT NOT: OUT NOT

Ladder Symbols	Operand Data Area	
	**B: Bit**	CIO, G, A
	**Mnemonics**	
	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.

Precautions

Any output bit is generally used in only one instruction that controls its status.

Note: Refer to page 115 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	
(013) DIFU B	**B: Bit**	CIO, G, A

Variations

!DIFU(013)

DIFFERENTIATE DOWN: DIFD(014)

Ladder Symbol	Operand Data Area	
(014) DIFD B	**B: Bit**	CIO, G, A

Variations

!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.

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* and *5-31 Interrupt Control* for details.

Note: Refer to page 115 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.

Diagram A

Address	Instruction	Operands
00000	LD	000000
00001	MOVQ(037)	
		1200
		A001

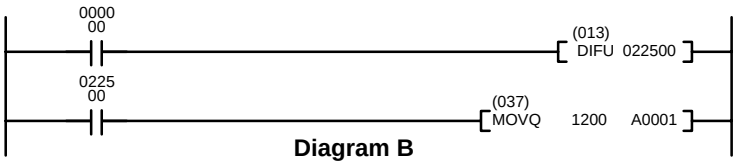


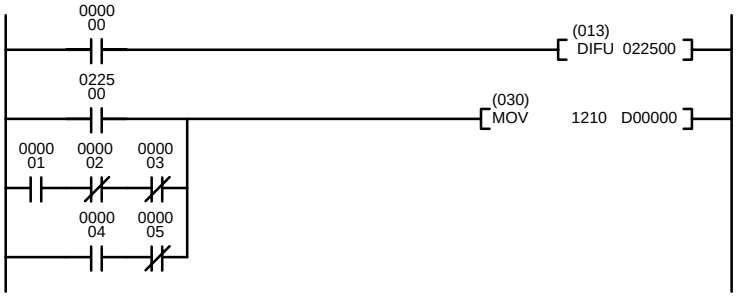
Diagram B

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 123 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

(016)
SET B

Operand Data Area

B: Bit CIO, G, A

Variations

↑SET(016) ↓SET(016) !SET(016)
!↑SET(016) !↓SET(016)

RESET: RSET(017)

Ladder Symbol

(017)
RSET B

Operand Data Area

B: Bit CIO, G, A

Variations

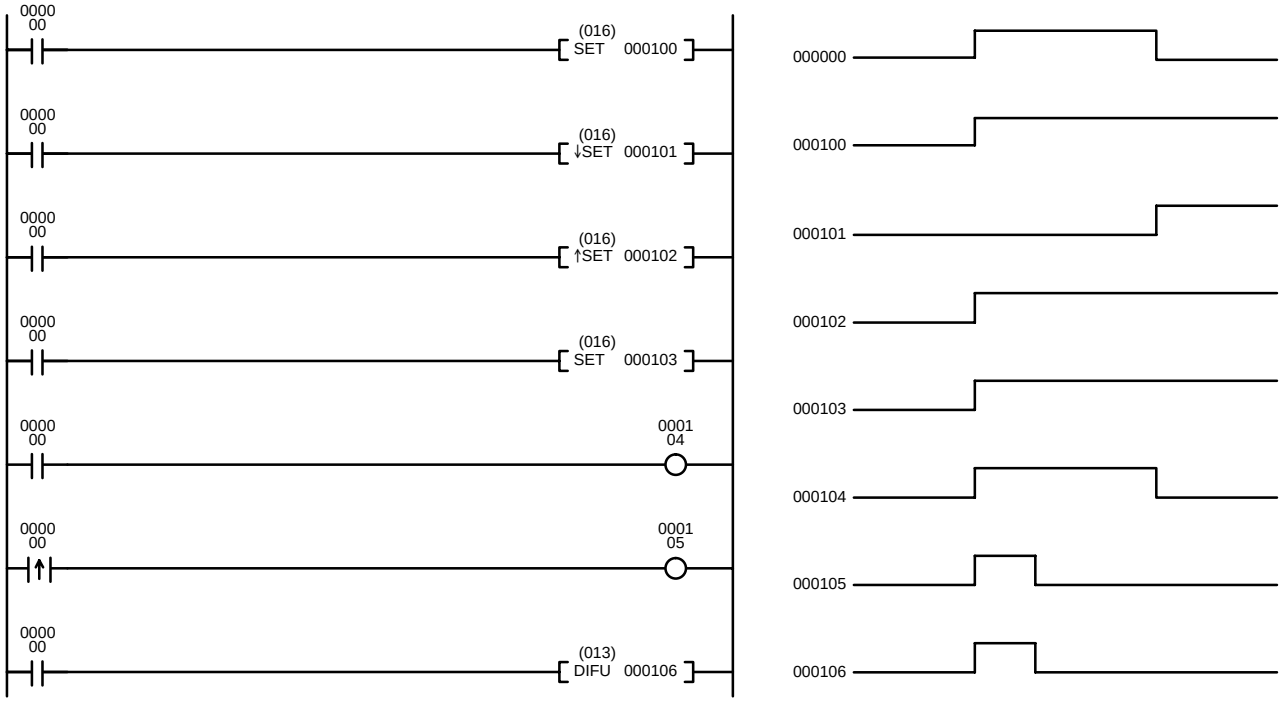
↑RSET(017) ↓RSET(017) !RSET(017)
!↑RSET(017) !↓RSET(017)

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).



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).

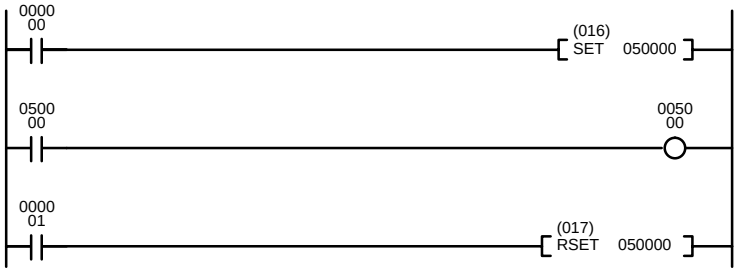
Note: Refer to page 115 for general precautions on operand data areas.

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) (CVM1 V2)

Ladder Symbol

(047)
SETA D N_1 N_2

(048)
RSTA D N_1 N_2

Operand Data Area

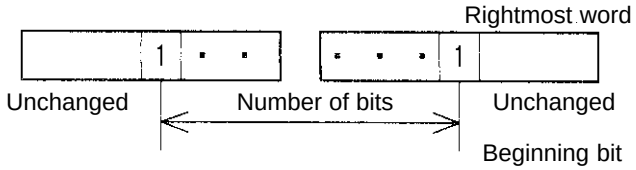
D: Rightmost word for set	CIO, G, A
N_1: Beginning bit	CIO, G, A, T, C, #, DM, DR, IR
N_2: Number of bits	CIO, G, A, T, C, #, DM, DR, IR

Variations

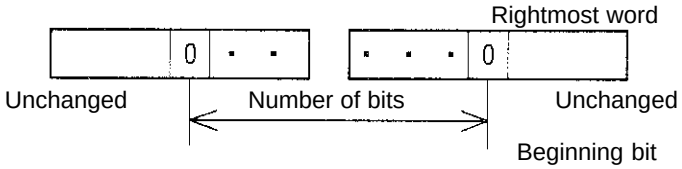
↑SETA(047), ↑RSTA(048)

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_1 must be between 0000 and 0015 and must be BCD. N_2 must be BCD.

Note: Refer to page 115 for general precautions on operand data areas.

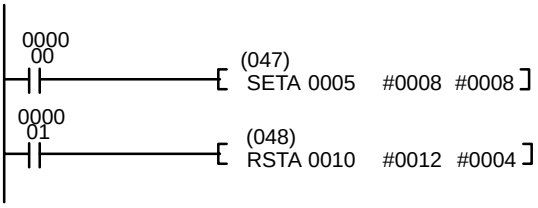
Flags

ER (A50003): N_1 is not 0000 to 0015 BCD.
N_2 is not BCD.
Content of ∗DM word is not BCD.

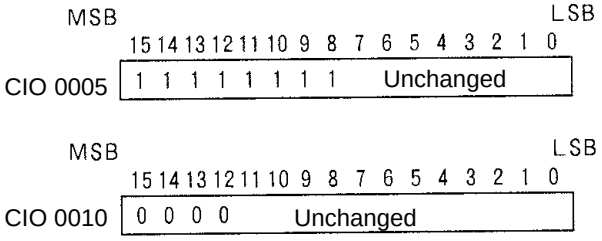
Example 1

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.



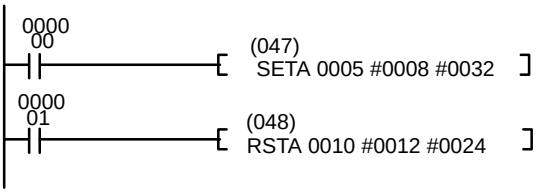
Address	Instruction	Operands
00000	LD	000000
00001	SETA(047)	
		0005
		#0008
		#0008
00002	LD	000001
00003	RSTA(048)	
		0010
		#0012
		#0004



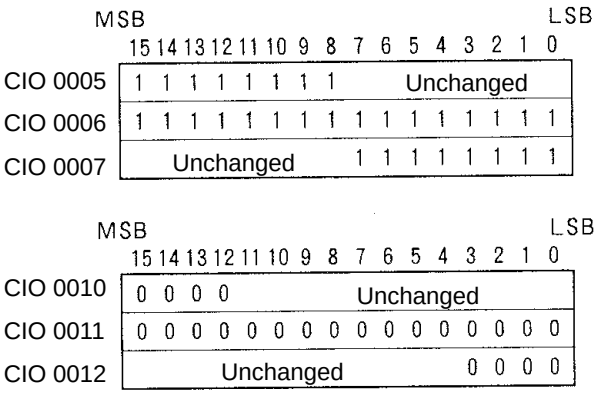
Example 2

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.

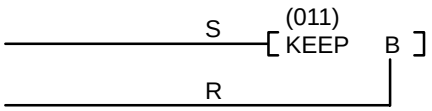


Address	Instruction	Operands
00000	LD	000000
00001	SETA(047)	
		0005
		#0008
		#0032
00002	LD	000001
00003	RSTA(048)	
		0010
		#0012
		#0024



5-7-5 KEEP: KEEP(011)

Ladder Symbol



Operand Data Area

B: Bit	CIO, G, A

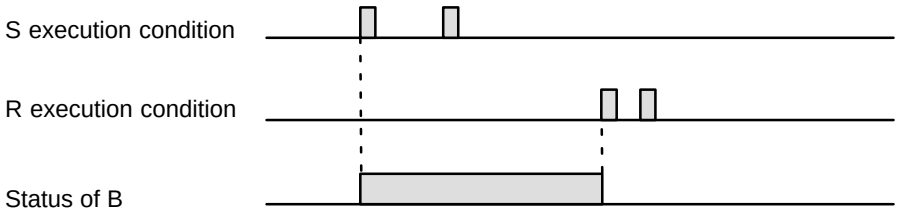
Variations

!KEEP(011)

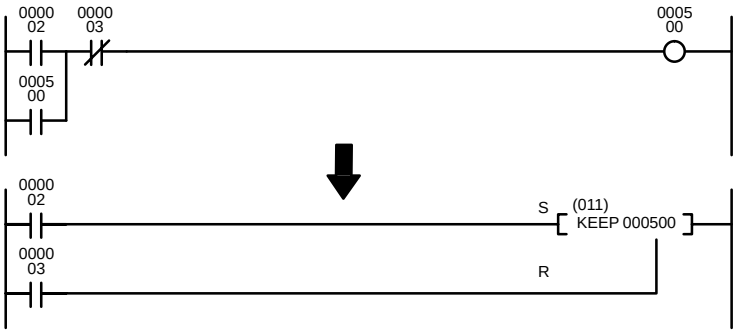
Description

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.

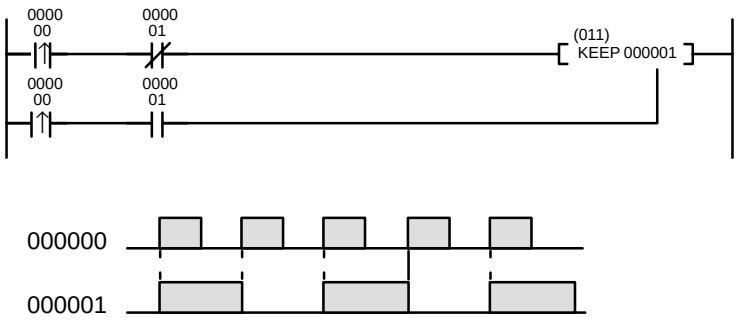


Address	Instruction	Operands
00000	LD	000002
00001	OR	000500
00002	AND NOT	000003
00003	OUT	000500

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.

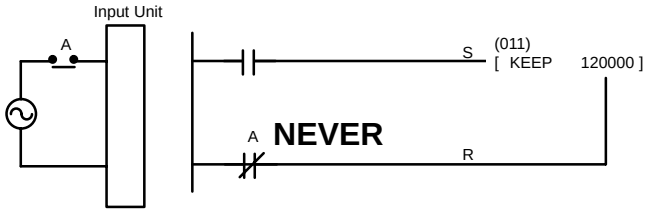


Address	Instruction	Operands
00000	LD↑	000000
00001	AND NOT	000001
00002	LD↑	000000
00003	AND	000001
00004	KEEP(011)	000001

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.

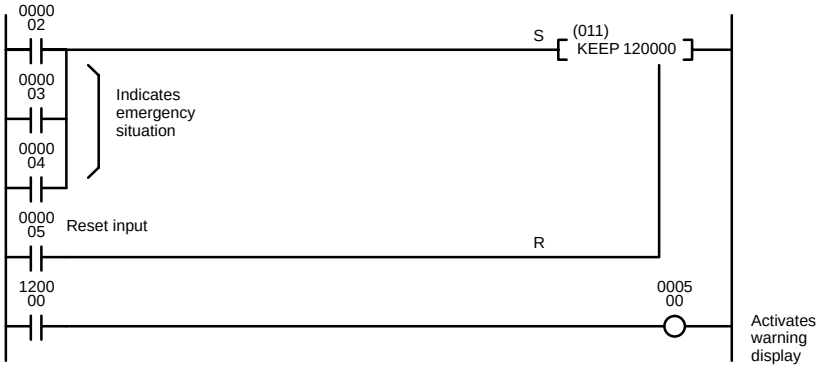
Note: Refer to page 115 for general precautions on operand data areas.

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.



Address	Instruction	Operands
00000	LD	000002
00001	OR	000003
00002	OR	000004
00003	LD	000005
00004	KEEP(011)	120000
00005	LD	120000
00006	OUT	000500

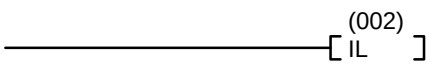
The status of I/O Area bits can be retained in the event of a power interruption by turning ON the IOM Hold Bit and setting IOM Hold Bit Hold in the PC Setup. If the IOM Hold Bit is not specified to be held in the PC Setup, all I/O Area bits will be turned OFF when the power is turned ON. Be sure to 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)

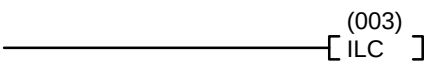
INTERLOCK: IL(002)

Ladder Symbol



INTERLOCK CLEAR: IL(003)

Ladder Symbol



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.

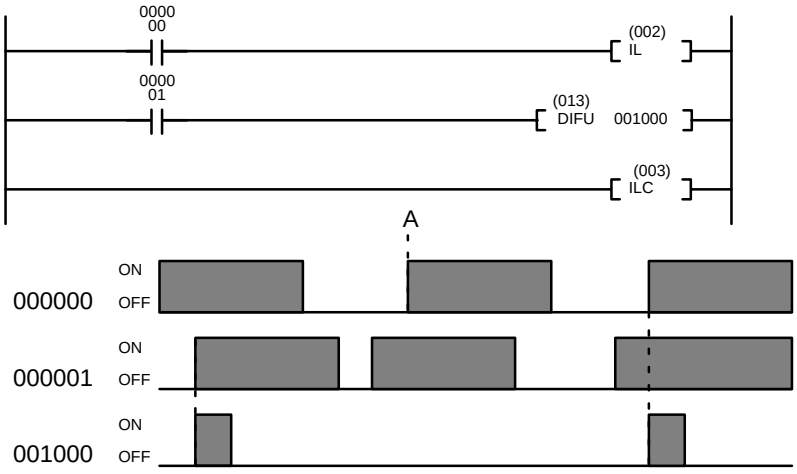
If the execution condition for IL(002) is OFF, the interlocked section between IL(002) and ILC(003) will be treated as shown in the following table:

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.

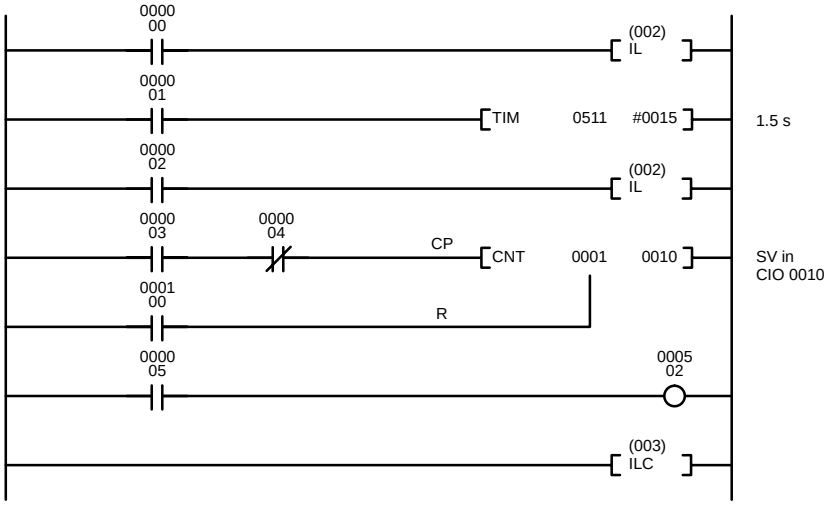
Note: Refer to page 115 for general precautions on operand data areas.

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
(004) JMP N	**N: Jump number** CIO, G, A, T, C, #, DM, DR, IR

JUMP END: JME(005)

Ladder Symbol	Operand Data Area
(005) JME N	**N: Jump number** #

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, except for TIM and TIMH(015), which continue counting. 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 an 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.

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.

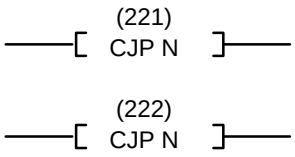
JMP(004) in the program without a corresponding JME(005). Also turns ON the Jump Error Flag A40213.

Example

Examples of jump programs are provided in *4-6 Jumps*.

5-10 CONDITIONAL JUMP: CJP(221)/CJPN(222) (CVM1 V2)

Ladder Symbol



Operand Data Areas

N: Jump number CIO, G, A, T, C, #, DM, DR, IR

Description

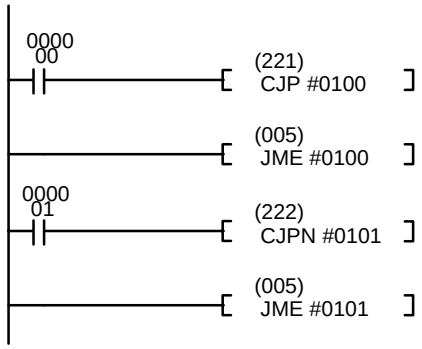
CJP(221) operates in the reverse of JMP(004). When the execution condition turns ON, the program up until JME(005) is skipped. When the execution condition is OFF, the instructions after CJP(221) are executed normally.

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.



When the execution condition (CIO 000000) is ON, the program up to the JME(005) instruction with jump number 0100 is ignored.

When the execution condition (CIO 000001) is OFF, the program up to the JME(005) instruction with jump number 0101 is ignored.

Precautions

The jump number (N) must be BCD between 0000 and 0999.

Flags

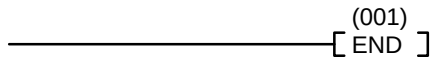
ER (A50003): Content of ∗DM word is not BCD when set for BCD.

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)

Ladder Symbol



Description

END(001) is required as the last instruction in any program, including all action and transition programs. No instruction written after END(001) will be executed.

The END(001) instruction indicates the end of the relevant program for that cycle. For SFC and ladder programming, it indicates the end of the relevant action or transition program. For ladder programming alone, it indicates the end of the entire program.

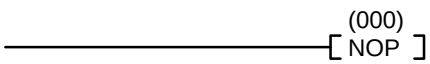
If there is no END(001) in a program, no instructions will be executed and the error message “NO END INST” will appear.

Flags

END(001) turns OFF the ER, CY, GR, EQ, LE, and N Flags.

5-12 NO OPERATION: NOP(000)

Ladder Symbol



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 T0511 in the CV500 or CVM1-CPU01-EV2 and from T0000 through T1023 in the CV1000, CV2000, CVM1-CPU11-EV2 or CVM1-CPU21-EV2, 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.

Present values (PV) and Completion Flags for TIM and TIMH(015) timers are refreshed as shown in the following table.

Instruction	At execution	At END(01)	Interrupts
TIM	PV refreshed and Completion Flag turned ON if PV is 0000.	PV refreshed	PV refreshed every 80 ms if cycle time exceeds 80 ms.
TIMH using T0000 to T0255	Not refreshed	Not refreshed	PV refreshed every 10 ms and Completion Flag turned ON if PV is 0000.
TIMH using T0256 to T1023	PV refreshed and Completion Flag turned ON if PV is 0000.	Not refreshed	Not refreshed

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. TCNT(123), an SFC control instruction, also requires a counter number. Refer to *5-37 SFC Control Instructions*, for details on TCNT(123).

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.

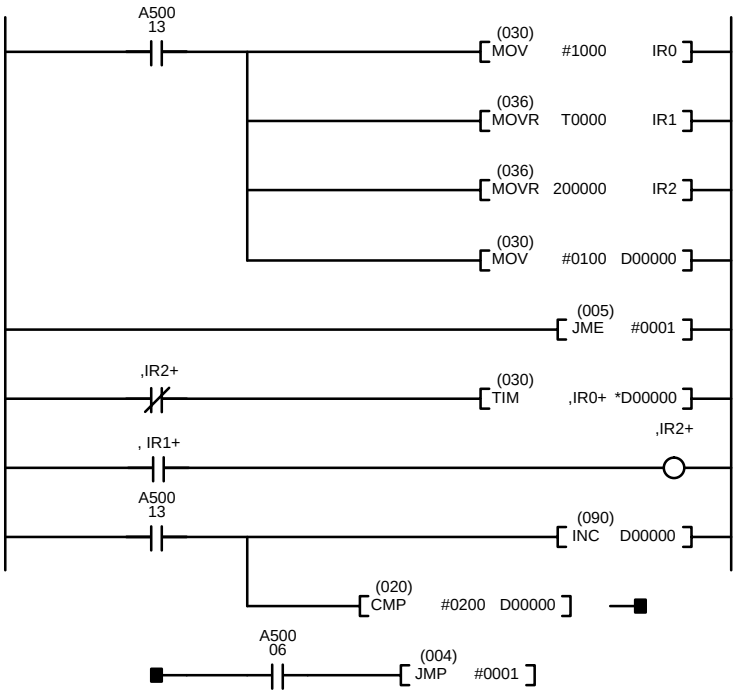
Indirect Addressing

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



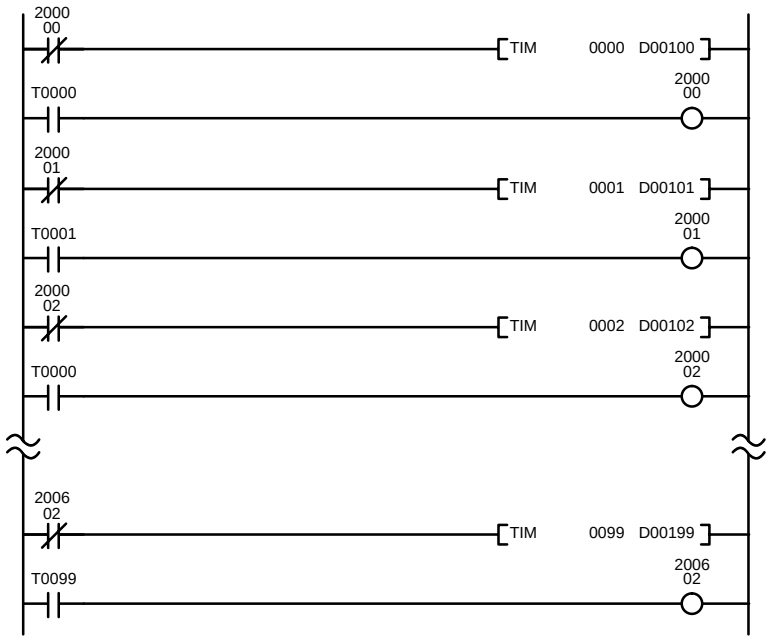
Address	Instruction	Operands
00000	LD	A50013
00001	MOV(030)	
		#1000
		IR0
00002	MOVR(036)	
		T0000
		IR1
00003	MOVR(036)	
		200000
		IR2
00004	MOV(030)	#0100
		D00000
00005	JME(005)	#0001
00006	LD NOT	,IR2+
00007	TIM	,IR0+
		∗D00000
00008	LD	,IR1+
00009	OUT	,IR2+
00010	LD	A50013
00011	INC(090)	
		D00000
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

Ladder Symbol

Operand Data Areas

N: Timer number #

S: Set value CIO, G, A, T, C, #, DM, DR, IR

*Refer to page 141 for details on indirectly addressing timers.

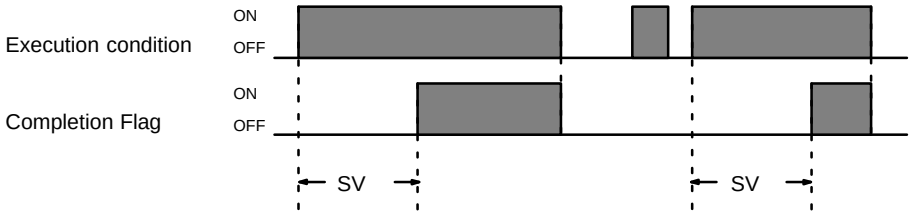
Description

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. TIM accuracy is +0.0/–0.1 second.

The timer PV is updated when the TIM instruction is executed, during cyclic refreshing, or interrupt refreshing (when the cycle time exceeds 80 ms). Refer to *6-2 Cycle Time* for details.

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.



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.

Timer numbers are as shown in the following table. The “high-speed” timer numbers should not be used for other timer instructions if they are required for TIMH(015). Refer to *5-13-2 HIGH-SPEED TIMER: TIMH(015)* for details.

PC	Instructions	Timer numbers
CV500 or CVM1-CPU01-EV2	All timers	T0000 through T0511
	High-speed timers	T0000 through T0127
CV1000, CV2000, CVM1-CPU11-EV2, CVM1-CPU21-EV2	All timers	T0000 through T1023
	High-speed timers	T0000 through T0255

When using SFC, TIM and TIMH(015) timers are reset at the transition between steps. If required, use the hold option with the action qualifier to prevent timers from being automatically resetting at transitions.

Timers in interlocked program sections are reset when the execution condition for IL(002) is OFF. Timers in jumped program sections continue timing. Power interruptions also reset timers. If a timer that is not reset under these conditions is desired, Auxiliary Area clock pulse bits can be counted to produce timers using CNT. Refer to *5-13-6 COUNTER: CNT* for details.

If the same timer number is used in different SFC program steps, or 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.

A delay of one cycle is sometimes required for a Completion Flag to be turned ON after the timer times out.

If you convert a TIM instruction to a TIMH(015) instruction using online programming operations, always reset the TIM instruction’s Completion Flag. Proper operation will not be possible unless the Completion Flag is reset.

Note: Refer to page 115 for general precautions on operand data areas.

Flags

ER (A50003): Content of ∗DM word is not BCD when set for BCD.

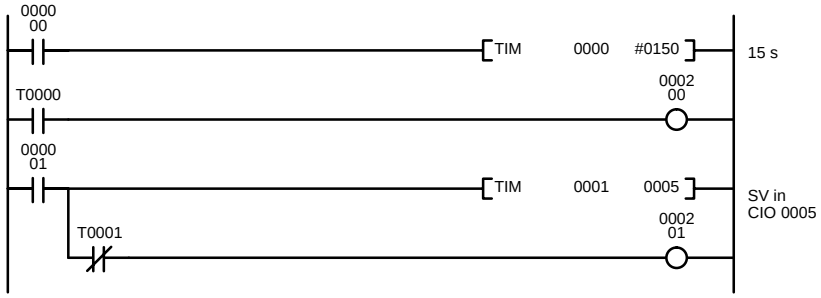
Content of S (SV) is not BCD.

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

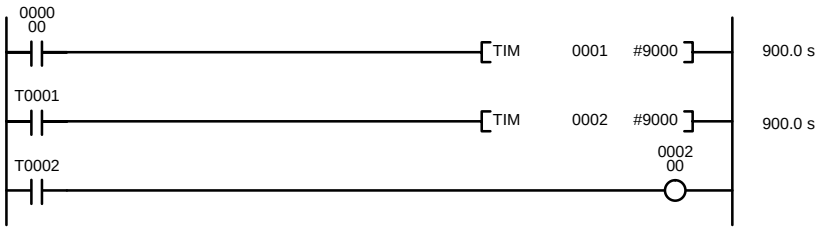
The following example shows two timers, one set with a constant and one set via input word 0005. Here, 000200 will be turned ON after 000000 goes ON and stays ON for at least 15 seconds. When 000000 goes OFF, the timer will be reset and 000200 will be turned OFF. When 000001 goes ON, T0001 is started 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.

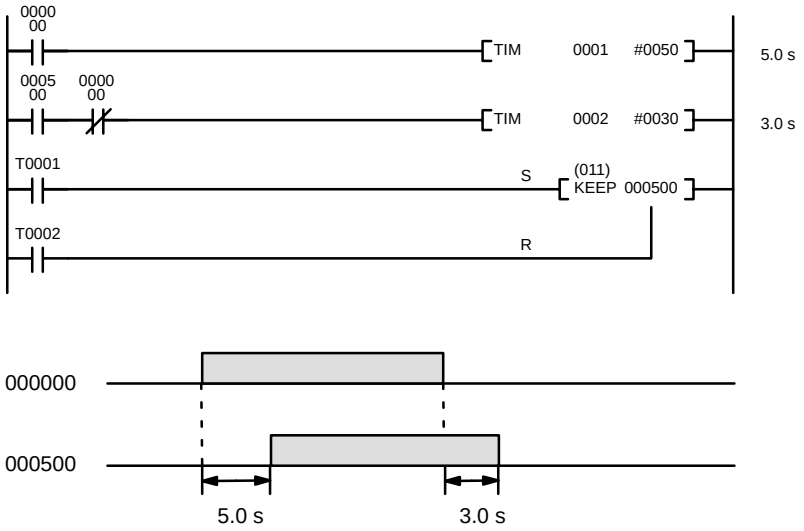
The third way to create longer timers is to combined TIM with CNT or to used CNT to count Auxiliary Area clock pulse bits to produce longer timers. An example is provided in *5-13-6 COUNTER: CNT*.

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.

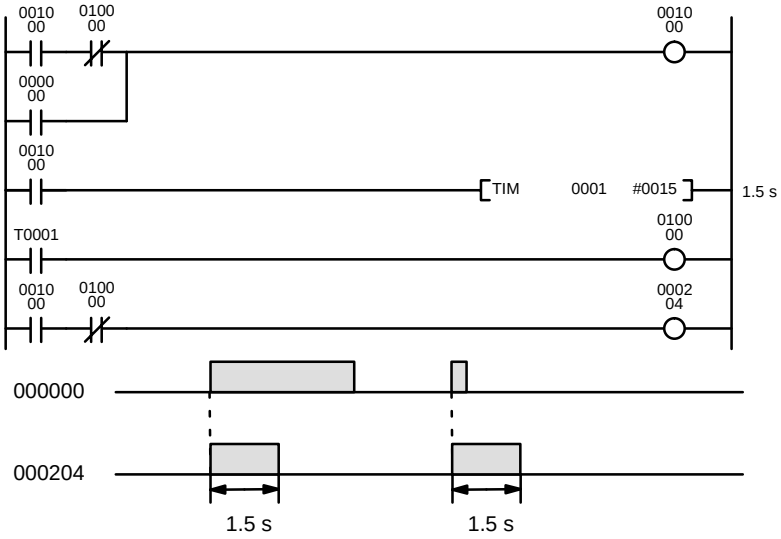
In the following example, 000500 would be turned ON 5.0 seconds after 000000 goes ON and then turned OFF 3.0 seconds after 000000 goes OFF. It is necessary to use both 000500 and 000000 to determine the execution condition for T0002; 000000 in a normally closed condition is necessary to reset T0002 when 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

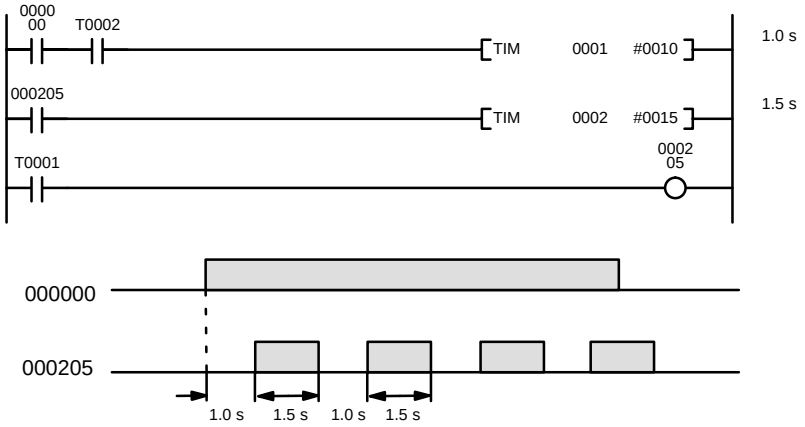
The length of time that a bit is kept ON or OFF can be controlled by combining TIM with OUT or OUT NOT. The following diagram demonstrates how this is possible. In this example, 000204 would remain ON for 1.5 seconds after 000000 goes ON regardless of the time 000000 stays ON. This is achieved by using 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

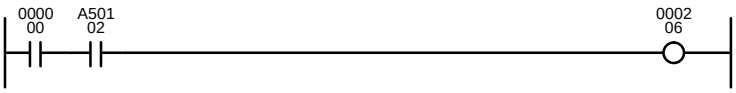
Bits can be programmed to turn ON and OFF at regular intervals while a designated execution condition is ON by using TIM twice. One TIM functions to turn ON and OFF a specified bit, i.e., the Completion Flag of this TIM turns the specified bit ON and OFF. The other TIM functions to control the operation 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.

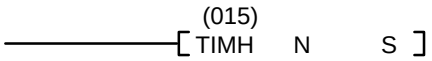
In the following example the 1-second clock pulse is used (A50102) so that 000206 would be turned ON and OFF every second, i.e., it would be ON for 0.5 seconds and OFF for 0.5 seconds. Precise timing and the initial status of 000206 would depend on the status of the clock pulse when 000000 goes ON.



Address	Instruction	Operands
00000	LD	000000
00001	AND	A50102
00002	OUT	000206

5-13-2 HIGH-SPEED TIMER: TIMH(015)

Ladder Symbol



Operand Data Areas

N: Timer number #

S: Set value CIO, G, A, T, C, #, DM, DR, IR

*Refer to page 141 for details on indirectly addressing timers.

Description

TIMH(015) operates in the same way as TIM except that TIMH measures in units of 0.01 second.

The cycle time affects TIMH(015) accuracy if T0128 through T0511 in the CV500 or CVM1-CPU01-EV2 or T0256 through T1023 in the CV1000, CV2000, or CVM1-CPU11/21-EV2 are used. If the cycle time is longer than 10 ms, use timer numbers T0000 through T0127 in the CV500 or CVM1-CPU01-EV2 or T0000 through T0255 in the CV1000, CV2000, or CVM1-CPU11/21-EV2.

High-speed timer PVs are updated when the TIMH instruction is executed, during cyclic refreshing, or interrupt refreshing. Interrupt refreshing updates active high-speed timers every 10 ms. Refer to *6-2 Cycle Time* for details.

Refer to *5-13-1 TIMER: TIM* for operational details and examples. Except for the items mentioned above, all aspects of operation are the same.

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.

Timer numbers are as shown in the following table. The “high-speed” timer numbers must be used for TIMH(015) to ensure accuracy if the cycle time is longer than 10 ms. The present value for high-speed timers created with these timer numbers are refreshed every 10 ms. The present value for high-speed timers created with other timer number are refreshed only when the instruction is executed.

PC	Instructions	Timer numbers
CV500 or CVM1-CPU01-EV2	All timers	T0000 through T0511
	High-speed timers	T0000 through T0127
CV1000, CV2000, or CVM1-CPU11/21-EV2	All timers	T0000 through T1023
	High-speed timers	T0000 through T0255

When using SFC, TIM and TIMH(015) timers are reset at the transition between steps. If required, use the hold option with the action qualifier to prevent timers from being automatically reset at transitions.

Timers in interlocked program sections are reset when the execution condition for IL(002) is OFF. Timers in jumped program sections continue timing. Power interruptions also reset timers. If a timer that is not reset under these conditions is desired, Auxiliary Area clock pulse bits can be counted to produce timers using CNT. Refer to *5-13-6 COUNTER: CNT* for details.

If the same timer number is used in different SFC program steps, or 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.

A delay of one cycle is sometimes required for a Completion Flag to be turned ON after the timer times out.

If you convert a TIM instruction to a TIMH(015) instruction using online programming operations, always reset the TIM instruction’s Completion Flag. Proper operation will not be possible unless the Completion Flag is reset.

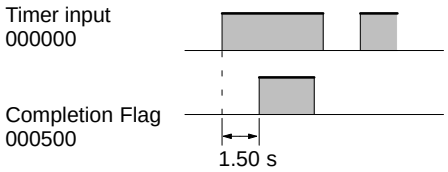
Note: Refer to page 115 for general precautions on operand data areas.

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.

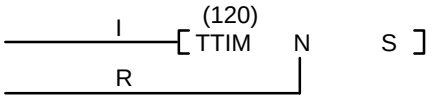


When CIO 000001 is ON, the SV for the second TIMH(015) in the following example will be read from CIO 0020, allowing the SV of the timer to be control from an external device connected through CIO 0020.

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)

Ladder Symbol



Operand Data Areas

N: Timer number #

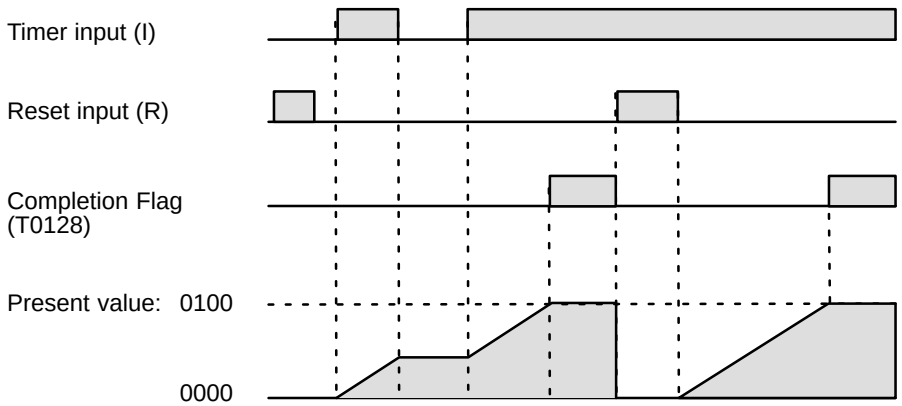
S: Set value CIO, G, A, T, C, #, DM, DR, IR

*Refer to page 141 for details on indirectly addressing timers.

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. TTIM(120) accuracy is +0.0/–0.1 second.



Precautions

SV must be between 000.0 and 999.9 and must be BCD. The decimal point is not entered.

Timer numbers are as shown in the following table. The “high-speed” timer numbers should not be used for other timer instructions if they are required for TIMH(015). Refer to *5-13-2 HIGH-SPEED TIMER: TIMH(015)* for details.

PC	Instructions	Timer numbers
CV500 or CVM1-CPU01-EV2	All timers	T0000 through T0511
	High-speed timers	T0000 through T0127
CV1000, CV2000, or CVM1-CPU11/21-EV2	All timers	T0000 through T1023
	High-speed timers	T0000 through T0255

The PVs of accumulative timers in interlocked program sections are maintained when the execution condition for IL(002) is OFF. Unlike timers and high-speed timers, accumulative timers in jumped program sections do not continue counting, but maintain the PV.

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.

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.

A delay of one cycle is sometimes required for a Completion Flag to be turned ON after the timer times out.

The PV and Completion Flag for TTIM(120) are refreshed when the instruction is executed. TTIM(120) will thus not execute correctly if the cycle time is 100 ms or larger.

Note: Refer to page 115 for general precautions on operand data areas.

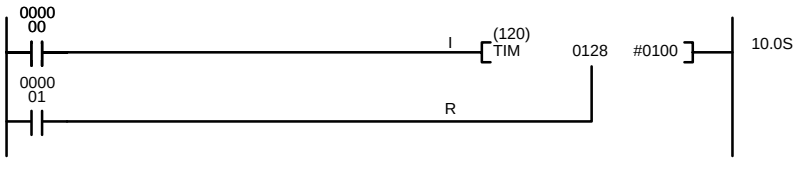
Flags

ER (A50003): Content of ∗DM word is not BCD when set for BCD.

SV is not BCD.

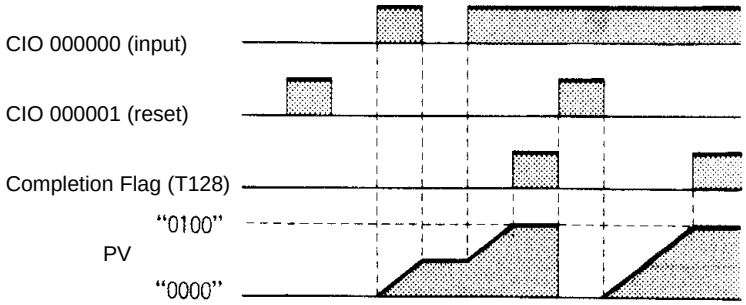
Example

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

(121)
TIML D_1 D_2 S

Operand Data Areas

D_1: Completion Flag	CIO, G, A, DM
D_2: PV word	CIO, G, A, DM
S: SV word	CIO, G, A, T, C, #, DM

Description

TIML(121) is a decrementing 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. TIML(121) accuracy is +0.0/–0.1 second.

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_1, the PV is maintained in D_2 and D_2+1, and the SV is specified in S and S+1. Bits 01 through 15 of D_1 cannot be used.

Precautions

D_2 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_1, D_2, and D_2+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. Unlike TIM timers and high-speed timers, long timers in jumped program sections do not continue counting, but maintain the PV.

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.

A delay of one cycle is sometimes required for a Completion Flag to be turned ON after the timer times out.

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 115 for general precautions on operand data areas.

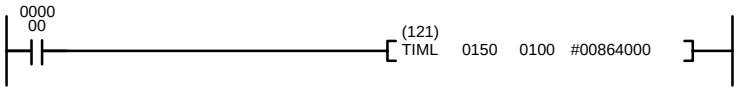
Flags

ER (A50003): Content of ∗DM word is not BCD when set for BCD.
SV is not BCD.

Example

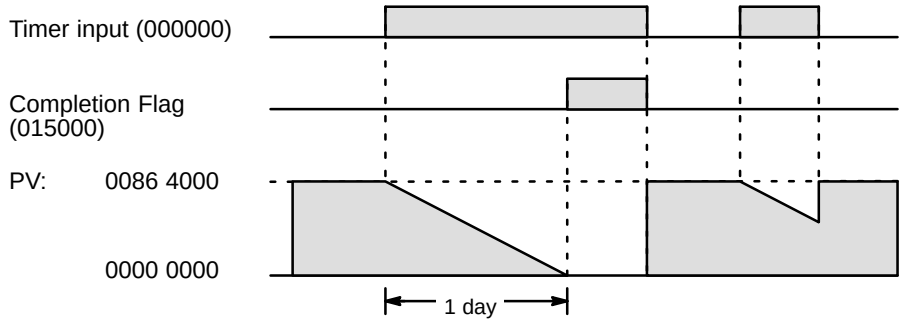
When CIO 00000 turns ON in the following example, the SV (0086 4000) will be set into CIO 0101 and CIO 0100. As long as CIO 000000 remains ON, the PV in

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.



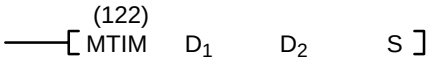
Address	Instruction	Operands
00000	LD	000000
00001	TIML(121)	0150
		0100
		#00864000

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

D_1: Completion Flags	CIO, G, A, T, C, DM
D_2: PV word	CIO, G, A, T, C, DM, DR, IR
S: First SV word	CIO, G, A, T, C, #, DM

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_1) goes from ON to OFF.

Once activated, MTIM(122) increments the content of D_2 (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. MTIM(122) accuracy is +0.0/–0.1 second.

Each time the instruction is executed, the PV (content of D_2) 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_1 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.

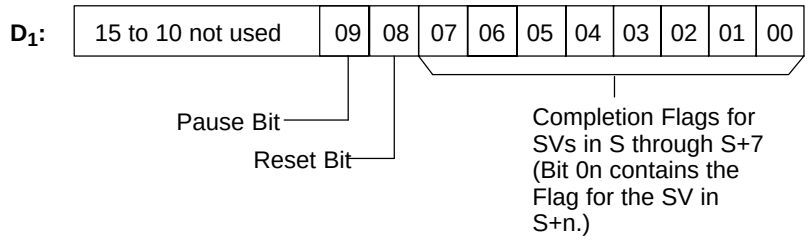
The timer can be reset by turning ON the reset bit (bit 08 of D_1), which resets the PV and all Completion Flags to zero. Counting resumes from zero when the reset bit is turned OFF if the instruction execution condition is ON.

The pause bit (bit 09 of D_1) 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_1 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_1 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_1.



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 115 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

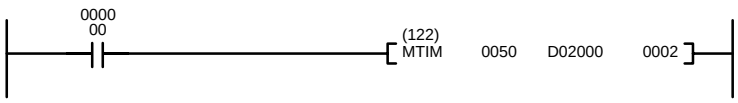
Comparisons will be made between the value in D02000 and the SV in CIO 0002 through CIO 0009 (S to S+7) and the corresponding bits in CIO 0050 will be turned on whenever the PV is greater than or equal to an SV, for example, CIO 005000 will be turned ON when the PV reaches 0155 (15.5 s).

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.

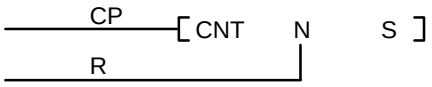


Address	Instruction	Operands
00000	LD	000000
00001	MTIM(122)	
		0050
		D02000
		0002

	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

Ladder Symbol



Operand Data Areas

N: Counter number #

S: Set value CIO, G, A, T, C, #, DM, DR, IR

*Refer to page 141 for details on indirectly addressing counters.

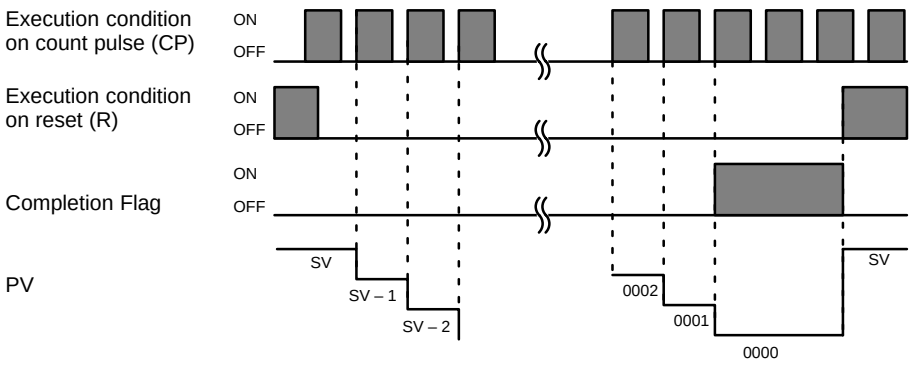
Description

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 are as shown in the following table. Each counter number can be used to define only one counter instruction unless the counters are never active simultaneously.

PC	Counter numbers
CV500 or CVM1-CPU01-EV2	C0000 through C0511
CV1000, CV2000, or CVM1-CPU11/21-EV2	C0000 through C1023

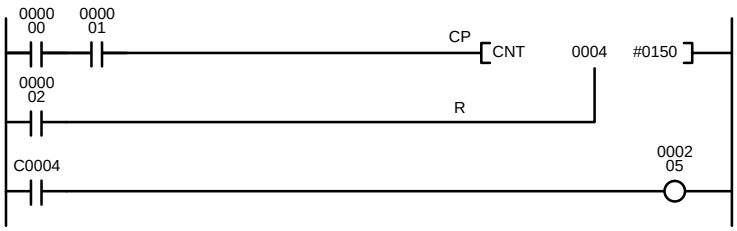
Note: Refer to page 115 for general precautions on operand data areas.

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

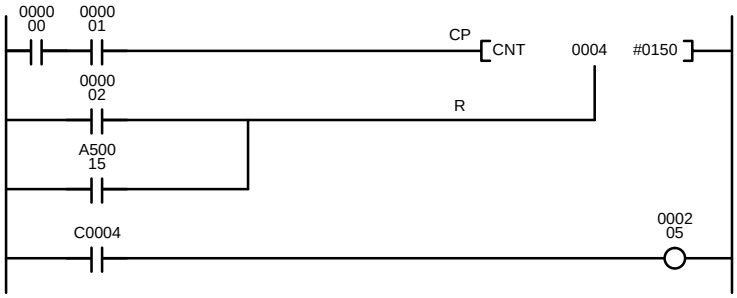
In the following example, the PV will be decremented whenever both 000000 and 000001 are ON provided that 000002 is OFF and either 000000 or 000001 was OFF the last time C0004 was executed. When 150 pulses have been counted down (i.e., when PV reaches zero), 000205 will be turned ON.



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

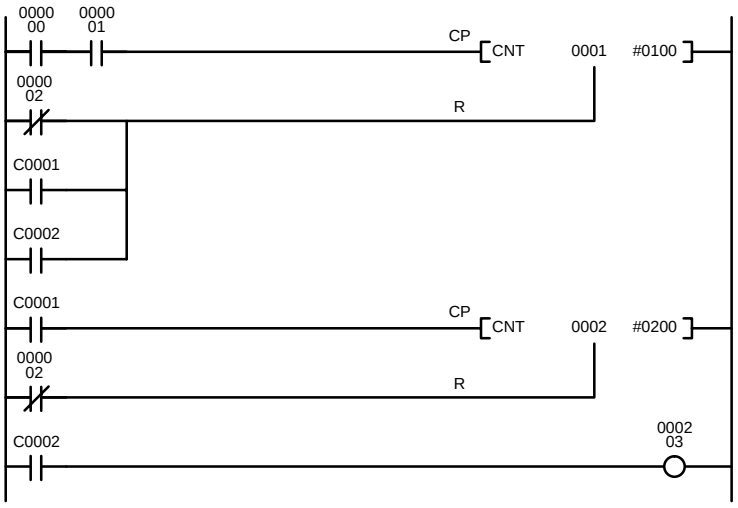
Example 2: Extended Counter

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.



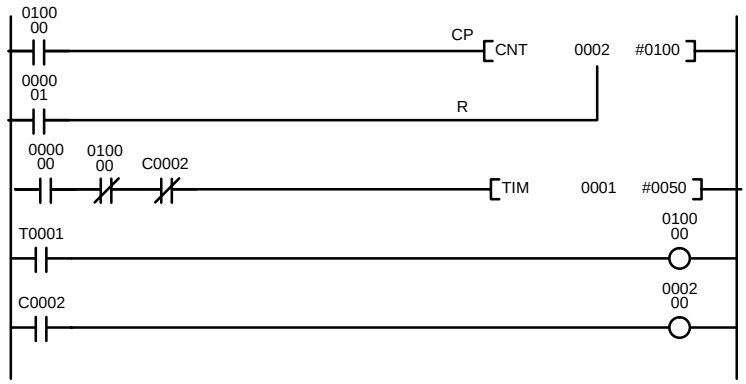
Address	Instruction	Operands
00000	LD	000000
00001	AND	000001
00002	LD NOT	000002
00003	OR	C0001
00004	OR	C0002
00005	CNT	0001
		#0100
00006	LD	C0001
00007	LD NOT	000002
00008	CNT	0002
		#0200
00009	LD	C0002
00010	OUT	000203

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.

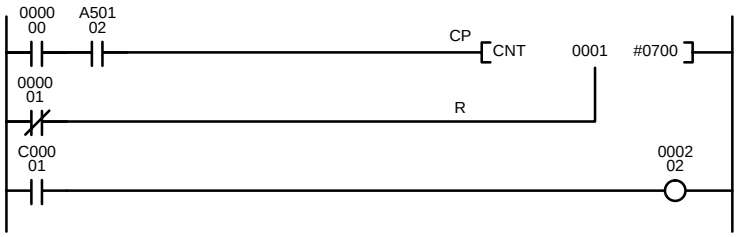
Because in this example the SV for T0001 is 5.0 seconds and the SV for C0002 is 100, the Completion Flag for C0002 turns ON when 5 seconds x 100 times, i.e., 500 seconds (or 8 minutes and 20 seconds), have expired. This would result in 000201 being turned ON.



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.

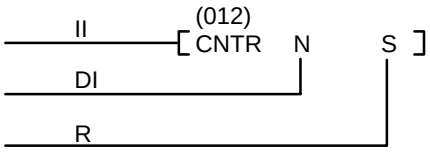


Address	Instruction	Operands
00000	LD	000000
00001	AND	A50102
00002	LD NOT	000001
00003	CNT	0001
		#0700
00004	LD	C00001
00005	OUT	000202

⚠ 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)

Ladder Symbol



Operand Data Areas

N: Counter number #

S: Set value CIO, G, A, T, C, #, DM, DR, IR

*Refer to page 141 for details on indirectly addressing counters.

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.

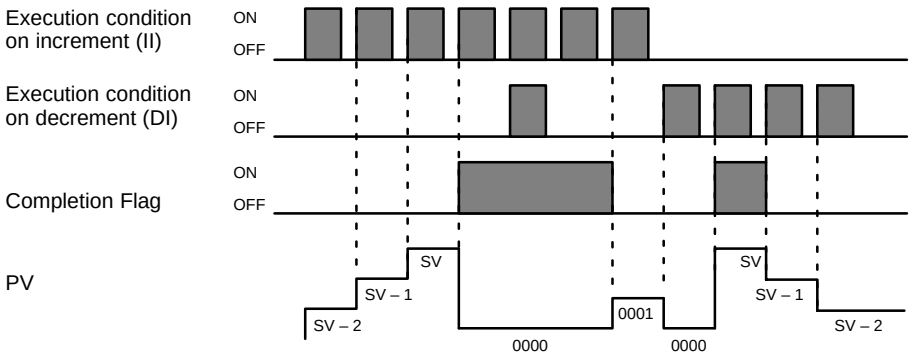
If the execution conditions have not changed or have changed from ON to OFF for both II and DI, the PV of CNTR(012) will not be changed.

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 are as shown in the following table. Each counter number can be used to define only one counter instruction unless the counters are never active simultaneously.

PC	Counter numbers
CV500 or CVM1-CPU01-EV2	C0000 through C0511
CV1000, CV2000, or CVM1-CPU11/21-EV2	C0000 through C1023

Note: Refer to page 115 for general precautions on operand data areas.

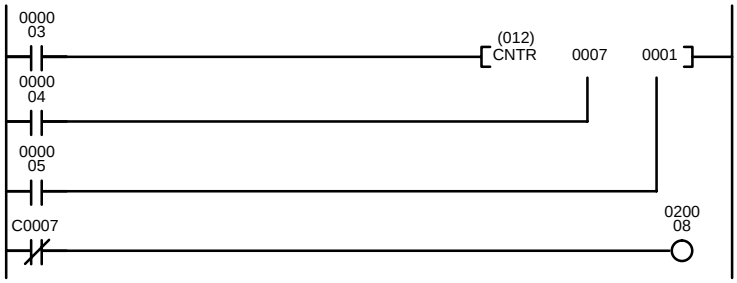
Flags

ER (A50003): Content of ∗DM word is not BCD when set for BCD.

Content of S (SV) is not BCD.

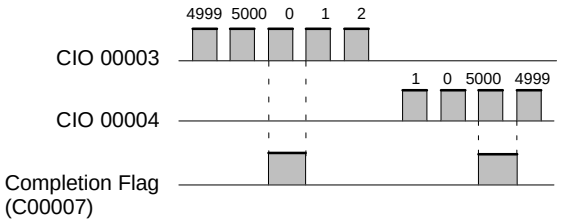
Example

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.



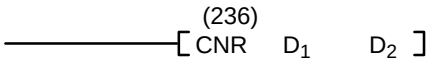
Address	Instruction	Operands
00000	LD	000003
00001	LD	000004
00002	LD	000005
00003	CNTR(012)	0007
		0001
00004	LD NOT	C0007
00005	OUT	020008

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)

Ladder Symbol



Variations

↑CNR(236)

Operand Data Areas

D_1: First Word	CIO, G, A, T, C, DM
D_2: Second Word	CIO, G, A, T, C, DM

Description

When D_1 and D_2 are timer or counter numbers, CNR(236) resets the PV of timers from D_1 through D_2 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_1 and D_2.

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_1 and D_2 are addresses in a data area, CNR(236) sets the content of words D_1 through D_2 to zero.

Precautions

D_1 and D_2 must be in the same data area, and D_1 must be less than or equal to D_2. If $D_1 \square D_2$, only D_1 will be reset.

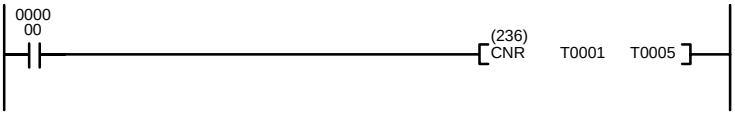
Note: Refer to page 115 for general precautions on operand data areas.

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.



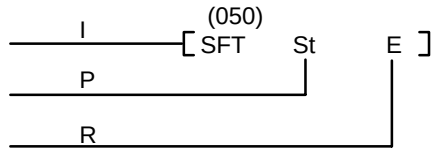
Address	Instruction	Operands
00000	LD	000000
00001	CNR(236)	
		T0001
		T0005

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

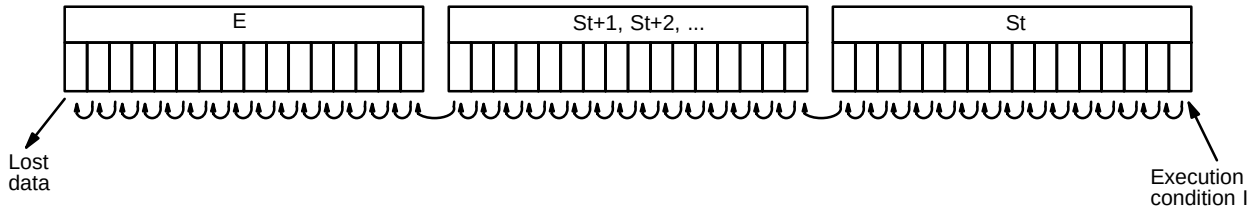


Operand Data Areas

St: Starting word	CIO, G, A
E: End word	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 Peripheral Device. The program, however, will be executed as written. See *Example 2: Controlling Bits in Shift Registers* for a programming example that does this.

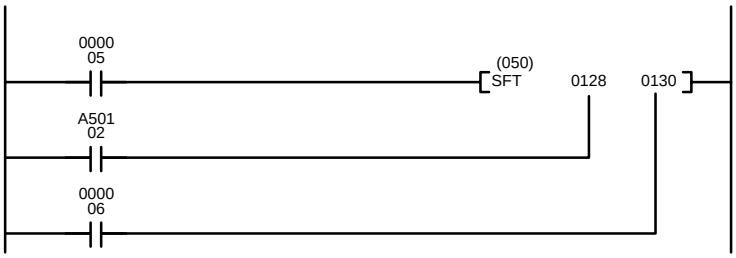
Note Refer to page 115 for general precautions on operand data areas.

Flags

There are no flags affected by SFT(050).

Example 1: Basic Application

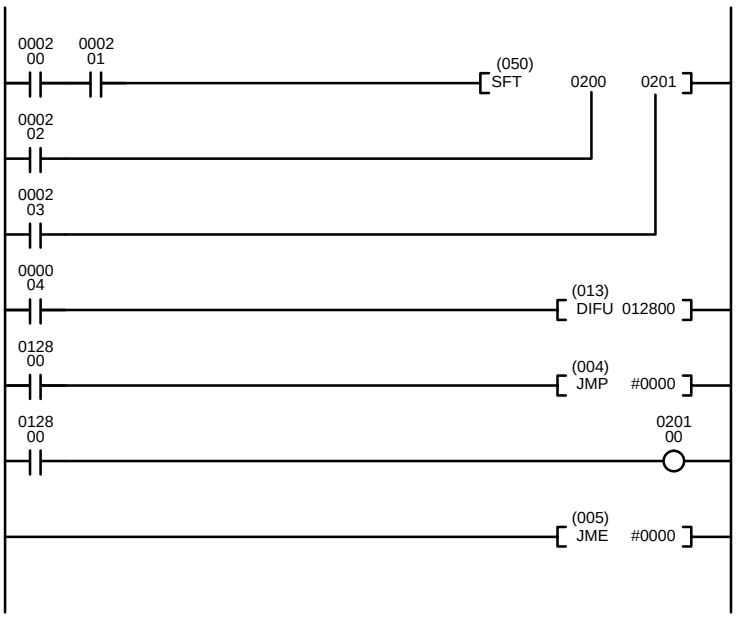
The following example uses the 1-second clock pulse bit (A50102) so that the execution condition produced by CIO 000005 is shifted into a 3-word register between CIO 0128 and CIO 0130 every second.



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.



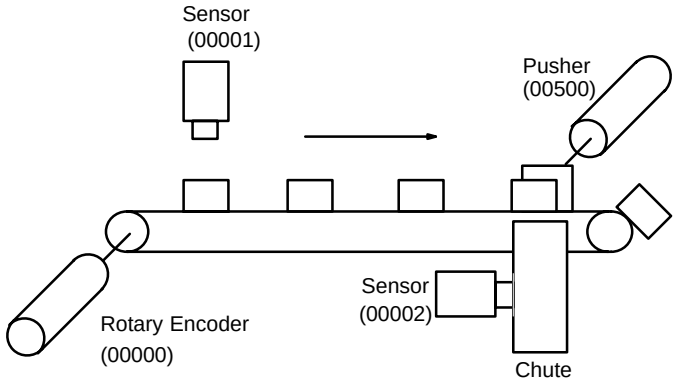
Address	Instruction	Operands
00000	LD	000200
00001	AND	000201
00002	LD	000202
00003	LD	000203
00004	SFT(050)	
		0200
		0201
00005	LD	000004
00006	DIFU(013)	012800
00007	LD	012800
00008	JMP(004)	#0000
00009	LD	012800
00010	OUT	020100
00011	JME(005)	#0000

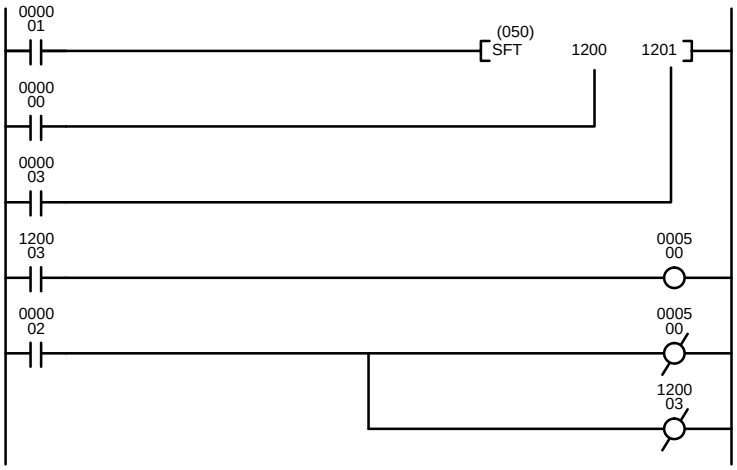
When a bit that is part of a shift register is used in OUT (or any other instruction that controls bit status), a syntax error will be generated during the program check, but the program will executed properly (i.e., as written).

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

(051)
SFTR C St E

Variations

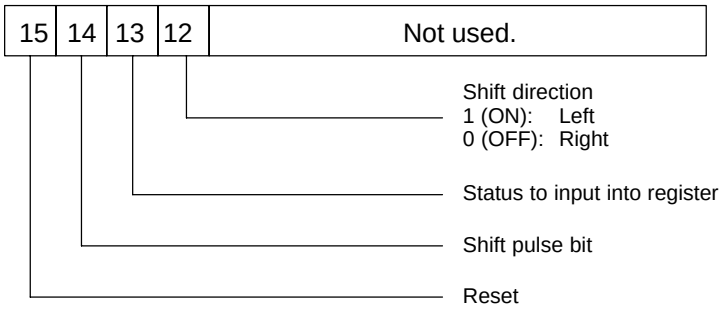
↑ SFTR(051)

Operand Data Areas

C: Control word	CIO, G, A, DM, DR, IR
St: Starting word	CIO, G, A, DM
E: End word	CIO, G, A, DM

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 115 for general precautions on operand data areas.

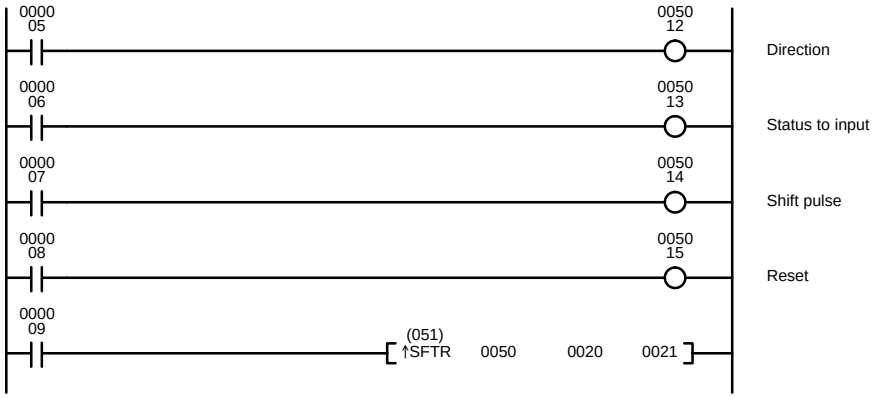
Flags

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 ↑ SFTR(051). The shift register is between words 0020 and 0021, and it is controlled through bit 000009.



Address	Instruction	Operands
00000	LD	000005
00001	OUT	005012
00002	LD	000006
00003	OUT	005013
00004	LD	000007
00005	OUT	005014
00006	LD	000008
00007	OUT	005015
00008	LD	000009
00009	↑SFTR(051)	
		0050
		0020
		0021

5-14-3 ASYNCHRONOUS SHIFT REGISTER: ASFT(052)

Ladder Symbol

(052)
ASFT C St E

Variations

↑ ASFT(052)

Operand Data Areas

C: Control word	CIO, G, A, DM, DR, IR
St: Starting word	CIO, G, A, DM
E: End word	CIO, G, A, DM

Description

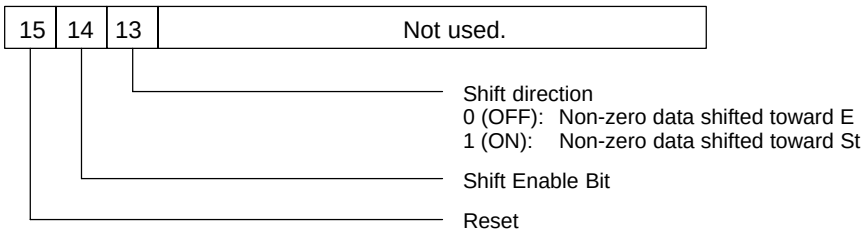
When the execution condition is OFF, ASFT(052) is not executed. When the execution condition is ON, ASFT(052) is used to create and control a reversible asynchronous word shift register between St and E. This shift register reverses the contents of adjacent words when the content of one of the words 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 115 for general precautions on operand data areas.

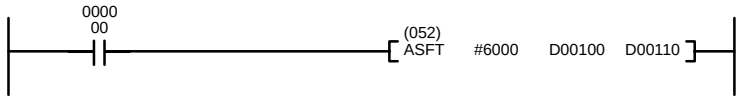
Flags

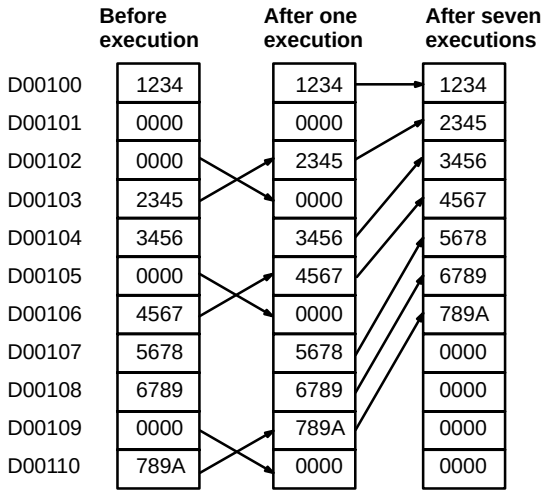
ER (A50003): Content of ∗DM word is not BCD when set for BCD.

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)

Ladder Symbol

(053)
WSFT S St E

Variations

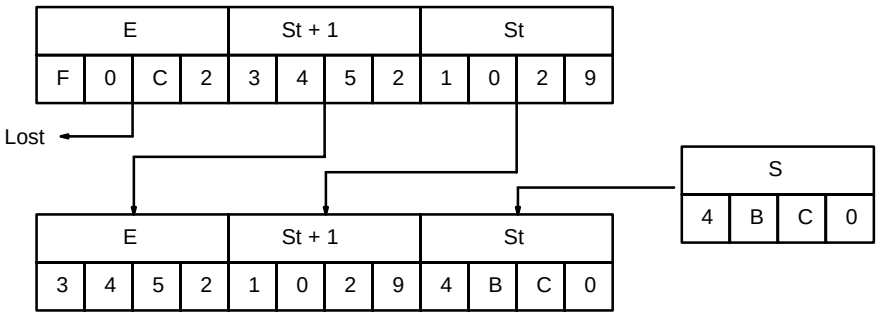
↑ WSFT(053)

Operand Data Areas

S: Source word	CIO, G, A, T, C, #, DM, DR, IR
St: Starting word	CIO, G, A, DM
E: End word	CIO, G, A, DM

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 115 for general precautions on operand data areas.

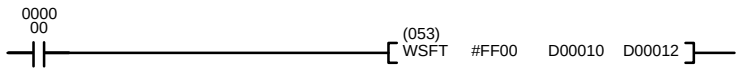
Flags

ER (A50003): Content of ∗DM word is not BCD when set for BCD.

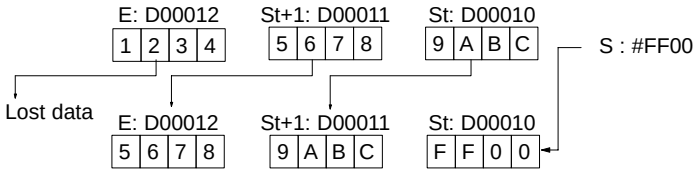
The St and E words are in different areas, or St is greater than E.

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) (CVM1 V2)

Ladder Symbol

(054)
NSFL D C N

Variations

↑NSFL(054)

Operand Data Areas

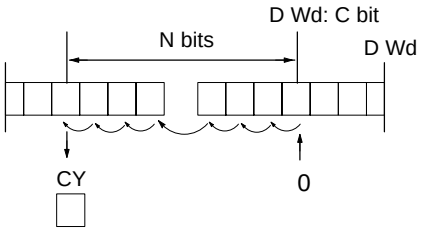
D: Beginning word for shift	CIO, G, A
C: Beginning bit	CIO, G, A, T, C, #, DM, DR, IR
N: Shift data length	CIO, G, A, T, C, #, DM, DR, IR

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 115 for general precautions on operand data areas.

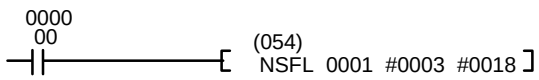
Flags

ER (A50003): C is not 0000 to 0015 BCD.
Content of a∗DM word is not BCD when set for BCD.

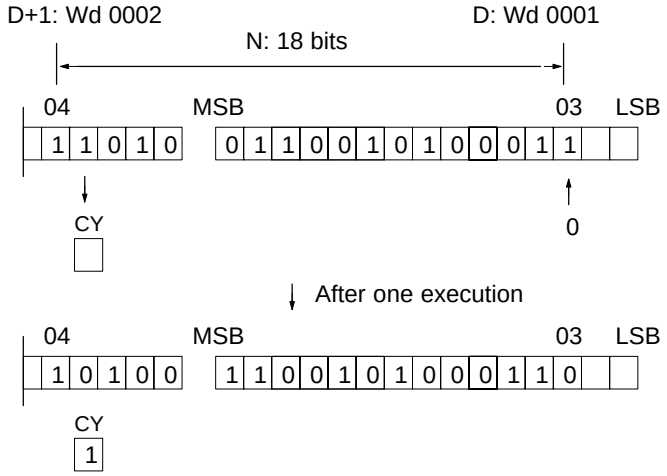
CY (A50004): "1" has been shifted to CY.

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) (CVM1 V2)

Ladder Symbol

(055)
NSFR D C N

Variations

↑NSFR(055)

Operand Data Areas

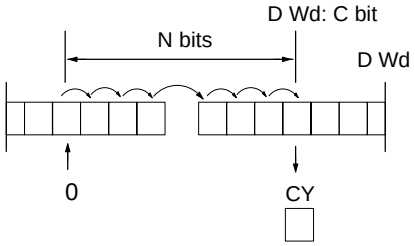
D: Beginning word for shift	CIO, G, A
C: Beginning bit	CIO, G, A, T, C, #, DM, DR, IR
N: Shift data length	CIO, G, A, T, C, #, DM, DR, IR

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 115 for general precautions on operand data areas.

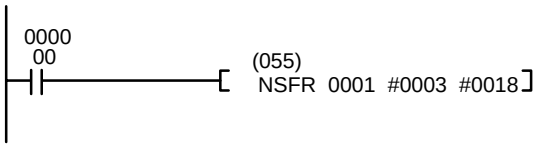
Flags

ER (A50003): C is not 0000 to 0015 BCD.
Content of a∗DM word is not BCD when set for BCD.

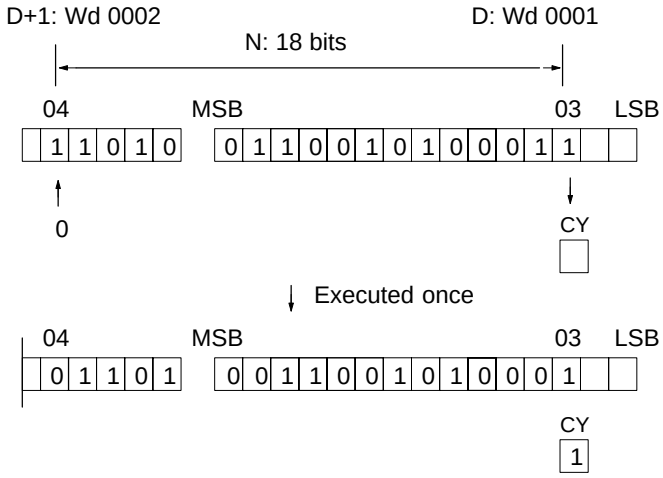
CY (A50004): "1" has been shifted to CY.

Example

When CIO 000000 is ON in the following example, the18 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.

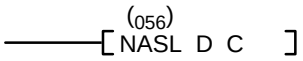


Address	Instruction	Operands
00000	LD	000000
00001	NSFR(055)	
		0001
		#0003
		#0018



5-14-7 SHIFT N-BITS LEFT: NASL(056) (CVM1 V2)

Ladder Symbol (NASL)



Operand Data Areas

D: Shift word	CIO, G, A, DM, DR, IR
C: Control word	CIO, G, A, T, C, #, DM, DR, IR

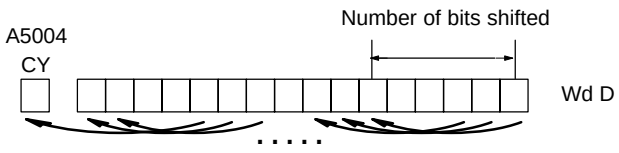
Variations

↑NASL(056)

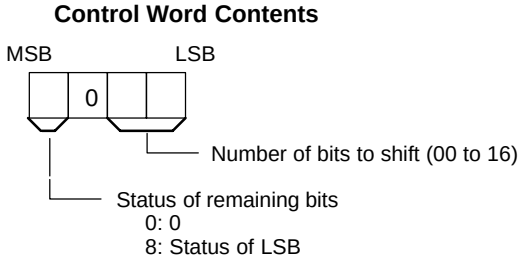
Description

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 115 for general precautions on operand data areas.

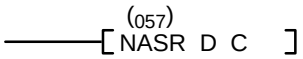
Flags

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) (CVM1 V2)

Ladder Symbol



Variations

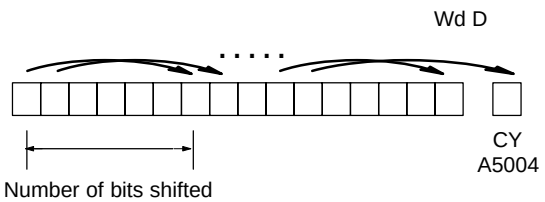
↑NASR(057)

Operand Data Areas

D: Shift word	CIO, G, A, DM, DR, IR
C: Control word	CIO, G, A, T, C, #, DM, DR, IR

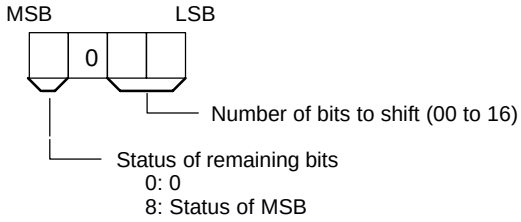
Description When the execution condition is OFF, NASR(057) is not executed. When the execution condition is ON, NASR(057) shifts the status of the 16 bits in the specified word to the right the specified number of bits

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.

Control Word Contents



Precautions

Refer to page 115 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) (CVM1 V2)

Ladder Symbol

Variations

↑NSLL(058)

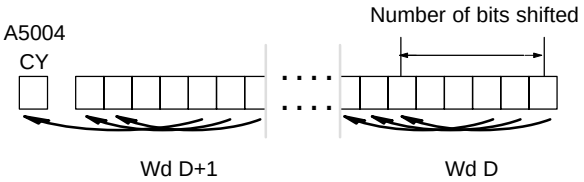
Operand Data Areas

D: Beginning shift word	CIO, G, A, DM
C: Control word	CIO, G, A, T, C, #, DM

Description

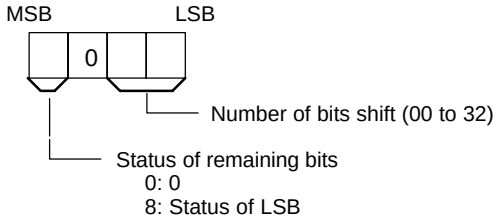
When the execution condition is OFF, NSLL(058) is not executed. When the execution condition is ON, NSLL(058) shifts the status of the 32 bits in the specified words to the left the specified number of bits.

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.

Control Word Contents



Precautions

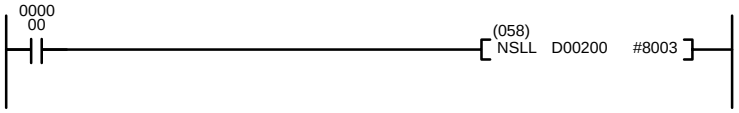
Refer to page 115 for general precautions on operand data areas.

Flags

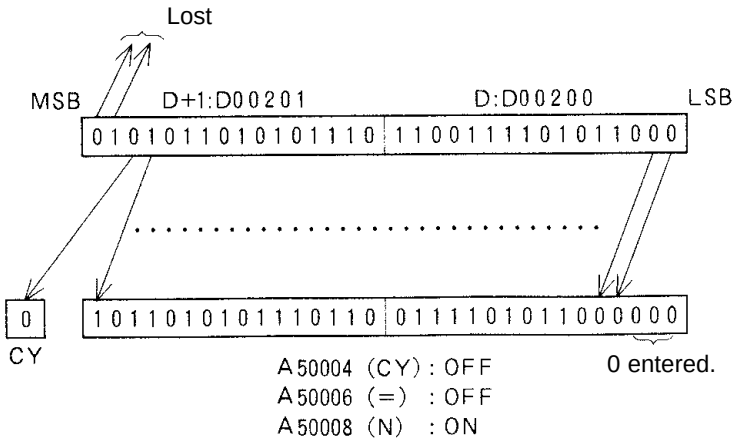
ER (A50003):	Number of bits to shift is out of range.
	Content of a∗DM word is not BCD when set for BCD.
CY (A50004):	“1” has been shifted to CY.
EQ (A50006)	Content of words D and D+1 after the shift is all zeros.
N (A50008)	Same status as leftmost bit (MSB) of word D+1 after shift.

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.”

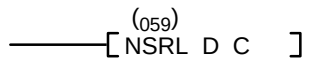


Address	Instruction	Operands
00000	LD	000000
00001	NSLL(058)	
		D00200
		#8003



5-14-10 DOUBLE SHIFT N-BITS RIGHT: NSRL(059) (CVM1 V2)

Ladder Symbol



Variations

↑NSRL(059)

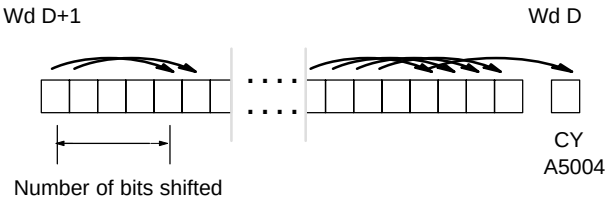
Operand Data Areas

D: Shift word address	CIO, G, A, DM,
C: Control word	CIO, G, A, T, C, #, DM,

Description

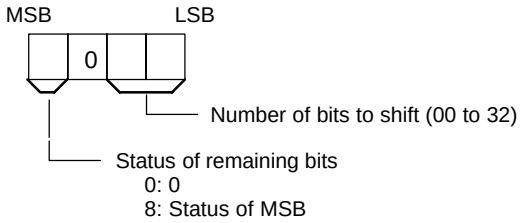
When the execution condition is OFF, NSRL(059) is not executed. When the execution condition is ON, NSRL(059) shifts the status of the 32 bits in the specified words to the right the specified number of bits.

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

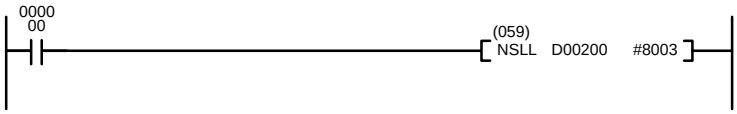
Refer to page 115 for general precautions on operand data areas.

Flags

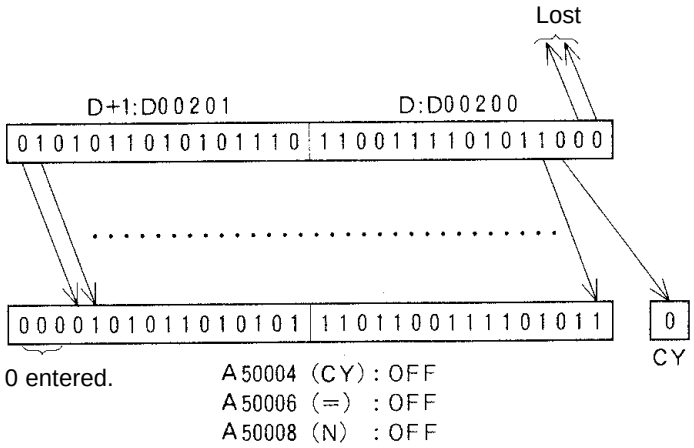
ER (A50003):	Number of bits to shift is out of range. Content of a∗DM word is not BCD when set for BCD.
CY (A50004):	"1" has been shifted to CY.
EQ (A50006)	Content of word D and D+1 after the shift is all zeros.
N (A50008)	Same as leftmost bit (MSB) of word D+1 after shift.

Example

When CIO 000000 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)

Ladder Symbol

(060)
ASL Wd

Operand Data Area

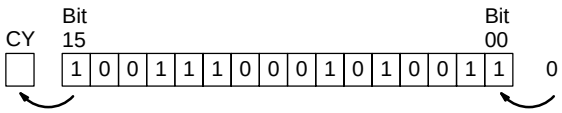
Wd: Word CIO, G, A, DM,

Variations

↑ ASL(060)

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.

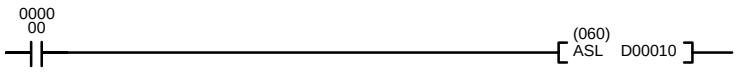
CY (A50004): Receives the status of bit 15.

EQ (A50006): Content of Wd is 0 after a shift.

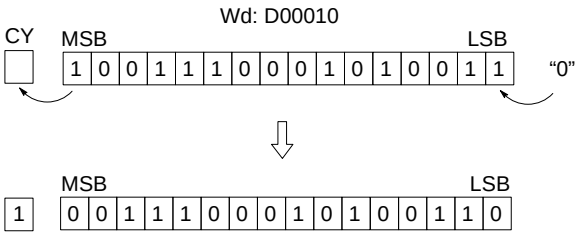
N (A50008): Same status as bit 15 of D + 1 after shift.

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.

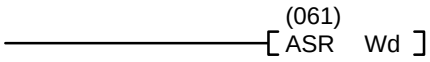


Address	Instruction	Operands
00000	LD	000000
00001	ASL(060)	
		D00010



5-14-12 ARITHMETIC SHIFT RIGHT: ASR(061)

Ladder Symbol



Operand Data Area

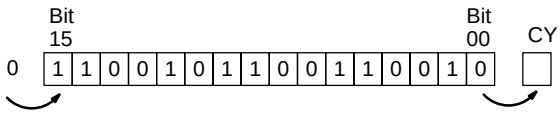
Wd: Word	CIO, G, A, DM, DR, IR

Variations

↑ ASR(061)

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.

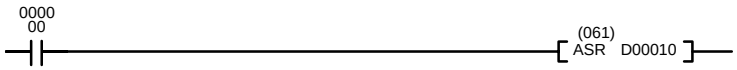
CY (A50004): Receives the status of bit 00.

EQ (A50006): Content of Wd is 0 after a shift.

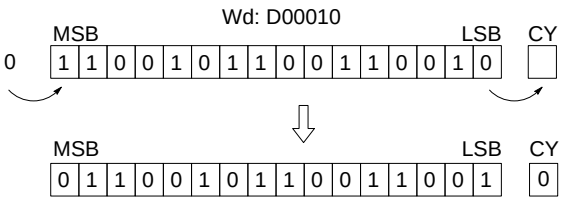
N (A50008): OFF.

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.

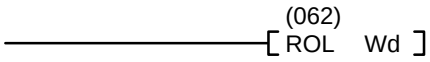


Address	Instruction	Operands
00000	LD	000000
00001	ASR(061)	
		D00010



5-14-13 ROTATE LEFT: ROL(062)

Ladder Symbol



Operand Data Area

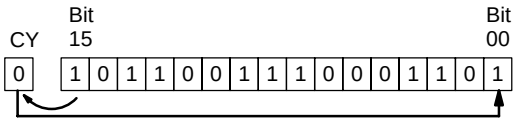
Wd: Word	CIO, G, A, DM, DR, IR

Variations

↑ ROL(062)

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).

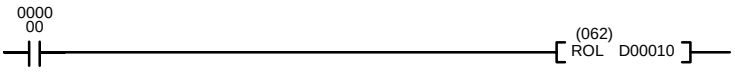
Note Refer to page 115 for general precautions on operand data areas.

Flags

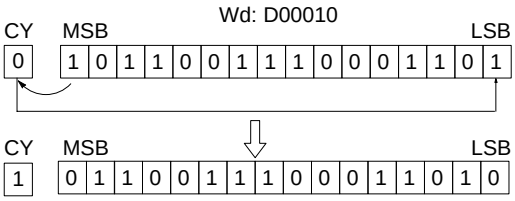
ER (A50003): Content of ∗DM word is not BCD when set for BCD.
CY (A50004): Receives the status of bit 15 from Wd.
EQ (A50006): Content of Wd is 0 after execution.
N (A50008): Same status as bit 15 of Wd after execution.

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.

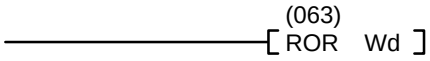


Address	Instruction	Operands
00000	LD	000000
00001	ROL(062)	
		D00010



5-14-14 ROTATE RIGHT: ROR(063)

Ladder Symbol



Operand Data Area

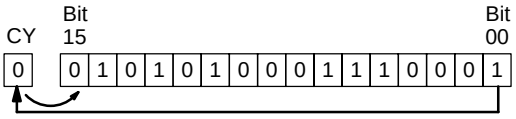
Wd: Word	CIO, G, A, DM, DR, IR

Variations

↑ ROR(063)

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 115 for general precautions on operand data areas.

Flags

ER (A50003): Content of ∗DM word is not BCD when set for BCD.

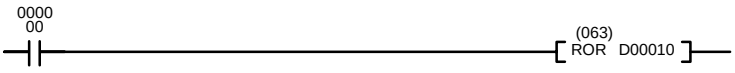
CY (A50004): Receives the status of bit 15 from Wd.

EQ (A50006): Content of Wd is 0 after execution.

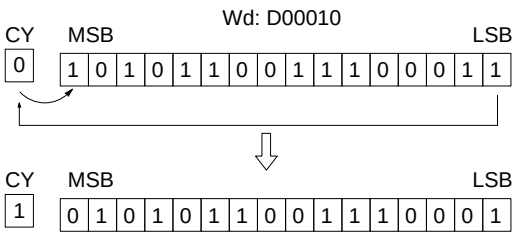
N (A50008): Same status as bit 15 of Wd after execution.

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)

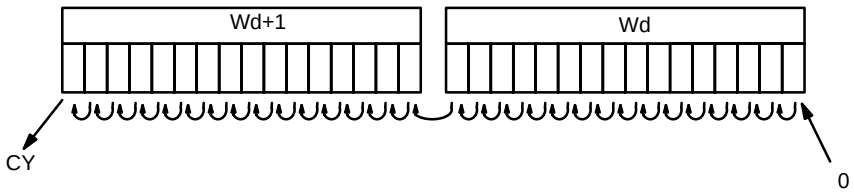
Ladder Symbol	Operand Data Area	
(064) ASLL Wd	**Wd: Word**	CIO, G, A, DM

Variations

↑ ASLL(064)

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 115 for general precautions on operand data areas.

Flags

ER (A50003): Content of ∗DM word is not BCD when set for BCD.

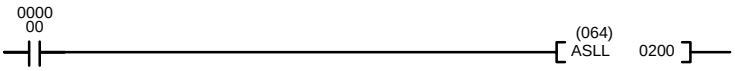
CY (A50004): Receives the status of bit 15 from Wd+1.

EQ (A50006): Content of Wd and Wd+1 are 0 after a shift.

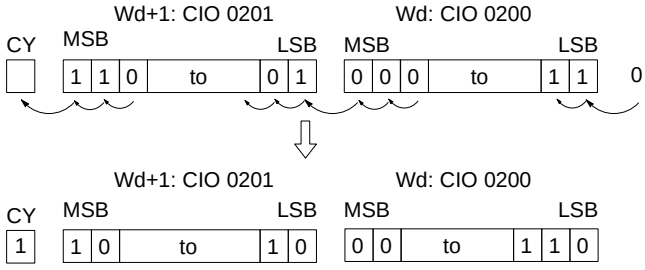
N (A50008): Same status as bit 15 of Wd+1 after shift.

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)

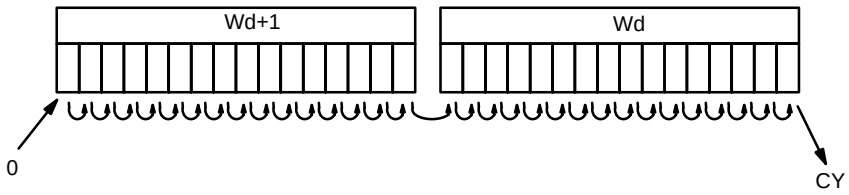
Ladder Symbol	Operand Data Area	
(065) ASRL Wd	**Wd: Word**	CIO, G, A, DM

Variations

↑ ASRL(065)

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 115 for general precautions on operand data areas.

Flags

ER (A50003): Content of ∗DM word is not BCD when set for BCD.

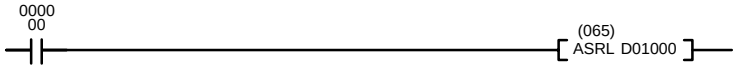
CY (A50004): Receives the status of bit 00 from Wd.

EQ (A50006): Content of Wd and Wd+1 are 0 after a shift.

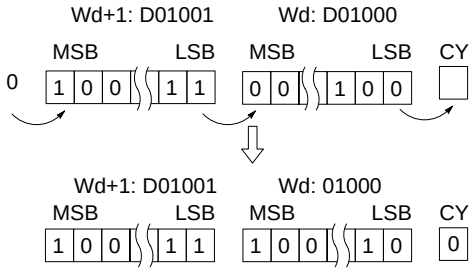
N (A50008): Same status as bit 15 of Wd+1 after shift.

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.

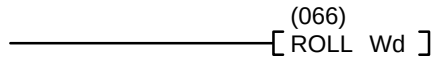


Address	Instruction	Operands
00000	LD	000000
00001	ASRL(065)	
		D01000



5-14-17 DOUBLE ROTATE LEFT: ROLL(066)

Ladder Symbol



Operand Data Area

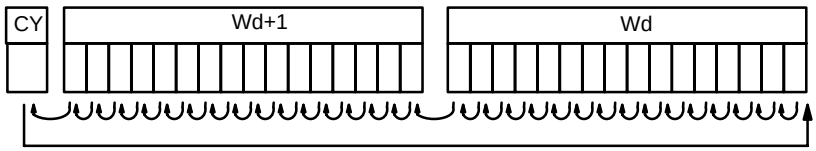
Wd: Word	CIO, G, A, DM

Variations

↑ ROLL(066)

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 115 for general precautions on operand data areas.

Flags

ER (A50003): Content of ∗DM word is not BCD when set for BCD.

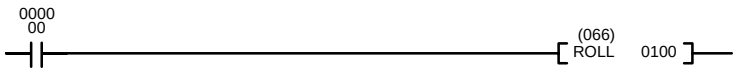
CY (A50004): Receives the status of bit 15 from Wd+1.

EQ (A50006): Content of Wd and Wd+1 are 0 after execution.

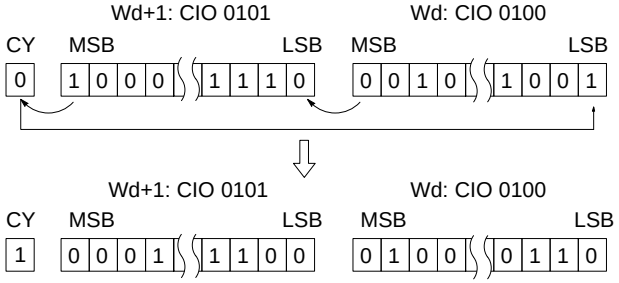
N (A50008): Same status as bit 15 of Wd+1 after execution.

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.

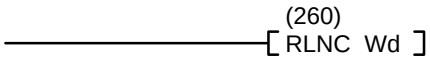


Address	Instruction	Operands
00000	LD	000000
00001	ROLL(066)	
		0100



5-14-18 ROTATE LEFT WITHOUT CARRY: RLNC(260) (CVM1 V2)

Ladder Symbol



Operand Data Area

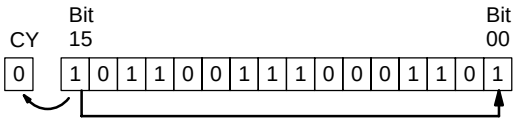
Wd: Word	CIO, G, A, DM, DR, IR

Variations

↑RNLC(260)

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.

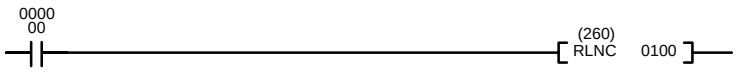
CY (A50004): Receives the status of bit 15 from Wd.

EQ (A50006): Content of Wd is 0 after execution.

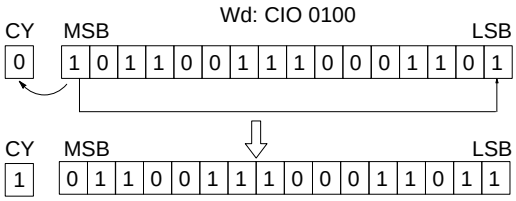
N (A50008): Same status as bit 15 of Wd after execution.

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) (CVM1 V2)

Ladder Symbol

(262)
RLNL Wd

Operand Data Area

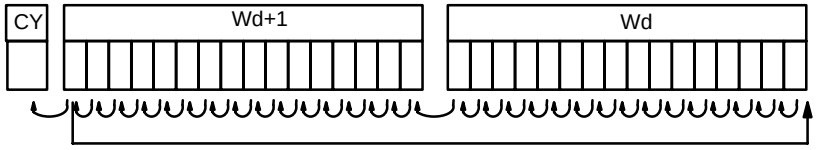
Wd: Word	CIO, G, A, DM

Variations

↑RLNL(262)

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 115 for general precautions on operand data areas.

Flags

ER (A50003): Content of ∗DM word is not BCD when set for BCD.

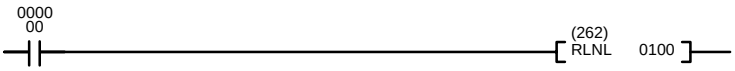
CY (A50004): Receives the status of bit 15 from Wd+1.

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	000000
00001	RLNL(262)	
		0100

5-14-20 DOUBLE ROTATE RIGHT: RORL(067)

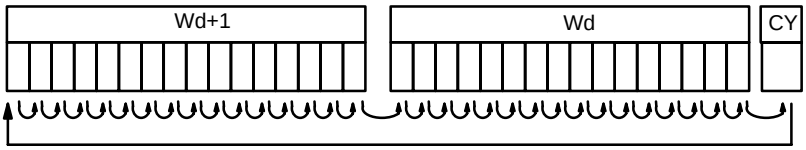
Ladder Symbol	Operand Data Area	
(067) RORL Wd	**Wd: Word**	CIO, G, A, DM

Variations

↑ RORL(067)

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 115 for general precautions on operand data areas.

Flags

ER (A50003): Content of ∗DM word is not BCD when set for BCD.

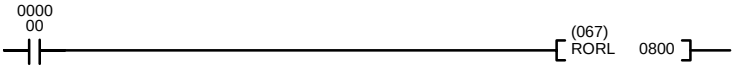
CY (A50004): Receives the status of bit 00 from Wd.

EQ (A50006): Content of Wd and Wd+1 are 0 after execution.

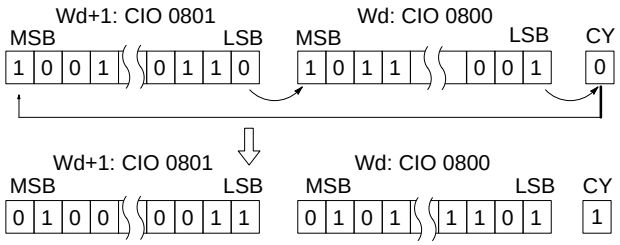
N (A50008): Same status as bit 15 of Wd+1 after execution.

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.

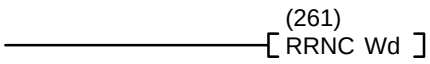


Address	Instruction	Operands
00000	LD	000000
00001	RORL(067)	
		0800



5-14-21 ROTATE RIGHT WITHOUT CARRY: RRNC(261) (CVM1 V2)

Ladder Symbol



Operand Data Area

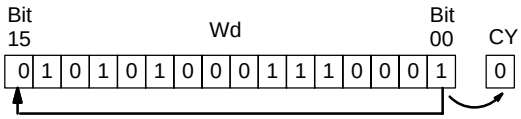
Wd: Word	CIO, G, A, DM, DR, IR

Variations

↑RRNC(261)

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.

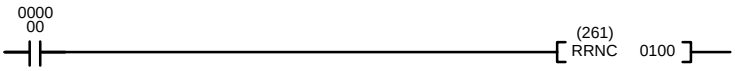
CY (A50004): Receives the status of bit 15 from Wd.

EQ (A50006): Content of Wd is 0 after execution.

N (A50008): Same status as bit 15 of Wd after execution.

Example

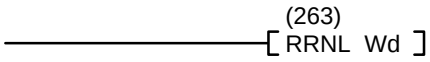
When CIO 000000 is ON in the following example, the status of bit 00 of CIO 0100 is shifted into bit 15 of CIO 0100 and into CR and the status of all bits within CIO 0100 are shifted right one position.



Address	Instruction	Operands
00000	LD	000000
00001	RRNC(261)	
		0100

5-14-22 DOUBLE ROTATE RIGHT W/O CARRY: RRNL(263) (CVM1 V2)

Ladder Symbol



Operand Data Area

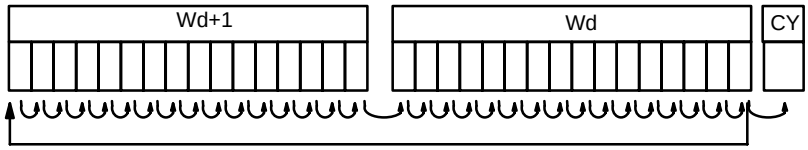
Wd: Word	CIO, G, A, DM

Variations

↑RRNL(263)

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

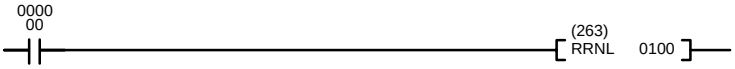
Refer to page 115 for general precautions on operand data areas.

Flags

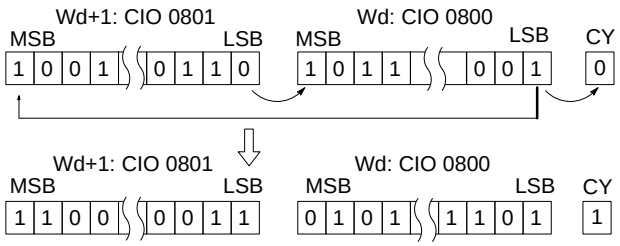
ER (A50003): Content of ∗DM word is not BCD when set for BCD.
CY (A50004): Receives the status of bit 00 from Wd.
EQ (A50006): Content of Wd and Wd+1 are 0 after execution.
N (A50008): Same status as bit 15 of Wd+1 after execution.

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)

Ladder Symbol

(068)
SLD St E

Operand Data Areas

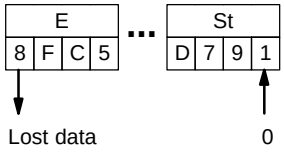
St: Starting word	CIO, G, A, DM
E: End word	CIO, G, A, DM

Variations

↑ SLD(068)

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 115 for general precautions on operand data areas.

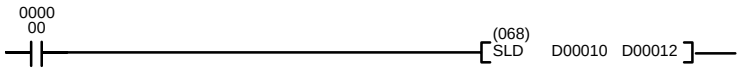
Flags

ER (A50003): Content of ∗DM word is not BCD when set for BCD.

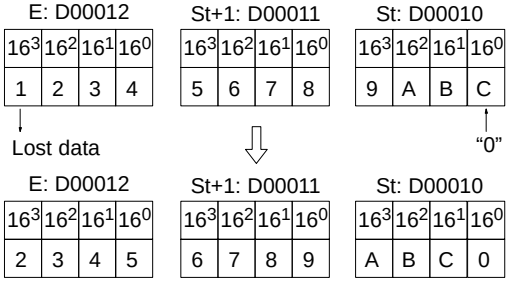
St and E are in different areas, or St is greater than E.

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)

Ladder Symbol

(069)
SRD St E

Operand Data Areas

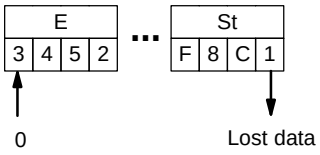
St: Starting word	CIO, G, A, DM
E: End word	CIO, G, A, DM

Variations

↑ SRD(069)

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 115 for general precautions on operand data areas.

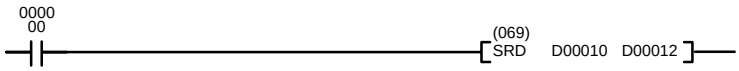
Flags

ER (A50003): Content of ∗DM word is not BCD when set for BCD.

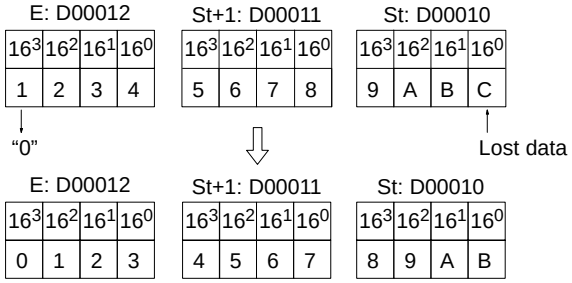
The St and E words are in different areas, or St is greater than E.

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

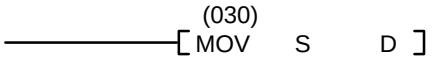


5-15 Data Movement Instructions

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)

Ladder Symbol



Operand Data Areas

S: Source	CIO, G, A, T, C, #, DM, DR, IR
D: Destination	CIO, G, A, T, C, DM, DR, IR

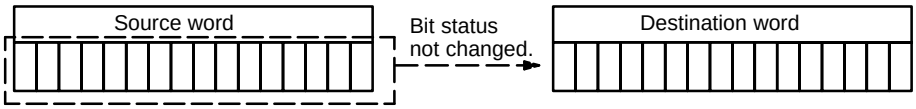
Variations

↑ MOV(030) ! ↑ MOV(030)
! MOV(030)

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 refreshed just before, and output bits used for D will be refreshed just after execution.



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.

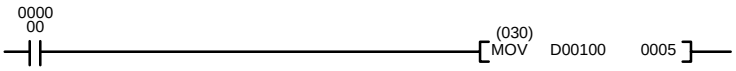
Note Refer to page 115 for general precautions on operand data areas.

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

Before execution

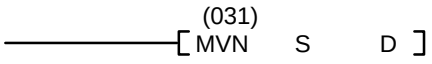
D00100	3	5	2	1
0005	0	0	0	0

After execution

D00100	3	5	2	1
0005	3	5	2	1

5-15-2 MOVE NOT: MVN(031)

Ladder Symbol



Variations

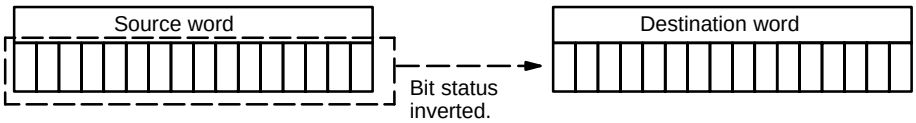
↑ MVN(031)

Operand Data Areas

S: Source	CIO, G, A, T, C, #, DM, DR, IR
D: Destination	CIO, G, A, T, C, DM, DR, IR

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

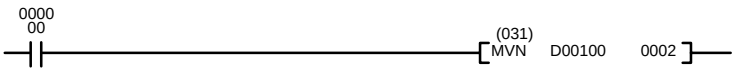
ER (A50003): Content of ∗DM word is not BCD when set for BCD.

EQ (A50006): Content of D is 0 after execution.

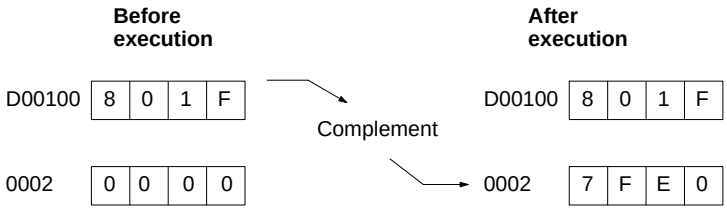
N (A50008): Same status as bit 15 of D after execution.

Example

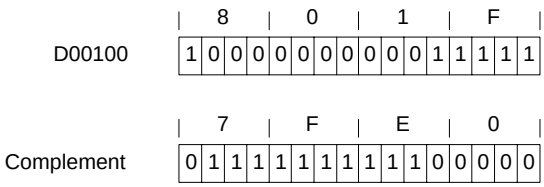
When CIO 000000 is ON in the following example, the complement of the content of D00100 is transferred to the CIO 0002.



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)

Ladder Symbol

(032)
MOVL S D

Variations

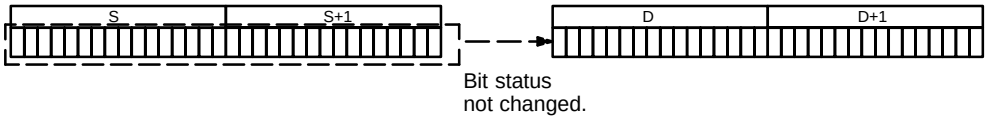
↑ MOVL(032)

Operand Data Areas

S: Source	CIO, G, A, T, C, #, DM
D: Destination	CIO, G, A, T, C, DM

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 115 for general precautions on operand data areas.

Flags

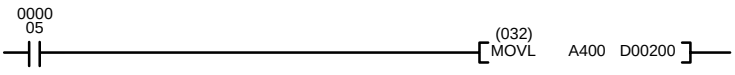
ER (A50003): Content of ∗DM word is not BCD when set for BCD.

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

A400	3	7	0	0
A401	0	1	F	9
D00200	0	0	0	0
D00201	0	0	0	0

After execution

A400	3	7	0	0
A401	0	1	F	9
D00200	3	7	0	0
D00201	0	1	F	9

5-15-4 DOUBLE MOVE NOT: MVNL(033)

Ladder Symbol

(033)
MVNL S D

Variations

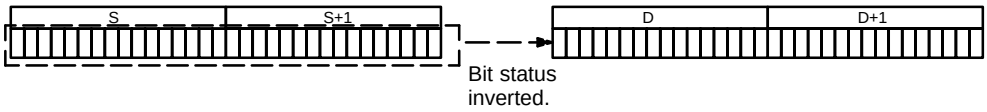
↑ MVNL(033)

Operand Data Areas

S: Source	CIO, G, A, T, C, #, DM
D: Destination	CIO, G, A, T, C, DM

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 115 for general precautions on operand data areas.

Flags

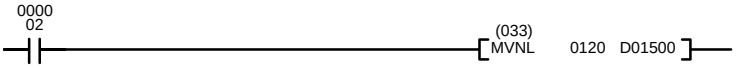
ER (A50003): Content of ∗DM word is not BCD when set for BCD.

EQ (A50006): Content of D and D+1 are 0 after execution.

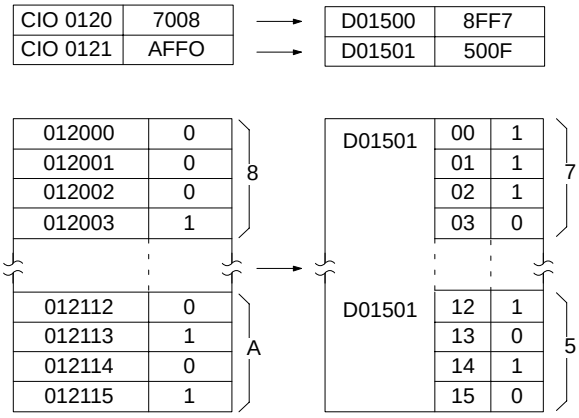
N (A50008): Same status as bit 15 of D+1 after execution.

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.

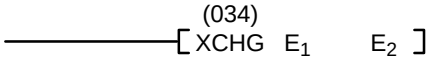


Address	Instruction	Operands
00000	LD	000002
00001	MVNL(033)	
		0120
		D01500



5-15-5 DATA EXCHANGE: XCHG(034)

Ladder Symbol



Variations

↑ XCHG(034)

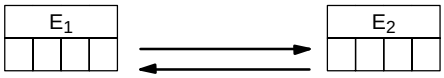
Operand Data Areas

E_1: 1st Exchange word CIO, G, A, T, C, DM, DR, IR

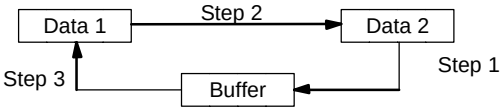
E_2: 2nd Exchange word CIO, G, A, T, C, DM, DR, IR

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_1 and E_2.



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.

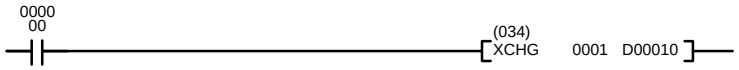


Flags

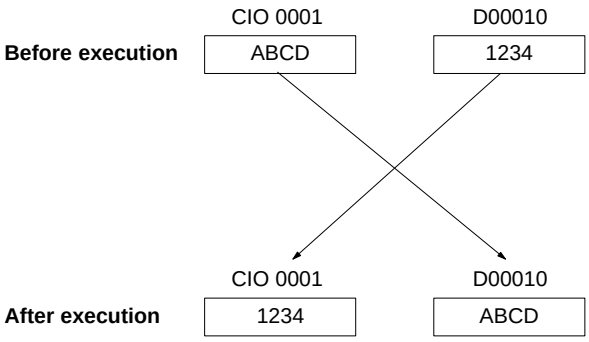
ER (A50003): Content of ∗DM word is not BCD when set for BCD.

Example

When CIO 000000 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.

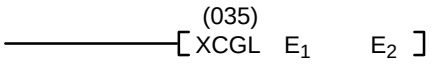


Address	Instruction	Operands
00000	LD	000000
00001	XCHG(034)	
		0001
		D00010



5-15-6 DOUBLE DATA EXCHANGE: XCGL(035)

Ladder Symbol



Operand Data Areas

E_1: 1st Exchange word CIO, G, A, T, C, DM

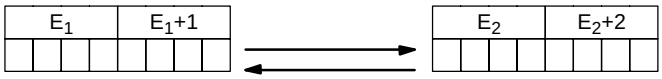
E_2: 2nd Exchange word CIO, G, A, T, C, DM

Variations

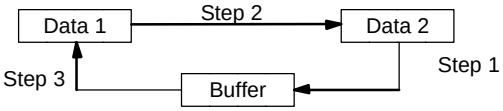
↑ XCGL(035)

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_1 and E_1+1 with that of E_2 and E_2+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

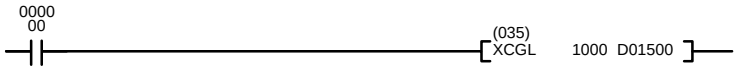
Refer to page 115 for general precautions on operand data areas.

Flags

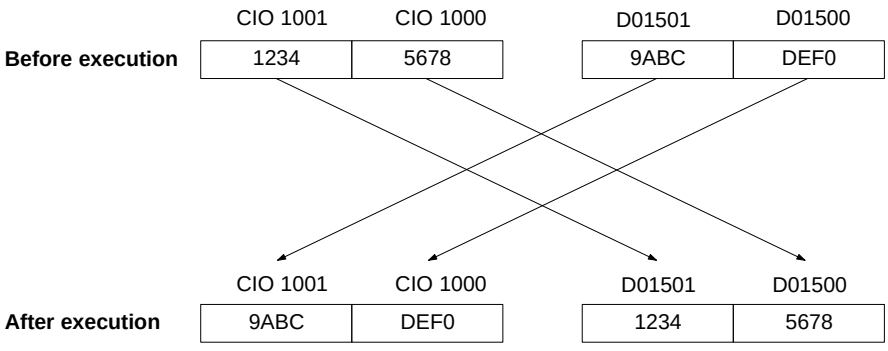
ER (A50003): Content of ∗DM word is not BCD when set for BCD.

Example

When CIO 000000 is ON in the following example, the contents of 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

(036)
MOVR S D

Operand Data Areas

S: Source	CIO, G, A, TN, ST, T, C, DM
D: Destination	IR

Variations

↑ 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.

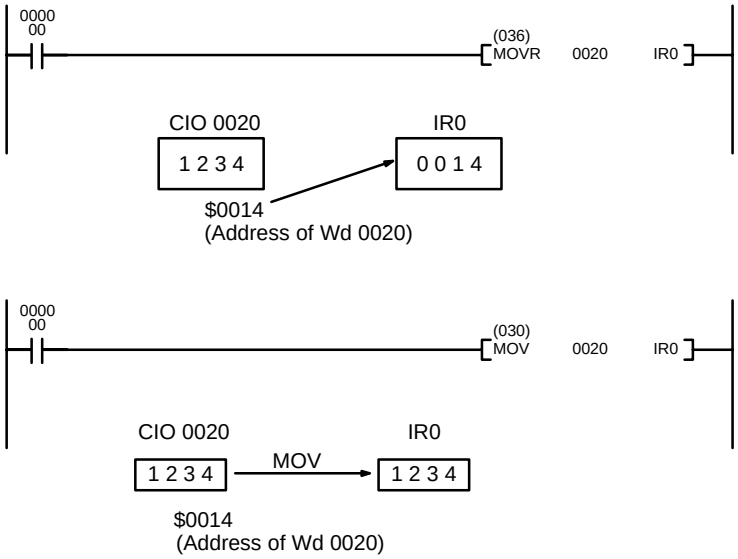
Flags

ER (A50003): Content of ∗DM word is not BCD when set for BCD.

EQ (A50006): 0000 was placed in the index register.

Example

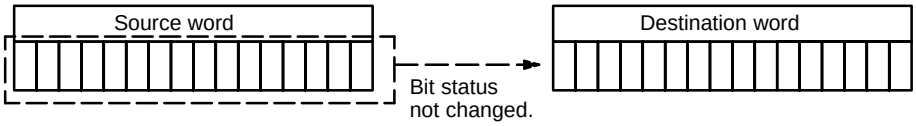
The following example demonstrates the difference between MOV(030) and MOVR(036). In the first instruction line, MOVR(036) copies the PC memory address of CIO 0020 to IR0 when CIO 000000 is ON. In the second instruction line, MOV(030) copies the content of CIO 0020 to IR0 when CIO 000000 is ON.



5-15-8 MOVE QUICK: MOVQ(037)

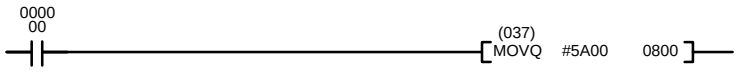
Ladder Symbol	Operand Data Areas	
(037) MOVQ S D	**S: Source**	CIO, G, A, T, C, #
	D: Destination	CIO, G, A, T, C

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

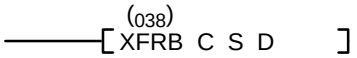
“0”	2^0	0
	2^1	0
	2^2	0
	2^3	0
“0”	2^4	0
	2^5	0
	2^6	0
	2^7	0
“A”	2^8	0
	2^9	1
	2^{10}	0
	2^{11}	0
“5”	2^{12}	1
	2^{13}	0
	2^{14}	1
	2^{15}	0

D: CIO 0800

080000	0
080001	0
080002	0
080003	0
080004	0
080005	0
080006	0
080007	0
080008	0
080009	1
080010	0
080011	1
080012	1
080013	0
080014	1
080015	0

5-15-9 MULTIPLE BIT TRANSFER: XFRB(038) (CVM1 V2)

Ladder Symbol



Operand Data Areas

C: Control word	CIO, G, A, T, C, #, DM, DR, IR
S: First source word	CIO, G, A, T, C, DM
D: First destination word	CIO, G, A, DM

Variations

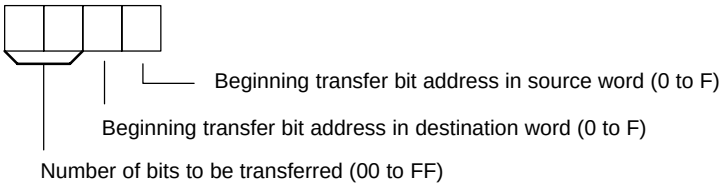
↑XFRB(038)

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

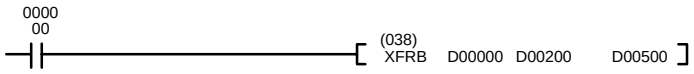
Be sure that the last word in the transfer source or transfer destination does not exceed the data area.

Flags

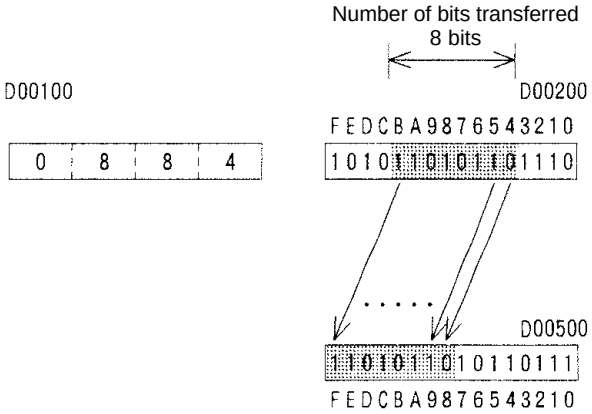
ER (A50003): Content of a∗DM word is not BCD when set for BCD.

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.

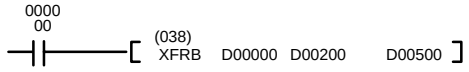


Address	Instruction	Operands
00000	LD	000000
00001	XFRB(038)	
		D00000
		D00200
		D00500

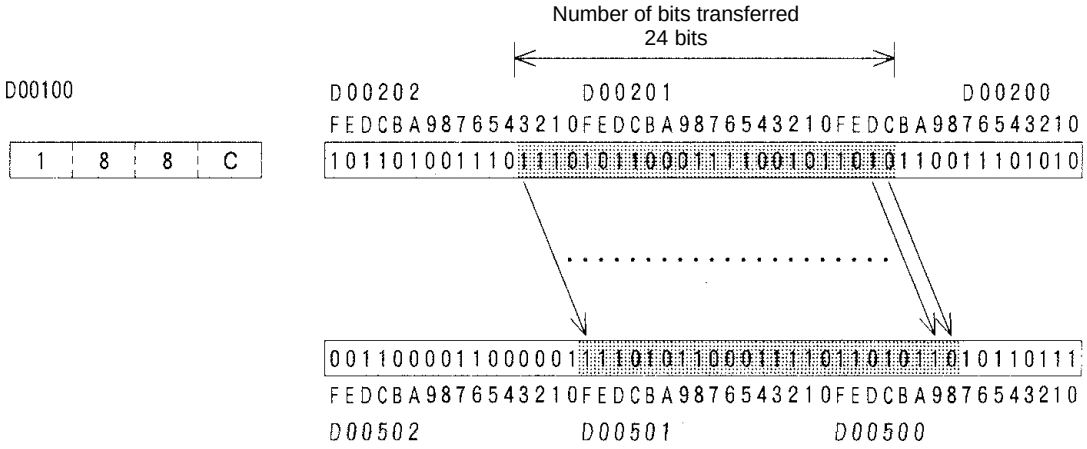


Example 2

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)

Ladder Symbol

```
(040)
XFER   N    S    D
```

Variations

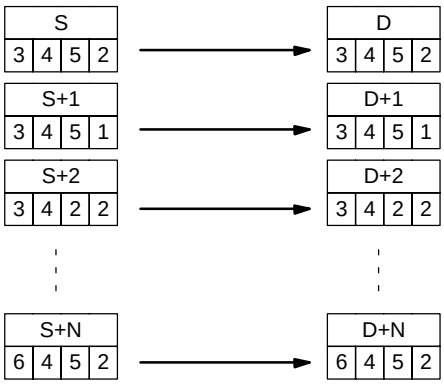
↑ XFER(040)

Operand Data Areas

N: Number of words	CIO, G, A, T, C, #, DM, DR, IR
S: 1st source word	CIO, G, A, T, C, DM
D: 1st destination word	CIO, G, A, T, C, DM

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. For version-2 CVM1 CPUs, transfer source and destination words can overlap. This is not possible for other CPUs.
2. Refer to page 115 for general precautions on operand data areas.

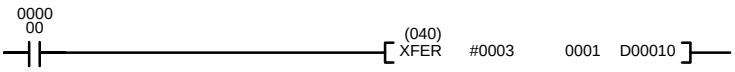
Flags

ER (A50003): N is not BCD.

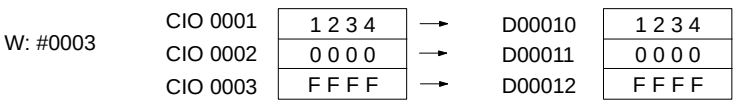
Content of ∗DM word is not BCD when set for BCD.

Example

When CIO 000000 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).

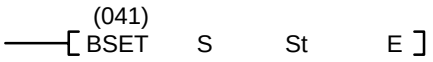


Address	Instruction	Operands
00000	LD	000000
00001	XFER(040)	
		#0003
		0001
		D00010



5-15-11 BLOCK SET: BSET(041)

Ladder Symbol



Variations

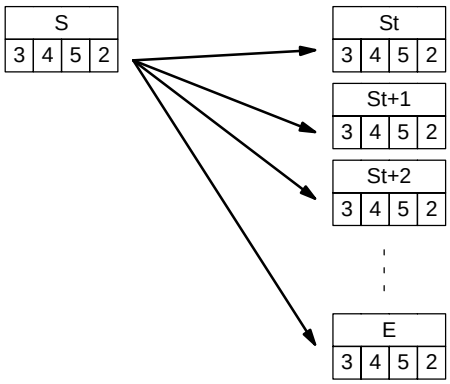
↑ BSET(041)

Operand Data Areas

S: Source word	CIO, G, A, T, C, #, DM, DR, IR
St: Starting word	CIO, G, A, T, C, DM
E: End word	CIO, G, A, T, C, DM

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.

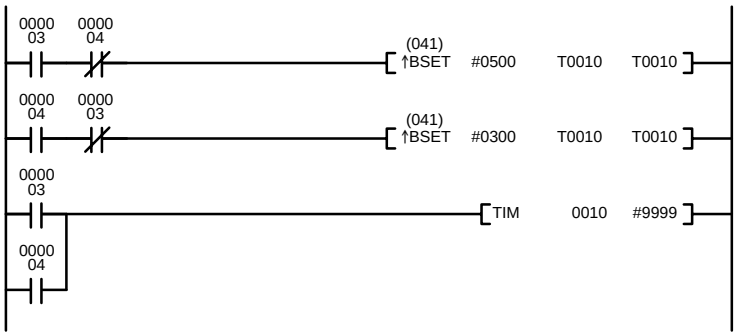
Note Refer to page 115 for general precautions on operand data areas.

Flags

ER (A50003): Content of ∗DM word is not BCD when set for BCD.
St is greater than E.

Example

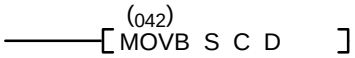
The following example shows how to use BSET(041) to change the PV of a timer depending on the status of CIO 000003 and CIO 000004. When CIO 000003 is ON, TIM 0010 will operate as a 50-second timer; when CIO 000004 is ON, TIM 0010 will operate as a 30-second timer.



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)

Ladder Symbol



Variations

↑MOVB(042)

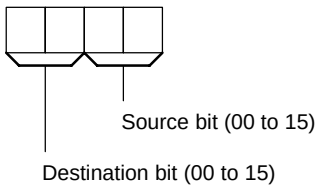
Operand Data Areas

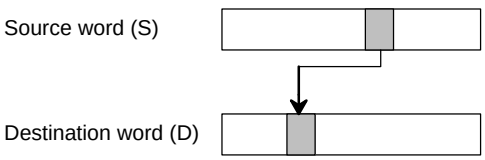
S: Source word or data	CIO, G, A, #, DM, DR, IR
C: Control word	CIO, G, A, T, C, #, DM, DR, IR
D: Destination word	CIO, G, A, DM, DR, IR

Description

When the execution condition is OFF, MOVB(042) is not executed. When the execution condition is ON, MOVB(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.

Control Word Contents





Precautions

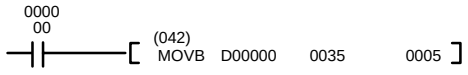
C must be BCD and must be within the values specified above.

Flags

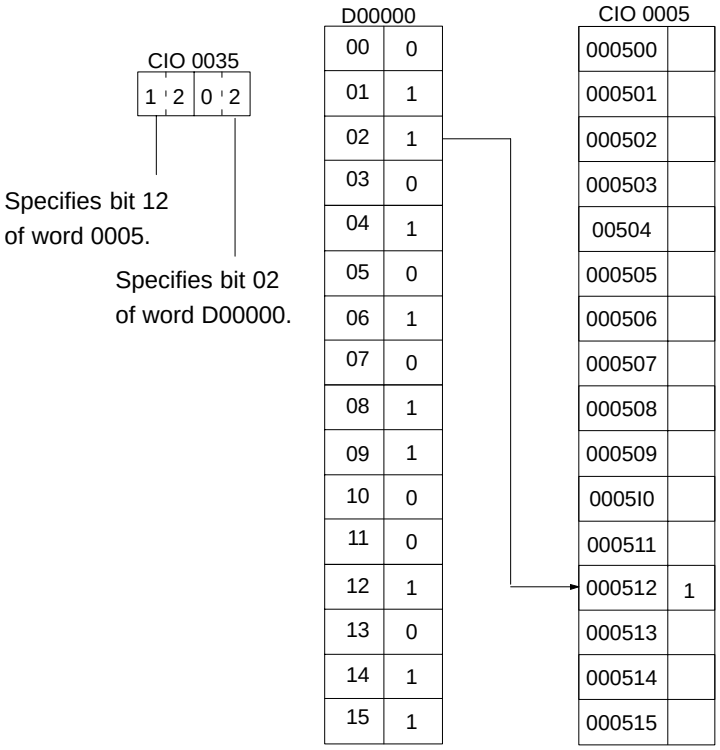
ER (A50003): Control word content is not BCD.
Rightmost and leftmost eight bits are not 00 to 15.
Content of a∗DM word is not BCD when set for BCD.

Example

When CIO 000000 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)

Ladder Symbol

(043)
MOVD S Di D

Variations

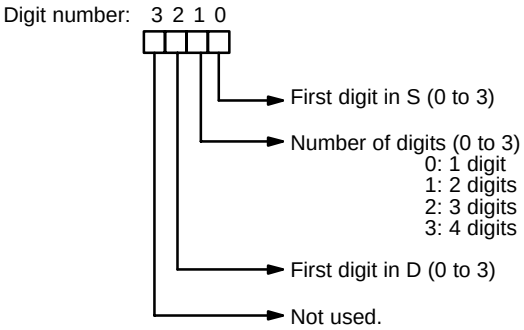
↑ MOVD(043)

Operand Data Areas

S: Source word	CIO, G, A, T, C, #, DM, DR, IR
Di: Digit designator	CIO, G, A, T, C, #, DM, DR, IR
D: Destination word	CIO, G, A, T, C, DM, DR, IR

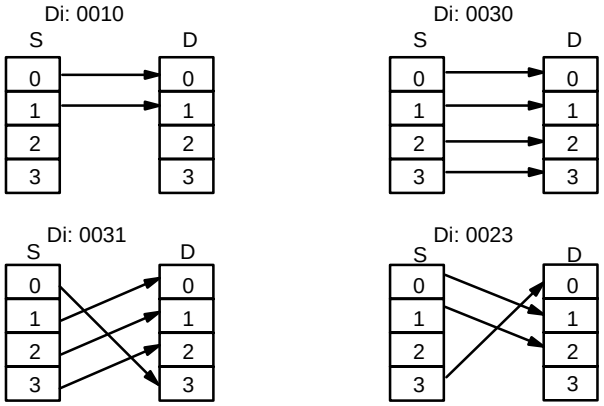
Description

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.



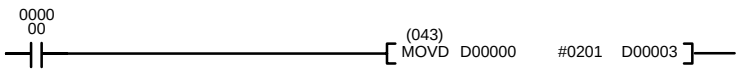
Precautions

The rightmost three digits of Di must each be between 0 and 3.

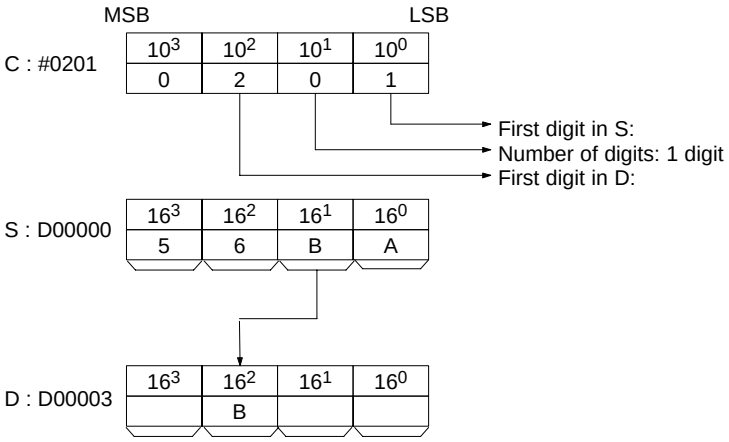
Note Refer to page 115 for general precautions on operand data areas.

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)

Ladder Symbol

(044)
DIST S DBs Of

Variations

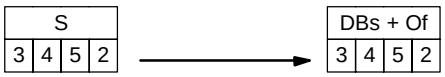
↑ DIST(044)

Operand Data Areas

S: Source data	CIO, G, A, T, C, #, DM, DR, IR
DBs: Destination base	CIO, G, A, T, C, DM
Of: Offset data	CIO, G, A, T, C, DM

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.

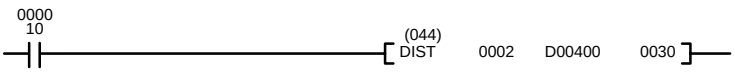
Note Refer to page 115 for general precautions on operand data areas.

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)

Ladder Symbol

(045)
COLL SBs Of D

Variations

↑ COLL(045)

Operand Data Areas

SBs: Source base CIO, G, A, T, C, DM

Of: Offset data CIO, G, A, T, C, #, DM, DR, IR

D: Destination word CIO, G, A, T, C, DM, DR, IR

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.

SBs + Of				→	D			
3	4	5	2		3	4	5	2

Precautions Of must be BCD.

Note Refer to page 115 for general precautions on operand data areas.

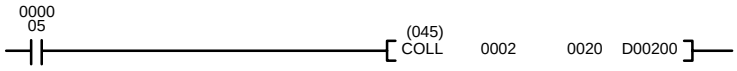
Flags

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).

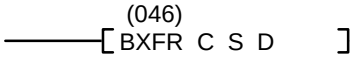


Address	Instruction	Operands
00000	LD	000005
00001	COLL(045)	
		0002
		0020
		D00200

	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) (CVM1 V2)

Ladder Symbol



Operand Data Areas

C: First control word	CIO, G, A, T, C, DM
S: First source word	CIO, G, A, T, C, DM
D: First destination word	CIO, G, A, T, C, DM

Variations

↑BXFR(046)

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.		
$x10^3$	$x10^2$	$x10^1$	$x10^0$	Bank no.: 0 to 7
0	0	0	$x10^4$	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 115 for general precautions on operand data areas.

Flags

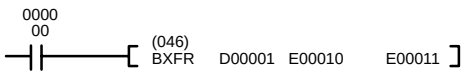
ER (A50003): Bank number is not 00 to 07.

No EM for the specified bank number.

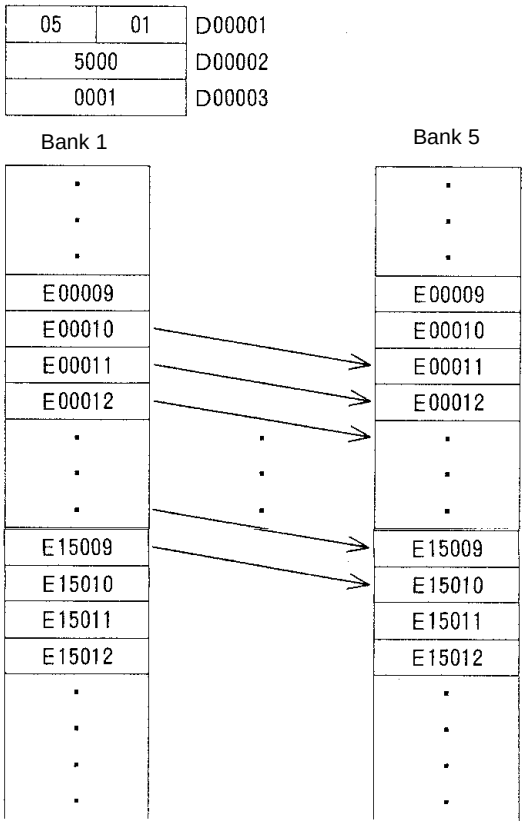
Content of a∗DM word is not BCD when set for BCD.

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.

Refer to page 99 for information explanations on comparison instructions supported by version-2 CVM1 CPUs.

5-16-1 COMPARE: CMP(020)

Ladder Symbol

(020)
CMP Cp_1 Cp_2

Operand Data Areas

Cp_1: 1st compare word CIO, G, A, T, C, #, DM, DR, IR

Cp_2: 2nd compare word CIO, G, A, T, C, #, DM, DR, IR

Variations

! CMP(020)

Description

When the execution condition is OFF, CMP(020) is not executed. When the execution condition is ON, CMP(020) compares Cp_1 and Cp_2 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_1 and Cp_2 are refreshed just before execution.

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.

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 115 for general precautions on operand data areas.

Flags

ER (A50003): Content of ∗DM word is not BCD when set for BCD.

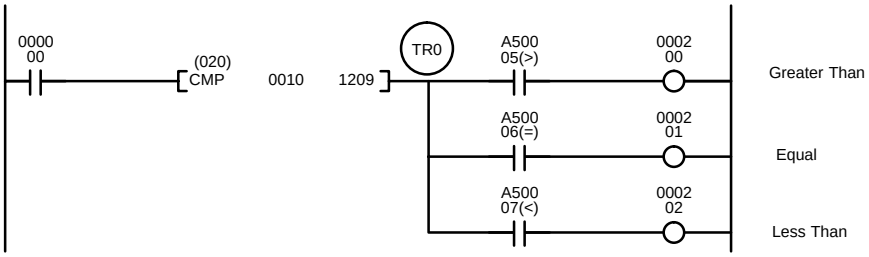
GR (A50005): ON if Cp_1 is greater than Cp_2.

EQ (A50006): ON if Cp_1 equals Cp_2.

LE (A50007): ON if Cp_1 is less than Cp_2.

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.

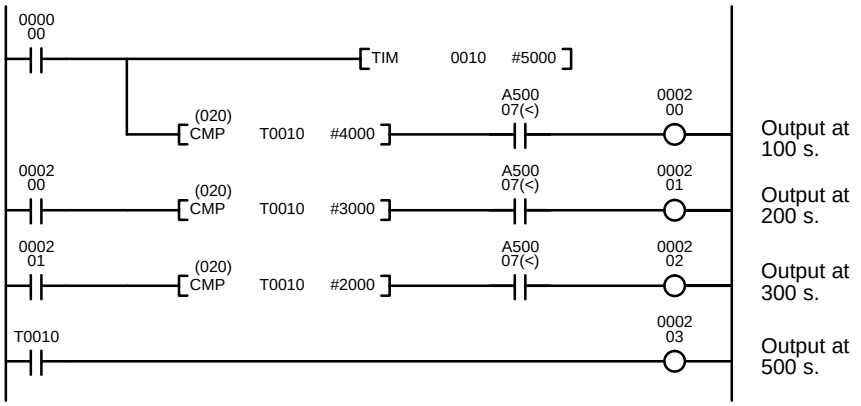


Address	Instruction	Operands
00000	LD	000000
00001	CMP(020)	
		0010
		1209
00002	OUT	TR0
00003	AND	A50005
00004	OUT	000200
00005	LD	TR0
00006	AND	A50006
00007	OUT	000201
00008	LD	TR0
00009	AND	A50007
00010	OUT	000202

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.

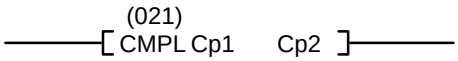
The branching structure of this diagram is important in order to ensure that 000200, 000201, and 000202 are controlled properly as the timer counts down. Because all of the comparisons here use the timer's PV as reference, the other operand for each CMP(020) must be in 4-digit BCD.



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

Cp1: 1st compare word CIO, G, A, T, C, #, DM

Cp2: 2nd compare word CIO, G, A, T, C, #, DM

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.

For version-2 CVM1 CPUs models onwards, non-intermediate instructions CMP(028) and CMPL(029) are standard.

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.

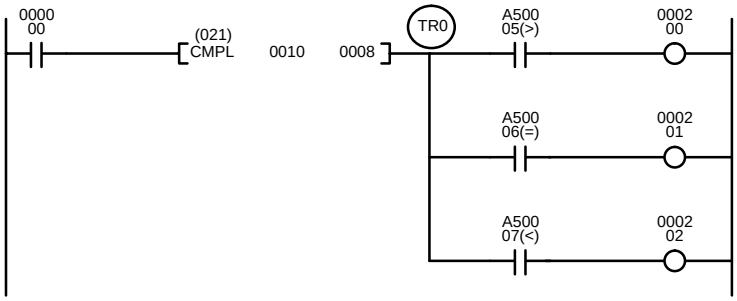
Note Refer to page 115 for general precautions on operand data areas.

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

(022)
BCMP S CB R

Variations

↑ BCMP(022)

Operand Data Areas

S: Source data	CIO, G, A, T, C, #, DM, DR, IR
CB: 1st block word	CIO, G, A, T, C, DM
R: Result word	CIO, G, A, T, C, DM, DR, IR

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 115 for general precautions on operand data areas.

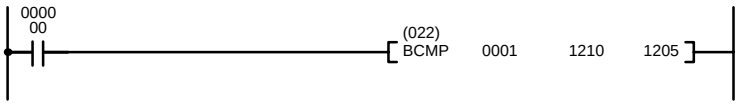
Flags

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

0001	0210

Compare data in 0001 (which contains 0210) with the given ranges.

Lower limits

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

Upper limits

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

R: 1205

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)

Ladder Symbol

(023)
TCMP S TB R

Variations

↑ TCMP(023)

Operand Data Areas

S: Source data	CIO, G, A, T, C, #, DM, DR, IR
TB: 1st table word	CIO, G, A, T, C, DM
R: Result word	CIO, G, A, T, C, DM, DR, IR

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 115 for general precautions on operand data areas.

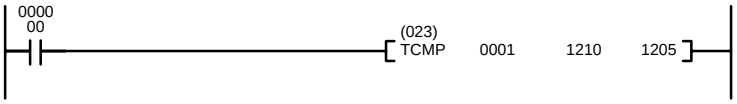
Flags

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

R: 1205

1210	0100	120500	0
1211	0200	120501	0
1212	0210	120502	1
1213	0400	120503	0
1214	0500	120504	0
1215	0600	120505	0
1216	0210	120506	1
1217	0800	120507	0
1218	0900	120508	0
1219	1000	120509	0
1220	0210	120510	1
1221	1200	120511	0
1222	1300	120512	0
1223	1400	120513	0
1224	0210	120514	1
1225	1600	120515	0

5-16-5 MULTIPLE COMPARE: MCMP(024)

Ladder Symbol

(024)
MCMP TB_1 TB_2 R

Variations

↑ MCMP(024)

Operand Data Areas

TB_1: 1st table word CIO, G, A, T, C, DM

TB_2: 2nd table word CIO, G, A, T, C, DM

R: Result word CIO, G, A, DM, DR, IR

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_1 through TB_1+15 to the contents of the 16 words TB_2 through TB_2+15, and turns ON the corresponding bit in word R when the contents are not equal. The content of TB_1 is compared to the content of TB_2, the content of TB_1+1 to the content of TB_2+1, ..., and the content of TB_1+15 to the content of TB_2+15. If the content of TB_1+n is equal to the content of TB_2+n, bit n of R is turned ON, if the contents are not equal, bit n of R is turned OFF.

Precautions

TB_1 and TB_2 cannot be one of the last 15 words in a data area because they designate the first of 16 words.

Note Refer to page 115 for general precautions on operand data areas.

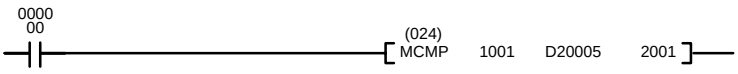
Flags

ER (A50003): Content of ∗DM word is not BCD when set for BCD.

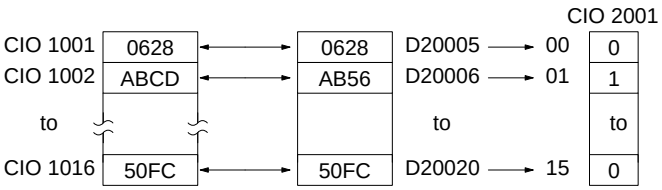
EQ (A50006): Content of R is zero after execution (i.e., if the contents of TB_1 through TB_1+15 and TB_2 through TB_2+15 are identical)

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)

Ladder Symbol

(025)
EQU Cp_1 Cp_2

Variations

↑ EQU(025)

Operand Data Areas

Cp_1: 1st compare word CIO, G, A, T, C, #, DM, DR, IR

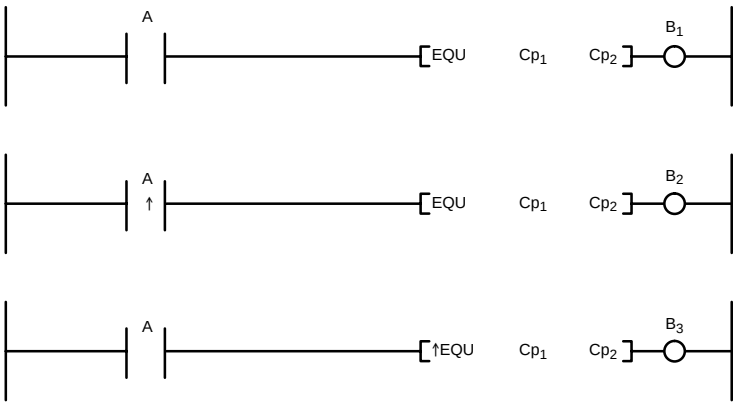
Cp_2: 2nd compare word CIO, G, A, T, C, #, DM, DR, IR

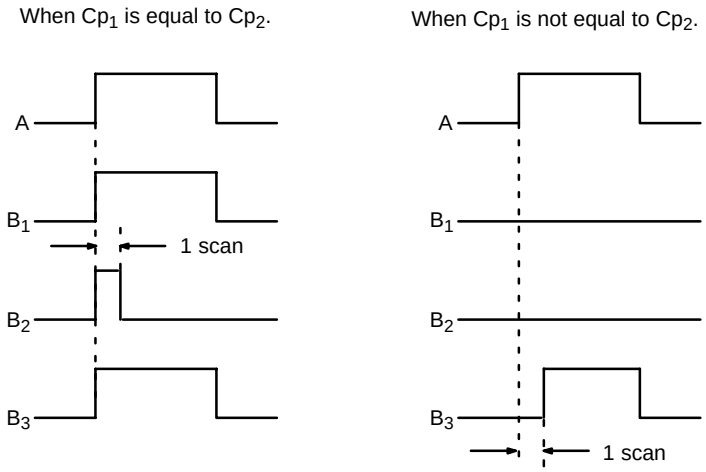
Description

When the execution condition is OFF, EQU(025) is not executed. When the execution condition is ON, EQU(025) compares the content of Cp_1 to the content of Cp_2 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_1 is equal to Cp_2 and will turn OFF for one scan only when Cp_1 is not equal to Cp_2.



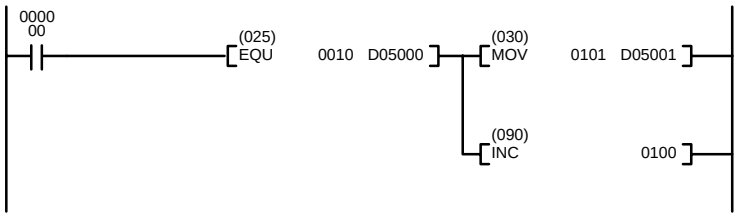


Flags

ER (A50003): Content of ∗DM word is not BCD when set for BCD.

Example

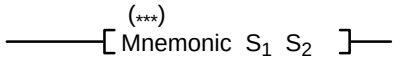
When CIO 000000 is ON in the following example, the content of CIO 0010 is compared to the content of D05000 and MOV(030) and INC(090) are executed only if the contents are the same.



Address	Instruction	Operands
00000	LD	000000
00001	EQU(025)	
		0010
		D05000
00002	MOV(030)	
		0101
		D05001
00003	INC(090)	
		0100

5-16-7 Input Comparison Instructions (300 to 328) (CVM1 V2)

Ladder Symbol



Operand Data Areas

S_1: Comparison data 1	CIO, G, A, T, C, #, DM, DR, IR
S_2: Comparison data 2	CIO, G, A, T, C, #, DM, DR, IR

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.

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 $S_1 = S_2$
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 115 for general precautions on operand data areas.

Example

< (310)

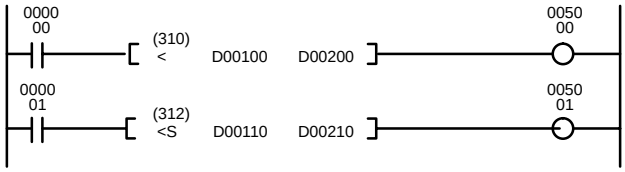
When CIO 000000 is ON in the following example, the contents of D00100 and D00200 are compared in as binary data. If the contents of D00100 is less than that of D00200, CIO 005000 is turned ON and execution proceeds to the next line. If the content of D00100 is not less than that of D00200, the remainder 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_1: D00100 — 8714 — Decimal: 34580

S_2: D00200 — 3A1C — Decimal: 14876

34580 > 14876

(Will not proceed to next line.)

Comparison with sign (<S)

S_1: D00110 — 8714 — Decimal: -30956

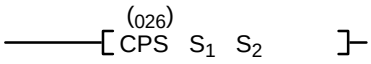
S_2: D00210 — 3A1C — Decimal: 14876

-30956 > 14876

(Will proceed to next line.)

5-16-8 SIGNED BINARY COMPARE: CPS(026) (CVM1 V2)

Ladder Symbol



Operand Data Areas

S_1: Comparison word 1	CIO, G, A, T, C, #, DM, DR, IR
S_2: Comparison word 2	CIO, G, A, T, C, #, DM, DR, IR

Variations

! CPS(026)

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)
$S_1 > S_2$	**ON**	OFF	OFF
$S_1 = S_2$	OFF	**ON**	OFF
$S_1 < S_2$	OFF	OFF	**ON**

The range that can be specified for comparison is 8000 to 7FFF (i.e., -32,768 to 32,767 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.)

5-16-9 DOUBLE SIGNED BINARY COMPARE: CPSL(027) (CVM1 V2)

Ladder Symbol

(027)
CPSL S_1 S_2

Operand Data Areas

S_1: First comparison word 1	CIO, G, A, T, C, #, DM,
S_2: First comparison word 2	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_1 and S_1+1 is compared to that of S_2 and S_2+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_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**

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

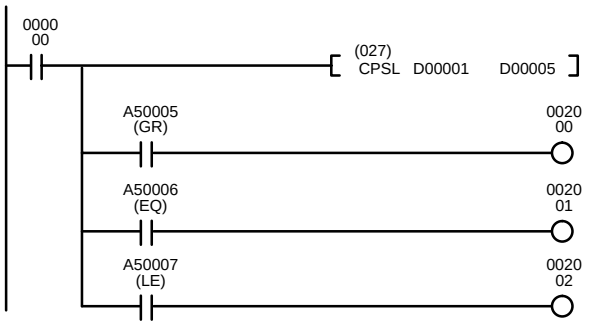
ER (A50003): Content of ∗DM word is not BCD when set for BCD.

GR (A40213), EQ (A50006), LE (A50007): (Refer to tables above.)

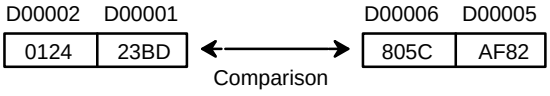
Example

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	00201
00006	AND	A50007
00007	OUT	002002

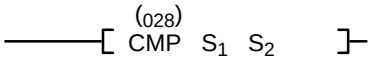


Comparison Results

A50005 (GR)	**ON**
A50006 (EQ)	OFF
A50007 (LE)	OFF

5-16-10 UNSIGNED COMPARE: CMP(028) (CVM1 V2)

Ladder Symbol



Operand Data Areas

S_1: Comparison word 1	CIO, G, A, T, C, #, DM, DR, IR
S_2: Comparison word 2	CIO, G, A, T, C, #, DM, DR, IR

Variations

! CMP(028)

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**

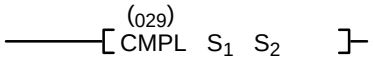
Flags

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) (CVM1 V2)

Ladder Symbol



Operand Data Areas

S_1: First comparison word 1	CIO, G, A, T, C, #, DM,
S_2: First comparison word 2	CIO, G, A, T, C, #, DM,

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_1 and S_1+1 is compared to that of S_2 and S_2+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.

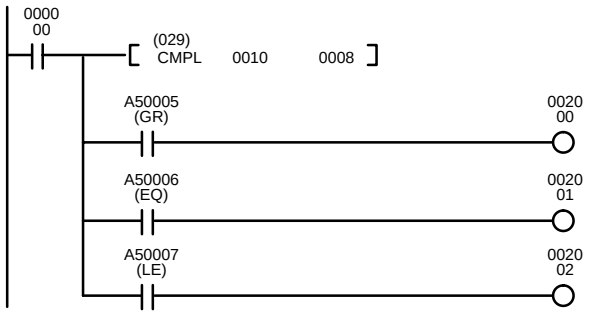
GR (A40213), EQ (A50006), LE (A50007): (Refer to tables above.)

Example

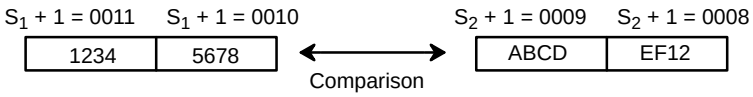
When CIO 000000 is ON in the following example, the content of CIO 0011 and CIO 0010 is compared with that of CIO 0009 and CIO 0008 as binary data.

- 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	00201
00006	AND	A50007
00007	OUT	002002



Comparison Results

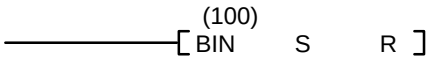
A50005 (GR)	OFF
A50006 (EQ)	OFF
A50007 (LE)	**ON**

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)

Ladder Symbol



Operand Data Areas

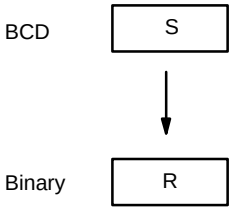
S: Source word	CIO, G, A, T, C, DM, DR, IR
R: Result word	CIO, G, A, DM, DR, IR

Variations

↑ BIN(100)

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 Peripheral 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.

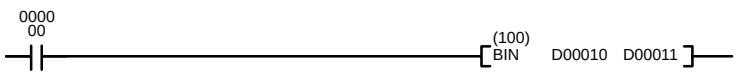
Note Refer to page 115 for general precautions on operand data areas.

Flags

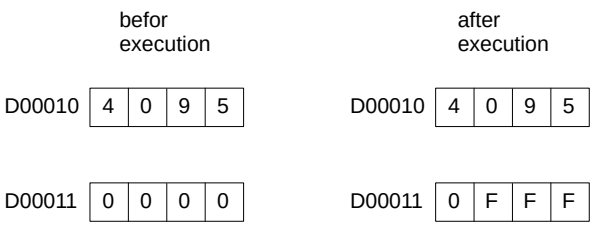
ER (A50003): S or content of ∗DM word is not BCD when set for BCD.
EQ (A50006): 0 has been placed in R
N (A50008): OFF

Example

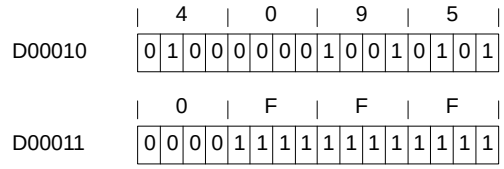
When CIO 000000 is ON in the following example, the content of D00010 is converted from BCD to binary and stored into D00011. The content of D00010 is left unchanged.



Address	Instruction	Operands
00000	LD	000000
00001	BIN(100)	
		D00010
		D00011

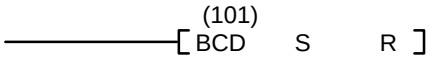


The bit contents of words D00010 and D00011 after execution are:



5-17-2 BINARY-TO-BCD: BCD(101)

Ladder Symbol



Variations

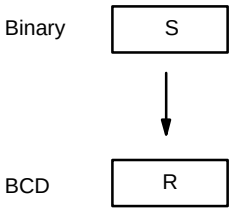
↑ BCD(101)

Operand Data Areas

S: Source word	CIO, G, A, T, C, DM, DR, IR
R: Result word	CIO, G, A, DM, DR, IR

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 Peripheral 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 115 for general precautions on operand data areas.

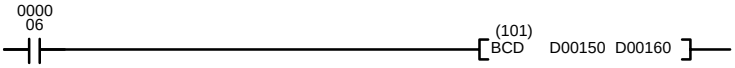
Flags

ER (A50003): Content of ∗DM word is not BCD when set for BCD.
Content of S exceeds 270F.

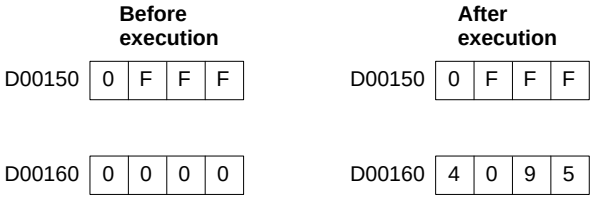
EQ (A50006): 0 has been placed in R

Example

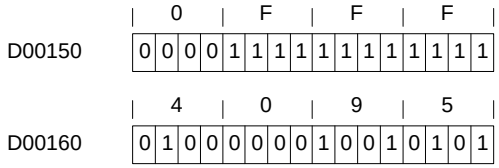
When CIO 000006 is ON in the following example, the content of word D00150 is converted from binary to BCD and stored in D00160. The content of D00150 is left unchanged.



Address	Instruction	Operands
00000	LD	000006
00001	BCD(101)	
		D00150
		D00160

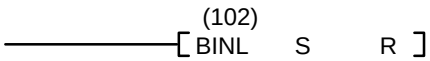


The bit contents of words D00150 and D00160 after execution are:



5-17-3 DOUBLE BCD-TO-DOUBLE BINARY: BINL(102)

Ladder Symbol



Variations

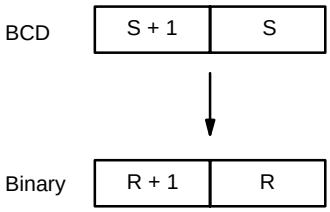
↑ BINL(102)

Operand Data Areas

S: Source word	CIO, G, A, T, C, DM
R: Result word	CIO, G, A, DM

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.

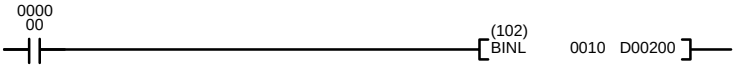
Note Refer to page 115 for general precautions on operand data areas.

Flags

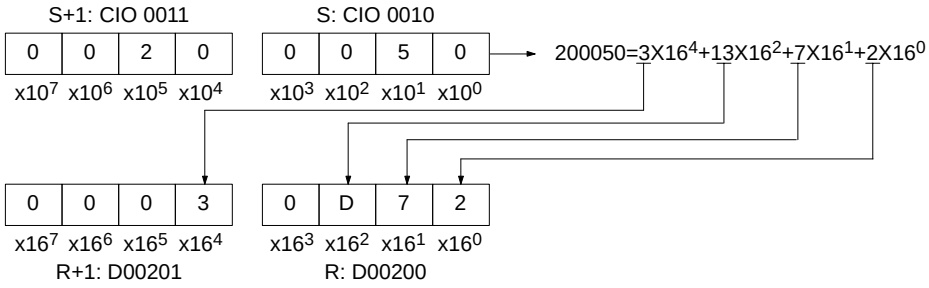
ER (A50003): Content of S or S+1 is not BCD.
Content of *DM word is not BCD when set for BCD.
EQ (A50006): 0 has been placed in R.
N (A50008): OFF

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.

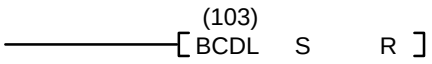


Address	Instruction	Operands
00000	LD	000000
00001	BINL(102)	
		0010
		D00200



5-17-4 DOUBLE BINARY-TO-DOUBLE BCD: BCDL(103)

Ladder Symbol



Variations

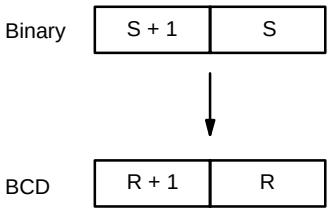
↑ BCDL(103)

Operand Data Areas

S: Source word	CIO, G, A, T, C, DM
R: Result word	CIO, G, A, DM

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 115 for general precautions on operand data areas.

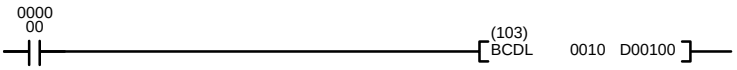
Flags

ER (A50003): Content of ∗DM word is not BCD when set for BCD.
Content of S and S+1 exceeds 05F5E0FF.

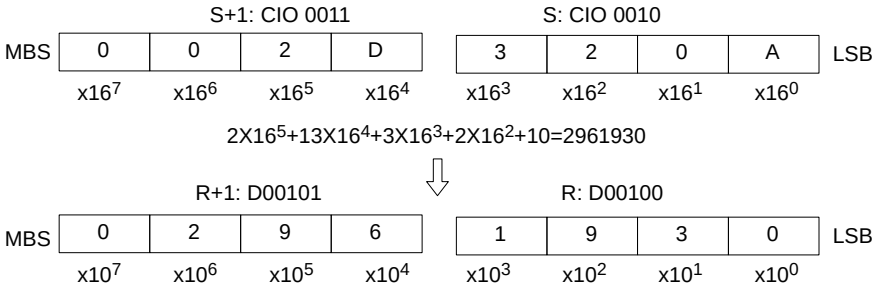
EQ (A50006): 0 has been placed in R

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)

Ladder Symbol

(104)
NEG S R

Operand Data Areas

S: Source word	CIO, G, A, T, C, #, DM, DR, IR
R: Result word	CIO, G, A, DM, DR, IR

Variations

↑ NEG(104)

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 115 for general precautions on operand data areas.

Flags

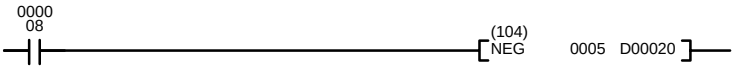
ER (A50003): Content of ∗DM word is not BCD when set for BCD.

EQ (A50006): Content of S is 0 (the content of R will also be 0 after execution)

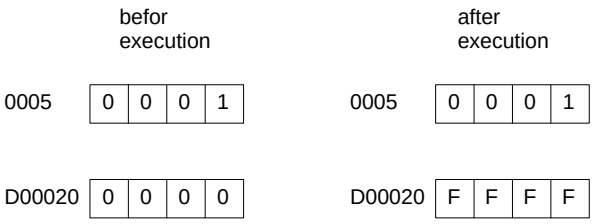
N (A50008): Shows the status of bit 15 of R after execution.

Example

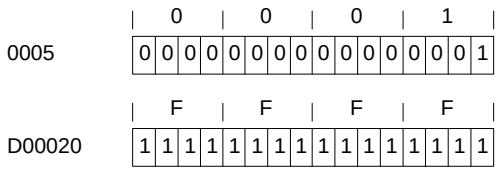
When CIO 000008 is ON in the following example, the 4-digit hexadecimal content of CIO 0005 is converted to its 2's complement equivalent and stored in D00020.



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)

Ladder Symbol

(105)
NEGL S R

Operand Data Areas

S: 1st source word CIO, G, A, T, C, #, DM

R: 1st result word CIO, G, A, DM

Variations

↑ NEGL(105)

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 115 for general precautions on operand data areas.

Flags

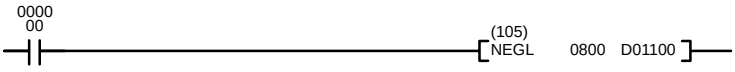
ER (A50003): Content of ∗DM word is not BCD when set for BCD.

EQ (A50006): Content of S and S+1 is 0 (the content of R and R+1 will also be 0 after execution)

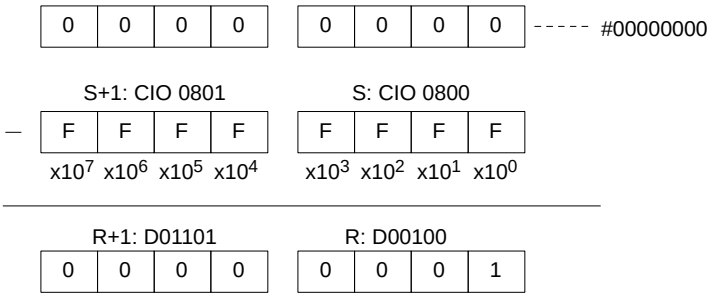
N (A50008): Shows the status of bit 15 of R+1 after execution.

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.

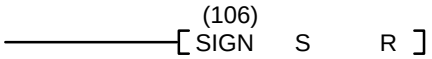


Address	Instruction	Operands
00000	LD	000000
00001	NEGL(105)	
		0800
		D01100



5-17-7 SIGN: SIGN(106)

Ladder Symbol



Variations

↑ SIGN(106)

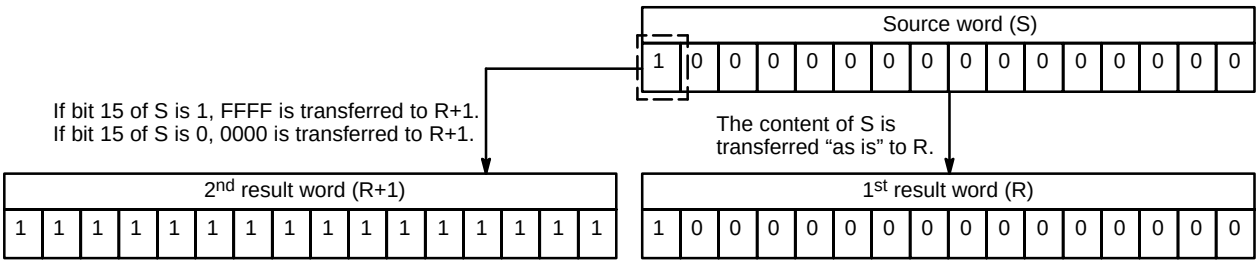
Operand Data Areas

S: Source word	CIO, G, A, T, C, #, DM, DR, IR
R: 1st result word	CIO, G, A, DM

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 115 for general precautions on operand data areas.

Flags

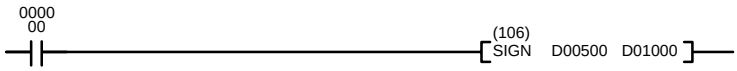
ER (A50003): Content of ∗DM word is not BCD when set for BCD.

EQ (A50006): Content of R and R+1 is 0 after execution

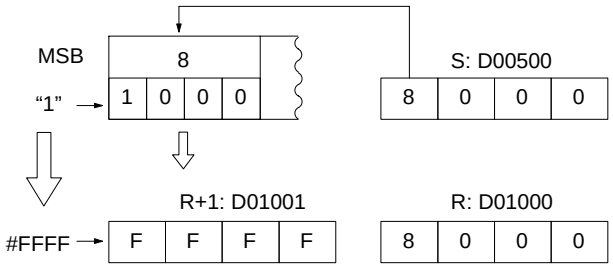
N (A50008): Content of R+1 is FFFF after execution.

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.

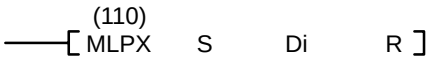


Address	Instruction	Operands
00000	LD	000000
00001	SIGN(106)	
		D00500
		D01000



5-17-8 DATA DECODER: MLPX(110)

Ladder Symbol



Variations

↑ MLPX(110)

Operand Data Areas

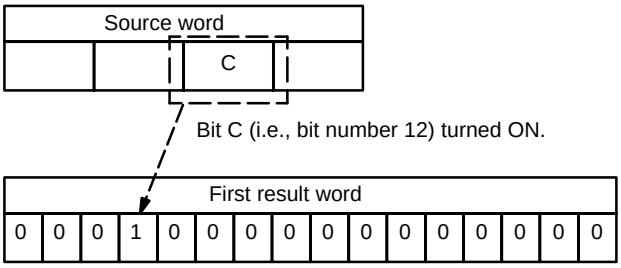
S: Source word	CIO, G, A, T, C, DM, DR, IR
Di: Digit designator	CIO, G, A, T, C, #, DM, DR, IR
R: 1st result word	CIO, G, A, DM,

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.

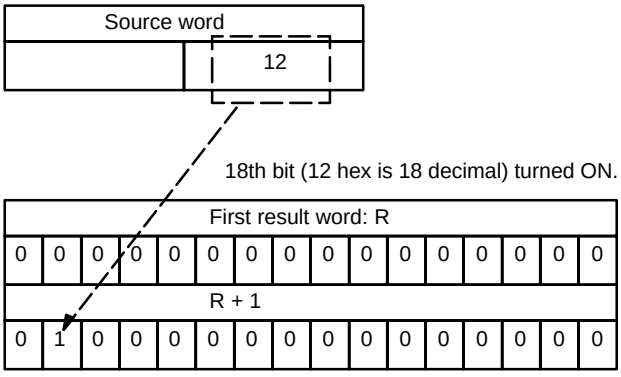
For 4-bit conversion, MLPX(110) converts up to four 4-bit hexadecimal digits from S into decimal values from 0 to 15, each of which is used to indicate a bit position. The bit whose number corresponds to each converted value is then turned ON in a result word. If more than one digit is specified, then 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.)

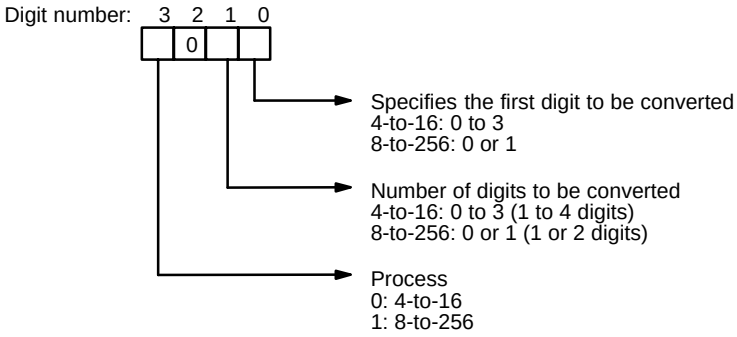
The following is an example of a one-digit decode operation from digit number 1 of S, i.e., here Di would be 1001.



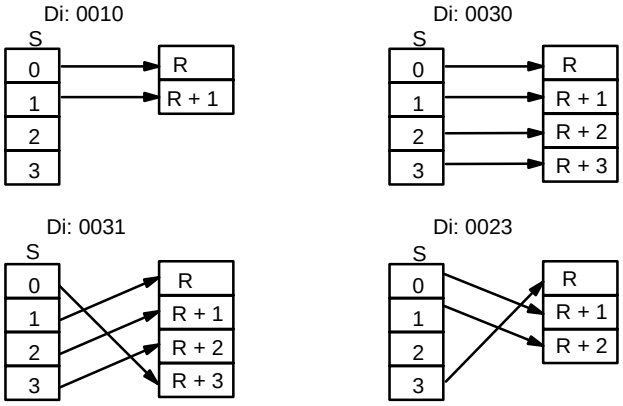
The first digit and the number of digits to be converted are designated in Di. If more digits are designated than remain in S (counting from the designated first digit), the remaining digits will be taken starting back at the beginning of S.

Digit Designator

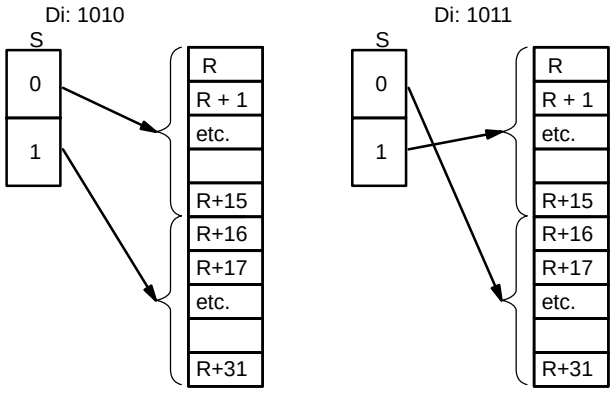
The digits of Di are set as shown below.



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.

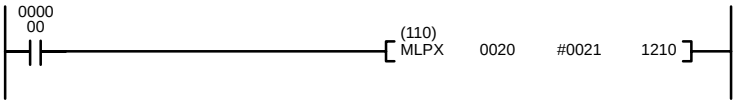
Note Refer to page 115 for general precautions on operand data areas.

Flags

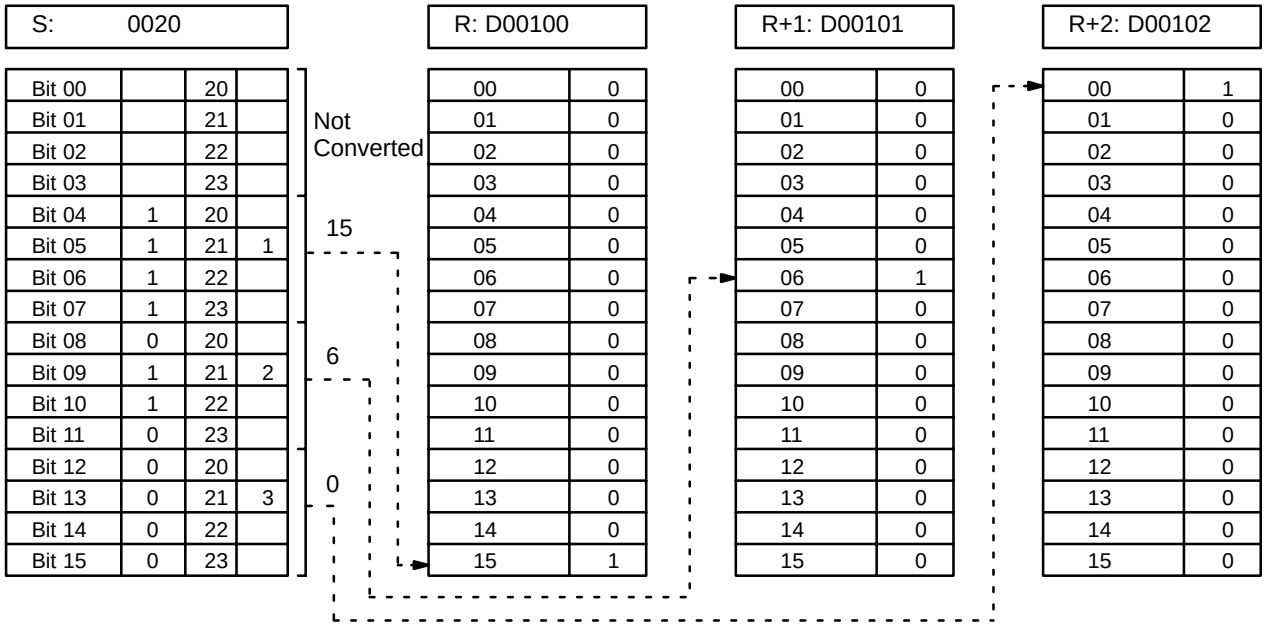
ER (A50003): Content of ∗DM word is not BCD when set for BCD.
Improper digit designator.

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



				S: 0020	R: D00100	R+1: D00101	R+2: D00102
Bit 00		20			00 0	00 0	00 1
Bit 01		21			01 0	01 0	01 0
Bit 02		22			02 0	02 0	02 0
Bit 03		23			03 0	03 0	03 0
Bit 04	1	20			04 0	04 0	04 0
Bit 05	1	21	1		05 0	05 0	05 0
Bit 06	1	22			06 0	06 1	06 0
Bit 07	1	23			07 0	07 0	07 0
Bit 08	0	20			08 0	08 0	08 0
Bit 09	1	21	2		09 0	09 0	09 0
Bit 10	1	22			10 0	10 0	10 0
Bit 11	0	23			11 0	11 0	11 0
Bit 12	0	20			12 0	12 0	12 0
Bit 13	0	21	3		13 0	13 0	13 0
Bit 14	0	22			14 0	14 0	14 0
Bit 15	0	23			15 1	15 0	15 0

5-17-9 DATA ENCODER: DMPX(111)

Ladder Symbol

Variations

↑ DMPX(111)

Operand Data Areas

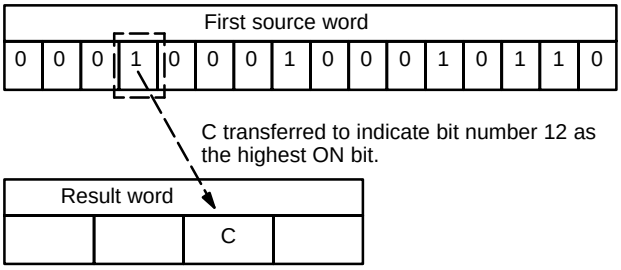
SB: 1st source word	CIO, G, A, T, C, DM
R: Result word	CIO, G, A, DM, DR, IR
Di: Digit designator	CIO, G, A, T, C, #, DM, DR, IR

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.

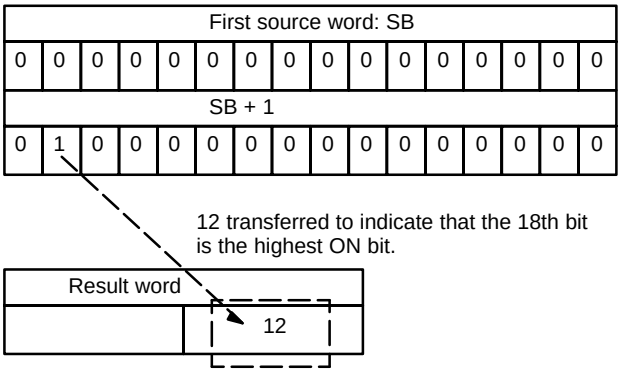
For 16-bit conversion, DMPX(111) determines the position of the highest ON bit in SB, encodes it into one-digit hexadecimal value corresponding to the bit number of the highest ON bit, then transfers the hexadecimal value to the specified 4-bit digit in R. The digits to receive the results are specified 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.

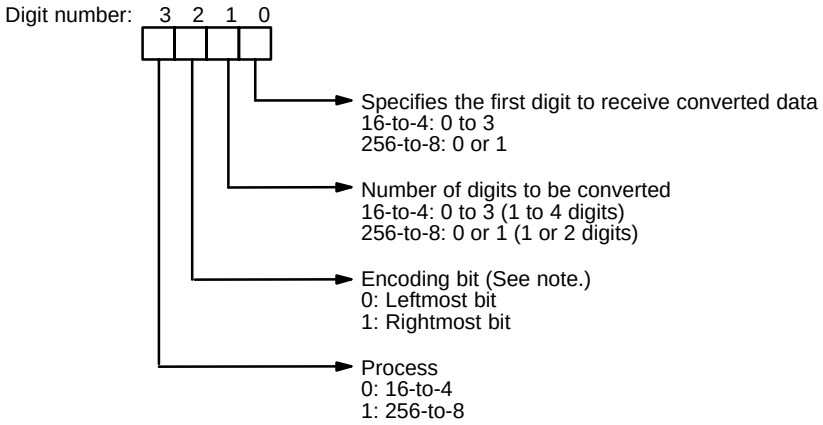


The number of digits (words) to be encoded and the first digit to receive converted data are specified in Di. If more digits are designated than remain in R (counting from the designated first digit), the remaining digits will be placed at digits starting back at the beginning of R.

The final word(s) to be converted must be in the same data area as SB.

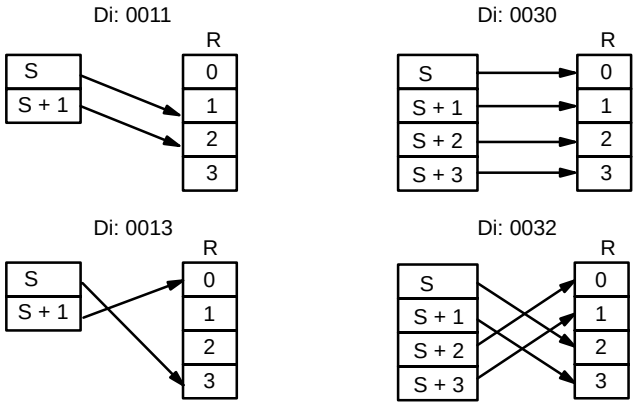
Digit Designator

The digits of Di are set as shown below.

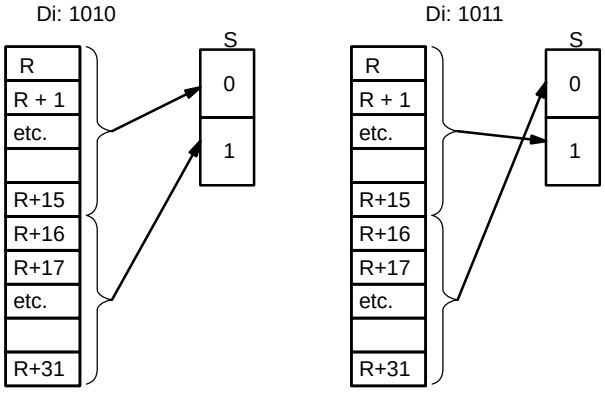


Note The encoding bit can be specified for version-2 CVM1 CPUs only. For earlier CPUs, only the leftmost bit can be encoded.

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.

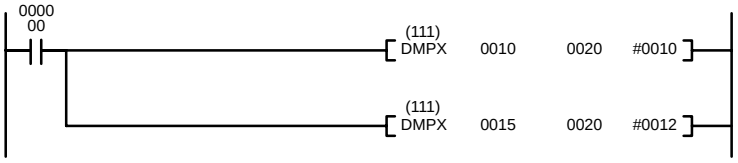
Note Refer to page 115 for general precautions on operand data areas.

Flags

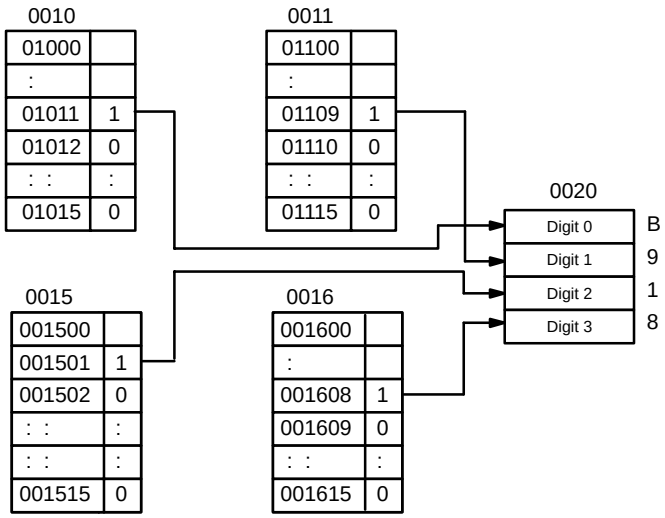
ER (A50003): Content of ∗DM word is not BCD when set for BCD.
Improper digit designator.
Content of a source word is 0.

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.

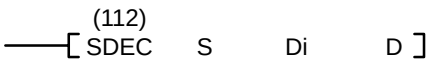


Address	Instruction	Operands
00000	LD	000000
00001	DMPX(111)	
		0010
		0020
		#0010
00002	DMPX(111)	
		0015
		0020
		#0012



5-17-10 7-SEGMENT DECODER: SDEC(112)

Ladder Symbol



Variations

↑ SDEC(112)

Operand Data Areas

S: Source word	CIO, G, A, T, C, DM, DR, IR
Di: Digit designator	CIO, G, A, T, C, #, DM, DR, IR
D: 1st destination word	CIO, G, A, DM

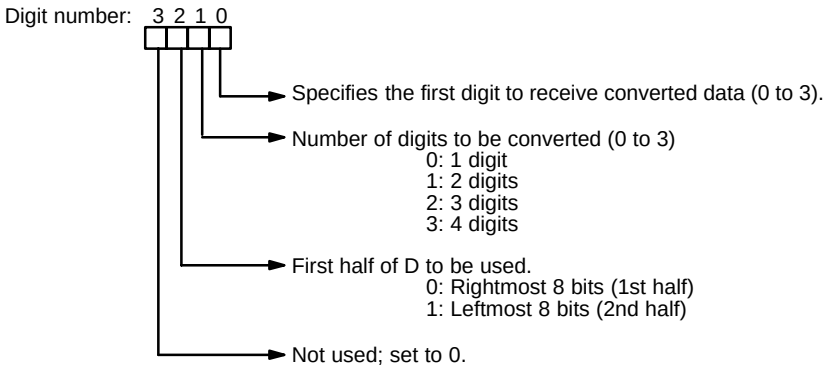
Description

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.

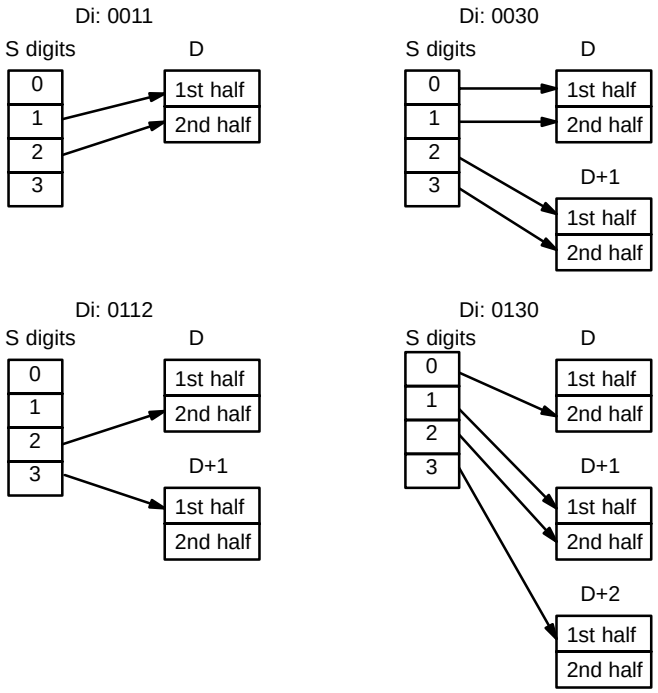
Any or all of the digits in S may be converted in sequence from the designated first digit. The first digit, the number of digits to be converted, and the half of D to receive the first 7-segment display code (rightmost or leftmost 8 bits) are designated in Di. If multiple digits are designated, they 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 115 for general precautions on operand data areas.

Flags

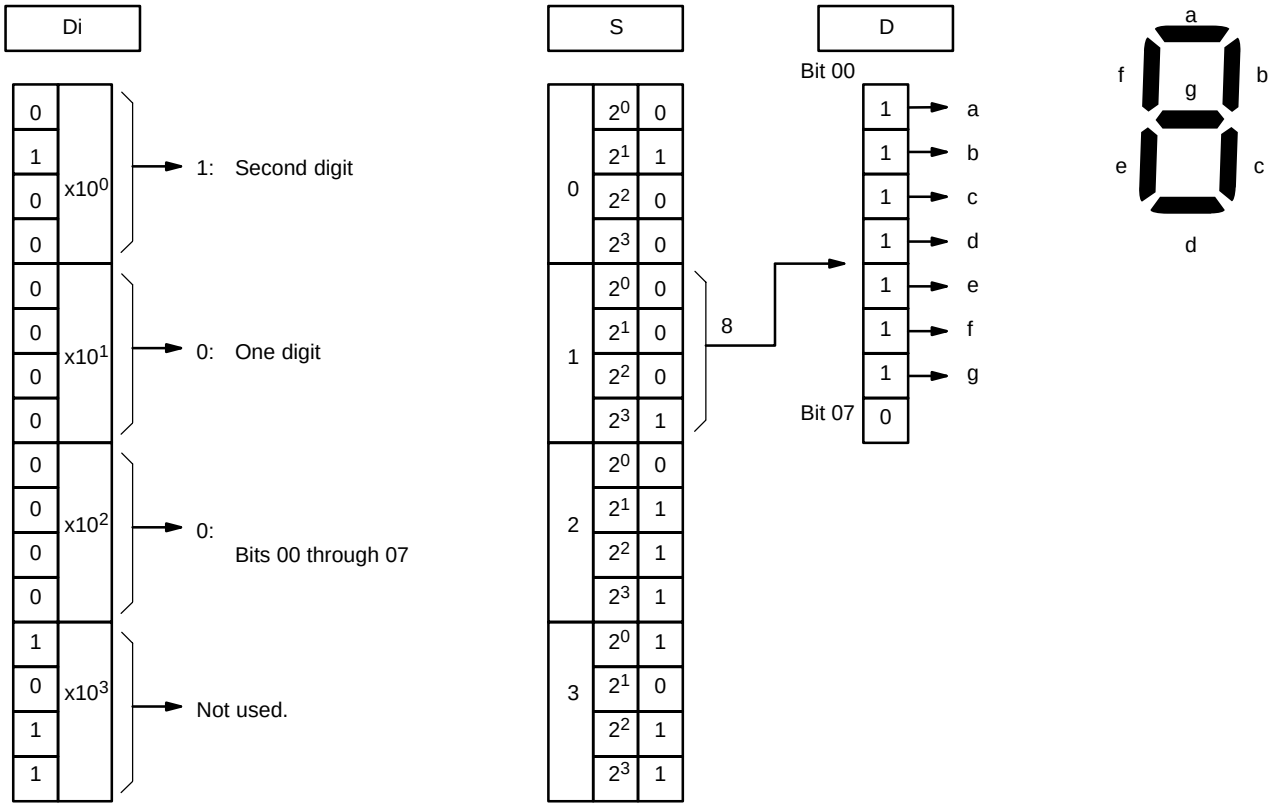
ER (A50003): Content of ∗DM word is not BCD when set for BCD.

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.

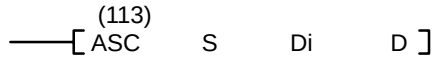
The table underneath shows the original data and converted code for all hexadecimal digits.



Original data					Converted code (segments)								Display
Digit	Bits				–	g	f	e	d	c	b	a	
0	0	0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	0	0	0	0	1	1	0	1
2	0	0	1	0	0	1	0	1	1	0	1	1	2
3	0	0	1	1	0	1	0	0	1	1	1	1	3
4	0	1	0	0	0	1	1	0	0	1	1	0	4
5	0	1	0	1	0	1	1	0	1	1	0	1	5
6	0	1	1	0	0	1	1	1	1	1	0	1	6
7	0	1	1	1	0	0	1	0	0	1	1	1	7
8	1	0	0	0	0	1	1	1	1	1	1	1	8
9	1	0	0	1	0	1	1	0	1	1	1	1	9
A	1	0	1	0	0	1	1	1	0	1	1	1	A
B	1	0	1	1	0	1	1	1	1	1	0	0	b
C	1	1	0	0	0	0	1	1	1	0	0	1	C
D	1	1	0	1	0	1	0	1	1	1	1	0	d
E	1	1	1	0	0	1	1	1	1	0	0	1	E
F	1	1	1	1	0	1	1	1	0	0	0	1	F

5-17-11 ASCII CONVERT: ASC(113)

Ladder Symbol



Variations

↑ ASC(113)

Operand Data Areas

S: Source word	CIO, G, A, T, C, DM, DR, IR
Di: Digit designator	CIO, G, A, T, C, #, DM, DR, IR
D: 1st destination word	CIO, G, A, DM

Description

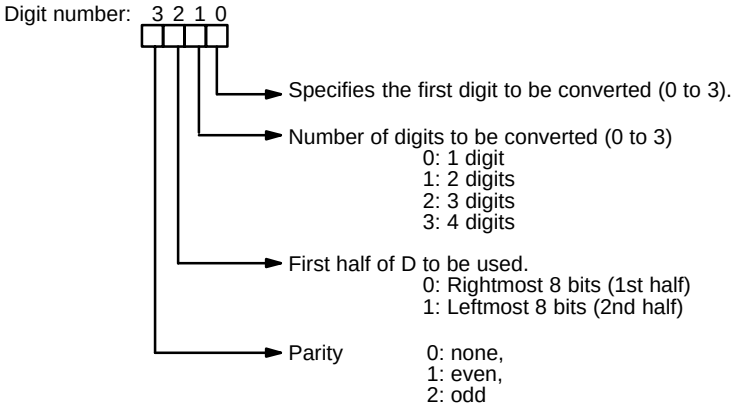
When the execution condition is OFF, ASC(113) is not executed. When the execution condition is ON, ASC(113) converts the designated digit(s) of S into the equivalent 8-bit ASCII code and places it into the destination word(s) beginning with D.

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.

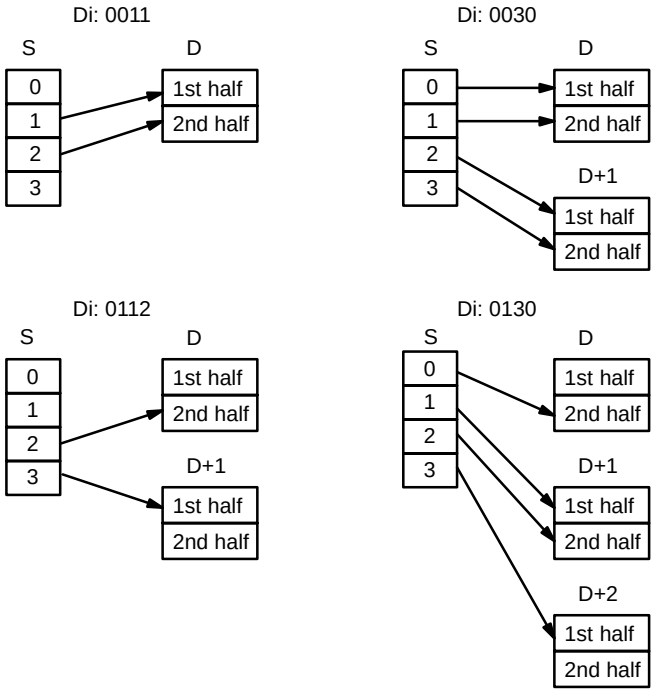
Refer to *Appendix H* for a table of extended ASCII characters.

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.

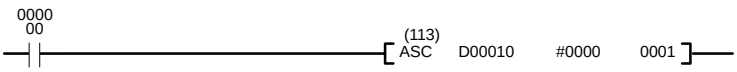
Note Refer to page 115 for general precautions on operand data areas.

Flags

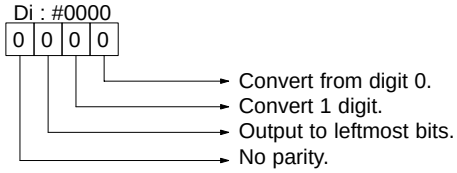
ER (A50003): Content of ∗DM word is not BCD when set for BCD.
Improper digit designator.

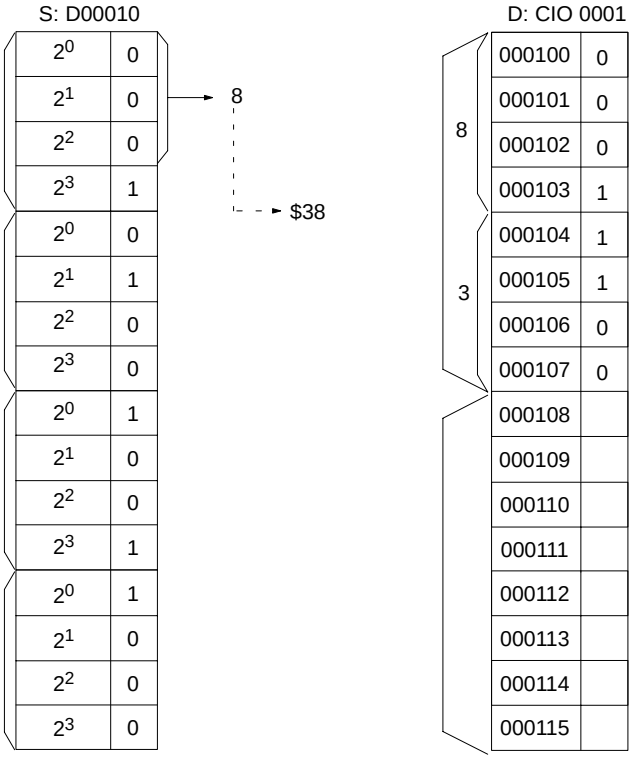
Example

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

(114)
BCNT N S R

Variations

↑ BCNT(114)

Operand Data Areas

N: Number of words	CIO, G, A, T, C, #, DM, DR, IR
S: 1st source word	CIO, G, A, T, C, DM
R: Result word	CIO, G, A, T, C, DM, DR, IR

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.

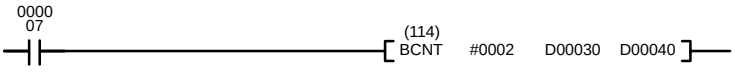
Note Refer to page 115 for general precautions on operand data areas.

Flags

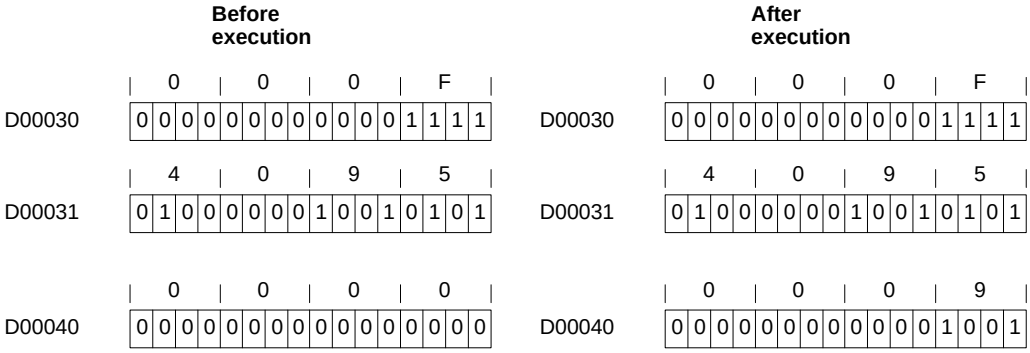
ER (A50003):	N is not BCD between 0001 and 9999.
	If the total exceeds 9999.
	Content of ∗DM word is not BCD when set for BCD.
EQ (A50006):	Result is 0.

Example

When CIO 000007 is ON in the following example, all ON bits in D00030 and D00031 are counted and the results is placed in D00040.

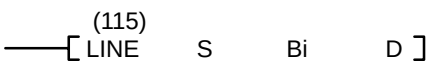


Address	Instruction	Operands
00000	LD	000007
00001	BCNT(114)	
		#0002
		D00030
		D00040



5-17-13 COLUMN TO LINE: LINE(115)

Ladder Symbol



Variations

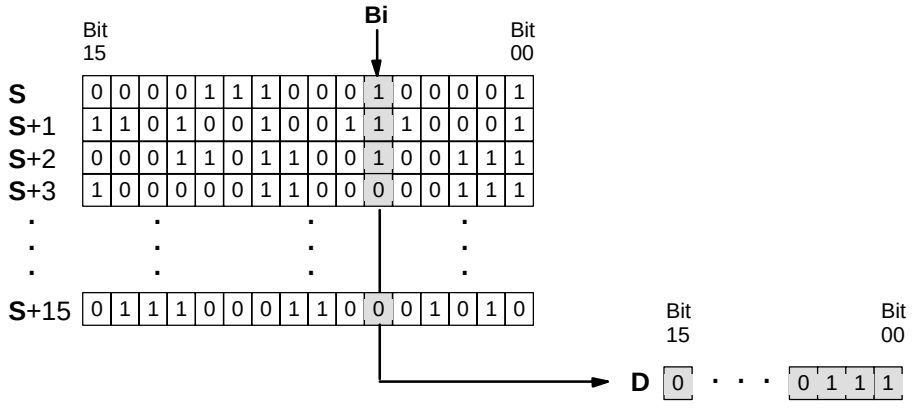
↑ LINE(115)

Operand Data Areas

S: 1st source word	CIO, G, A, T, C, DM
Bi: Bit designator	CIO, G, A, T, C, #, DM, DR, IR
D: Destination word	CIO, G, A, DM, DR, IR

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 115 for general precautions on operand data areas.

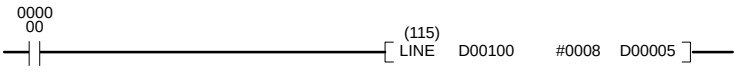
Flags

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

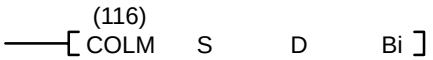
When CIO 000000 is ON in the following example, the status of bits number 08 in D00100 through D00115 are output in order to D00005, with the status of bit 08 in D00100 being output to bit 00 of D00005.



Address	Instruction	Operands
00000	LD	000000
00001	LINE(115)	
		D00100
		#0008
		D00005

5-17-14 LINE TO COLUMN: COLM(116)

Ladder Symbol



Variations

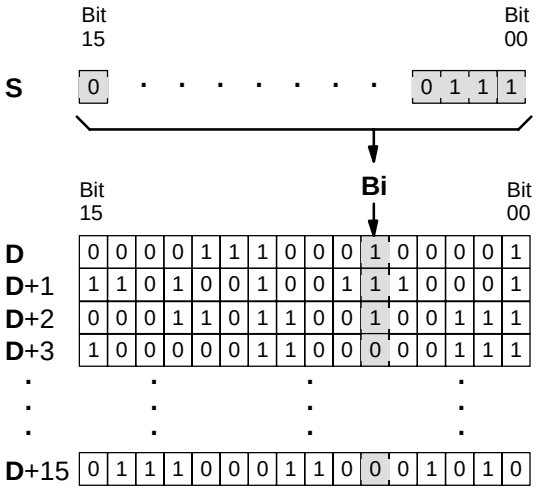
↑ COLM(116)

Operand Data Areas

S: Source word	CIO, G, A, T, C, #, DM, DR, IR
D: 1st destination word	CIO, G, A, DM
Bi: Bit designator	CIO, G, A, T, C, #(BCD), DM, DR, IR

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 115 for general precautions on operand data areas.

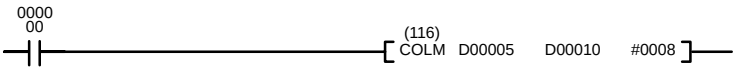
Flags

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

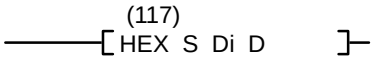
When CIO 000000 is ON in the following example, the status of bits 00 to 15 in D00005 are copied consecutively to bits number 08 of D00010 through D00025, with the status of bit 00 being transferred to bit 08 of D00010.



Address	Instruction	Operands
00000	LD	000000
00001	COLM(116)	
		D00005
		D00010
		#0008

5-17-15 ASCII TO HEX: HEX(117) (CVM1 V2)

Ladder Symbol



Variations

↑HEX(117)

Operand Data Areas

S: First source word	CIO, G, A, T, C, DM
Di: Digit designator	CIO, G, A, T, C, #, DM, DR, IR
D: Destination word	CIO, G, A, DM

Description

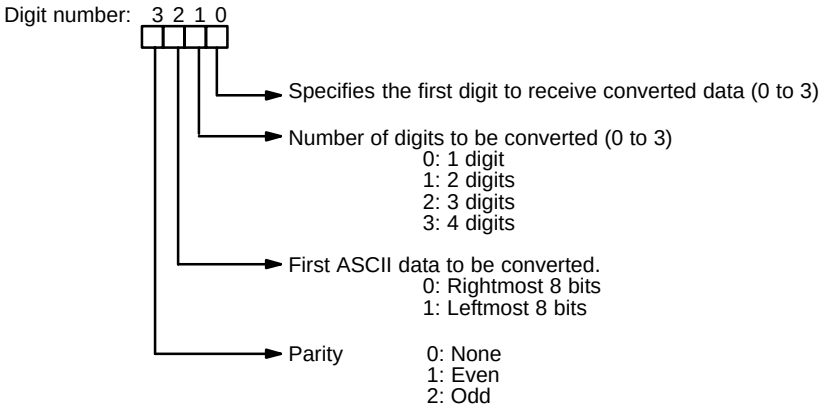
When the execution condition is OFF, HEX(117) is not executed. When the execution condition is ON, HEX(117) converts the data in specified source words from ASCII to hexadecimal data, and outputs the results to a specified destination word.

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).

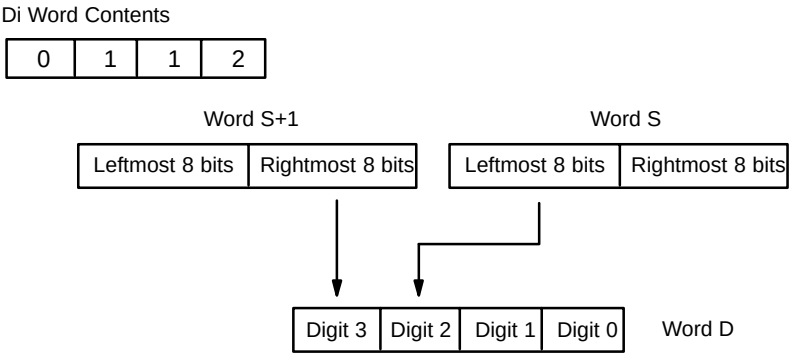
Digit Designator



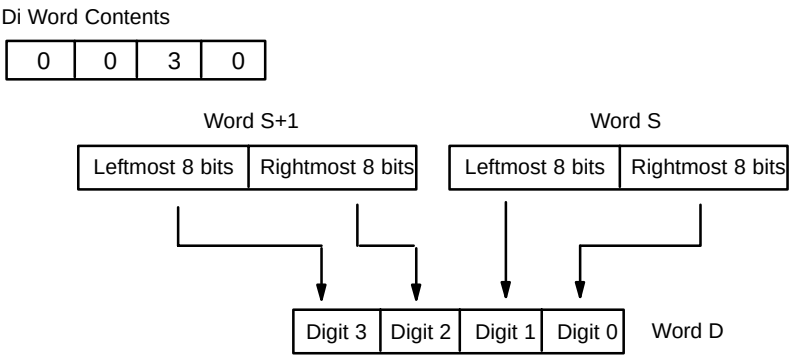
Data in the destination word (D) will not be changed except for the digits that are converted to hexadecimal.

Digit Designator Examples The following examples show the digit designators (Di) used to make various multiple-word conversions.

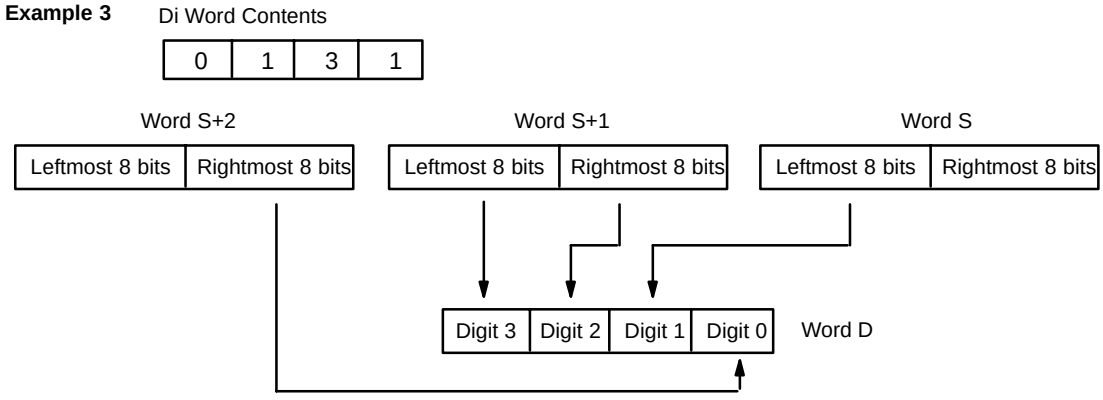
Example 1



Example 2



Example 3



Parity

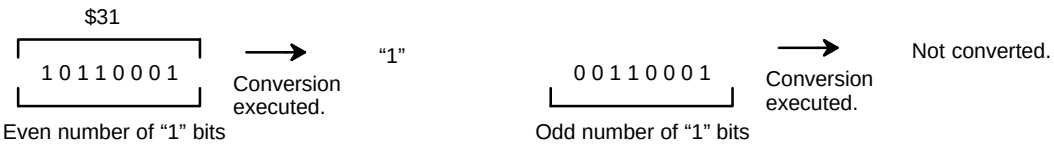
0: None

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.

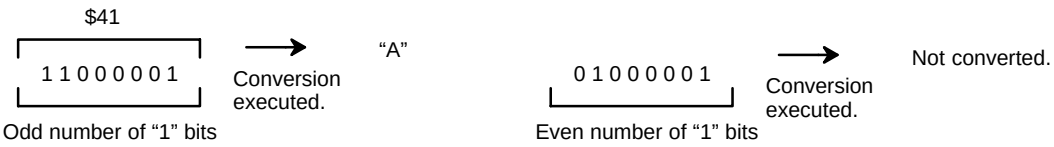
Example



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	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

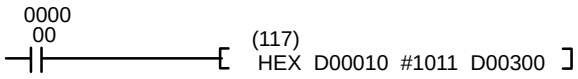
Refer to page 115 for general precautions on operand data areas.

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.

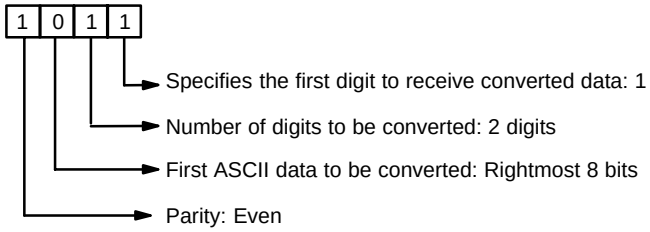
Programming Example

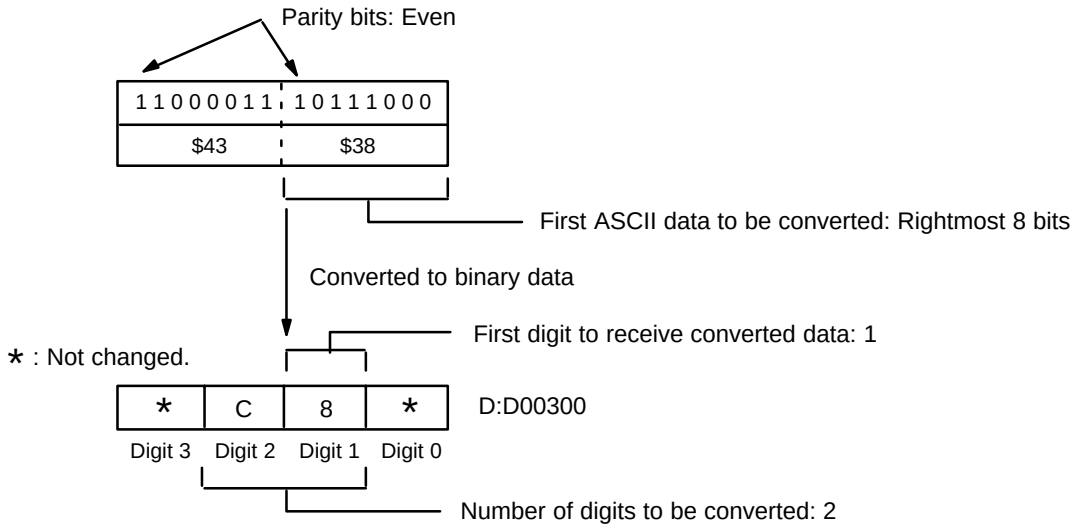
When CIO 000000 is ON in the following example, the ASCII data in D00010 is converted to binary data and then output to D00300.



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) (CVM1 V2)

Ladder Symbol

(275)
BINS C S D

Variations

↑BINS(275)

Operand Data Areas

C: Control word	CIO, G, A, T, C, #, DM, DR, IR
S: Source word	CIO, G, A, T, C, DM, DR, IR
D: Destination word	CIO, G, A, DM, DR, IR

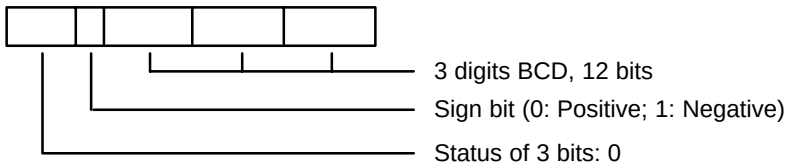
Description

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).

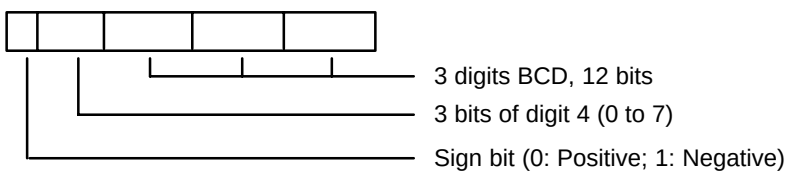
Note Special I/O Units sometimes output signed BCD data. Calculations using this data will normally be easier if it is first converted to signed binary data by means of BINS(275) or BISL(277).

The input data format and range designations for the various control word contents are as follows:

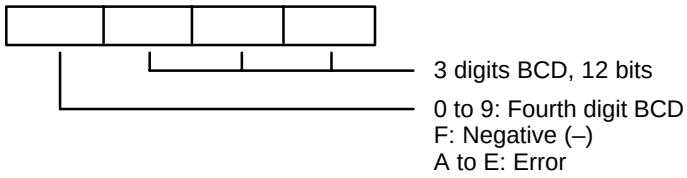
When C = 0000 (Input Data Range: –999 to 999 BCD)



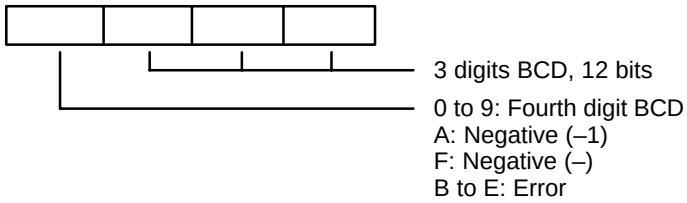
When C = 0001 (Input Data Range: –7999 to 7999 BCD)



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

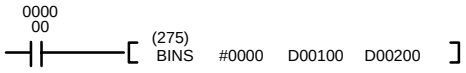
Refer to page 115 for general precautions on operand data areas.

Flags

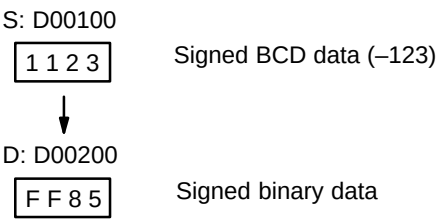
ER (A50003):	Data format is 0002, and leftmost digit is A to E.
	Data format is 0003, and the leftmost is B to E.
	Data to be converted is not BCD.
	Content of ∗DM word is not BCD when set for BCD.
EQ (A50006)	Content of the converted data is all zeroes.
N (A50008)	Converted number is negative.

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.

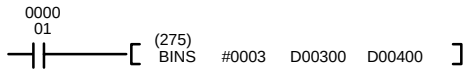


Address	Instruction	Operands
00000	LD	000000
00001	BINS(275)	
		#0000
		D00100
		D00200

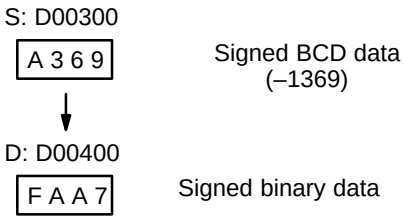


Example 2

When CIO 000001 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.

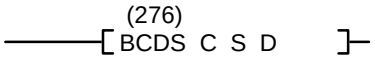


Address	Instruction	Operands
00000	LD	000001
00001	BINS(275)	
		#0003
		D00300
		D00400



5-17-17 SIGNED BINARY-TO-BCD: BCDS(276) (CVM1 V2)

Ladder Symbol



Variations

↑BCDS(276)

Operand Data Areas

C: Control word	CIO, G, A, T, C, #, DM, DR, IR
S: Source word	CIO, G, A, T, C, DM, DR, IR
D: Destination word	CIO, G, A, DM, DR, IR

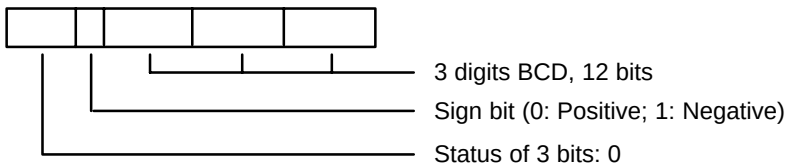
Description

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).

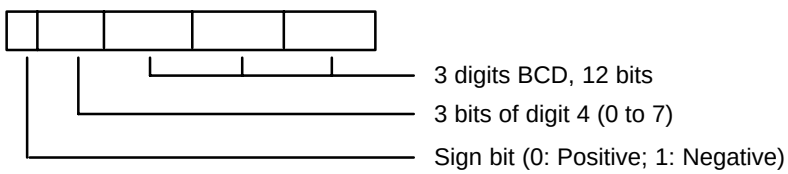
Note Special I/O Units sometimes require input of signed BCD data. BCDS(276) or BDSL(278) can be used to easily convert signed binary data to signed BCD data.

The output data format and range designations for the various control word contents are as follows:

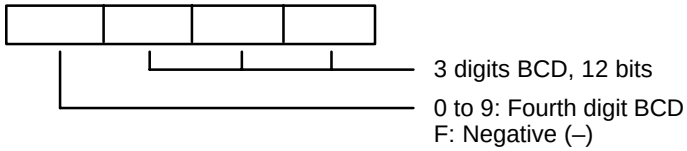
When C = 0000 (Output Data Range: –999 to 999 BCD)



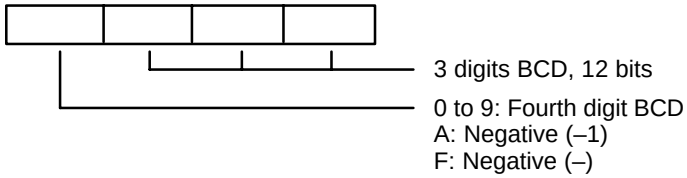
When C = 0001 (Output Data Range: –7999 to 7999 BCD)



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 115 for general precautions on operand data areas.

Flags

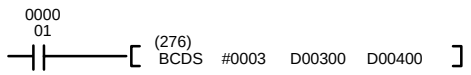
ER (A50003): Data is not within allowable range for data format.
Content of∗DM word is not BCD when set for BCD.

EQ (A50006) Content of the converted data is all zeroes.

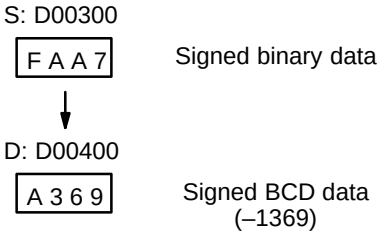
N (A50008) Data to be converted is a negative number.

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.

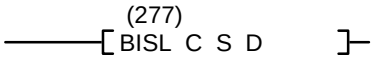


Address	Instruction	Operands
00000	LD	000001
00001	BCDS(276)	
		#0003
		D00300
		D00400



5-17-18 DOUBLE SIGNED BCD-TO-BINARY: BISL(277) (CVM1 V2)

Ladder Symbol



Variations

↑BISL(277)

Operand Data Areas

C: Control word	CIO, G, A, T, C, #, DM,
S: 1st source word	CIO, G, A, T, C, DM,
D: 1st destination word	CIO, G, A, DM,

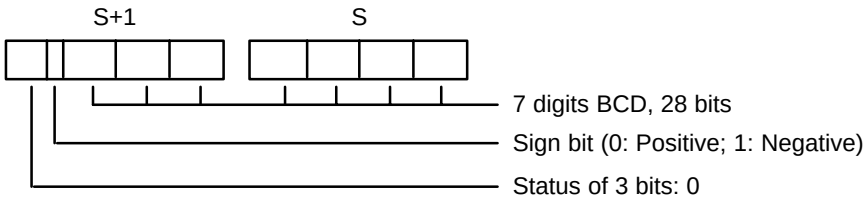
Description

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).

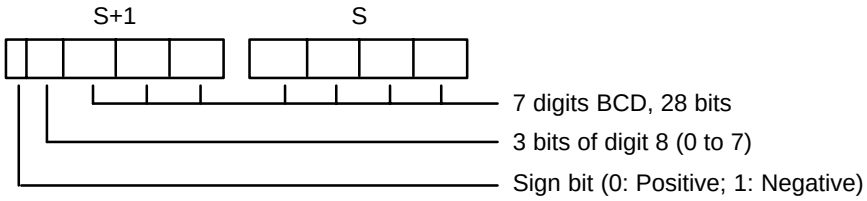
Note Special I/O Units sometimes output signed BCD data. Calculations using this data will normally be easier if it is first converted to signed binary data by means of BINS(275) or BISL(277).

The input data format and range designations for the various control word contents are as follows:.

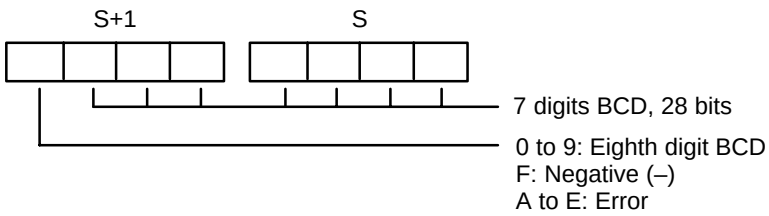
When C = 0000 (Input Data Range: –999 9999 to 999 9999 BCD)



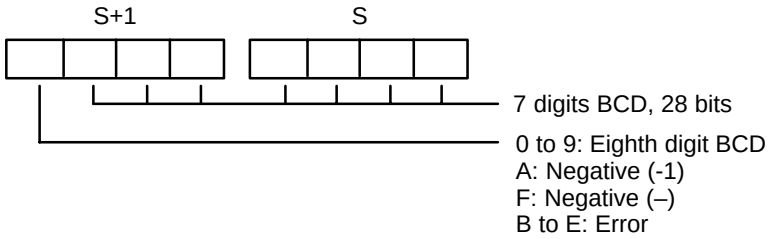
When C = 0001 (Input Data Range: –7999 9999 to 7999 9999 BCD)



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

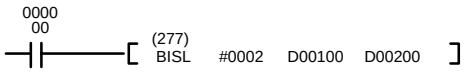
Refer to page 115 for general precautions on operand data areas.

Flags

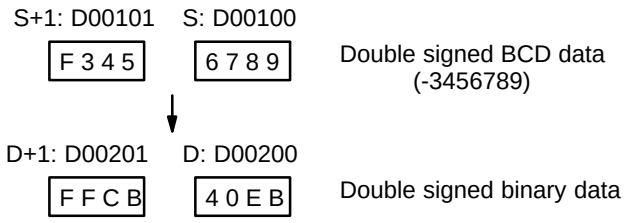
ER (A50003):	Data format is 0002, and the leftmost digit is A to E
	Data format is 0003, and the leftmost digit is B to E.
	Data to be converted is not BCD.
	Content of ∗DM word is not BCD when set for BCD.
EQ (A50006)	Content of the converted data is all zeroes.
N (A50008)	Converted number is negative.

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.

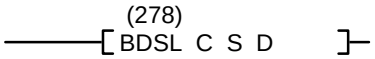


Address	Instruction	Operands
00000	LD	000000
00001	BISL(277)	
		#0002
		D00100
		D00200



5-17-19 DOUBLE SIGNED BINARY-TO-BCD: BDSL(278) (CVM1 V2)

Ladder Symbol



Variations

↑BDSL(278)

Operand Data Areas

C: Control word	CIO, G, A, T, C, #, DM,
S: 1st source word	CIO, G, A, T, C, DM,
D: 1st destination word	CIO, G, A, DM,

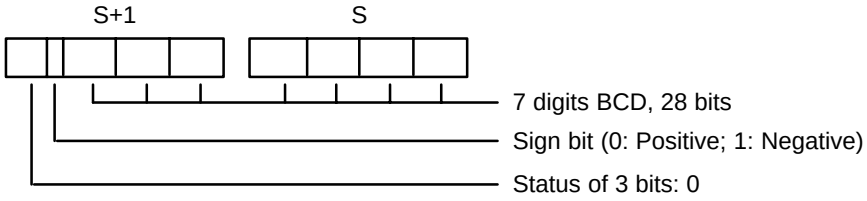
Description

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).

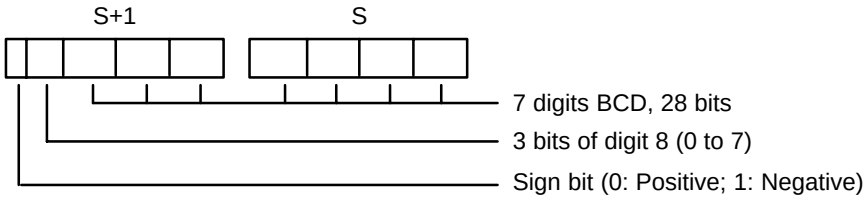
Note Special I/O Units sometimes sometimes require input of signed BCD data. BCDS(276) or BDSL(278) can be used to easily convert signed binary data to signed BCD data.

The output data format and range designations for the various control word contents are as follows:

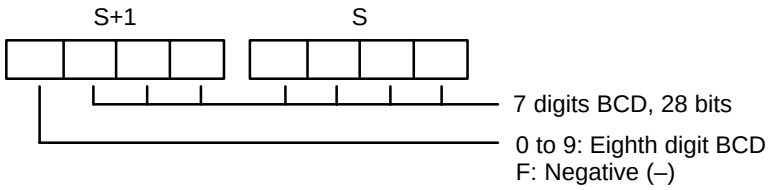
When C = 0000 (Output Data Range: –999 9999 to 999 9999 BCD)



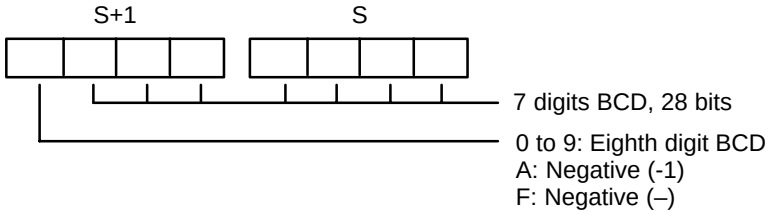
When C = 0001 (Output Data Range: –7999 9999 to 7999 9999 BCD)



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

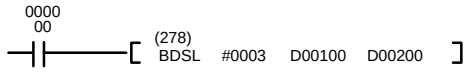
Refer to page 115 for general precautions on operand data areas.

Flags

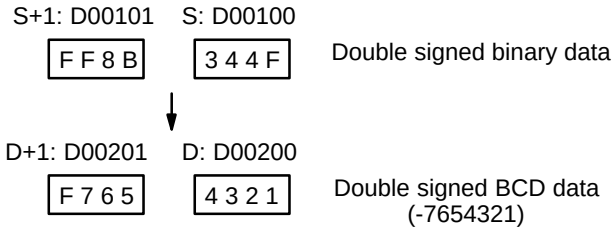
ER (A50003):	Data to be converted is not within range for data format.
	Content of a∗DM word is not BCD when set for BCD.
EQ (A50006)	Content of the converted data is all zeroes.
N (A50008)	Data to be converted is a negative number.

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 (V1/V2 CPUs)

The BCD Calculation Instructions perform arithmetic operations on BCD data. These instructions are supported only by version-1 or later CPUs.

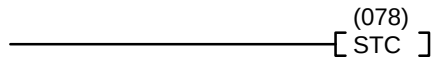
STC(078) and CLC(079), which set and clear the carry flag, are included in this group because most of the BCD operations make use of the carry flag (CY) in their results. Binary calculations and shift operations also use CY.

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.

Note BCD calculation instructions are also supported by version-2 CVM1 CPUs as symbol math instructions. Refer to *5-20 Symbol Math Instructions* for details.

5-18-1 SET CARRY: STC(078)

Ladder Symbol



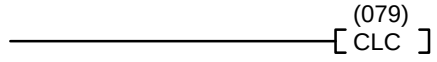
Variations

↑ 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(078)

When the execution condition is OFF, CLC(079) is not executed.When the execution condition is ON, CLC(079) turns OFF CY (A50004).

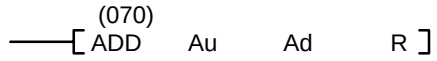
ADD(070), ADDL(074), ADB(080), ADBL(084), SUB(0710), SUBL(075), SBB(081), and SBBL(085) all make use of the carry flag in their calculations. When using any of these instructions, use CLC(079) to clear the carry flag in order to avoid having the calculations affected by previous instructions.

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.

Version-2 CVM1 CPUs supports 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



Variations

↑ ADD(070)

Operand Data Areas

Au: Augend word	CIO, G, A, T, C, #, DM, DR, IR
Ad: Addend word	CIO, G, A, T, C, #, DM, DR, IR
R: Result word	CIO, G, A, DM, DR, IR

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.

Note With version-2 CVM1 CPUs, mathematics instructions can use symbols. The instructions corresponding to ADD(070) and ADDL(074) are +BC(406) and +BCL(407).

Precautions

Au and Ad must be BCD.

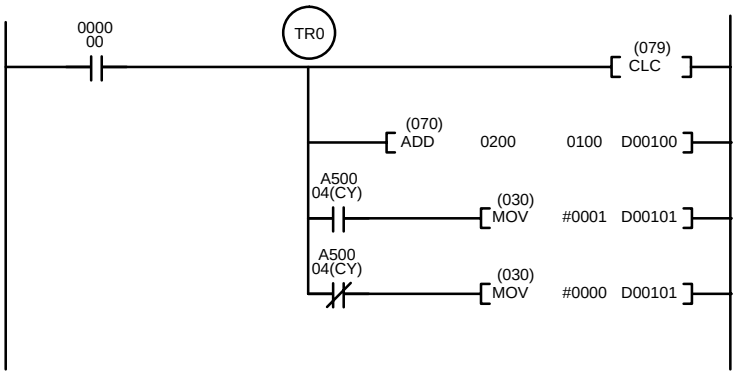
Note Refer to page 115 for general precautions on operand data areas.

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

(071)
SUB Mi Su R

Variations

↑ SUB(071)

Operand Data Areas

Mi: Minuend word	CIO, G, A, T, C, #, DM, DR, IR
Su: Subtrahend word	CIO, G, A, T, C, #, DM, DR, IR
R: Result word	CIO, G, A, DM, DR, IR

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).

Mi – Su – CY → CY R

Note With version-2 CVM1 CPUs, mathematics instructions can use symbols. The instructions corresponding to SUB(071) and SUBL(075) are –BC(416) and –BCL(417).

Precautions

Mi and Su must be BCD.

Note Refer to page 115 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.

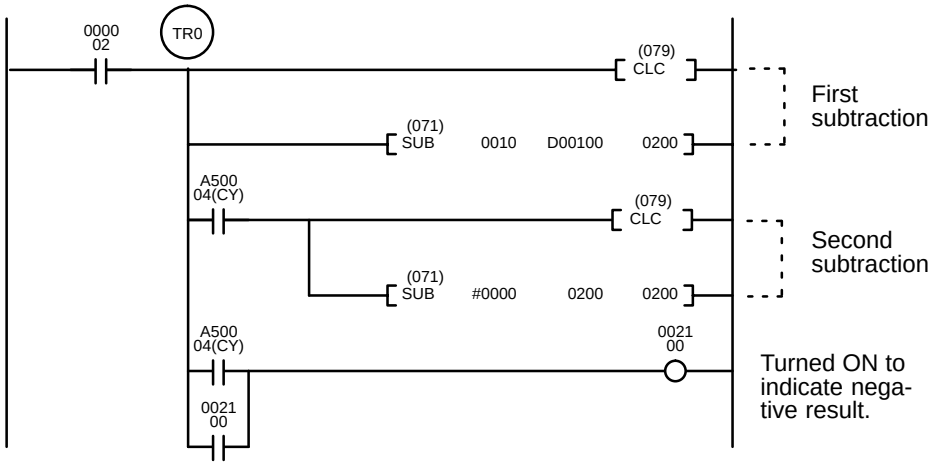
Example

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.



Address	Instruction	Operands
00000	LD	000002
00001	OUT	TR0
00002	CLC(079)	
00003	SUB(071)	
		0010
		D00100
		0200
00004	AND	A50004
00005	CLC(079)	
00006	SUB(071)	
		#0000
		0200
		0200
00007	LD	TR0
00008	AND	A50004
00009	OR	002100
00010	OUT	002100

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)

Ladder Symbol

```
   (072)
 ──[ MUL   Md    Mr    R ]
```

Variations

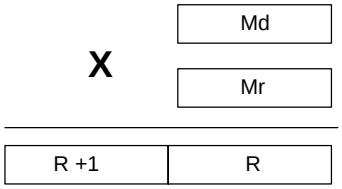
↑ MUL(072)

Operand Data Areas

Md: Multiplicand word	CIO, G, A, T, C, #, DM, DR, IR
Mr: Multiplier word	CIO, G, A, T, C, #, DM, DR, IR
R: Result word	CIO, G, A, DM

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 With version-2 CVM1 CPUs, mathematics instructions can use symbols. The instructions corresponding to MUL(072) and MULL(076) are *B(424) and *BL(425).

Precautions

Md and Mr must be BCD.

Note Refer to page 115 for general precautions on operand data areas.

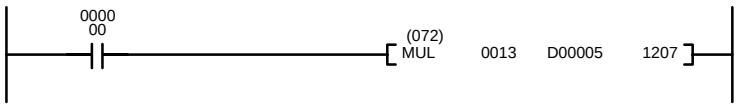
Flags

ER (A50003): Content of Md or Mr is not BCD.
The content of a *DM word is not BCD when set for BCD.

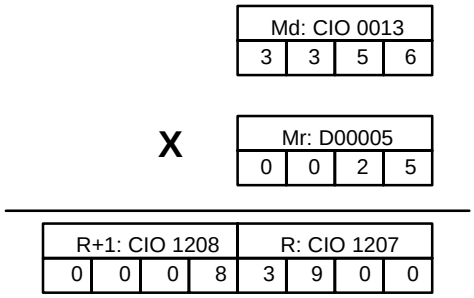
EQ (A50006): The result is 0.

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



5-18-6 BCD DIVIDE: DIV(073)

Ladder Symbol

(073)
DIV Dd Dr R

Variations

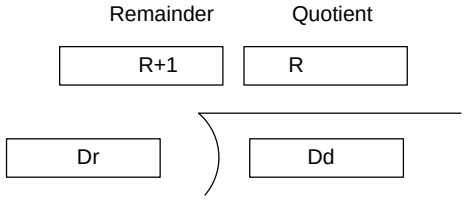
↑ DIV(073)

Operand Data Areas

Dd: Dividend word	CIO, G, A, T, C, #, DM, DR, IR
Dr: Divisor word	CIO, G, A, T, C, #, DM, DR, IR
R: Result word	CIO, G, A, DM

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 With version-2 CVM1 CPUs, mathematics instructions can use symbols. The instructions corresponding to DIV(073) and DIVL(077) are /B(434) and /BL(435).

Precautions

Dd and Dr must be BCD.

Note Refer to page 115 for general precautions on operand data areas.

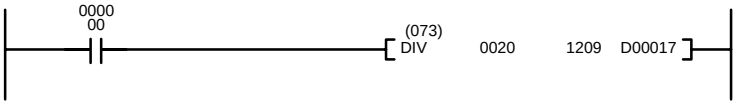
Flags

ER (A50003): Content of Dd or Dr is not BCD.
Content of *DM word is not BCD when set for BCD.

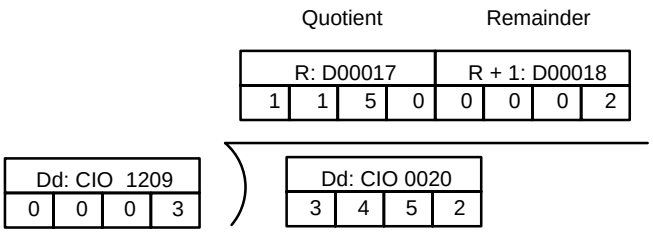
EQ (A50006): The result is 0.

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.

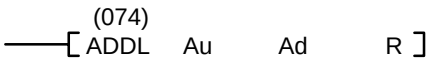


Address	Instruction	Operands
00000	LD	000000
00001	DIV(073)	
		0020
		1209
		D00017



5-18-7 DOUBLE BCD ADD: ADDL(074)

Ladder Symbol



Variations

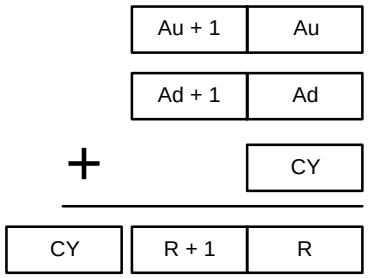
↑ ADDL(074)

Operand Data Areas

Au: 1st augend word	CIO, G, A, T, C, #, DM
Ad: 1st addend word	CIO, G, A, T, C, #, DM
R: 1st result word	CIO, G, A, DM

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 With version-2 CVM1 CPUs, mathematics instructions can use symbols. The instructions corresponding to ADD(070) and ADDL(074) are +BC(406) and +BCL(407).

Precautions

Au and Ad must be BCD.

Note Refer to page 115 for general precautions on operand data areas.

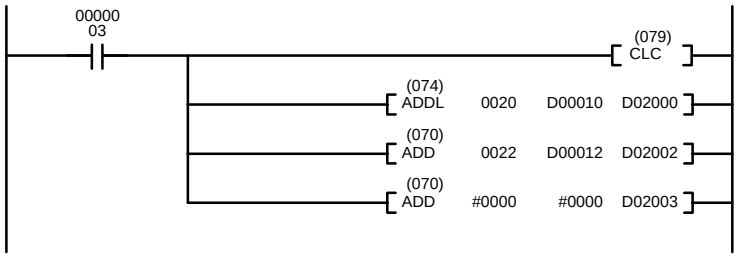
Flags

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

(075)
SUBL Mi Su R

Variations

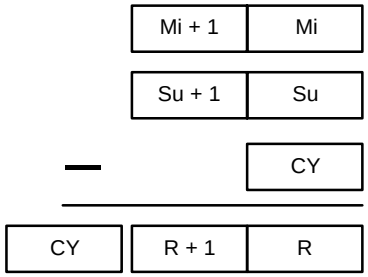
↑ SUBL(075)

Operand Data Areas

Mi: 1st minuend word	CIO, G, A, T, C, #, DM
Su: 1st subtrahend word	CIO, G, A, T, C, #, DM
R: 1st result word	CIO, G, A, DM

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 With version-2 CVM1 CPUs, mathematics instructions can use symbols. The instructions corresponding to SUB(071) and SUBL(075) are –BC(416) and –BCL(417).

Precautions

Mi and Su must be BCD.

Constants are input using eight digits.

Note Refer to page 115 for general precautions on operand data areas.

Flags

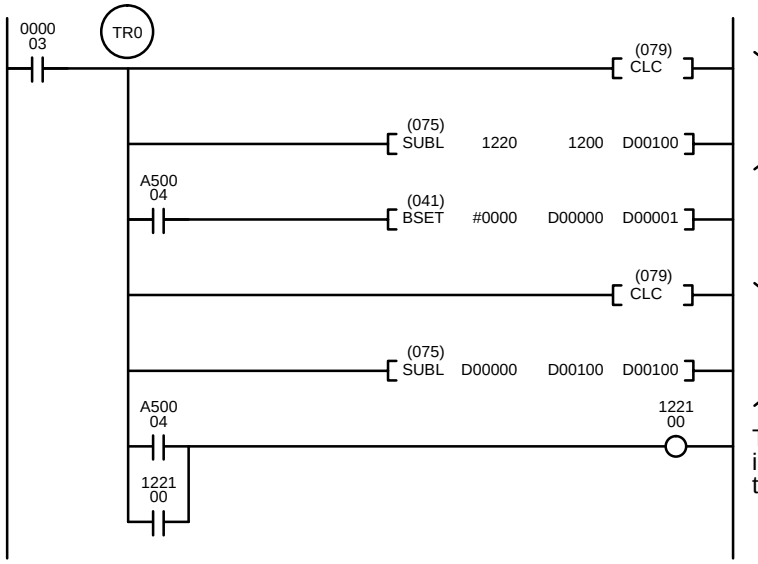
ER (A50003): Content of Mi, Mi+1,Su or Su+1 is not BCD.
Content of *DM word is not BCD when set for BCD.

CY (A50004): There is a carry in the result.

EQ (A50006): The result is 0.

Example

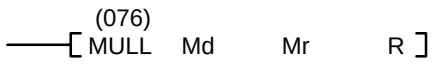
The following example works much like that for single-word subtraction. In this example, however, the 8-digit number in CIO 0121 and CIO 0120 is subtracted from the 8-digit number in CIO 0201 and CIO 0200 when CIO 000003 is ON, and the result is output to D00101 and D00100. If the result is negative, 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)

Ladder Symbol



Variations

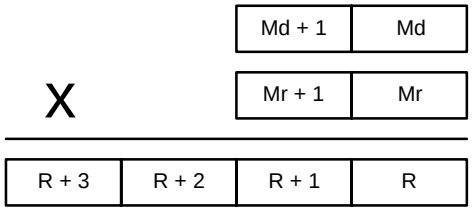
↑ MULL(076)

Operand Data Areas

Md: 1st multiplicand wd	CIO, G, A, T, C, #, DM
Mr: 1st multiplier word	CIO, G, A, T, C, #, DM
R: 1st result word	CIO, G, A, DM

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 With version-2 CVM1 CPUs, mathematics instructions can use symbols. The instructions corresponding to MUL(072) and MULL(076) are *B(424) and *BL(425).

Precautions

Md, Md+1, Mr, and Mr+1 must be BCD.

Note Refer to page 115 for general precautions on operand data areas.

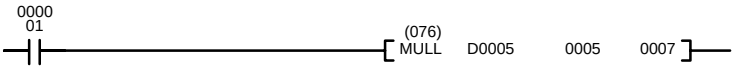
Flags

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.

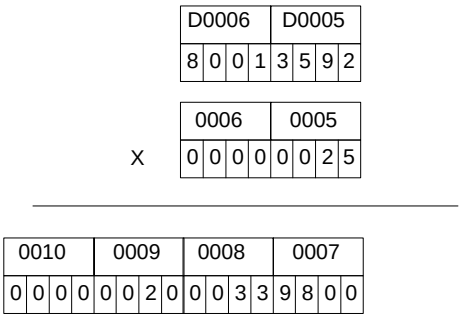
EQ (A50006): The result is 0.

Example

When CIO 000001 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



5-18-10 DOUBLE BCD DIVIDE: DIVL(077)

Ladder Symbol

(077)
DIVL Dd Dr R

Variations

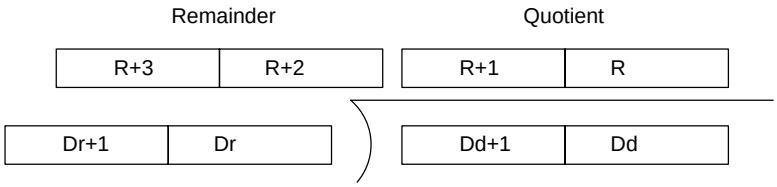
↑ DIVL(077)

Operand Data Areas

Dd: 1st dividend word	CIO, G, A, T, C, #, DM
Dr: 1st divisor word	CIO, G, A, T, C, #, DM
R: 1st result word	CIO, G, A, DM

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 With version-2 CVM1 CPUs, mathematics instructions can use symbols. The instructions corresponding to DIV(073) and DIVL(077) are /B(434) and /BL(435).

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 115 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

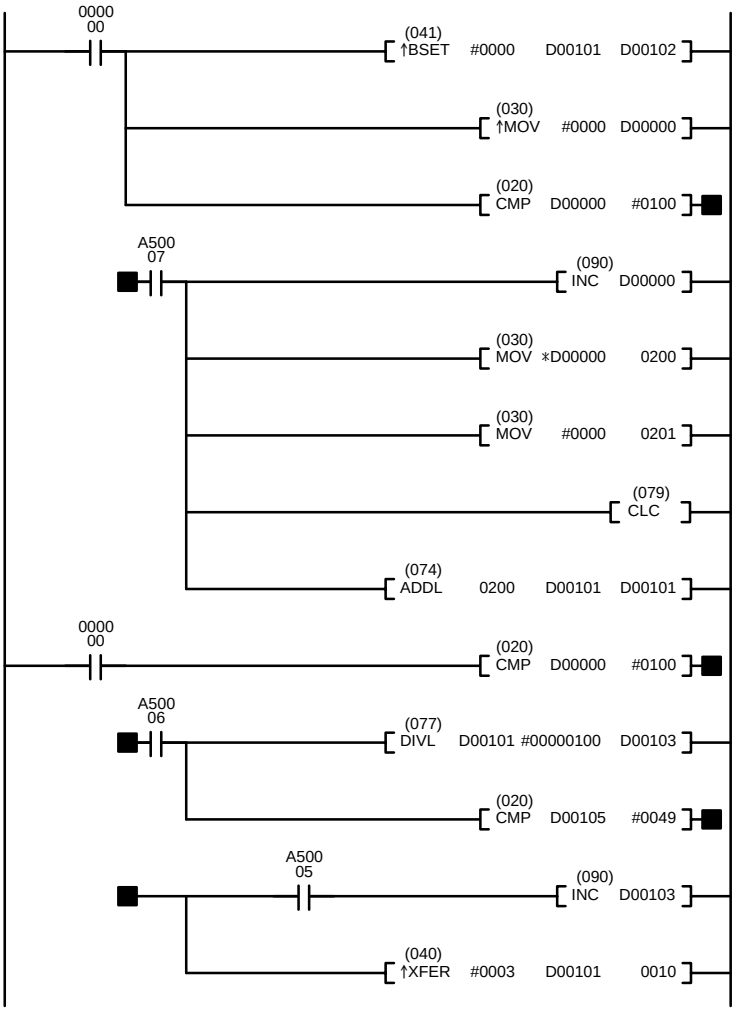
The following example shows how to use DIVL(77) to calculate the average of 100 four-digit numbers. These numbers are added and divided using the long versions of the instructions so that the answer can be rounded to preserve accuracy. This example illustrates not only the use of DIVL(77), but also the 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 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).

Note Binary calculation instructions are also supported by version-2 CVM1 CPUs as symbol math instructions. Refer to *5-20 Symbol Math Instructions* for details.

5-19-1 BINARY ADD: ADB(080)

Ladder Symbol

Variations

↑ ADB(080)

Operand Data Areas

Au: Augend word	CIO, G, A, T, C, #, DM, DR, IR
Ad: Addend word	CIO, G, A, T, C, #, DM, DR, IR
R: Result word	CIO, G, A, DM, DR, IR

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.

Note With version-2 CVM1 CPUs, mathematics instructions can use symbols. The instructions corresponding to ADB(080) and ADBL(084) are +C(402) and +CL(403). In addition, Overflow (A50009) and Underflow (A50010) Flags are added.

Precautions

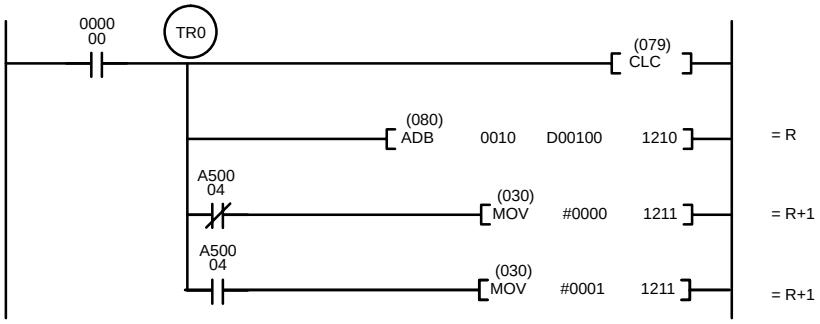
Refer to page 115 for general precautions on operand data areas.

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				R: D01000			
0	0	0	1	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

(081)
SBB Mi Su R

Variations

↑ SBB(081)

Operand Data Areas

Mi: Minuend word	CIO, G, A, T, C, #, DM, DR, IR
Su: Subtrahend word	CIO, G, A, T, C, #, DM, DR, IR
R: Result word	CIO, G, A, DM, DR, IR

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.

Mi – Su – CY → CY R

Note With version-2 CVM1 CPUs, mathematics instructions can use symbols. The instructions corresponding to SUB(081) and SUBL(085) are –C(412) and –CL(413). In addition, Overflow (A50009) and Underflow (A50010) Flags are added.

Precautions

Refer to page 115 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.

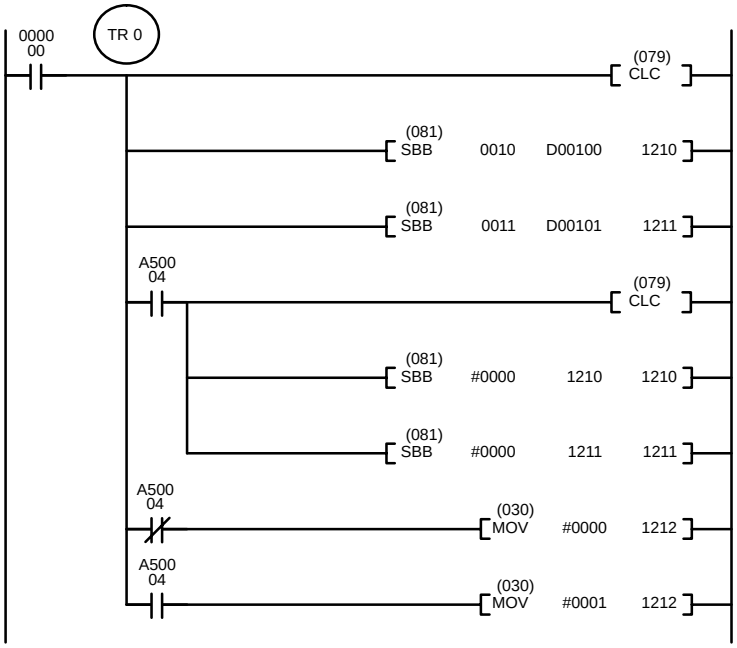
EQ (A50006): The result is 0.

N (A50008): Shows the status of bit 15 of R.

Example

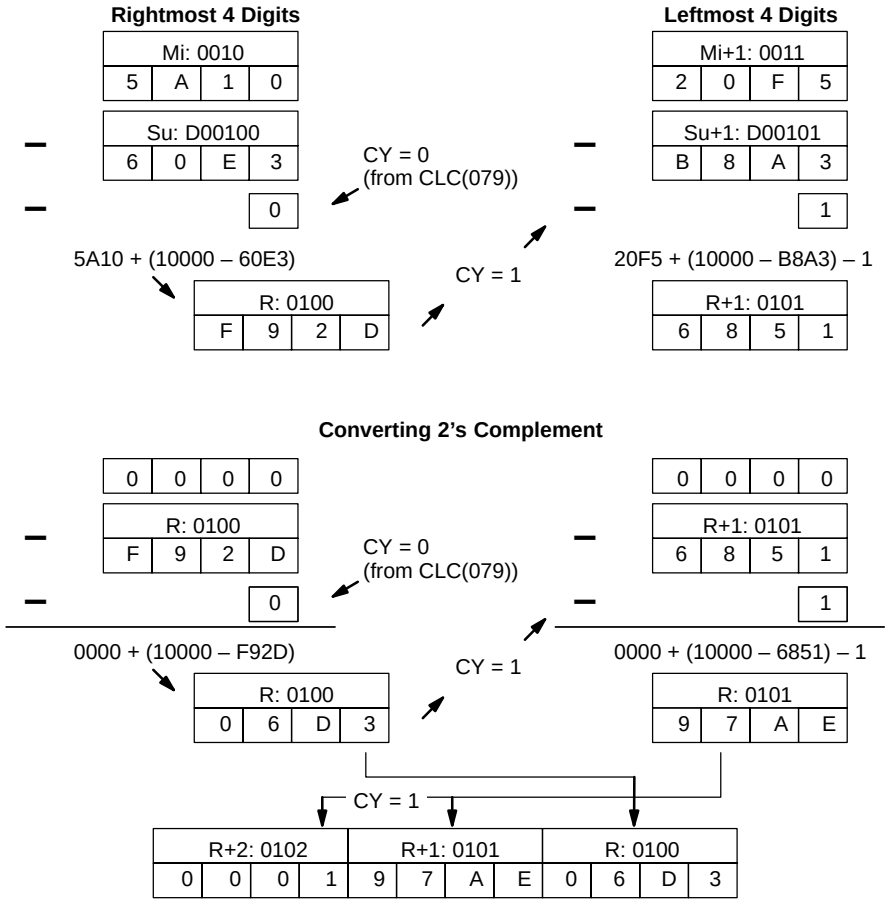
The following example demonstrates the use of SBB(081) in an 8-digit subtraction. In actual practice, 8-digit binary numbers can be subtracted more quickly and easily using the DOUBLE BINARY SUBTRACT: SBBL(085) instruction instead of a combination of SBB(081) instructions.

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, Su is less than Mi, 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)

Ladder Symbol

(082)
MLB Md Mr R

Variations

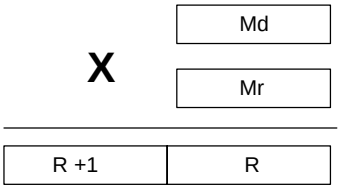
↑ MLB(082)

Operand Data Areas

Md: Multiplicand word	CIO, G, A, T, C, #, DM, DR, IR
Mr: Multiplier word	CIO, G, A, T, C, #, DM, DR, IR
R: Result word	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 With version-2 CVM1 CPUs, mathematics instructions can use symbols. The instructions corresponding to MLB(082) and MLBL(086) are *U(422) and *UL(423).

Precautions Refer to page 115 for general precautions on operand data areas.

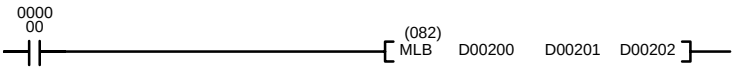
Flags

ER (A50003): Content of *DM word is not BCD when set for BCD.

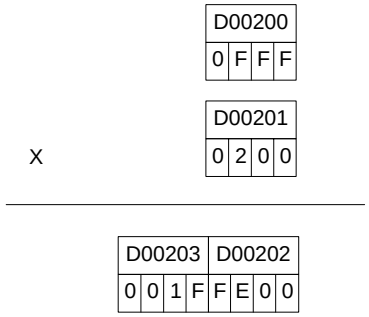
EQ (A50006): The result is 0.

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.

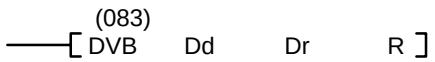


Address	Instruction	Operands
00000	LD	000000
00001	MLB(0820	
		D00200
		D00201
		D00202



5-19-4 BINARY DIVIDE: DVB(083)

Ladder Symbol



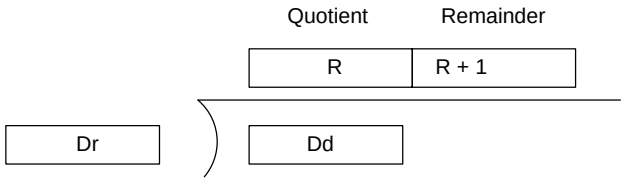
Variations

↑ DVB(083)

Operand Data Areas

Dd: Dividend word	CIO, G, A, T, C, #, DM, DR, IR
Dr: Divisor word	CIO, G, A, T, C, #, DM, DR, IR
R: Result word	CIO, G, A, DM

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 With version-2 CVM1 CPUs, mathematics instructions can use symbols. The instructions corresponding to DVB(083) and DVBL(085) are /U(432) and /UL(433).

Precautions Dr must not be 0.

Note Refer to page 115 for general precautions on operand data areas.

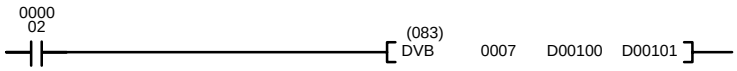
Flags

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.

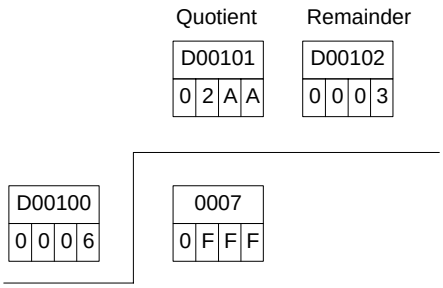
Example

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.

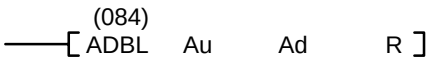


Address	Instruction	Operands
00000	LD	000002
00001	DVB(083)	
		0007
		D00100
		D00101



5-19-5 DOUBLE BINARY ADD: ADBL(084)

Ladder Symbol



Variations

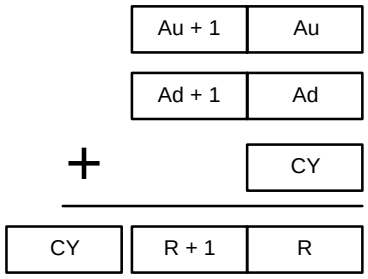
↑ ADBL(084)

Operand Data Areas

Au: 1st augend word	CIO, G, A, T, C, #, DM
Ad: 1st addend word	CIO, G, A, T, C, #, DM
R: 1st result word	CIO, G, A, DM

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 With version-2 CVM1 CPUs, mathematics instructions can use symbols. The instructions corresponding to ADB(080) and ADBL(084) are +C(402) and +C(403). In addition, Overflow (OF) and Underflow (UF) Flags are added.

Precautions

Refer to page 115 for general precautions on operand data areas.

Flags

ER (A50003): Content of ∗DM word is not BCD when set for BCD.

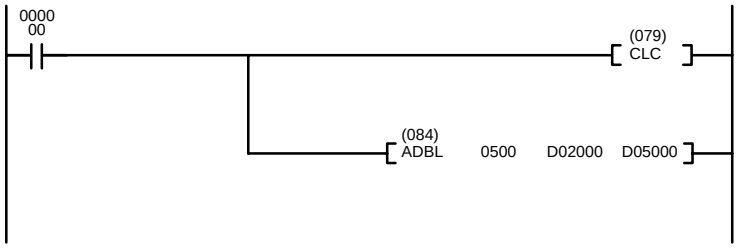
CY (A50004): The result is greater than FFFF FFFF.

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 9th 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

	Au + 1 : CIO 0501	Au : CIO 0500	
	5 5 4 B	5 9 5 2	
	Ad + 1 : D02001	Ad : D02000	
	6 1 4 3	2 9 D 8	
+		0	- - - - CY (Cleared with CLC(079))
	D + 1 : D05001	D : D05000	
	B 6 8 E	8 3 2 A	
		0	- - - - CY (No carry)
		1	- - - - N (Leftmost bit is ON)

5-19-6 DOUBLE BINARY SUBTRACT: SBBL(085)

Ladder Symbol

(085)
SBBL Mi Su R

Variations

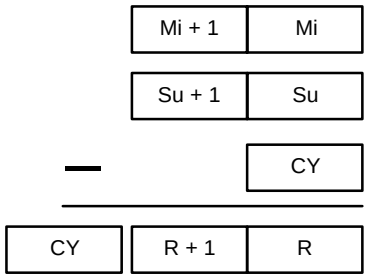
↑ SBBL(085)

Operand Data Areas

Mi: 1st minuend word	CIO, G, A, T, C, #, DM
Su: 1st subtrahend word	CIO, G, A, T, C, #, DM
R: 1st result word	CIO, G, A, DM

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 With version-2 CVM1 CPUs, mathematics instructions can use symbols. The instructions corresponding to SUB(081) and SUBL(085) are –C(412) and –CL(413). In addition, Overflow (A50009) and Underflow (A50010) Flags are added.

Precautions

Refer to page 115 for general precautions on operand data areas.

Flags

ER (A50003): Content of ∗DM word is not BCD when set for BCD.

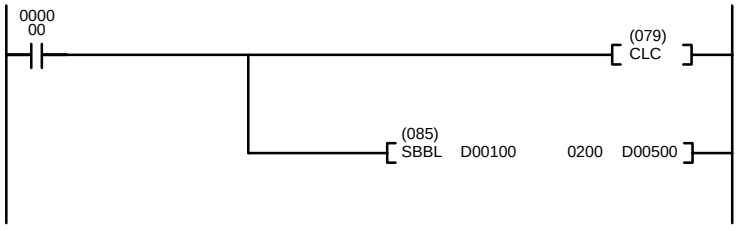
CY (A50004): The result is negative.

EQ (A50006): The result is 0.

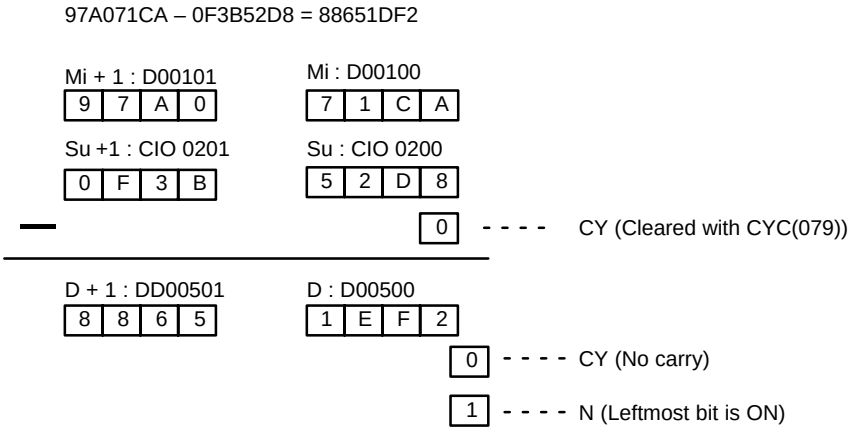
N (A50008): Shows the status of bit 15 of R+1.

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.

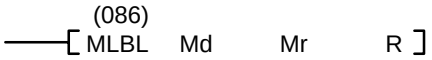


Address	Instruction	Operands
00000	LD	000000
00001	CLC(079)	
00002	SBBL(085)	
		D00100
		0200
		D00500



5-19-7 DOUBLE BINARY MULTIPLY: MLBL(086)

Ladder Symbol



Variations

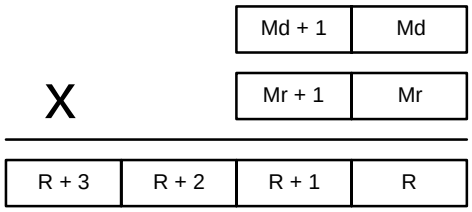
↑ MLBL(086)

Operand Data Areas

Md: 1st multiplicand wd	CIO, G, A, T, C, #, DM
Mr: 1st multiplier word	CIO, G, A, T, C, #, DM
R: 1st result word	CIO, G, A, DM

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 With version-2 CVM1 CPUs, mathematics instructions can use symbols. The instructions corresponding to MLB(082) and MLBL(086) are *U(422) and *UL(423).

Precautions

Refer to page 115 for general precautions on operand data areas.

Flags

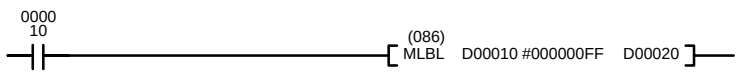
ER (A50003): Content of *DM word is not BCD when set for BCD.

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

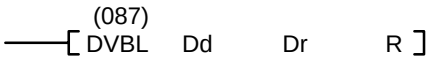
X

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)

Ladder Symbol



Variations

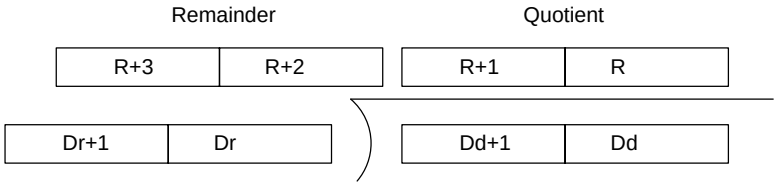
↑ DVBL(087)

Operand Data Areas

Dd: 1st dividend word	CIO, G, A, T, C, #, DM
Dr: 1st divisor word	CIO, G, A, T, C, #, DM
R: 1st result word	CIO, G, A, DM

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 With version-2 CVM1 CPUs, mathematics instructions can use symbols. The instructions corresponding to DVB(083) and DVBL(085) are /U(432) and /UL(433). In addition, Overflow (A50009) and Underflow (A50010) Flags are added.

Precautions

Dr and Dr+1 must not contain 0.

Constants are expressed in eight digits.

Note Refer to page 115 for general precautions on operand data areas.

Flags

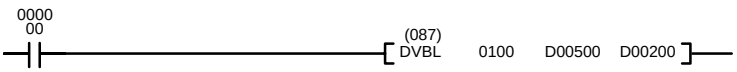
ER (A50003): Dr and Dr+1 contain 0.
Content of ∗DM word is not BCD when set for BCD.

EQ (A50006): The result is 0.

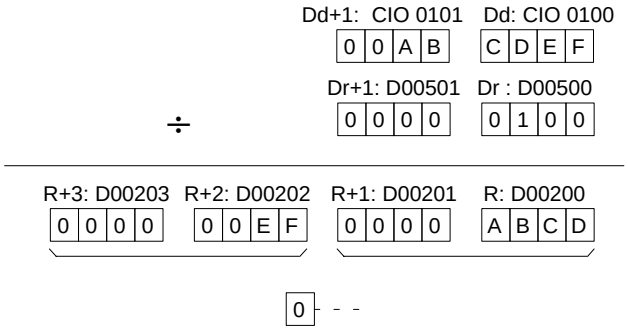
N (A50008): Shows the status of bit 15 of R+1.

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


Dd+1: CIO 0101
0 0 A B
Dd: CIO 0100
C D E F
Dr+1: D00501
0 0 0 0
Dr : D00500
0 1 0 0
÷
R+3: D00203
0 0 0 0
R+2: D00202
0 0 E F
R+1: D00201
0 0 0 0
R: D00200
A B C D
0

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) (CVM1 V2)

SIGNED BINARY ADD WITHOUT CARRY: +(400)

Ladder Symbol

(400)
+ Au Ad R

Variations

↑ +(400)

Operand Data Areas

Au: Augend word	CIO, G, A, T, C, #, DM, DR, IR
Ad: Addend word	CIO, G, A, T, C, #, DM, DR, IR
R: Result word	CIO, G, A, DM, DR, IR

DOUBLE SIGNED BINARY ADD WITHOUT CARRY: +L(401)

Ladder Symbol

(401)
+L Au Ad R

Variations

↑ +L(401)

Operand Data Areas

Au: 1st augend word	CIO, G, A, T, C, #, DM
Ad: 2nd addend word	CIO, G, A, T, C, #, DM
R: 1st result word	CIO, G, A, DM

SIGNED BINARY ADD WITH CARRY: +C(402)

Ladder Symbol

(402)
+C Au Ad R

Variations

↑ +C(402)

Operand Data Areas

Au: Augend word	CIO, G, A, T, C, #, DM, DR, IR
Ad: Addend word	CIO, G, A, T, C, #, DM, DR, IR
R: Result word	CIO, G, A, DM, DR, IR

DOUBLE SIGNED BINARY ADD WITH CARRY: +CL(403)

Ladder Symbol

(403)
+CL Au Ad R

Variations

↑ +CL(403)

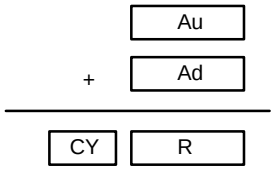
Operand Data Areas

Au: 1st augend word	CIO, G, A, T, C, #, DM
Ad: 2nd addend word	CIO, G, A, T, C, #, DM
R: 1st result word	CIO, G, A, DM

Description

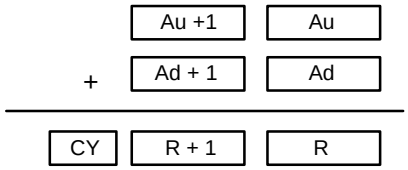
SIGNED BINARY ADD WITHOUT CARRY

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.



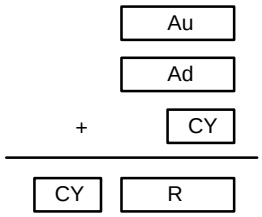
DOUBLE SIGNED BINARY ADD WITHOUT CARRY

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.



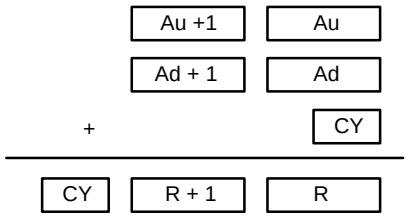
SIGNED BINARY ADD WITH CARRY

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 115 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) (CVM1 V2)

BCD ADD WITHOUT CARRY: +B(404)

Ladder Symbol

Variations

↑ +B(404)

Operand Data Areas

Au: Augend word	CIO, G, A, T, C, #, DM, DR, IR
Ad: Addend word	CIO, G, A, T, C, #, DM, DR, IR
R: Result word	CIO, G, A, DM, DR, IR

DOUBLE BCD ADD WITHOUT CARRY: +BL(405)

Ladder Symbol

Variations

↑ +BL(405)

Operand Data Areas

Au: 1st augend word	CIO, G, A, T, C, #, DM
Ad: 2nd addend word	CIO, G, A, T, C, #, DM
R: 1st result word	CIO, G, A, DM

BCD ADD WITH CARRY: +BC(406)

Ladder Symbol

Variations

↑ +BC(406)

Operand Data Areas

Au: Augend word	CIO, G, A, T, C, #, DM, DR, IR
Ad: Addend word	CIO, G, A, T, C, #, DM, DR, IR
R: Result word	CIO, G, A, DM, DR, IR

DOUBLE BCD ADD WITH CARRY: +BCL(407)

Ladder Symbol

Variations

↑ +BCL(407)

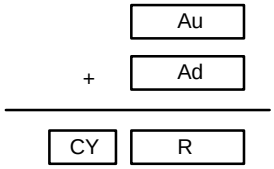
Operand Data Areas

Au: 1st augend word	CIO, G, A, T, C, #, DM
Ad: 2nd addend word	CIO, G, A, T, C, #, DM
R: 1st result word	CIO, G, A, DM

Description

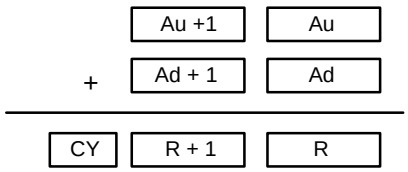
BCD ADD WITHOUT CARRY

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.



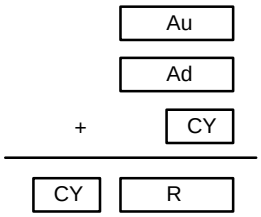
DOUBLE BCD ADD WITHOUT CARRY

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.



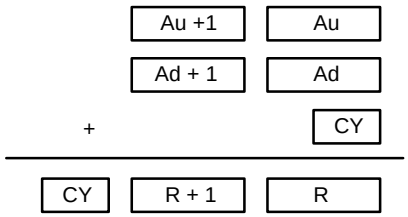
BCD ADD WITH CARRY

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 115 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.

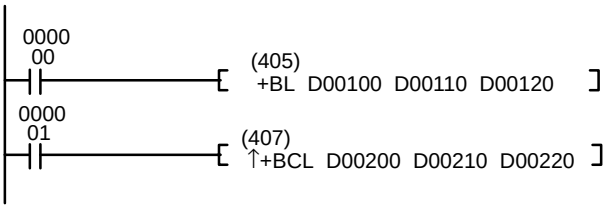
Example

+BL Operation

When CIO 000000 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) (CVM1 V2)

SIGNED BINARY SUBTRACT WITHOUT CARRY: –(410)

Ladder Symbol

(410)
– Mi Su R

Variations

↑ –(410)

Operand Data Areas

Mi: Minuend word	CIO, G, A, T, C, #, DM, DR, IR
Su: Subtrahend word	CIO, G, A, T, C, #, DM, DR, IR
R: Result word	CIO, G, A, DM, DR, IR

DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY: –L(411)

Ladder Symbol

(411)
–L Mi Su R

Variations

↑ –L(411)

Operand Data Areas

Mi: Minuend word	CIO, G, A, T, C, #, DM,
Su: Subtrahend word	CIO, G, A, T, C, #, DM,
R: Result word	CIO, G, A, DM,

SIGNED BINARY SUBTRACT WITH CARRY: –C(412)

Ladder Symbol

(412)
–C Mi Su R

Variations

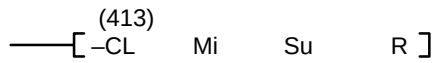
↑ –C(412)

Operand Data Areas

Mi: Minuend word	CIO, G, A, T, C, #, DM, DR, IR
Su: Subtrahend word	CIO, G, A, T, C, #, DM, DR, IR
R: Result word	CIO, G, A, DM, DR, IR

DOUBLE SIGNED BINARY SUBTRACT WITH CARRY: –CL(413)

Ladder Symbol



Variations

↑ –CL(413)

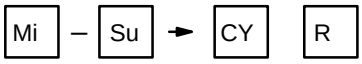
Operand Data Areas

Mi: Minuend word	CIO, G, A, T, C, #, DM,
Su: Subtrahend word	CIO, G, A, T, C, #, DM,
R: Result word	CIO, G, A, DM,

Description

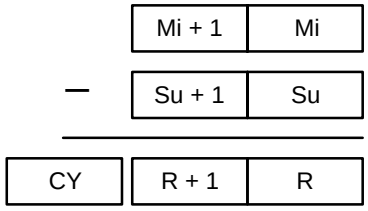
SIGNED BINARY SUBTRACT WITHOUT CARRY

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.

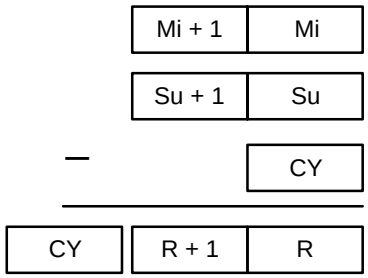


SIGNED BINARY SUBTRACT WITH CARRY

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 115 for general precautions on operand data areas.

Flags

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
	$FFFF_H$ →	–1	65535
–)	0001_H →	–) +1	–) 1
	$FFFE_H$ →	–2 (Note 1)	65534 (Note 2)

Negative Flag ON
Carry Flag OFF

Numeric Example 2

		Signed data	Unsigned data
	$FFFD_H$ →	–3	65533
–)	$FFFF_H$ →	–) –1	–) 65535
	$FFFE_H$ →	–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.

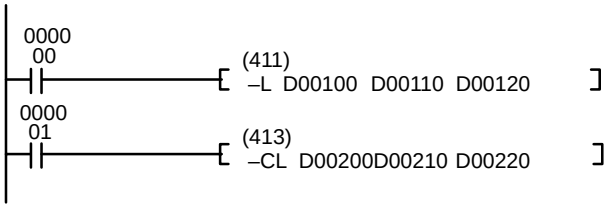
Programming Example 1

–L Operation

When CIO 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 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.

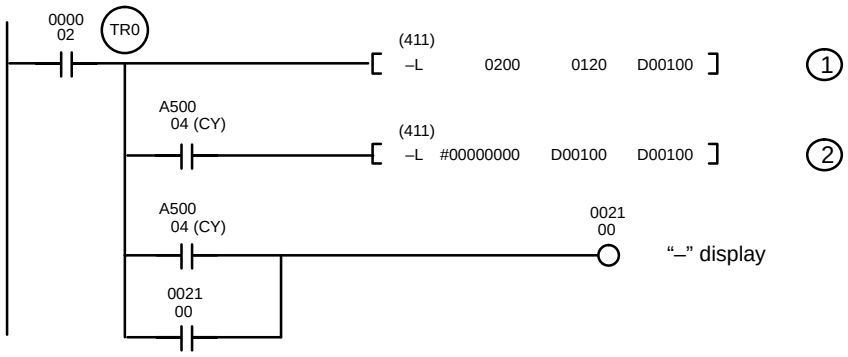


Address	Instruction	Operands
00000	LD	000000
00001	–L(411)	
		D00100
		D00110
		D00120
00002	LD	000001
00003	–CL(413)	
		D00200
		D00210
		D00220

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) (CVM1 V2)

BCD SUBTRACT WITHOUT CARRY: –B(414)

Ladder Symbol

Variations

↑ –B(414)

Operand Data Areas

Mi: Minuend word	CIO, G, A, T, C, #, DM, DR, IR
Su: Subtrahend word	CIO, G, A, T, C, #, DM, DR, IR
R: Result word	CIO, G, A, DM, DR, IR

DOUBLE BCD SUBTRACT WITHOUT CARRY: –BL(415)

Ladder Symbol

Variations

↑ –BL(415)

Operand Data Areas

Mi: 1st minuend word	CIO, G, A, T, C, #, DM
Su: 1st subtrahend word	CIO, G, A, T, C, #, DM
R: 1st result word	CIO, G, A, DM

BCD SUBTRACT WITH CARRY: –BC(416)

Ladder Symbol

Variations

↑ –BC(416)

Operand Data Areas

Mi: Minuend word	CIO, G, A, T, C, #, DM, DR, IR
Su: Subtrahend word	CIO, G, A, T, C, #, DM, DR, IR
R: Result word	CIO, G, A, DM, DR, IR

DOUBLE BCD SUBTRACT WITH CARRY: –BCL(417)

Ladder Symbol

Variations

↑ –BCL(417)

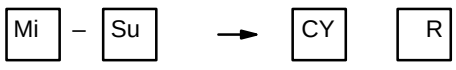
Operand Data Areas

Mi: 1st minuend word	CIO, G, A, T, C, #, DM
Su: 1st subtrahend word	CIO, G, A, T, C, #, DM
R: 1st result word	CIO, G, A, DM

Description

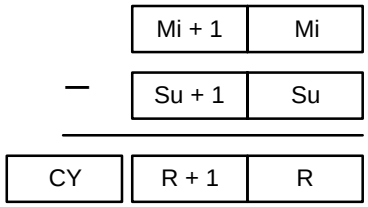
BCD SUBTRACT WITHOUT CARRY

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.

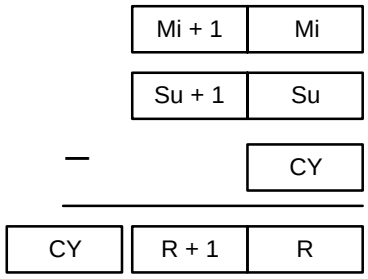


BCD SUBTRACT WITH CARRY

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.

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.



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 115 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.

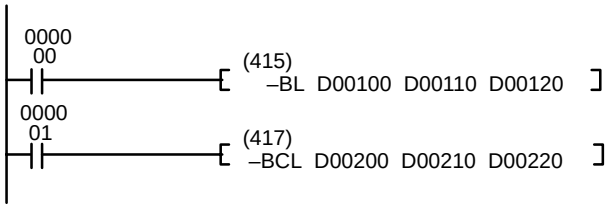
Example

–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 000001 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



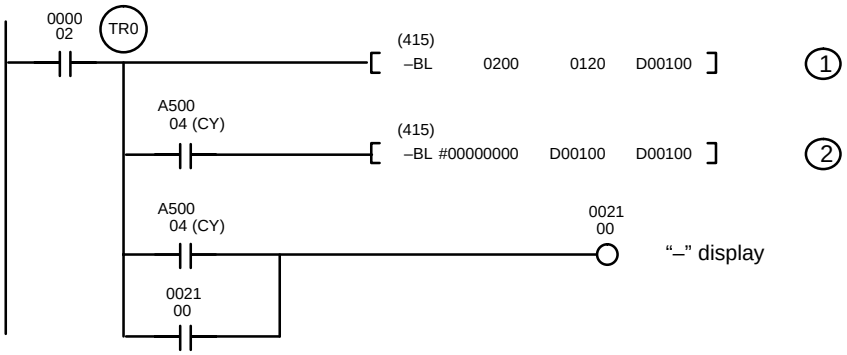
Address	Instruction	Operands
00000	LD	000000
00001	–BL(415)	
		D00100
		D00110
		D00120
00002	LD	000001
00003	–BCL(417)	
		D00200
		D00210
		D00220

Program Example

Example: 9,583,960 – 17,072,641 = –7,488,681.

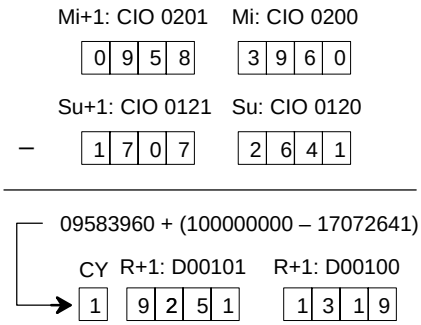
In this example, the eight-digit BCD content of CIO 0121 and CIO 0120 is subtracted from the content of CIO 0201 and CIO 0200, and the result is output in eight-digit BCD to D00101 and D00100. The result is negative, so the instruction at (2) will be executed, and the true value will then be output to 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.



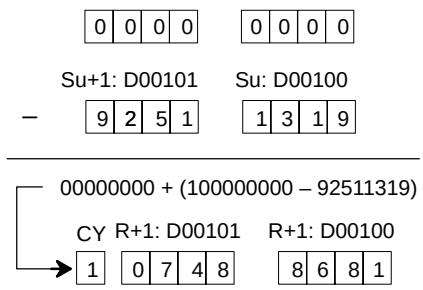
Address	Instruction	Operands
00000	LD	000002
00001	OUT	TR0
00002	–BL(415)	
		0200
		0120
		D00100
00003	LD	TR0
00004	AND	A50004
00005	–BL(415)	
		#00000000
		D00100
		D00100
00006	LD	TR0
00007	AND	A50004
00008	OR	002100
00009	OUT	002100

Subtraction at 1

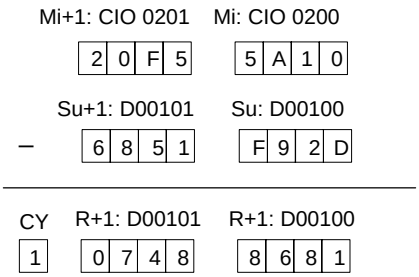


The Carry Flag (A50004) is ON, so the result is subtracted from 0000 0000.

Subtraction at 2



Final Subtraction Result



5-20-5 Binary Multiplication: ✳(420)/ ✳L(421)/✳U(422)/✳UL(423) (CVM1 V2)

SIGNED BINARY MULTIPLY: ✳(420)

Ladder Symbol

(420)
✳ Md Mr R

Variations

↑ ✳(420)

Operand Data Areas

Md: Multiplicand word	CIO, G, A, T, C, #, DM, DR, IR
Mr: Multiplier word	CIO, G, A, T, C, #, DM, DR, IR
R: Result word	CIO, G, A, DM

DOUBLE SIGNED BINARY MULTIPLY: ✳L(421)

Ladder Symbol

(421)
✳L Md Mr R

Variations

↑ ✳L(421)

Operand Data Areas

Md: 1st multiplicand wd	CIO, G, A, T, C, #, DM
Mr: 1st multiplier word	CIO, G, A, T, C, #, DM
R: 1st result word	CIO, G, A, DM

UNSIGNED BINARY MULTIPLY: ✳U(422)

Ladder Symbol

(422)
✳U Md Mr R

Variations

↑ ✳U(422)

Operand Data Areas

Md: Multiplicand word	CIO, G, A, T, C, #, DM, DR, IR
Mr: Multiplier word	CIO, G, A, T, C, #, DM, DR, IR
R: Result word	CIO, G, A, DM

DOUBLE UNSIGNED BINARY MULTIPLY: ✳UL(423)

Ladder Symbol

(423)
✳UL Md Mr R

Variations

↑ ✳UL(423)

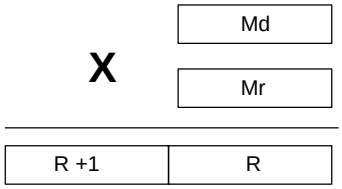
Operand Data Areas

Md: 1st multiplicand wd	CIO, G, A, T, C, #, DM
Mr: 1st multiplier word	CIO, G, A, T, C, #, DM
R: 1st result word	CIO, G, A, DM

Description

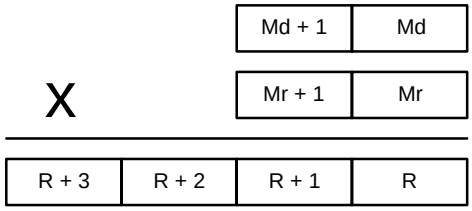
SIGNED BINARY MULTIPLY

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.



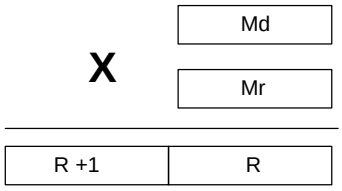
DOUBLE SIGNED BINARY MULTIPLY

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.



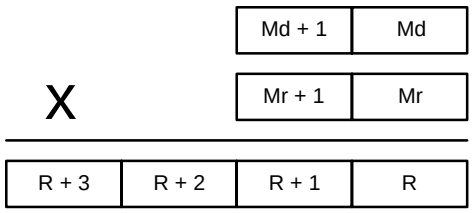
UNSIGNED BINARY MULTIPLY

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 115 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."

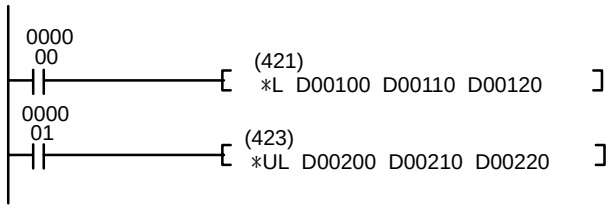
Example

✳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) (CVM1 V2)

BCD MULTIPLY: ∗B(424)

Ladder Symbol

(424)
∗B Md Mr R

Variations

↑ ∗B(424)

Operand Data Areas

Md: Multiplicand word	CIO, G, A, T, C, #, DM, DR, IR
Mr: Multiplier word	CIO, G, A, T, C, #, DM, DR, IR
R: Result word	CIO, G, A, DM

DOUBLE BCD MULTIPLY: ∗BL(425)

Ladder Symbol

(425)
∗BL Md Mr R

Variations

↑ ∗BL(425)

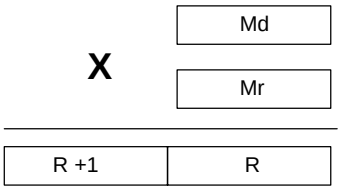
Operand Data Areas

Md: 1st multiplicand wd	CIO, G, A, T, C, #, DM
Mr: 1st multiplier word	CIO, G, A, T, C, #, DM
R: 1st result word	CIO, G, A, DM

Description

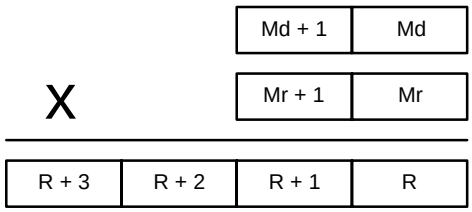
BCD MULTIPLY

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 115 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.

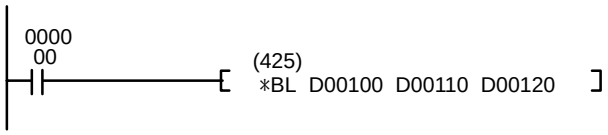
CY (A50004): There is a carry in the result.

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) (CVM1 V2)

SIGNED BINARY DIVIDE: /(430)

Ladder Symbol

(430)
/ Dd Dr R

Variations

↑ /(430)

Operand Data Areas

Dd: Dividend word	CIO, G, A, T, C, #, DM, DR, IR
Dr: Divisor word	CIO, G, A, T, C, #, DM, DR, IR
R: Result word	CIO, G, A, DM

DOUBLE SIGNED BINARY DIVIDE: /L(431)

Ladder Symbol

(431))
/L Dd Dr R

Variations

↑ /L(431)

Operand Data Areas

Dd: 1st dividend word	CIO, G, A, T, C, #, DM
Dr: 1st divisor word	CIO, G, A, T, C, #, DM
R: 1st result word	CIO, G, A, DM

UNSIGNED BINARY DIVIDE: /U(432)

Ladder Symbol

(432)
/U Dd Dr R

Variations

↑ /U(432)

Operand Data Areas

Dd: Dividend word	CIO, G, A, T, C, #, DM, DR, IR
Dr: Divisor word	CIO, G, A, T, C, #, DM, DR, IR
R: Result word	CIO, G, A, DM

DOUBLE UNSIGNED BINARY DIVIDE: /UL(433)

Ladder Symbol

(433)
/UL Dd Dr R

Variations

↑ /UL(433)

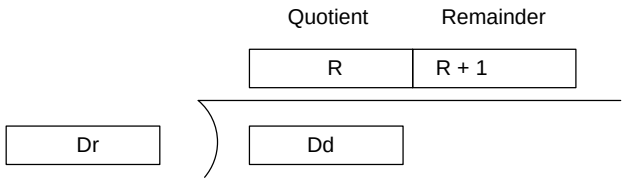
Operand Data Areas

Dd: 1st dividend word	CIO, G, A, T, C, #, DM
Dr: 1st divisor word	CIO, G, A, T, C, #, DM
R: 1st result word	CIO, G, A, DM

Description

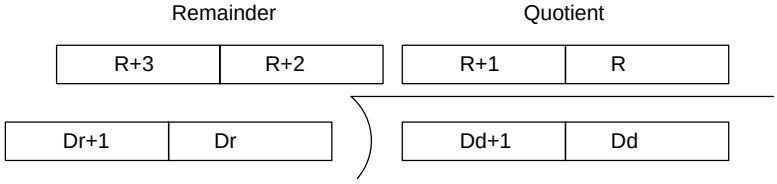
SIGNED BINARY DIVIDE

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.



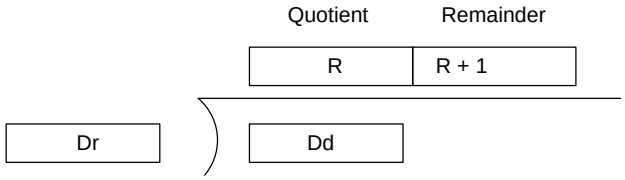
DOUBLE SIGNED BINARY DIVIDE

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.



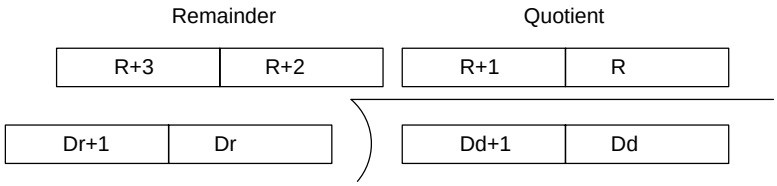
UNSIGNED BINARY DIVIDE

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_2 (or S_2, and S_2+1) must not be all zeros.

Note Refer to page 115 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.”

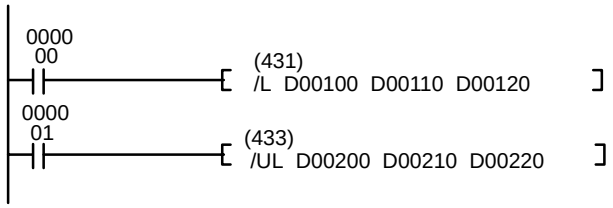
Example

/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 000001 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.

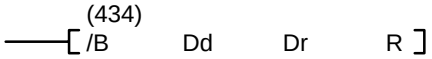


Address	Instruction	Operands
00000	LD	000000
00001	/L(431)	
		D00100
		D00110
		D00120
00002	LD	000001
00003	/UL(433)	
		D00200
		D00210
		D00220

5-20-8 BCD Division: /B(434)/ /BL(435) (CVM1 V2)

BCD DIVIDE: /B(434)

Ladder Symbol



Variations

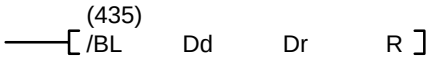
↑ /B(434)

Operand Data Areas

Dd: Dividend word	CIO, G, A, T, C, #, DM, DR, IR
Dr: Divisor word	CIO, G, A, T, C, #, DM, DR, IR
R: Result word	CIO, G, A, DM

DOUBLE BCD DIVIDE: /BL(435)

Ladder Symbol



Variations

↑ /BL(435)

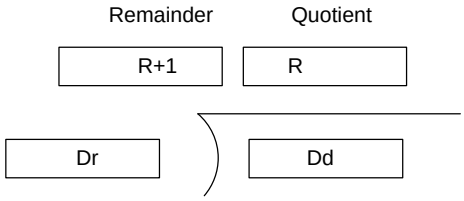
Operand Data Areas

Dd: 1st dividend word	CIO, G, A, T, C, #, DM
Dr: 1st divisor word	CIO, G, A, T, C, #, DM
R: 1st result word	CIO, G, A, DM

Description

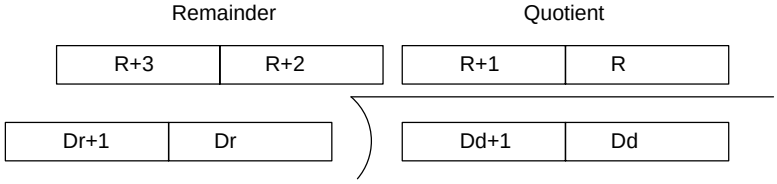
BCD DIVIDE

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 115 for general precautions on operand data areas.

Flags

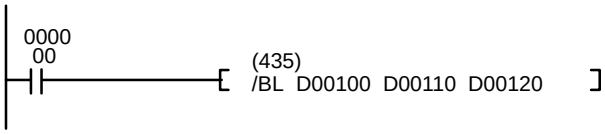
ER (A50003): Dd and Dr (or Dd, Dd+1, Dr, and Dr+1) are not BCD.
The content of a∗DM word is not BCD when set for BCD.

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. Version-2 CVM1 CPUs support the following instructions.

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

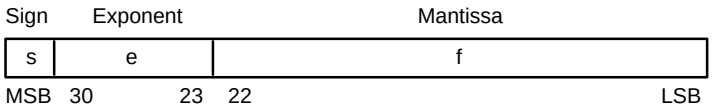
Data Format

Floating-point data expresses real numbers using a sign, exponent, and mantissa. When data is expressed in floating-point format, the following formula applies.

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 ≥ 1.0.

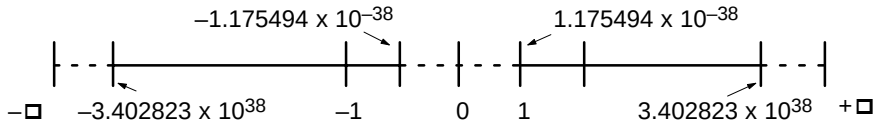
Number of Digits

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:

- –□
- -3.402823×10^{38} □ value □ $-1.175494 \times 10^{-38}$
- 0
- 1.175494×10^{-38} □ value □ 3.402823×10^{38}
- +□
- Not a number (NaN)



Special Numbers

The formats for NaN, ±□, and 0 are as follows:

NaN*:	e = 255, f ≠ 0
+□:	e = 255, f = 0, s= 0
–□:	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.

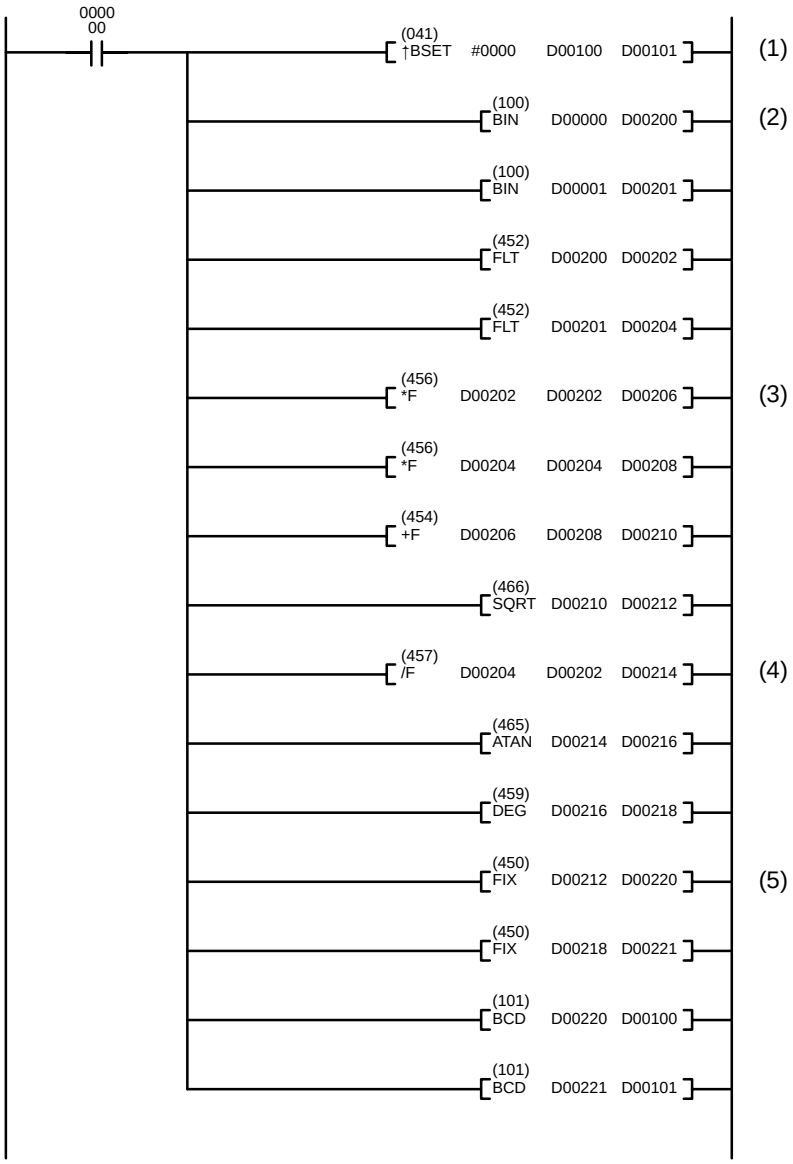
Floating-point Data Calculation Results

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 ±□. If the result is positive, it will be output as +□; if negative, then –□.

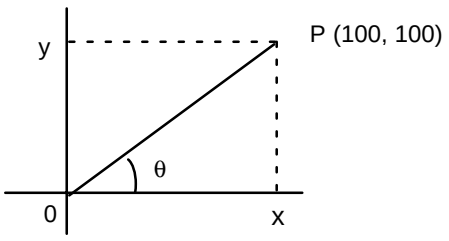
The Equals Flag (A50006) will only turn ON when both the exponent (e) and the mantissa (f) are zero after a calculation. A calculation result will also be output as zero when the absolute value of the result is less than the minimum value that can be expressed for floating-point data. In that case the 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 (θ, 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

Distance r = $\sqrt{x^2 + y^2}$

Angle $\theta = \tan^{-1}\left(\frac{y}{x}\right)$

Example

Distance r = $\sqrt{100^2 + 100^2}$ = 141.4214

Angle $\theta = \tan^{-1}\left(\frac{100}{100}\right)$ = 45.0

DM Contents

D00000	0100	x	→	D00100	0141	r
D00001	0100	y		D00101	0045	θ

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 to D00203 and D00202.
   d) The value of y that has been converted to floating-point data is output to 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 to D00100.
   c) The angle θ is output to to D00101.

5-21-1 FLOATING TO 16-BIT: FIX(450) (CVM1 V2)

Ladder Symbol

(450)
FIX S R

Variations

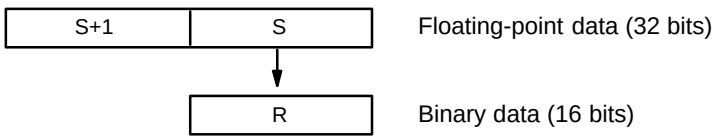
↑FIX(450)

Operand Data Areas

S: First source word	CIO, G, A, T, C, #, DM
R: Result word	CIO, G, A, DM, DR, IR

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 115 for general precautions on operand data areas.

Flags

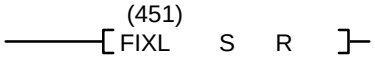
ER (A50003): S is not floating-point data.
The floating-point data is not between –32,768 to 32,767.
The content of a∗DM word is not BCD when set for BCD.

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) (CVM1 V2)

Ladder Symbol



Variations

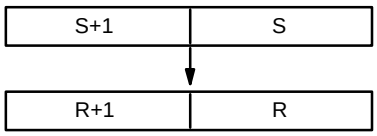
↑FIXL(451)

Operand Data Areas

S: First source word	CIO, G, A, T, C, #, DM
R: First result word	CIO, G, A, DM,

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 –2,147483648 and 2,147483647.

Note Refer to page 115 for general precautions on operand data areas.

Flags

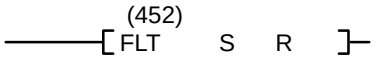
ER (A50003): S is not floating-point data.
Floating-point data is not between –2,147483648 to 2,147483647.
The content of a∗DM word is not BCD when set for BCD.

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) (CVM1 V2)

Ladder Symbol



Operand Data Areas

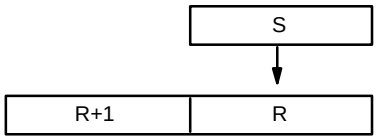
S: Source word	CIO, G, A, T, C, #, DM, DR, IR
R: First result word	CIO, G, A, DM

Variations

↑FLT(452)

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

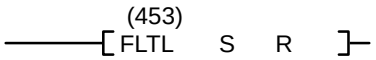
Refer to page 115 for general precautions on operand data areas.

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) (CVM1 V2)

Ladder Symbol



Operand Data Areas

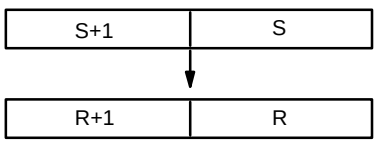
S: First source word	CIO, G, A, T, C, #, DM,
R: First result word	CIO, G, A, DM

Variations

↑FLTL(453)

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

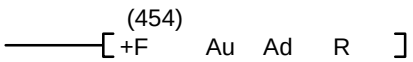
Refer to page 115 for general precautions on operand data areas.

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) (CVM1 V2)

Ladder Symbol



Variations

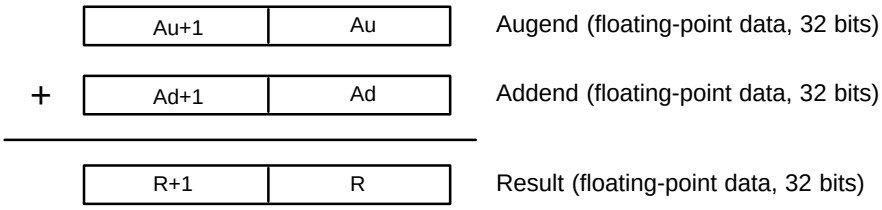
↑+F(454)

Operand Data Areas

Au: First augend word	CIO, G, A, T, C, #, DM
Ad: First addend word	CIO, G, A, T, C, #, DM
R: First result word	CIO, G, A, DM

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 ±□.

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.

	Augend				
Addend	**0**	**Numeral**	**+□**	**–□**	**NaN**
0	0	Numeral	+□	–□	
Numeral	Numeral	See note 1.	+□	–□	
+□	+□	+□	+□	ER	
–□	–□	–□	ER	–□	
NaN					ER

Note 1. The results could be zero (including underflows), a numeral, +∞, or –∞.
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 115 for general precautions on operand data areas.

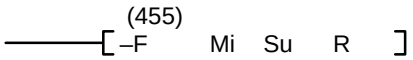
Flags

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) (CVM1 V2)

Ladder Symbol



Variations

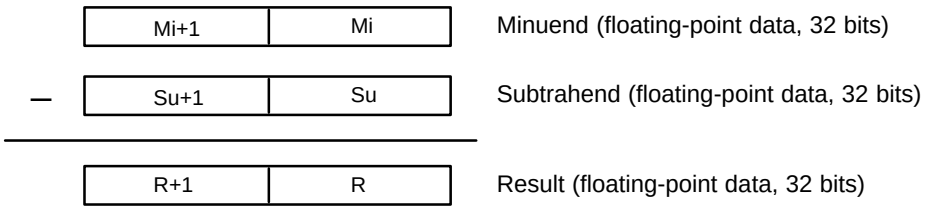
↑–F(455)

Operand Data Areas

Mi: First minuend word	CIO, G, A, T, C, #, DM
Su: First subtrahend word	CIO, G, A, T, C, #, DM
R: First result word	CIO, G, A, DM

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 ±□.

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.

	Minuend				
Subtrahend	**0**	**Numeral**	**+□**	**–□**	**NaN**
0	0	Numeral	+□	–□	
Numeral	Numeral	See note 1.	+□	–□	
+□	–□	–□	ER	–□	
–□	+□	+□	+□	ER	
NaN					ER

Note 1. The results could be zero (including underflows), a numeral, +∞, or –∞.
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 115 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.

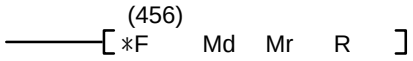
N (A50008): The result is a negative number.

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) (CVM1 V2)

Ladder Symbol



Variations

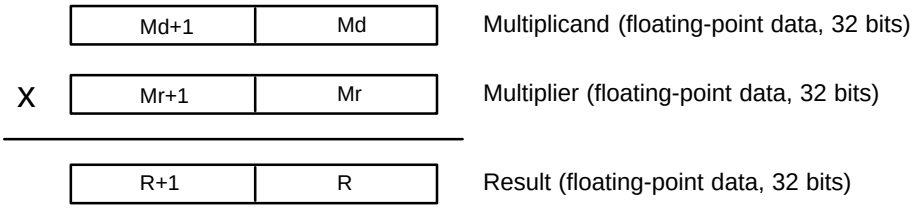
↑∗F(456)

Operand Data Areas

Md: First multiplicand word	CIO, G, A, T, C, #, DM
Mr: First multiplier word	CIO, G, A, T, C, #, DM
R: First result word	CIO, G, A, DM

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 ±□.

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.

	Multiplicand				
Multiplier	**0**	**Numeral**	**+□**	**–□**	**NaN**
0	0	0	ER	ER	
Numeral	0	See note 1.	+/–□	+/–□	
+□	ER	+/–□	+□	–□	
–□	ER	+/–□	–□	+□	
NaN					ER

Note 1. The results could be zero (including underflows), a numeral, +∞, or –∞.

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 115 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) (CVM1 V2)

Ladder Symbol

(457)
/F Dd Dr R

Variations

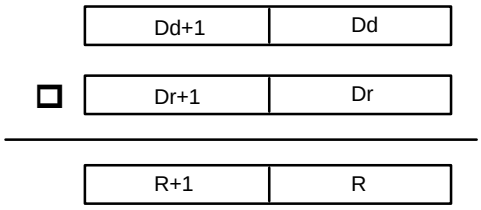
↑/F(457)

Operand Data Areas

Dd: First dividend word	CIO, G, A, T, C, #, DM
Dr: First divisor word	CIO, G, A, T, C, #, DM
R: First result word	CIO, G, A, DM

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.

	Dividend				
Divisor	**0**	**Numeral**	**+□**	**–□**	**NaN**
0	ER	+/–□	+□	–□	
Numeral	0	See note 2.	+/–□	+/–□	
+□	0	0 (see note 1)	ER	ER	
–□	0	0*	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 115 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.

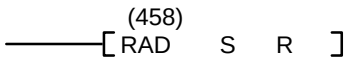
N (A50008): The result is a negative number.

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) (CVM1 V2)

Ladder Symbol



Variations

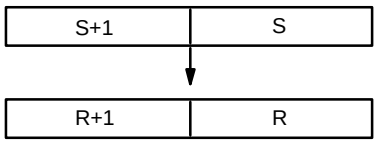
↑RAD(458)

Operand Data Areas

S: First source word	CIO, G, A, T, C, #, DM
R: First result word	CIO, G, A, DM

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.



Source (degrees, floating-point data, 32 bits)

Result (radians, floating-point data, 32 bits)

Degrees are converted to radians by means of the following formula:

Degrees x π/180 = 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 ±□.

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+1must be floating-point data.

Note Refer to page 115 for general precautions on operand data areas.

Flags

ER (A50003): S and S+1is 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.

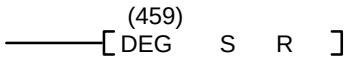
N (A50008): The result is a negative number.

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) (CVM1 V2)

Ladder Symbol



Variations

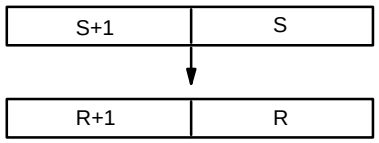
↑RAD(459)

Operand Data Areas

S: First source word	CIO, G, A, T, C, #, DM
R: First result word	CIO, G, A, DM

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.



Source (radians, floating-point data, 32 bits)

Result (degrees, floating-point data, 32 bits)

Degrees are converted to radians by means of the following formula:

Radians x 180/π = 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 ±□.

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+1must be floating-point data.

Note Refer to page 115 for general precautions on operand data areas.

Flags

ER (A50003):	S and S+1is 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) (CVM1 V2)

Ladder Symbol

(460)
SIN S R

Variations

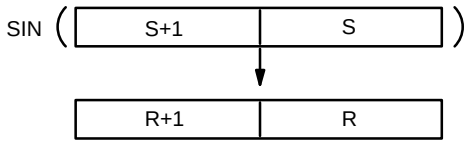
↑SIN(460)

Operand Data Areas

S: First source word	CIO, G, A, T, C, #, DM
R: First result word	CIO, G, A, DM

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.

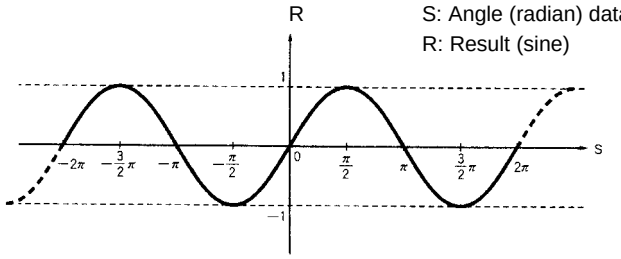


Source (floating-point data, 32 bits)

Result (sine value, floating-point data, 32 bits)

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+1must be floating-point data and its absolute value of must be less than 65,536.

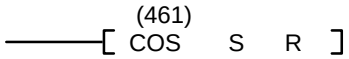
Note Refer to page 115 for general precautions on operand data areas.

Flags

ER (A50003):	The absolute value of S and S+1is 65,536 or greater.
	S and S+1is 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) (CVM1 V2)

Ladder Symbol



Variations

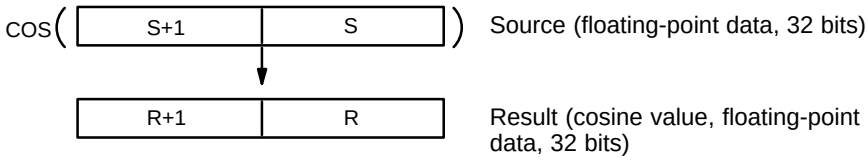
↑COS(461)

Operand Data Areas

S: First source word	CIO, G, A, T, C, #, DM
R: First result word	CIO, G, A, DM

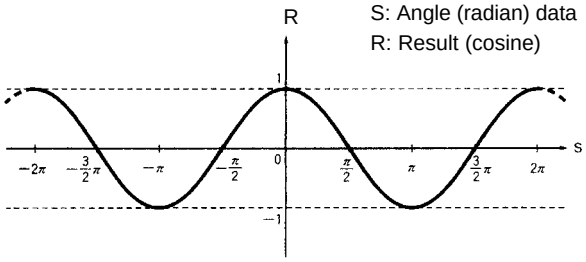
Description

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+1must be floating-point data and its absolute value must be less than 65,536.

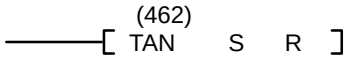
Note Refer to page 115 for general precautions on operand data areas.

Flags

ER (A50003):	The absolute value of S and S+1is 65,536 or greater.
	S and S+1is 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) (CVM1 V2)

Ladder Symbol



Variations

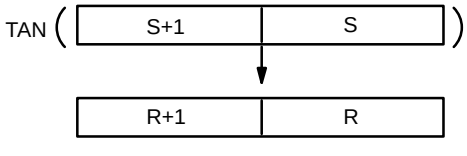
↑TAN(462)

Operand Data Areas

S: First source word	CIO, G, A, T, C, #, DM
R: First result word	CIO, G, A, DM

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.



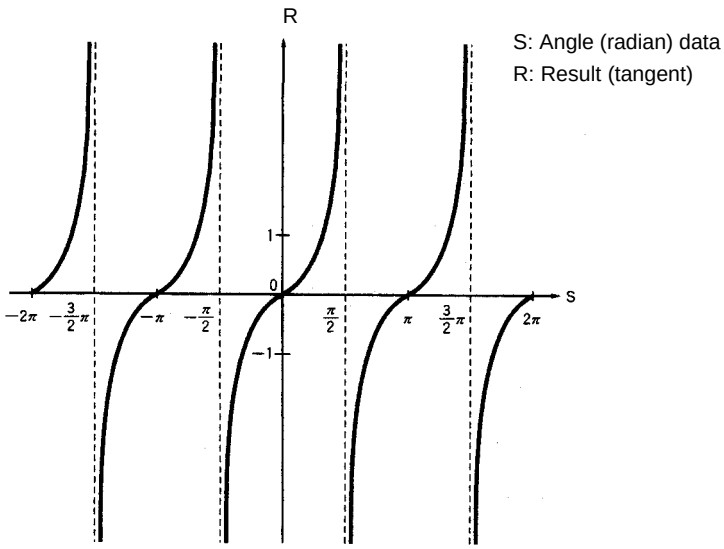
Source (floating-point data, 32 bits)

Result (tangent value, floating-point data, 32 bits)

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



S: Angle (radian) data
R: Result (tangent)

Precautions

S and S+1must be floating-point data and its absolute value must be less than 65,536.

Note Refer to page 115 for general precautions on operand data areas.

Flags

ER (A50003): The absolute value of S and S+1is 65,536 or greater.
S and S+1is 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) (CVM1 V2)

Ladder Symbol

(463)
ASIN S R

Operand Data Areas

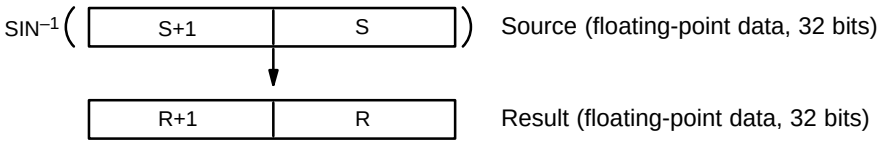
S: First source word	CIO, G, A, T, C, #, DM
R: First result word	CIO, G, A, DM

Variations

↑ASIN(463)

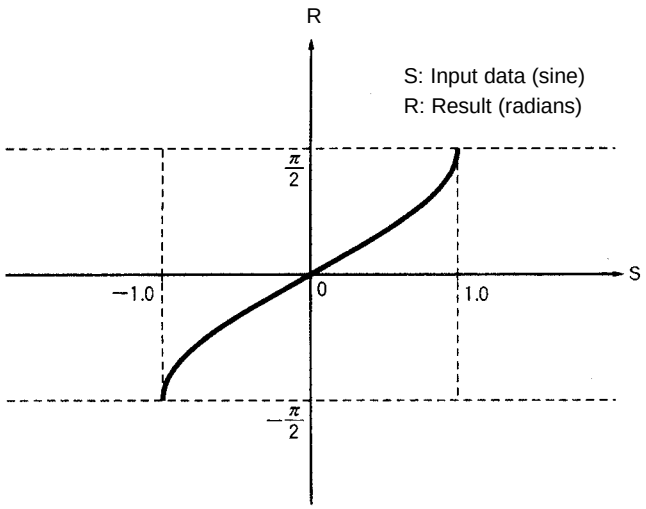
Description

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 115 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) (CVM1 V2)

Ladder Symbol

```
(464)
ACOS  S  R
```

Operand Data Areas

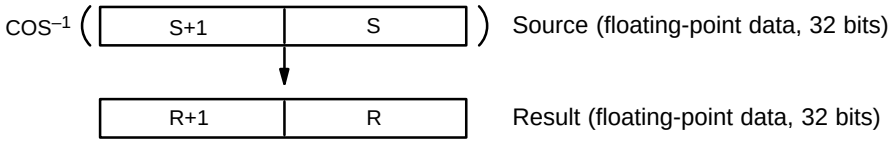
S: First source word	CIO, G, A, T, C, #, DM
R: First result word	CIO, G, A, DM

Variations

↑ACOS(464)

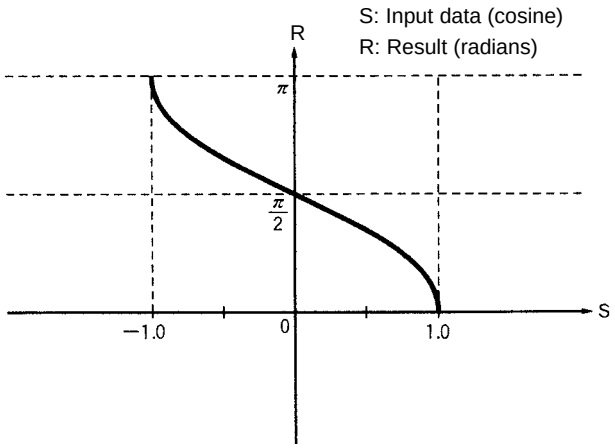
Description

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 π.

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 115 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) (CVM1 V2)

Ladder Symbol

(465)
ATAN S R

Variations

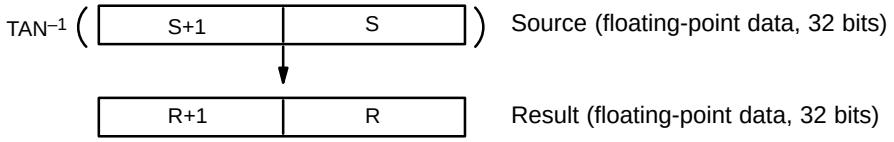
↑ATAN(465)

Operand Data Areas

S: First source word	CIO, G, A, T, C, #, DM
R: First result word	CIO, G, A, DM

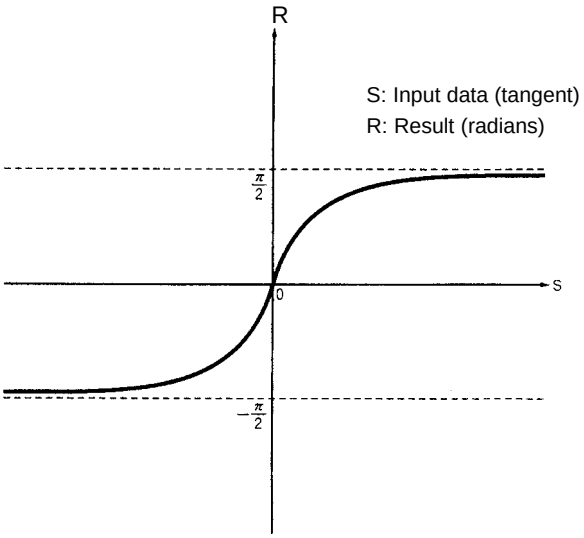
Description

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 115 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) (CVM1 V2)

Ladder Symbol

(466)
SQRT S R

Variations

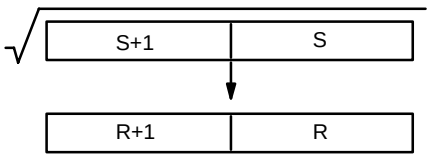
↑SQRT(466)

Operand Data Areas

S: First source word	CIO, G, A, T, C, #, DM
R: First result word	CIO, G, A, DM

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.

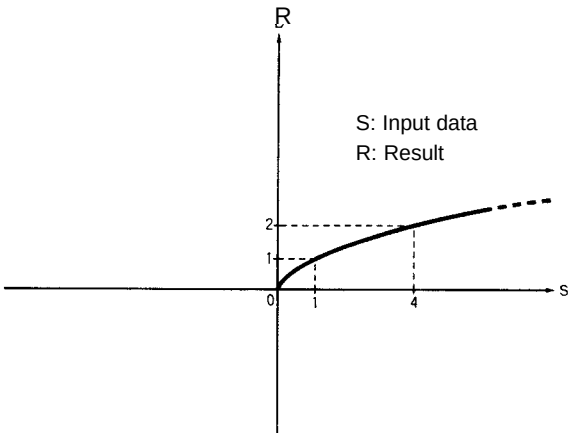


Source (floating-point data, 32 bits)

Result (square root, floating-point data, 32 bits)

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+1must be non-negative floating-point data.

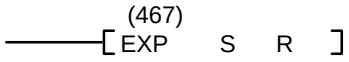
Note Refer to page 115 for general precautions on operand data areas.

Flags

ER (A50003):	S and S+1is a negative number.
	S and S+1is 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) (CVM1 V2)

Ladder Symbol



Variations

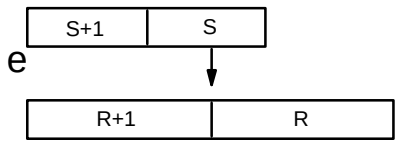
↑EXP(467)

Operand Data Areas

S: First source word	CIO, G, A, T, C, #, DM
R: First result word	CIO, G, A, DM

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).



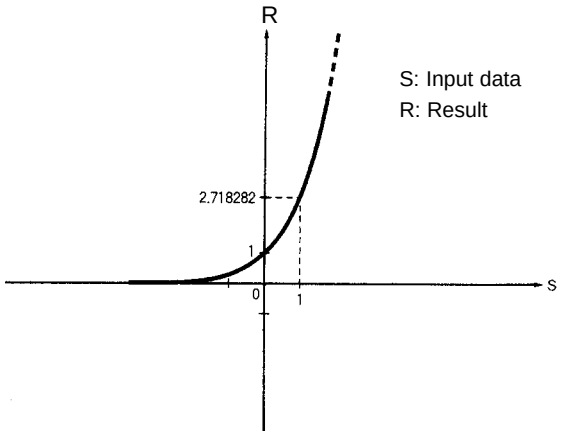
Source (floating-point data, 32 bits)

Result (exponent, floating-point data, 32 bits)

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.

Relation Between Input Data and Result



Precautions

S and S+1must be floating-point data.

Note Refer to page 115 for general precautions on operand data areas.

Flags

ER (A50003):	S and S+1is 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) (CVM1 V2)

Ladder Symbol

(468)
LOG S R

Operand Data Areas

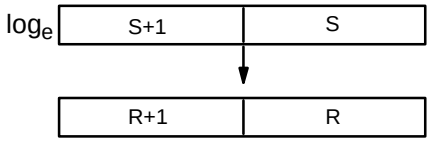
S: First source word	CIO, G, A, T, C, #, DM
R: First result word	CIO, G, A, DM

Variations

↑LOG(468)

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).

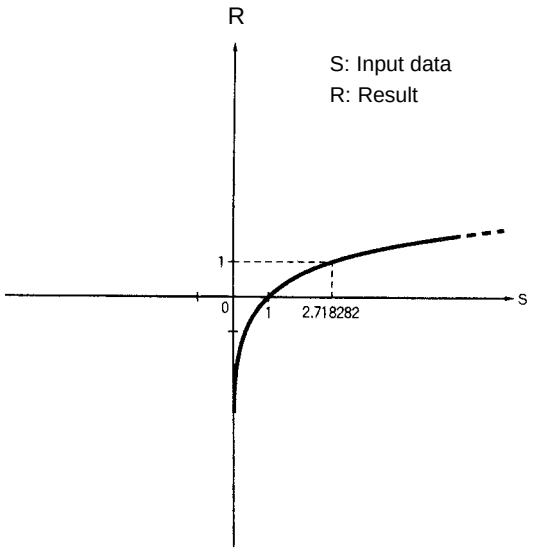


Source (floating-point data, 32 bits)

Result (exponent, floating-point data, 32 bits)

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+1must be non-negative floating-point data.

Note Refer to page 115 for general precautions on operand data areas.

Flags

ER (A50003):	S and S+1is a negative number. S and S+1is 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)

Ladder Symbol

(090)
INC Wd

Operand Data Area

Wd: Word CIO, G, A, DM, DR, IR

Variations

↑ 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.

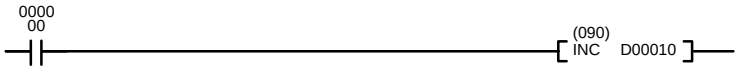
Note Refer to page 115 for general precautions on operand data areas.

Flags

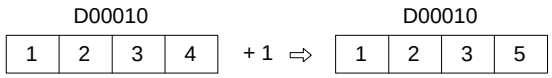
ER (A50003): Wd is not BCD
Content of ∗DM word is not BCD when set for BCD.

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)

Ladder Symbol

(091)
DEC Wd

Operand Data Area

Wd: Word CIO, G, A, DM, DR, IR

Variations

↑ 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 115 for general precautions on operand data areas.

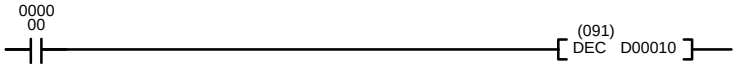
Flags

ER (A50003): Wd is not BCD
Content of ∗DM word is not BCD when set for BCD.

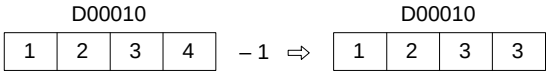
EQ (A50006): The result is 0.

Example

When CIO 000000 is ON in the following example, the content of D00010 is decremented by 1 as a BCD value.

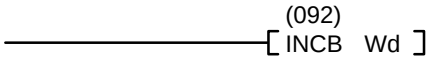


Address	Instruction	Operands
00000	LD	000000
00001	DEC(091)	
		D00010



5-22-3 INCREMENT BINARY: INCB(092)

Ladder Symbol



Operand Data Area

Wd: Word	CIO, G, A, DM, DR, IR

Variations

↑ 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 115 for general precautions on operand data areas.

Flags

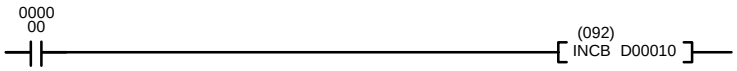
ER (A50003): Content of ∗DM word is not BCD when set for BCD.

EQ (A50006): The result is 0.

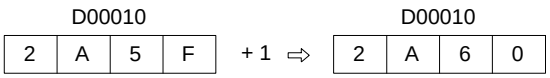
N (A50008): Shows the status of bit 15 of Wd after execution.

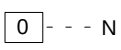
Example

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





5-22-4 DECREMENT BINARY: DECB(093)

Ladder Symbol	Operand Data Area	
(093) DECB Wd	**Wd: Word**	CIO, G, A, DM, DR, IR

Variations

↑ DECB(093)

Description When the execution condition is OFF, DECB(093) is not executed. When the execution condition is ON, DECB(093) decrements Wd, without affecting carry (CY). DECB(093) works the same way as DEC(091) except that it decrements a binary value instead of a BCD value.

Precautions Refer to page 115 for general precautions on operand data areas.

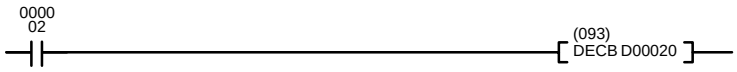
Flags

ER (A50003): Content of ∗DM word is not BCD when set for BCD.

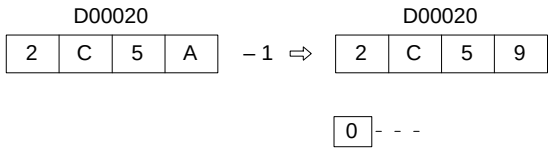
EQ (A50006): The result is 0.

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 D00020 is decremented by 1 as a binary value.



Address	Instruction	Operands
00000	LD	000002
00001	DECB(093)	
		D00020



5-22-5 DOUBLE INCREMENT BCD: INCL(094)

Ladder Symbol	Operand Data Area	
(094) INCL Wd	**Wd: Word**	CIO, G, A, DM

Variations

↑ INCL(094)

Description When the execution condition is OFF, INCL(094) is not executed. When the execution condition is ON, INCL(094) increments the 8-digit BCD number contained in Wd+1 and Wd, without affecting carry (CY). Wd+1 contains the 10^4, 10^5, 10^6, and 10^7 digits.

Precautions Wd and Wd+1 must be BCD.

Note Refer to page 115 for general precautions on operand data areas.

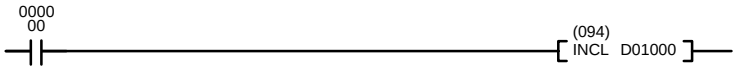
Flags

ER (A50003): Wd or Wd+1 is not BCD

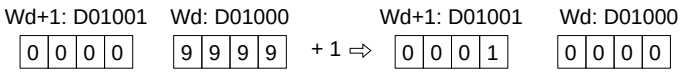
Content of ∗DM word is not BCD when set for BCD.

EQ (A50006): The result is 0.

Example When CIO 000000 is ON in the following example, the content of D0100 and D01001 is incremented by 1 as a BCD value.

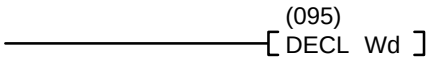


Address	Instruction	Operands
00000	LD	000000
00001	INCL(094)	
		D01000



5-22-6 DOUBLE DECREMENT BCD: DECL(095)

Ladder Symbol



Operand Data Area

Wd: Word CIO, G, A, DM

Variations

↑ DECL(095)

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 10^4, 10^5, 10^6, and 10^7 digits.

Precautions Wd and Wd+1 must be BCD.

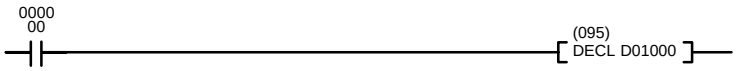
Note Refer to page 115 for general precautions on operand data areas.

Flags

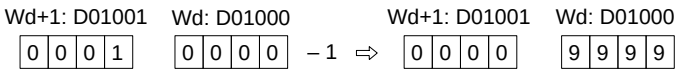
ER (A50003): Wd or Wd+1 is not BCD
Content of ∗DM word is not BCD when set for BCD.

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.

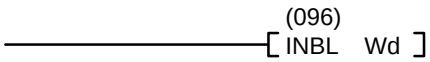


Address	Instruction	Operands
00000	LD	000000
00001	DECL(095)	
		D01000



5-22-7 DOUBLE INCREMENT BINARY: INBL(096)

Ladder Symbol



Operand Data Area

Wd: Word CIO, G, A, DM

Variations

↑ INBL(096)

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 16^4, 16^5, 16^6, and 16^7 digits.

Precautions Refer to page 115 for general precautions on operand data areas.

Flags

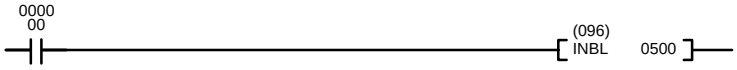
ER (A50003): Content of ∗DM word is not BCD when set for BCD.

EQ (A50006): The result is 0.

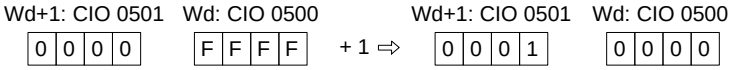
N (A50008): Shows the status of bit 15 of Wd+1 after execution.

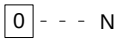
Example

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.



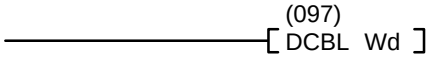
Address	Instruction	Operands
00000	LD	000000
00001	INBL(096)	
		0500





5-22-8 DOUBLE DECREMENT BINARY: DCBL(097)

Ladder Symbol



Operand Data Area

Wd: Word CIO, G, A, DM

Variations

↑ DCBL(097)

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 16^4, 16^5, 16^6, and 16^7 digits.

Precautions

Refer to page 115 for general precautions on operand data areas.

Flags

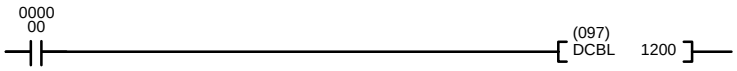
ER (A50003): Content of ∗DM word is not BCD when set for BCD.

EQ (A50006): The result is 0.

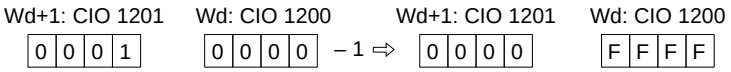
N (A50008): Shows the status of bit 15 of Wd+1 after execution.

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.



Address	Instruction	Operands
00000	LD	000000
00001	DCBL(097)	
		1200



5-23 Special Math Instructions

The Special Math Instructions preform 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 float-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

(165)
MAX C R_1 D

Variations

↑ MAX(165)

Operand Data Areas

C: Control word	CIO, G, A, #, DM, DR, IR
R_1: 1st word in range	CIO, G, A, T, C, DM,
D: Destination word	CIO, G, A, DM, DR, IR

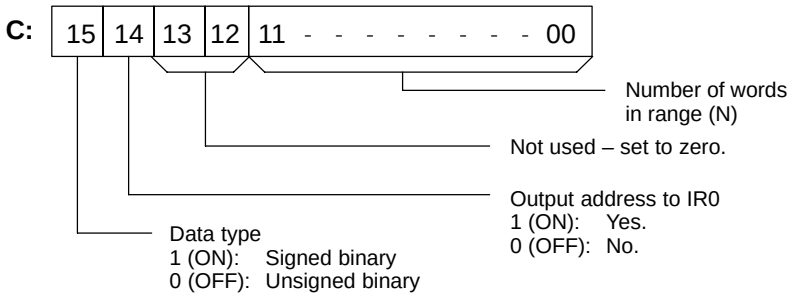
Description

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_1 to R_1+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 115 for general precautions on operand data areas.

Flags

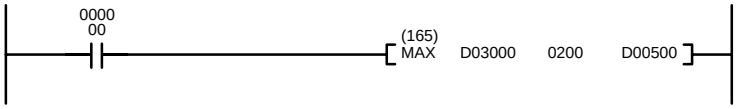
ER (A50003): The 3 rightmost digits of C are not BCD between 001 and 999.
Content of ∗DM word is not BCD when set for BCD.

EQ (A50006): The maximum value is zero.

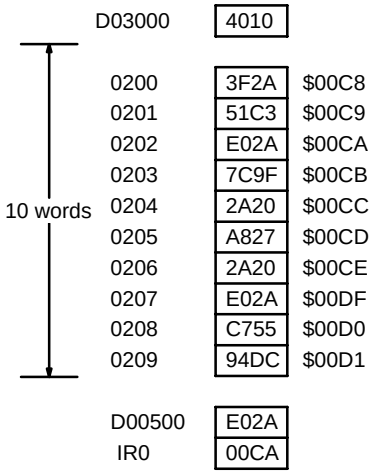
N (A50008): Shows the status of bit 15 of the maximum value.

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.

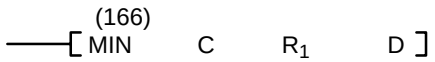


Address	Instruction	Operands
00000	LD	000000
00001	MAX(165)	
		D03000
		0200
		D00500



5-23-2 FIND MINIMUM: MIN(166)

Ladder Symbol



Variations

↑ MIN(166)a

Operand Data Areas

C: Control word	CIO, G, A, #, DM, DR, IR
R_1: 1st word in range	CIO, G, A, T, C, DM
D: Destination word	CIO, G, A, DM, DR, IR

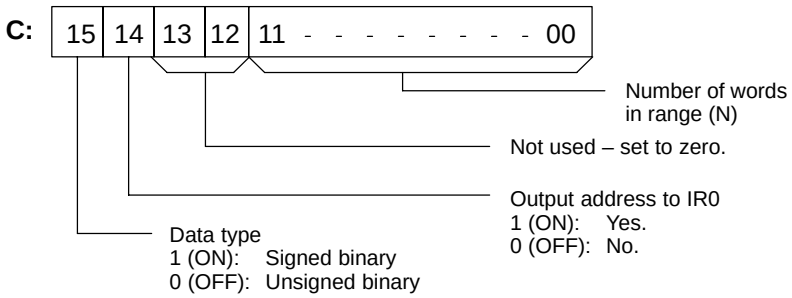
Description

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_1 to R_1+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 115 for general precautions on operand data areas.

Flags

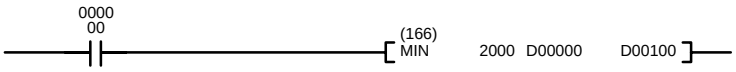
ER (A50003): The 3 rightmost digits of C are not BCD between 001 and 999.
Content of ∗DM word is not BCD when set for BCD.

EQ (A50006): The minimum value is zero.

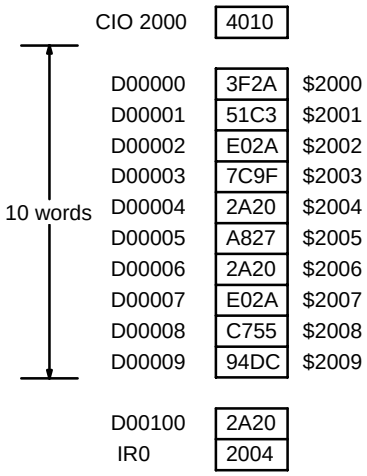
N (A50008): Shows the status of bit 15 of the minimum value.

Example

When CIO 000000 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.

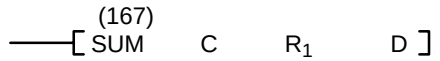


Address	Instruction	Operands
00000	LD	000000
00001	MIN(166)	
		2000
		D00000
		D00200



5-23-3 SUM: SUM(167)

Ladder Symbol



Variations

↑ SUM(167)

Operand Data Areas

C: Control word	CIO, G, A, #, DM, DR, IR
R_1: 1st word in range	CIO, G, A, T, C, DM
D: 1st destination word	CIO, G, A, DM

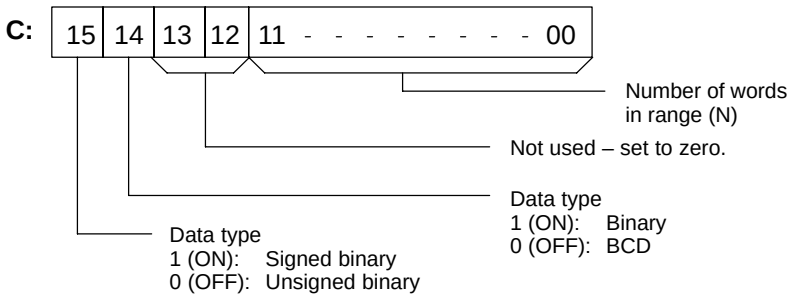
Description

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_1 to R_1+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_1 and R_1+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_1 to R_1+N–1 must be BCD.

Note Refer to page 115 for general precautions on operand data areas.

Flags

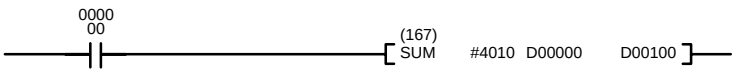
ER (A50003): The 3 rightmost digits of C are not BCD between 001 and 999.
Bit 14 of C is OFF, indicating BCD data, but the content of a word within the range is not BCD.
Content of ∗DM word is not BCD when set for BCD.

EQ (A50006): The result is zero.

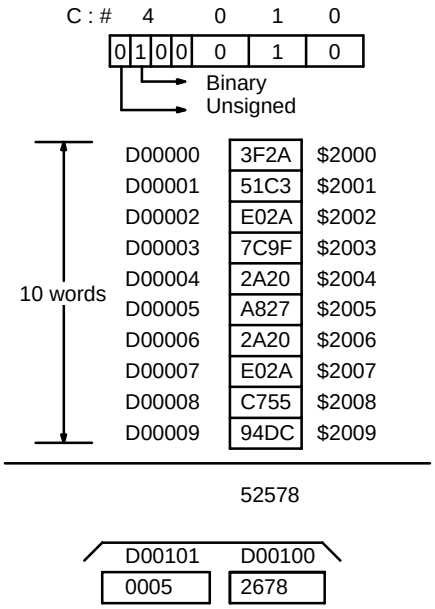
N (A50008): Shows the status of bit 15 of D+1.

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.

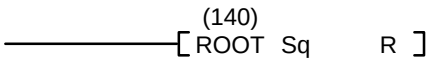


Address	Instruction	Operands
00000	LD	000000
00001	SUM(167)	
		#4010
		D00000
		D00100



5-23-4 BCD SQUARE ROOT: ROOT(140)

Ladder Symbol



Variations

↑ ROOT(140)

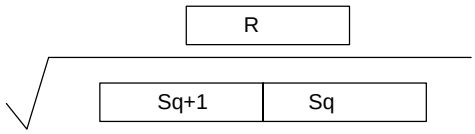
Operand Data Areas

Sq: 1st source word CIO, G, A, T, C, #, DM

R: Result word CIO, G, A, DM, DR, IR

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 115 for general precautions on operand data areas.

Flags

ER (A50003): Sq or the content of ∗DM word is not BCD when set for BCD.

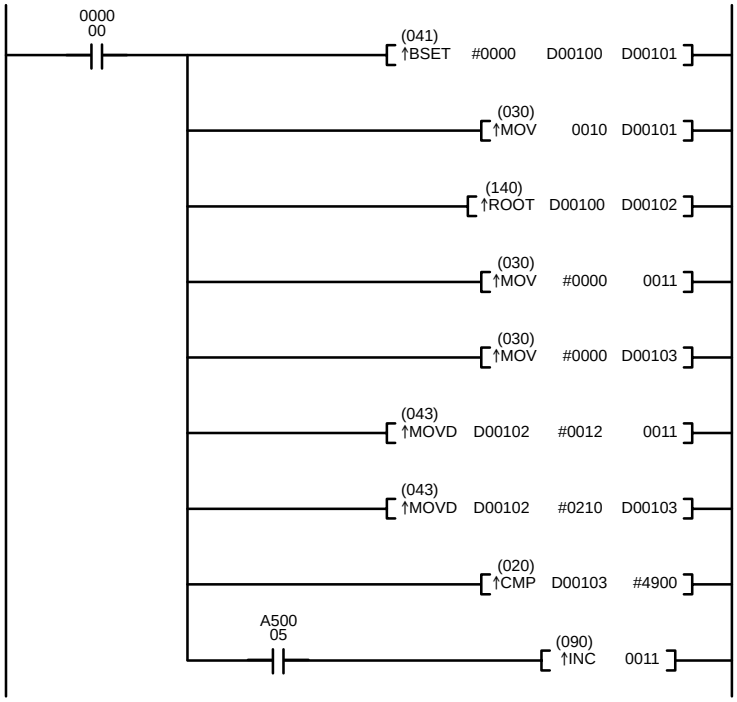
EQ (A50006): ON when the result is 0.

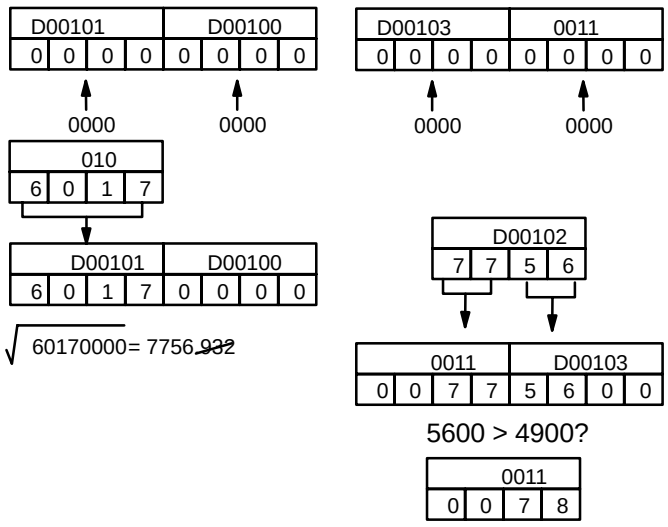
Example

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.

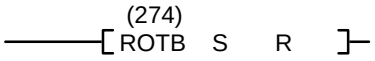




Address	Instruction	Operands
00000	LD	000000
00001	↑BSET(041)	
		#0000
		D00100
		D00101
00002	↑MOV(030)	
		0010
		D00101
00003	↑ROOT(140)	
		D00100
		D00102
00004	↑MOV(030)	
		#0000
		0011
00005	↑MOV(030)	
		#0000
		D00103
00006	↑MOVD(043)	
		D00102
		#0012
		0011
00007	↑MOVD(043)	
		D00102
		#0210
		D00103
00008	↑CMP(020)	
		D00103
		#4900
00009	LD	A50005
00010	↑INC(090)	
		0011

5-23-5 BINARY ROOT: ROTB(274) (CVM1 V2)

Ladder Symbol



Operand Data Areas

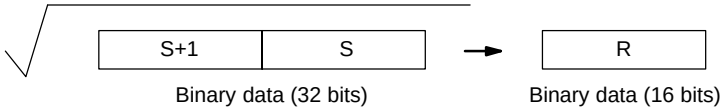
S: First source word	CIO, G, A, T, C, #, DM
R: Result word	CIO, G, A, DM, DR, IR

Variations

↑ROTB(274)

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 115 for general precautions on operand data areas.

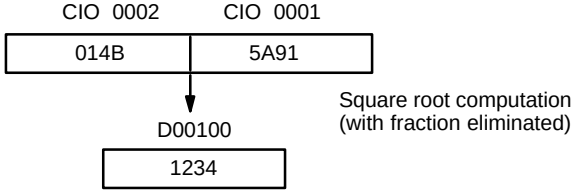
Flags

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)

Ladder Symbol

(141)
FDIV Dd Dr R

Variations

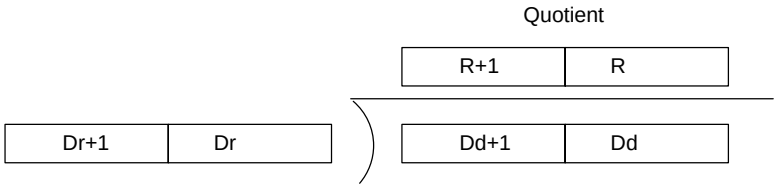
↑ FDIV(141)

Operand Data Areas

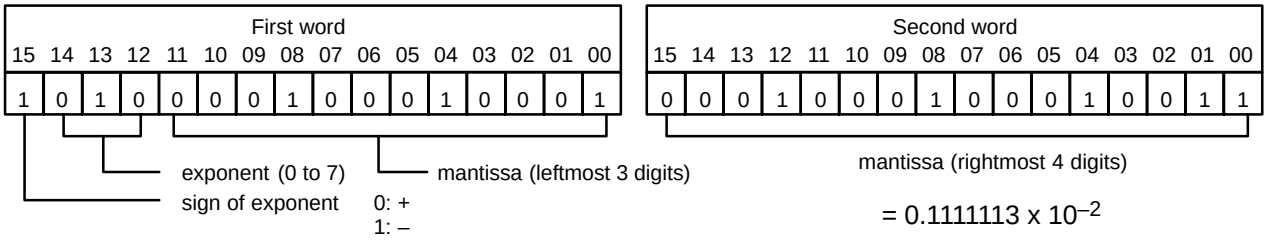
Dd: 1st dividend word	CIO, G, A, T, C, DM
Dr: 1st divisor word	CIO, G, A, T, C, DM
R: 1st result word	CIO, G, A, DM

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 0.0000001×10^{-7} and 0.9999999×10^{7}. The results must be between 0.1×10^{-7} and 0.9999999×10^{7}.

Note Refer to page 115 for general precautions on operand data areas.

Flags

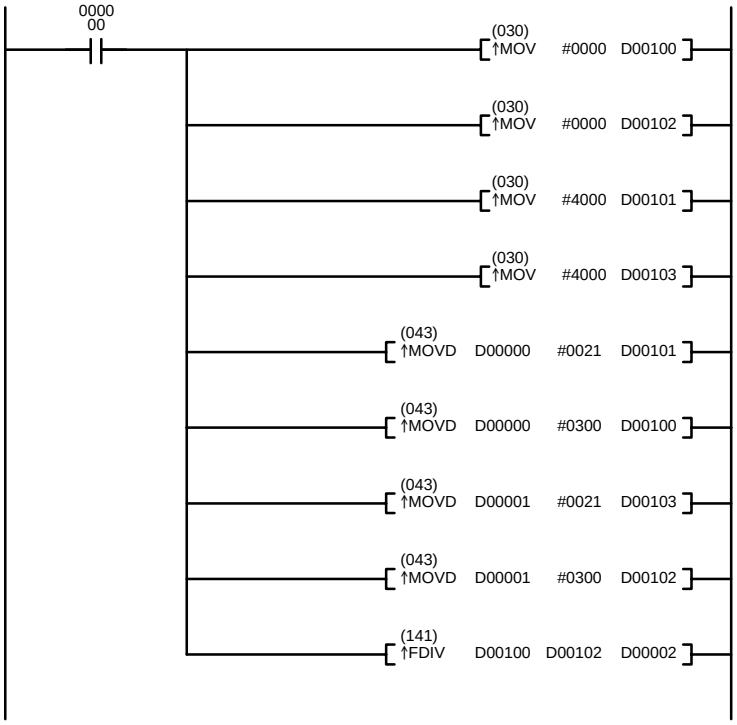
ER (A50003): Dr and Dr+1 contain 0.
Dd, Dd+1, Dr, or Dr+1 is not BCD.
The result is not between 0.1×10^{-7} and 0.999999×10^{7}.
Content of ∗DM word is not BCD when set for BCD.

EQ (A50006): ON when the result is 0.

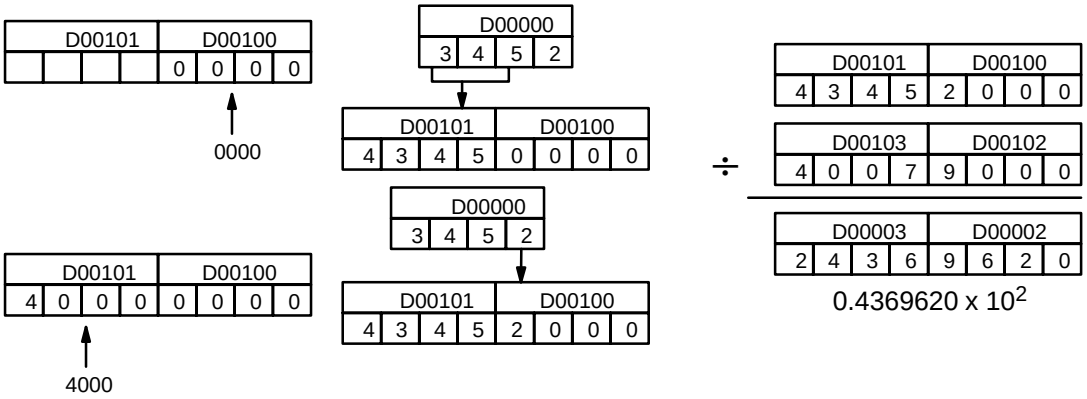
Example

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 0.3452×10^4). 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)

Ladder Symbol

(142)
APR C S R

Variations

↑ APR(142)

Operand Data Areas

C: Control word	CIO, G, A, #, DM, DR, IR
S: Source data	CIO, G, A, T, C, #, DM, DR, IR
R: Result word	CIO, G, A, DM, DR, IR

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 115 for general precautions on operand data areas.

Flags

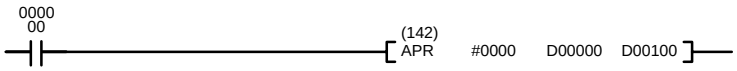
ER (A50003): For trigonometric functions, S is greater than 0900 or not BCD.
For linear extrapolation, S is not BCD when set for BCD or the table is not readable.
Content of *DM word is not BCD when set for BCD.

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	10^1	10^0	10^{-1}
0	3	0	0

Enter input data not exceeding #0900 in BCD form.

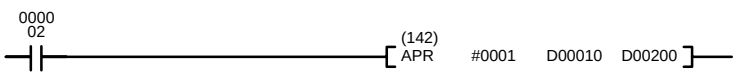
Result

R: D00100			
10^{-1}	10^{-2}	10^{-3}	10^{-4}
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	10^1	10^0	10^{-1}
0	3	0	0

Enter input data not exceeding #0900 in BCD form.

Result

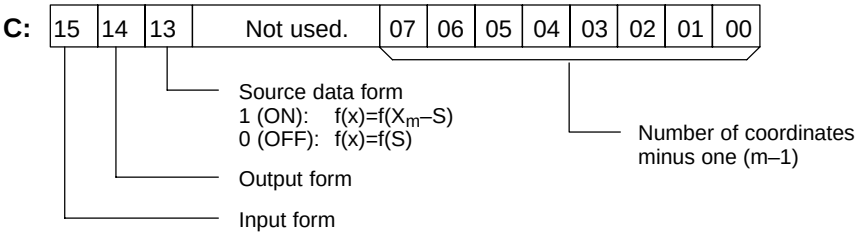
R: D00200			
10^{-1}	10^{-2}	10^{-3}	10^{-4}
8	6	6	0

Result data has four significant digits, fifth and higher digits are ignored. The result for cos(0) will be 0.9999, not 1.

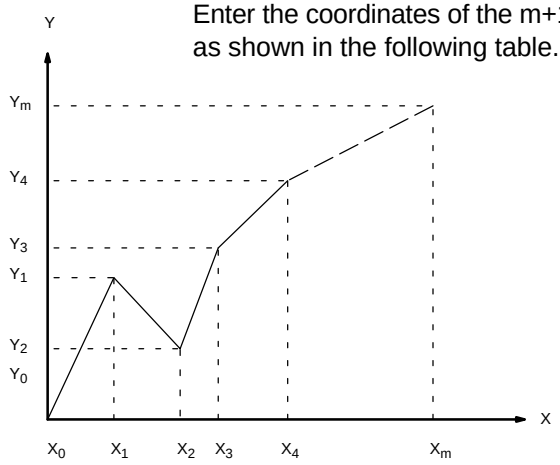
Linear Extrapolation

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_m–S): OFF specifies f(x)=f(S) and ON specifies f(x)=f(X_m–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.

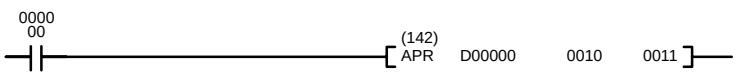


Enter the coordinates of the m+1 end points, which define the m line segments, as shown in the following table. Enter all coordinates in binary form.

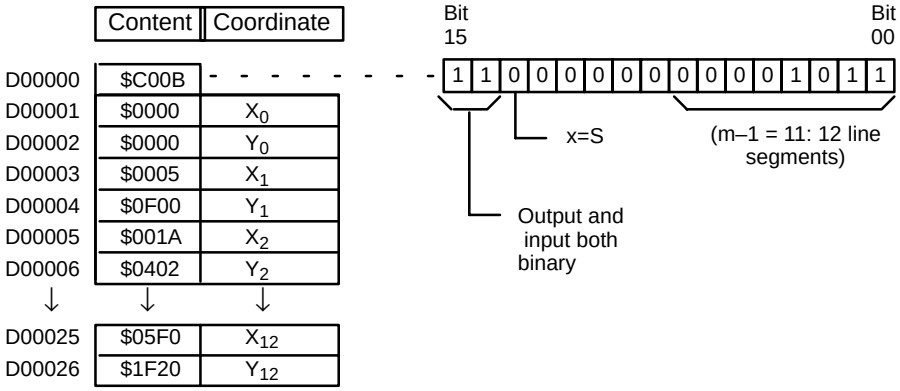


Word	Coordinate
C+1	X_0
C+2	Y_0
C+3	X_1
C+4	Y_1
C+5	X_2
C+6	Y_2
↓	↓
C+(2m+1)	X_m (max. X value)
C+(2m+2)	Y_m

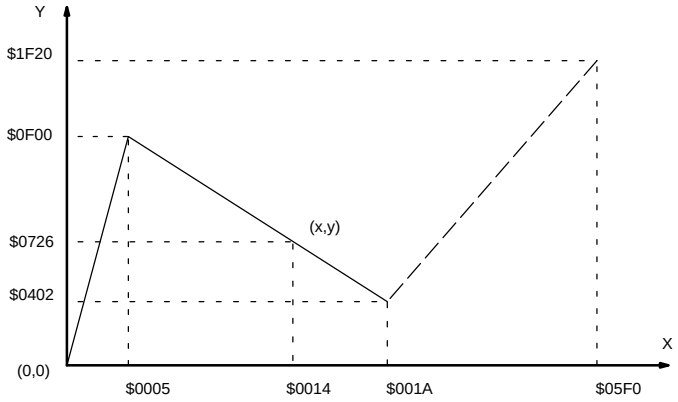
The following example demonstrates the construction of a linear extrapolation with 12 coordinates. The block of data is continuous, as it must be, from D00000 to D00026 (C to C + (2 × 12 + 2)). The input data is taken from CIO 0010, and the result is output to CIO 0011.



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) (CVM1 V2)

Ladder Symbol

(270)
PID S C D

Operand Data Areas

S: Input word	CIO, G, A, DM, DR, IR
C: First parameter word	CIO, G, A, DM,
D: Output word	CIO, G, A, DM, DR, IR

⚠ Caution A total of 33 continuous words starting with P1 must be provided for PID(—) to operate correctly. Also, PID(—) may not operate dependably in any of the following situations: In interrupt programs, 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(—) in these situations.

Description

When the execution condition 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 (τ)			
C+5	2-PID parameter (α)			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 (τ)	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 (α)	The input filter coefficient. Normally use 0.65 (i.e., a setting of 000). The filter efficiency decreases as the coefficient approaches 0.	000: α = 0.65 Setting from 100 to 199 means that the value of the rightmost two digits is set from α= 0.00 to α= 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 2: 10 bits 7: 15 bits 3: 11 bits 8: 16 bits 4: 12 bits
Output range	The number of output data bits. {The number of output bits is automatically the same as the number of input 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 (α) 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 □ lower limit □ upper limit □ 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

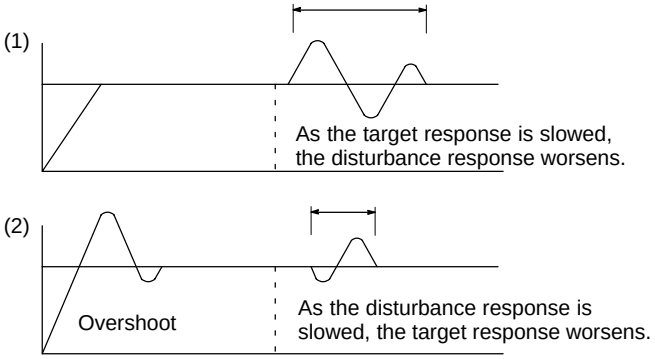
The sampling period is the time interval to retrieve the measurement data for carrying out a PID action. PID(270), however, is executed according to CPU cycle, so there may be cases where the sampling period is exceeded. In such cases, the time interval until the next sampling is reduced.

PID Control Method

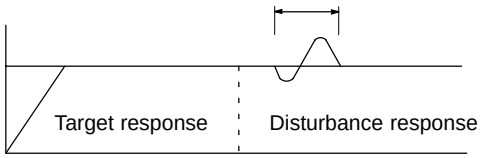
PID control actions are executed by means of PID control with feed-forward control (two degrees of freedom).

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).

Simple PID Control



Feed-forward PID control



Control Actions

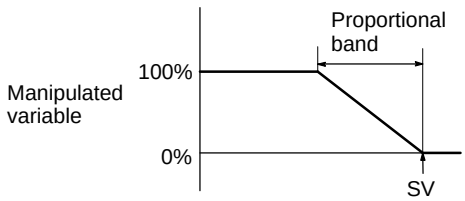
Proportional Action (P)

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

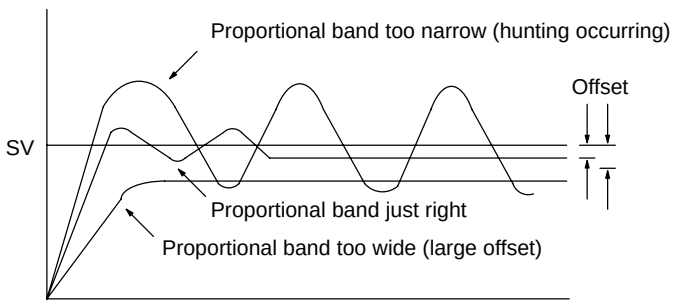
If the proportional action is used and the present value (PV) becomes smaller than the proportional band, the manipulated variable (MV) is 100% (i.e., the maximum value). Within the proportional band, the MV is made proportional to the deviation (the difference between from SV and PV) and gradually decreased 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)



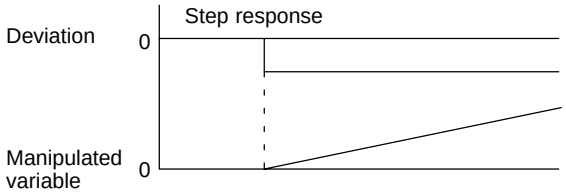
Adjusting the Proportional Band



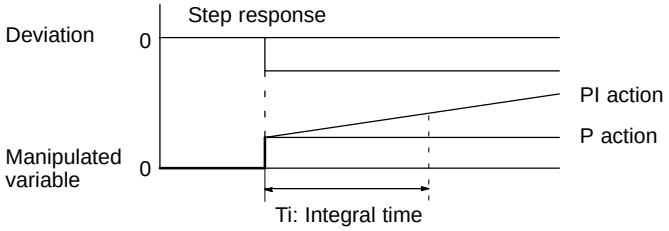
Integral Action (I)

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.

Integral Action



Pi Action and Integral Time

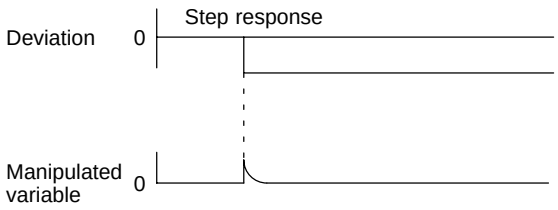


Derivative Action (D)

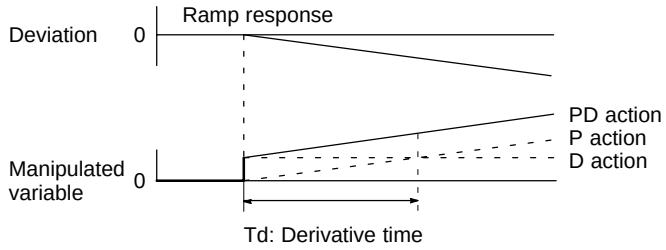
Proportional action and integral action both make corrections with respect to the control results, so there is inevitably a response delay. Derivative action compensates for that drawback. In response to a sudden disturbance it delivers a large manipulated variable and rapidly restores the original status. 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.

Derivative Action



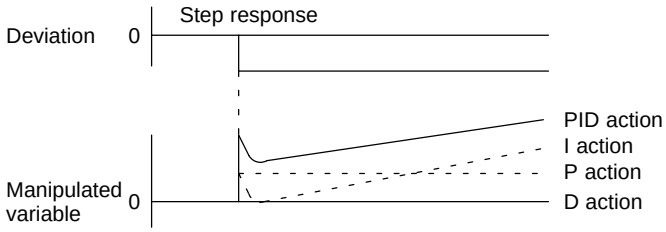
PD Action and Derivative Time



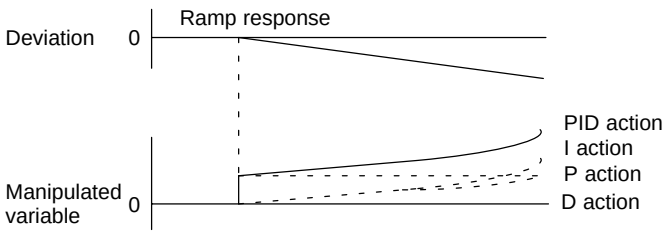
PID Action

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.

Step Response of PID Control Action Output



Ramp Response of PID Control Action Output

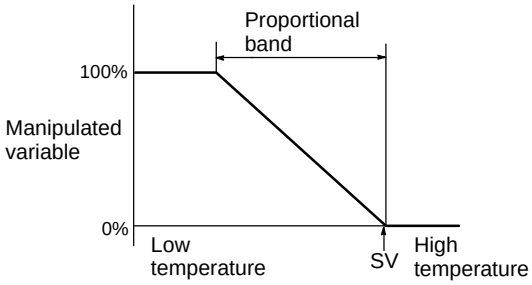


Direction of Action

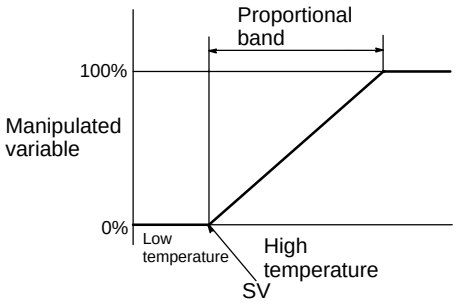
When using PID action, select either of the following two control directions. In either direction, the MV increases as the difference between the SV and the PV increases.

- 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.

Reverse Action



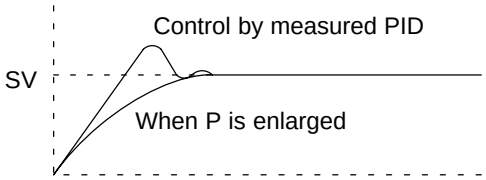
Forward Action



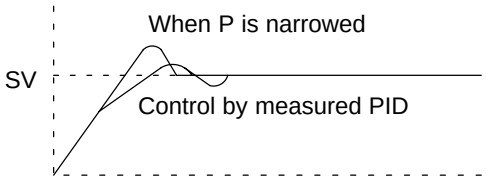
Adjusting PID Parameters

The general relationship between PID parameters and control status is shown below.

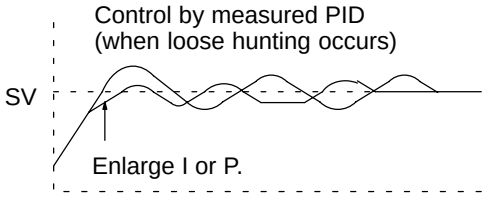
- When it is not a problem if a certain amount of time is required for stabilization (settlement time), but it is important not to cause overshooting, then enlarge the proportional band.



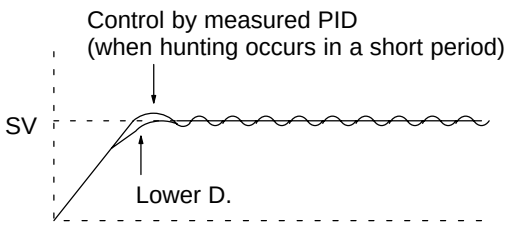
- When overshooting is not a problem but it is desirable to quickly stabilize control, then narrow the proportional band. If the proportional band is narrowed too much, however, then hunting may occur.



- 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 115 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.

5-24-2 LIMIT CONTROL: LMT(271) (CVM1 V2)

Ladder Symbol

(271)
LMT S C D

Variations

↑LMT(271)

Operand Data Areas

S: Input word	CIO, G, A, T, C, #, DM, DR, IR
C: First limit word	CIO, G, A, T, C, DM
D: Output word	CIO, G, A, T, C, DM, DR, IR

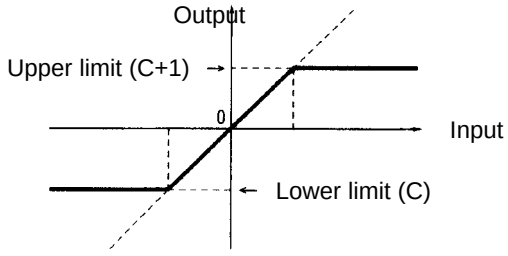
Description When the execution condition is OFF, LMT(271) is not executed. When the execution condition is ON, LMT(271) controls output data according to whether or not the specified input data (signed 16-bit binary) is within the upper and lower limits. The contents of words C and C+1 are as follows:

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 115 for general precautions on operand data areas.

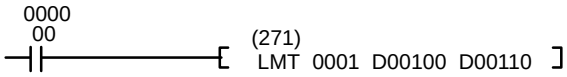
Flags

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.

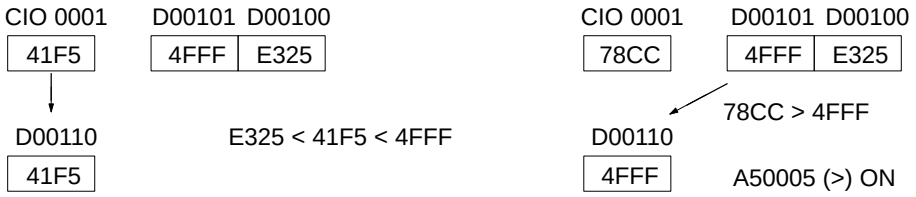
Example

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.

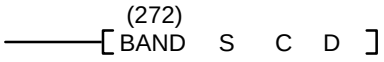


Address	Instruction	Operands
00000	LD	000000
00001	LMT(271)	
		0001
		D00100
		D00110



5-24-3 DEAD-BAND CONTROL: BAND(272) (CVM1 V2)

Ladder Symbol



Variations

↑BAND(272)

Operand Data Areas

S: Input word	CIO, G, A, T, C, #, DM, DR, IR
C: First limit word	CIO, G, A, T, C, DM
D: Output word	CIO, G, A, T, C, DM, DR, IR

Description

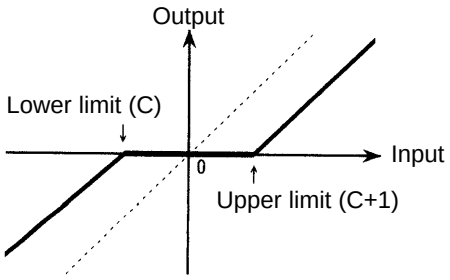
When the execution condition is OFF, BAND(272) is not executed. When the execution condition is ON, BAND(272) controls output data according to whether or not the specified input data (signed 16-bit binary) is within the upper and lower limits (dead band). The contents of words C and C+1 are as follows:

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 115 for general precautions on operand data areas.

Flags

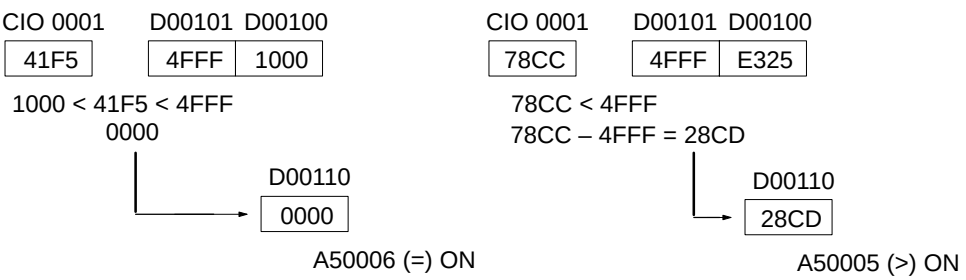
ER (A50003): The upper limit setting is less than the lower limit.
The content of a∗DM word is not BCD when set for BCD.
> (A50005): The input data (S) is greater than the upper limit (C+1).
EQ (A50006): The output data is all zeros.
< (A50007): The input data (S) is less than the lower limit (C).
N (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	000000
00001	BAND(272)	
		0001
		D00100
		D00110



5-24-4 DEAD-ZONE CONTROL: ZONE(273) (CVM1 V2)

Ladder Symbol

(273)
ZONE S C D

Variations

↑ZONE(273)

Operand Data Areas

S: Input word	CIO, G, A, T, C, #, DM, DR, IR
C: First bias word	CIO, G, A, T, C, DM
D: Output word	CIO, G, A, T, C, DM, DR, IR

Description

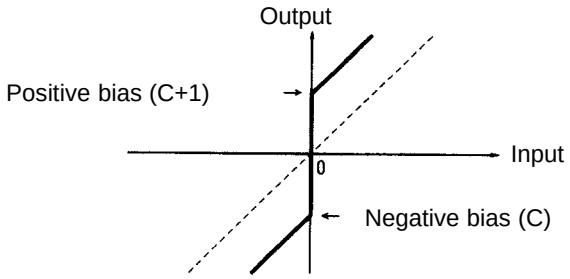
When the execution condition is OFF, ZONE(273) is not executed. When the execution condition is ON, ZONE(273) adds the specified bias to the specified input data (signed 16-bit binary) and places the result in a specified word. The contents of words C and C+1 are as follows:

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:

8000 +FF00 = 7F00
$(-32{,}768)_{10}$ $(-256)_{10}$ $(32{,}512)_{10}$

Precautions

The negative bias (C) must be less than or equal to the positive bias (C+1).

Note Refer to page 115 for general precautions on operand data areas.

Flags

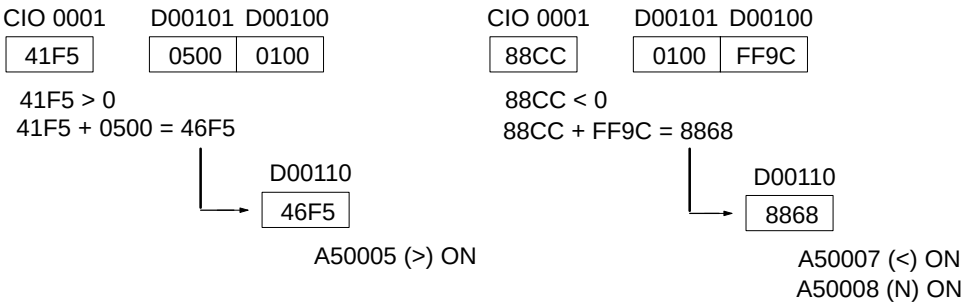
ER (A50003):	C is less than C+1.
	The content of a∗DM word is not BCD when set for BCD.
> (A50005):	The input data (S) is greater than 0000.
EQ (A50006):	The output data is all zeros.
< (A50007):	The input data (S) is less than 0000.
N (A50008):	The output data is a negative number.

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

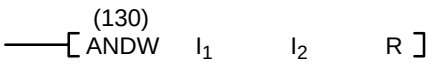


5-25 Logic Instructions

The logic instructions perform logic operations on word data.

5-25-1 LOGICAL AND: ANDW(130)

Ladder Symbol



Variations

↑ ANDW(130)

Operand Data Areas

I_1: Input 1	CIO, G, A, T, C, #, DM, DR, IR
I_2: Input 2	CIO, G, A, T, C, #, DM, DR, IR
R: Result word	CIO, G, A, DM, DR, IR

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_1 and I_2 and places the result in R.

Precautions

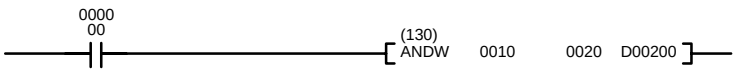
Refer to page 115 for general precautions on operand data areas.

Flags

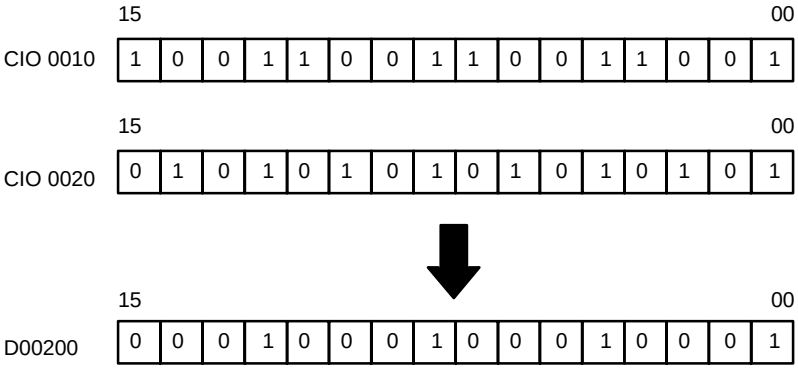
ER (A50003): Content of ∗DM word is not BCD when set for BCD.
EQ (A50006): The result is 0.
N (A50008): Shows the status of bit 15 of R after execution.

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)

Ladder Symbol

(131)
ORW I_1 I_2 R

Variations

↑ ORW(131)

Operand Data Areas

I_1: Input 1	CIO, G, A, T, C, #, DM, DR, IR
I_2: Input 2	CIO, G, A, T, C, #, DM, DR, IR
R: Result word	CIO, G, A, DM, DR, IR

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_1 and I_2 and places the result in R.

Precautions Refer to page 115 for general precautions on operand data areas.

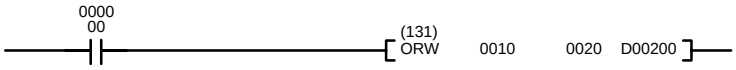
Flags

ER (A50003): Content of ∗DM word is not BCD when set for BCD.

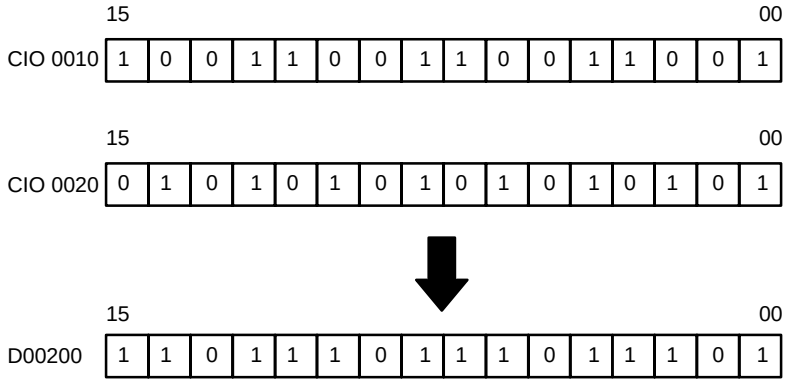
EQ (A50006): The result is 0.

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.

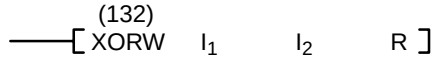


Address	Instruction	Operands
00000	LD	000000
00001	ORW(131)	
		0010
		0020
		D0020



5-25-3 EXCLUSIVE OR: XORW(132)

Ladder Symbol



Variations

↑ XORW(132)

Operand Data Areas

I_1: Input 1	CIO, G, A, T, C, #, DM, DR, IR
I_2: Input 2	CIO, G, A, T, C, #, DM, DR, IR
R: Result word	CIO, G, A, DM, DR, IR

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_1 and I_2 and places the result in R.

Precautions Refer to page 115 for general precautions on operand data areas.

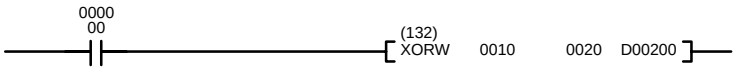
Flags

ER (A50003): Content of ∗DM word is not BCD when set for BCD.

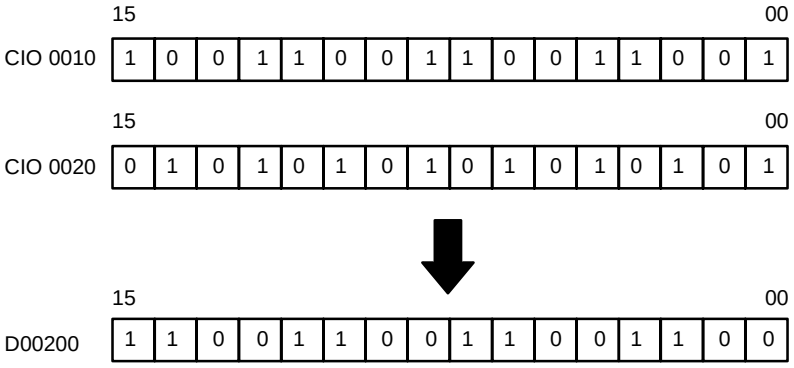
EQ (A50006): The result is 0.

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.

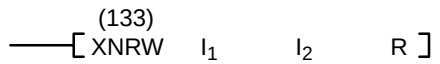


Address	Instruction	Operands
00000	LD	000000
00001	XORW(132)	
		0010
		0020
		D0020



5-25-4 EXCLUSIVE NOR: XNRW(133)

Ladder Symbol



Variations

↑ XNRW(133)

Operand Data Areas

I_1: Input 1	CIO, G, A, T, C, #, DM, DR, IR
I_2: Input 2	CIO, G, A, T, C, #, DM, DR, IR
R: Result word	CIO, G, A, DM, DR, IR

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_1 and I_2 and places the result in R.

Precautions Refer to page 115 for general precautions on operand data areas.

Flags

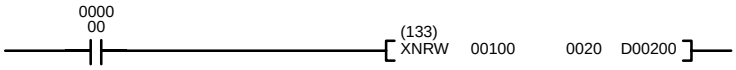
ER (A50003): Content of ∗DM word is not BCD when set for BCD.

EQ (A50006): The result is 0.

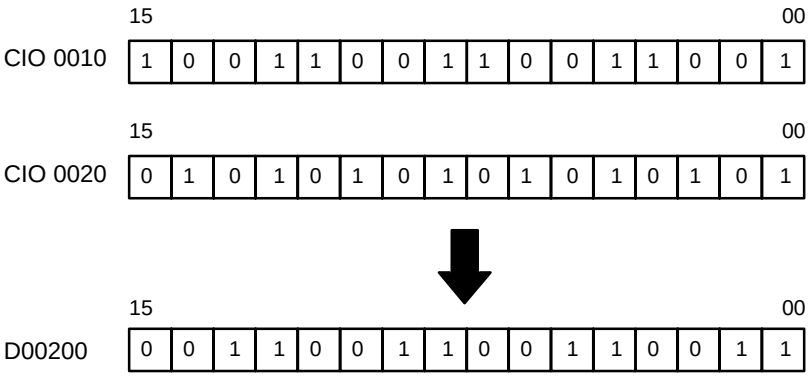
N (A50008): Shows the status of bit 15 of R after execution.

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.

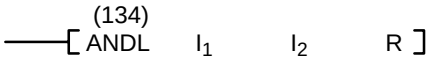


Address	Instruction	Operands
00000	LD	000000
00001	XNRW(133)	
		0010
		0020
		D0020



5-25-5 DOUBLE LOGICAL AND: ANDL(134)

Ladder Symbol



Variations

↑ ANDL(134)

Operand Data Areas

I_1: Input 1	CIO, G, A, T, C, #, DM
I_2: Input 2	CIO, G, A, T, C, #, DM
R: Result word	CIO, G, A, DM

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_1 and I_1+1 with I_2 and I_2+1 and places the result in R and R+1.

Precautions

Refer to page 115 for general precautions on operand data areas.

Flags

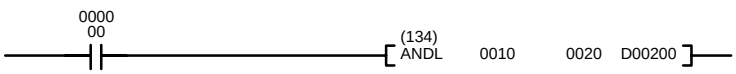
ER (A50003): Content of ∗DM word is not BCD when set for BCD.

EQ (A50006): The result is 0.

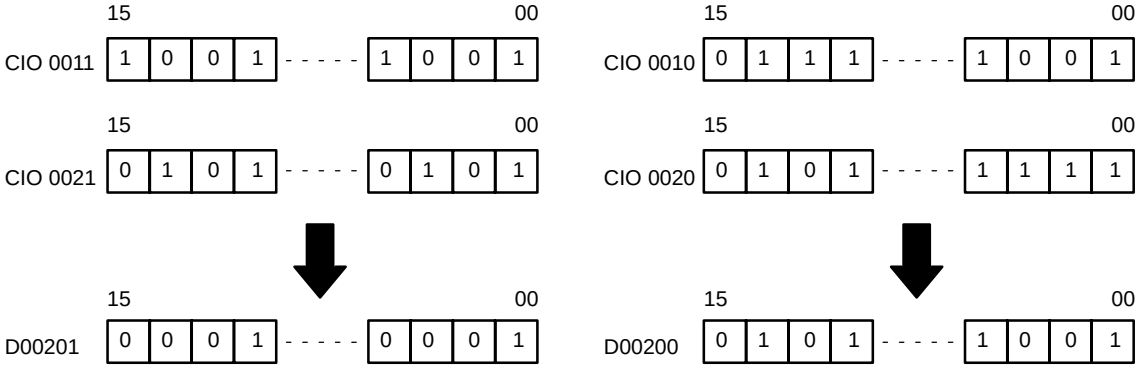
N (A50008): Shows the status of bit 15 of R+1 after execution.

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)

Ladder Symbol

(135)
ORWL I_1 I_2 R

Variations

↑ ORWL(135)

Operand Data Areas

I_1: Input 1	CIO, G, A, T, C, #, DM
I_2: Input 2	CIO, G, A, T, C, #, DM
R: Result word	CIO, G, A, DM

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_1 and I_1+1 with I_2 and I_2+1 and places the result in R and R+1.

Precautions Refer to page 115 for general precautions on operand data areas.

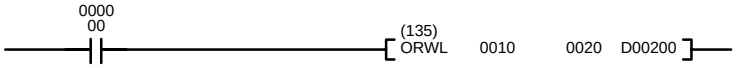
Flags

ER (A50003): Content of ∗DM word is not BCD when set for BCD.

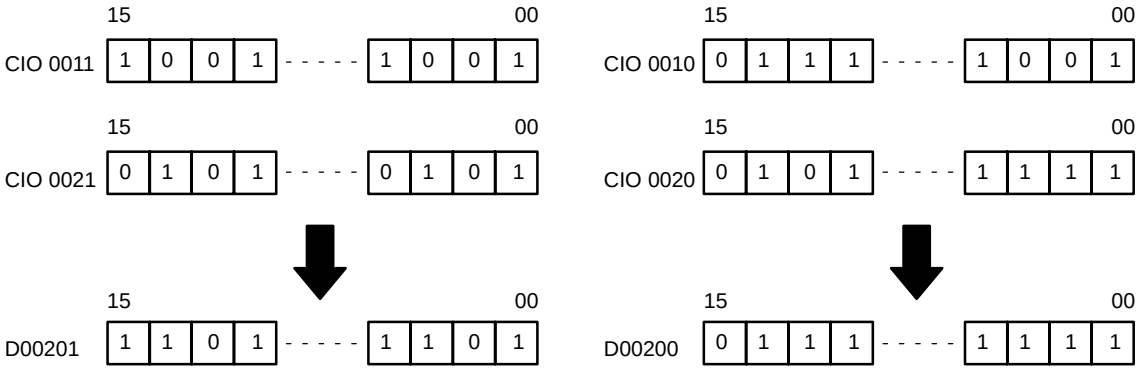
EQ (A50006): The result is 0.

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.



Address	Instruction	Operands
00000	LD	000000
00001	ORWL(135)	
		0010
		0020
		D0020



5-25-7 DOUBLE EXCLUSIVE OR: XORL(136)

Ladder Symbol

(136)
XORL I_1 I_2 R

Variations

↑ XORL(136)

Operand Data Areas

I_1: Input 1	CIO, G, A, T, C, #, DM
I_2: Input 2	CIO, G, A, T, C, #, DM
R: Result word	CIO, G, A, DM

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_1 and I_1+1 with I_2 and I_2+1 bit-by-bit and places the result in R and R+1.

Precautions Refer to page 115 for general precautions on operand data areas.

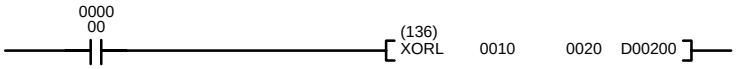
Flags

ER (A50003): Content of ∗DM word is not BCD when set for BCD.

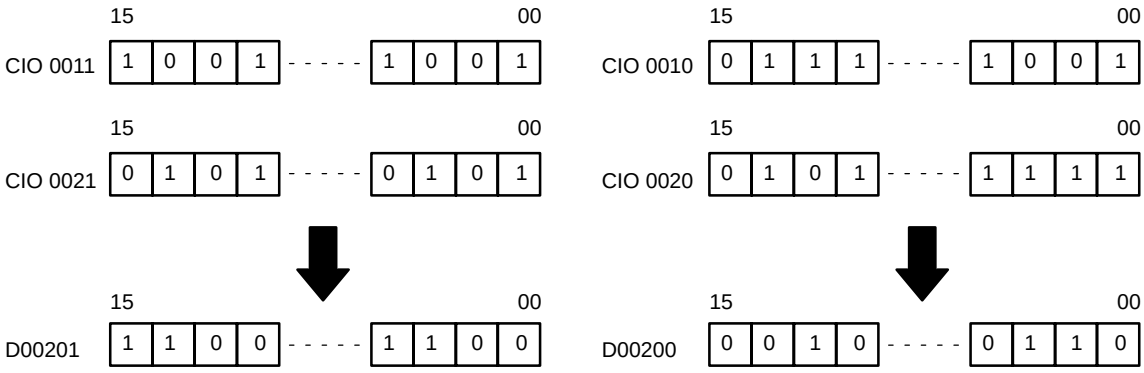
EQ (A50006): The result is 0.

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
		D0020



5-25-8 DOUBLE EXCLUSIVE NOR: XNRL(137)

Ladder Symbol

(137)
XNRL I_1 I_2 R

Variations

↑ XNRL(137)

Operand Data Areas

I_1: Input 1	CIO, G, A, T, C, #, DM
I_2: Input 2	CIO, G, A, T, C, #, DM
R: Result word	CIO, G, A, DM

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_1 and I_1+1 with I_2 and I_2+1 bit-by-bit and places the result in R and R+1.

Precautions Refer to page 115 for general precautions on operand data areas.

Flags

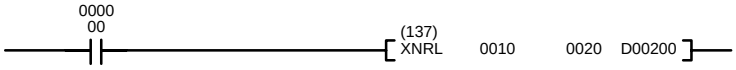
ER (A50003): Content of ∗DM word is not BCD when set for BCD.

EQ (A50006): The result is 0.

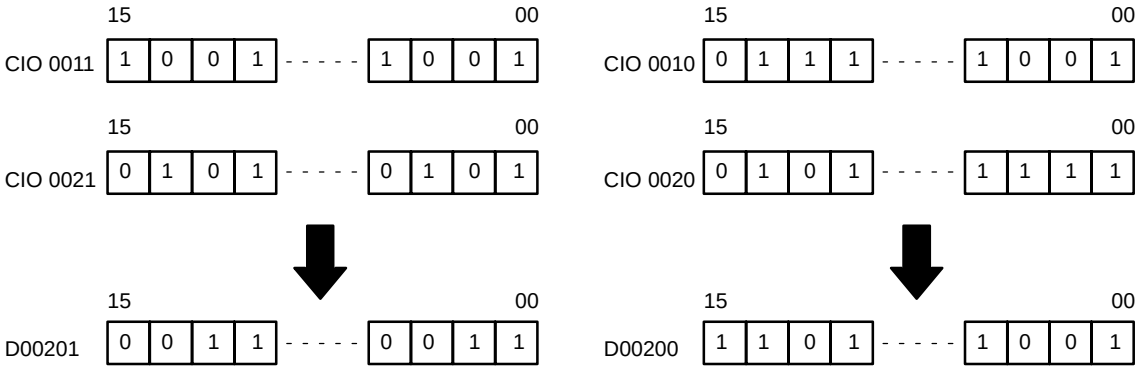
N (A50008): Shows the status of bit 15 of R+1 after execution.

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.

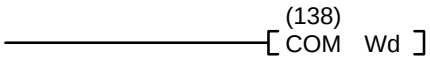


Address	Instruction	Operands
00000	LD	000000
00001	XNRL(137)	
		0010
		0020
		D0020



5-25-9 COMPLEMENT: COM(138)

Ladder Symbol



Operand Data Area

Wd: Word CIO, G, A, DM, DR, IR

Variations

↑ 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 115 for general precautions on operand data areas.

Flags

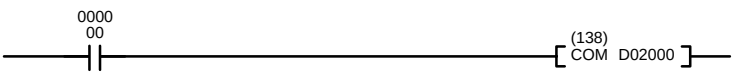
ER (A50003): Content of ∗DM word is not BCD when set for BCD.

EQ (A50006): The result is 0.

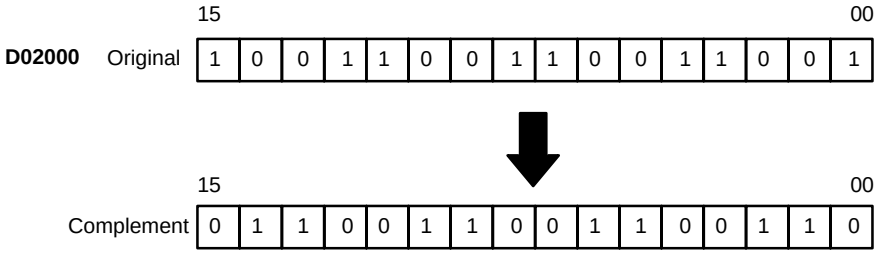
N (A50008): Shows the status of bit 15 of R after execution.

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)

Ladder Symbol	Operand Data Area	
(139) COML Wd	**Wd: Word**	CIO, G, A, DM

Variations

↑ COML(139)

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 115 for general precautions on operand data areas.

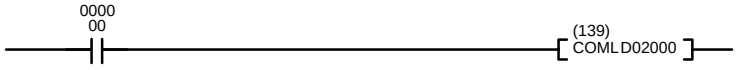
Flags

ER (A50003): Content of ∗DM word is not BCD when set for BCD.

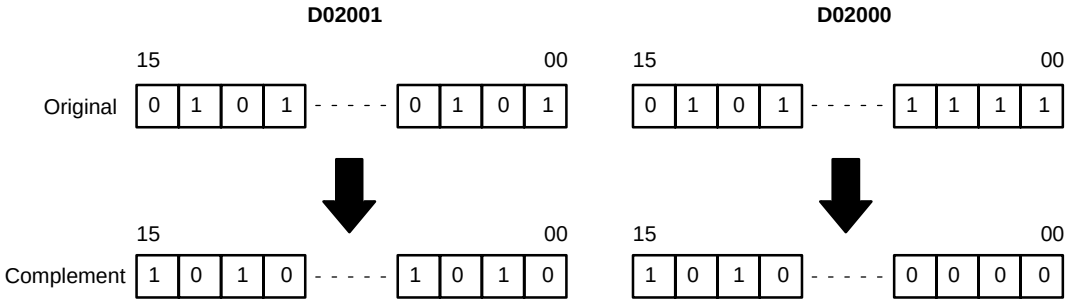
EQ (A50006): The result is 0.

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

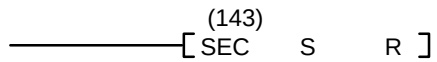


5-26 Time Instructions

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)

Ladder Symbol



Variations

↑ SEC(143)

Operand Data Areas

S: 1st source word CIO, G, A, T, C, #, DM

R: 1st result word CIO, G, A, DM

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 115 for general precautions on operand data areas.

Flags

ER (A50003): S or S+1 are not BCD.

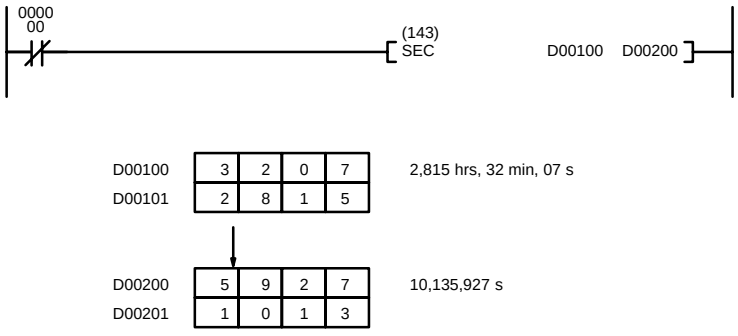
Number of seconds or minutes exceeds 59.

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)

Ladder Symbol

(144)
HMS S R

Variations

↑ HMS(144)

Operand Data Areas

S: 1st source word CIO, G, A, T, C, #, DM

R: 1st result word CIO, G, A, DM

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 115 for general precautions on operand data areas.

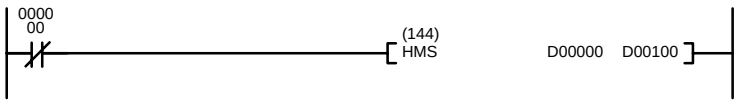
Flags

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.

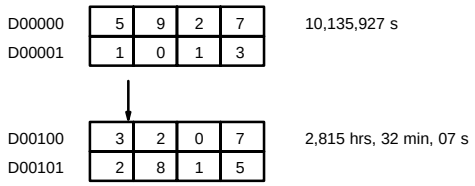
EQ (A50006): ON when the result is 0.

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

(145)
CADD C T R

Variations

↑ CADD(145)

Operand Data Areas

C: 1st calendar word CIO, G, A, T, C, DM

T: 1st time word CIO, G, A, T, C, #, DM

R: 1st result word CIO, G, A, DM

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 115 for general precautions on operand data areas.

Flags

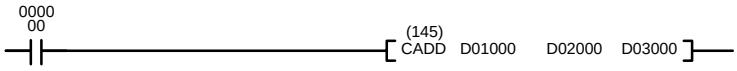
ER (A50003): Time or calendar data is not in the correct format (including impossible dates such as Feb. 30).

Content of ∗DM word is not BCD when set for BCD.

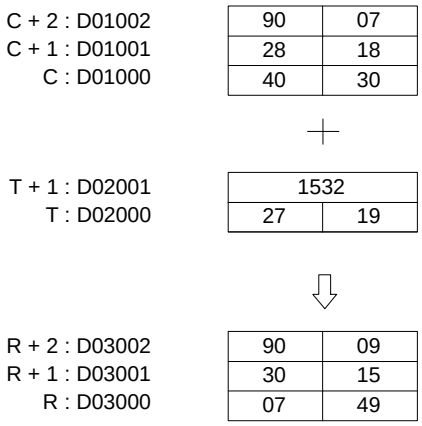
EQ (A50006): ON when the content of R, R+1, and R+2 is 0 after execution.

Example

When CIO 000000 is ON in the following example, the time data in D02000 and D02001 is added to the calender data in D01000 through D01002 and output as calender data to D03000 through D03002.

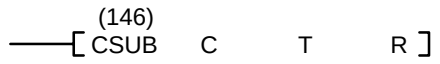


Address	Instruction	Operands
00000	LD	000000
00001	CADD(145)	
		D01000
		D02000
		D03000



5-26-4 CALENDAR SUBTRACT: CSUB(146)

Ladder Symbol



Variations

↑ CSUB(146)

Operand Data Areas

C: 1st calendar word	CIO, G, A, T, C, DM
T: 1st time word	CIO, G, A, T, C, #, DM
R: 1st result word	CIO, G, A, DM

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 115 for general precautions on operand data areas.

Flags

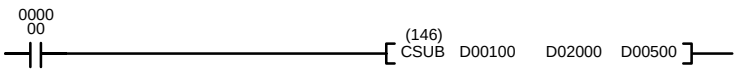
ER (A50003): Time or calendar data is not in the correct format (including impossible dates such as Feb. 30).

Content of ∗DM word is not BCD when set for BCD.

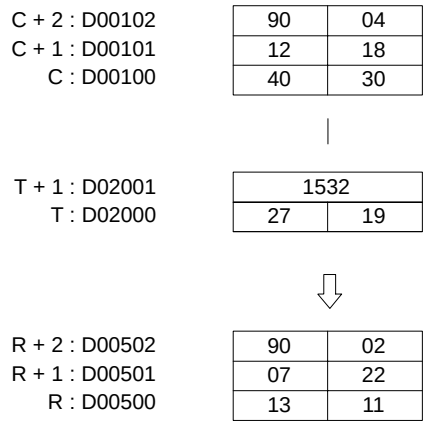
EQ (A50006): ON when the content of R, R+1, and R+2 is 0 after execution.

Example

When CIO 000000 is ON in the following example, the time data in D02000 and D02001 is subtracted from the calender data in D01000 through D01002 and output as calender data to D00500 through D00502.

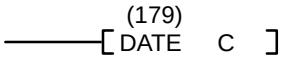


Address	Instruction	Operands
00000	LD	000000
00001	CSUB(146)	
		D00100
		D00200
		D00500



5-26-5 CLOCK COMPENSATION: DATE(179) (CVM1 V2)

Ladder Symbol



Operand Data Areas

C: Control word	CIO, G, A, T, C, DM

Variations

↑DATE(179)

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).

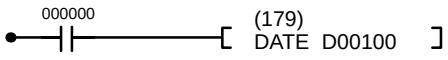
Note Refer to page 115 for general precautions on operand data areas.

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

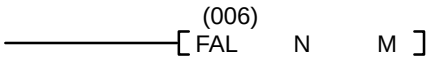
5-27 Special Instructions

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)

Ladder Symbol



Variations

↑ FAL(006)

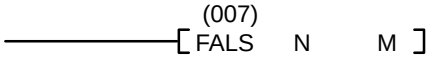
Operand Data Areas

N: FAL number 000 to 511

M: 1st message word CIO, G, A, #, DM

SEVERE FAILURE ALARM: FALS(007)

Ladder Symbol



Operand Data Areas

N: FAL number 001 to 511

M: 1st message word CIO, G, A, #, DM

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 8 Error Processing* 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 *5-36-3 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 115 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.

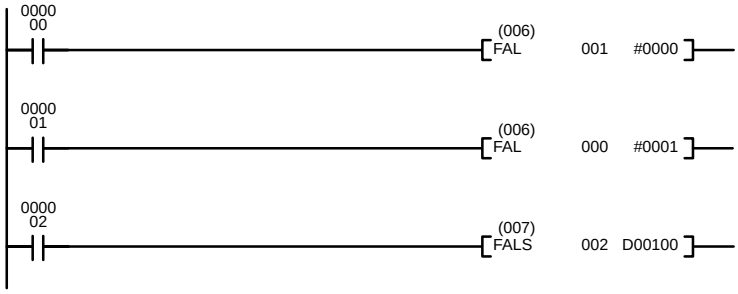
A43001 to A46115:
These flags are turned ON to indicate the FAL numbers for which FAL(006) or FALS(007) have been executed.

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).

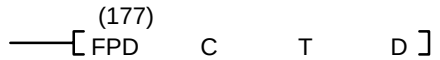
When CIO 000002 turns ON, a fatal error is generated as FAL 002, the ERROR indicator on the CPU lights, all outputs are turned OFF, and C102 is output to A400. Program execution will stop. If a Programming Device is connected and online, the message stored in D00100 through D00107 will also be displayed 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) (CVM1 V2)

Ladder Symbol



Operand Data Areas

C: Control data	#
T: Monitoring time	CIO, G, A, #, DM
D: First register word	CIO, G, A, DM

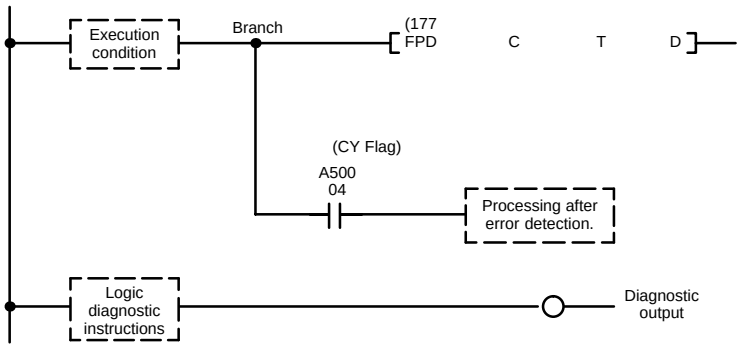
Description

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 fist 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 used 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.



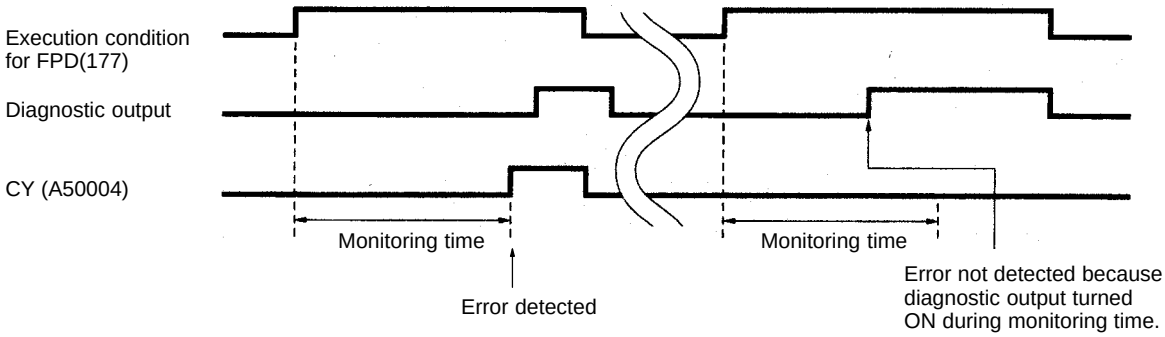
Time Monitoring

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 Peripheral Device along with the bit address mentioned in step 2.

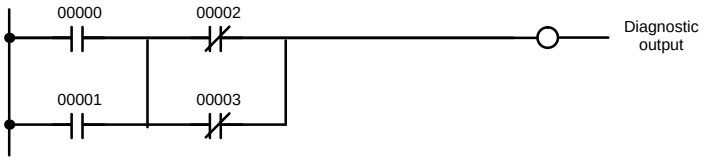
The following chart shows the timing that occurs when the execution condition for FPC(177) turns ON.



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 indirectly 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.

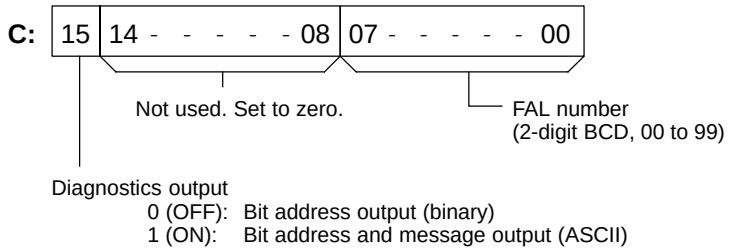
When IR 00000 to IR 00003 are ON in the following example, the normally closed condition IR 00002 would be found as the cause of the diagnostic output not turning ON.



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.

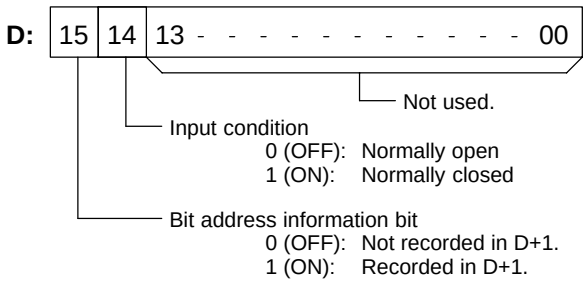


Diagnostics Output

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 70 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 Peripheral 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 Peripheral Device, such as a Programming Console.
2. Use the Peripheral 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.

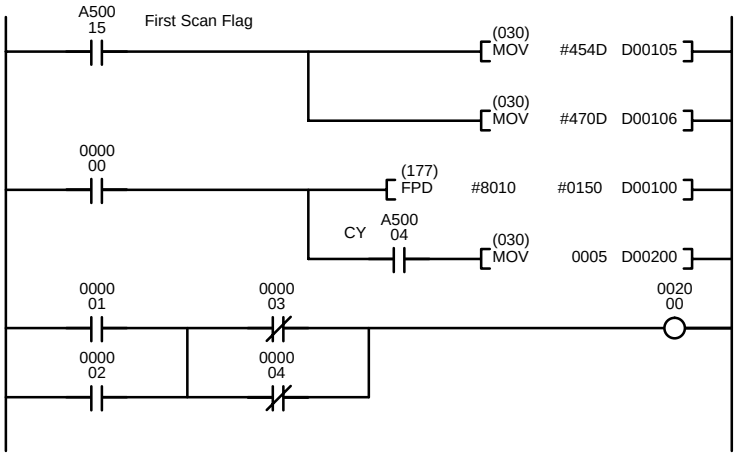
Example

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 Peripheral 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 Peripheral 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.

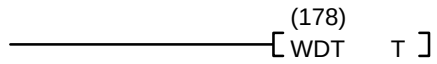
Content of ∗DM word is not BCD when set for BCD.

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) (CVM1 V2)

Ladder Symbol



Operand Data Area

T: Timer value # (0000 to 3999)

Variations

↑ WDT(178)

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.

Time extension = 10 ms x 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.

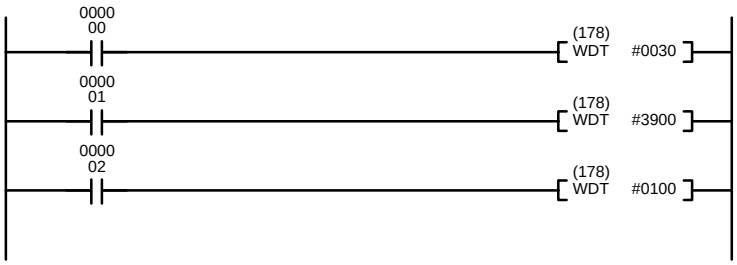
Example

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)

Ladder Symbol

(184)
IORF St E

Variations

↑ IORF(184)

Operand Data Areas

St: Starting word CIO

E: Ending word CIO

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.

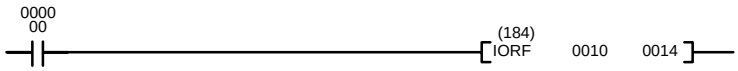
Note Refer to page 115 for general precautions on operand data areas.

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)

Ladder Symbol

(189)
IODP C S

Variations

↑ IODP(189)

Operand Data Areas

C: Control word CIO, G, A, T, C, #, DM, DR, IR

S: Source word(s) CIO, G, A, T, C, DM

Description

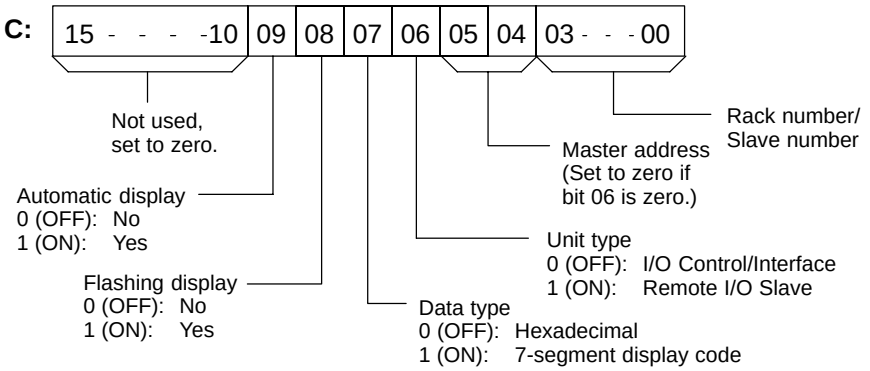
When the execution condition is OFF, IODP(189) is not executed. When the execution condition is ON, IODP(189) outputs four characters to the 7-segment display on a SYSMAC BUS/2 Remote I/O Slave Unit, I/O Control Unit, or I/O Interface Unit. The four characters can be in 7-segment display code in source 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).

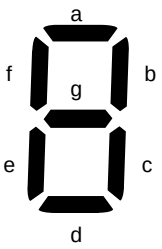
If bit 09 of C is set for automatic display (ON), the characters will be displayed regardless of the Unit's display mode, but if bit 09 of C is OFF, the characters will be displayed only when the Unit is set to display mode 3.



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 115 for general precautions on operand data areas.

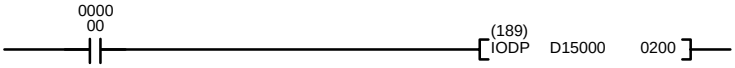
Flags

ER (A50003): Content of ∗DM word is not BCD when set for BCD.
Control word data is incorrect.

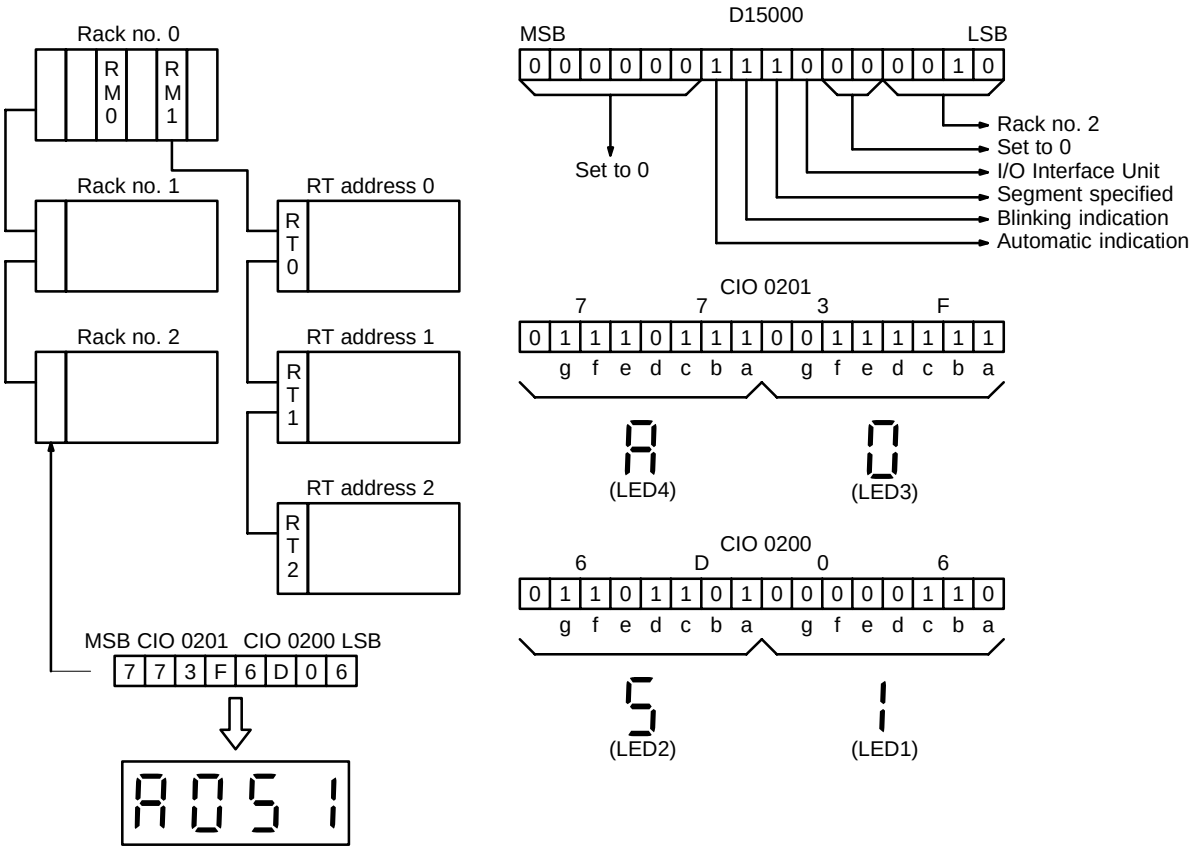
Example

The following example show how to produce the display "*R05 1*" 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.

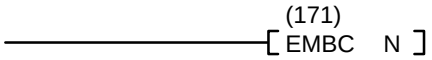


Address	Instruction	Operands
00000	LD	000000
00001	IODP(189)	
		D15000
		0200



5-27-6 SELECT EM BANK: EMBC(171)

Ladder Symbol



Operand Data Area

N: EM bank number	CIO, G, A, #, DM, DR, IR

Variations

↑ EMBC(171)

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.

When power returns to the CPU after an interruption, the current EM bank number will revert to the bank number recorded in A511 before the power was interrupted, even if EMBC(171) is used in a power OFF interruption program to change the current bank number regardless of the IOM Hold settings.

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 115 for general precautions on operand data areas.

Flags

ER (A50003): N is not between 0000 and 0007.

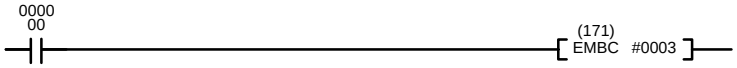
The EM Unit does not have the EM bank indicated by N, or the indicated bank is being used as file memory.

The CPU does not support the EM Area.

Content of ∗DM word is not BCD when set for BCD.

Example

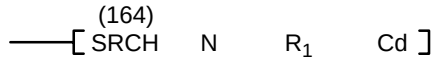
When CIO 000000 is ON in the following example, the current EM bank is changed to bank 3. The contents of A511 would change to “8003” to indicate that bank 3 is the current bank.



Address	Instruction	Operands
00000	LD	000000
00001	EMBC(171)	
		#0003

5-27-7 DATA SEARCH: SRCH(164)

Ladder Symbol



Variations

↑ SRCH(164)

Operand Data Areas

N: Number of words	CIO, G, A, T, C, #, DM, DR, IR
R_1: 1st word in range	CIO, G, A, T, C, DM
Cd: Comparison data	CIO, G, A, T, C, #, DM, DR, IR

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_1 to R_1+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_1 and R_1+N–1 must be in the same data area.

Note Refer to page 115 for general precautions on operand data areas.

Flags

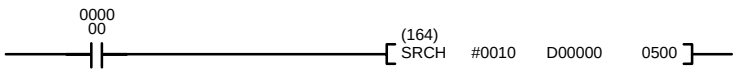
ER (A50003): N is 0000 or is not BCD.

Content of ∗DM word is not BCD when set for BCD.

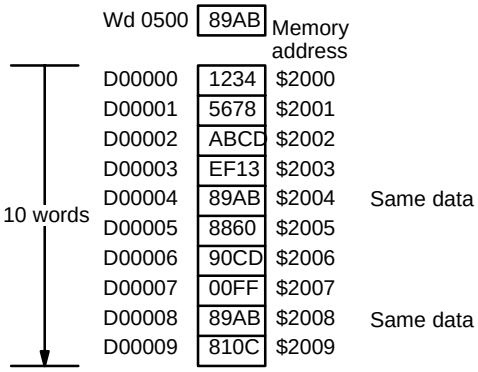
EQ (A50006): An address within the range contains the comparison data.

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

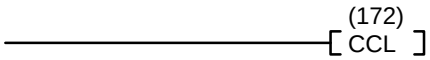


5-28 Flag/Register Instructions

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)

Ladder Symbol



Variations

↑ 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

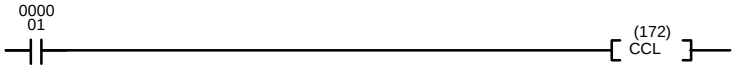
CCL(172) should not be executed unless CCS(173) has been executed earlier.

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)

Ladder Symbol

(173)
CCS

Variations

↑ CCS(173)

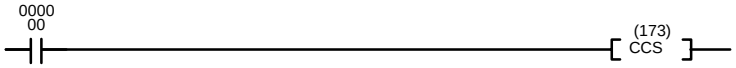
Description When the execution condition is OFF, CCS(173) is not executed. When the execution condition is ON, CCS(173) records the current status of the Arithmetic Flags in the CPU for later retrieval by the CCL(172) instruction.

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)

Ladder Symbol

(175)
REGL S

Operand Data Area

S: 1st source word CIO, G, A, DM

Variations

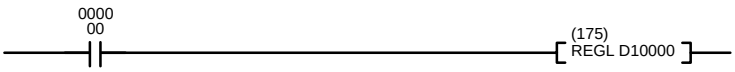
↑ REGL(175)

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.

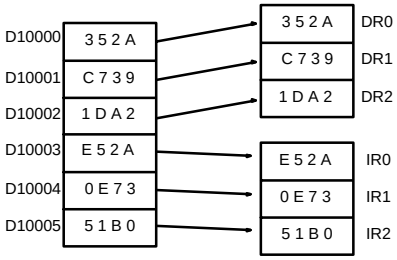
Precautions Refer to page 115 for general precautions on operand data areas.

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)

Ladder Symbol

(176)
REGS D

Operand Data Area

D: 1st destination word CIO, G, A, DM

Variations

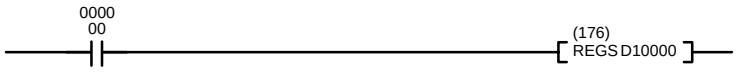
↑ REGS(176)

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.

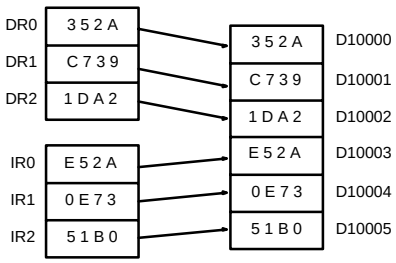
Precautions Refer to page 115 for general precautions on operand data areas.

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.

The steps described here are not related to SFC programming.

STEP DEFINE: STEP(008)

Ladder Symbol	Operand Data Area	
(008) STEP B	**B: Bit**	CIO, G, A
(008) STEP		

STEP START: SNXT(009)

Ladder Symbol	Operand Data Area	
(009) SNXT B	**B: Bit**	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.

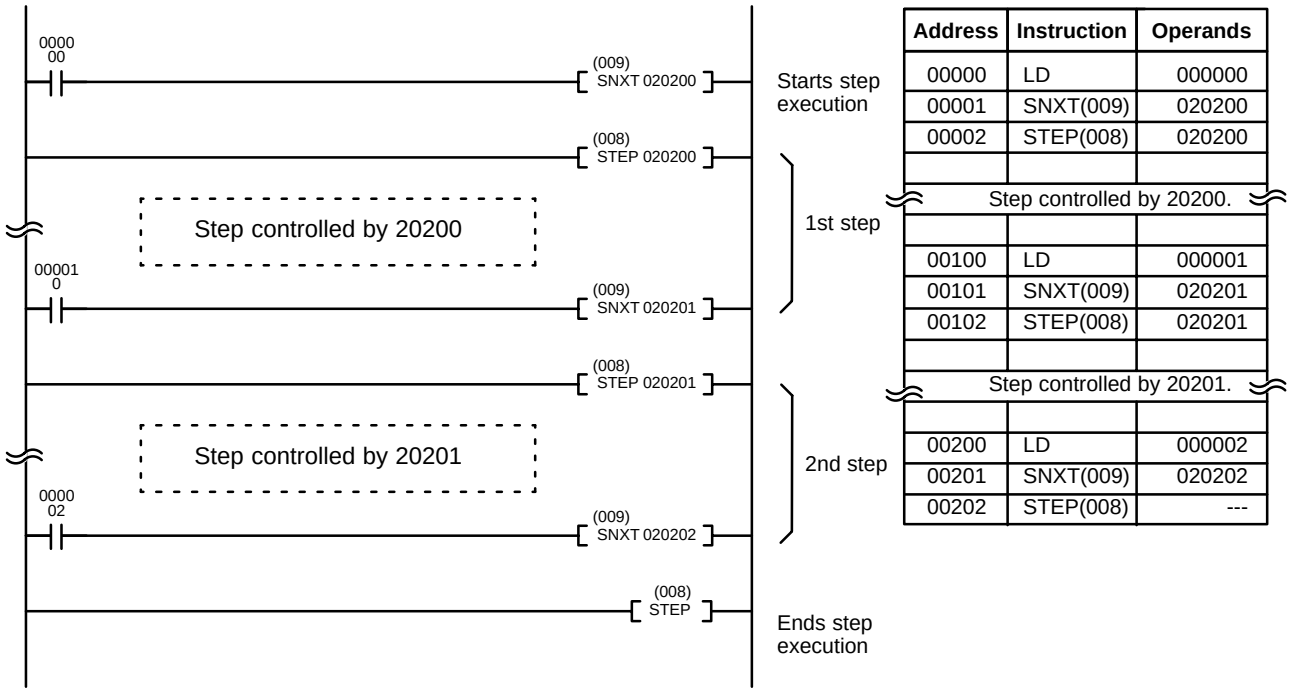
Once SNXT(009) is used to start step execution, step execution will continue until STEP(008) is executed without a control bit. STEP(008) executed without a control bit is used to stop step execution and return to normal execution. STEP(008) without a control bit must be preceded by SNXT(009) with a 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.



Address	Instruction	Operands
00000	LD	000000
00001	SNXT(009)	020200
00002	STEP(008)	020200
Step controlled by 20200.		
00100	LD	000001
00101	SNXT(009)	020201
00102	STEP(008)	020201
Step controlled by 20201.		
00200	LD	000002
00201	SNXT(009)	020202
00202	STEP(008)	---

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, interrupt programs, 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.

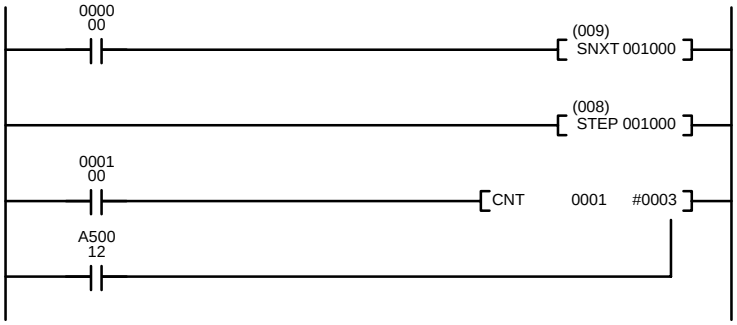
Control bit status will be lost during any power interruption if it is not in the Holding Area. If it is necessary to maintain status to resume execution at the same step, Holding Area bits must be used.

Only one section of step programming can be started during a cycle. Be sure that two steps aren't started during a cycle, particularly when using steps with SFC.

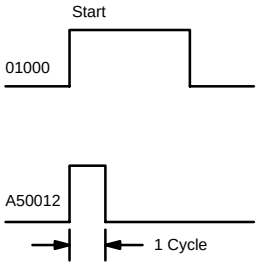
Note Refer to page 115 for general precautions on operand data areas.

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

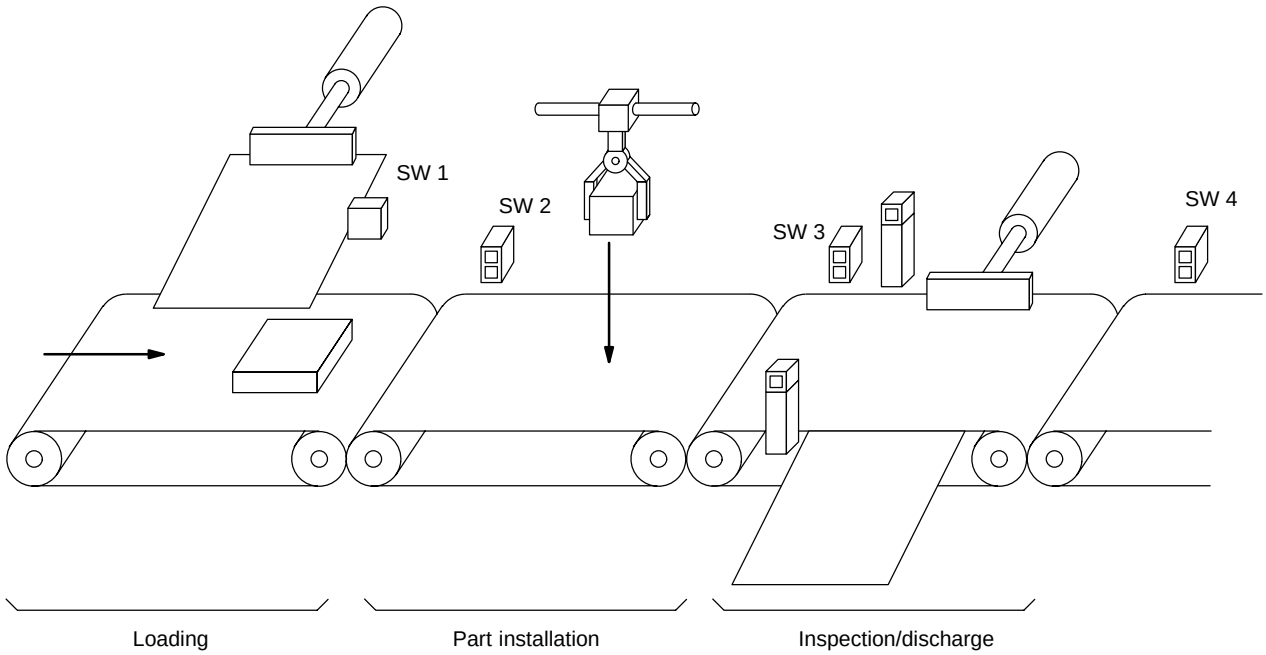


Examples

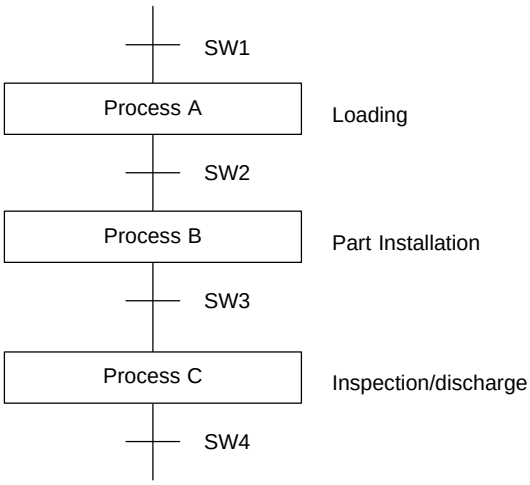
The following three examples demonstrate the three types of execution control possible with step programming. *Example 1* demonstrates sequential execution; *Example 2*, branching execution; and *Example 3*, parallel execution.

Example 1: Sequential Execution

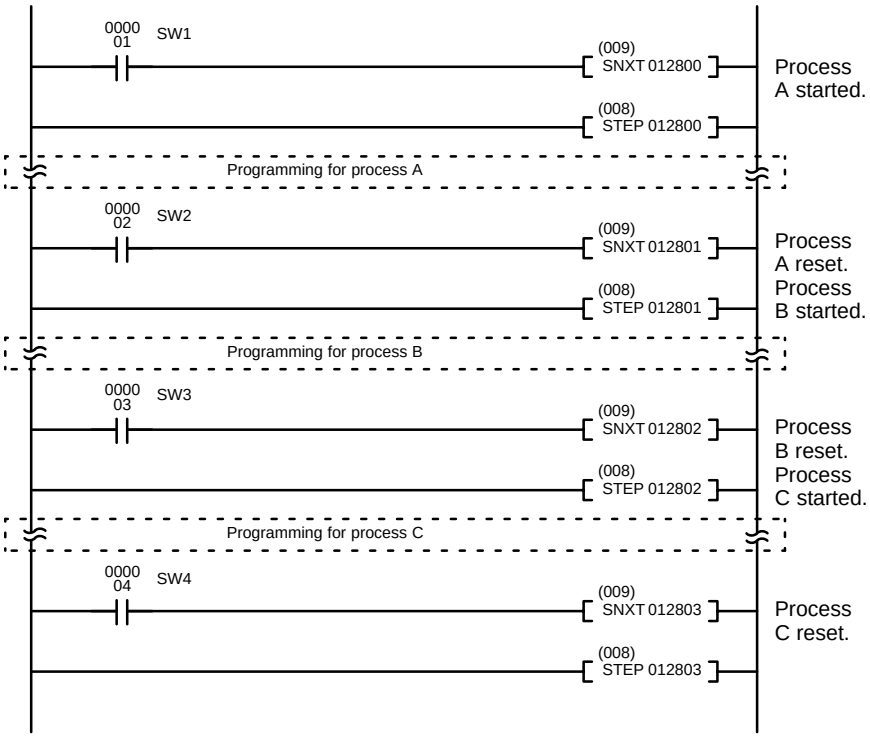
The following process requires that three processes, loading, part installation, and inspection/discharge, be executed in sequence with each process being reset before continuing on the the next process. Various sensors (SW1, SW2, SW3, and SW4) are positioned to signal when processes are to start and end.



The following diagram demonstrates the flow of processing and the switches that are used for execution control.



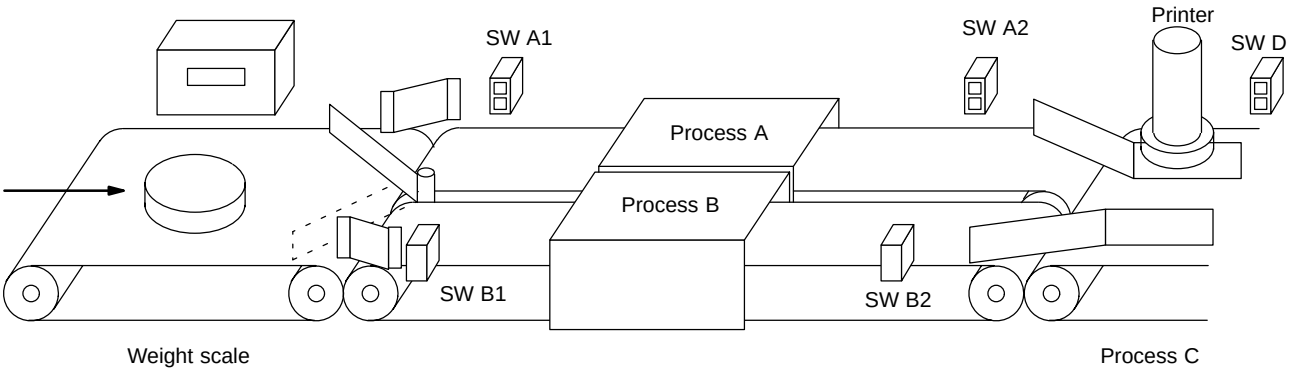
The program for this process, shown below, utilizes the most basic type of step programming: each step is completed by a unique SNXT(009) that starts the next step. Each step starts when the switch that indicates the previous step has been completed turns ON.



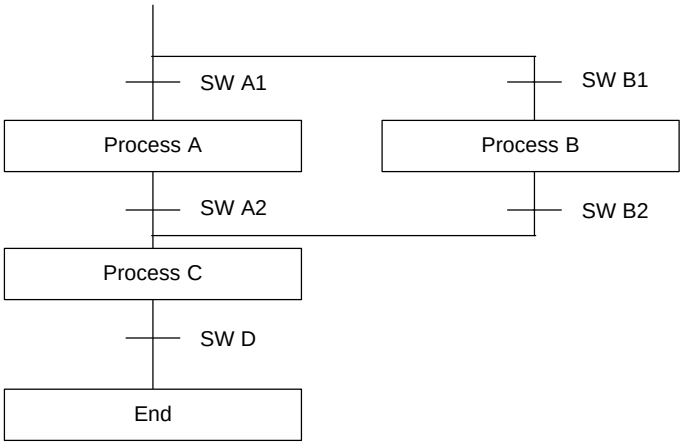
Address	Instruction	Operands
00000	LD	000001
00001	SNXT(009)	012800
00002	STEP(008)	012800
Process A		
00100	LD	000002
00101	SNXT(009)	012801
00102	STEP(008)	012801
Process B		
00100	LD	000003
00101	SNXT(009)	012802
00102	STEP(008)	012802
Process C		
00200	LD	000004
00201	SNXT(009)	012803
00202	STEP(008)	012803

Example 2: Branching Execution

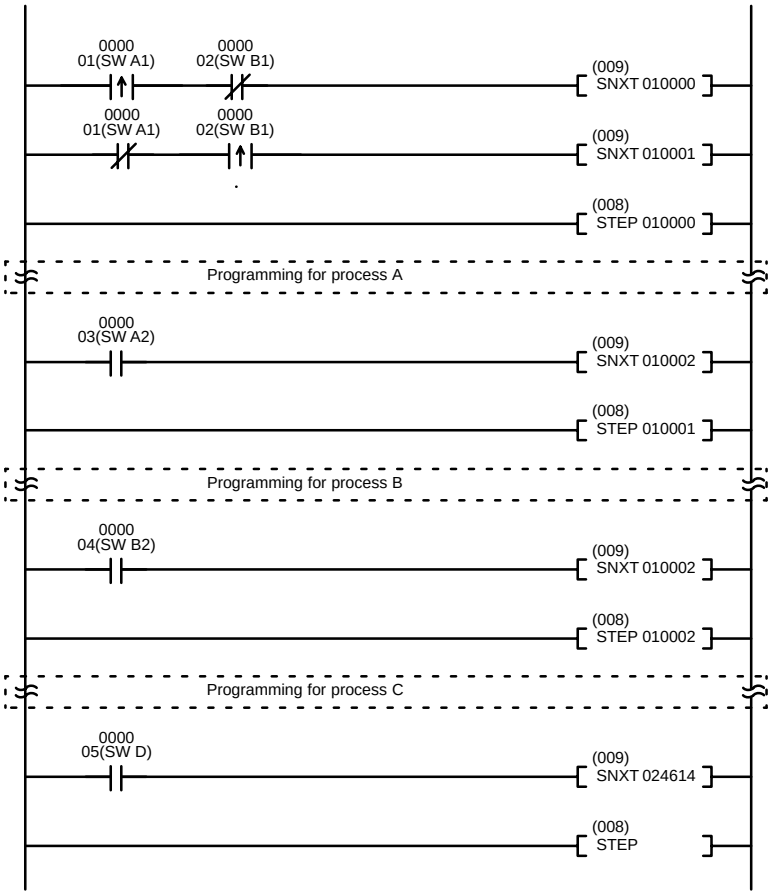
The following process requires that a product is processed in one of two ways, depending on its weight, before it is printed. The printing process is the same regardless of which of the first processes is used. Various sensors are positioned to signal when processes are to start and end.



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).



Process A started.

Process A reset. Process B started.

Process B reset. Process C started.

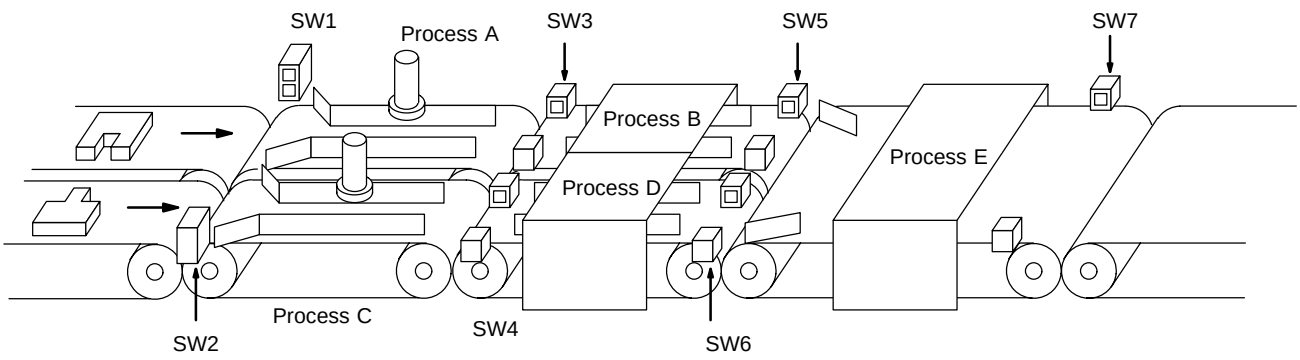
Process C reset.

Address	Instruction	Operands
00000	↑LD	000001
00001	AND NOT	000002
00002	SNXT(009)	010000
00003	LD NOT	000001
00004	↑AND	000002
00005	SNXT(009)	010001
00006	STEP(008)	010000
Process A		
00100	LD	000003
00101	SNXT(009)	010002
00102	STEP(008)	010001
Process B		
00100	LD	000004
00101	SNXT(009)	010002
00102	STEP(008)	010002
Process C		
00200	LD	000005
00201	SNXT(009)	024614
00202	STEP(008)	---

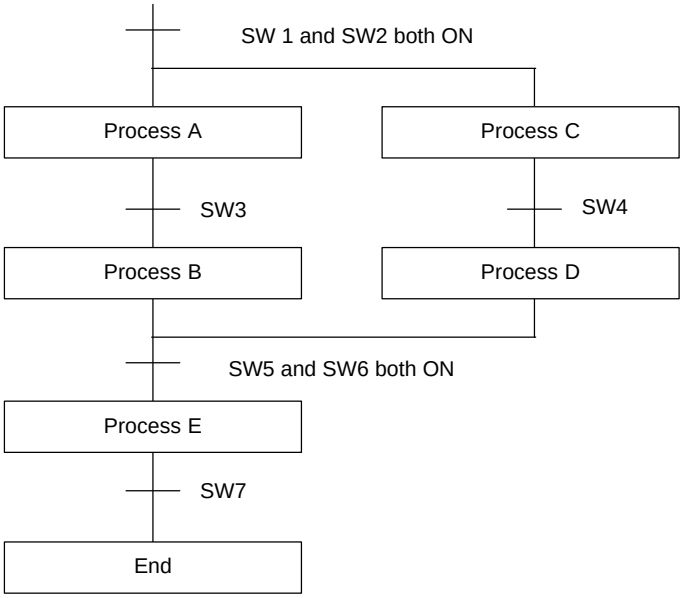
Note In the above programming, CIO 010002 is used in two SNXT(0009) instructions. This will not produce a duplication error during the program check.

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.

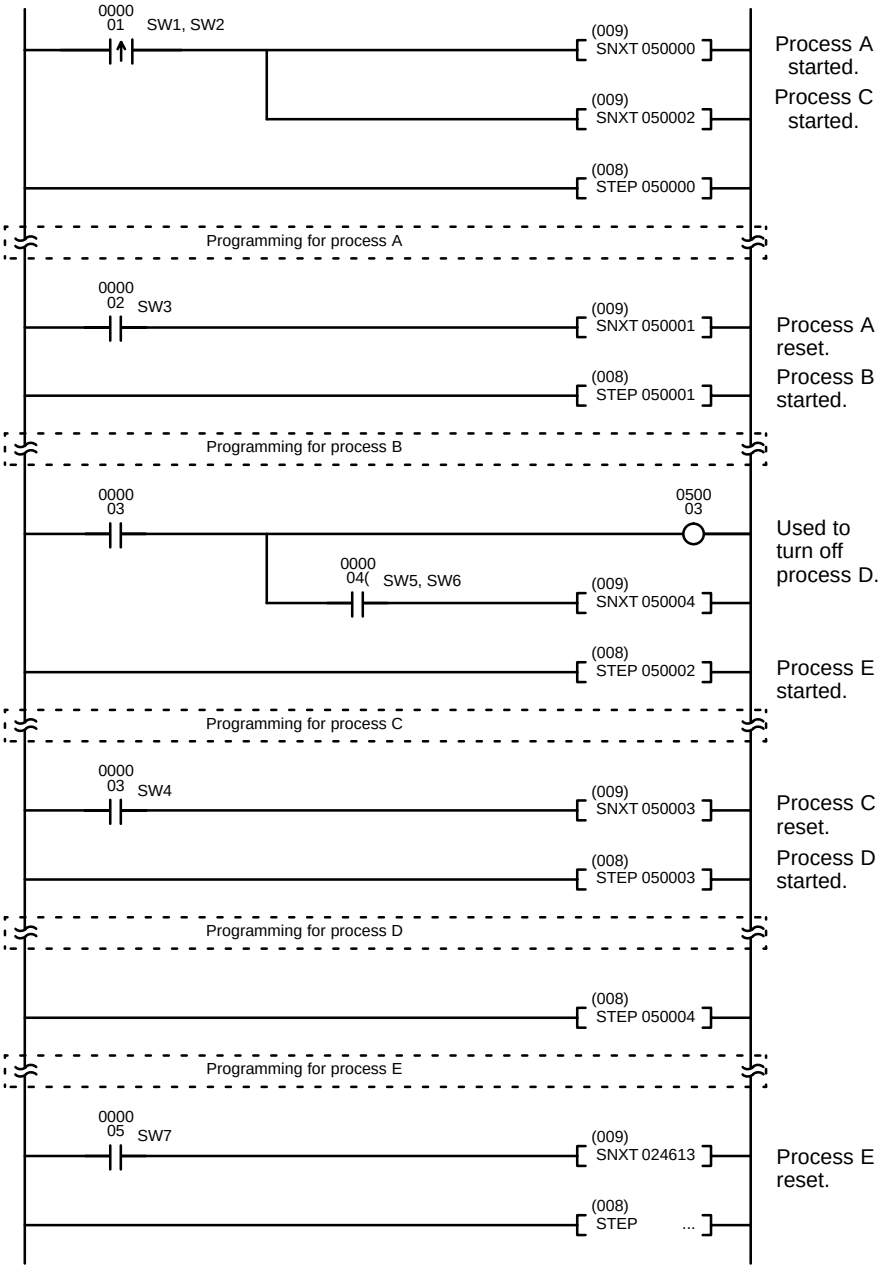


The following diagram demonstrates the flow of processing and the switches that are used for execution control. Here, process A and process C are started together. When process A finishes, process B starts; when process C finishes, process D starts. When both processes B and D have finished, process E starts.



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.



Address	Instruction	Operands
00000	LD	000001
00001	SNXT(009)	050000
00002	SNXT(009)	050002
00003	STEP(008)	050000
	Process A	
00100	LD	000002
00101	SNXT(009)	050001
00102	STEP(008)	050001
	Process B	
00100	LD	000003
00101	OUT	050003
00101	AND	000004
00101	SNXT(009)	050004
00102	STEP(008)	050002
	Process C	
00200	LD	000003
00201	SNXT(009)	050003
00202	STEP(008)	050003
	Process D	
00300	STEP(008)	050004
	Process E	
00400	LD	000005
00401	SNXT(009)	024613
00402	STEP(008)	---

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
(150) SBN N	**N: Subroutine number** #

SUBROUTINE RETURN: RET(152)

Ladder Symbol
(152) RET

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) or activated by an interrupt.

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 for the CV1000, CV2000, CVM1-CPU11-EV2, or CVM1-CPU21-EV2 or between 000 and 099 for the CV500 or CVM1-CPU01-EV2. 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.

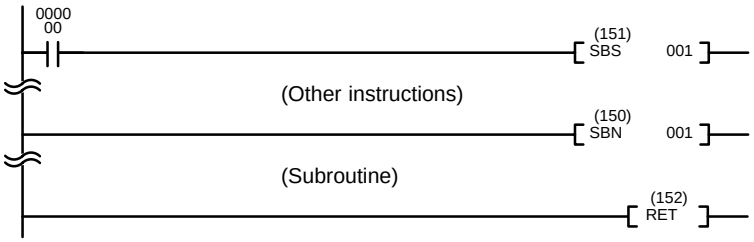
The step instructions, STEP(008) and SNXT(009), cannot be used within a subroutine.

Flags

There are no flags directly affected by these instructions.

Example

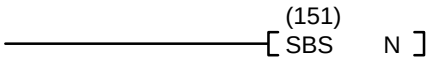
When CIO 000000 is ON in the following example, the instructions between SBN(150) 001 and RET(152) are executed once before returning to execute the next instruction line after SBS(151).



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)

Ladder Symbol



Operand Data Area

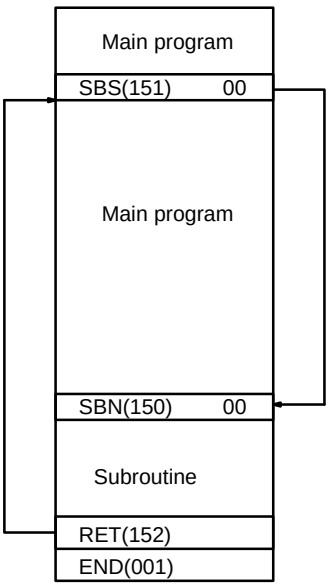
N: Subroutine number #

Variations

↑ SBS(151)

Description

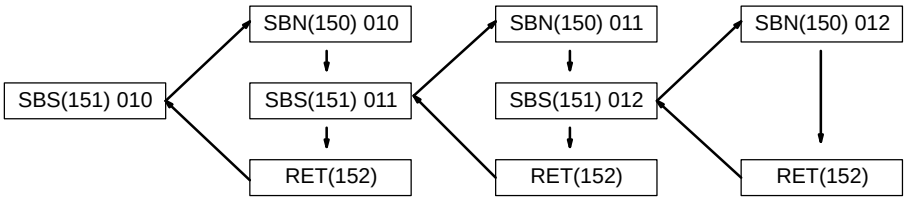
A subroutine can be executed by placing SBS(151) in the main program at the point where the subroutine is desired. The subroutine number used in SBS(151) indicates the desired subroutine. When SBS(151) is executed with an ON execution, the instructions between the SBN(150) with the same subroutine number 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.

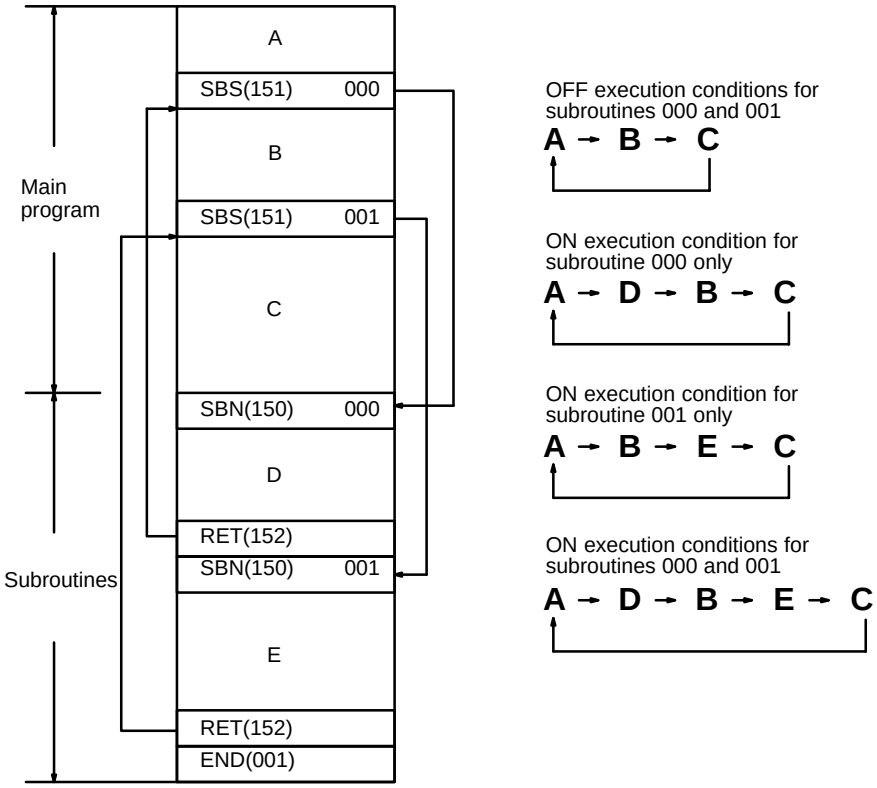
SBS(151) may also be placed into a subroutine to shift program execution from one subroutine to another, i.e., subroutines may be nested. When the second subroutine has been completed (i.e., RET(152) has been reached), program execution returns to the original subroutine which is then completed before 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.



Although subroutines 00 through 31 can be called by using SBS(151), they are also activated by interrupt signals from Interrupt Input Units. Subroutine 99, which can also be called using SBS(151), is used for the scheduled interrupt. (Refer to the next subsection for details.)

The following diagram illustrates program execution flow for various execution conditions for two SBS(151).



Precautions

N must be between 000 and 999 for the CV1000, CV2000, CVM1-CPU11-EV2, or CVM1-CPU21-EV2, or between 000 and 099 for the CV500 or CVM1-CPU01-EV2.

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) (CVM1 V2)

Ladder Symbol

(156)
MCRO N S D

Variations

↑MCRO(156)

Operand Data Areas

N: Subroutine number	000 to 999 or 000 to 099
S: First input parameter word	CIO, G, A, T, C, DM
D: First output parameter word	CIO, G, A, T, C, DM

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 for the CV1000, CV2000, CVM1-CPU11-EV2, or CVM1-CPU21-EV2 or between 000 and 099 for the CV500 or CVM1-CPU01-EV2.

Note Refer to page 115 for general precautions on operand data areas.

Flags

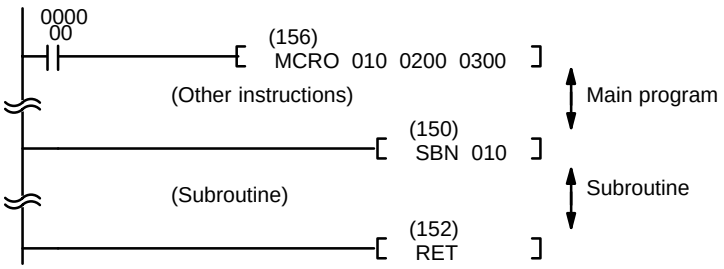
ER (A50003): Subroutine does not exist for the specified subroutine number.

A subroutine has called itself.

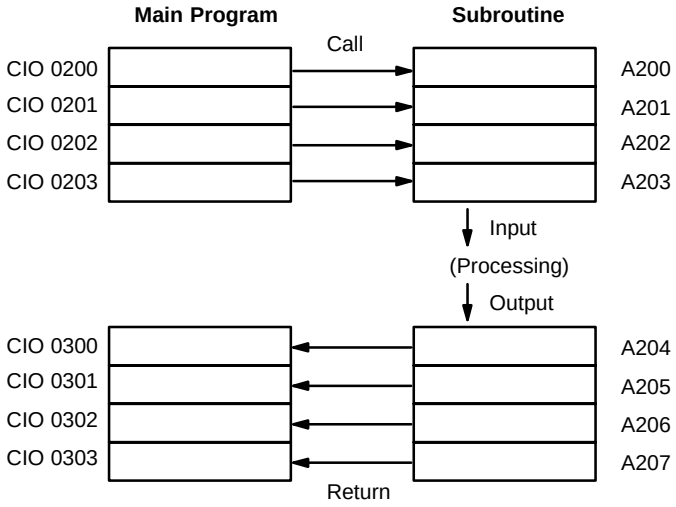
An active subroutine has been called.

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.

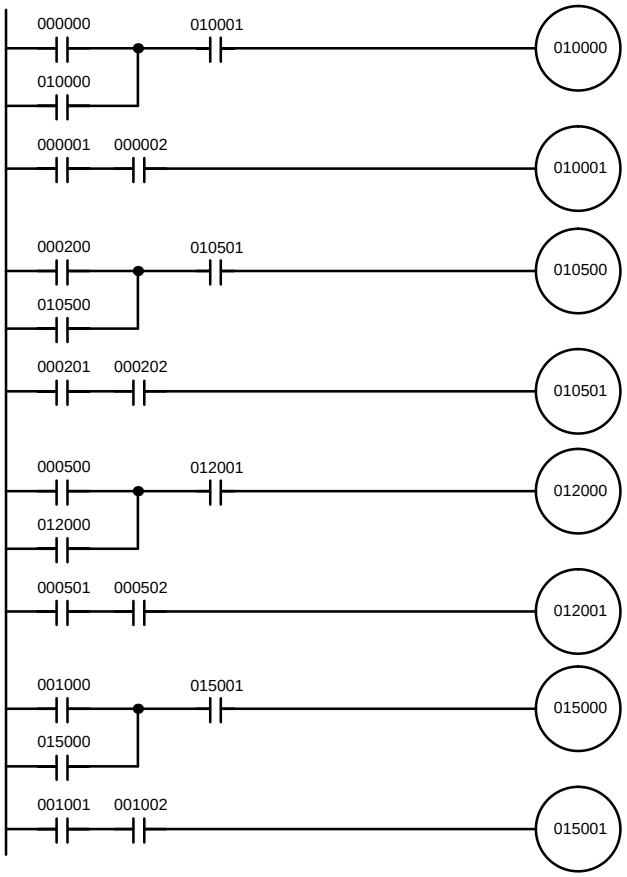


Address	Instruction	Operands
00000	LD	000000
00001	MCRO(156)	010
		0200
		0300
(Other instructions)		
00023	SBN(150)	010
(Subroutine)		
00035	RET(152)	

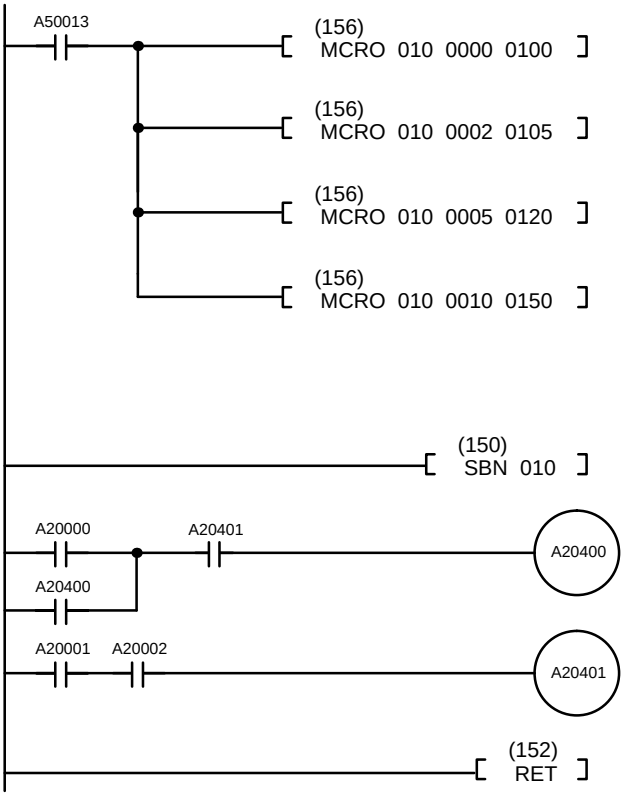


Program Examples

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 Interrupt Control

Interrupts cause a break in the flow of the main program execution such that the flow can be resumed from that point after completion of the interrupt program. Interrupt programs should be entered in SFC if SFC programming is being used. Interrupt programs are written as a ladder program only if the program type is ladder.

Note Be sure to include an END(01) instruction at the end of interrupt programs if the program type is ladder.

There are three instructions that control and monitor I/O interrupts and scheduled interrupts, MSKS(153), CLI(154), and MSKR(155). MSKS(153) masks interrupts from Interrupt Input Units (so that they are recorded but ignored) and sets the time interval for scheduled interrupts, CLI(154) clears interrupts, and MSKR(155) writes either the current mask status of a designated Interrupt Input Unit or the current scheduled interrupt interval into a designated word.

The four types of interrupts, I/O, scheduled, power OFF, and power ON are described briefly below.

An **I/O interrupt** is caused by an input signal from an Interrupt Input Unit.

Up to four Interrupt Input Units (0 to 3) can be mounted on the CPU Rack and Expansion CPU Racks. Each Unit is allocated one I/O word.

For each Interrupt Input Unit, bits 00 through 07 may be used for interrupt signals. Bits 08 through 15 are not used. When one of the bits assigned to an Interrupt Input Unit turns ON, the interrupt program associated with it is called and executed. A unique interrupt program number is associated with each bit according to the following table.

Interrupt Input Unit		Program	Interrupt Input Unit		Program
Unit no.	Bit no.		Unit no.	Bit no.	
0	0	00	2	0	16
	1	01		1	17
	2	02		2	18
	3	03		3	19
	4	04		4	20
	5	05		5	21
	6	06		6	22
	7	07		7	23
1	0	08	3	0	24
	1	09		1	25
	2	10		2	26
	3	11		3	27
	4	12		4	28
	5	13		5	29
	6	14		6	30
	7	15		7	31

A **scheduled interrupt** is repeated at regular intervals.

The time interval between interrupts is set by the user and is unrelated to the cycle timing of the PC. The time interval can be set between 10 and 99,990 ms. This capability is useful for periodic supervisory or executive program execution.

A **power OFF interrupt** is generated by an interruption of power to the CPU longer than the momentary power interruption time set in the PC Setup (0 to 9 ms).

If the PC Setup (Execution controls 2) has been preset to enable power OFF interrupts, the interrupt program is activated to manage the power outage.

The length of the power OFF interrupt program is limited. Refer to *6-1-6 Power OFF Interruption and Restart Continuation* for details.

A **power ON interrupt** is activated when power returns to the CPU after an interruption. The power ON interrupt is effective only if there is a power ON interrupt program. The power ON interrupt program is executed after initialization.

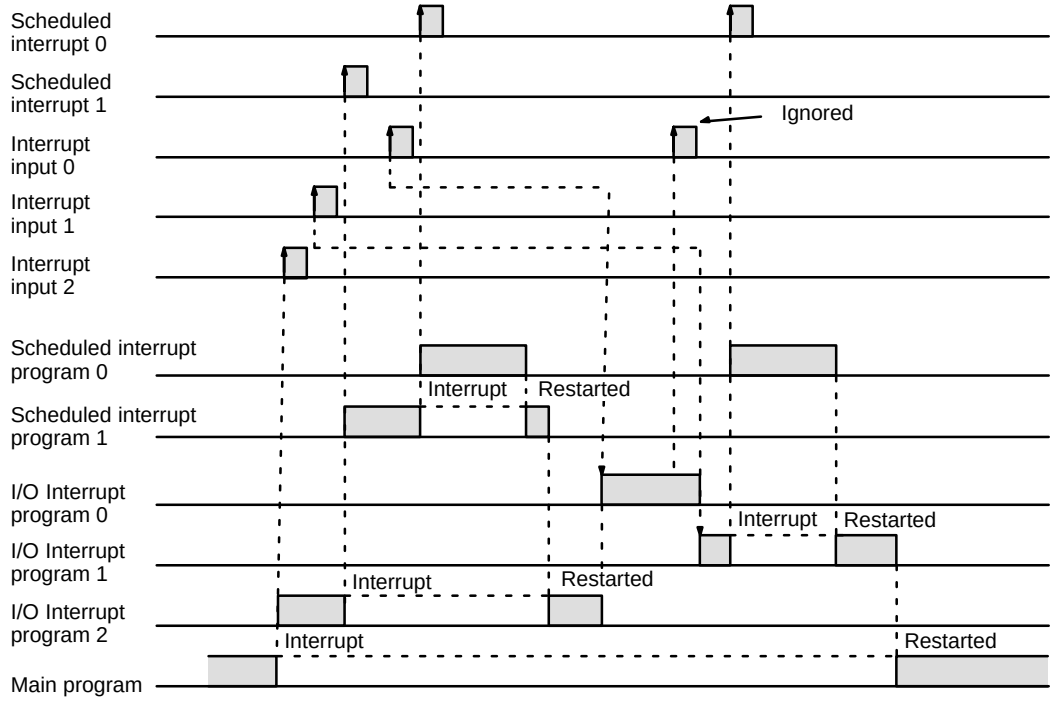
Interrupt Priority Levels

The PC employs a priority system for handling interrupts. A power OFF interrupt is given the highest priority, followed by power ON, scheduled, and I/O interrupts, in that order. Lower numbered I/O interrupts are given priority over a higher numbered I/O interrupts.

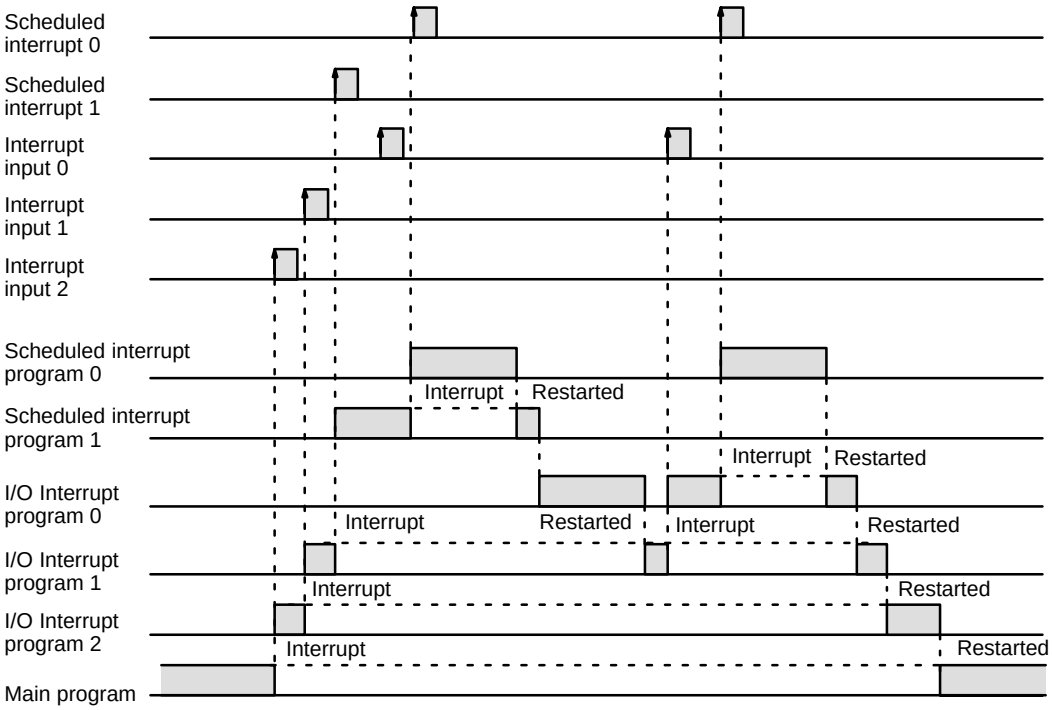
In the CV1000, CV2000, CVM1-CPU11-EV2, or CVM1-CPU21-EV2 a level 0 scheduled interrupt (defined by setting N=4 in MSKS(153)) takes priority over a level 1 scheduled interrupt (defined by setting N=5 when MSKS(153)). If a level 0 scheduled interrupt occurs while a level 1 scheduled interrupt is being executed, the level 1 scheduled interrupt program is halted until the level 0 scheduled interrupt is completed.

The I/O interrupt setting in the PC Setup determines whether I/O interrupts with higher priority will be serviced immediately (interrupting the I/O interrupt program currently being executed) or wait until the current I/O interrupt is completed.

When the I/O interrupt setting in the PC Setup (execution controls 2) is set to hold other I/O interrupts, an incoming I/O interrupt must wait until the first I/O interrupt is finished, regardless of the priority ranking of the two I/O interrupts. The following example shows program execution with this I/O interrupt setting.

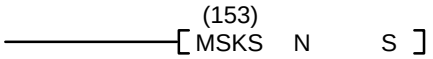


When the I/O interrupt setting in the PC Setup is set to execute higher priority interrupts, an incoming I/O interrupt will be serviced immediately if its interrupt program number is higher than that of the I/O interrupt currently being serviced. The following example shows program execution with this I/O interrupt setting.



5-31-1 INTERRUPT MASK: MSKS(153)

Ladder Symbol



Variations

↑ MSKS(153)

Operand Data Areas

N: Interrupt source	# (0 to 5)
S: Data source word	CIO, G, A, T, C, #, DM, DR, IR

Description

When the execution condition is OFF, MSKS(153) is not executed. When the execution condition is ON, MSKS(153) masks interrupts from Interrupt Input Units (so that they are recorded but ignored) if N is 0 to 3 or sets the time interval for scheduled interrupts if N is 4 or 5.

N specifies the interrupt. Numbers 0 to 3 indicate I/O Interrupt Input Units 0 to 3, and numbers 4 and 5 indicate scheduled interrupts 0 and 1, respectively. The CV500 or CVM1-CPU01-EV2 has scheduled interrupt 0 only.

If N is 0 to 3, it designates an Interrupt Input Unit, and MSKS(153) causes the bits of the designated Interrupt Input Unit corresponding to ON bits in S to be masked, and those corresponding to OFF bits in S to be unmasked. All masked interrupts will still be recorded. When a masked bit has been recorded as being ON, the interrupt program for it will be run as soon as the bit is unmasked (unless it is cleared first; see *5-31-2 CLEAR INTERRUPT: CLI(154)*). All interrupts are initially masked.

If N is 4 or 5, it designates a scheduled interrupt and MSKS(153) sets the time interval for the scheduled interrupt. The interval between interrupts can be set between 10 and 99,990 ms, depending on both the content of S and the time unit set in the PC Setup. S can have any value between 0001 and 9999, and time units can be and set to 0.5 ms, 1 ms, or 10 ms.

To cancel the scheduled interrupt, execute MSKS(153) with S set to 0000. This is the initial setting. Refer to *5-31-2 CLEAR INTERRUPT: CLI(154)* for an example of scheduled interrupts.

CLI(154) should be used to set the time to the first scheduled interrupt. Unstable operation may result if the time to the first interrupt is not set. Refer to *5-31-2 CLEAR INTERRUPT: CLI(154)*.

Precautions

N must be between 0 and 5 for the CV1000, CV2000, CVM1-CPU11-EV2, or CVM1-CPU21-EV2 or between 0 and 4 for the CV500 or CVM1-CPU01-EV2.

S must be between 0000 and 00FF when N is between 0 and 3. S must be BCD between 0001 and 9999 when N is 4 or 5.

I/O interrupts and scheduled interrupts are masked when power is turned on or the PC mode is changed.

Both the scheduled interrupt time and the time to the first scheduled interrupt must be 10 ms or greater.

Note Refer to page 115 for general precautions on operand data areas.

Flags

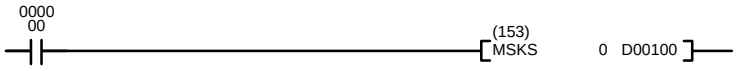
ER (A50003): N or S contain improper data
(e.g., data in S is not BCD when N is 4 or 5).

Content of ∗DM word is not BCD when set for BCD.

A value of 5 is entered for N in the CV500 or CVM1-CPU01-EV2.

Example

In the following example, inputs 0 to 3 of Interrupt Input Unit 0 are unmasked, and inputs 4 to 7 are masked when CIO 000000 is ON.

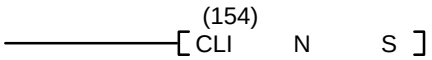


Address	Instruction	Operands
00000	LD	000000
00001	MSKS(153)	
		0
		D00100

D00100	0 0 F 0
D00101	0 0 F 2

5-31-2 CLEAR INTERRUPT: CLI(154)

Ladder Symbol



Variations

↑ CLI(154)

Operand Data Areas

N: Interrupt source	# (0 to 5)
S: Data source word	CIO, G, A, T, C, #, DM, DR, IR

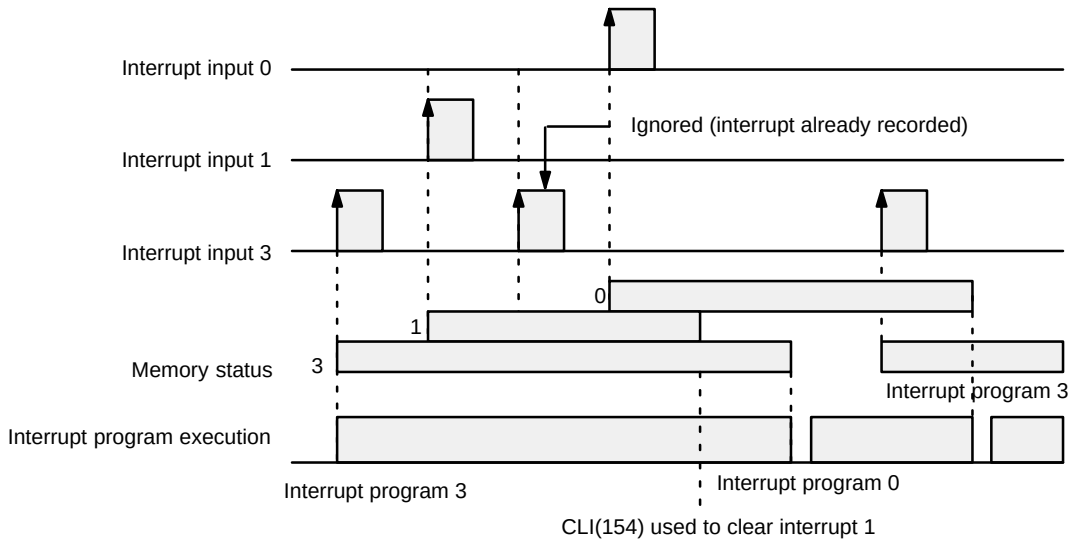
Description

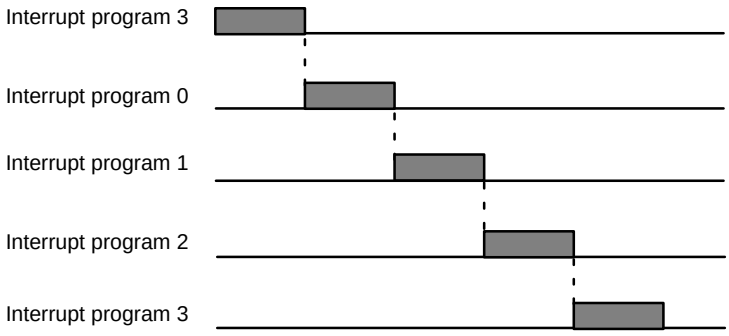
When the execution condition is OFF, CLI(154) is not executed. When the execution condition is ON, CLI(154) clears the recorded interrupts of the designated Interrupt Input Unit if N is 0 to 3, or sets the time to the first scheduled interrupt if N is 4 or 5.

N specifies the interrupt. Numbers 0 to 3 indicate I/O Interrupt Input Units 0 to 3, and numbers 4 and 5 indicate scheduled interrupts 0 and 1, respectively. The CV500 or CVM1-CPU01-EV2 has scheduled interrupt 0 only.

Because interrupt inputs are stored, masked interrupts will be serviced after the mask is removed, unless they are cleared first. If N designates an Interrupt Input Unit, CLI(154) clears the recorded interrupts of the inputs from the designated Interrupt Input Unit corresponding to ON bits in S. Once an interrupt is cleared, the interrupt program will not be executed even if the interrupt is unmasked.

In the following example, CLI(154) is executed with S=00F2, so the recorded interrupts for inputs 1, 4, 5, 6, and 7 are cleared. Recorded interrupts for inputs 0, 2, and 3 are unaffected.





Numbers 4 or 5 designate a scheduled interrupt, and CLI(154) sets the time to the first interrupt. The time to the first interrupt can be set between 10 and 99,990 ms, depending on both the content of S and the time unit set in the PC Setup. S can have any value between 0001 and 9999, and time units can be and set to 0.5 ms, 1 ms, or 10 ms.

⚠ Caution CLI(154) can be used to change the time interval to the next interrupt for one cycle. The new time interval is effective immediately. The scheduled interrupt may never actually occur if the time to the first interrupt is changed repeatedly, i.e., before the interrupt has time to occur.

Precautions

N must be between 0 and 5 for the CV1000, CV2000, CVM1-CPU11-EV2, or CVM1-CPU21-EV2 or between 0 and 4 for the CV500 or CVM1-CPU01-EV2.

S must be between 0000 and 00FF when N is between 0 and 3. S must be BCD between 0000 and 9999 when N is 4 or 5.

If the ER Flag (A50003) is turned ON, the instruction will not be executed.

I/O interrupts and scheduled interrupts are masked when power is turned on or the PC mode is changed.

Both the scheduled interrupt time and the time to the first scheduled interrupt must be 10 ms or longer.

END(001) is required at the end of both the main program and scheduled interrupt program.

Note Refer to page 115 for general precautions on operand data areas.

Flags

ER (A50003): N or S contain incorrect data (e.g., data in S is not BCD when N is 4 or 5).

Content of ∗DM word is not BCD when set for BCD.

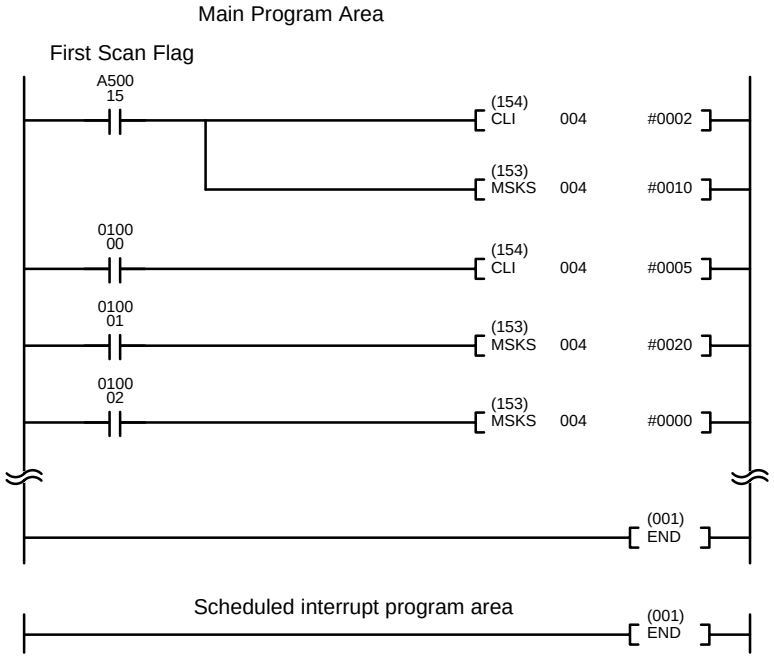
A value of 5 is entered for N in the CV500 or CVM1-CPU01-EV2.

Example

The following program shows the overall structure and operation of the scheduled interrupt.

Here, the interrupt program is started 20 ms after execution of CLI(154), and repeated every 100 ms thereafter. When bit 010000 is turned ON, CLI(154) changes the time interval to the next interrupt to 50 ms for one cycle. When bit 010001 is turned ON, MSKS(153) changes the scheduled time interval to 200 ms. The scheduled interrupts continue until bit 010002 is turned ON, and MSKS(153) is executed with S set to 0000.

The control flow logic of the main program is unaffected by execution of the interrupt program, i.e., immediately after the interrupt program has finished execution, control returns to the point in the main program where it was suspended.



Address	Instruction	Operands
00000	LD	A50015
00001	CLI(154)	
		004
		#0002
00002	MSKS(153)	
		004
		#0010
00003	LD	010000
00004	CLI(154)	
		004
		#0005
00005	LD	010001
00006	MSKS(153)	
		004
		#0020
00007	LD	010002
00008	MSKS(153)	
		004
		#0000
00500	END(001)	
Scheduled interrupt program area		
00600	END(001)	

5-31-3 READ MASK: MSKR(155)

Ladder Symbol

(155)
MSKR N D

Operand Data Areas

N: Interrupt source # (0 to 5)

D: Destination word CIO, G, A, DM, DR, IR

Variations

↑ MSKR(155)

Description When the execution condition is OFF, MSKR(155) is not executed. When the execution condition is ON, MSKR(155) writes the current mask status of the designated Interrupt Input Unit into D if N is 0 to 3, or writes the scheduled interrupt interval into D if N is 4 or 5. If N is between 0 and 3, it designates the unit number of the Interrupt Input Units. If N is 4, it designates scheduled interrupt #0. If N is 5, it designates scheduled interrupt #1.

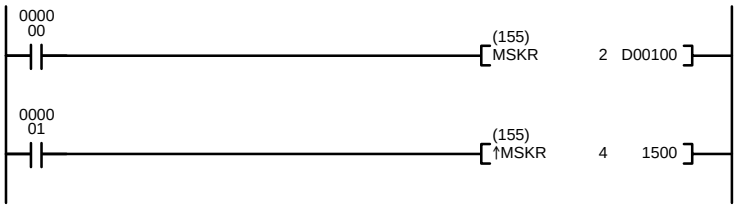
Precautions N must be between 0 and 5 for the CV1000, CV2000, CVM1-CPU11-EV2, or CVM1-CPU21-EV2 or between 0 and 4 for the CV500 or CVM1-CPU01-EV2.

Note Refer to page 115 for general precautions on operand data areas.

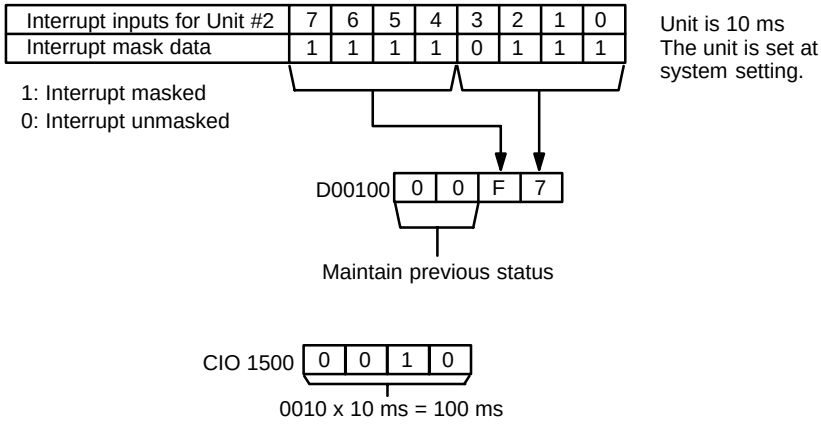
Flags ER (A50003): N contains incorrect data.
A value of 5 is entered for N in the CV500 or CVM1-CPU01-EV2.
Content of ∗DM word is not BCD when set for BCD.

Example

In the following example, MSKR(155) writes the current mask status of Interrupt Input Unit number 2 into D00100 when CIO 000000 is ON. ↑MSKR(155) writes the time interval for scheduled interrupt 0 into CIO 1500 the next execution after CIO 000001 turns ON.



Address	Instruction	Operands
00000	LD	000000
00001	MSKR(155)	
		2
		D00100
00002	↑MSKR(155)	
		4
		1500

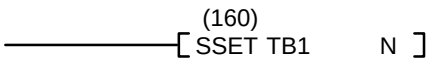


5-32 Stack Instructions

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.

5-32-1 SET STACK: SSET(160)

Ladder Symbol



Variations

↑ SSET(160)

Operand Data Areas

TB1: 1st stack address	CIO, G, A, DM
N: Number of words	CIO, G, A, T, C, #, DM, DR, IR

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 TB+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 115 for general precautions on operand data areas.

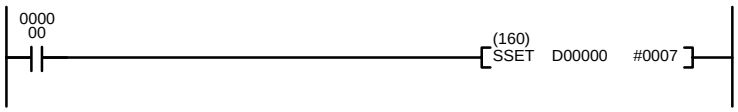
Flags

ER (A50003): Content of N is less than 0003, or is not BCD.

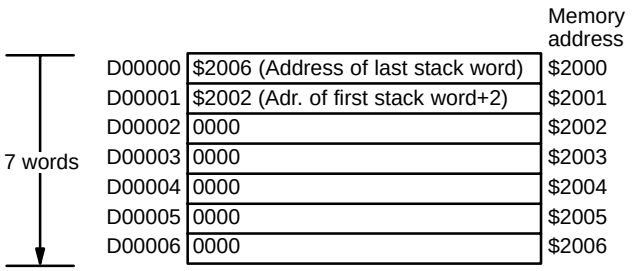
Content of ∗DM word is not BCD when set for BCD.

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.

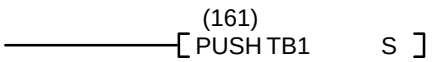


Address	Instruction	Operands
00000	LD	000000
00001	SSET(160)	
		D00000
		#0007



5-32-2 PUSH ONTO STACK: PUSH(161)

Ladder Symbol



Operand Data Areas

TB1: 1st stack address	CIO, G, A, DM
S: Source word	CIO, G, A, T, C, #, DM, DR, IR

Variations

↑ PUSH(161)

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 115 for general precautions on operand data areas.

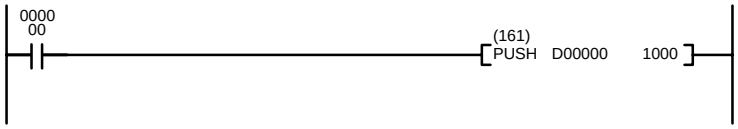
Flags

ER (A50003): Content of TB1+1 is greater than the content of TB1.

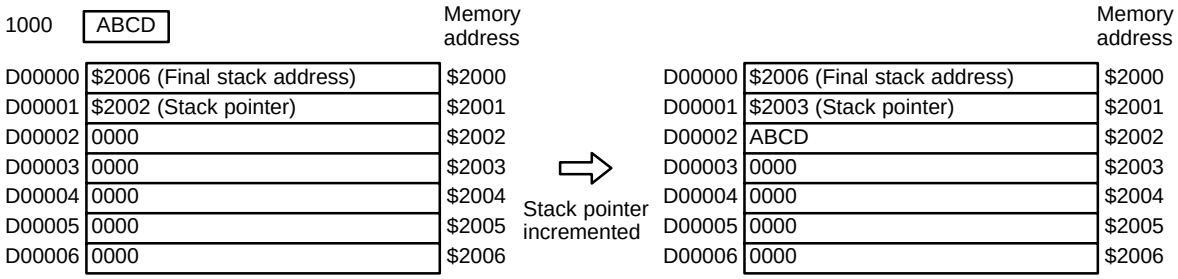
Content of ∗DM word is not BCD when set for BCD.

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.

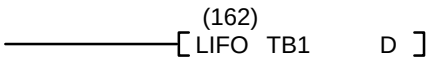


Address	Instruction	Operands
00000	LD	000000
00001	PUSH(161)	
		D00000
		1000



5-32-3 LAST IN FIRST OUT: LIFO(162)

Ladder Symbol



Variations

↑ LIFO(162)

Operand Data Areas

TB1: 1st stack address	CIO, G, A, DM
D: Destination word	CIO, G, A, DM, DR, IR

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.

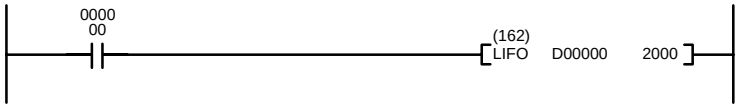
Note Refer to page 115 for general precautions on operand data areas.

Flags

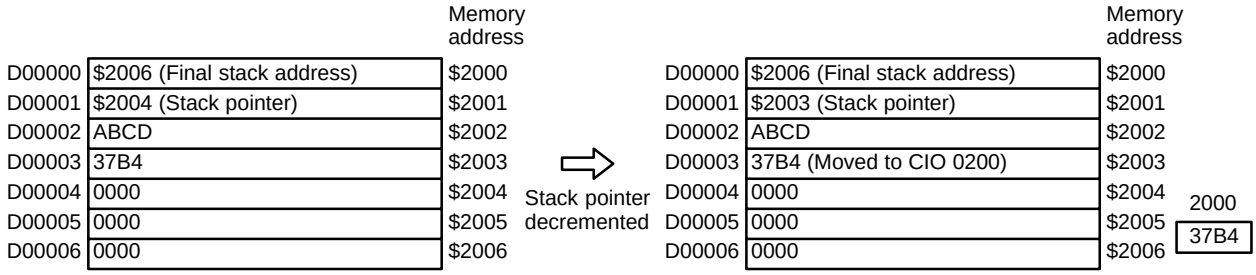
ER (A50003): Content of TB1+1 is less than or equal to the address of TB1+1.
Content of *DM word is not BCD when set for BCD.

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.

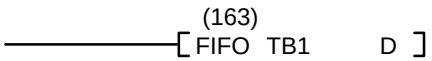


Address	Instruction	Operands
00000	LD	000000
00001	LIFO(162)	
		D00000
		2000



5-32-4 FIRST IN FIRST OUT: FIFO(163)

Ladder Symbol



Operand Data Areas

TB1: 1st stack address	CIO, G, A, DM
D: Destination word	CIO, G, A, DM, DR, IR

Variations

↑ FIFO(162)

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.

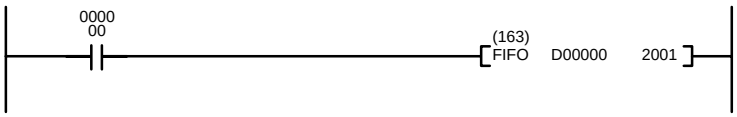
Note Refer to page 115 for general precautions on operand data areas.

Flags

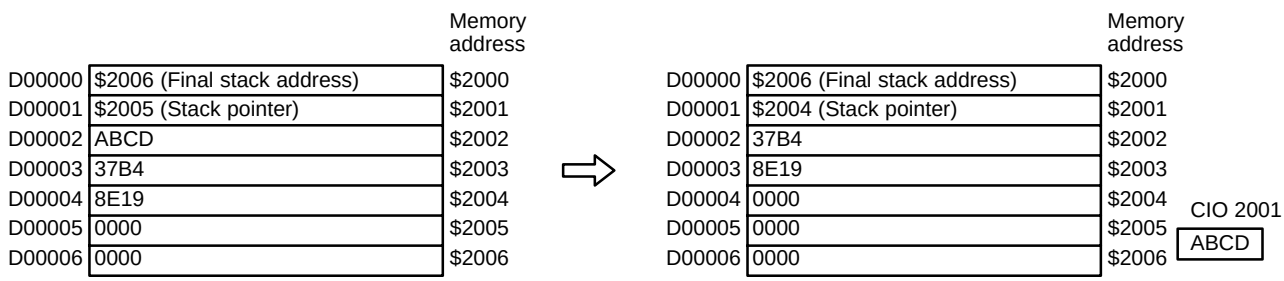
ER (A50003): TB1+1 is less than or equal to the address of the stack pointer.
Content of ∗DM word is not BCD when set for BCD.

Example

When CIO 000000 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

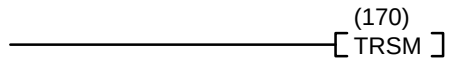


5-33 Data Tracing

Data tracing can be used to facilitate debugging programs and is described in detail in the CVSS *Operation Manual: Online*. This section shows the ladder symbols for TRSM(170) and MARK(171) and provides example programs.

5-33-1 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 CVSS. 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 CVSS. 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 CVSS so that the specified data will be traced at a regular interval independent of the cycle time (refer to *4-2 Data Tracing* in the *CV Support Software: Online 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 CVSS.

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 CVSS.

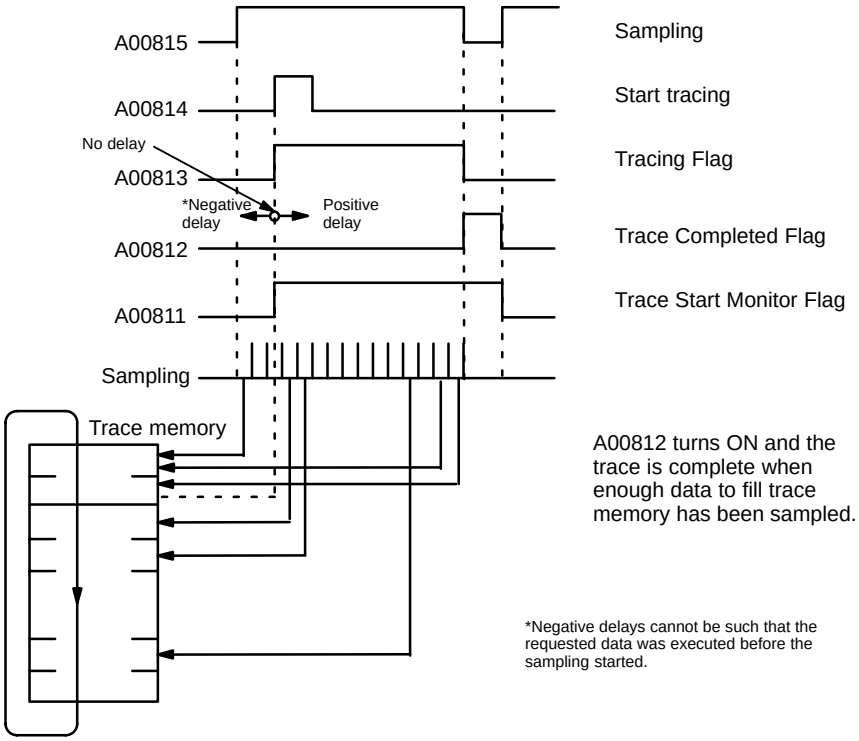
Flag	Function
A00811	Trace Trigger Monitor Flag
A00812	Trace Completed Flag
A00813	Trace Busy Flag
A00814	Trace Start Bit
A00815	Sampling Start Bit

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



5-33-2 MARK TRACE: MARK(174)

Ladder Symbol	Operand Data Area
(174) MARK N	**N: Mark number** #

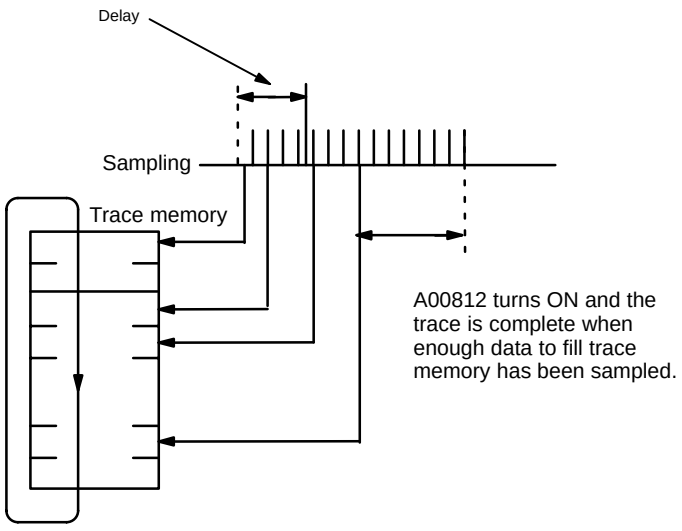
Description

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 CVSS *Operation Manual: Online* for details.

The word addresses, trigger conditions, and delay must be set beforehand from the CVSS. 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.

5-34 Memory Card Instructions

Memory Card Instructions all involve the transfer of data to and from the Memory Card in the memory card drive. The instructions described in this section can thus only be used if there is a Memory Card in the drive.

Exercise care when transferring a very large number of words, because it can greatly increase the overall cycle time.

The following flags are used by all of the Memory Card instructions. Refer to *3-6-20 Memory Card Flags* for details.

A343 bit(s)	Function
00 to 03	Memory Card Type (0:None, 1: RAM, 2: EPROM, 3: EEPROM)
07	Memory Card Format Error Flag
08	Memory Card Transfer Error Flag
09	Memory Card Write Error Flag
10	Memory Card Read Error Flag
11	FIle Missing Flag
12	Memory Card Write Flag
13	Memory Card Instruction Flag
14	Accessing Memory Card Flag
15	Memory Card Protected Flag

A346 contains the number of words (4-digit BCD) left to transfer from the Memory Card with FILR(180)/FILW(181).

5-34-1 READ DATA FILE: FILR(180)

Ladder Symbol

```
     (180)
 ----[ FILR    N     D     C ]
```

Variations

↑ FILR(180)

Operand Data Areas

N: Words to transfer	CIO, G, A, T, C, #, DM, DR, IR
D: 1st destination word	CIO, G, A, T, C, DM
C: 1st control word	CIO, G, A, T, C, DM

Description

When the execution condition is OFF, FILR(180) is not executed. When the execution condition is ON, FILR(180) reads N words of data from the memory card data file (*filename*.IOM) specified in C+1 to C+4, and outputs the data to the designated memory area beginning at D.

The name of the file from which the data is read is specified by eight ASCII characters in C+1 to C+4. Data will be read from the word indicated in C+5 if C bit 04 (the Offset Enable Bit) is ON. The following table shows the function of bits in the control words.

Word	Bit(s)	Function	Bit(s)	Function
C	04	Offset Enable Bit (ON: enabled, OFF: disabled)	Other	Turn OFF.
C+1	08 to 15	First character in name	00 to 07	Second character in name
C+2	08 to 15	Third character in name	00 to 07	Fourth character in file
C+3	08 to 15	Fifth character in name	00 to 07	Sixth character in file
C+4	08 to 15	Seventh character in name	00 to 07	Eighth character in file
C+5	00 to 15	Offset from beginning of file (0000 to 9999, BCD)		

If the filename is less than 8 characters long, enter #20 for the bytes that aren't required. It is not necessary to input the filename extension ".IOM."

When FILR(180) is executed, the CPU first checks whether the ER Flag (A50003) is ON, then processes the data transfer and following instructions in parallel, so check the Memory Card Instruction Flag (A34313) to verify that FILR(180) has been completed correctly.

Precautions

The number of words to transfer (N) must be BCD.

If bit 04 of C is ON, C+5 must be BCD.

If N exceeds the number of words remaining in the file, only the words in the file will be transferred and no error will occur.

Write the execution condition for FLSP(183) so that the instruction will not be executed if the Memory Card Instruction Flag (A34313) is ON (when another memory card instruction is being executed).

Note Refer to page 115 for general precautions on operand data areas.

Flags

ER (A50003): N is not BCD.
Bit 04 of C is ON, but the content C+5 is not BCD.
Content of ∗DM word is not BCD when set for BCD.
A Memory Card is not mounted.

A34310: The file not read if the offset in C+5 is larger than the file.

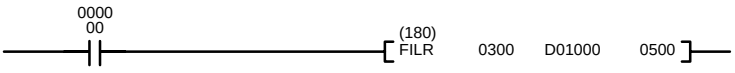
A34311: The specified file doesn't exist on the card.

Example

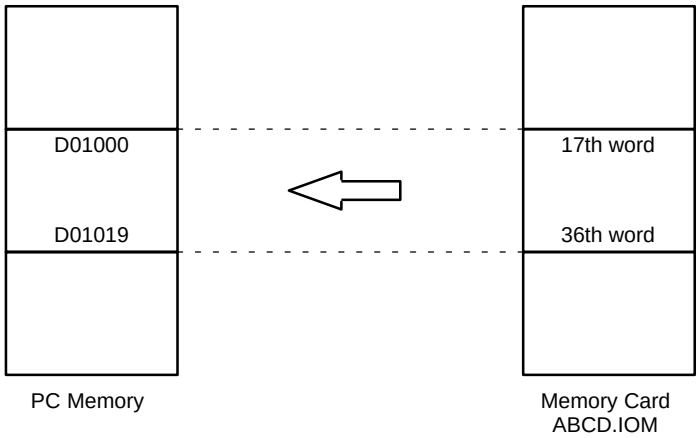
In the following example, 20 words, beginning from the 17th word of the file, are transferred from memory card data file "ABCD" to D01000 to D01019.

Here, the content of CIO 0300 would be 0020 (BCD) to indicate reading 20 words. The contents of the control words would be as follows:

Word	Leftmost byte	Rightmost byte	Meaning
CIO 0500	0 0	1 0	Enable offset
CIO 0501	4 1	4 2	A B
CIO 0502	4 3	4 4	C D
CIO 0503	2 0	2 0	Indicates end of name
CIO 0504	2 0	2 0	Indicates end of name
CIO 0505	0 0	1 7	Offset

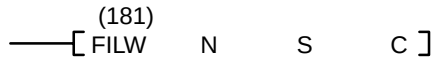


Address	Instruction	Operands
00000	LD	000000
00001	FILR(180)	
		0300
		D01000
		0500



5-34-2 WRITE DATA FILE: FILW(181)

Ladder Symbol



Variations

↑ FILW(181)

Operand Data Areas

N: Words to transfer CIO, G, A, T, C, #, DM, DR, IR

S: 1st source word CIO, G, A, T, C, DM

C: 1st control word CIO, G, A, T, C, DM

Description

When the execution condition is OFF, FILW(181) is not executed. When the execution condition is ON, FILW(181) writes the data in S to S+N–1 to a Memory Card data file (*FILENAME*.IOM) specified in C+1 to C+4. The data will replace data in the file if C bit 07 (the Overwrite Bit) is ON, or be added to the end of the file if C bit 07 is OFF.

The name of the file to which the data is written is specified by eight ASCII characters in C+1 to C+4. Data will be written from the word indicated in C+5 if C bit 04 (the Offset Enable Bit) is ON. If the specified file name does not exist, a new file by that name will be created; the data will be written from the beginning of the file, regardless of the status of the Offset Enable bit. The following table shows the function of bits in the control words.

Word	Bit(s)	Function	Bit(s)	Function
C	04	Offset Enable Bit (ON: enabled, OFF: disabled)	07	Overwrite Bit (ON: replace data, OFF: add data)
C+1	08 to 15	First character in name	00 to 07	Second character in name
C+2	08 to 15	Third character in name	00 to 07	Fourth character in file
C+3	08 to 15	Fifth character in name	00 to 07	Sixth character in file
C+4	08 to 15	Seventh character in name	00 to 07	Eighth character in file
C+5	00 to 15	Offset from beginning of file (0000 to 9999, BCD)		

If the filename is less than 8 characters long, enter #20 to the bytes that aren't required. It is not necessary to input the filename extension ".IOM."

When FILW(181) is executed, the CPU first checks whether the ER Flag (A50003) is ON, then processes the data transfer and following instructions in parallel, so check the Memory Card Write Flag (A34312) or the Memory Card Instruction Flag (A34313) to verify that FILW(181) has been completed correctly.

Precautions

The number of words to transfer (N) must be BCD.

If bit 04 of C is ON, C+5 must contain BCD.

Do not use the following filenames: AUX, CON, LST, PRN, NUL.

If N exceeds the number of words remaining in the file, data will be transferred until the end of the file is reached, the remaining data will not be transferred and no error will occur.

Write the execution condition for FILW(181) so that the instruction will not be executed if the Memory Card Instruction Flag (A34313) is ON (when another Memory Card instruction is being executed).

Note Refer to page 115 for general precautions on operand data areas.

Flags

ER (A50003): N is not BCD.
Bit 04 of C is ON, but the content C+5 is not BCD.
Content of *DM word is not BCD when set for BCD.
A Memory Card is not mounted.

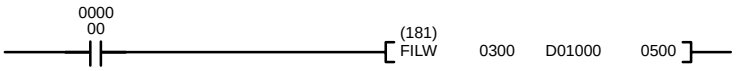
A34308: Turned ON and the data not written if the offset in C+5 is larger than the file size.

Example

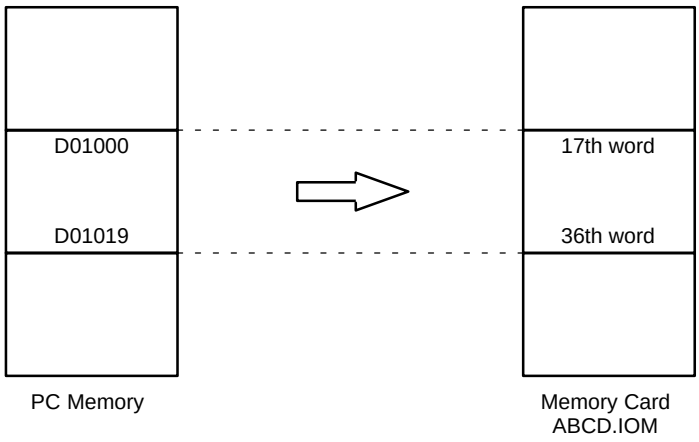
In the following example the data in D01000 to D01019 is written over the 20 words in memory card data file “ABCD” beginning at the 17th word of file.

Here, the content of CIO 0300 would be 0020 (BCD) to indicate reading 20 words. The contents of the control words would be as follows:

Word	Leftmost byte	Rightmost byte	Meaning
CIO 0500	0 0	9 0	Overwrite; enable offset
CIO 0501	4 1	4 2	A B
CIO 0502	4 3	4 4	C D
CIO 0503	2 0	2 0	Indicates end of name
CIO 0504	2 0	2 0	Indicates end of name
CIO 0505	0 0	1 7	Offset

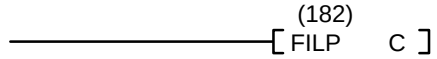


Address	Instruction	Operands
00000	LD	000000
00001	FILW(181)	
00002		0300
		D01000
		0500



5-34-3 READ PROGRAM FILE: FILP(182)

Ladder Symbol



Operand Data Area

C: 1st control word CIO, G, A, T, C, DM

Variations

↑ FILP(182)

Description

When the execution condition is OFF, FILP(182) is not executed. When the execution condition is ON, FILP(182) reads the ladder program file (*filename*.LDP) specified in C+1 to C+4 from the Memory Card, and writes the data over the current ladder program beginning at the instruction just after FILP(182). The program file must be written to the Memory Card beforehand with the CVSS.

The program will be executed from the beginning when FILP(182) has been completed.

Word	Bit(s)	Function	Bit(s)	Function
C	07	Write Method (ON: overwrite, OFF: replace to END(001))	Other	Set to OFF.
C+1	08 to 15	First character in name	00 to 07	Second character in name
C+2	08 to 15	Third character in name	00 to 07	Fourth character in file
C+3	08 to 15	Fifth character in name	00 to 07	Sixth character in file
C+4	08 to 15	Seventh character in name	00 to 07	Eighth character in file

If the filename is less than 8 characters long, enter #20 to the bytes that aren't required. It is not necessary to input the filename extension ".LDP."

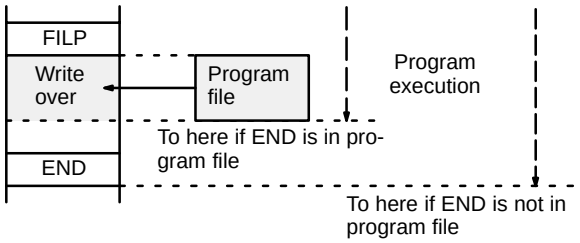
Write the execution condition so that the instruction will not be executed the first time it is examined (i.e., the first cycle when SFC programming is not being used and the first cycle that the step containing FILP(182) goes from inactive to active for SFC programming). If FILP(182) is executed during the first cycle, a non-fatal SFC continuation error will occur.

When executing FILP(182), set the maximum cycle time to 2 seconds in the PC Setup.

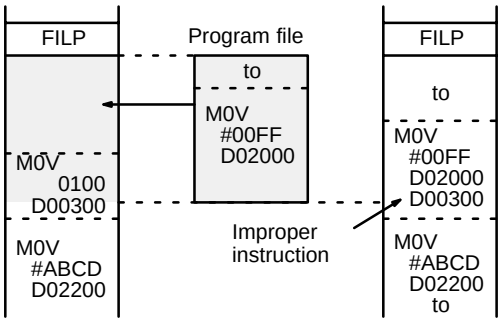
The program will not be executed for several cycles (30 seconds or more max.) when FILP(182) is executed; the cycle time will be reduced during this period.

Overwrite Method

When the Write Method Bit (C bit 07) is ON, FILP(182) will erase just enough of the current ladder program to accommodate the program file. If the program file transferred from the Memory Card ends with END(001), the ladder program will end there. If the transferred program file does not end in END(001), the ladder program will be executed until the END(001) of the original program is reached. A program file that is longer than the original ladder program from FILP(182) to END(001) must have its own END(001) instruction because the END(001) instruction in the original ladder program will be overwritten.



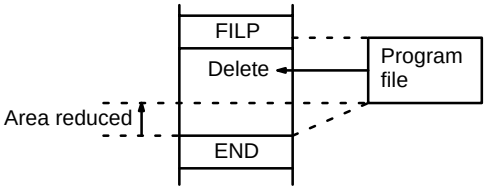
If the program file is shorter than the original ladder program from FILP(182) to END(001) and doesn't end in END(001), be sure that the program file doesn't overwrite only the first part of an instruction in the original ladder program creating an instruction format error.



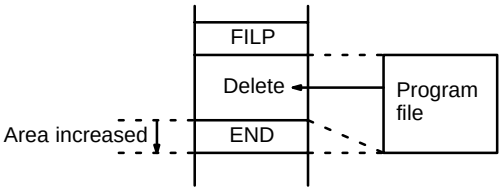
Replace to END(001) Method

When the Write Method Bit (C bit 07) is OFF, FILP(182) will erase the current ladder program from the instruction just after FILP(182) to END(001), then insert the program file. It is not necessary for the program file to end in END(001).

If the program file is shorter than the original ladder program from FILP(182) to END(001), the ladder program will be shorter after FILP(182) is executed.



If the program file is longer than the original ladder program from FILP(182) to END(001), the ladder program will be longer after FILP(182) is executed.



Precautions

FILP(182) cannot be used in an interrupt program.

The instruction trace operation and online editing cannot be performed while FILP(182) is being executed.

Write the execution condition for FILP(182) so that the instruction will not be executed if the Memory Card Instruction Flag (A34313) is ON (when another Memory Card instruction is being executed).

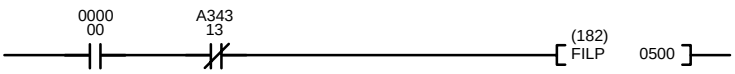
Note Refer to page 115 for general precautions on operand data areas.

Flags

ER (A50003): A Memory Card is not mounted.
Content of ∗DM word is not BCD when set for BCD.

Example

In the following example, the ladder program file "ABCD.LDP" is transferred from the Memory Card using the "replace to END(001)" method. When CIO 000000 is ON and A34313 (Memory Card Instruction Flag) is OFF, the contents of ABCD.LDP is written over all instructions after the instruction line containing FILP(182).

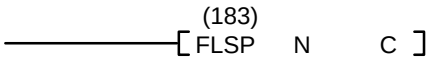


Address	Instruction	Operands
00000	LD	000000
00001	AND NOT	A34313
00002	FILP(182)	
		0500

	MSB			LSB	
CIO 0500	0	0	0	0	—Overwrite
CIO 0501	4	1	4	2	—“A,” “B”
CIO 0502	4	3	4	4	—“C,” “D”
CIO 0503	2	0	2	0	
CIO 0504	2	0	2	0	

5-34-4 CHANGE STEP PROGRAM: FLSP(183)

Ladder Symbol



Operand Data Areas

N: Step number	CIO, G, A, T, C, #, DM, DR, IR
C: 1st control word	CIO, G, A, T, C, DM

Variations

↑ FLSP(183)

Description

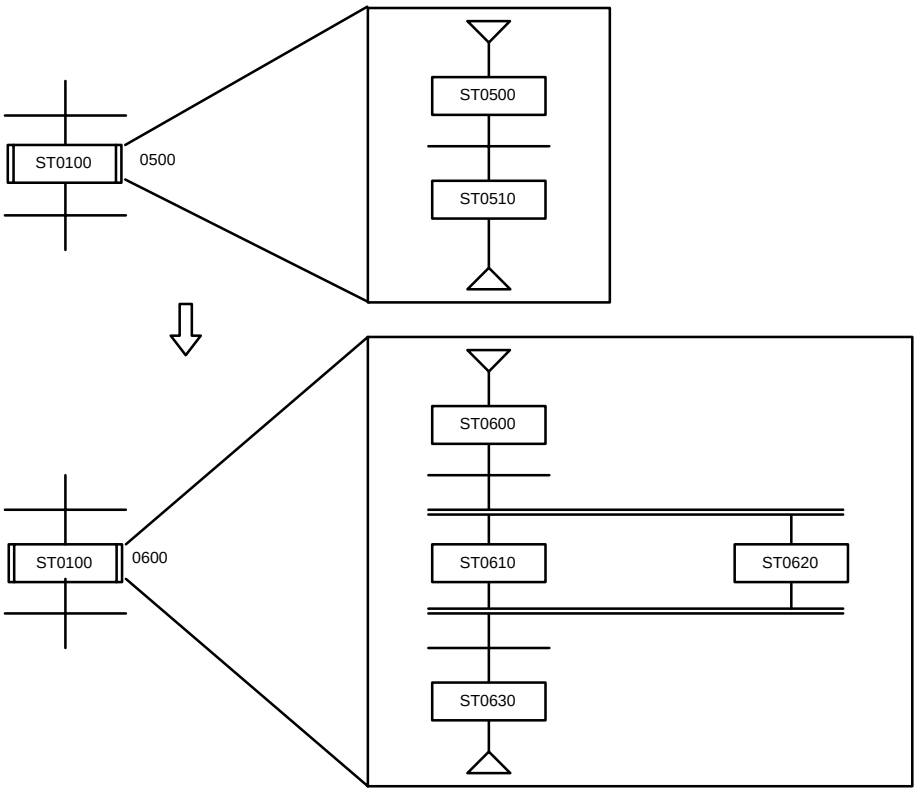
When the execution condition is OFF, FLSP(183) is not executed. When the execution condition is ON, FLSP(183) reads the step program file specified in C+1 to C+4 (*filename*.SFC) from the Memory Card and writes the data over the step indicated by N, changing the contents of the action block. The step program file must be written to the Memory Card beforehand with the CVSS.

A step that is active or that contains actions that are still being executed cannot be changed. Also, the number of actions in the new step must be the same as the number in the step that is being overwritten.

The contents of C must be all zeros to indicate the Memory Card and the file name is specified in C+1 to C+4 as shown in the following table. If the filename is less than 8 characters long, enter #20 for the bytes that aren’t required. It is not necessary to input the filename extension “.SFC.” It is assumed.

Word	Bit(s)	Function	Bit(s)	Function
C	00 to 15	OFF to indicate the Memory Card.		
C+1	08 to 15	First character in name	00 to 07	Second character in name
C+2	08 to 15	Third character in name	00 to 07	Fourth character in file
C+3	08 to 15	Fifth character in name	00 to 07	Sixth character in file
C+4	08 to 15	Seventh character in name	00 to 07	Eighth character in file

The contents of a subchart will be changed if the step number of a subchart dummy step is specified for N. The number of the steps in the subchart can be changed, as shown in the following diagram.



Precautions

FLSP(183) can be used only by CPUs that support SFC programming.

N must be BCD between 0000 and 0511 for the CV500, or between 0000 and 1023 for the CV1000 or CV2000.

FLSP(183) cannot be used in an interrupt program.

Online editing cannot be performed while FLSP(183) is being executed.

Write the execution condition for FLSP(183) so that the instruction will not be executed if the Memory Card Instruction Flag (A34313) is ON (when another Memory Card instruction is being executed).

The number of actions in the step program file must match the number in the step being replaced. If the number of actions is not the same, the Memory Card Read Error Flag (A34310) will be turned ON and the instruction won't be executed.

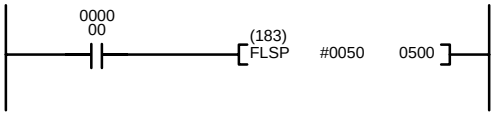
Note Refer to page 115 for general precautions on operand data areas.

Flags

ER (A50003):	A Memory Card is not mounted.
	N is not BCD between 0000 and 0511 for the CV500, or between 0000 and 1023 for the CV1000 or CV2000.
	Content of ∗DM word is not BCD when set for BCD.
A34310:	The number of actions in the step program file doesn't match the number in the step being changed.
	The indicated step is active, or an action within the step with an action qualifier of "S" is being executed.
	The indicated step doesn't exist.

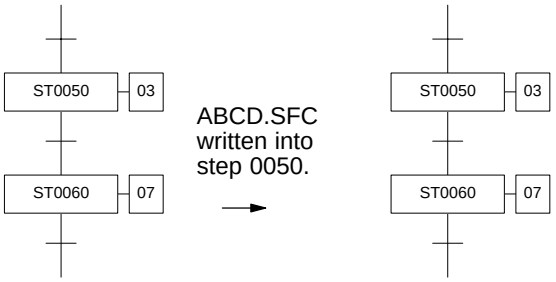
Example

When CIO 000000 is ON in the following example with the memory contents shown, the contents of step 0050 will be overwritten with the contents of ABCD.SFC. The structure of the SFC program will not change, but the contents of the action block for step 0050 will be replaced with the contents of ABCD.SFC. The extension ".SFC" is assumed and is not input.



Address	Instruction	Operands
00000	LD	000000
00001	FLSP(183)	
		#0050
		0500

Word	Leftmost byte	Rightmost byte	Meaning
CIO 0500	0 0	0 0	Indicates the Memory Card
CIO 0501	4 1	4 2	A B
CIO 0502	4 3	4 4	C D
CIO 0503	2 0	2 0	Indicates end of name
CIO 0504	2 0	2 0	Indicates end of name



Action Block for Step 0050

AQ	SV	Action	FV
N		AC 0300	
N		AC 0310	
N		AC 0320	

ABCD.SFC written into step 0050.

Action Block for Step 0050

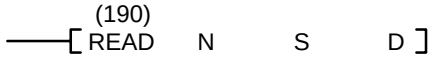
AQ	SV	Action	FV
N		AC 0550	
S		AC 0580	
N		AC 0800	

5-35 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.

5-35-1 I/O READ: READ(190)

Ladder Symbol



Variations

↑ READ(190)

Operand Data Areas

N: Words to transfer	CIO, G, A, T, C, #, DM, DR, IR
S: Source word	CIO
D: 1st destination word	CIO, G, A, T, C, DM

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 43 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.

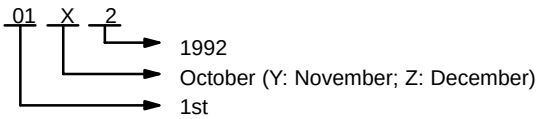
If the data cannot be sent or the Special I/O Unit is busy, the data will be transferred during the next cycle. To make sure that READ(190) execution has been completed, check EQ (A50006).

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 115 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.
The Slave is not set to 54MH.

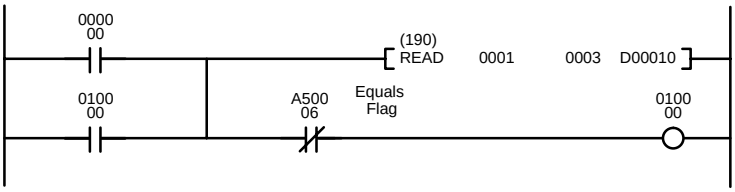
CY (A50004): This Flag operates only for Special I/O Units mounted to SYSMAC BUS/2 Slave Racks and turns on for the following:
The Special I/O Unit or Slave is busy.
An error has occurred in the Special I/O Unit.
Word operands have been designated for a Special I/O Unit that requires a constant.
The 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.

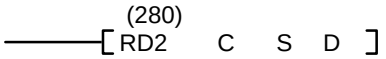
The following program section uses a self-holding bit to ensure that the read operation is executed even if the Special I/O Unit (or Slave Rack) is busy when READ(190) is executed. The type of programming is also effective when the read operation requires more than one cycle. The Equals Flag, which turns 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

5-35-2 I/O READ 2: RD2(280) (CVM1 V2)

Ladder Symbol



Operand Data Areas

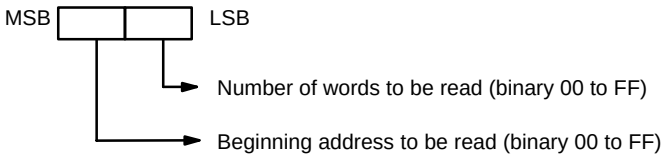
C: Control word	CIO, G, A, T, C, **#**, DM, DR, IR
S: Source word	CIO
D: First destination word	CIO, G, A, T, C, DM

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 SYSMAC BUS).

Note Refer to page 115 for general precautions on operand data areas.

Flags

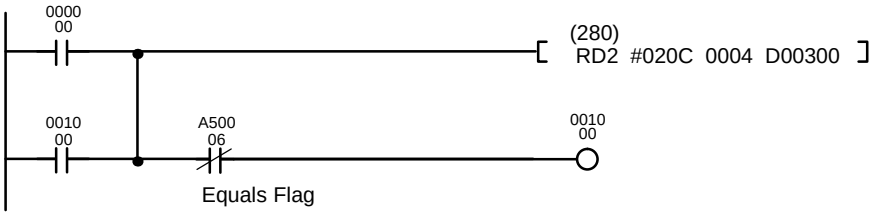
ER (A50003): Beginning address plus number of words exceeds 100 binary.
Content of ∗DM word is not BCD when set for BCD.

CY (A50004): An error has occurred at the Special I/O Unit.

EQ (A50006): The read operation has been completed.

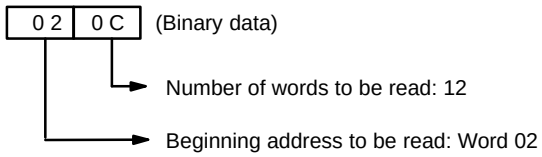
Example

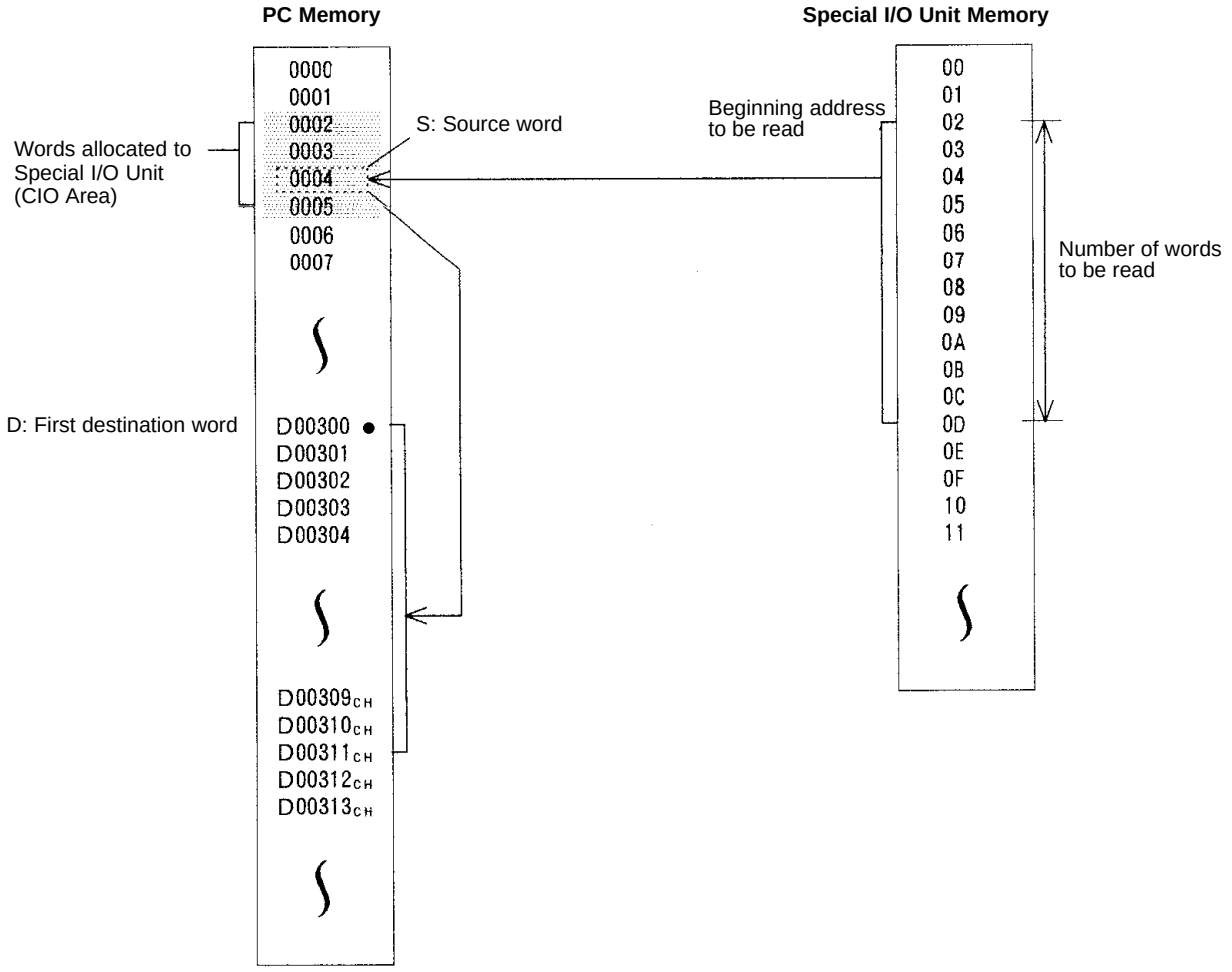
When CIO 000000 is ON in the following example, the contents of words 02 through 12 in the Special I/O Unit's memory area are read in order, one word at a time, through CIO 0004, and the contents that are read are transferred in order to D00300 through D00311.



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





5-35-3 I/O WRITE: WRIT(191)

Ladder Symbol

(191)
WRIT N S D

Variations

↑ WRIT(191)

Operand Data Areas

N: Words to transfer	CIO, G, A, T, C, #, DM, DR, IR
S: 1st source word	CIO, G, A, T, C, DM
D: Destination word	CIO

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).

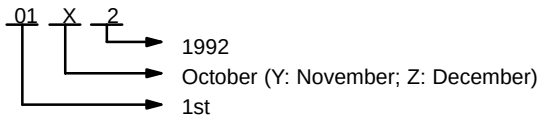
A maximum of two READ(190)/WRIT(191) instructions can be used for Slaves connected to the same Master. If a third instruction is attempted when two instructions are already being executed, the third instruction will not be executed

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 115 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.

The Slave is not set to 54MH.

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.

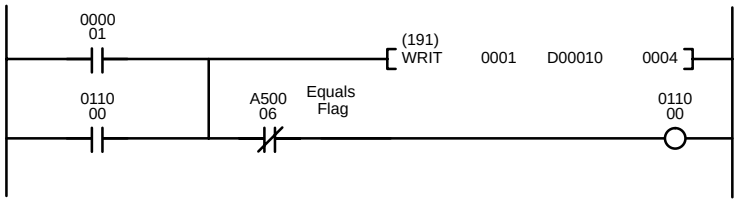
The communications path is not normal.

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).

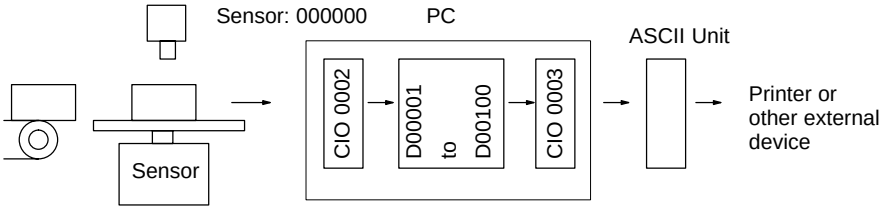
The following program section uses a self-holding bit to ensure that the write operation is executed even if the Special I/O Unit (or Slave Rack) is busy when WRIT(191) is executed. The type of programming is also effective when the read operation requires more than one cycle. The Equals Flag, which turns ON when the reading operation is completed, is used to end execution.

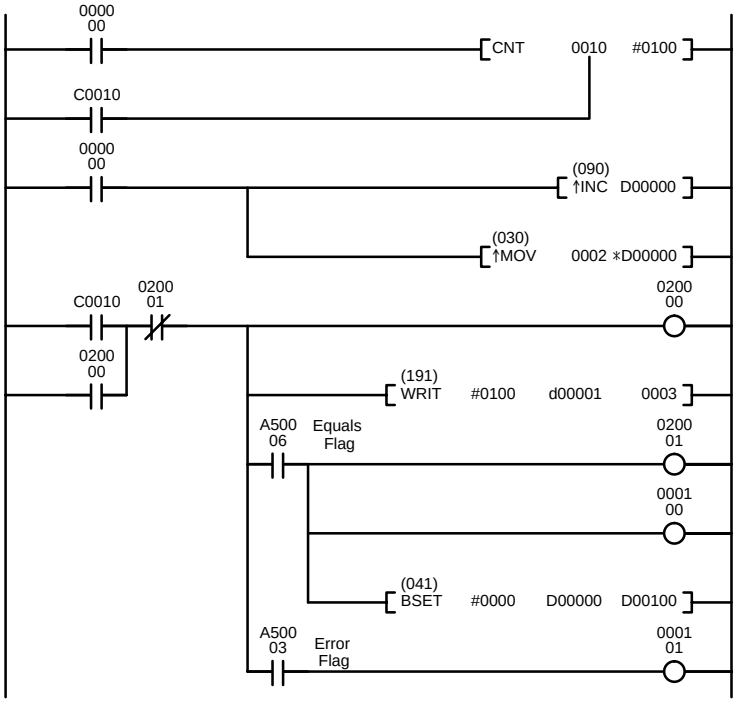


Address	Instruction	Operands
00000	LD	000000
00001	OR	011000
00002	WRIT(191)	
		0001
		D00010
		0004
00003	AND NOT	A50006
00004	OUT	011000

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.

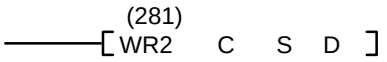




Address	Instruction	Operands
00000	LD	000000
00001	LD	C0010
00002	CNT	0010
		#0100
00003	LD	000000
00004	↑INC(090)	
		D00000
00005	↑MOV(030)	
		0002
		*D00000
00006	LD	C0010
00007	OR	020000
00008	AND NOT	020001
00009	OUT	020000
00010	WRIT(191)	
		#0100
		D00001
		0003
00011	AND	A50006
00012	OUT	020001
00013	OUT	000100
00014	BSET(041)	
		#0000
		D00000
		D0010
00015	AND	A50003
00016	OUT	000101

5-35-4 I/O WRITE 2: WR2(281) (CVM1 V2)

Ladder Symbol



Operand Data Areas

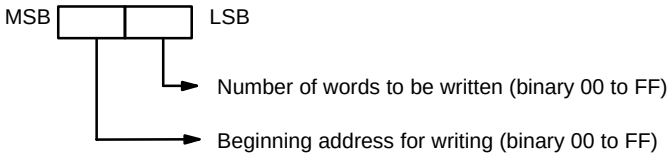
C: Control word	CIO, G, A, T, C, **#**, DM, DR, IR
S: First source word	CIO, G, A, T, C, DM
D: Destination word	CIO

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).

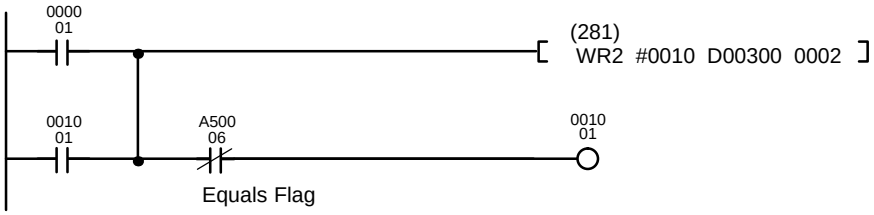
Note Refer to page 115 for general precautions on operand data areas.

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.

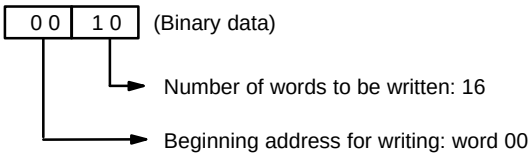
Example

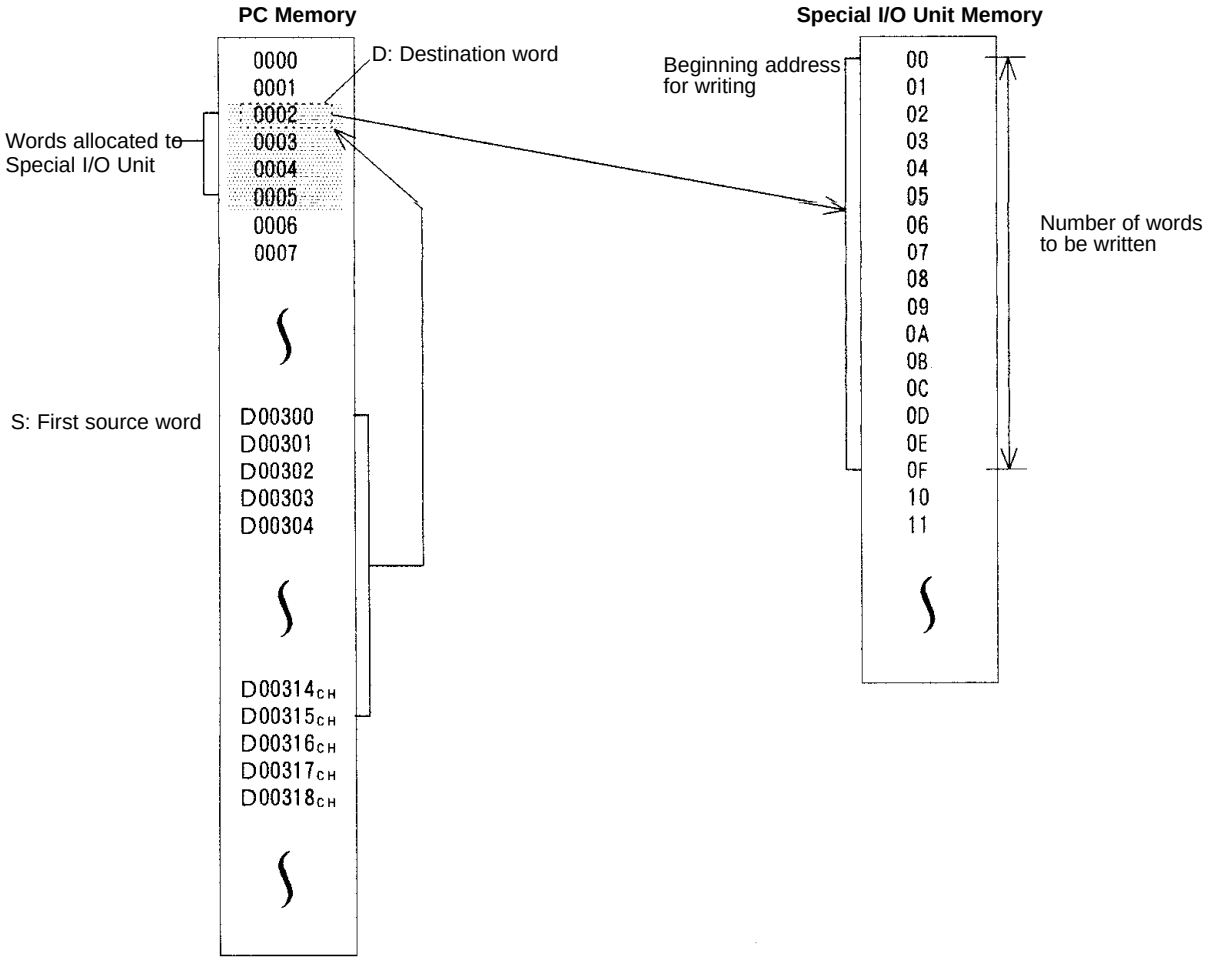
When CIO 000001 is ON in the following example, the 16 words beginning with D00300 are transferred in order through CIO 0002 and are written in order to words 00 through 0F in the Special I/O Unit's memory area.



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

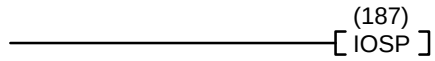




5-36 Network Instructions

The Network Instructions are used for communicating with or control PCs or other Units linked through the SYSMAC NET Link System or SYSMAC LINK System. The first two Network Instructions are used to control the access right to the PC from both local and remote Peripheral Devices.

5-36-1 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 from all Peripheral Devices (GPC, CVSS, SSS, and Programming Console) and access from BASIC Units, Personal Computer Units, Ethernet Units, and Temperature Controller Data Link Units, and through SYSMAC NET Link Systems, SYSMAC BUS/2 Systems, SYSMAC LINK Systems, Host Link Systems, etc., is disabled. 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 CPU Bus Units, the host link interface, and Peripheral Devices can also be disabled by turning ON the bits shown in the following table. These bit 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).

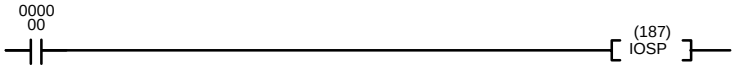
Bit(s)	Name	Function
A01500 through A01515	CPU Service Disable Bits	Turned ON to stop service to CPU Bus Units numbered #0 through #15, respectively.
A01703	Host Link Service Disable Bit	Turned ON to stop Host Link and NT Link System servicing.
A01704	Peripheral Service Disable Bit	Turned ON to stop service to Peripheral Devices.

Flags

There are no flags affected by this instruction.

Example

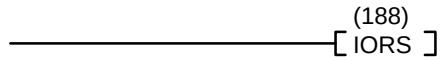
When CIO 000000 is ON in the following example, memory access is disabled from Peripheral Devices, through communications systems, and from other specified Units.



Address	Instruction	Operands
00000	LD	000000
00001	IOSP(187)	

5-36-2 ENABLE ACCESS: IORS(188)

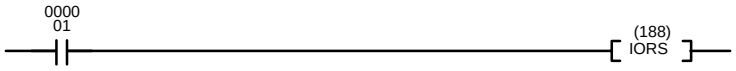
Ladder Symbol



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 Peripheral Devices is enabled.

Flags There are no flags affected by this instruction.

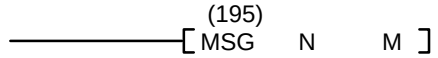
Example When CIO 000001 is ON in the following example, memory access is enabled.



Address	Instruction	Operands
00000	LD	000000
00001	IORS(188)	

5-36-3 DISPLAY MESSAGE: MSG(195)

Ladder Symbol



Variations

↑ MSG(195)

Operand Data Areas

N: Message number	CIO, G, A, T, C, #, DM, DR, IR
M: 1st message word	CIO, G, A, T, C, #, DM

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 CVSS screen or other Peripheral Device. The displayed message can be up to 32 characters long, i.e., each ASCII character code requires eight bits (two 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 Peripheral 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.

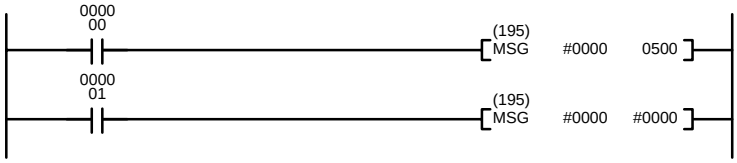
Note Refer to page 115 for general precautions on operand data areas.

Flags ER (A50003): N is not between 0 and 7.
Content of ∗DM word is not BCD when set for BCD.

Example

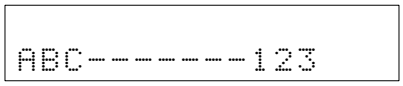
The following example shows the display that would be produced for the instruction and data given when CIO 000000 is ON. If CIO 000001 goes ON, message 0 will be cleared.

The display message number must be set to 0 in the Peripheral Device before executing the instruction.



Address	Instruction	Operands
00000	LD	000000
00001	MSG(195)	
		#0000
		0500
00002	LD	000001
00003	MSG(195)	
		#0000
		#0000

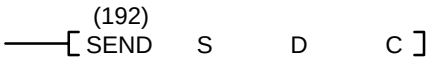
DM contents					ASCII equivalent	
0500	4	1	4	2	A	B
0501	4	3	4	4	C	D
0502	4	5	4	6	E	F
. . .	. . .	. . .	. . .	. . .	. . .	. . .
0513	3	1	3	2	1	2
0514	3	3	0	D	3	(end)
0515	3	4	3	5	5	6



The last character displayed is just before “OD” in the message data.

5-36-4 NETWORK SEND: SEND(192)

Ladder Symbol



Variations

↑ SEND(192)

Operand Data Areas

S: 1st source word	CIO, G, A, T, C, DM
D: 1st destination word	CIO, G, A, T, C, DM
C: 1st control word	CIO, G, A, T, C, DM

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 or SYSMAC 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 System or a SYSMAC LINK System.

SEND(192) only starts the transmission. Verify that the transmission has been completed with the Network Status Flags in A502.

Note The node number for SYSMAC LINK Units and SYSMAC NET Link Units corresponds to the unit number for the host link interface in the PC Setup.

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)[1]	Bits 08 to 11: ($0 to $F)[2] Bits 12 to 15: Set to 0.
C+2	Destination unit address[3]	Destination node number[4]
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)[5]	

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. 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 or SYSMAC 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 $FF to send to all nodes, $00 to send data 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. The response function is not used when the destination node number is set to $FF, broadcasting to all nodes in the network.
6. Transmissions cannot be sent to the PC executing the send.

SYSMAC LINK System

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* 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)[1]	Bits 08 to 11: ($0 to $F)[2] Bits 12 to 15: Set to 0.
C+2	Destination unit address[3]	Destination node number[4]
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)[5]	

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. 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 $FF to send to all nodes, $00 to send data 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. The response function is not used when the destination node number is set to $FF, broadcasting to all nodes in the network.
6. 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 or SYSMAC LINK Unit mounted.

Do not change the control data during a transmission (i.e., while the corresponding Port Enabled Flag in A502 is OFF).

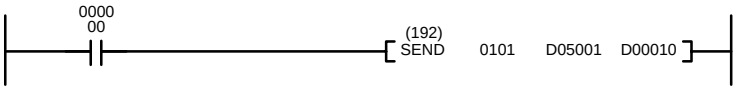
Note Refer to page 115 for general precautions on operand data areas.

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 000000 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

	15			0
D00010	0	0	0	5
D00011	0	0	0	1
D00012	0	3	0	0
D00013	0	2	0	7
D00014	0	0	6	4

	Node 3
CIO 0101	D05001
CIO 0102	D05002
CIO 0103	D05003
CIO 0104	D05004
CIO 0105	D05005

5-36-5 NETWORK RECEIVE: RECV(193)

Ladder Symbol

```
   (193)
——[ RECV   S    D    C ]
```

Variations

↑ RECV(193)

Operand Data Areas

S: 1st source word	CIO, G, A, T, C, DM
D: 1st destination word	CIO, G, A, T, C, DM
C: 1st control word	CIO, G, A, T, C, DM

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 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 System or a SYSMAC 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 The node number for SYSMAC LINK Units and SYSMAC NET Link Units corresponds to the unit number for the host link interface in the PC Setup.

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)[1]	Bits 08 to 11: ($0 to $F)[2] Bits 12 to 15: Set to 0.
C+2	Source unit address[3]	Source node number[4]
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)[5]	

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. 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 or SYSMAC 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 System

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* 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)[1]	Bits 08 to 11: ($0 to $F)[2] Bits 12 to 15: Set to 0.
C+2	Source unit address[3]	Source node number[4]
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)[5]	

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. 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.
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 or SYSMAC LINK Unit mounted.

Do not change the control data during a transmission (i.e., while the corresponding Port Enabled Flag in A502 is OFF).

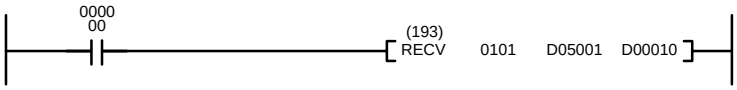
Note Refer to page 115 for general precautions on operand data areas.

Flags

ER (A50003): Content of ∗DM word is not BCD when set for BCD.

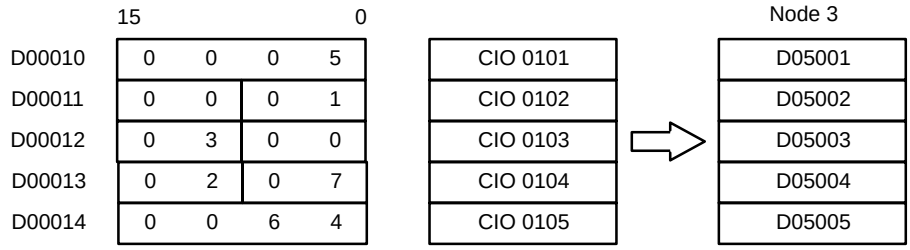
Example

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.



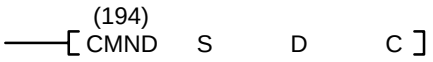
Address	Instruction	Operands
00000	LD	000000
00001	RECV(193)	
		0101
		D05001
		D00010

Control Words



5-36-6 DELIVER COMMAND: CMND(194)

Ladder Symbol



Variations

↑ CMND(194)

Operand Data Areas

S: 1st command word	CIO, G, A, T, C, DM
D: 1st response word	CIO, G, A, T, C, DM
C: 1st control word	CIO, G, A, T, C, DM

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 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.)	Only the command part is required. The header code, FCS, etc. are not required.

Note The node number for SYSMAC LINK Units and SYSMAC NET Link Units corresponds to the unit number for the host link interface in the PC Setup.

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 System or a SYSMAC 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)[1]	
C+1	Number of bytes to receive (0 to 1990, i.e., $0000 to $07C6)[1]	
C+2	Destination network address (0 to 127, i.e., $00 to $7F)[2]	Bits 08 to 11: ($0 to $F)[3] Bits 12 to 15: Set to 0.
C+3	Destination unit address[4]	Destination node number[5]
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)[6]	

Note

1. Maximum number of bytes that can be sent or received:

System	Max. number of bytes
SYSMAC NET Link	$07C6 (1990)
SYSMAC LINK	$021E (542)
SYSMAC BUS/2	$021E (542)

2. Set the destination network address to $00 when transmitting within the same network.
3. 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 or SYSMAC 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/type of transmission	Possible values
SYSMAC NET Link System	$01 to $7E (nodes 1 to 126)
SYSMAC LINK System	$01 to $3E (nodes 1 to 62)
Broadcast to all nodes in network	$FF
Transmit within the PC (to/from CPU Bus Units)	$00

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, 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.

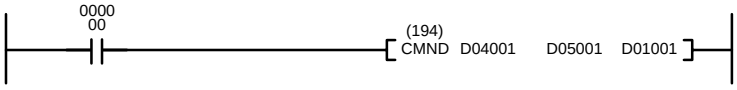
Note Refer to page 115 for general precautions on operand data areas.

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

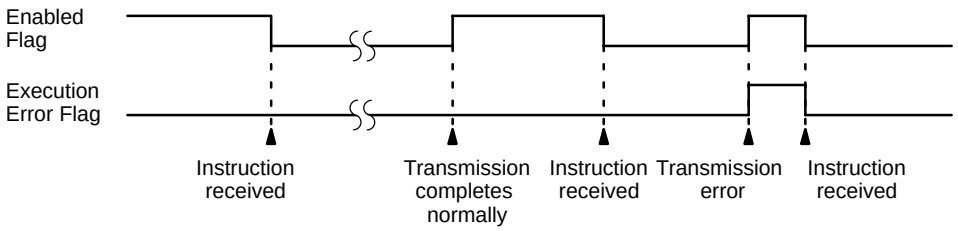
5-36-7 About SYSMAC NET Link/SYSMAC LINK Operations

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* or *SYSMAC 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 *SYSMAC NET Link System Manual* or *SYSMAC LINK System Manual* for details about error codes.

Timing



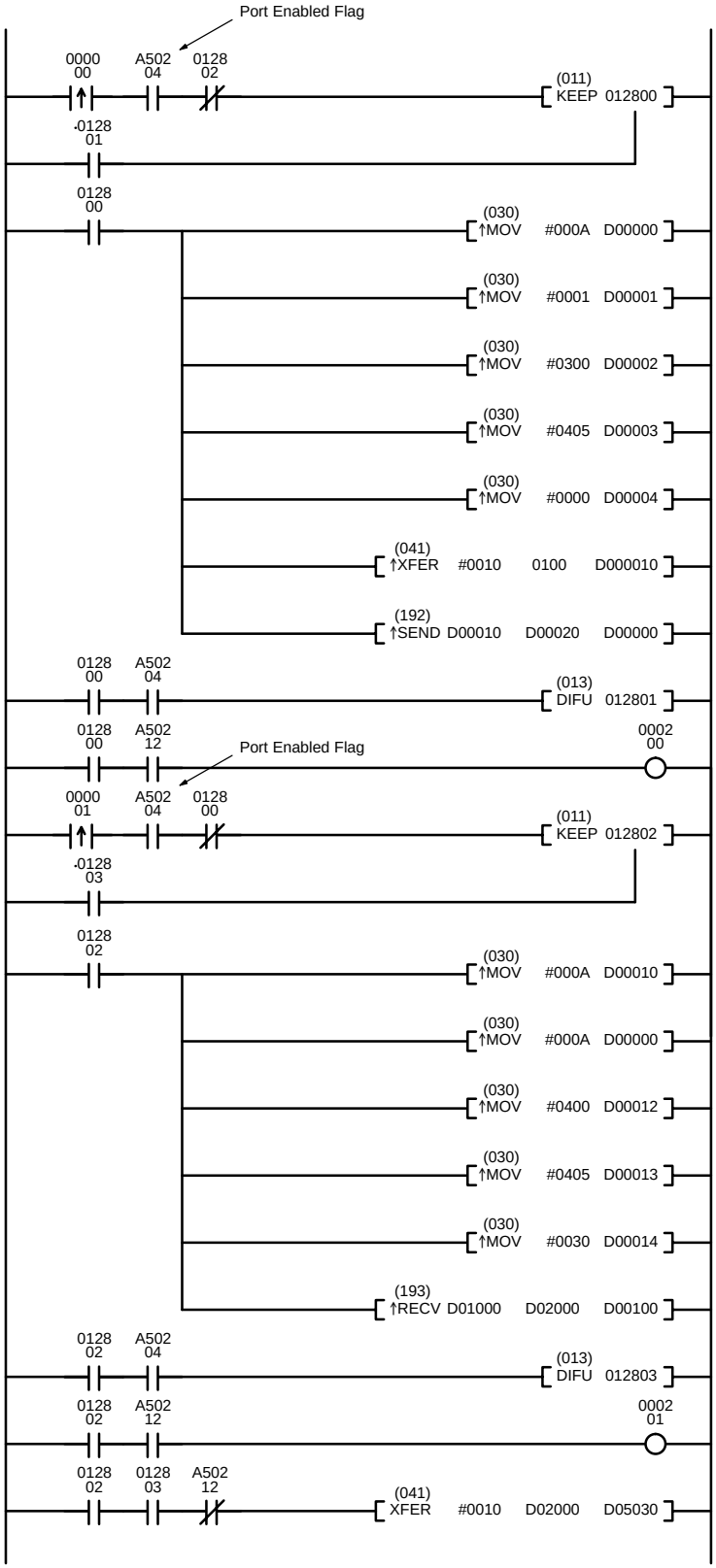
Data Processing for Network Instructions

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.

Instruction execution and peripheral device servicing are processed in parallel, thus data from network instructions might not be processed synchronously. To synchronize data processing, set the PC to synchronous execution or use the IOSP(187) and IORS(188) instructions.

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 RECV(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 RECV(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.

RECV(193) is executed.

When the Port 4 Enable Flag goes ON, CIO 012803 is turned ON to indicate completion of the RECV(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

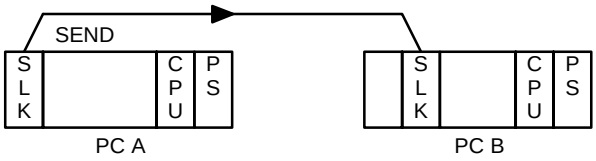
Address	Instruction	Operands
00020	AND	A50204
00021	AND NOT	012800
00022	LD	012803
00023	KEEP(011)	012802
00024	LD	012802
00025	↑MOV(030)	
		#000A
		D00010
00026	↑MOV(030)	
		#000A
		D00000
00027	↑MOV(030)	
		#0400
		D00012
00028	↑MOV(030)	
		#0405
		D00013
00029	↑MOV(030)	
		#0030
		D00014
00030	↑RECV(193)	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

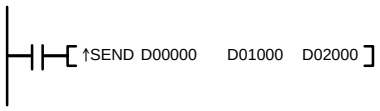
Programming Example: Synchronizing Data

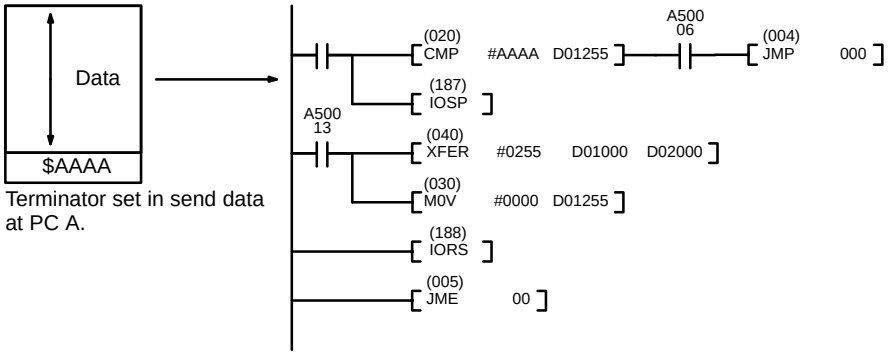
The following program shows how to synchronize data transmission during asynchronous operation using IOSP(187) and IORS(188).

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.







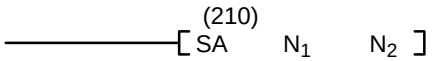
5-37 SFC Control Instructions

SFC Control Instructions are used to control step status in the SFC program or to output transition conditions from transition programs (TOUT(202) and TCNT(123)). Refer to the *CV-series PC Operation Manual: SFC* for details on SFC programming.

Note The CVM1 does not support SFC programming and must be programmed using ladder diagrams only.

5-37-1 ACTIVATE STEP: SA(210)

Ladder Symbol



Operand Data Areas

N_1: Step number	ST
N_2: Subchart entry step	ST or 9999

Variations

↑ SA(210)

Description

When the execution condition is OFF, SA(210) is not executed. When the execution condition is ON, SA(210) changes a designated step or subchart to execute status and starts execution of actions. A step activated by SA(210) will be executed as the last active step in the execution cycle for the first cycle after the step goes active.

To designate a step in a subchart, specify the step number of the entry step calling the subchart as the subchart entry step (N_2) and then designate the step number within the subchart.

To designate a subchart, designate the step number of the entry step calling the subchart as the step number, N_1 and designate the subchart entry step as 9999.

To designate a step not in a subchart, designate the step number and designate a subchart entry step of 9999.

Note SA(210) cannot be executed either from an interrupt program or for steps in interrupt programs.

Normal Steps

The specific results of executing SA(210) for steps in each step status are described below for normal steps inside or outside of subcharts.

Execute Status does not change, but status will not be transferred to the next step for one execution cycle after execution of SA(210).

Pause Changes the step status from pause to execute. The output status that was in effect at the time the status was changed from execute to pause will be continued, and execution will begin from the top of the step.

Halt Changes the step status from halt to execute. The output status that was in effect at the time the status was changed from execute to pause will be continued, and execution will begin from the top of the designated step.

Inactive Changes the step status from inactive to execute. Execution will begin from the top action of the designated step. The step timer will begin.

Subchart Dummy Steps

The specific results of executing SA(210) for steps in each step status are described below for dummy steps controlling subcharts.

Execute SA(210) does not change the subchart dummy step itself. All the steps in a subchart that are active when SA(210) is executed go to execute status. The output status that was in effect at the time the status was changed from execute to pause or halt will be continued, and execution will begin from the action at the top of the step.

Pause Changes the subchart dummy step and the steps in the subchart that are in pause status to execute status. The output status that was in effect at the time the status was changed from execute to pause will be continued, and execution will begin from the top.

Halt Changes the subchart dummy step, and the steps in the subchart that are in halt status, to execute status. The output status that was in effect at the time the status was changed from execute to halt will be continued, and execution will begin from the top.

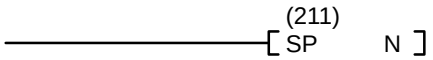
Inactive Changes the subchart dummy step, and the steps in the subchart that are inactive, to execute status. SA(210) also places the designated subchart entry step in execute status, and starts operation from the top action. The step timer will begin.

Flags

ER (A50003): Turns ON when the step designated by N_1 is undefined.
Turns ON when the subchart designated by N_2 is undefined.
Turns ON when the subchart designated by N_2 is not being executed.

5-37-2 PAUSE STEP: SP(211)

Ladder Symbol



Operand Data Area

N: Step number ST

Variations

↑ SP(211)

Description When the execution condition is OFF, SP(211) is not executed. When the execution condition is ON, SP(211) changes the status of a step or subchart from execute to pause. To designate a subchart, designate the step number of the dummy step calling the subchart as the step number, N

In pause status, the execution of actions with N, P, L, and D action qualifiers is stopped, but the present values of TIM and TIMH instructions that have been started continue to operate. The present values of other timers and counters, as well as other outputs, are maintained. Step timers continue to operate, so be careful when using L and D action qualifiers.

Note SP(211) cannot be executed for steps in an interrupt program.

Normal Steps Status

The specific results of executing SP(211) for steps in each step status are described below for normal steps inside or outside of subcharts.

Execute Changes step status from execute to pause. Execution of actions will be paused, but output status will be maintained and the step timer will continue. When SP(211) is executed on a step in a subchart, execution of the actions in that step will be paused beginning with the next execution cycle.

Pause, Halt, Inactive Step status does not change.

Subchart Dummy Steps

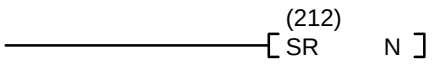
The specific results of executing SP(211) for steps in each step status are described below for dummy steps controlling subcharts.

Execute Changes the status of the subchart dummy step from execute to pause. All active steps in the designated subchart (including those in halt status) will be placed in pause status. Execution of actions will be paused, but output status will be maintained and the step timer will continue. If SP(211) is executed for a subchart from within the same subchart, the actions within the only step containing SP(211) will be paused beginning with the next execution cycle.

Pause, Halt, Inactive Step status does not change.

5-37-3 RESTART STEP: SR(212)

Ladder Symbol



Operand Data Area

N: Step number ST

Variations

↑ SR(212)

Description When the execution condition is OFF, SR(212) is not executed. When the execution condition is ON, SR(212) changes the status of a step or subchart from pause to execute.

Note SR(212) cannot be executed for steps in an interrupt program.

Normal Steps Status

The specific results of executing SR(212) for steps in each step status are described below for normal steps inside or outside of subcharts.

Execute Step status does not change.

Pause Changes the step status from pause to execute. Execution will be resumed from the beginning of the subchart, and output status will continue with the status that existed at the time the step was changed to pause status.

Halt, Inactive Step status does not change.

Subchart Dummy Steps

The specific results of executing SR(212) for steps in each step status are described below for dummy steps controlling subcharts.

Execute Step status does not change.

Pause Changes the status of the subchart dummy step and all steps in the subchart that are in pause status from pause to execute. Execution will be resumed from the beginning of the subchart, and output status will continue with the status that existed at the time the dummy step was changed to pause status.

Halt, Inactive Step status does not change.

5-37-4 END STEP: SF(213)

Ladder Symbol	Operand Data Area
(213) SF N	**N: Step number** ST

Variations

↑ SF(213)

Description When the execution condition is OFF, SF(213) is not executed. When the execution condition is ON, SF(213) changes the status of a step or subchart from execute or pause to halt.

In halt status, the execution of actions with N, P, L, and D action qualifiers is stopped, but the present values of TIM and TIMH instructions that have been started continues to operate. The present values of other timers and counters, as well as other outputs, are maintained. Step timers continue to operate, so be careful when using L and D action qualifiers.

Note SF(213) cannot be executed for steps in an interrupt program.

Normal Steps Status

The specific results of executing SF(213) for steps in each step status are described below for normal steps inside or outside of subcharts.

Execute Changes the step status from execute to halt. Execution of actions will be stopped, but output status will be maintained and the step timer will continue counting. When SF(213) is executed on a step in a subchart, execution of the actions in that step will be stopped beginning with the next execution cycle.

Pause Changes the step status from pause to halt.

Halt, Inactive Step status does not change.

Subchart Dummy Steps

The specific results of executing SF(213) for steps in each step status are described below for dummy steps controlling subcharts.

Execute Changes the status of the subchart dummy step from execute to halt. All active steps in the designated subchart (including those in pause status) will be changed to halt status. Execution of actions will be stopped, but output status will be maintained and the step timer will continue counting. If SF(213) is executed for a subchart from within the same subchart, the actions within only the step containing SF(213) will be stopped beginning with the next execution cycle.

Pause Changes the status of the subchart dummy step from pause to halt. All paused steps in the designated subchart will be changed to halt status. Execution of actions will be stopped, but output status will be maintained and step timers will continue.

Halt, Inactive Step status does not change.

5-37-5 DEACTIVATE STEP: SE(214)

Ladder Symbol	Operand Data Area	
(214) SE N	**N: Step number**	ST

Variations

↑ SE(214)

Description

When the execution condition is OFF, SE(214) is not executed. When the execution condition is ON, SE(214) changes the status of a step or subchart from active (execute, pause or halt) to inactive status.

SE(214) does not necessarily result in transfer of active status from one step to another. When SE(214) is executed, it simply makes the designated step or subchart inactive.

Note SE(214) cannot be executed from any interrupt program other than a power-on interrupt program. In addition, SE(214) cannot be executed for steps in any interrupt program (including a power-on interrupt program).

Normal Steps Status

The specific results of executing SE(214) for steps in each step status are described below for normal steps inside or outside of subcharts.

Execute Changes the step status from execute to inactive. Execution of actions will be stopped and the actions will be reset. Actions with the optional hold action qualifier, however, will be stopped but not reset. In addition, execution will be continued for actions with S-group AQs. The step timer will continue. If SE(214) is executed on steps in a subchart, step execution will be stopped directly after execution of the instruction.

Pause Changes the step status from pause to inactive. Execution of actions will be stopped and the actions will be reset. Actions with the optional hold action qualifier, however, will be stopped but not reset. In addition, execution will be continued for actions with S-group AQs. The step timer will continue.

Halt Changes the step status from halt to inactive. The step's actions will be reset. Actions with the optional hold action qualifier, however, will not be reset. In addition, execution will be continued for actions with S-group AQs. The step timer will continue.

Inactive Step status does not change.

Subchart Dummy Steps

The specific results of executing SE(214) for steps in each step status are described below for dummy steps controlling subcharts.

Execute Changes status of subchart dummy step from execute to inactive, and makes inactive all of the steps in the subchart that were active at the time the instruction was executed. Actions will be stopped and reset. Actions with the optional hold action qualifier, however, will be stopped but not reset. In addition, execution will be continued for actions with S-group AQs. Step timers will continue. If SE(214) is executed on steps in a subchart, step execution will be stopped directly after execution of the instruction.

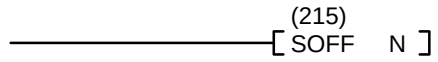
Pause Changes status of subchart dummy step from pause to inactive, and makes inactive all of the steps in the subchart that were active at the time the instruction was executed. Actions will be stopped and reset. Actions with the optional hold action qualifier, however, will be stopped but not reset. In addition, execution will be continued for actions with S-group AQs. Step timers will continue.

Halt Changes status of subchart dummy step from halt to inactive, and makes inactive all of the steps in the subchart that were active at the time the instruction was executed. Actions will be reset. Actions with the optional hold action qualifier, however, will not be reset. In addition, execution will be continued for actions with S-group AQs. Step timers will continue.

Inactive Step status does not change.

5-37-6 RESET STEP: SOFF(215)

Ladder Symbol



Operand Data Area

N: Step number ST

Variations

↑ SOFF(215)

Description When the execution condition is OFF, SOFF(215) is not executed. When the execution condition is ON, SOFF(215) places a designated step or subchart inactive and resets all of the actions in the step.

SOFF(215) does not necessarily result in transfer of active status from one step to another. When SOFF(215) is executed, it simply makes the designated step or subchart inactive.

Note SOFF(215) cannot be executed from any interrupt program other than a power-on interrupt program. In addition, SE(214) cannot be executed for steps in any interrupt program (including a power-on interrupt program).

Normal Steps Status

The specific results of executing SOFF(215) for steps in each step status are described below for normal steps inside or outside of subcharts.

Execute Changes the step status from execute to inactive. Execution of all actions (including actions with the optional hold action qualifier and actions with S-group AQs) will be stopped and the actions will be reset. The step timer will also be stopped. If SOFF(215) is executed on steps in a subchart, step execution will be stopped directly after execution of the instruction.

Pause Changes the step status from pause to inactive. Execution of all actions (including actions with the optional hold action qualifier and actions with S-group AQs) will be stopped and the actions will be reset. The step timer will also be stopped.

Halt Changes the step status from halt to inactive. All actions (including actions with the optional hold action qualifier and actions with S-group AQs) will be stopped and reset. The step timer will be stopped.

Inactive Execution of actions with S-group AQs will be stopped and the actions will be reset. Actions with the optional hold action qualifier will be reset. The step timer will also be stopped.

Subchart Dummy Steps

The specific results of executing SOFF(215) for steps in each step status are described below for dummy steps controlling subcharts.

Execute Changes the status of the subchart dummy step from execute to inactive and places all of the steps in the subchart inactive. Execution of all actions (including actions with the optional hold action qualifier and actions with S-group AQs) will be stopped and the actions will be reset. Step timers will also be stopped. If SOFF(215) is executed on steps in a subchart, step execution will be stopped directly after execution of the instruction.

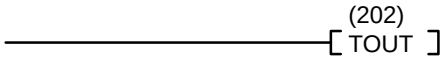
Pause Changes the status of the subchart dummy step from pause to inactive and places all of the steps in the subchart inactive. Execution of all actions (including actions with the optional hold action qualifier and actions with S-group AQs) will be stopped and the actions will be reset. Step timers will also be stopped.

Halt Changes the status of the subchart dummy step from halt to inactive and places all of the steps in the subchart inactive. Execution of all actions (including actions with the optional hold action qualifier and actions with S-group AQs) will be stopped and the actions will be reset. The step timer will also be stopped.

Inactive Resets all actions of steps in the subchart. Execution of Actions with the optional hold action qualifier and actions with S-group AQs will also be stopped and the actions will be reset. Step timers will also be stopped.

5-37-7 TRANSITION OUTPUT: TOUT(202)

Ladder Symbol



Description When the execution condition is OFF, TOUT(202) is not executed. When the execution condition is ON, TOUT(202) outputs the result of a transition condition operation to the Transition Flag to control the transition condition. If TOUT(202) is executed with an ON execution condition, the Transition Flag will be turned ON. If TOUT(202) is executed with an OFF execution condition, the Transition Flag will be turned OFF.

TOUT(202) cannot be used outside of a transition program and writing to the Transition Flag area cannot be achieved with any instructions other than TOUT(202) and TCNT(123) (see next section).

If either TOUT(202) and/or TCNT(123) is used more than once in one transition program, the status of the Transition Flag will be controlled by the last TOUT(202) or TCNT(123) in the transition program.

The ON/OFF status of a Transition Flag determined by TOUT(202) is held until the next execution of TOUT(202), even if the step status changes.

Note

1. A50015 (First Cycle Flag) cannot be used within a transition program.
2. The following three conditions must be met in order for active status to be transferred from one step to another:

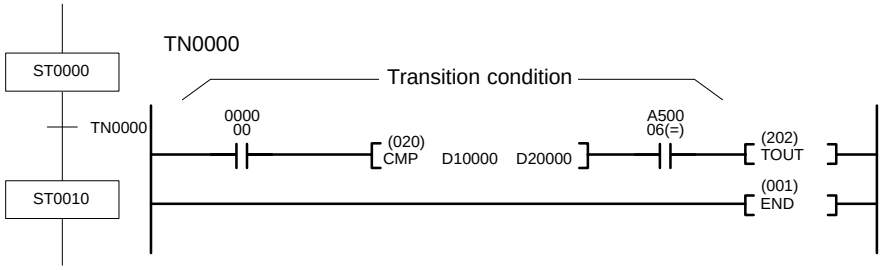
   The transition condition (i.e., the Transition Flag) must be ON.

   The step(s) before the transition must be active (execute or halt, but not pause).

   The step(s) after the transition must be inactive.

Example If in the following example CIO 000000 is ON and D10000 and D20000 have the same content, the Equals Flag, A50006, will turn ON, and TOUT(202) will turn ON the Transition Flag. If CIO 00000 is OFF or D10000 and D20000 have different contents, TOUT(202) will turn OFF the Transition Flag.

If the Transition Flag turns ON, ST0000 is active (but not in pause status), and ST0010 is inactive, then all of the transition conditions are met and active status will be transferred from ST0000 to ST0010.



Address	Instruction	Operands
00000	LD	000000
00001	CMP(020)	
		D10000
		D20000
00002	AND	A50006
00003	TOUT(202)	
00004	END(001)	

5-37-8 TRANSITION COUNTER: TCNT(123)

Ladder Symbol

(123)
TCNT N S

Operand Data Areas

N: Counter number C

S: No. of executions CIO, G, A, T, C, #, DM, DR, IR

Description

When the execution condition is OFF, TCNT(123) is not executed. When the execution condition is ON, TCNT(123) turns a corresponding Transition Flag ON or OFF according to the number of times the transition program is executed.

The counter is started the first time TCNT(123) is executed with an ON execution condition after it is reset and the present value is incremented starting with the second execution. Therefore, the actual number of execution will be S + 1.

When the present value reaches the value set for S, the Transition Flag and the Counter Flag will turn ON.

The status of the transition counter is maintained even when step status is changed, so the transition counter should normally be reset using CNR(236) in the previous step.

If either TOUT(202) and/or TCNT(123) is used more than once in one transition program, the status of the Transition Flag will be controlled by the last TOUT(202) or TCNT(123) in the transition program.

TCNT(123) cannot be used outside of a transition program and writing to the Transition Flag area cannot be achieved with any instructions other than TOUT(202) (see previous section) and TCNT(123).

Note

1. A50015 (First Cycle Flag) cannot be used within a transition program.
2. The following three conditions must be met in order for active status to be transferred from one step to another:

   The transition condition (i.e., the Transition Flag) must be ON.

   The step(s) before the transition must be active (execute or halt, but not pause).

   The step(s) after the transition must be inactive.

Precautions

S must be in BCD.

The same counter numbers are used by CNT, CNTR(012), and TCNT(123). Do not use the same number to define more than one counter, regardless of the counter instruction used.

The transition counter counts each time the transition program is executed. When using a parallel join, be particularly careful of the number that is set, because the transition program will be executed each time the steps just above the transition are executed.

Flags

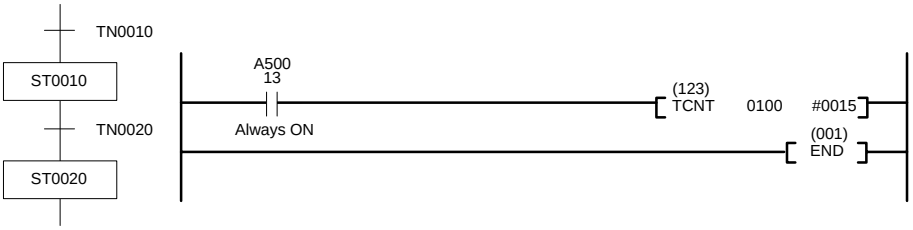
ER (A50003): ON when the content of N is not a counter number.

ON when the content of S is not BCD data.

On when the content of ∗DM or ∗EM word is not BCD.

Example

When the program for TN0020 is executed for the first time, counter C0100 will be started. After that, the transition program will be counted each time it is executed. As a result, the Completion Flag for C0100 will turn ON when the program has been executed 1 + 15 times, turning ON the Transition Flag for TN0020. From the 17th execution onwards, C0100 and the Transition Flag will remain ON until reset in a following execution cycle.

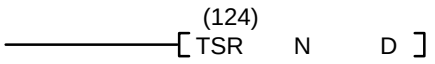


Address	Instruction	Operands
00000	LD	A50013
00001	TCNT(123)	0100
		#0015
00002	END(001)	

Note Here, CNR(236) would be used in the ST0010 action block to reset counter C0100 with A50015 (First Cycle Flag).

5-37-9 READ STEP TIMER: TSR(124)

Ladder Symbol



Variations

↑ TSR(124)

Operand Data Areas

N: Step number	CIO, G, A, T, C, #, DM, DR, IR
D: Destination	CIO, G, A, T, C, DM, DR, IR

Description

When the execution condition is OFF, TSR(124) is not executed. When the execution condition is ON, TSR(124) reads the present value (PV) of the step timer for the specified step and stores it as a binary value in the word designated in D. When executed with an OFF execution condition, TSR(124) does nothing.

Step Timers

Step timers time SFC steps and are used for operations such as measuring AQ timing when the AQ uses a set value to control action execution. Each step has its own step timer. Step timers are incremental timers that are reset and begin counting when the step becomes active, and which retain the present value when the step becomes inactive. If, however, the actions in the step continue to be executed (e.g., for S-group AQs), the step timer continues operating until all execution has been completed.

The PC can be set so that step timers operate in increments of either 0.1 second or 1 second. The increment is set in the PC system settings. The default setting is 0.1 second. Step timers can count up to 6553.5 s (when the unit is 0.1 s) or 65,535 s (when the unit is 1 s). The step timer retains the maximum value if the present value goes beyond the timeable range.

Step timer values are stored and handled as binary data.

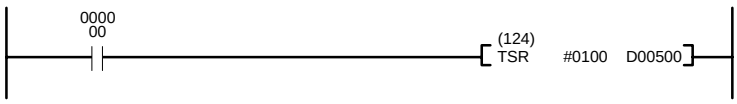
Flags

ER (A50003): ON when the N is set outside of the range.

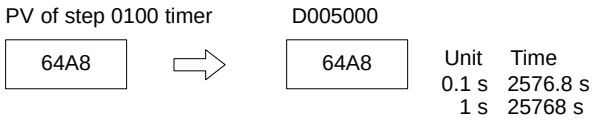
On when the content of ∗DM or ∗EM word is not BCD.

Example

When CIO 000000 is ON in the following example, the present value of the step timer for step ST0100 will be output to D00500 in binary data. This is unrelated to the active/inactive status of the step.

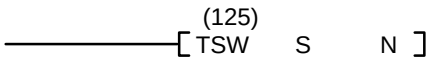


Address	Instruction	Operands
00000	LD	000000
00001	TSR(124)	
		#0100
		D00500



5-37-10 WRITE STEP TIMER: TSW(125)

Ladder Symbol



Variations

↑ TSW(125)

Operand Data Areas

S: Source	CIO, G, A, T, C, DM, DR, IR
N: Step number	CIO, G, A, T, C, #, DM, DR, IR

Description

When the execution condition is OFF, TSW(125) is not executed. When the execution condition is ON, TSW(125) changes the present value (PV) of the step timer. If TSW(125) is executed with an ON execution condition, the contents of word S is written as the present value for the step timer for step N. When executed with an OFF execution condition, TSW(125) does nothing.

Refer to the previous instruction for details on set timers.

Precautions

The contents of S must be set with binary data.

Step timers are used to control the execution timing of actions with certain AQ. Be careful when making changes with TSW(125).

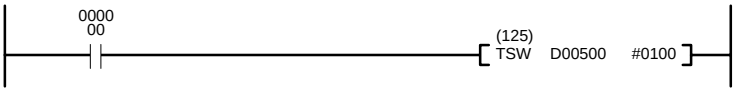
Flags

ER (A50003): ON when the N is set outside of the range.

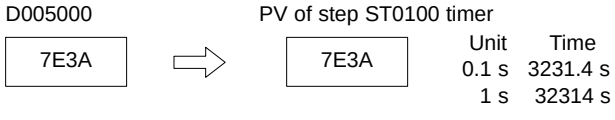
ON when the content of ∗DM or ∗EM word is not BCD.

Example

When CIO 000000 is ON in the following example, the contents of D00500 ($7E3A) will be written as the present value for the step timer of step ST0100. This is unrelated to the active/inactive status of the step.

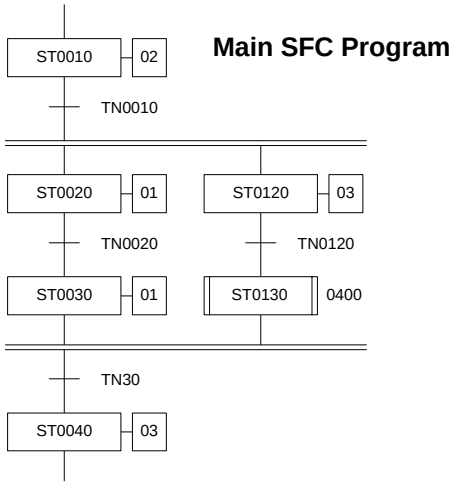


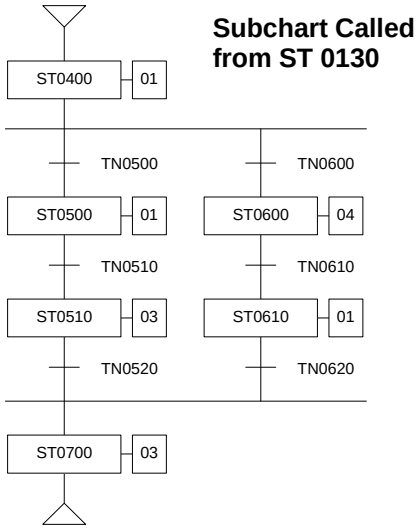
Address	Instruction	Operands
00000	LD	000000
00001	TSW(125)	
		D00500
		#0100



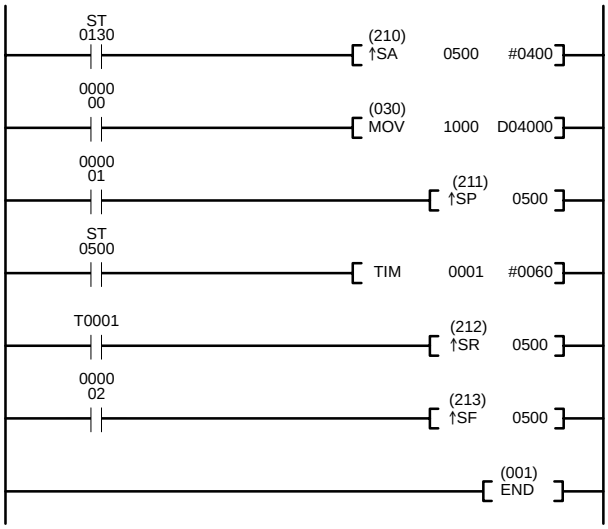
5-37-11 SFC Control Program Example

The operation of the SFC control instructions used in the following programming example is described following the program.

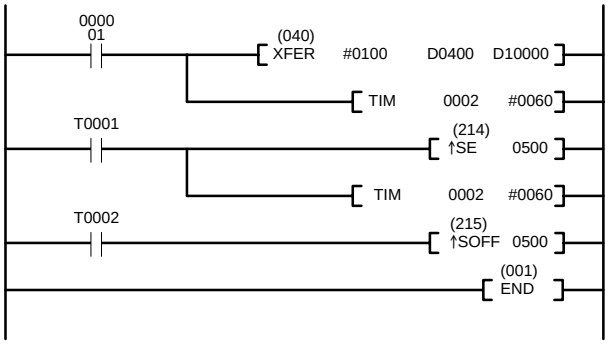




Action Program for AC0020 (ST0020)



Action Program for AC0500 (ST0500)



Action Block for ST0020

AQ	SV	Action	FV
N	********	AC0020	********

Action Block for ST0030

AQ	SV	Action	FV
N	********	AC0030	********

Action Block for ST0500

AQ	SV	Action	FV
S	********	AC0050	********

SA(210) in AC0020 When subchart ST0400 (i.e., the subchart with step ST0400) is active, SA(210) changes the status of ST0500 in the subchart to execute status. (Subchart ST0400 is executed while ST0130 is active.)

SP(211) in AC0020 SP(211) changes the status of ST0500 to pause.

SR(212) in AC0020 SR(212) changes the status of ST0500 back from pause to execute.

SF(213) in AC0020 SF(213) changes the status of ST0500 to halt. Execution of action AC0500 will continue because it has an “S” AQ.

SE(214) in AC0500 SE(214) changes the status of ST0500 to inactive. Execution of action AC0500 will continue because it has an “S” AQ.

SOFF(215) in AC0500 SOFF(215) changes the status of ST0500 to inactive. The action in the step is stopped and reset even though is has an S-group AQ.

5-38 Block Programming Instructions

Block programming can be used with version-2 CVM1 CPUs to program operations that are difficult to program with ladder diagrams, such as certain data computations. Effective block programming can be use 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.

5-38-1 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.

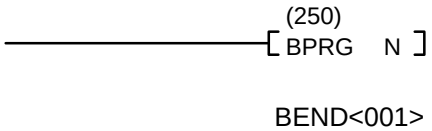
Note The operation of the above instructions will not be dependable is they are not used in combination as described or if they are used in combination with other instructions.

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.

5-38-2 BLOCK PROGRAM BEGIN/END: BPRG(250) / BEND<001> (CVM1 V2)

Ladder Symbol



BEND<001>

Operand Data Area

N: Block program number # (00 to 99)

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.

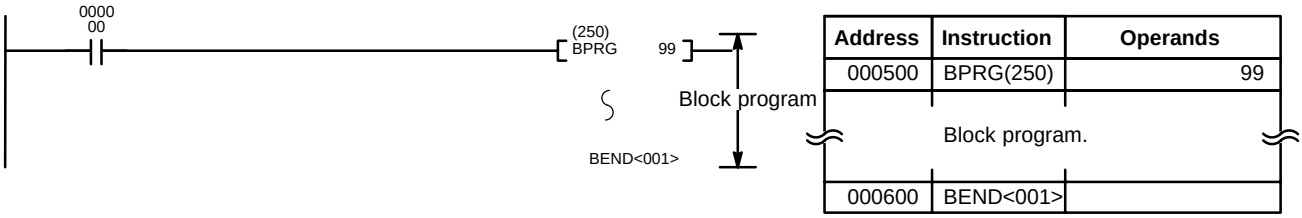
Precautions

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.



5-38-3 Branching–IF<002>, ELSE<003>, and IEND<004> (CVM1 V2)

Ladder Symbol		Operand Data Area	
IF<002>	B	**B: Bit**	CIO, G, A, T, C
IF<002>			
IF<002> NOT	B		
ELSE<003>			
IEND<004>			

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.

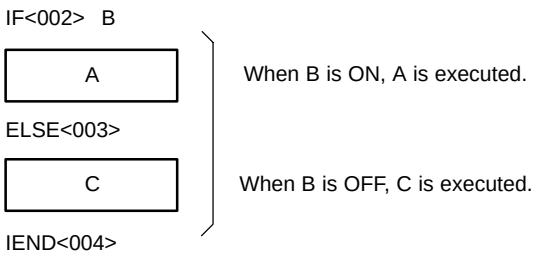
If the IF condition is NO, execution will jump to ELSE<003> or to IEND<004>, whichever appears first after the IF<002> or IF<002> NOT.

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

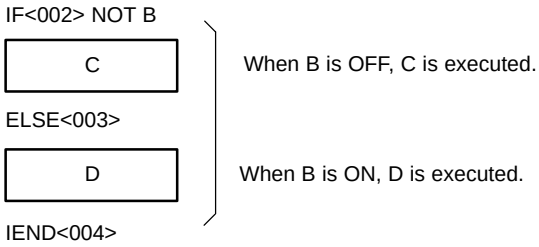
IF<002> with an Operand

IF<002> to ELSE to IEND



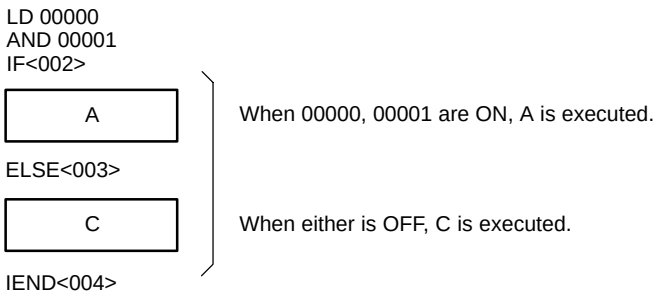
IF<002> NOT with an Operand

IF<002> NOT to ELSE to IEND



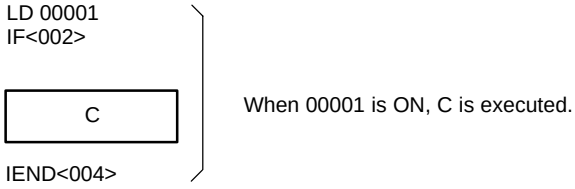
IF<002> without an Operand

IF<002> to ELSE to IEND



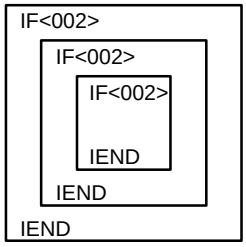
IF<002> without ELSE

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

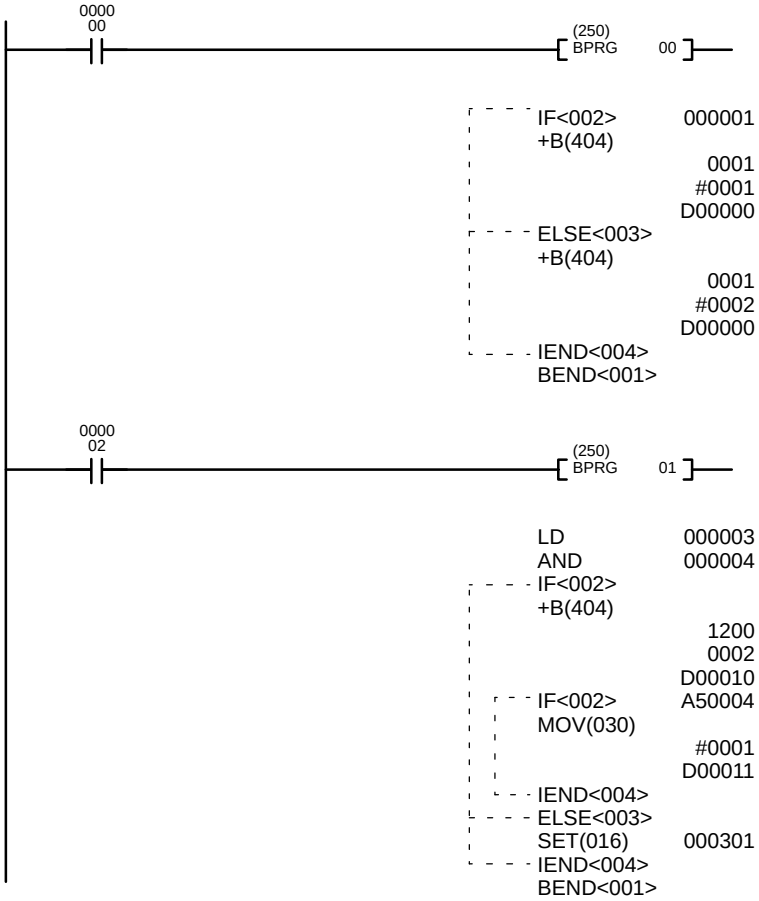
No flags are affected by these instructions.

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
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>	

5-38-4 ONE CYCLE AND WAIT: WAIT<005> (CVM1 V2)

Ladder Symbol		Operand Data Area	
		B: Bit	CIO, G, A, T, C
WAIT<005>			
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 of 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

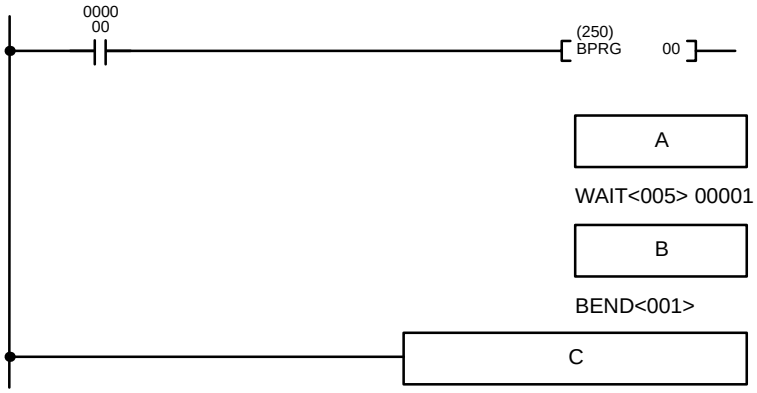
WAIT<005> NOT cannot be used without an operand bit.

If programs are edited online from a Peripheral 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.

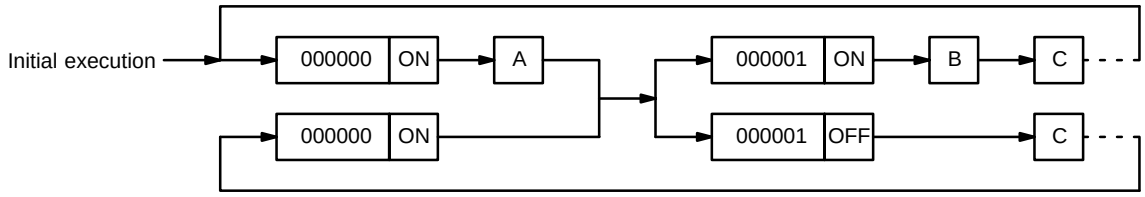
Execution Flow Examples

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.

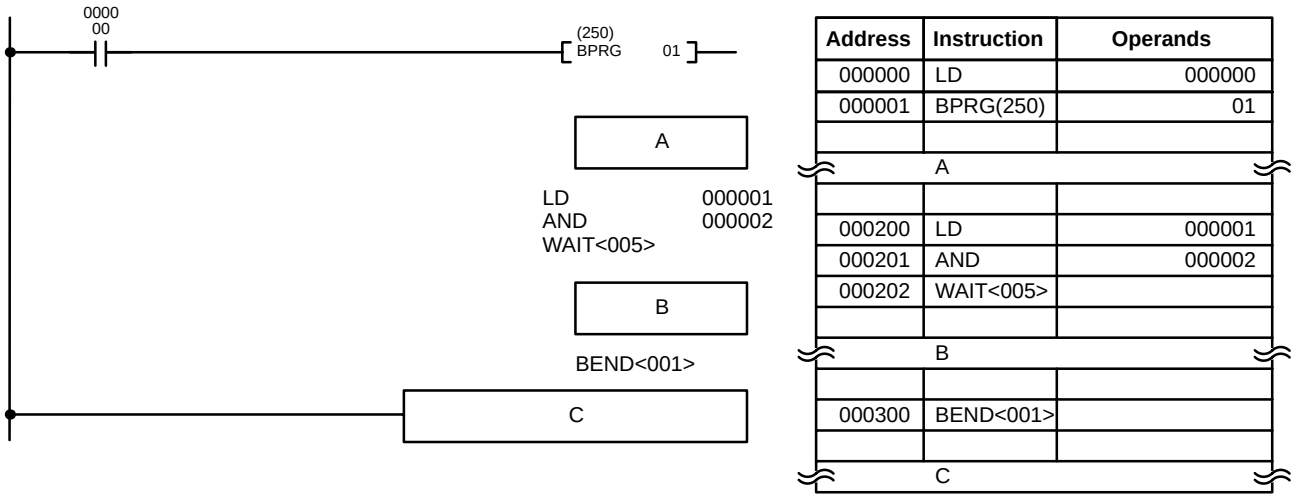


Address	Instruction	Operands
000000	LD	000000
000001	BPRG(250)	00
	A	
000100	WAIT<005>	000001
	B	
000200	BEND<001>	
	C	

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	

5-38-5 CONDITIONAL BLOCK EXIT: EXIT<006> (CVM1 V2)

Ladder Symbol

EXIT<006>
EXIT<006> B
EXIT<006> NOT B

Operand Data Area

B: Bit	CIO, G, A, T, C

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>.

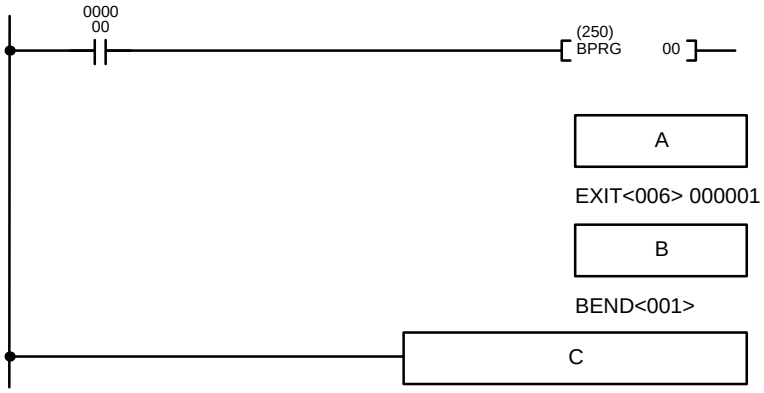
Precautions

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.



Address	Instruction	Operands
000000	LD	000000
000001	BPRG(250)	00
	A	
000100	EXIT<006>	000001
	B	
000200	BEND<001>	
	C	

5-38-6 Loop Control–LOOP<009>/LEND<010> (CVM1 V2)

Ladder Symbol		Operand Data Area	
LOOP<009>		**B: Bit**	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>

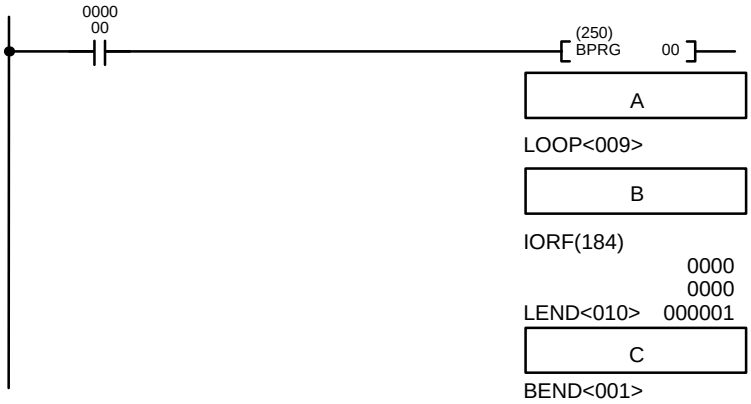
:

:

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>	

5-38-7 BLOCK PROGRAM PAUSE/RESTART : BPPS<011>/BPRS<012> (CVM1 V2)

Ladder Symbol		Operand Data Area
BPPS<011>	N	**N: Block program number** # (00 to 99)
BPRS<012>	N	

Description

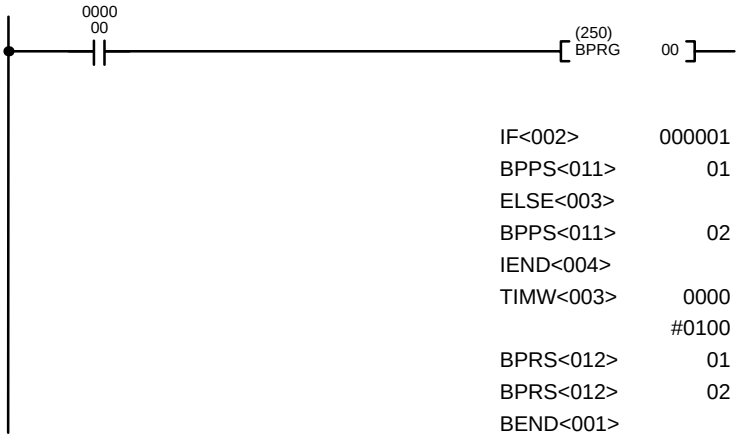
BPPS<011> is used inside one block program to suspend the execution of another block program. BPRS<012> restarts the specified block program. These instructions are effective whenever executed, i.e., they do not rely on operand bit status or execution condition.

Precautions

Even if the execution of a block program is suspended, PVs of all timer and high-speed timer with timer number T0000 through T0127 for the CVM1-CPU01-EV2 and T0000 through T0255 for the CVM1-CPU11-EV2 or CVM1-CPU21-EV2 defined by TIMW<013> or TMHW<015> inside 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
000008	BPRS<012>	01
000009	BPRS<012>	02
000010	BEND<001>	

5-38-8 HIGH-SPEED TIMER/TIMER WAIT: TIMW<013>/TMHW<015> (CVM1 V2)

Ladder Symbol		Operand Data Areas	
TIMW<013>	N SV	**N: Timer number**	T
		SV: 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.

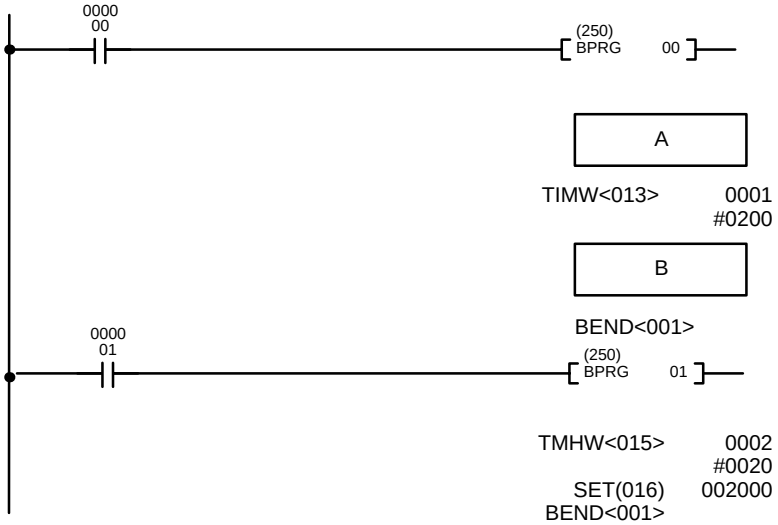
Precautions

Each timer number can be used as the definer in only one timer instruction, including those used in normal ladder-diagram timers.

If cycle time is greater than 10 ms, TC 0000 through TC 0255 must be used for TMHW<015> to ensure accuracy.

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.)

5-38-9 COUNTER WAIT: CNTW<014> (CVM1 V2)

Ladder Symbol		Operand Data Areas	
CNTW<014>	N	**N: Counter number**	C
	SV	**SV: Set value**	CIO, G, A, T, C, #, DM, DR, IR
	I	**I: Count input**	CIO, G, A, T, C

Description

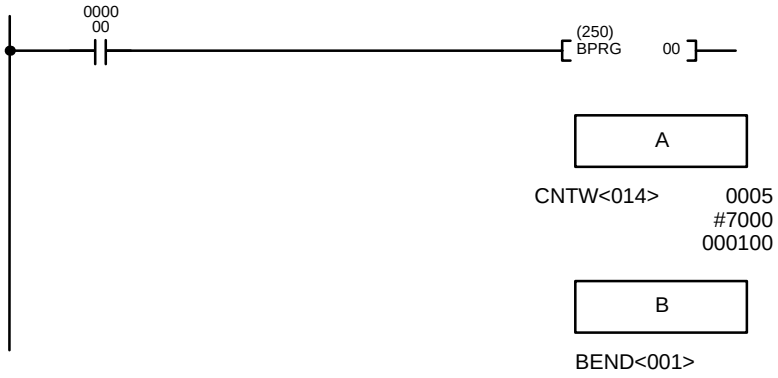
CNTW<014> allows you to create a ‘count’ lag (SV) between execution of the program part preceding the CNTW<014> (i.e., between BPRG(250) and CNTW<014>) and the part following it (i.e., between CNTW<014> and BEND<001>). The first part will be executed the first time the block program is entered. When 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.



Address	Instruction	Operands
000000	LD	000000
000001	BPRG(250)	00
	A	
000200	CNTW<014>	0005
		# 7000
		000100
	B	
000300	BEND<001>	

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. These manuals are listed at the end of *Section 1 Introduction*.

6-1 PC Operation

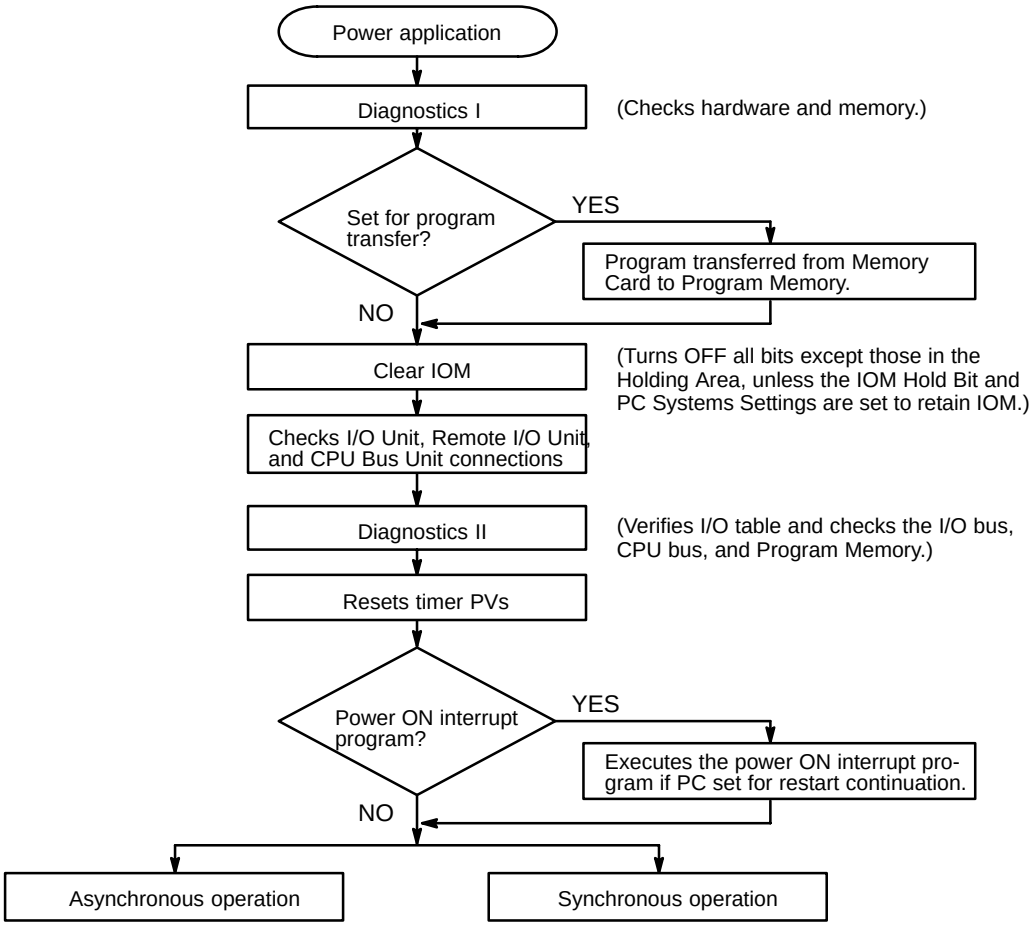
This section details basic CPU operation of CV-series PCs. The CV-series PCs can process instruction execution and I/O refreshing independently of peripheral servicing. Independent parallel processing is called asynchronous operation, and synchronized processing is called synchronous operation. Select asynchronous or synchronous operation in the PC Setup.

6-1-1 Initialization

The flow of initialization on power-up is as follows:

1, 2, 3...

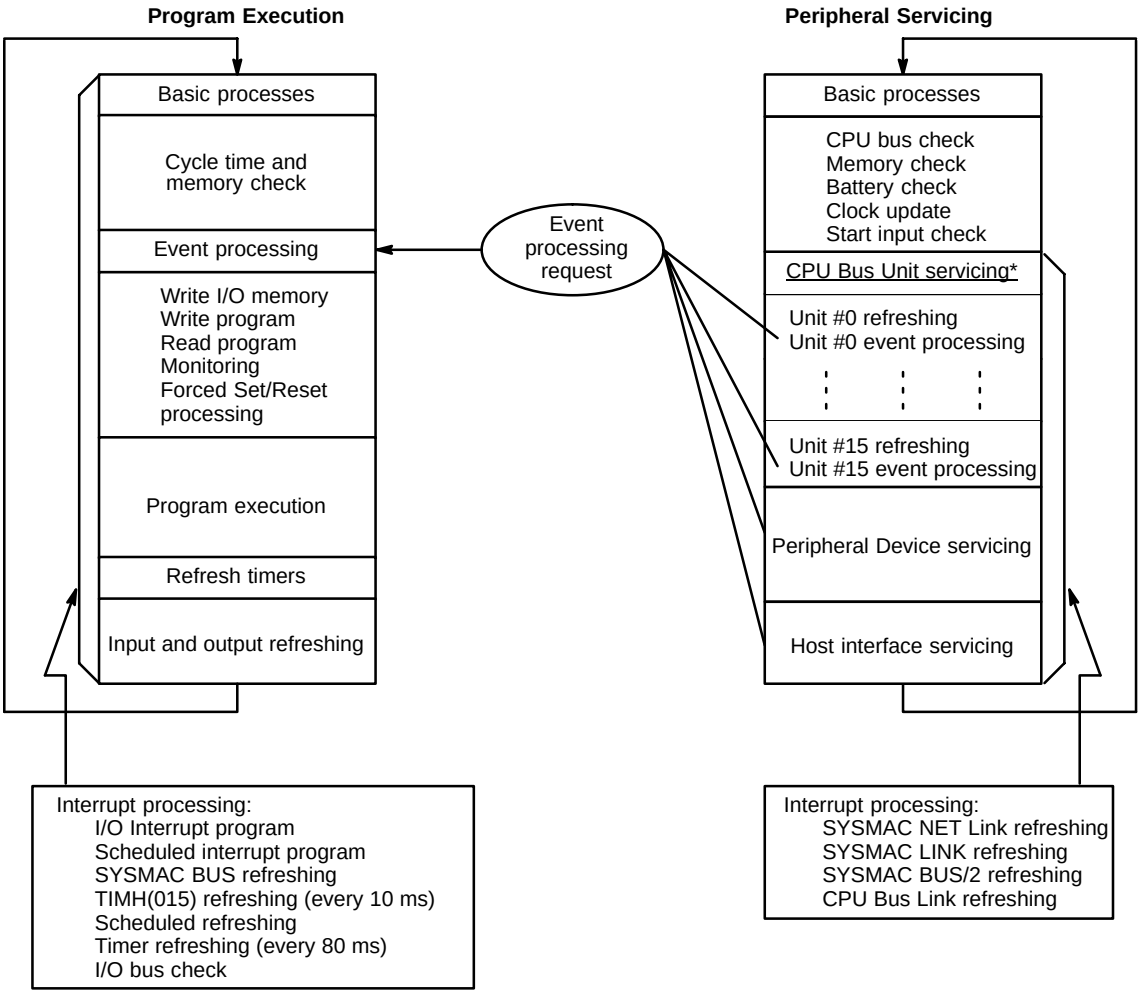
1. When power is turned on to the CPU, the first diagnostic program is run to check hardware and memory.
2. If the system is set for transfer of the program and/or Extended PC Setup, the data is transferred from the Memory Card.
3. All bits in memory except Holding Bits and bits set to be held in the PC Setup are turned OFF.
4. All I/O Units, Remote I/O Units, and CPU Bus Unit connections are checked.
5. The section diagnostic program is run to check the I/O table, I/O bus, CPU bus, and program memory.
6. All timer PVs are reset.
7. If the system is set to execute a power ON interrupt, an interrupt is generated and the power ON interrupt program is executed.
8. Synchronous or asynchronous operation is entered as specified in the PC Setup.



6-1-2 Asynchronous Operation

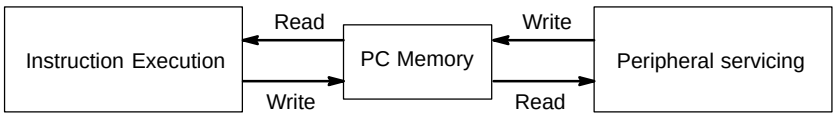
The CV-series PCs can execute instructions and refresh I/O in parallel with peripheral servicing (CPU Bus Units, Host Link Units, etc.). Peripheral servicing will access data in the Link Area, SYSMAC BUS/2 Area, CPU Bus Unit Area, and CPU Bus Link Area completely independently of program execution timing. This independent parallel processing is called asynchronous operation.

Normally, instruction execution and peripheral servicing are independent, however, an event processing request from a Unit is synchronized with program execution. The cycle time for asynchronous operation is the time required for instruction execution.



Note *Only BASIC Units and Personal Computer Units are refreshed in CPU Bus Unit servicing. Other CPU Bus Units are refreshed in interrupt processing.

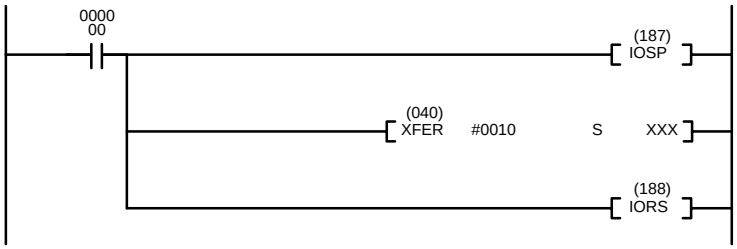
Programming Precautions When the PC is in asynchronous operation, instruction execution and peripheral servicing accesses I/O memory independently.



It is possible for the data in the CPU Bus Link Area, SYSMAC BUS/2 Area, CPU Bus Unit Area, etc., to be changed by peripheral servicing between the execution of two instructions or even during the execution of an instruction accessing many words. For example, if data is moved from XXXX in the CPU Bus Link Area to words D_1 and D_2 in two consecutive MOV(030) instructions, the data in D_1 and D_2 might be different if CPU Bus Link Area was refreshed between the execution of the first and second MOV(030) instructions.

If an instruction changing many words in the CPU Bus Link Area such as XFER(040) is executed while the CPU Bus Link Area is being refreshed, all of the words might not be transferred to the peripheral unit at the same time.

The DISABLE ACCESS IOSP(187) and ENABLE ACCESS IORS(188) instructions can be used to ensure that the data being accessed is not simultaneously accessed by peripheral servicing. In the example below, IOSP(187) and IORS(188) are used to ensure that the ten words of data transferred to the CPU Bus Link Area will be transmitted to the Peripheral Device at the same time.

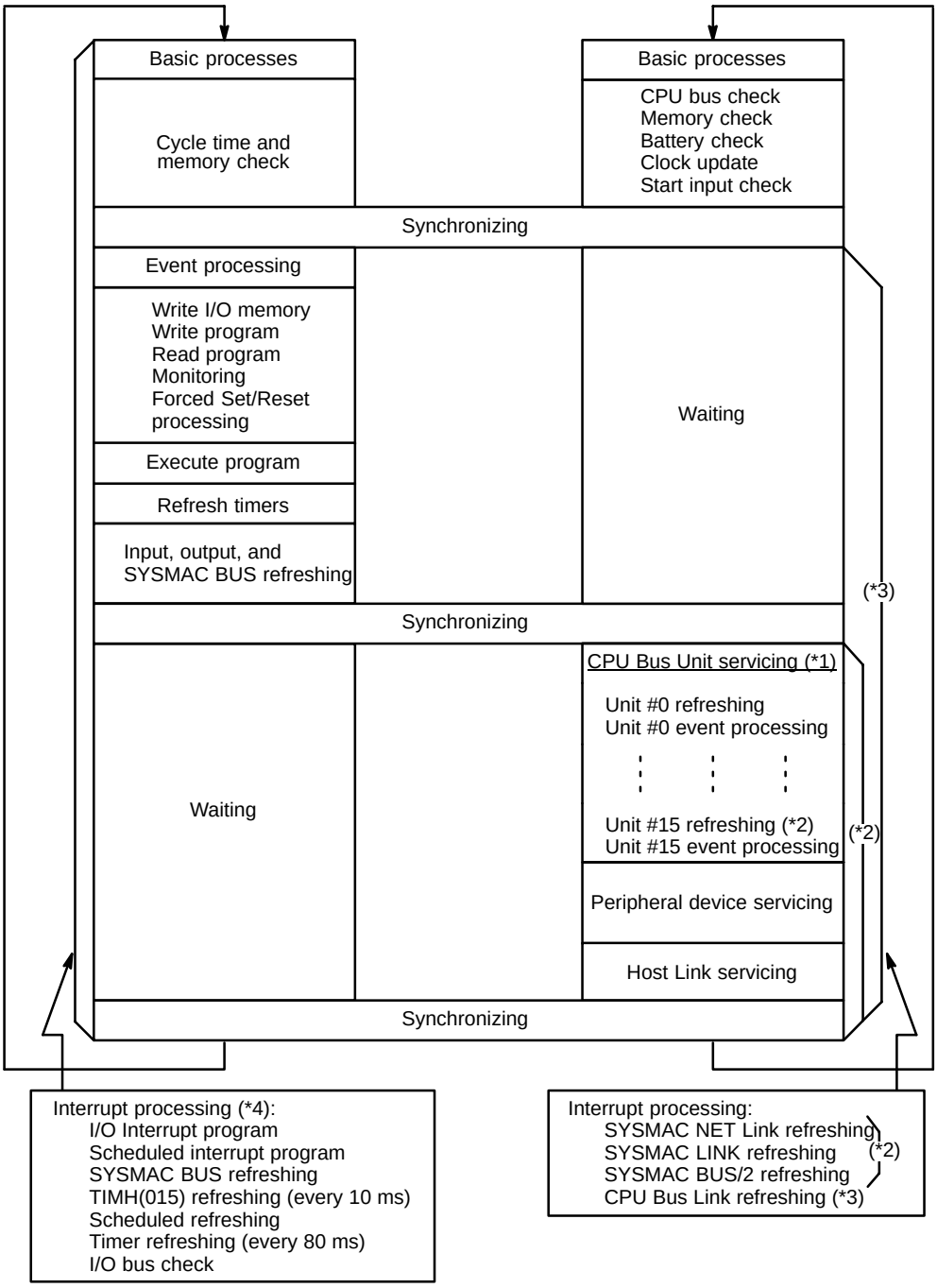


Address	Instruction	Operands
00000	LD	000000
00001	IOSP(187)	
00002	XFER(040)	
		#0010
		S
		XXX
00003	IORS(188)	

If IOSP(187) is executed while the CPU Bus Link Area is being refreshed, access to the CPU Bus Link Area won't be stopped until the refreshing has been completed. When XFER(040) is executed, CPU Bus Link Area refreshing will be disabled. CPU Bus Link Area refreshing will be enabled by IORS(188) after the 10 words of data have been transferred to the CPU Bus Link Area.

6-1-3 Synchronous Operation

PC operation can be set to synchronous operation in the PC Setup to synchronize instruction execution and peripheral servicing. The following diagram shows CPU operation during synchronous operation.



Note

1. Only BASIC Units and Personal Computer Units are refreshed in CPU Bus Unit servicing. Other CPU Bus Units are refreshed in interrupt processing.
2. Interrupts (for SYSMAC NET Link, SYSMAC LINK, and SYSMAC BUS/2 refreshing) are processed once each cycle, but if the cycle time is less than the communications cycle time, these Units might not be serviced every cycle.
3. The CPU Bus Link Area data may not be synchronized even with synchronous operation.
4. I/O interrupts, scheduled interrupts, and scheduled refreshing will be performed even during peripheral servicing.

6-1-4 I/O Refreshing

I/O refreshing refers to the reading of ON/OFF input bit data from Input Units to I/O memory and the writing of ON/OFF output bit data from I/O memory to Output Units.

CPU, CPU Expansion, and Expansion I/O Racks

The following list shows the five methods for refreshing I/O words allocated to Units on the CPU, CPU Expansion, and Expansion I/O Racks. The first three methods refresh all I/O points at once, the fourth refreshes a selected group of I/O words, and the fifth refreshes only those I/O points affected by an instruction.

1, 2, 3...

1. Cyclic refreshing
2. Scheduled refreshing
3. Zero-cross refreshing
4. IORF(184) refreshing
5. Immediate refreshing

One of the first three refreshing methods must be selected in the PC Setup. The default method is cyclic refreshing. Immediate and IORF(184) refreshing can be used in addition to the I/O refreshing method selected in the PC Setup. If all three methods are disabled by setting scheduled refreshing with an interval of 00, only Immediate and IORF(184) refreshing will be possible.

Cyclic Refreshing

In cyclic refreshing, all I/O points are refreshed once each cycle after the program is executed. When SFC programming is not used, I/O points are refreshed when the program has been executed from the first instruction to END(001); with SFC programming, I/O points are refreshed when the actions in active steps have all been executed. Output points are refreshed first, then input points.

Scheduled Refreshing

In scheduled refreshing, all I/O points are refreshed at a regular interval preset between 10 ms and 120 ms in the PC Setup. Scheduled refreshing is only possible when the PC is set for asynchronous operation. If a scheduled refresh occurs during program execution or interrupt processing, the scheduled refreshing will not begin until that procedure is completed. Scheduled refreshing can be stopped by turning ON bit A01705 (the I/O Refresh Disable Bit).

If the interval between refreshes is set to 00, refreshing will be disabled and only immediate or IORF(184) refreshing will be possible.

Zero-cross Refreshing

In zero-cross refreshing, all output points are refreshed when the AC power supply voltage crosses 0 V, the program is executed next, and then input points are refreshed. Zero-cross refreshing is for use with output points allocated to Output Units mounted on a CPU Rack, CPU Expansion Rack, or Expansion I/O Rack.

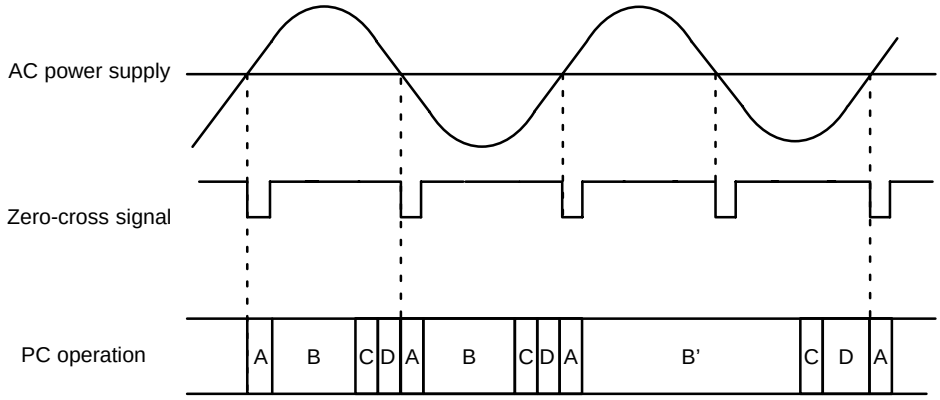
Zero-cross refreshing has the following advantages, particularly when used with AC output devices:

1, 2, 3...

1. The voltage supplied to the load is near zero when it is switched ON or OFF, so surge voltages are not generated.
2. The simultaneous ON time for devices such as solenoids operating against each other can be reduced and coil burning can be prevented.

Use a commercial power supply with a sine-wave output.

As shown in the following diagram, output points are refreshed (A) when the zero voltage signal is received, the program is executed (B), input points are refreshed (C), and the CPU waits for the next zero voltage signal (D). If I/O refreshing and program execution exceed one half the AC cycle (B'), the next output refreshing will occur when the next zero voltage signal is received.



A: Output points are refreshed
B: Program is executed
B': I/O refreshing and program execution exceed one half the AC cycle
C: Input points are refreshed
D: CPU waits for the next zero voltage signal

⚠ Caution Do not use zero-cross refreshing with the CV2000 or CVM1-CPU21-EV2 if the number of points output from the PC is greater than 1,024 (64 words). If zero-cross refreshing is used with more than 1,024 output points, zero-cross effectiveness will be lost for all outputs past the first 1,024 in memory.

IORF(184) Refreshing

IORF(184) requires two operands, St and E, which define the first and last word in a group of I/O words. When IORF(184) is executed, all words between St and E are refreshed. This is in addition to the I/O refreshing performed by the I/O refreshing method selected in the PC Setup. Refer to *5-27-4 I/O REFRESH – IORF(184)* for details.

Immediate Refreshing

Immediate refreshing refreshes the input/output points affected by an instruction immediately before/after the instruction is executed; this is in addition to the I/O refreshing performed by the I/O refreshing method selected in the PC Setup. Immediate refreshing is available for many instructions and is selected by entering a "!" prefix before the instruction when writing the program. Only I/O points allocated to I/O Units (except High-density I/O Units using dynamic I/O) mounted on a CPU Rack, Expansion CPU Rack, or Expansion I/O Rack can be refreshed with immediate refreshing.

SYSMAC BUS/2 and SYSMAC BUS Systems

The five methods for refreshing I/O described above cannot be used for I/O points allocated in SYSMAC BUS/2 and SYSMAC BUS Systems. Refreshing of I/O points in SYSMAC BUS/2 and SYSMAC BUS Systems differs for synchronous and asynchronous operation, as shown in the following tables.

SYSMAC BUS/2

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.

Asynchronous operation	Synchronous operation
I/O refreshing takes place once each communications cycle. (Refer to the *SYSMAC BUS/2 Remote I/O System Manual* for details on the communications cycle.)	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.

SYSMAC BUS

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.

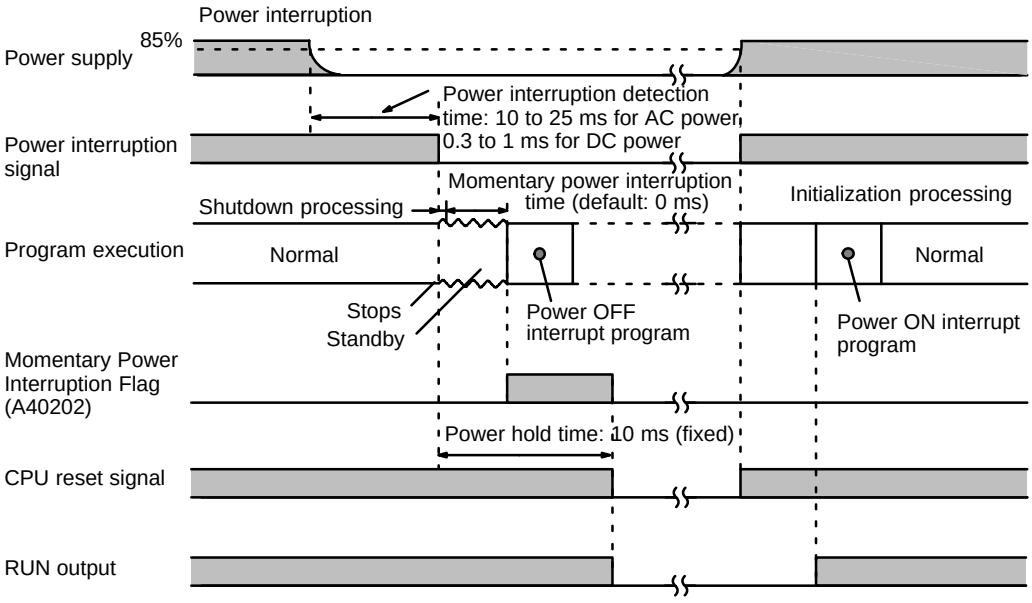
Asynchronous operation	Synchronous operation
I/O refreshing takes place at regular intervals. The length of the interval depends on the number of Masters. Multiply the number of Masters connected by 5 ms to calculate the interval.	I/O refreshing takes place once each PC cycle.

6-1-5 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, output status and timer PV status are maintained.

3. After the power interruption signal is output, the CPU waits for the momentary power interruption time set in the PC Setup (default: 0 ms) before proceeding. Set the power interruption time to 9–T ms max. (T is the time required to execute the power OFF interrupt program if there is one), because the power maintenance time is 10 ms and 1 ms is required for the system shutdown procedure.
4. The CPU determines that a power interruption has occurred if power hasn't returned by the end of the power interruption time. If a power OFF interrupt program has been prepared, it will be executed as soon as the instruction interrupted by the power interruption signal has been completed. The power OFF interrupt program will be stopped if it is still running 10 ms after the power interrupt signal was received (10 ms is the power hold time).

   Be sure that the total time required for the system shutdown procedure (1 ms), momentary power interruption time (set in the PC Setup), interrupted instruction completion time (depends on the interruption point), and the power OFF interrupt program (depends on program length) does not exceed 10 ms.
5. A CPU reset signal will be generated and the CPU will be stopped 10 ms after the power interruption signal is output. At this point, all outputs will turn OFF.

Power ON Procedure

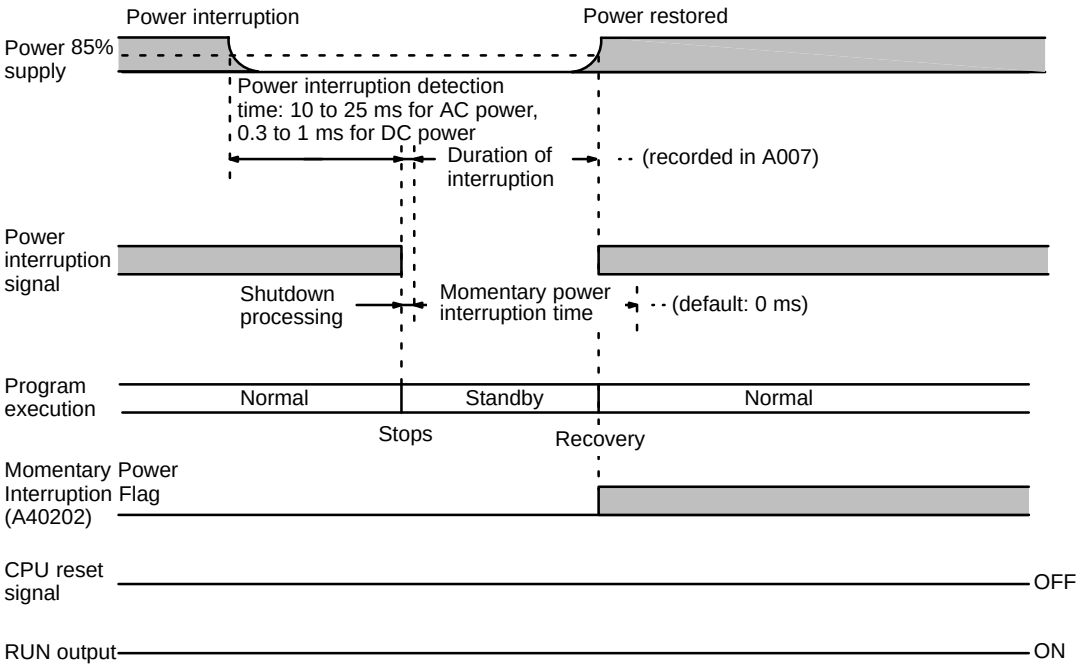
The following list shows CPU procedures when power is returned to the PC.

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. If the PC is set for restart continuation and a power ON interrupt program has been prepared, the power ON interrupt program will be executed first, then the main program.

Momentary Power Interruptions

The following diagram and explanation show the CPU operation when the power goes off momentarily.



The initial CPU operation during a momentary power interruption is identical to the first two items under the power OFF procedure heading above. In a momentary power interruption, power returns to 85% of full power before the power interruption time is exceeded.

When the CPU detects a power supply voltage above 85% of full power, the power interruption signal will be turned OFF and program execution will continue from the point at which it was interrupted. The momentary power interruption time (the time from the system shutdown procedure to the point that the power interruption signal was turned OFF) is recorded in milliseconds (0000 ms to 9999 ms) in A007. The day and time that the most recent power interruption occurred are recorded in A012 and A013, and the total number of interruptions is recorded in BCD in A014.

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)

The default for the momentary power interruption time in the PC Setup is 0 ms. Set the momentary power interruption time to 9 ms or less. Even if the momentary power interruption time exceeds the power hold time (10 ms), only those power interruptions shorter than 10 ms will be considered as momentary power interruptions.

To determine the actual length of a power interruption, add the time recorded in A007 to the time required to detect a power interruption (10 ms to 25 ms) and the time required to execute the system shut down procedure (1 ms).

6-1-6 Power OFF Interruption and Restart Continuation

This section details the steps required to prepare a power OFF interrupt program and to restart the PC after a power interruption.

Power OFF Interrupts

Power OFF Interrupt Program

Follow the steps below to use a power OFF interrupt program.

1, 2, 3...

1. Enable the power OFF interrupt in the PC Setup.
2. Write the power OFF interrupt program as an SFC program if SFC programming is being used or as a ladder diagram if SFC programming is not being used.

   The following instructions cannot be included in a power OFF interrupt program: FAL(006), FALS(007), FILR(180), FILW(181), FILP(182), IOSP(187), READ(190), WRIT(191), SEND(192), RECV(193), CMND(194), SA(210), SP(211), SR(212), SF(213), SE(214), and SOFF(215).
3. The time allowed for a power OFF interrupt program is limited. Verify that the total time required for the system shutdown procedure (1 ms), momentary power interruption time (set in the PC Setup), interrupted instruction completion time (depends on the instruction that is interrupted), and the power OFF interrupt program (depends on program length) does not exceed 10 ms.

   The power OFF interrupt program will stopped if it is still running 10 ms after the power interrupt signal was received.

Restart Continuation

Settings

The following steps are required to cause the PC to automatically resume operation after the program has been stopped because of a power interruption.

1, 2, 3...

1. Select the following settings in the PC Setup.

Item		Required setting
Startup Hold Settings	IOM Hold Bit Status	Hold (Yes)
	Restart Continuation Bit Status	Hold (Yes)
Startup Mode Setting		RUN or MONITOR
Execution Controls 2	Power OFF interrupt	Enabled (Yes)

2. Turn ON the following Auxiliary Area control bits. These are normally turned ON from the program.

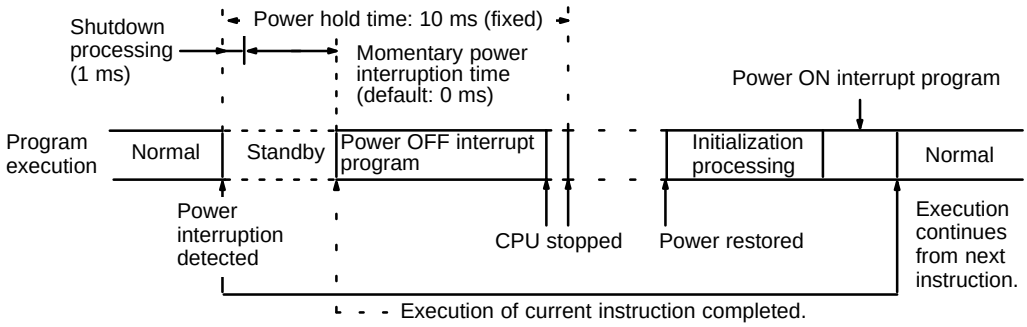
Bit	Bit name	Setting
A00011	Restart Continuation Bit Hold	ON
A00012	IOM Hold Bit	ON

3. A power OFF interrupt program must be prepared for restart continuation.
4. A power ON interrupt program may be written if required. The power ON interrupt program will only be executed if the PC is set for restart continuation.

   Write the power ON interrupt program as an SFC program if SFC programming is being used or as a ladder diagram if SFC programming is not being used.
5. When the CPU is stopped due to a power interruption, all outputs will be turned OFF and the Output OFF Bit (A00015) will be turned ON. When restarting the PC, the Output OFF Bit must be turned OFF in either the power ON interrupt program or the main program.

Operation

The diagram and explanation below describe CPU operation when the PC is set for restart continuation and power is interrupted and then returned after the CPU is stopped.



1, 2, 3...

1. Execution of the main program is stopped when the power interruption signal is output, and the CPU then waits for the momentary power interruption time set in the PC Setup.
2. The power OFF interrupt program is executed when the momentary power interruption time is exceeded. If program execution was interrupted by the power interruption signal, the power OFF interrupt program is executed after the interrupted instruction has been completed.
3. The CPU will be stopped and all outputs turned OFF when either the power OFF interrupt program is completed or 10 ms has passed since the power interruption signal was output, whichever occurs first.
4. Initialization processing will be performed when power returns to the PC.
5. After initialization, the power ON interrupt program will be executed if one has been prepared.
6. The main program will then be executed from the instruction just after the one that was being executed when the power interruption signal was output.

Precautions

Take the following precautions to ensure proper restart continuation.

1, 2, 3...

1. If 10 ms has elapsed since the power interrupt signal was received, the CPU will be stopped even if the power OFF interrupt program is still running. Be sure that the power OFF interrupt program is as short as possible.

   Verify that the total time required for the system shutdown procedure (1 ms), momentary power interruption time (set in the PC Setup), interrupted instruction completion time (depends on the instruction that is interrupted), and the power OFF interrupt program (depends on program length) does not exceed 10 ms.
2. When program execution was interrupted by the power interruption signal, the power OFF interrupt program is executed after the interrupted instruction is completed.

   A particularly long instruction, such as SEND(192) or RECV(193), might not be completed by the end of the 10 ms power hold time, and some data might be lost.
3. Restart continuation will not occur when:
   d) A fatal error (one stopping the CPU) occurs.
   e) The power OFF interrupt program was not completed within the power hold time (10 ms).
   f) The PC memory has somehow been completely altered when power was interrupted.

   If memory is disrupted, the main program will be executed from the beginning if the PC can still run when power returns.

4. If a power interruption occurs during initialization or execution of the power ON interrupt program, restart continuation will begin at initialization or from the beginning of the power ON interrupt program.
5. Timers are stopped when a power interruption occurs and their PVs are maintained. The timers begin counting again when the power OFF interrupt program is executed, and are once again stopped with PVs maintained when the power OFF interrupt program is completed.

   Timing begins again when the main program is started. Timers are held during initialization and execution of the power ON interrupt program.
6. An I/O interrupt program will be executed from the instruction just after the one that was being executed when the power interruption signal was output, just like the main program.

   When power returns, all I/O interrupts will be masked, so the masks will have to be removed with MSKS(153) in the power ON interrupt program if the interrupts are to operate.
7. Scheduled interrupts are cancelled by a power interruption, so reset scheduled interrupts in the power ON program if necessary.
8. If SE(214)/SOFF(215) are executed in the power ON interrupt program, main program execution will start from the active step with the highest priority.
9. When power is interrupted, the address being executed is recorded in the CPU and program execution proceeds from the next address when the PC is restarted. The program on the Memory Card must therefore be identical to the one in the PC if the program is automatically transferred from the Memory Card when power returns to the PC. If the program has been changed, the PC might not operate properly.
10. If the DIP switch on the CPU is set to transfer the program from the Memory Card on power-up, the PC Setup will be transferred at the same time. Be sure that any important changes to the PC Setup are recorded in the version of the PC Setup on the Memory Card.

Parameters Maintained for Restart Continuation

The following parameters are maintained when the PC is restarted.

1, 2, 3...

1. Active status of steps
2. Execution status of interrupt programs
3. The cause of interrupt programs received before the power interruption (except for power OFF interrupt program)
4. The address being executed in the program
5. Interlock status
6. IOM (I/O memory) status
7. Forced Set/Reset status (if PC Setup and A00013 are set to maintain forced status)
8. Timer and counter PVs
9. Step timer PVs
10. Elapsed time for AQ (L, D, SL, SD, DS)
11. Trace execution status
12. Trace initial start, when the PC Setup is set for Trace execution on power-up
13. Arithmetic Flag status saved with CCS(173)
14. Index register contents
15. The current EM bank number

Note Even if the current EM bank number is changed in the power OFF interrupt program, the EM bank number will revert to the one that was valid just before the power OFF interrupt program was executed.

Parameters Not Maintained for Restart Continuation

The following parameters are not maintained when the PC is restarted.

1, 2, 3...

1. Error status (but the Error Log is maintained.)
2. Disabled access to I/O memory by IOSP(187)
3. Data displayed on I/O Control Units, I/O Interface Units, and Slave Units by IODP(189)
4. Status of CPU Bus Unit Service Disable Bits (A015)
5. Status of Service Disable Bits in A017
6. Message data
7. I/O interrupt mask status
8. Scheduled interrupt interval data
9. Data Register contents
10. Execution status of differentiated status monitoring
11. Execution status of execution time measurements
12. Execution status of stop monitoring
13. Execution of the following instructions may not be completed depending on how much of the instruction had been executed when the power interruption occurred: FILR(180), FILW(181), FILP(182), FLSP(183), READ(190), WRIT(191), SEND(192), RECV(193), CMND(194)

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 Asynchronous Operation

In asynchronous operation, instruction execution and I/O refreshing are processed in a cycle that is independent of the cycle for peripheral servicing (CPU Bus Units, host interface, etc.). Since the cycles are independent, the number of CPU Bus Units, host interface links, etc., that are connected to the PC has no effect on the instruction execution cycle time.

Instruction Execution Cycle Time

Process	Actions	Processing time	
		CV500/CVM1-CPU01-EV2	**CV1000/2000/ CVM1-CPU11/21-EV2**
Basic processes	Checks the cycle time and PC memory.	0.6 ms	0.5 ms
Event processing	Execution of non-scheduled requests for data processing from peripherals.	0 ms if there is no processing request; 20 ms max. (12% of 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 Instruction Execution Times* for details. For SFC programs, refer to the *CV-series PC Operation Manual: SFC*.	
Timer refreshing	Updates the PVs of all timers in the program.	10 μs + 1.1 μs per timer	8 μs + 0.9 μs per timer
Output refreshing	Output terminals updated according to status of output bits in memory (including Special I/O Units).	8 μs per word (per 16 pts.)	7 μs per word (per 16 pts.)
Input refreshing	Input bits updated according to the status of input signals (including Special I/O Units).	10 μs per word (per 16 pts.)	8 μs per word (per 16 pts.)

Interrupt Processing

Depending on the program, the following interrupt processes is executed in addition to the processes detailed in the table above. The actual cycle time is the sum of the cycle time calculated in the table above and the time required for the processes in the table below.

Process	Actions	Processing time	
		CV500/CVM1-CPU01-EV2	**CV1000/2000/ CVM1-CPU11/21-EV2**
I/O interrupts	I/O interrupt programs started with receipt of interrupt requests from Interrupt Input Units.	Depends on the I/O interrupt program.	
Scheduled interrupt	Scheduled interrupt program(s) started at preset interval(s).	Depends on the Scheduled interrupt program.	
SYSMAC BUS refreshing	Masters receive I/O data from Slaves every 5 ms.	0.6 ms per Master plus 60 μs per word controlled though each Master	0.5 ms per Master plus 50 μs per word controlled though each Master
TIMH(015) refreshing	Updates the PVs of high-speed timers in the program every 10 ms.	12 μs + 0.8 μs per timer	10 μs + 0.7 μs per timer
Timer refreshing	Updates the PVs of all timers in the program every 80 ms if the cycle time exceeds 80 ms.	10 μs + 1.1 μs per timer	8 μs + 0.9 μs per timer
I/O bus check	Checks the I/O bus on the CPU, Expansion CPU, and Expansion I/O Racks	0.7 μs every 20 ms	0.6 μs every 20 ms

Peripheral Servicing Cycle Time

Processes	Actions	Processing time	
		CV500/CVM1-CPU01-EV2	CV1000/2000/ CVM1-CPU11/21-EV2
Basic processes	Checks the CPU bus, PC memory, battery, start input, and updates the clock.	Approx. 2.8 ms	
CPU Bus Unit servicing	Starting from Unit #0, events (requests for non-scheduled data processing) are processed to refresh BASIC Unit and Personal Computer Units.	Approx. 1 ms for event processing; approx. 0.8 ms for BASIC Unit refreshing; and approx. 0.8 ms for Personal Computer Unit refreshing.	
Peripheral Device servicing	If a peripheral Device is connected, commands from it are processed.	1.2 ms	
Host interface servicing (Pin 3 of DIP switch OFF)	Commands from computers connected through the host interface are processed.	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.	Fixed at 2.2 ms	

Interrupt Refresh Processing

Depending on the program, the following interrupt processes might be executed in addition to the processes detailed in the table above. The actual cycle time is the sum of the cycle time calculated in the table above and the time required for the processes in the table below.

Process	Actions	Processing time	
		CV500/CVM1-CPU01-EV2	CV1000/2000/ CVM1-CPU11/21-EV2
SYSMAC NET Link refreshing	The data link words allocated to SYSMAC NET Link Units are refreshed.	Approx. 1.4 ms + 1 µs per data link word, add 0.3 ms if both DM and CIO are used for data links.	
SYSMAC LINK refreshing	The data link words allocated to SYSMAC LINK Units are refreshed.	Approx. 1.4 ms + 1 µs per data link word, add 0.3 ms if both DM and CIO are used for data links.	
SYSMAC BUS/2 refreshing	Masters receive I/O data from Slaves.	Approx. 2.0 ms plus 1 µs per word refreshed	
CPU Bus Link refreshing	Data in the CPU Bus Link Area is refreshed.	0.8 ms every 10 ms (when the PC is set to use the CPU Bus Link Area in the PC Setup)	

6-2-2 Synchronous Operation

In synchronous operation, instruction execution and I/O refreshing are processed along with peripheral servicing (CPU Bus Units, host interface, etc.) in a single cycle. The cycle time is thus the sum of the time required for instruction execution and that required for peripheral servicing, and the cycle time will lengthen as more peripherals are connected.

Instruction Execution Cycle Time

Process	Actions		Processing time	
			CV500/CVM1-CPU01-EV2	CV1000/2000/ CVM1-CPU11/21-EV2
Basic processes	Checks the cycle time and PC memory.	Checks the CPU bus, PC memory, battery, start input, and updates the clock.	Approx. 3.7 ms	Approx. 3.5 ms
Event processing	Execution of non-scheduled requests for data processing from peripherals.		0 ms if there is no processing request; 20 ms max. (12% of 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 Instruction Execution Times* for details. For SFC programs, refer to the *CV-series PC Operation Manual: SFC*.	
Timer refreshing	Updates the PVs of all timers in the program.		10 μs + 1.1 μs per timer	8 μs + 0.9 μs per timer
Output refreshing	Output terminals are updated according to status of output bits in memory (including Special I/O Units).		8 μs per word (per 16 pts.)	7 μs per word (per 16 pts.)
Input refreshing	Input bits are updated according to status of input signals (including Special I/O Units).		10μs per word (per 16pts.)	8 μs per word (per 16 pts.)
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 60 μs per word controlled though each Master	1 ms per Master plus 50 μs per word controlled though each Master
CPU Bus Unit servicing	Starting from Unit #0, events (requests for non-scheduled data processing) are processed to refresh BASIC Unit and Personal Computer Units.		Approx. 1 ms for event processing; approx. 0.8 ms for BASIC Unit refreshing; and approx. 0.8 ms for Personal Computer	
Peripheral Device servicing	If a peripheral device is connected, commands from it are processed.		1.2 ms	
Host interface servicing (Pin 3 of DIP switch OFF)	Commands from computers connected through the host interface are processed.		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.		Fixed at 2.2 ms	

Interrupt Processing

Depending on the program, the following interrupt processes might be executed in addition to the processes detailed in the table above. The actual cycle time is the sum of the cycle time calculated in the table above and the time required for the processes in the table below.

Process	Actions	Processing time	
		CV500/CVM1-CPU01-EV2	CV1000/2000/ CVM1-CPU11/21-EV2
I/O interrupts	I/O interrupt programs started with receipt of interrupts from Interrupt Input Units.	Depends on the I/O interrupt programs.	
Scheduled interrupt	Scheduled interrupt program(s) started at preset interval(s).	Depends on the scheduled interrupt program(s).	
TIMH(015) refreshing	Updates the PVs of high-speed timers in the program every 10 ms.	12 μs + 0.8 μs per timer	10 μs + 0.7 μs per timer
Timer refreshing	Updates the PVs of all timers in the program every 80 ms if the cycle time exceeds 80 ms.	10 μs + 1.1 μs per timer	8 μs + 0.9 μs per timer
I/O bus check	Checks the I/O bus on the CPU, Expansion CPU, and Expansion I/O Racks	0.6 μs every 20 ms	
SYSMAC NET Link refreshing	The data link words allocated to SYSMAC NET Link Units are refreshed. Interrupt processing can occur as often as once each cycle.	Approx. 1.4 ms + 1 μs per data link word, add 0.3 ms if both DM and CIO are used for data links.	
SYSMAC LINK refreshing	The data link words allocated to SYSMAC LINK Units are refreshed. Interrupt processing can occur as often as once each cycle.	Approx. 1.4 ms + 1 μs per data link word, add 0.3 ms if both DM and CIO are used for data links.	
SYSMAC BUS/2 refreshing	Masters receive I/O data from Slaves.	Approx. 2.0 ms plus 1 μs per word refreshed	
CPU Bus Link refreshing	Data in the CPU Bus Link Area is refreshed.	0.8 ms every 10 ms (when the PC is set to use the CPU Bus Link Area in the PC Setup)	

Service Disable Bits

The Service Disable Bits shown in the table below are primarily used during synchronous operation 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	Peripheral Service Disable Bit
	05	I/O Refresh Disable Bit

6-2-3 Operations Significantly Increasing Cycle Time

The instruction trace operation described below can significantly affect the cycle time when performed from CVSS/SSS.

Operation	Effect on cycle time	Precautions
Instruction trace	Cycle time increased 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.

Note

1. In the earlier version, executing the online edit operation could increase cycle time by as much as 3 seconds. In the CVM1(V2), 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). While PC operation is stopped, output status is retained and inputs are not accepted.

Online edit operation	Stopped time
Instruction block inserted or deleted at the beginning of the 62K-word users program.	Approx. 0.5 s
Instruction block including JME(005) deleted at the beginning of the 62K-word users program.	Approx. 2.0 s

2. In the earlier version, executing FILP(182) could increase cycle time by as much as 3 seconds, but this instruction does not affect cycle time in the CVM1(V2). Just as in the earlier version, however, PC operation is stopped for as much as 30 seconds for program replacement processing. While PC operation is stopped, output status is retained and inputs are not accepted.

   Communications with SYSMAC NET, SYSMAC LINK, SYSMAC BUS/2, Host Link, and peripheral devices also stop during those periods. Therefore, when using a program in which FILP(182) is executed, set the response time in the System Setup to 30 seconds or more.

3. These changes apply not only to the CVM1(V2), but also to CV500, CV1000, and CV2000 products in which the rightmost digit of the lot number is “5.”

6-2-4 Potential Problems with Event Processing

It is possible for two or more events (unscheduled requests for data processing) from Peripheral 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 CVSS/SSS operations are writing events:

Data modification, set, reset, online edit, DM edit, mode change, transfer block (CVSS/SSS to PC), save block (CVSS/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, Memory Card to PC transfer, 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 CVSS/SSS operations are instruction execution events:

Monitoring, data modification, set, reset, online edit, DM edit, search, transfer block (CVSS/SSS to PC), save block (CVSS/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, Memory Card (program, I/O memory)

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 read, program area write, program area clear, cycle time read, program area to file transfer, force set/reset, force set/reset for all bits

SFC Online Editing

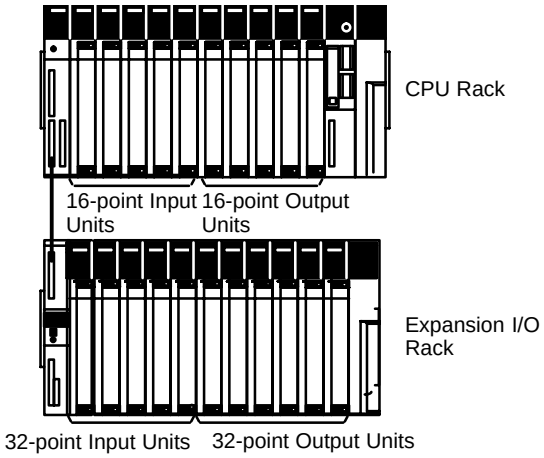
When SFC online editing is selected from the CVSS, all writing events will be suspended until the online editing has been completed.

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 taking into account such things as the number of I/O points, the programming instructions used, and whether or not Peripheral 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 CV1000 or CV2000 set for cyclic refreshing. The PC controls only I/O Units, ten on the CPU Rack and eleven 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 equation for the cycle time from above is as follows:

Cycle time = overseeing time (basic processes)
+ program execution
+ output refreshing
+ input refreshing time

The overseeing time is fixed at 0.5 ms for nonsynchronous processing.

The program execution time is 6 ms (0.3 μs/instruction times 20,000 instructions).

The output refreshing time would be as follows for the five 16-point Output Units and six 32-point Output Units controlled by the PC:

$$\frac{(16 \text{ points} \times 5) + (32 \text{ points} \times 6)}{16 \text{ points}} \times 7\ \mu s = 0.12 \text{ ms}$$

The input refresh time would be as follows for the five 16-point Input Units and five 32-point Input Units controlled by the PC:

$$\frac{(16 \text{ points} \times 5) + (32 \text{ points} \times 5)}{16 \text{ points}} \times 8\ \mu s = 0.12 \text{ ms}$$

The cycle time would thus be:

0.5 ms + 6.0 ms + 0.12 ms + 0.12 ms □ 6.74 ms

6-4 Instruction Execution Times

This following table lists the execution times for CV-series PC 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. With CV-series PCs, 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)	
			CV500*	CV1000*	CV500*	CV1000*
LD	1	CIO 000000 to CIO 051115 for operand	0.15	0.13	0.15	0.13
	2	CIO 051200 to CIO 255515 for operand	0.3 ↑/↓: 0.45 !: 2.25	0.25 ↑/↓: 0.375 !: 2.0	0.3 ↑/↓: 0.75 !: 2.25	0.25 ↑/↓: 0.625 !: 2.0
LD NOT	1	CIO 000000 to CIO 051115 for operand	0.15	0.13	0.15	0.13
	2	CIO 051200 to CIO 255515 for operand	0.3 ↑/↓: 0.45 !: 2.25	0.25 ↑/↓: 0.375 !: 2.0	0.3 ↑/↓: 0.45 !: 2.25	0.25 ↑/↓: 0.375 !: 2.0
AND	1	CIO 000000 to CIO 051115 for operand	0.15	0.13	0.15	0.13
	2	CIO 051200 to CIO 255515 for operand	0.3 ↑/↓: 0.45 !: 2.25	0.25 ↑/↓: 0.375 !: 2.0	0.3 ↑/↓: 0.45 !: 2.25	0.25 ↑/↓: 0.375 !: 2.0
AND NOT	1	CIO 000000 to CIO 051115 for operand	0.15	0.13	0.15	0.13
	2	CIO 051200 to CIO 255515 for operand	0.3 ↑/↓: 0.45 !: 2.25	0.25 ↑/↓: 0.375 !: 2.0	0.3 ↑/↓: 0.45 !: 2.25	0.25 ↑/↓: 0.375 !: 2.0
OR	1	CIO 000000 to CIO 051115 for operand	0.15	0.13	0.15	0.13
	2	CIO 051200 to CIO 255515 for operand	0.3 ↑/↓: 0.45 !: 2.25	0.25 ↑/↓: 0.375 !: 2.0	0.3 ↑/↓: 0.45 !: 2.25	0.25 ↑/↓: 0.375 !: 2.0
*Note: CV500 = CV500 or CVM1-CPU01-EV2; CV1000 = CV1000, CV2000, or CVM1-CPU11/21-EV2						

Instruction	Words	Conditions	ON execution time (μs)		OFF execution time (μs)	
			CV500*	CV1000*	CV500*	CV1000*
OR NOT	1	CIO 000000 to CIO 051115 for operand	0.15	0.13	0.15	0.13
	2	CIO 051200 to CIO 255515 for operand	0.3 ↑/↓: 0.45 !: 2.25	0.25 ↑/↓: 0.375 !: 2.0	0.3 ↑/↓: 0.45 !: 2.25	0.25 ↑/↓: 0.375 !: 2.0
AND LD	1	---	0.15	0.13	0.15	0.13
OR LD	1	---	0.15	0.13	0.15	0.13
OUT	2	---	0.45 !: 2.25	0.38 !: 1.75	0.45 !: 2.25	0.38 !: 1.75
OUT NOT	2	---	0.45 !: 2.25	0.375 !: 1.75	0.45 !: 2.25	0.375 !: 1.75
TIM	3	Constant for SV	1.2	1.0	R or IL: 6.6	R or IL: 5.5
		∗DM for SV			R or IL: 10.2	R or IL: 8.5
CNT	3	Constant for SV	1.2	1.0	R or IL: 1.2	R or IL: 1.0
		∗DM for SV			R: 6.6 IL: 1.2	R: 5.5 IL: 1.0
NOP(000)	1	---	0.15	0.13	---	---
END(001)	1		4.5	3.75		
IL(002)	2	---	1.8	1.5	1.2	1.0
ILC(003)	2		1.2	1.0	1.2	1.0
JMP(004)	3	---	13.2	11.0	1.05	0.88
JME(005)	2	---	0.3	0.25	0.3	0.25
FAL(006)	4	---	465	457	1.2	1.0
FAL(006) 00			620	611	1.2	1.0
FALS(007)	4		837	825	1.2	1.0
STEP(008)	3	---	262	218	245	207
SNXT(009)	3	---	270	225	1.05	0.875
NOT(010)	1	---	0.15	0.13	0.15	0.13
KEEP(011)	2	---	0.45	0.38	0.45	0.38
CNTR(012)	4	Constant for SV	7.2	6.0	R: 9.0 IL: 6.3	R: 7.5 IL: 5.3
		∗DM for SV	9.6	8.0	R: 10.7 IL: 6.3	R: 8.9 IL: 5.3
DIFU(013)	2	---	0.75	0.63	0.75	0.63
DIFD(014)	2	---	0.75	0.63	0.75	0.63
TIMH(015)	3	Constant for SV	1.2	1.0	R or IL: 6.6	R or IL: 5.5
		∗DM for SV	1.2	1.0	R or IL: 10.2	R or IL: 8.5
SET(016)	2	---	0.45	0.375	0.45	0.375
RSET(017)	2					
UP(018)	2	---	4.8	4.0	4.8	4.0
DOWN(019)	2					
CMP(020)	4	When comparing a constant to a word	3.9	3.3	1.05	0.88
		When comparing two ∗DM	7.1	5.9		

*Note: CV500 = CV500 or CVM1-CPU01-EV2; CV1000 = CV1000, CV2000, or CVM1-CPU11/21-EV2

Instruction	Words	Conditions	ON execution time (μs)		OFF execution time (μs)	
			CV500*	CV1000*	CV500*	CV1000*
!CMP(020)	4	Additional time over CMP(020) for each input word being compared	+5.5	+5.0	1.05	0.88
		Additional time over CMP(020) for each output word being compared	+4.4	+4.0		
		Additional time over CMP(020) for words other than I/O words	+0	+0		
CMPL(021)	4	When comparing two words	5.1	4.3	1.2	1.0
		When comparing two *DM	8.1	6.8		
BCMP(022)	5	Comparing constant to table with results to a word	20.3	16.9	1.35	1.13
		Comparing *DM to table with results to *DM	24.3	20.3		
TCMP(023)	5	When comparing a word to a word table	17.9	14.9		
		When comparing a *DM to *DM table	21.9	18.3		
MCMP(024)	5	When comparing two words	20.6	17.3		
		When comparing two *DM	24.2	20.3		
EQU(025)	4	Comparing two words	3.8	3.1	1.2	1.0
		Comparing two *DM	6.9	5.8		
CPS(026)	4	Comparing constants and words	4.2	3.5		
		Comparing two *DM	7.5	6.25		
!CPS(026)	4	Amount added per input word at time of comparison	+5.5	+5.0		
		Amount added per output word at time of comparison	+4.4	+4.0		
		Other areas at time of comparison	+0	+0		
CPSL(027)	4	Comparing constants and words	5.4	4.5		
		Comparing two *DM	8.25	6.88		
CMP(028)	4	Comparing constants and words	3.9	3.3	1.05	0.88
		Comparing two *DM	7.1	5.9		
!CMP(028)	4	Amount added per input word at time of comparison	+5.5	+5.0		
		Amount added per output word at time of comparison	+4.4	+4.0		
		Other areas at time of comparison	+0	+0		
CMPL(029)	4	Comparing two words	5.1	4.3	1.2	1.0
		Comparing two *DM	8.1	6.8		
MOV(030)	4	When transferring a constant to a word	5.1	4.3		
		When transferring *DM to *DM	6.3	5.3		
!MOV(030)		Additional time over MOV(020) for each input word being used	+5.5	+5.0		
		Additional time over MOV(020) for each output word being used	+4.4	+4.0		
		Additional time over MOV(020) for words other than I/O words	+0	+0		
MVN(031)	4	When transferring a constant to a word	5.3	4.4		
		When transferring *DM to *DM	6.5	5.4		
MOVL(032)	4	When transferring a word to a word	6.5	5.4		
		When transferring *DM to *DM	7.5	6.3		

*Note: CV500 = CV500 or CVM1-CPU01-EV2; CV1000 = CV1000, CV2000, or CVM1-CPU11/21-EV2

Instruction	Words	Conditions	ON execution time (μs)		OFF execution time (μs)	
			CV500*	CV1000*	CV500*	CV1000*
MVNL(033)	4	When transferring a word to a word	6.5	5.4	1.2	1.0
		When transferring *DM to *DM	7.5	6.3		
XCHG(034)	4	Word → word	7.7	6.4		
		*DM → *DM	8.3	6.9		
XCGL(035)	4	Word → word	10.2	8.5		
		*DM → *DM	10.8	9.0		
MOVR(036)	3	Word → IR	5.0	4.1	1.05	0.88
		*DM → IR	5.3	4.4		
MOVQ(037)	3	When transferring a word to a word	0.6	0.5	0.45	0.38
		When transferring a constant to a word	0.75	0.63		
XFRB(038)	3	When transferring 1 bit from word to word	15.0	12.5	1.35	1.13
		When transferring 255 bits from *DM to *DM	58.7	48.9		
XFER(040)	5	When transferring 1 word	14.7	12.3		
		When transferring 1000 words by *DM	1.81 ms	1.51 ms		
BSET(041)	5	When setting a constant to one word	13.7	11.4		
		When setting 1000 DM words using *DM	1.39 ms	1.16 ms		
MOVB(042)	5	When transferring a word to a word	9.8	8.1		
		When transferring *DM to *DM	13.7	11.4		
MOVD(043)	5	When transferring a word data to a word	8.9	7.4		
		When transferring *DM to *DM	12.8	10.8		
DIST(044)	5	Constant → (word + word)	6.9	5.8		
		*DM → (*DM + *DM)	9.3	7.8		
COLL(045)	5	(word + word) → word	7.5	6.3		
		(*DM + *DM) → *DM	11.3	9.4		
BXFR(046)	5	Transferring 1 word from word to word	19.4	16.1		
		*DM → *DM, *DM (1000 words) transfer	1.82 ms	1.52 ms		
SETA(047)	5	Setting 1 bit in a word	16.5	13.8		
		*DM → *DM (1000 bits) set	119	99.5		
RSTA(048)	5	Resetting 1 bit in a word	16.7	13.9		
		*DM → *DM, *DM (1000 bits) reset	120	99.8		
SFT(050)	5	With 1 word shift register	16.8	14.0	R: 14.4 IL: 1.2	R: 12.0 IL: 1.0
		With 1000 words shift register	1.54 ms	1.28 ms	R: 1.38 ms IL: 1.2	R: 1.15 ms IL: 1.0
SFTR(051)	5	When shifting 1 word	16.8	13.5	1.35	1.13
		When shifting DM 1000 words using *DM	1.56 ms	1.30 ms		
ASFT(052)	5	When shifting 1 word	439	366		
		When shifting DM 1000 words using *DM	16.0 ms	13.3 ms		

*Note: CV500 = CV500 or CVM1-CPU01-EV2; CV1000 = CV1000, CV2000, or CVM1-CPU11/21-EV2

Instruction	Words	Conditions	ON execution time (μs)		OFF execution time (μs)	
			CV500*	CV1000*	CV500*	CV1000*
WSFT(053)	5	When shifting 1 word	13.8	11.5	1.35	1.13
		When shifting 1000 DM words using *DM	1.40 ms	1.17 ms		
NSFL(054)	5	Shifting 1 bit in a word	18.9	15.8		
		*DM → *DM (15 bits) shift	38.9	32.4		
NSFR(055)	5	Shifting 1 bit in a word	18.8	15.6		
		*DM → *DM (15 bits) shift	38.9	32.4		
NASL(056)	4	1-bit word shift	17.0	14.1	1.2	1.0
		*DM (16 bits) shift	23.3	19.4		
NASR(057)	4	1-bit word shift	17.0	14.1		
		*DM (16 bits) shift	23.3	19.4		
NSLL(058)	4	1-bit word shift	18.0	15.0		
		*DM (32 bits) shift	42.5	35.4		
NSRL(059)	4	1-bit word shift	17.7	14.8		
		*DM (32 bits) shift	42.2	35.1		
ASL(060)	3	When shifting a word to left	6.2	5.1	1.05	0.88
		When shifting *DM to left	7.4	6.1		
ASR(061)	3	When shifting a word to right	6.3	5.3		
		When shifting *DM to right	7.5	6.3		
ROL(062)	3	When rotating a word to counterclockwise	6.3	5.3		
		When rotating *DM to counterclockwise	7.5	6.3		
ROR(063)	3	When rotating a word to clockwise	6.5	5.4		
		When rotating *DM to clockwise	7.7	6.4		
ASLL(064)	3	When shifting two words to left	7.5	6.3		
		When shifting two *DM to left	8.7	7.3		
ASRL(065)	3	When shifting two words to right	7.5	6.3		
		When shifting two *DM to right	8.7	7.3		
ROLL(066)	3	When rotating two words to counterclockwise	7.7	6.4		
		When rotating two *DM to counterclockwise	8.9	7.4		
RORL(067)	3	When rotating two words to clockwise	7.7	6.4		
		When rotating two *DM to clockwise	8.9	7.4		
SLD(068)	4	When shifting 1 word	13.7	11.4	1.2	1.0
		When shifting 1000 DM words using *DM	181	151		
SRD(069)	4	When shifting 1 word	13.7	11.4		
		When shifting 1000 DM words using *DM	181	151		
ADD(070)	5	Constant + word → word	9.0	7.5	1.35	1.13
		*DM + *DM → *DM	10.8	9.0		
SUB(071)	5	Constant – word → word	8.9	7.4		
		*DM – *DM → *DM	10.7	8.9		
MUL(072)	5	Constant x word → word	22.1	18.4		
		*DM x *DM → word	23.7	19.8		
DIV(073)	5	Word ÷ constant → word	21.0	17.5		

*Note: CV500 = CV500 or CVM1-CPU01-EV2; CV1000 = CV1000, CV2000, or CVM1-CPU11/21-EV2

Instruction	Words	Conditions	ON execution time (μs)		OFF execution time (μs)	
			CV500*	CV1000*	CV500*	CV1000*
		∗DM ÷ ∗DM → ∗DM	25.4	21.1		
ADDL(074)	5	Constant + word → word	21.8	18.0	1.35	1.13
		∗DM + ∗DM → ∗DM	25.8	21.5		
SUBL(075)	5	Constant – word → word	21.0	17.5		
		∗DM – ∗DM → ∗DM	25.2	21.0		
MULL(076)	5	Constant x word → word	64.5	53.8		
		∗DM x ∗DM → word	68.7	57.3		
DIVL(077)	5	Word ÷ constant → word	74.6	62.1		
		∗DM ÷ ∗DM → ∗DM	78.6	65.5		
STC(078)	2	---	1.65	1.38	0.9	0.75
CLC(079)	2	---	1.65	1.38		
ADB(080)	5	Constant + word → word	7.2	6.0	1.35	1.13
		∗DM + ∗DM → ∗DM	11.6	9.6		
SBB(081)	5	Constant – word → word	7.4	6.1		
		∗DM – ∗DM → ∗DM	11.7	9.8		
MLB(082)	5	Constant x word → word	15.8	13.1		
		∗DM x ∗DM → word	20.1	16.8		
DVB(083)	5	Word ÷ constant → word	19.1	15.9		
		∗DM ÷ ∗DM → ∗DM	23.4	19.5		
ADBL(084)	5	Constant + word → word	9.2	7.6		
		∗DM + ∗DM → ∗DM	13.4	11.1		
SBBL(085)	5	Constant – word → word	9.3	7.8		
		∗DM – ∗DM → ∗DM	13.5	11.3		
MLBL(086)	5	Constant x word → word	46.1	38.4		
		∗DM x ∗DM → word	50.3	41.9		
DVBL(087)	5	Word ÷ constant → word	57.3	47.8		
		∗DM ÷ ∗DM → ∗DM	61.5	51.3		
INC(090)	3	Word increment	6.3	5.3	1.05	0.88
		∗DM increment	7.5	6.3		
DEC(091)	3	Word decrement	6.5	5.4		
		∗DM decrement	7.7	6.4		
INCB(092)	3	Word increment	5.6	4.6		
		∗DM increment	6.8	5.6		
DECB(093)	3	Word decrement	5.9	4.9		
		∗DM decrement	7.1	5.9		
INCL(094)	3	Word increment	15.5	12.9		
		∗DM increment	16.7	13.9		
DECL(095)	3	Word decrement	15.3	12.8	1.05	0.875
		∗DM decrement	16.5	13.8		
INBL(096)	3	Word increment	6.8	5.6		
		∗DM increment	8.0	6.6		
DCBL(097)	3	Word decrement	6.9	5.8		
		∗DM decrement	8.0	6.6		
BIN(100)	4	When converting a word to a word	6.2	5.1	1.2	1.0
		When converting ∗DM to ∗DM	8.6	7.1		
BCD(101)	4	When converting a word to a word	6.0	5.0		

*Note: CV500 = CV500 or CVM1-CPU01-EV2; CV1000 = CV1000, CV2000, or CVM1-CPU11/21-EV2

Instruction	Words	Conditions	ON execution time (μs)		OFF execution time (μs)	
			CV500*	CV1000*	CV500*	CV1000*
		When converting ∗DM to ∗DM	8.4	7.0		
BINL(102)	4	When converting a word to a word	10.2	8.5	1.2	1.0
		When converting ∗DM to ∗DM	12.6	10.5		
BCDL(103)	4	When converting a word to a word	11.6	9.6		
		When converting ∗DM to ∗DM	14.0	11.6		
NEG(104)	4	When converting a word to a word	5.7	4.8		
		When converting ∗DM to ∗DM	8.3	6.9		
NEGL(105)	4	When converting a word to a word	7.1	5.9		
		When converting ∗DM to ∗DM	9.5	7.9		
SIGN(106)	4	When expanding a word to a word	6.6	5.5		
		When expanding ∗DM to ∗DM	9.0	7.5		
MLPX(110)	5	When decoding a word to a word	7.2	6.0	1.35	1.13
		When decoding ∗DM to ∗DM	15.6	13.0		
DMPX111)	5	When encoding a word to a word	9.8	8.1		
		When encoding ∗DM to ∗DM	16.2	13.5		
SDEC(112)	5	When decoding a word to a word	317	264		
		When decoding ∗DM to ∗DM	424	353		
ASC(113)	5	Word → word	324	270		
		∗DM → ∗DM	431	359		
BCNT(114)	5	When counting 1 word	412	344		
		When counting 1000 words with ∗DM	20.5 ms	17.1 ms		
LINE(115)	5	When converting word to word using constant specification	14.0	11.6		
		When converting ∗DM to ∗DM with DM specification	18.0	15.0		
COLM(116)	5	When converting word to word using constant specification	25.1	20.9		
		When converting ∗DM to ∗DM with DM specification	29.4	24.5		
HEX(117)	5	Converting 1 digit, word → word	20.6	17.1		
		Converting 4 digits, ∗DM → ∗DM	48.6	40.5		
TTIM(120)	4	Constant for SV	11.1	9.3	R: 9.8 IL: 3.0	R: 8.1 IL: 2.5
		∗DM for SV	12.8	10.6	R: 11.4 IL: 3.0	R: 9.5 IL: 2.5
TIML(121)	5	---	1.71 ms	1.42 ms	R or IL: 1.15	R or IL: 0.96
MTIM(122)	5	---	2.25 ms	1.88 ms	1.35	1.113
TCNT(123)	4	Constant for execution times	11.4	9.5	1.2	1.0
		∗DM for execution times	13.1	10.9		
TSR(124)	4	When reading to a word	6.3	5.3		
		When reading to ∗DM	9.9	8.3		
TSW(125)	4	When writing to a word	5.9	4.9		
		When writing to ∗DM	9.6	8.0		
ANDW(130)	5	Constant AND word → word	6.8	5.6	1.35	1.13
		∗DM AND ∗DM → ∗DM	11.0	9.1		
ORW(131)	5	Constant OR word → word	6.8	5.6		
		∗DM OR ∗DM → ∗DM	11.1	9.3		

*Note: CV500 = CV500 or CVM1-CPU01-EV2; CV1000 = CV1000, CV2000, or CVM1-CPU11/21-EV2

Instruction	Words	Conditions	ON execution time (μs)		OFF execution time (μs)	
			CV500*	CV1000*	CV500*	CV1000*
XORW(132)	5	Constant XOR word → word	6.8	5.6		
		∗DM XOR ∗DM → ∗DM	11.1	9.3		
XNRW(133)	5	Constant XNOR word → word	6.8	5.6	1.35	1.13
		∗DM XNOR ∗DM → ∗DM	11.1	9.3		
ANDL(134)	5	Constant AND word → word	8.7	9.3		
		∗DM AND ∗DM → ∗DM	12.9	10.8		
ORWL(135)	5	Constant OR word → word	8.7	7.3		
		∗DM OR ∗DM → ∗DM	12.9	10.8		
XORL(136)	5	Constant XOR word → word	8.7	7.3		
		∗DM XOR ∗DM → ∗DM	12.9	10.8		
XNRL(137)	5	Constant XNOR word → word	8.7	7.3		
		∗DM XNOR ∗DM → ∗DM	12.9	10.8		
COM(138)	3	When inverting a word	5.6	4.6	1.05	0.88
		When inverting ∗DM	6.8	5.6		
COML(139)	3	When inverting a word	6.6	5.5		
		When inverting ∗DM	7.8	6.5		
ROOT(140)	4	---	777	647	1.2	1.0
FDIV(141)	5	Word ÷ word → word (equals 0)	366	305	1.35	1.13
		Word ÷ word → word (not 0)	1.72 ms	1.43 ms		
		∗DM ÷ ∗DM → ∗DM	1.73 ms	1.45 ms		
APR(142)	5	When specifying sine or cosine	405	338		
		When specifying a word with 256-word table	2.82 ms	2.35 ms		
SEC(143)	4	When converting a word to a word	411	342	1.2	1.0
		When converting ∗DM to ∗DM	573	477		
HMS(144)	4	---	888	740		
CADD(145)	5	---	1.13 ms	0.94 ms	1.35	1.13
CSUB(146)	5	---	1.13 ms	0.94 ms		
SBN(150)	2	---	---	---	---	---
SBS(151)	3	---	3.8	3.1	1.05	0.88
RET(152)	2	---	13.8	11.5	8.6	7.1
MSKS(153)	4	When setting a constant	5.0	4.1	1.2	1.0
		When setting ∗DM	6.8	5.6		
CLI(154)	4	When setting a constant	5.4	4.5		
		When setting ∗DM	6.8	5.6		
MSKR(155)	4	When setting a word	7.4	6.1		
		When setting ∗DM	8.6	7.1		
MCRO(156)	5	Parameter word designation	35.4	29.5	1.35	1.13
		Parameter ∗DM designation	37.7	31.4		
SSET(160)	4	When setting a 3-word stack	15.2	12.6	1.2	1.0
		When setting a 999-word stack	773	644		
PUSH(161)	4	When designating stack through word	3.9	3.3		
		When designating stack through ∗DM	5.7	4.8		
LIFO(162)	4	When designating stack through word	4.1	3.4		
		When designating stack through ∗DM	5.3	4.4		
FIFO(163)	4	When using 3-word stack	14.7	12.3		
		When using 999-word stack	1.25 ms	1.04 ms		

*Note: CV500 = CV500 or CVM1-CPU01-EV2; CV1000 = CV1000, CV2000, or CVM1-CPU11/21-EV2

Instruction	Words	Conditions	ON execution time (μs)		OFF execution time (μs)	
			CV500*	CV1000*	CV500*	CV1000*
SRCH(164)	5	When searching 1 word	347	289	1.35	1.13
		When searching 1000 words for *DM	11.4 ms	9.54 ms		
MAX(165)	5	When searching 1 word	464	386	1.35	1.13
		When searching 1000 words for *DM	68.2 ms	56.9 ms		
MIN(166)	5	When searching 1 word	465	388		
		When searching 1000 words for *DM	68.2 ms	57.1 ms		
SUM(167)	5	When adding 1 word	758	632		
		When adding 1000 words via *DM	173 ms	144 ms		
TRSM(170)	2	When sampling 1 point + 0 word	21.6	18.0		
		When sampling 12 points + 3 words	50.4	42.0		
EMBC(171)	3	When setting a constant	3.6	3.0	1.05	0.88
		When setting *DM	5.4	4.5		
CCL(172)	2	---	2.6	2.1	0.9	0.75
CCS(173)	2	---	2.0	1.6		
MARK(174)	3	When sampling 0 words	17.7	14.8	2.7	2.25
		When sampling 2 words	24.8	20.6		
REGL(175)	3	When setting a word	5.9	4.9	1.05	0.88
		When setting *DM	7.1	5.9		
REGS(176)	3	When setting a word	9.8	8.1		
		When setting *DM	11.0	9.1		
FPD(177)	5	Without message output: execution	930	775	263	219
		Without message output: first time	971	809		
		With message output: execution	1.00 ms	835		
		With message output: first time	1.05 ms	874		
WDT(178)	3	---	8.61	7.18	1.05	0.88
DATE(179)	3	Word designation	310	259		
		*DM designation	315	262		
FILR(180)	5	---	415	406	1.35	1.13
FILW(181)	5	---	423	414		
FILP(182)	5	When reading 10 steps	99.2 ms	99.2 ms	1.05	0.88
		When reading 1000 steps	519 ms	519 ms		
FLSP(183)	4	---	29.4 ms	29.4 ms	1.2	1.0
IORF(184)	4	When refreshing 1 word	27.6	23.6	1.2	1.0
		When refreshing 32 words	325	279		
IOSP(187)	2	---	230	228	0.9	0.75
IORS(188)	2	---	2.4	2.0	IL: 0.9	IL: 0.75
IODP(189)	4	---	369	308	1.2	1.0
READ(190)	5	When reading 1 word	568	473	1.35	1.13
		When reading 255 words	4.64 ms	3.87 ms		
WRIT(191)	5	When writing 1 word	661	551		
		When writing 255 words	4.9 ms	4.1 ms		
SEND(192)	5	---	244	235		
RECV(193)	5	---	251	241		
CMND(194)	5	---	240	230		
MSG(195)	4	When specifying constant and a word	6.2	5.1	1.2	1.0
		When specifying two *DM	9.2	7.6		
TOUT(202)	2	---	4.7	3.9	4.7	3.9
*Note: CV500 = CV500 or CVM1-CPU01-EV2; CV1000 = CV1000, CV2000, or CVM1-CPU11/21-EV2						

Instruction	Words	Conditions	ON execution time (μs)		OFF execution time (μs)	
			CV500*	CV1000*	CV500*	CV1000*
SA(210)	4	When activating 1 step	39.5	32.9	1.2	1.0
		When activating a 15-step subchart	48.2	40.1		
SP(211)	3	When pausing 1 step	25.8	21.5	1.05	0.88
		When pausing a 15-step subchart	64.8	54.0		
SR(212)	3	When releasing 1 step	26.4	22.0		
		When releasing a 15-step subchart	65.4	54.5		
SF(213)	3	When stopping 1 step	25.7	21.4		
		When stopping a 15-step subchart	62.4	52.0		
SE(214)	3	When inactivating 1 step	609	507		
		When inactivating a 15-step subchart	810	675		
SOFF(215)	3	When resetting 1 step	621	518		
		When resetting a 15-step subchart	2.2 ms	1.9 ms		
CJP(221)	3	Jumping to designated word	10.5	8.75	2.85	2.38
		Jumping to designated *DM	11.9	9.88		
CJPN(222)	3	Jumping to designated word	10.5	8.75	2.70	2.25
		Jumping to designated *DM	11.9	9.88		
CNR(236)	4	When resetting 1 word	20.3	16.9	1.2	1.0
		When resetting 1000 words via *DM	623	519		
BPRG(250)	3	---	5.85	4.88	3.00	2.50
RLNC(260)	3	Rotating word	11.6	9.63	1.05	0.88
		Rotating *DM	14.1	11.8		
RRNC(261)	3	Rotating word	11.7	9.75		
		Rotating *DM	14.3	11.9		
RLNL(262)	3	Rotating word	12.9	10.8		
		Rotating *DM	15.5	12.9		
RLNL(263)	3	Rotating word	12.9	10.8		
		Rotating *DM	15.5	12.9		
PID(270)	5	When not sampling	22.7	18.9	5.40	4.50
		When sampling	368	306		
		When executing first time	807	673		
LMT(271)	5	Input, output word designation	17.3	14.3	1.35	1.13
		Input, output *DM designation	25.1	20.9		
BAND(272)	5	Input, output word designation	17.3	14.4		
		Input, output *DM designation	25.2	21.0		
ZONE(273)	5	Input, output word designation	17.6	14.6		
		Input, output *DM designation	25.5	21.3		
ROTB(274)	3	Word → word	41.6	34.6	1.20	1.00
		*DM → *DM	46.8	39.0		
BINS(275)	5	Word → word	16.1	13.4	1.35	1.13
		*DM → *DM	25.1	20.9		
BCDS(276)	5	Word → word	16.2	13.5		
		*DM → *DM	24.3	20.3		
BISL(277)	5	Word → word	21.3	17.8		
		*DM → *DM	30.5	25.4		
BDSL(278)	5	Word → word	23.3	19.4		
		*DM → *DM	31.4	26.1		
RD2(280)	5	Reading 1 word to word	20.3	16.9	5.25	4.38

*Note: CV500 = CV500 or CVM1-CPU01-EV2; CV1000 = CV1000, CV2000, or CVM1-CPU11/21-EV2

Instruction	Words	Conditions	ON execution time (μs)		OFF execution time (μs)	
			CV500*	CV1000*	CV500*	CV1000*
		Reading 255 words *DM to *DM	26.0	21.6		
WR2(281)	5	Writing 1 word to word	21.3	17.8	5.40	4.50
		Writing 255 words *DM to *DM	27.0	22.5		
=(300)	5	Comparing constant and word	14.7	12.3	6.15	5.13
		Comparing *DM and *DM	20.1	16.8		
=L(301)	5	Comparing constant and word	15.6	13.0		
		Comparing *DM and *DM	21.0	17.5		
=S(302)	5	Comparing constant and word	15.0	12.5		
		Comparing *DM and *DM	20.4	17.0		
=SL(303)	5	Comparing constant and word	15.9	13.3		
		Comparing *DM and *DM	21.3	17.8		
<>(305)	5	Comparing constant and word	14.7	12.3		
		Comparing *DM and *DM	20.1	16.8		
<>L(306)	5	Comparing constant and word	15.6	13.0		
		Comparing *DM and *DM	21.0	17.5		
<>S(307)	5	Comparing constant and word	15.0	12.5		
		Comparing *DM and *DM	20.4	17.0		
<>SL(308)	5	Comparing constant and word	15.9	13.3		
		Comparing *DM and *DM	21.3	17.8		
<(310)	5	Comparing constant and word	14.7	12.3		
		Comparing *DM and *DM	20.1	16.8		
<L(311)	5	Comparing constant and word	15.6	13.0		
		Comparing *DM and *DM	21.0	17.5		
<S(312)	5	Comparing constant and word	15.0	12.5		
		Comparing *DM and *DM	20.4	17.0		
<SL(313)	5	Comparing constant and word	15.9	13.3		
		Comparing *DM and *DM	21.3	17.8		
<=(315)	5	Comparing constant and word	14.7	12.3		
		Comparing *DM and *DM	20.1	16.8		
<=L(316)	5	Comparing constant and word	15.6	13.0		
		Comparing *DM and *DM	21.0	17.5		
<=S(317)	5	Comparing constant and word	15.0	12.5		
		Comparing *DM and *DM	20.4	17.0		
<=SL(318)	5	Comparing constant and word	15.9	13.3		
		Comparing *DM and *DM	21.3	17.8		
>(320)	5	Comparing constant and word	14.7	12.3		
		Comparing *DM and *DM	20.1	16.8		
>L(321)	5	Comparing constant and word	15.6	13.0		
		Comparing *DM and *DM	21.0	17.5		
>S(322)	5	Comparing constant and word	15.0	12.5		
		Comparing *DM and *DM	20.4	17.0		
>SL(323)	5	Comparing constant and word	15.9	13.3		
		Comparing *DM and *DM	21.3	17.8		
>=(325)	5	Comparing constant and word	14.7	12.3		
		Comparing *DM and *DM	20.1	16.8		
>=(326)	5	Comparing constant and word	15.6	13.0		
		Comparing *DM and *DM	21.0	17.5		

*Note: CV500 = CV500 or CVM1-CPU01-EV2; CV1000 = CV1000, CV2000, or CVM1-CPU11/21-EV2

Instruction	Words	Conditions	ON execution time (μs)		OFF execution time (μs)	
			CV500*	CV1000*	CV500*	CV1000*
>=S(327)	5	Comparing constant and word	15.0	12.5	6.15	5.13
		Comparing ∗DM and ∗DM	20.4	17.0		
>=SL(328)	5	Comparing constant and word	15.9	13.3		
		Comparing ∗DM and ∗DM	21.3	17.8		
TST(350)	5	Designating constant for word bit	14.6	12.1		
		Designating constant for ∗DM bit	20.4	17.0		
TSTN(351)	5	Designating constant for word bit	14.4	12.0		
		Designating constant for ∗DM bit	20.3	16.9		
+(400)	5	Constant + word → word	16.4	13.6	1.35	1.13
		∗DM + ∗DM → ∗DM	24.0	20.0		
+L(401)	5	Constant + word → word	18.5	15.4		
		∗DM + ∗DM → ∗DM	26.4	22.0		
+C(402)	5	Constant + word → word	8.40	7.00		
		∗DM + ∗DM → ∗DM	12.6	10.5		
+CL(403)	5	Constant + word → word	10.4	8.63		
		∗DM + ∗DM → ∗DM	14.9	12.4		
+B(404)	5	Constant + word → word	16.8	14.0		
		∗DM + ∗DM → ∗DM	25.7	21.4		
+BL(405)	5	Constant + word → word	29.3	24.4		
		∗DM + ∗DM → ∗DM	38.0	31.6		
+BC(406)	5	Constant + word → word	9.00	7.50		
		∗DM + ∗DM → ∗DM	13.8	11.5		
+BCL(407)	5	Constant + word → word	21.5	17.9		
		∗DM + ∗DM → ∗DM	26.1	21.8		
–(410)	5	Constant – word → word	16.7	13.9		
		∗DM – ∗DM → ∗DM	24.6	20.5		
–L(411)	5	Constant – word → word	18.3	15.3		
		∗DM – ∗DM → ∗DM	26.3	21.9		
–C(412)	5	Constant – word → word	8.25	6.88		
		∗DM – ∗DM → ∗DM	13.4	11.1		
–CL(413)	5	Constant – word → word	10.4	8.63		
		∗DM – ∗DM → ∗DM	14.6	12.1		
–B(414)	5	Constant – word → word	16.5	13.8		
		∗DM – ∗DM → ∗DM	25.4	21.1		
–BL(415)	5	Constant – word → word	28.7	23.9		
		∗DM – ∗DM → ∗DM	36.9	30.8		
–BC(416)	5	Constant – word → word	8.85	7.38		
		∗DM – ∗DM → ∗DM	13.7	11.4		
–BCL(417)	5	Constant – word → word	21.0	17.5		
		∗DM – ∗DM → ∗DM	25.2	21.0		
∗(420)	5	Constant x word → word	24.9	20.8		
		∗DM x ∗DM → ∗DM	34.4	28.6		
∗L(421)	5	Constant x word → word	55.4	46.1		
		∗DM x ∗DM → ∗DM	66.6	55.5		
∗U(422)	5	Constant x word → word	16.2	13.5		
		∗DM x ∗DM → ∗DM	20.9	17.4		

*Note: CV500 = CV500 or CVM1-CPU01-EV2; CV1000 = CV1000, CV2000, or CVM1-CPU11/21-EV2

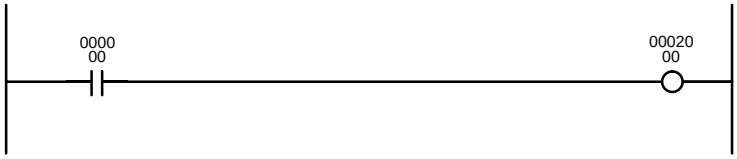
Instruction	Words	Conditions	ON execution time (μs)		OFF execution time (μs)	
			CV500*	CV1000*	CV500*	CV1000*
*UL(423)	5	Constant x word → word	46.8	39.0	1.35	1.13
		*DM x *DM → *DM	54.0	45.0		
*B(424)	5	Constant x word → word	22.1	18.4		
		*DM x *DM → *DM	27.0	22.5		
*BL(425)	5	Constant x word → word	64.2	53.5		
		*DM x *DM → *DM	72.2	60.1		
/(430)	5	Constant □ word → word	27.8	23.1		
		*DM □ *DM → *DM	35.9	29.9		
/L(431)	5	Constant □ word → word	69.6	58.0		
		*DM □ *DM → *DM	77.6	64.6		
/U(432)	5	Constant □ word → word	18.9	15.8		
		*DM □ *DM → *DM	22.5	18.8		
/UL(433)	5	Constant □ word → word	59.7	49.8		
		*DM □ *DM → *DM	63.6	53.0		
/B(434)	5	Constant □ word → word	20.7	17.3		
		*DM □ *DM → *DM	24.9	20.8		
/BL(435)	5	Constant □ word → word	75.2	62.6		
		*DM □ *DM → *DM	79.1	65.9		
FIX(450)	5	Word → word	335	279	1.2	1.0
		*DM → *DM	336	280		
FIXL(451)	5	Word → word	413	344		
		*DM → *DM	413	344		
FLT(452)	4	Word → word	327	272		
		*DM → *DM	332	277		
FLTL(453)	5	Word → word	402	335		
		*DM → *DM	408	340		
+F(454)	5	Constant + word → word	421	351	1.35	1.13
		*DM + *DM → *DM	429	358		
–F(455)	5	Constant – word → word	449	374		
		*DM – *DM → *DM	457	381		
*F(456)	5	Constant x word → word	436	363		
		*DM x *DM → *DM	444	370		
/F(457)	5	Constant □ word → word	481	401		
		*DM □ *DM → *DM	490	408		
RAD(458)	5	Word → word	438	365	1.2	1.0
DEG(459)	5	Word → word	438	365		
SIN(460)	5	Word → word	2.80 ms	2.34 ms		
COS(461)	5	Word → word	2.68 ms	2.23 ms		
TAN(462)	5	Word → word	4.31 ms	3.59 ms		
ASIN(463)	5	Word → word	4.51 ms	3.76 ms		
ACOS(464)	5	Word → word	4.61 ms	3.85 ms		
ATAN(465)	5	Word → word	2.18 ms	1.82 ms		
SQRT(466)	5	Word → word	890	742		
EXP(467)	5	Word → word	4.47 ms	3.72 ms		
LOG(468)	5	Word → word	2.25 ms	1.87 ms		
BEND<001>	2	---	4.65	3.88	2.85	2.38
*Note: CV500 = CV500 or CVM1-CPU01-EV2; CV1000 = CV1000, CV2000, or CVM1-CPU11/21-EV2						

Instruction	Words	Conditions	ON execution time (μs)		OFF execution time (μs)	
			CV500*	CV1000*	CV500*	CV1000*
IF<002>	3	Without operand	4.20	3.50	2.70	2.25
		With operand	8.10	6.75		
ELSE<003>	2	---	4.35	3.63		
IEND<004>	2	---	4.50	3.75		
WAIT<005>	3	Without operand	4.35	3.63		
		With operand	8.70	7.25		
EXIT<006>	3	Without operand	4.95	4.13		
		With operand	8.85	7.38		
LOOP<009>	2	---	4.20	3.50	2.55	2.13
LEND<010>	3	Without operand	4.05	3.38	2.70	2.25
		With operand	7.95	6.63		
BPPS<011>	3	---	4.95	4.13	3.00	2.50
BPRS<012>	3	---	4.05	3.38		
TIMW<013>	4	Initial startup	24.6	20.5	3.45	2.88
		Normal execution	20.0	16.6		
CNTW<014>	5	Initial startup	21.5	17.9	3.30	2.75
		Normal execution	21.2	17.6		
TMHW<015>	4	Initial startup	24.8	20.6	3.45	2.88
		Normal execution	20.0	16.6		
*Note: CV500 = CV500 or CVM1-CPU01-EV2; CV1000 = CV1000, CV2000, or CVM1-CPU11/21-EV2						

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

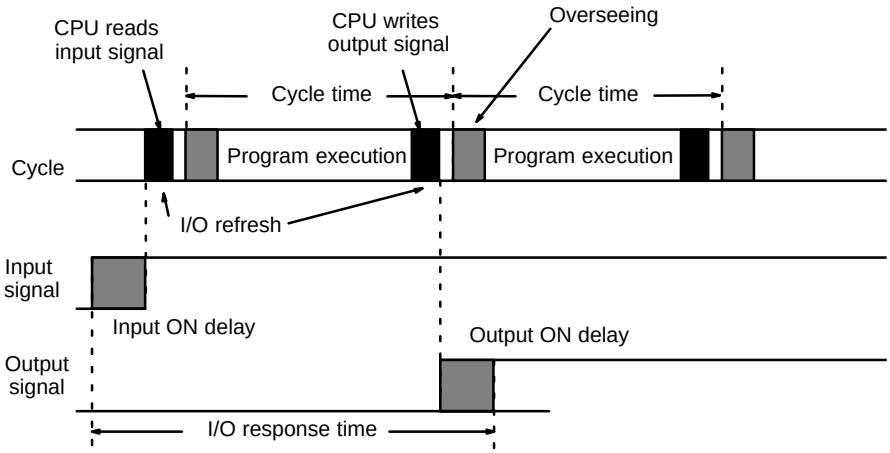
Here, we'll compute the minimum and maximum I/O response times for a CV1000 set for cyclic refreshing. The PC controls only I/O Units, mounted on the CPU Rack, Expansion CPU Rack, or Expansion I/O Rack. The calculation is identical for asynchronous and synchronous operation.

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

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.

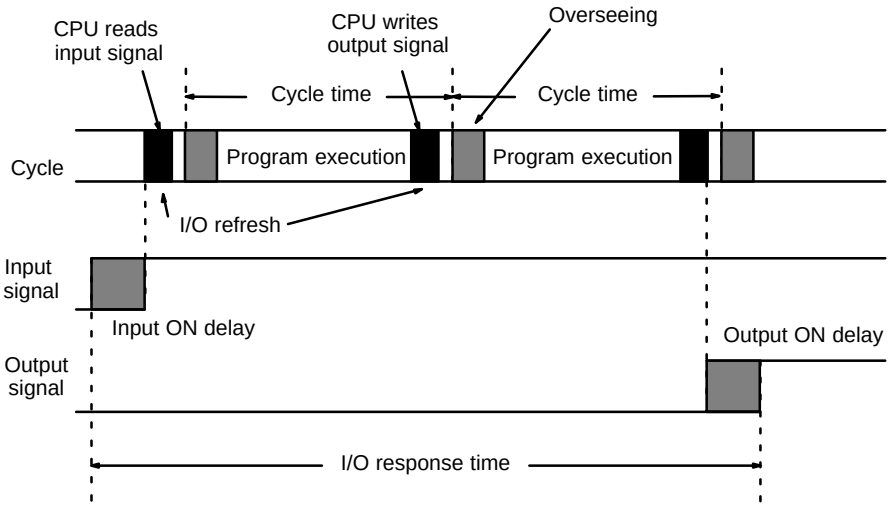


Minimum I/O response time = input ON delay + cycle time + output ON delay

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 Asynchronous Operation with a SYSMAC BUS System

Here, we'll compute the minimum and maximum I/O response times for a CV1000 that is set for asynchronous operation and controls a SYSMAC BUS System with a single Master. Both the input and output are on I/O Units connected to Slave Racks. In asynchronous operation, SYSMAC BUS refreshing occurs every 5×n ms, where n is the number of Masters mounted, and can occur at any point of the instruction execution cycle. In this case only one Master is involved, so refreshing occurs every 5 ms.

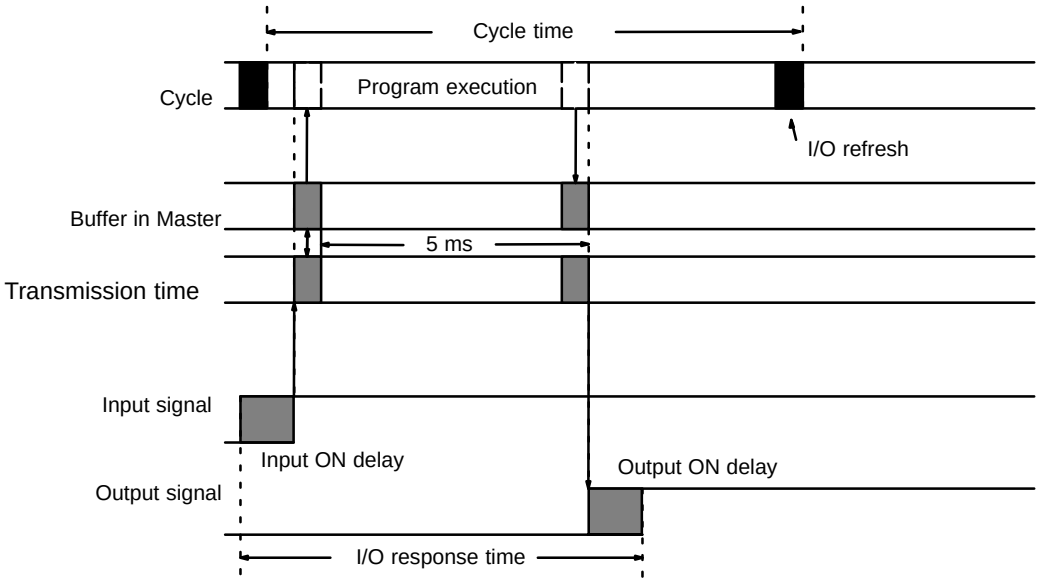
The transmission time for a Master is the sum total of the transmission times for all Slaves connected to it. The transmission time for Slave Racks is 1.4 + (0.2×*a*) ms, where a is the number of I/O words on the Slave. The transmission time for I/O Terminals is 2×b ms, where b is the number of I/O words 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 + (0.2×4) = 2.2 ms
I/O Terminal transmission time	2×3 = 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 instruction that uses the input signal is executed between two SYSMAC BUS refreshes. This situation is illustrated below.



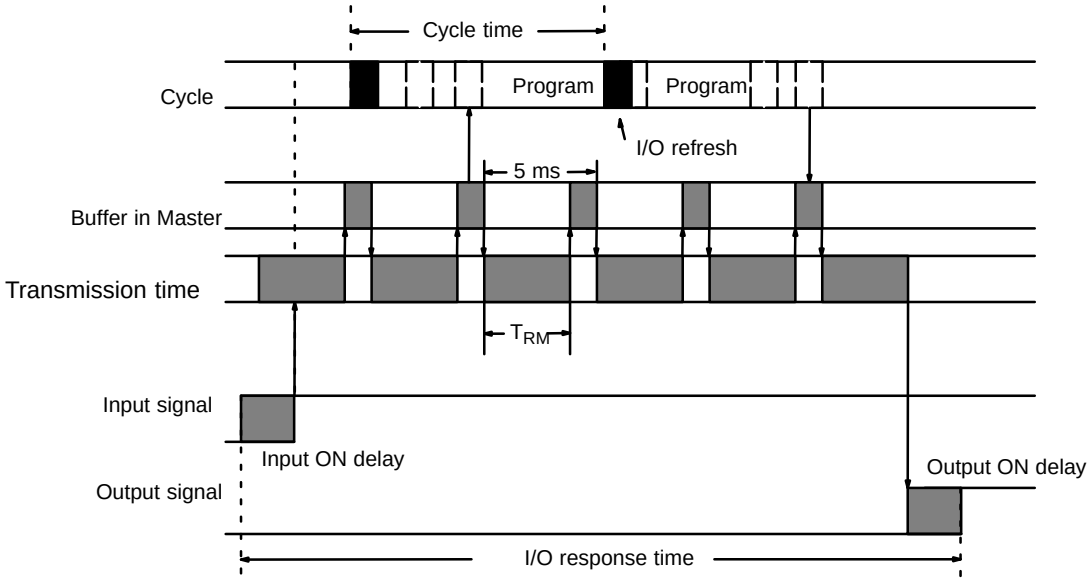
Minimum I/O response time = Input ON delay +
Refreshing interval (5 ms × n)+ Master transmission time
+ Output ON delay

Where, n = Number of SYSMAC BUS Remote I/O Master Units

Minimum I/O response time = 1.5 + 5 + 8.2 +15 = 29.7 ms

Maximum I/O Response Time

The PC takes longest to respond when the instruction that uses the input signal is executed just before SYSMAC BUS refreshing, so the instruction won't be executed with the new input bit status until the next cycle. This situation is illustrated below.



Maximum I/O response time = Input ON delay + (Cycle time + 10 ms × n) +
+ Master transmission time × 2 + Slave transmission time + Output ON delay

Where, n = Number of SYSMAC BUS Remote I/O Master Units

Master transmission time = Total refresh time for all Slaves controlled =
Sum of all Slave transmission times

Slave transmission time = 1.4 + (0.2 ms × Number of Slave I/O words)

Maximum I/O response time = 1.5 + (20 + 10) + (8.2 × 2) + 2.2 + 15 = 65.1 ms

6-5-3 Synchronous Operation with a SYSMAC BUS System

Here, we'll compute the minimum and maximum I/O response times for a CV1000 that is set for synchronous operation and controls a SYSMAC BUS System. Both the input and output are on I/O Units connected to Slave Racks. In synchronous operation, SYSMAC BUS refreshing is carried out just after I/O refreshing as one phase of a single PC cycle.

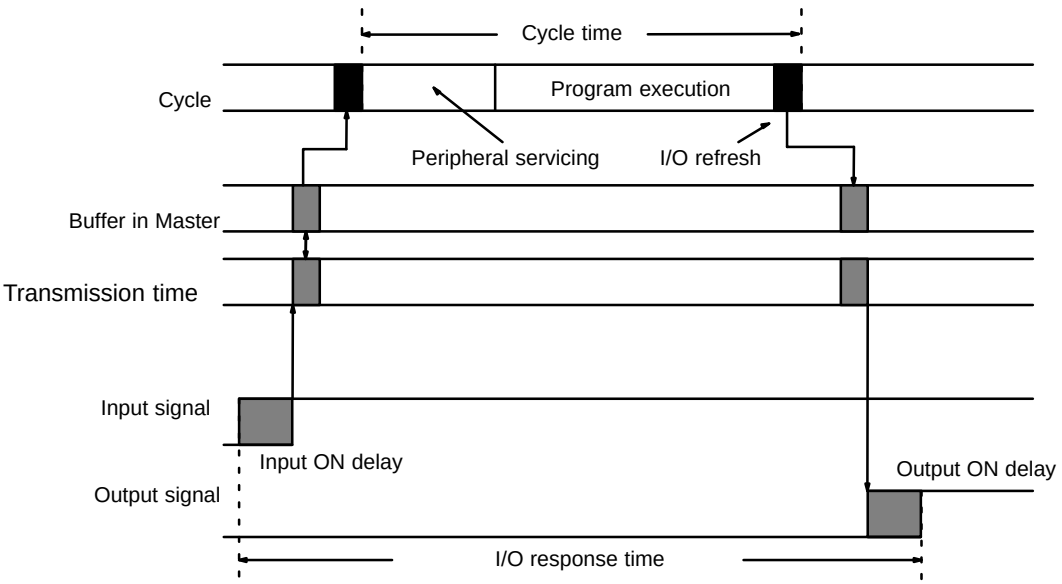
The transmission time for a Master is the sum total of the transmission times for all Slaves connected to it. The transmission time for Slave Racks is 1.4 + (0.2×*a*) ms, where *a* is the number of I/O words on the Slave. The transmission time for I/O Terminals is 2×*b* ms, where *b* is the number of 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 + (0.2×4) = 2.2 ms
I/O Terminal transmission time	2×3 = 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.

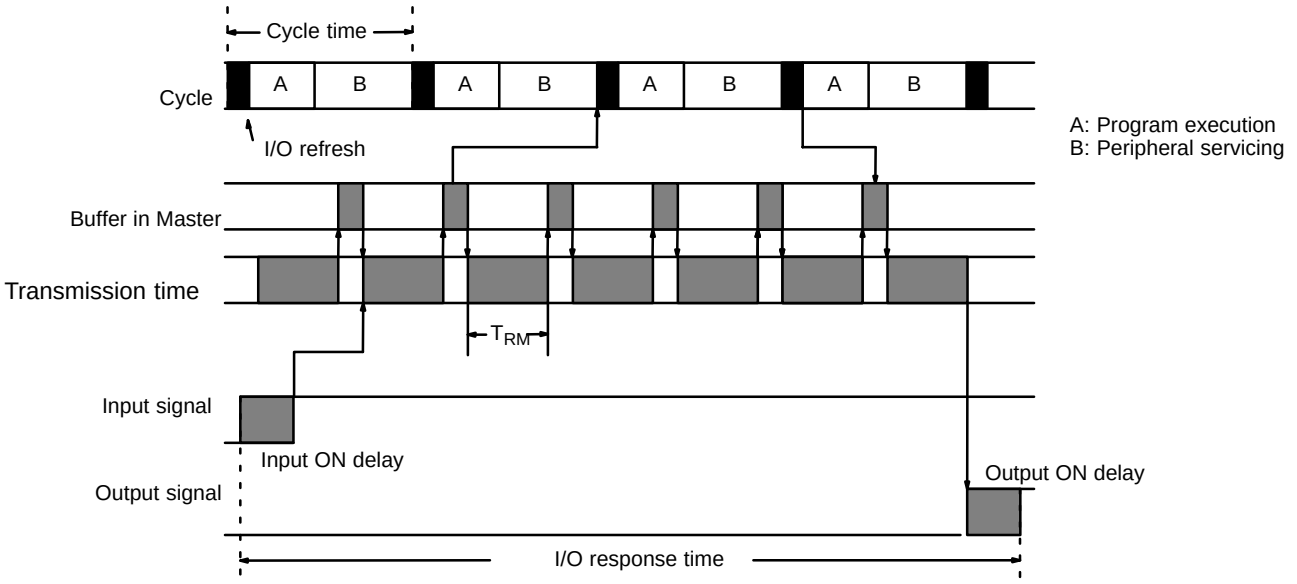


Minimum I/O response time = input ON delay + cycle time
+ Slave transmission time x 2 + output ON delay

Minimum I/O response time = 1.5 + 20 + 2.2×2 +15 = 40.9 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.



Maximum I/O response time = input ON delay + cycle time x 2
+ Master transmission time x 2 + Slave transmission time x 2
+ output ON delay

Maximum I/O response time = 1.5 + (20×2) +(8.2×2) +(2.2×2) +15 = 77.3 ms

6-5-4 Asynchronous Operation with a SYSMAC BUS/2 System

Here, we'll compute the minimum and maximum I/O response times for a CV1000 that is set for asynchronous operation and controls a SYSMAC BUS/2 System. Both the input and output are on I/O Units connected to Slave Racks. In asynchronous operation, SYSMAC BUS/2 refreshing occurs at the end of the SYSMAC BUS/2 communications cycle.

This calculation only applies when the SYSMAC BUS/2 Master is the only CPU Bus Unit connected to the CPU. If other CPU Bus Units are connected, add the following delay to the maximum I/O response time calculated later in this section: (other Unit's refreshing time +1.5 ms)×(number of CPU Bus Units connected)

If a higher priority peripheral process such as a SEND(192), RECV(193), or FAL(006) instruction occurs, it will be processed before the SYSMAC BUS/2 servicing, increasing the I/O response time.

The remote refresh time is 2.0 + (0.2×*a*) ms, where *a* is the number of words to refresh. The communications cycle time is 5 ms or the sum total of the communications times of Slaves connected to the Master. The communications cycle time can also be set in the SYSMAC BUS/2 settings in the PC Setup, refer to the *SYSMAC BUS/2 Remote I/O System Manual* for details. The following table lists the communications times of the various Slaves.

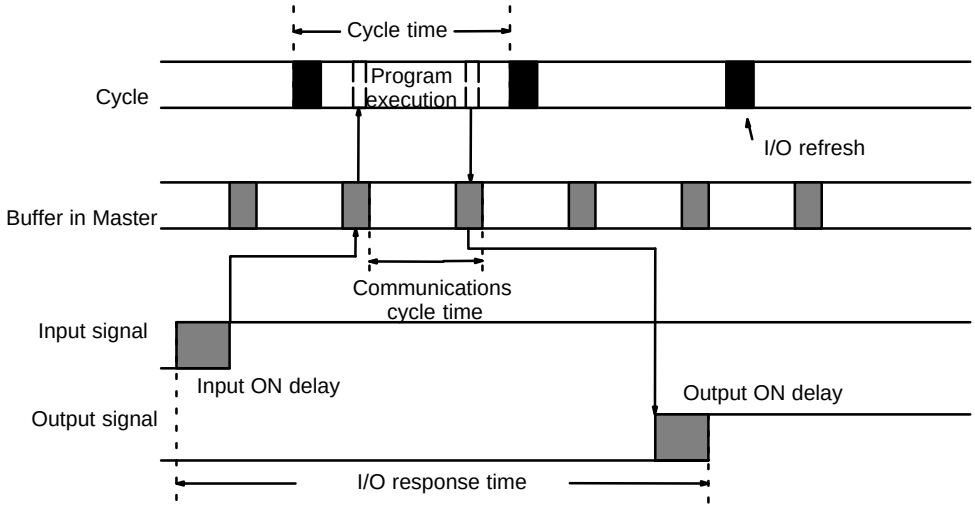
Slave		Transmission type	Communications time
Group 1		Wired	0.16 ms
		Optical	0.32 ms
Group 2		Wired	0.31 ms
		Optical	0.47 ms
Group 3	58M	Wired	1.25 ms
		Optical	1.89 ms
	54MH	Wired	2.5 ms
		Optical	4.5 ms
	122M	Wired	2.5 ms
		Optical	4.5 ms

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)
Remote refresh time	Approx. 2 ms

Minimum I/O Response Time

The PC responds most quickly when the Master receives the input signal just prior to SYSMAC BUS/2 refreshing and the relevant instruction is executed within the communications cycle time. This situation is illustrated below.

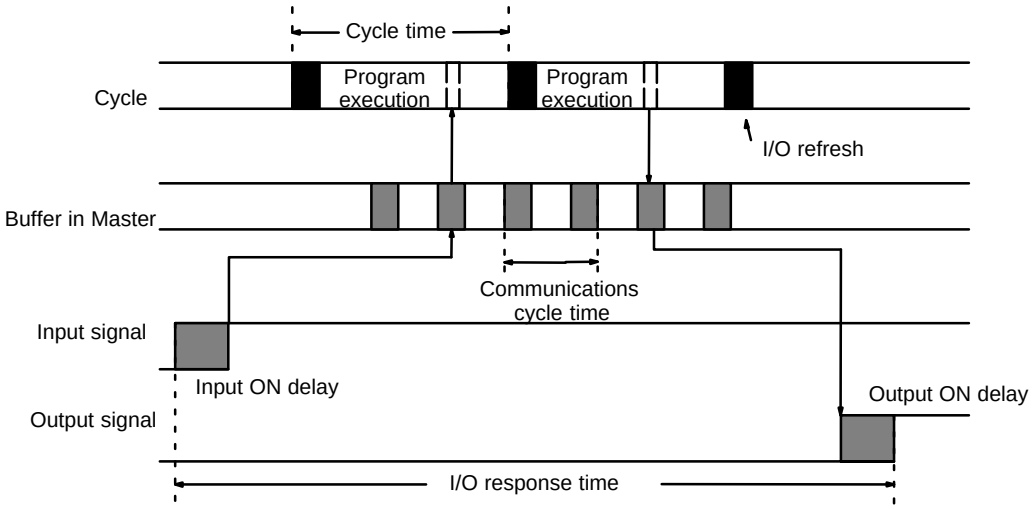


Minimum I/O response time = input ON delay
+ communications cycle time × 6 + remote refresh time
+ output ON delay

Minimum I/O response time = 1.5 + (5×6) + 2 +15 = 48.5 ms

Maximum I/O Response Time

The PC takes longest to respond when the relevant instruction is executed just prior to SYSMAC BUS/2 refreshing. In this case the CPU does not execute the instruction with the new input bit status until the next cycle. This situation is illustrated below.



Maximum I/O response time = input ON delay
+ communications cycle time × 8 + cycle time + remote refresh time
+ 15 ms + output ON delay

Maximum I/O response time = 1.5 + (5×8) + 20 +2 + 15 + 15 = 93.5 ms

6-5-5 Synchronous Operation with a SYSMAC BUS/2 System

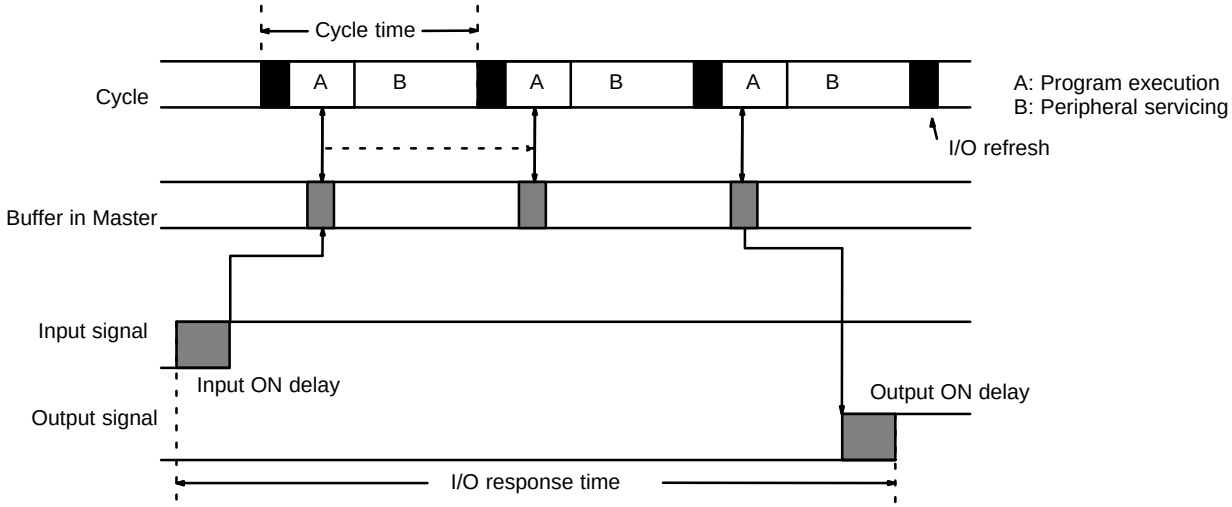
Here, we'll compute the minimum and maximum I/O response times for a CV1000 that is set for synchronous operation and controls a SYSMAC BUS/2 System. Both the input and output are on Units connected to Slave Racks. The PC receives data from the Master once each cycle. The Master waits to transfer to 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 SYSMAC BUS/2 refreshing.

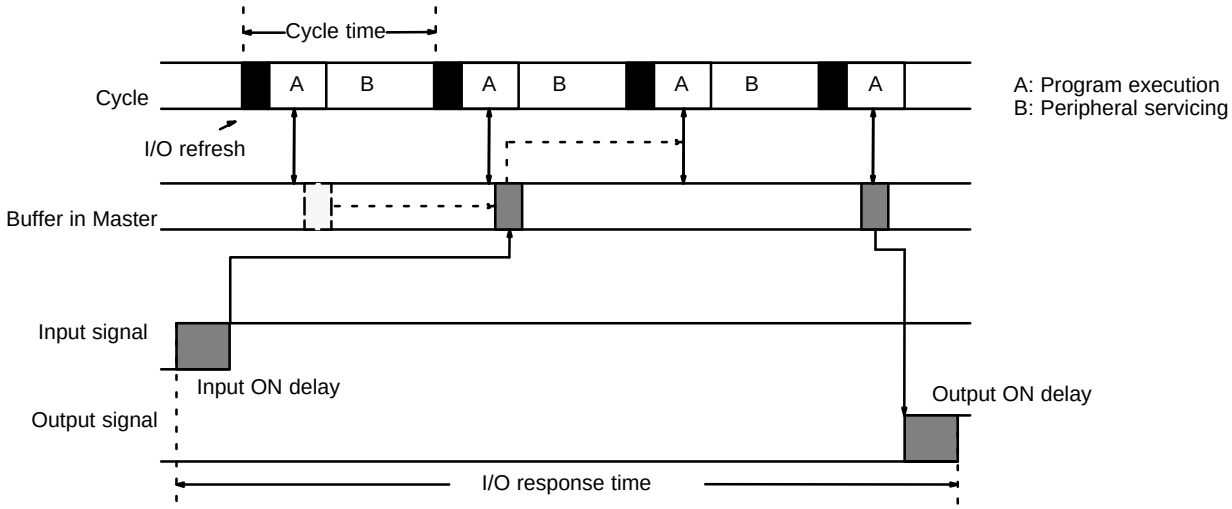


Minimum I/O response time = input ON delay
+ communications cycle time × 5+ cycle time x 2
+ output ON delay

Minimum I/O response time = 1.5 + (5×5) + (20×2) +15 = 81.5 ms

Maximum I/O Response Time

The PC takes longest to respond when it receives the input signal just after SYSMAC BUS/2 refreshing. In this case the CPU does not recognize the input signal until the next cycle. This situation is illustrated below.



Maximum I/O response time = input ON delay
+ communications cycle time × 7 + cycle time × 3
+ 10 ms + output ON delay

Maximum I/O response time = 1.5 + (5×7) + (20×3) +10 +15 = 121.5 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 CVSS/SSS. Refer to *CVSS/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 CV-series manuals.

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 300.
	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)	Maintains the status of the Restart Continuation Bit when power is turned off and on.	These parameters must be set to YES when using restart continuation.
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		Sets the refresh method to cyclic, zero-cross, or scheduled.	To reduce the cycle time by using immediate refreshing or to reduce surge voltages for AC outputs.
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	Specifies if momentary power interruptions are to be treated as errors.	To generate an error for momentary power interrupts when they adversely affect system operation.
	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	Specifies whether Peripheral Devices are to be serviced synchronously or asynchronously with program execution.	To increase processing capacity (speed) by using asynchronous processing.
	I:I/O interrupt	Specifies whether higher-priority I/O interrupts are to be executed before a current I/O interrupt.	
	D:Power OFF interrupt	Specifies whether a power off interrupt is to be executed.	To save system status when power turns off.
	A:Dup action process	Specifies whether an error is to be generated when the same action is executed simultaneously from two different locations in the program.	
	T:Step timer	Sets the units for the step timer to 0.1 or to 1 s.	
	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.

Parameter	Function	Normal application(s)
J:Scheduled interrupt	Sets the unit for setting the scheduled interrupt to 10.0, 1.0, or 0.5 ms.	
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.
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 then 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	Sets the length of time to be treated as a momentary power interruption.	To enable ignoring short primary voltage drops for poor power supplies.
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 or 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)

Name		Operation
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)	Specify whether the status of the Restart Continuation Bit is to be maintained or reset to OFF when power is turned on. If A00011 is reset, restart continuation won't be performed when power is turned on. Changes to this setting are effective the next time the power is turned ON. (Default: A00011 turned OFF) The following settings are required to continue operation after a power interruption: Restart Continuation Bit (A00011): ON and maintained IOM Hold Bit (A00012): ON and maintained Startup mode: RUN or MONITOR Power OFF interrupt program: Exists
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)
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		Designate the I/O refresh method as cyclic, zero-cross, scheduled, or immediate. Cyclic refreshing occurs once each cycle at the end of program execution. Zero-cross refreshing occurs each time the AC power supply voltage crosses zero. Set this method to more accurately turn off output devices when using AC power supplies. Scheduled refreshing occurs at a specific timer interval. The scheduled refresh interval must also be set. Set the execution interval between 10 and 120 ms. Scheduled refreshing cannot be used if the PC is set for synchronous operation. Immediate refreshing occurs when certain instructions are set to interrupt for refreshing from the user program. To set immediate refreshing, specify scheduled refreshing with a refresh interval of 00 ms. If this is done, I/O status will be refreshed only when instructions in the user program call for it. Changes to this setting are effective immediately. (Default: Cyclic)

Name		Operation
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	Designate whether a momentary power interruption is ignored or treated as a non-fatal error. If momentary power interrupts are treated as errors, they will be recorded in the error log (see setting F). Changes to this setting are effective immediately. (Default: Not an error)
	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	Designate whether instruction execution and Peripheral Device servicing are to be carried out synchronously or asynchronously. If synchronous execution is used, Peripheral Device access to IOM can be disabled during user program execution. Changes to this setting are effective the next time the power is turned ON. (Default: Asynchronous)
	I/O interrupts	Designate whether or not I/O interrupt program execution is interrupted for higher-priority I/O interrupts. The I/O interrupt program with the lowest input number has highest priority. Power OFF interrupts, power ON interrupts, and scheduled interrupts take priority over I/O interrupts regardless of this setting. Changes to this setting are effective immediately. (Default: Do not interrupt lower-priority I/O interrupts.)
	Power OFF interrupt	Designate whether or not power OFF interrupts are generated. If an interrupt is generated, the power OFF interrupt program will be executed. Changes to this setting are effective immediately. (Default: No power OFF interrupt)
	Dup action process	Not used.
	Step timer	Designate whether the step timer is set in increments of 0.1 s or 1.0 s. Changes to this setting are effective immediately. (Default: 0.1 s)
	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 10 ms. Changes to this setting are effective immediately. (Default: Don't use CPU bus link)
J:Scheduled interrupt interval		Designate whether the scheduled interrupt time is set in increments of 10.0 ms, 1.0 ms, or 0.5 ms. Changes to this setting are effective the next time the power is turned ON. (Default: 10 ms)
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.

Name	Operation
P:Power break	Designate the momentary power interruption time between 0 and 9 ms. Operation will continue for momentary power interruptions if the power supply is restored within this time after a power interruption. If the momentary power interruption time is greater than 0 ms, Peripheral Device and Host Link communications may be disrupted and may go on standby for momentary power interruptions. This setting will be ignored and the default value will be used if a C500 Expansion I/O Rack is connected to the System. Changes to this setting are effective immediately. (Default: 0 ms)
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. If an interrupt program is executed, the actual cycle time might be extended by the additional time it takes to execute the interrupt program. (Default: Variable cycle)
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	
	D:Power on flag	
C:Startup mode		PROGRAM
D:Startup processing		Don't transfer program.
E:I/O refresh		Cyclic refreshing
F:Execute control 1	B:Detect low battery	Detect
	S:Error on power off	Fatal
	T:CPU standby	CPU waits
	K:Measure CPU SIOU cycle	Don't measure cycle.

Parameter		Default value
G:Execute control 2	C:Execute process	Asynchronous
	I:I/O interrupt	Nesting
	D:Power OFF interrupt	Disable
	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		10.0 ms
K:1st Rack addr (First words for local racks)		0 for CPU Rack
L:Group 1,2 1st addr (First words for SYSMAC BUS/2 Slaves)		RM0 RM1 RM2 RM3 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)		RM0 RM1 RM2 RM3 CIO 2300 CIO2332 CIO 2364 CIO2396 RM4 RM5 RM6 RM7 CIO 2428 CIO 2460 CIO 2492 CIO 2524
N:Group 3, RT 1st addr (First words for group-3 Slave Racks)		Group 3 (SYSMAC BUS/2): RM0 RM1 RM2 RM3 CIO 0300 CIO 0500 CIO 0700 CIO 0900 Words allocated to Units in order under each Master. 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.
O:CV–SIOU 1st addr		Not used at present.
P:Power break (Momentary power interruption time)		0 ms
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

SECTION 8
Error Processing

This section provides information on hardware and software errors that occur during PC operation. Program input and program syntax errors are described in the *CVSS/SSS Operation Manuals*. Although described in *Section 3 Memory Areas*, flags and other error information provided in the Auxiliary Area are listed in *8-5 Error Flags*.

8-1 Alarm Indicators

There are two indicators on the front of the CPU that provide visual indication of an abnormality in the PC. The error indicator (ERROR) indicates fatal errors (i.e., ones that will stop PC operation); the alarm indicator (ALARM) indicates non-fatal ones. These indicators are shown in *2-1-1 Indicators*.

⚠ Caution The PC will turn ON the error indicator (ERROR), stop program execution, and turn OFF all outputs from the PC for most hardware errors, for certain fatal software errors, or when FALS(007) is executed in the program (see tables on following pages). PC operation will continue for all other errors. It is the user's responsibility to take adequate measures to ensure that a hazardous situation will not result from automatic system shutdown for fatal errors and to ensure that proper actions are taken for errors for which the system is not automatically shut down. System flags and other system and/or user-programmed error indications can be used to program proper actions.

8-2 Programmed Alarms and Error Messages

FAL(006) and FALS(007) can be used in the program to provide user-programmed information on error conditions. With these instructions, the user can tailor error diagnosis to aid in troubleshooting.

FAL(006) and FALS(007) share FAL numbers 001 to 511. If two instructions use the same FAL number, the instruction executed later will not be recognized.

Executing FAL(006) will not stop PC operation or directly affect any outputs from the PC. Executing FALS(007) will stop PC operation and will cause all outputs from the PC to be turned OFF.

It is possible to program the FAL(006) and FALS(007) instructions to output a message when executed. The use of these instructions is described in detail in *Section 5 Instruction Set*.

8-3 Reading and Clearing Errors and Messages

Errors should be cleared promptly by the displaying and clearing errors operation with CVSS/SSS. The cause of fatal errors must be corrected in PROGRAM mode before clearing the error and resuming operation. FAL errors can also be cleared using FAL(006); refer to *5-27-1 FAILURE ALARM – FAL(006)* for details.

Errors can also be cleared by turning the PC power off and on, or switching from PROGRAM to RUN, MONITOR, or DEBUG modes. I/O bus errors, however, should be cleared with CVSS/SSS.

8-4 Error Messages

There are basically three types of errors for which messages are displayed: initialization errors, non-fatal operating errors, and fatal operating errors.

The type of error can be quickly determined from the indicators on the CPU, as described below for the three types of errors. If the status of an indicator is not mentioned in the description, it makes no difference whether it is lit or not.

If an error has an error code, that code will be output to A400 when the error occurs. If more than one error occurs simultaneously, the code of the highest priority error will be output to A400. Errors are listed in order of their priority in the following tables, with the highest priority errors listed first. Fatal errors have higher priority than non-fatal errors.

The Error Log contains a record the last 100 errors and can be expanded in the PC Setup to record up to 2,047 errors. Each record stores the error code, the error contents (for example the SFC error code for an SFC error), and the date and time that the error occurred. Refer to *3-6-15 Error Log Area* for details.

After eliminating the cause of an error, clear the error message from memory before resuming operation.

8-4-1 Initialization Errors

The following errors occur before program execution has been started. The POWER indicator will be lit and the RUN indicator will not be lit for any of these. The RUN output will be OFF for each of these errors. The alarm indicator (ALARM) will be ON for the I/O table verification error.

Error	Probable cause	Flag(s)	Error code (A400)	Possible remedy
Waiting for start input	Start input on CPU Power Unit is OFF.	A30600	None	Turn ON the start input on the CPU Power Unit or short start input terminals on the CPU Power Unit.
Waiting for SYSMAC BUS	Power to Slave is OFF or terminator is not set or doubly set.	A30602	None	Check power supply to Slaves and terminator setting.
Initializing CPU Bus Unit	SYSMAC BUS/2 terminator is missing or power to Slave is OFF.	A30603	None	Check power supply to Slaves and terminator setting.
I/O table error	A Unit has been changed making the I/O table incorrect.	A30601 A40209	00E7	Check the I/O table from the CVSS/SSS and either connect Dummy I/O Units or correct the I/O table.

Note The I/O table verification error can be set as an initialization error in the PC Setup.

8-4-2 Non-fatal Operating Errors

The following errors occur after program execution has been started. PC operation and program execution will continue after one or more of these errors have occurred. For each of these errors, the POWER, RUN, and ALARM indicators will be lit and the ERROR indicator will not be lit. The RUN output will be ON.

Error	Probable cause	Flag(s)	Error code (A400)	Possible remedy
FAL error	FAL(006) has been executed in program. Check the FAL number to determine conditions that would cause execution (programmed by user).	A40215	4101 to 42FF correspond to FAL numbers 001 to 511.	Correct according to cause indicated by FAL number (set by user).
Jump error	No JME(005) for a JMP(004), CJP(221), or CJPN(222) in the program.	A40213	00F9	Check and correct the program.
Indirect DM BCD error	Indirectly addressed DM address data is not BCD.	A40212	00F8	
Non-fatal SFC error	An error has occurred during SFC execution.	A40211	00F4	Check and correct the program. (Refer to the next table for information on non-fatal SFC errors.)
Expansion I/O Rack power interruption (when set in PC Setup)	Power is not being supplied to an Expansion I/O Rack.	A40210	00001	Supply power to the Racks. Check A419 (CPU-recognized Rack Numbers)

Error	Probable cause	Flag(s)	Error code (A400)	Possible remedy
I/O table error	Unit has been removed making I/O table incorrect.	A40209	00E7	Check the I/O table from the CVSS/SSS and either connect Dummy I/O Units or correct the I/O table.
CPU Bus Unit error	A parity error has occurred in the transfer of data between the CPU and a CPU Bus Unit or in the CPU Bus Link.	A40207	0200 to 0215 or 0231[1]	Check the errant Unit.
SYSMAC BUS/2 error	An error has occurred between a Master and Slave Rack.	A40206	00B0 to 00B3 (Masters #0 to #3.)[2]	Verify that the Slave Rack is operating properly, and check the transmission line.
SYSMAC BUS error	An error has occurred between a Master and Slave Rack.	A40205	00A0 to 00A7 (Masters #0 to #7.)[3]	Verify that the Slave Rack is operating properly, and check the transmission line.
Battery error	CPU or Memory Card backup battery is missing or its voltage has dropped.	A40204	00F7	Check battery and replace if necessary.[4]
CPU Bus Unit setting error	The registered CPU Bus Unit number doesn't agree with the registered unit number.	A40203	0400 to 0415 or 0431[5]	Check the errant Unit.
Momentary power interruption	A momentary power interruption can be set as an non-fatal error in the PC Setup.	A40202	0002	Check the power supply voltage and lines.

Note

1. Error codes 0200 to 0215 indicate CPU Bus Units #00 to #15, respectively, while 0231 indicates a CPU bus link error. Also, A422 contains the errant Unit's number, and A42315 is turned ON to indicate a CPU bus link error.
2. A424 contains the unit number of the Master involved, and A480 to A499 contain the unit number of the Slave involved.
3. A425 contains the unit number of the Master involved, and A470 to A477 contain the unit number of the Slave involved.
4. A42615 is turned ON to indicate a battery error, and A42614 is turned ON to indicate a Memory Card battery error.
5. Error codes 0400 to 0415 indicate CPU Bus Units #00 to #15, respectively, while 0431 indicates a CPU Bus Link error. Also, A427 contains the errant Unit's number.

SFC Non-fatal Errors

The following table describes SFC non-fatal errors. When an SFC non-fatal error occurs, the SFC Non-fatal Error Flag (A40211) is turned ON, and the error code is output to A418.

Error	Error code (A418)	Probable cause	Possible remedy
Overlapping action execution	0001	The program attempted to execute the same action from more than one step at the same time.	Check the program. If you wish, you can make a PC system setting so that overlapping action execution will not generate an error. With the default setting, this non-fatal SFC error is generated.
S-group AQ overuse (Note)	0002	The program attempted to simultaneously execute 128 or more actions with S-group AQs. Any actions that exceed the maximum allowable 127 will not be executed.	Check the program. S-group AQs include S, SL, SD, and DS.
Overlapping S-group AQ execution (Note)	0003	The program attempted multiple execution of the same action having an S-group AQ.	Check the program.

8-4-3 Fatal Operating Errors

The following errors occur after program execution has been started. PC operation and program execution will stop and all outputs from the PC will be turned OFF when any of the following errors occur.

None of the CPU indicators will be lit for the power interruption error, and only the POWER indicator will be lit for the Expansion Rack power interruption error. The POWER and WDT indicators will be lit for the CPU error. For all other fatal operating errors, the POWER and ERROR indicators will be lit. The RUN output will be OFF.

Error and message	Probable Cause	Flag(s)	Error code (A400)	Possible remedy
Power interruption error	A power interruption longer than the momentary power interruption time has occurred.	None	None	Check power supply voltage and power lines. Try to power-up again.
Expansion Rack power interruption error	A power interruption to an Expansion Rack has occurred.	None	None	Turn on the power supply to the Expansion CPU and Expansion I/O Racks.
CPU error	Watchdog timer has exceeded maximum setting.	None	80FF	Turn the power OFF and restart.
Memory error	An error has occurred during check of the PC, Memory Card or EM Unit memory, or program transfer could not be completed at start-up.	A40115	80F1	Check Memory Card and EM Unit to make sure they are mounted and backed up properly. Perform a Program Check Operation to locate cause of error. If error not correctable, try inputting program again.
I/O Bus error	Error has occurred in the bus line between the CPU and I/O Units.	A40114	80C0 to 80C7 or 80CE or 80CF[1]	Check cable connections between the I/O Units and Racks. Verify that terminators are connected, and then clear the error. A404 contains the errant rack/slot number.
Duplicate number error	The same number has been allocated to more than one Expansion Rack, more than one CPU Bus Unit, or one I/O word has been allocated to more than one I/O Unit.	A40113	80E9	After checking the rack numbers, unit numbers, or word allocation, turn the relevant power supply ON, OFF, and ON again, and then clear the error. A409 contains the duplicate rack number, and A410 contains the duplicate unit number.
CPU bus error	Watchdog timer error has occurred during data transfer between the CPU and CPU Bus Unit.	A40112	8100 to 8115 (indicate Units 0 to 15)	Check cable connections between the CPU and CPU Bus Units, and then clear the error. A405 contains the errant unit number.
Too many I/O points	Maximum number of I/O points or I/O Units has been exceeded in the I/O Table.	A40111	80E1	Check the number of points with I/O Table Read. If necessary, reduce number of Units in the system to keep within maximum number of I/O points and register the I/O table again.[2]
Input-output I/O table error	Input and output word designations registered in I/O table do no agree with input/output words required by Units actually mounted.	A40110	80E0	Check the I/O table with I/O Table Verification operation and check all Units to see that they are in correct configuration. When the system has been confirmed, register the I/O table again.
Program error	END(001) is not written anywhere in program or the program exceeds memory capacity.	A40109	80F0	Correct the program and then clear the error.

Error and message	Probable Cause	Flag(s)	Error code (A400)	Possible remedy
Cycle time too long	The cycle time has exceeded the maximum cycle time set in the PC Setup.	A40108	809F	Change the program or the maximum cycle time.[3]
SFC fatal error	The program contains an error in SFC syntax or an END(01) instruction is missing.	A40107	80F3	Correct the program and then clear the error. (Refer to the next table for information on SFC fatal errors.) Check to be sure that all programs end in END(01).
FALS error	FALS has been executed by the program. Check the FAL number to determine conditions that would cause execution.	A40106	C101 to C2FF correspond to FAL numbers 001 to 511.	Correct according to cause indicated by FAL number.

Note
1. Error codes 80C0 to 80C7 indicate Rack numbers 0 to 7, respectively. Error codes 80CE and 80CF indicate that the terminator is missing in operating system 0 and 1, respectively.
2. The total number of I/O words allocated to the CPU, CPU Expansion, and Expansion I/O Racks is contained in A407. A408 contains the total number of I/O words allocated to the SYSMAC BUS/2 System, and A478 contains the total number of I/O words allocated to the SYSMAC BUS System.
3. The maximum cycle time since start-up is contained in A462 and 463, and the present cycle time is contained in A464 and 465.

SFC Fatal Errors

The following table describes SFC fatal errors. When an SFC fatal error occurs, the SFC Fatal Error Flag (A40107) is turned ON, and the error code is output to A414.

Error	Error code (A418)	Probable cause	Possible remedy
No active step	0001	There is not even one active step.	Check the program.
No subchart	0002	There is no subchart entry step for the subchart dummy step, or the subchart number is incorrect.	Check and then re-transfer the program.
No initial step	0003	There is not even one initial step in the program.	Create an initial step and then re-transfer the program.
No action set	0007	There is no action program set, or the bit address for the action is incorrect.	Check and then re-transfer the program.
No transition set	0008	There is no transition program set, or the bit address for the transition is incorrect.	
Interrupt return error	0014	An interrupt return terminal was detected outside of an interrupt program.	Check the program.
Subchart return error	0015	A subchart return step was detected outside of a subchart program.	
Memory error	0004 to 0006, 0009 to 0013	Memory contents are incorrect.	Re-transfer the program.

8-5 Error Flags

The following table lists the flags and other information provided in the Auxiliary Area that can be used in troubleshooting. Details are provided in *3-6 Auxiliary Area.*

Fatal Errors

Error	Address(es)	Function
Power interruption error	A012 and A013	Date and time of last power interruption
	A014	Number of power interruptions
Expansion Rack power interruption error	None	---
CPU error	None	---
Memory error	A40115	Memory Error Flag
	A403	Memory error area location
I/O bus error	A40114	I/O Bus Error Flag
	A404	I/O Bus error rack/slot number
Duplicate number error	A40113	Duplication Error Flag
	A409	Duplicate rack number
	A410	CPU Bus Unit duplicate number
CPU Bus error	A40112	CPU Bus Error Flag
	A405	CPU Bus Unit error unit number
Too many I/O points	A40111	Too Many I/O Points Flag
	A407	Total I/O words on CPU and Expansion Racks
	A408	Total SYSMAC BUS/2 I/O words
	A478	Total SYSMAC BUS I/O words
Input-output I/O table error	A40110	I/O Setting Error Flag
Program error	A40109	Program Error Flag
Cycle time too long	A40108	Cycle Time Too Long Flag
	A462 and A463	Maximum cycle time since start-up
	A464 and A465	Present cycle time
SFC fatal error	A40107	SFC Fatal Error Flag
	A414	SFC fatal error code
FALS error	A40106	FALS Instruction Flag

Non-fatal Errors

Error	Address(es)	Function
FAL error	A40215	FAL Flag
	A430 to A461	Executed FAL number
Jump error	A40213	Jump Error Flag
Indirect DM BCD error	A40212	Indirect DM Error Flag
SFC non-fatal error	A40211	SFC Non-fatal Error Flag
	A418	SFC non-fatal error code
I/O table error	A40209	I/O Verification Error Flag
CPU Bus Unit error	A40207	CPU Bus Unit Error Flag
	A422	CPU Bus Unit error unit number
	A42315	CPU Bus Link Error Flag
SYSMAC BUS/2 error	A40206	SYSMAC BUS/2 Error Flag
	A424	SYSMAC BUS/2 error Master number
	A480 to A499	Slave unit number

Error	Address(es)	Function
SYSMAC BUS error	A40205	SYSMAC BUS Error Flag
	A425	SYSMAC BUS error Master number
	A470 to A477	Slave unit number
Battery error	A40204	Battery Low Flag
	A42614	Memory Card Battery Low Flag
	A42615	PC Battery Low Flag
CPU Bus Unit setting error	A40203	CPU Bus Unit Parameter Error Flag
	A427	CPU Bus Unit Parameter Error unit number
Momentary power interruption error	A40202	Power Interruption Flag
	A412 and A413	Date and time of last power interruption
	A014	Number of power interruptions

Appendix A
Instruction Set

Alphabetic List of Instructions by Mnemonics

The DM and EM areas can be indirectly addressed by specifying the data area as *DM or *EM, and then entering the address of the DM or EM word that contains the actual data. Index and data registers can also be used for indirect addressing.

Mnemonic	Code	Name
ACOS(↑)*	464	COSINE-TO-ANGLE
ADB(↑)	080	BINARY ADD
ADBL(↑)	084	DOUBLE BINARY ADD
ADD(↑)	070	BCD ADD
ADDL(↑)	074	DOUBLE BCD ADD
ANDL(↑)	134	DOUBLE LOGICAL AND
ANDW(↑)	130	LOGICAL AND
APR(↑)	142	ARITHMETIC PROCESS
ASC(↑)	113	ASCII CONVERT
ASFT(↑)	052	ASYNCHRONOUS SHIFT REGISTER
ASIN(↑)*	463	SINE-TO-ANGLE
ASL(↑)	060	ARITHMETIC SHIFT LEFT
ASLL(↑)	064	DOUBLE SHIFT LEFT
ASR(↑)	061	ARITHMETIC SHIFT RIGHT
ASRL(↑)	065	DOUBLE SHIFT RIGHT
ATAN(↑)*	465	TANGENT-TO-ANGLE
BAND(↑)*	272	DEAD BAND CONTROL
BCD(↑)	101	BINARY-TO-BCD
BCDL(↑)	103	DOUBLE BINARY-TO-DOUBLE BCD
BCDS(↑)*	276	SIGNED BINARY-TO-BCD
BCMP(↑)	022	BLOCK COMPARE
BCNT(↑)	114	BIT COUNTER
BDSL(↑)*	278	DOUBLE SIGNED BINARY-TO-BCD
BEND*	<001>	BLOCK PROGRAM END
BIN(↑)	100	BCD-TO-BINARY
BINL(↑)	102	DOUBLE BCD-TO-DOUBLE BINARY
BINS(↑)*	275	SIGNED BCD-TO-BINARY
BISL(↑)*	277	DOUBLE SIGNED BCD-TO-BINARY
BPPS*	<011>	BLOCK PROGRAM PAUSE
BPRG*	250	BLOCK PROGRAM
BPRS*	<012>	BLOCK PROGRAM RESTART
BSET(↑)	041	BLOCK SET

Mnemonic	Code	Name
BXFR(↑)*	046	INTERBANK BLOCK TRANSFER
CADD(↑)	145	CALENDAR ADD
CCL(↑)	172	LOAD FLAGS
CCS(↑)	173	SAVE FLAGS
CJP*	221	CONDITIONAL JUMP
CJPN*	222	CONDITIONAL JUMP
CLC(↑)	079	CLEAR CARRY
CLI(↑)	154	CLEAR INTERRUPT
CMND(↑)	194	DELIVER COMMAND
CMP(!)	020	COMPARE
CMP(!)*	028	UNSIGNED COMPARE
CMPL	021	DOUBLE COMPARE
CMPL*	029	DOUBLE UNSIGNED COMPARE
CNR(↑)	236	RESET TIMER/COUNTER
CNTR	012	REVERSIBLE COUNTER
CNTW*	<014>	COUNTER WAIT
COLL(↑)	045	DATA COLLECT
COLM(↑)	116	LINE-TO-COLUMN
COM(↑)	138	COMPLEMENT
COML(↑)	139	DOUBLE COMPLEMENT
COS*	461	COSINE
CPS(!)*	026	SIGNED BINARY COMPARE
CPSL*	027	DOUBLE SIGNED BINARY COMPARE
CSUB(↑)	146	CALENDAR SUBTRACT
DATE(↑)*	179	CLOCK COMPENSATION
DCBL(↑)	097	DOUBLE DECREMENT BINARY
DEC(↑)	091	DECREMENT BCD
DECB(↑)	093	DECREMENT BINARY
DECL(↑)	095	DOUBLE DECREMENT BCD
DEG(↑)*	459	RADIANS-TO-DEGREES
DIFD(!)	014	DIFFERENTIATE DOWN
DIFU(!)	013	DIFFERENTIATE UP
DIST(↑)	044	SINGLE WORD DISTRIBUTE
DIV(↑)	073	BCD DIVIDE
DIVL(↑)	077	DOUBLE BCD DIVIDE

Note Instructions marked with an asterisk (*) are supported by version-2 CVM1 CPUs only.

Mnemonic	Code	Name
DMPX(↑)	111	16-TO-4/256-8 ENCODER
DOWN*	019	CONDITION OFF
DVB(↑)	083	BINARY DIVIDE
DVBL(↑)	087	DOUBLE BINARY DIVIDE
ELSE*	<003>	NO CONDITIONAL BRANCH
EMBC(↑)	171	SELECT EM BANK
END	001	END
EQU(↑)	025	EQUAL
EXIT(NOT)*	<006>	CONDITIONAL END
EXP(↑)*	467	EXPONENT
FAL(↑)	006	FAILURE ALARM
FALS(↑)	007	FAILURE ALARM
FDIV(↑)	141	FLOATING POINT DIVIDE(BCD)
FIFO(↑)	163	FIRST IN FIRST OUT
FILP(↑)	182	READ PROGRAM FILE
FILR(↑)	180	READ DATA FILE
FILW(↑)	181	WRITE DATA FILE
FIX(↑)*	450	FLOATING-TO-16-BIT
FIXL(↑)*	451	FLOATING-TO-32-BIT
FLSP(↑)	183	CHANGE STEP PROGRAM
FLT(↑)*	452	16-BIT-TO-FLOATING
FLTL(↑)*	453	32-BIT-TO-FLOATING
FPD*	177	FAILURE POINT DETECTION
HEX(↑)*	117	ASCII-TO-HEX
HMS(↑)	144	SECONDS-TO-HOURS
IEND*	<004>	END OF BRANCH
IF(NOT)*	<002>	CONDITIONAL BRANCH
IL	002	INTERLOCK
ILC	003	INTERLOCK CLEAR
INBL(↑)	096	DOUBLE INCREMENT BINARY
INC(↑)	090	INCREMENT BCD
INCB(↑)	092	INCREMENT BINARY
INCL(↑)	094	DOUBLE INCREMENT BCD
IODP(↑)	189	I/O DISPLAY
IORF(↑)	184	I/O REFRESH
IORS	188	ENABLE ACCESS
IOSP(↑)	187	DISABLE ACCESS
JME	005	JUMP END
JMP	004	JUMP
KEEP(!)	011	KEEP

Mnemonic	Code	Name
LEND(NOT)*	<010>	REPEAT BLOCK END
LIFO(↑)	162	LAST IN FIRST OUT
LINE(↑)	115	COLUMN-TO-LINE
LMT(↑)*	271	LIMIT CONTROL
LOG(↑)*	468	LOGARITHM
LOOP*	<009>	REPEAT BLOCK
MARK	174	MARK TRACE
MAX(↑)	165	FIND MAXIMUM
MCMP(↑)	024	MULTIPLE COMPARE
MCRO(↑)*	156	MACRO
MIN(↑)	166	FIND MINIMUM
MLB(↑)	082	BINARY MULTIPLY
MLBL(↑)	086	DOUBLE BINARY MULTIPLY
MLPX(↑)	110	4-TO-16/8-TO-256 DECODER
MOV(!↑)	030	MOVE
MOVB(↑)	042	MOVE BIT
MOVD(↑)	043	MOVE DIGIT
MOVL(↑)	032	DOUBLE MOVE
MOVQ	037	MOVE QUICK
MOVR(↑)	036	MOVE TO REGISTER
MSG(↑)	195	MESSAGE
MSKR(↑)	155	READ MASK
MSKS(↑)	153	INTERRUPT MASK
MTIM	122	MULTI-OUTPUT TIMER
MUL(↑)	072	BCD MULTIPLY
MULL(↑)	076	DOUBLE BCD MULTIPLY
MVN(↑)	031	MOVE NOT
MVNL(↑)	033	DOUBLE MOVE NOT
NASL(↑)*	056	SHIFT N-BITS LEFT
NASR(↑)*	057	SHIFT N-BITS RIGHT
NEG(↑)	104	2'S COMPLEMENT
NEGL(↑)	105	DOUBLE 2'S COMPLEMENT
NOP	000	NO OPERATION
NOT	010	NOT
NSFL(↑)*	054	SHIFT N-BIT DATA LEFT
NSFR(↑)*	055	SHIFT N-BIT DATA RIGHT
NSLL(↑)*	058	DOUBLE SHIFT N-BIT LEFT
NSRL(↑)*	059	DOUBLE SHIFT N-BIT RIGHT
ORW(↑)	131	LOGICAL OR
ORWL(↑)	135	DOUBLE LOGICAL OR
PID*	270	PID CONTROL
PUSH(↑)	161	PUSH ONTO STACK

Note Instructions marked with an asterisk (*) are supported by version-2 CVM1 CPUs only.

Mnemonic	Code	Name
RAD(↑)*	458	DEGREES-TO-RADIANS
RD2*	280	I/O READ 2
READ	190	I/O READ
RECV(↑)	193	NETWORK RECEIVE
REGL(↑)	175	LOAD REGISTER
REGS(↑)	176	SAVE REGISTER
RET	152	SUBROUTINE RETURN
RLNC(↑)*	260	ROTATE LEFT WITHOUT CARRY
RLNL(↑)*	262	DOUBLE ROTATE LEFT WITHOUT CARRY
ROL(↑)	062	ROTATE LEFT WITH CARRY
ROLL(↑)	066	DOUBLE ROTATE LEFT WITH CARRY
ROOT(↑)	140	BCD SQUARE ROOT
ROR(↑)	063	ROTATE RIGHT WITH CARRY
RORL(↑)	067	DOUBLE ROTATE RIGHT WITH CARRY
ROTB(↑)*	274	BINARY ROOT
RRNC(↑)*	261	ROTATE RIGHT WITHOUT CARRY
RRNL(↑)*	263	ROTATE LEFT WITHOUT CARRY
RSET(!↑↓)	017	RSET
RSTA(↑)*	048	MULTIPLE BIT RESET
SA(↑)	210	ACTIVATE STEP
SBB(↑)	081	BINARY SUBTRACT
SBBL(↑)	085	DOUBLE BINARY SUBTRACT
SBN	150	SUBROUTINE ENTER
SBS(↑)	151	SUBROUTINE CALL
SDEC(↑)	112	7-SEGMENT DECODER
SE(↑)	214	DEACTIVATE STEP
SEC(↑)	143	HOURS-TO-SECONDS
SEND(↑)	192	NETWORK SEND
SET(!↑↓)	016	SET
SETA(↑)*	047	MULTIPLE BIT SET
SF(↑)	213	END STEP
SFT	050	SHIFT REGISTER
SFTR(↑)	051	REVERSIBLE SHIFT REGISTER
SIGN(↑)	106	SIGN
SIN(↑)*	460	SINE
SLD(↑)	068	SHIFT DIGIT LEFT
SNXT	009	STEP START
SOFF(↑)	215	RESET STEP
SP(↑)	211	PAUSE STEP

Mnemonic	Code	Name
SQRT(↑)*	466	SQUARE ROOT
SR(↑)	212	RESTART STEP
SRCH(↑)	164	DATA SEARCH
SRD(↑)	069	SHIFT DIGIT RIGHT
SSET(↑)	160	SET STACK
STC(↑)	078	SET CARRY
STEP	008	STEP DEFINE
SUB(↑)	071	BCD SUBTRACT
SUBL(↑)	075	DOUBLE BCD SUBTRACT
SUM(↑)	167	SUM
TAN(↑)*	462	TANGENT
TCMP(↑)	023	TABLE COMPARE
TCNT	123	TRANSITION COUNTER
TIMH	015	HIGH-SPEED TIMER
TIML	121	DOUBLE TIMER
TIMW*	<013>	TIMER WAIT
TMHW*	<015>	HIGH-SPEED TIMER WAIT
TOUT	202	TRANSITION OUTPUT
TRSM	170	TRACE MEMORY
TSR(↑)	124	READ STEP TIMER
TST*	350	BIT TEST
TSTN*	351	BIT TEST
TSW(↑)	125	WRITE STEP TIMER
TTIM	120	ACCUMULATIVE TIMER
UP*	018	CONDITION ON
WAIT(NOT)*	<005>	1-SCAN WAIT
WDT*	178	MAXIMUM CYCLE TIME EXTEND
WR2*	281	I/O UNIT WRITE 2
WRIT	191	I/O WRITE
WSFT(↑)	053	WORD SHIFT
XCGL(↑)	035	DOUBLE DATA EXCHANGE
XCHG(↑)	034	DATA EXCHANGE
XFER(↑)	040	BLOCK TRANSFER
XFRB(↑)*	038	MULTIPLE BIT TRANSFER
XNRL(↑)	137	DOUBLE EXCLUSIVE NOR
XNRW(↑)	133	EXCLUSIVE NOR
XORL(↑)	136	DOUBLE EXCLUSIVE OR
XORW(↑)	132	EXCLUSIVE OR
ZONE(↑)*	273	DEAD-ZONE CONTROL
/(↑)*	430	SIGNED BINARY DIVIDE
/B(↑)*	434	BCD DIVIDE
/BL(↑)*	435	DOUBLE BCD DIVIDE

Note Instructions marked with an asterisk (*) are supported by version-2 CVM1 CPUs only.

Mnemonic	Code	Name
/F(↑)*	457	FLOATING-POINT DIVIDE
/L(↑)*	431	DOUBLE SIGNED BINARY DIVIDE
/U(↑)*	432	UNSIGNED BINARY DIVIDE
/UL(↑)*	433	DOUBLE UNSIGNED BINARY DIVIDE
+(↑)*	400	SIGNED BINARY ADD WITHOUT CARRY
+B(↑)*	404	BCD ADD WITHOUT CARRY
+BC(↑)*	406	BCD ADD WITH CARRY
+BCL(↑)*	407	DOUBLE BCD ADD WITH CARRY
+BL(↑)*	405	DOUBLE BCD ADD WITHOUT CARRY
+C(↑)*	402	SIGNED BINARY ADD WITH CARRY
+CL(↑)*	403	DOUBLE SIGNED BINARY ADD WITH CARRY
+F(↑)*	454	FLOATING-POINT ADD
+L(↑)*	401	DOUBLE SIGNED BINARY ADD WITHOUT CARRY
–(↑)*	410	SIGNED BINARY SUBTRACT WITHOUT CARRY
–B(↑)*	414	BCD SUBTRACT WITHOUT CARRY
–BC(↑)*	416	BCD SUBTRACT WITH CARRY
–BCL(↑)*	417	DOUBLE BCD SUBTRACT WITH CARRY
–BL(↑)*	415	DOUBLE BCD SUBTRACT WITHOUT CARRY
–C(↑)*	412	SIGNED BINARY SUBTRACT WITH CARRY
–CL(↑)*	413	DOUBLE SIGNED BINARY SUBTRACT WITH CARRY
–F(↑)*	455	FLOATING-POINT SUBTRACT
–L(↑)*	411	DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY
=*	300	EQUAL
=L*	301	DOUBLE EQUAL

Mnemonic	Code	Name
=S*	302	SIGNED EQUAL
=SL*	303	DOUBLE SIGNED EQUAL
<*	310	LESS THAN
<=*	315	LESS THAN OR EQUAL
<=L*	316	DOUBLE LESS THAN OR EQUAL
<=S*	317	SIGNED LESS THAN OR EQUAL
<=SL*	318	DOUBLE SIGNED LESS THAN OR EQUAL
<>*	305	NOT EQUAL
<>L*	306	DOUBLE NOT EQUAL
<>S*	307	SIGNED NOT EQUAL
<>SL*	308	DOUBLE SIGNED NOT EQUAL
<L*	311	DOUBLE LESS THAN
<S*	312	SIGNED LESS THAN
<SL*	313	DOUBLE SIGNED LESS THAN
>*	320	GREATER THAN
>=*	325	GREATER THAN OR EQUAL
>=L*	326	DOUBLE GREATER THAN OR EQUAL
>=S*	327	SIGNED GREATER THAN OR EQUAL
>=SL*	328	DOUBLE SIGNED GREATER THAN OR EQUAL
>L*	321	DOUBLE GREATER THAN
>S*	322	SIGNED GREATER THAN
>SL*	323	DOUBLE SIGNED GREATER THAN
∗(↑)*	420	SIGNED BINARY MULTIPLY
∗B(↑)*	424	BCD MULTIPLY
∗BL(↑)*	425	DOUBLE BCD MULTIPLY
∗F(↑)*	456	FLOATING-POINT MULTIPLY
∗L(↑)*	421	DOUBLE SIGNED BINARY MULTIPLY
∗U(↑)*	422	UNSIGNED BINARY MULTIPLY
∗UL(↑)*	423	DOUBLE UNSIGNED BINARY MULTIPLY

Note Instructions marked with an asterisk (*) are supported by version-2 CVM1 CPUs only.

Alphabetic List of Instructions by Function Code

Sequence Control, Error Handling, and Step Control Instructions

Code	Mnemonic	Name
000	NOP	NO OPERATION
001	END	END
002	IL	INTERLOCK
003	ILC	INTERLOCK CLEAR
004	JMP	JUMP
005	JME	JUMP END
006	FAL(↑)	FAILURE ALARM
007	FALS	FAILURE ALARM
008	STEP	STEP DEFINE
009	SNXT	STEP START

Sequence I/O Instructions

Code	Mnemonic	Name
010	NOT	NOT
011	KEEP(!)	KEEP
012	CNTR	REVERSIBLE COUNTER
013	DIFU(!)	DIFFERENTIATE UP
014	DIFD(!)	DIFFERENTIATE DOWN
015	TIMH	HIGH-SPEED TIMER
016	SET(!↑↓)	SET
017	RSET(!↑↓)	RSET
018	UP*	CONDITION ON
019	DOWN*	CONDITION OFF

Data Compare Instructions

Code	Mnemonic	Name
020	CMP(!↑)	COMPARE
021	CMPL	DOUBLE COMPARE
022	BCMP(↑)	BLOCK COMPARE
023	TCMP(↑)	TABLE COMPARE
024	MCMP(↑)	MULTIPLE COMPARE
025	EQU(↑)	EQUAL
026	CPS(!)*	SIGNED BINARY COMPARE
027	CPSL*	DOUBLE SIGNED BINARY COMPARE
028	CMP(!)*	UNSIGNED COMPARE
029	CMPL*	DOUBLE UNSIGNED COMPARE

Data Move and Sequence Output Instructions

Code	Mnemonic	Name
030	MOV(!↑)	MOVE
031	MVN(↑)	MOVE NOT
032	MOVL(↑)	DOUBLE MOVE
033	MVNL(↑)	DOUBLE MOVE NOT
034	XCHG(↑)	DATA EXCHANGE
035	XCGL(↑)	DOUBLE DATA EXCHANGE
036	MOVR(↑)	MOVE TO REGISTER
037	MOVQ	MOVE QUICK
038	XFRB(↑)*	MULTIPLE BIT TRANSFER
040	XFER(↑)	BLOCK TRANSFER
041	BSET(↑)	BLOCK SET
042	MOVB(↑)	MOVE BIT
043	MOVD(↑)	MOVE DIGIT
044	DIST(↑)	SINGLE WORD DISTRIBUTE
045	COLL(↑)	DATA COLLECT
046	BXFR(↑)*	INTERBANK BLOCK TRANSFER
047	SETA(↑)*	MULTIPLE BIT SET
048	RSTA(↑)*	MULTIPLE BIT RESET

Note Instructions marked with an asterisk (*) are supported by version-2 CVM1 CPUs only.

Data Shift Instructions

Code	Mnemonic	Name
050	SFT	SHIFT REGISTER
051	SFTR(↑)	REVERSIBLE SHIFT REGISTER
052	ASFT(↑)	ASYNCHRONOUS SHIFT REGISTER
053	WSFT(↑)	WORD SHIFT
054	NSFL*	SHIFT N-BIT DATA LEFT
055	NSFR*	SHIFT N-BIT DATA RIGHT
056	NASL*	SHIFT N-BITS LEFT
057	NASR*	SHIFT N-BITS RIGHT
058	NSLL*	DOUBLE SHIFT N-BIT LEFT
059	NSRL*	DOUBLE SHIFT N-BIT RIGHT
060	ASL(↑)	ARITHMETIC SHIFT LEFT
061	ASR(↑)	ARITHMETIC SHIFT RIGHT
062	ROL(↑)	ROTATE LEFT
063	ROR(↑)	ROTATE RIGHT
064	ASLL(↑)	DOUBLE SHIFT LEFT
065	ASRL(↑)	DOUBLE SHIFT RIGHT
066	ROLL(↑)	DOUBLE ROTATE LEFT
067	RORL(↑)	DOUBLE ROTATE RIGHT
068	SLD(↑)	SHIFT DIGIT LEFT
069	SRD(↑)	SHIFT DIGIT RIGHT

BCD Calculation and Carry Instructions

Code	Mnemonic	Name
070	ADD(↑)	BCD ADD
071	SUB(↑)	BCD SUBTRACT
072	MUL(↑)	BCD MULTIPLY
073	DIV(↑)	BCD DIVIDE
074	ADDL(↑)	DOUBLE BCD ADD
075	SUBL(↑)	DOUBLE BCD SUBTRACT
076	MULL(↑)	DOUBLE BCD MULTIPLY
077	DIVL(↑)	DOUBLE BCD DIVIDE
078	STC(↑)	SET CARRY
079	CLC(↑)	CLEAR CARRY

Binary Calculation Instructions

Code	Mnemonic	Name
080	ADB(↑)	BINARY ADD
081	SBB(↑)	BINARY SUBTRACT
082	MLB(↑)	BINARY MULTIPLY
083	DVB(↑)	BINARY DIVIDE
084	ADBL(↑)	DOUBLE BINARY ADD
085	SBBL(↑)	DOUBLE BINARY SUBTRACT
086	MLBL(↑)	DOUBLE BINARY MULTIPLY
087	DVBL(↑)	DOUBLE BINARY DIVIDE

Increment/Decrement Instructions

Code	Mnemonic	Name
090	INC(↑)	INCREMENT BCD
091	DEC(↑)	DECREMENT BCD
092	INCB(↑)	INCREMENT BINARY
093	DECB(↑)	DECREMENT BINARY
094	INCL(↑)	DOUBLE INCREMENT BCD
095	DECL(↑)	DOUBLE DECREMENT BCD
096	INBL(↑)	DOUBLE INCREMENT BINARY
097	DCBL(↑)	DOUBLE DECREMENT BINARY

Data Format Conversion and Special Calculation Instructions

Code	Mnemonic	Name
100	BIN(↑)	BCD-TO-BINARY
101	BCD(↑)	BINARY-TO-BCD
102	BINL(↑)	DOUBLE BCD-TO-DOUBLE BINARY
103	BCDL(↑)	DOUBLE BINARY-TO-DOUBLE BCD
104	NEG(↑)	2'S COMPLEMENT
105	NEGL(↑)	DOUBLE 2'S COMPLEMENT
106	SIGN(↑)	SIGN

Basic I/O Unit Instructions

Code	Mnemonic	Name
110	MLPX(↑)	4-TO-16 DECODER
111	DMPX(↑)	16-TO-4 ENCODER
112	SDEC(↑)	7-SEGMENT DECODER
113	ASC(↑)	ASCII CONVERT
114	BCNT(↑)	BIT COUNTER
115	LINE(↑)	COLUMN-TO-LINE
116	COLM(↑)	LINE-TO-COLUMN
117	HEX(↑)*	ASCII-TO-HEX

Note Instructions marked with an asterisk (*) are supported by version-2 CVM1 CPUs only.

Special Timer and SFC Control Instructions

Code	Mnemonic	Name
120	TTIM	ACCUMULATIVE TIMER
121	TIML	DOUBLE TIMER
122	MTIM	MULTI-OUTPUT TIMER
123	TCNT	TRANSITION COUNTER
124	TSR(↑)	READ STEP TIMER
125	TSW(↑)	WRITE STEP TIMER

Logical Instructions

Code	Mnemonic	Name
130	ANDW(↑)	LOGICAL AND
131	ORW(↑)	LOGICAL OR
132	XORW(↑)	EXCLUSIVE OR
133	XNRW(↑)	EXCLUSIVE NOR
134	ANDL(↑)	DOUBLE LOGICAL AND
135	ORWL(↑)	DOUBLE LOGICAL OR
136	XORL(↑)	DOUBLE EXCLUSIVE OR
137	XNRL(↑)	DOUBLE EXCLUSIVE NOR
138	COM(↑)	COMPLEMENT
139	COML(↑)	DOUBLE COMPLEMENT

Special and Time-related Instructions

Code	Mnemonic	Name
140	ROOT(↑)	BCD SQUARE ROOT
141	FDIV(↑)	FLOATING POINT DIVIDE(BCD)
142	APR(↑)	ARITHMETIC PROCESS
143	SEC(↑)	HOURS-TO-SECONDS
144	HMS(↑)	SECONDS-TO-HOURS
145	CADD(↑)	CALENDAR ADD
146	CSUB(↑)	CALENDAR SUBTRACT

Subroutine and Interrupt Instructions

Code	Mnemonic	Name
150	SBN	SUBROUTINE ENTER
151	SBS(↑)	SUBROUTINE CALL
152	RET	SUBROUTINE RETURN
153	MSKS(↑)	INTERRUPT MASK
154	CLI(↑)	CLEAR INTERRUPT
155	MSKR(↑)	READ MASK
156	MCRO(↑)*	MACRO

Table Data Processing Instructions

Code	Mnemonic	Name
160	SSET(↑)	SET STACK
161	PUSH(↑)	PUSH ONTO STACK
162	LIFO(↑)	LAST IN FIRST OUT
163	FIFO(↑)	FIRST IN FIRST OUT
164	SRCH(↑)	DATA SEARCH
165	MAX(↑)	FIND MAXIMUM
166	MIN(↑)	FIND MINIMUM
167	SUM(↑)	SUM

Debugging, Special, Error Processing and Time-related Instructions

Code	Mnemonic	Name
170	TRSM	TRACE MEMORY
171	EMBC(↑)	SELECT EM BANK
172	CCL(↑)	LOAD FLAGS
173	CCS(↑)	SAVE FLAGS
174	MARK	MARK TRACE
175	REGL(↑)	LOAD REGISTER
176	REGS(↑)	SAVE REGISTER
177	FPD*	FAILURE POINT DETECTION
178	WDT(↑)*	MAXIMUM CYCLE TIME EXTEND
179	DATE(↑)*	CLOCK COMPENSATION

File Memory, Basic I/O, and Special I/O Instructions

Code	Mnemonic	Name
180	FILR(↑)	READ DATA FILE
181	FILW(↑)	WRITE DATA FILE
182	FILP(↑)	READ PROGRAM FILE
183	FLSP(↑)	CHANGE STEP PROGRAM
184	IORF(↑)	I/O REFRESH
187	IOSP(↑)	DISABLE ACCESS
188	IORS	ENABLE ACCESS
189	IODP(↑)	I/O DISPLAY
190	READ	I/O READ
191	WRIT	I/O WRITE
192	SEND(↑)	NETWORK SEND
193	RECV(↑)	NETWORK RECEIVE
194	CMND(↑)	DELIVER COMMAND
195	MSG(↑)	MESSAGE

Note Instructions marked with an asterisk (*) are supported by version-2 CVM1 CPUs only.

SFC Control Instructions

Code	Mnemonic	Name
202	TOUT	TRANSITION OUTPUT
210	SA(↑)	ACTIVATE STEP
211	SP(↑)	PAUSE STEP
212	SR(↑)	RESTART STEP
213	SF(↑)	END STEP
214	SE(↑)	DEACTIVATE STEP
215	SOFF(↑)	RESET STEP

Sequence Control and Timer/Counter Reset Instructions

Code	Mnemonic	Name
221	CJP*	CONDITIONAL JUMP
222	CJPN*	CONDITIONAL JUMP
236	CNR(↑)	RESET TIMER/COUNTER

Block Program Instruction( )

Code	Mnemonic	Name
250	BPRG*	BLOCK PROGRAM

Block Program Instructions < >

Code	Mnemonic	Name
<001>	BEND*	BLOCK PROGRAM END
<002>	IF(NOT)*	CONDITIONAL BRANCH
<003>	ELSE*	NO CONDITIONAL BRANCH
<004>	IEND*	END OF BRANCH
<005>	WAIT(NOT)*	1-SCAN WAIT
<006>	EXIT(NOT)*	CONDITIONAL END
<009>	LOOP*	REPEAT BLOCK
<010>	LEND(NOT)*	REPEAT BLOCK END
<011>	BPPS*	BLOCK PROGRAM PAUSE
<012>	BPRS*	BLOCK PROGRAM RESTART
<013>	TIMW*	TIMER WAIT
<014>	CNTW*	COUNTER WAIT
<015>	TMHW*	HIGH-SPEED TIMER WAIT

Data Shift Instructions

Code	Mnemonic	Name
260	RLNC(↑)*	ROTATE LEFT WITHOUT CARRY
261	RRNC(↑)*	ROTATE RIGHT WITHOUT CARRY
262	RLNL(↑)*	DOUBLE ROTATE LEFT WITHOUT CARRY
263	RRNL(↑)*	ROTATE LEFT WITHOUT CARRY

Data Control, Special Calculation, and Data Conversion Instructions

Code	Mnemonic	Name
270	PID*	PID CONTROL
271	LMT(↑)*	LIMIT CONTROL
272	BAND(↑)*	DEAD BAND CONTROL
273	ZONE(↑)*	DEAD-ZONE CONTROL
274	ROTB(↑)*	BINARY ROOT
275	BINS(↑)*	SIGNED BCD-TO-BINARY
276	BCDS(↑)*	SIGNED BINARY-TO-BCD
277	BISL(↑)*	DOUBLE SIGNED BCD-TO-BINARY
278	BDSL(↑)*	DOUBLE SIGNED BINARY-TO-BCD

Special I/O Instructions

Code	Mnemonic	Name
280	RD2*	I/O READ 2
281	WR2*	I/O UNIT WRITE 2

Data Comparison Instructions

Code	Mnemonic	Name
300	=*	EQUAL
301	=L*	DOUBLE EQUAL
302	=S*	SIGNED EQUAL
303	=SL*	DOUBLE SIGNED EQUAL
305	<>*	NOT EQUAL
306	<>L*	DOUBLE NOT EQUAL
307	<>S*	SIGNED NOT EQUAL
308	<>SL*	DOUBLE SIGNED NOT EQUAL
310	<*	LESS THAN
311	<L*	DOUBLE LESS THAN
312	<S*	SIGNED LESS THAN
313	<SL*	DOUBLE SIGNED LESS THAN
315	<=*	LESS THAN OR EQUAL
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
321	>L*	DOUBLE GREATER THAN
322	>S*	SIGNED GREATER THAN
323	>SL*	DOUBLE SIGNED GREATER THAN
325	>=*	GREATER THAN OR EQUAL
326	>=L*	DOUBLE GREATER THAN OR EQUAL
327	>=S*	SIGNED GREATER THAN OR EQUAL
328	>=SL*	DOUBLE SIGNED GREATER THAN OR EQUAL

Note Instructions marked with an asterisk (*) are supported by version-2 CVM1 CPUs only.

Bit Tests

Code	Mnemonic	Name
350	TST*	BIT TEST
351	TSTN*	BIT TEST

Symbol Math Instructions

Code	Mnemonic	Name
400	+(↑)*	SIGNED BINARY ADD WITHOUT CARRY
401	+L(↑)*	DOUBLE SIGNED BINARY ADD WITHOUT CARRY
402	+C(↑)*	SIGNED BINARY ADD WITH CARRY
403	+CL(↑)*	DOUBLE SIGNED BINARY ADD WITH CARRY
404	+B(↑)*	BCD ADD WITHOUT CARRY
405	+BL(↑)*	DOUBLE BCD ADD WITHOUT CARRY
406	+BC(↑)*	BCD ADD WITH CARRY
407	+BCL(↑)*	DOUBLE BCD ADD WITH CARRY
410	–(↑)*	SIGNED BINARY SUBTRACT WITHOUT CARRY
411	-L(↑)*	DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY
412	–C(↑)*	SIGNED BINARY SUBTRACT WITH CARRY
413	–CL(↑)*	DOUBLE SIGNED BINARY SUBTRACT WITH CARRY
414	–B(↑)*	BCD SUBTRACT WITHOUT CARRY
415	–BL(↑)*	DOUBLE BCD SUBTRACT WITHOUT CARRY
416	–BC(↑)*	BCD SUBTRACT WITH CARRY
417	–BCL(↑)*	DOUBLE BCD SUBTRACT WITH CARRY
420	∗(↑)*	SIGNED BINARY MULTIPLY
421	∗L(↑)*	DOUBLE SIGNED BINARY MULTIPLY
422	∗U(↑)*	UNSIGNED BINARY MULTIPLY
423	∗UL(↑)*	DOUBLE UNSIGNED BINARY MULTIPLY
424	∗B(↑)*	BCD MULTIPLY
425	∗BL(↑)*	DOUBLE BCD MULTIPLY
430	/(↑)*	SIGNED BINARY DIVIDE
431	/L(↑)*	DOUBLE SIGNED BINARY DIVIDE
432	/U(↑)*	UNSIGNED BINARY DIVIDE
433	/UL(↑)*	DOUBLE UNSIGNED BINARY DIVIDE
434	/B(↑)*	BCD DIVIDE
435	/BL(↑)*	DOUBLE BCD DIVIDE

Floating-point Math Instructions

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(↑)*	COSINE-TO-ANGLE
465	ATAN(↑)*	TANGENT-TO-ANGLE
466	SQRT(↑)*	SQUARE ROOT
467	EXP(↑)*	EXPONENT
468	LOG(↑)*	LOGARITHM

Note Instructions marked with an asterisk (*) are supported by version-2 CVM1 CPUs only.

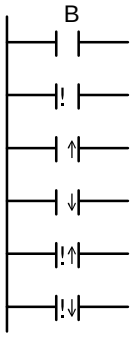
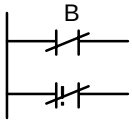
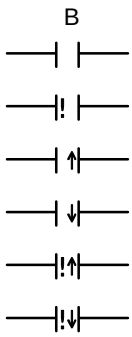
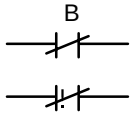
Programming Instructions

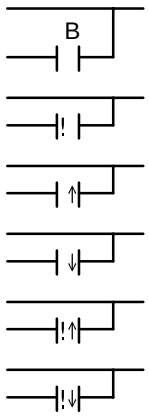
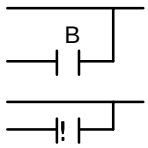
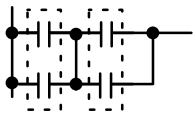
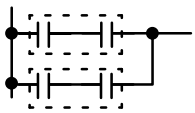
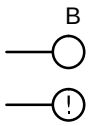
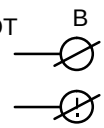
The following tables detail all of the ladder diagram programming instructions for the CV-series PCs and the applicable data areas for each. Bit and word addresses for each area are given in the footnotes.

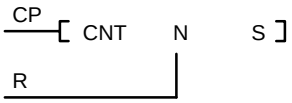
Up and down differentiated instructions are indicated with an up or down arrow (↑ or ↓) prefix. Immediate refresh instructions are indicated with a "!" prefix.

The DM and EM areas can be indirectly addressed by specifying the data area as ∗DM or ∗EM, and then entering the address of the DM or EM word that contains the actual data. Index and data registers can also be used for indirect addressing.

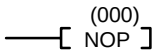
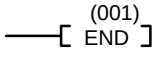
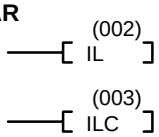
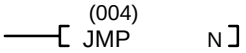
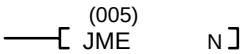
BASIC Instructions

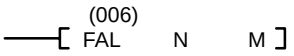
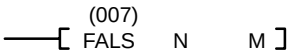
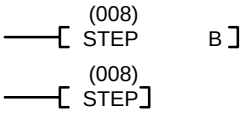
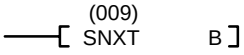
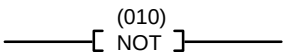
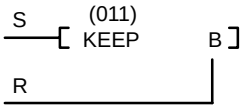
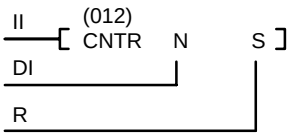
Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
LOAD LD, !LD, ↑LD, ↓LD, !↑LD, !↓LD 	Defines the status of bit B as the execution condition for subsequent operations on the instruction line.	**B:** CIO G A T/C ST TN	121
LOAD NOT LD NOT, !LD NOT 	Defines the inverse of the status of bit B as the execution condition for subsequent operations on the instruction line.	**B:** CIO G A T/C ST TN	121
AND AND, !AND, ↑AND, ↓AND, !↑AND, !↓AND 	Logically ANDs the status of the designated bit with the current execution condition.	**B:** CIO G A T/C ST TN	121
AND NOT AND NOT, !AND NOT 	Logically ANDs the inverse of the status of the designated bit with the current execution condition.	**B:** CIO G A T/C ST TN	121

Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
OR OR, !OR, ↑OR, ↓OR, !↑OR, !↓OR 	Logically ORs the status of the designated bit with the current execution condition.	**B:** CIO G A T/C ST TN	121
OR NOT OR NOT, !OR NOT 	Logically ORs the inverse of the status of the designated bit with the current execution condition.	**B:** CIO G A T/C ST TN	121
AND LOAD AND LD 	Logically ANDs the execution conditions left over from preceding logic blocks.	None	125
OR LOAD OR LOAD 	Logically ORs the execution conditions left over from preceding logic blocks.	None	125
OUTPUT OUT, !OUT 	Turns B ON for an ON execution condition; turns B OFF for an OFF execution condition.	**B:** CIO G A TR	126
OUTPUT NOT OUT NOT, !OUT NOT 	Turns B OFF for an ON execution condition; turns B ON for an OFF execution condition.	**B:** CIO G A	126
TIMER TIM 	Creates a decrementing ON-delay timer. S (set value): 000.0 to 999.9 s. Each timer number (BCD) can be used in only one timer instruction (TIM, TIMH(015), and TTIM(120)), unless the timers are never active simultaneously. The timer number can be designated as a constant or indirectly addressed by placing the address of the present value for the timer in an Index Register.	**N:** # **S:** CIO G A T/C # DM DR IR	142

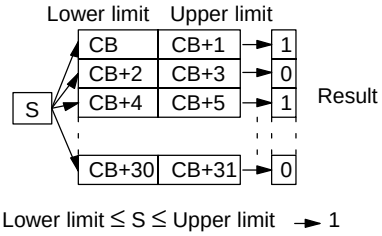
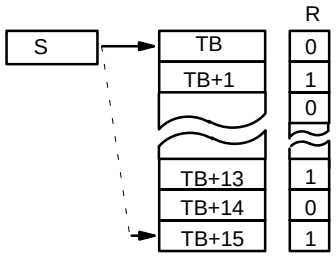
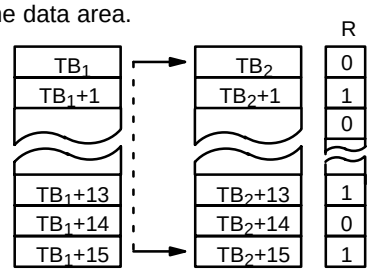
Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
COUNTER CNT 	Creates a decrementing counter. S (set value): 0 to 9999; CP: count pulse; R: reset input. Each counter number (BCD) can be used in only one counter instruction (CNT, CNTR(012), and TCNT(123)), unless the counters are never active simultaneously. The counter number can be designated as a constant or indirectly addressed by placing the address of the present value of the counter in an Index Register.	**N:** # **S:** CIO G A T/C # DM DR IR	153

Special Instructions

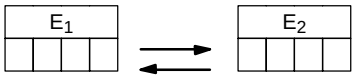
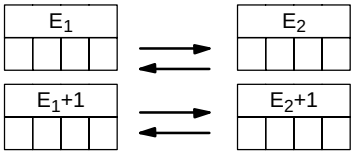
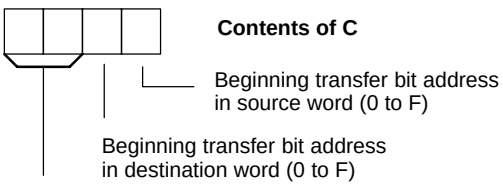
Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
NO OPERATION NOP 	Nothing is executed and program operation moves to the next instruction.	None	139
END END 	Required at the end of each program, including action and transition programs. Instructions located after END(01) will not be executed.	None	139
INTERLOCK IL **INTERLOCK CLEAR** ILC 	If an interlock condition is OFF, all outputs between the current IL(002) and the next ILC(003) are turned OFF; the PVs of timers that use timer numbers (TIM, TIMH(015), and TIML(121)) are reset. Other instructions are treated as NOP. The PVs of counters, TTIM(120), and MTIM(122) are maintained. If the execution condition is ON, execution continues normally.	None	134
JUMP JMP 	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) in 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 JME(005) with the same jump number without executing any instructions in between. TIM and TIMH(015) continue counting even when jumped.	**N:** CIO G A T/C # DM DR IR	136
JUMP END JME 	JME(005) defines the destination of a jump started by JMP(004).	**N:** #	136

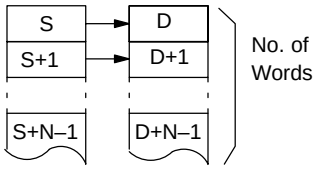
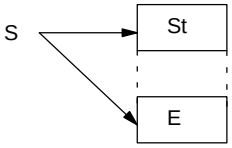
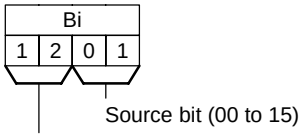
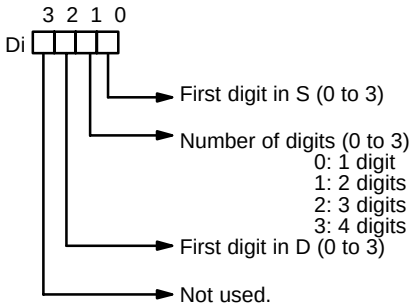
Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
FAILURE ALARM FAL, ↑FAL 	Outputs a FAL error number (N) and generates a non-fatal error when the execution condition is ON. N must be between 001 and 511. When the FAL number is generated, a corresponding bit is turned ON in the FAL output area, and the FAL number is output to A400. A 16 character message stored in M to M+7 can be output to a Peripheral Device if desired. The same FAL numbers are used for both FAL(006) and FALS(007), except 000. N can be set to 000 to reset a single FAL error designated by M or to reset all non-fatal errors by designating FFFF for M.	**N:** # **M:** CIO G A # DM	354
FAILURE ALARM FALS, ↑FALS 	Outputs the FAL error number (N) and generates a fatal error when the execution condition is ON. N must be between 001 and 511. The error number is output to A400. The same FAL numbers are used for both FAL(006) and FALS(007), except N cannot be defined as 000 with FALS(007). A 16 character message stored in M to M+7 can be output to a Peripheral Device if desired.	**N:** # **M:** CIO G A # DM	354
STEP DEFINE STEP 	When used with a control bit (B), defines the start of a new step and resets the previous step. When used without B, it defines the end of step execution. The steps referred to here are for step execution of ladder diagram programs and are not related to SFC.	**B:** CIO G A	368
STEP START SNXT 	Used with a control bit (B) to indicate the end of the previous step, reset the previous step, and start the step which has been defined with the same control bit. The steps referred to here are for step execution of ladder diagram programs and are not related to SFC.	**B:** CIO G A	368
NOT NOT 	NOT(010) is used along an instruction line to invert the current execution condition. It cannot be placed at the end of an instruction line, only between conditions or between a condition and a right-hand instruction.	None	125
KEEP KEEP, !KEEP 	Defines a bit (B) as a latch controlled by the set (S) and reset (R) inputs. B will turn ON when S turns ON and will turn OFF when R turns OFF. B will turn OFF if both S and R are ON.	**B:** CIO G A	132
REVERSIBLE COUNTER CNTR 	Increases or decreases the PV by one whenever the increment input (II) or decrement input (DI) condition, respectively, go from OFF to ON. S (set value): 0 to 9999; R: reset input. Each counter number can be used in only one counter instruction (CNT, CNTR(012), and TCNT(123)), unless the counters are never active simultaneously. The counter number can be entered as a constant or indirectly addressed by placing the address of the present value of the counter in an Index Register.	**N:** # **S:** CIO G A T/C # DM DR IR	156

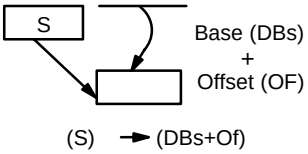
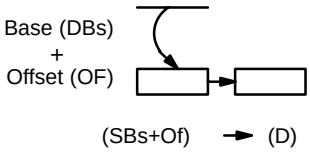
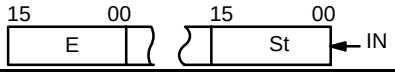
Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
DIFFERENTIATE UP DIFU, !DIFU (013) DIFU B	DIFU(013) turns ON the designated bit (B) for one cycle when the execution condition changes from OFF to ON.	**B:** CIO G A	127
DIFFERENTIATE DOWN DIFD, !DIFD (014) DIFD B	DIFD(014) turns ON the designated bit (B) for one cycle when the execution condition changes from ON to OFF.	**B:** CIO G A	127
HIGH-SPEED TIMER TIMH (015) TIMH N S	Creates a high-speed, decrementing, ON-delay timer. S (set value): 00.02 to 99.99 s. Each timer number can be used in only one timer instruction (TIM, TIMH(015), and TTIM(120)), unless the timers are never active simultaneously.	**N:** **S:** # CIO G G A T/C # DM DR IR	146
SET SET, !SET, ↑SET, !↓SET, !↑SET, !↓SET (016) SET B	Turns ON the designated bit when the execution condition is ON, and does not affect the status of the designated bit when the execution condition is OFF.	**B:** CIO G A	129
RSET RSET, !RSET, ↑RSET, !↓RSET, !↑RSET, !↓RSET (017) RSET B	Turns OFF the designated bit when the execution condition is ON, and does not affect the status of the designated bit when the execution condition is OFF.	**B:** CIO G A	129
CONDITION ON (V2 only) UP (018) UP	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.	None	123
CONDITION OFF (V2 only) DOWN (019) DOWN	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.	None	
COMPARE CMP, !CMP (020) CMP Cp_1 Cp_2	Compares the data in two 4-digit hexadecimal words (Cp_1 and Cp_2) and outputs result to the GR, EQ, or LE Flags. CMP(020) cannot be placed at the end of an instruction line, only between conditions or between a condition and a right-hand instruction.	**Cp_1:** **Cp_2:** CIO CIO G G A A T/C T/C # # DM DM DR DR IR IR	205
DOUBLE COMPARE CMPL (021) CMPL Cp_1 Cp_2	Compares the 8-digit hexadecimal content of Cp_1+1 and Cp_1 to the 8-digit hexadecimal content of Cp_2+1 and Cp_2 and outputs result to the GR, EQ, or LE Flags. CMPL(021) cannot be placed at the end of an instruction line, only between conditions or between a condition and a right-hand instruction.	**Cp_1:** **Cp_2:** CIO CIO G G A A T/C T/C # # DM DM	207

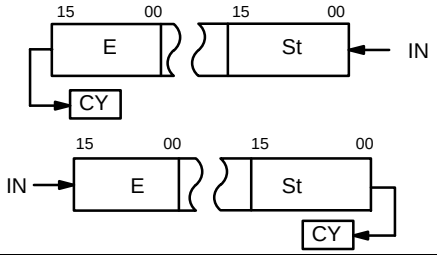
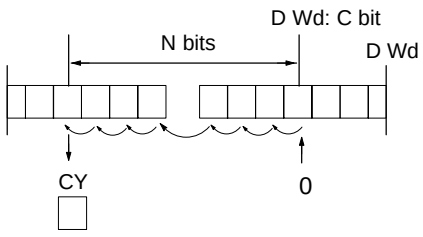
Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
BLOCK COMPARE BCMP, ↑BCMP (022) BCMP S CB R	Compares a 1-word binary value (S) with the 16 ranges given in the comparison table (CB is the starting word of the comparison block). If the value falls within any of the ranges, the corresponding bits in the result word (R) is turned ON (set to 1). The comparison block must be within one data area.  Lower limit ≤ S ≤ Upper limit → 1	**S:** CIO, G, A, T/C, #, DM, DR, IR **CB:** CIO, G, A, T/C, DM **R:** CIO, G, A, T/C, DM, DR, IR	208
TABLE COMPARE TCMP, ↑TCMP (023) TCMP S TB R	Compares a 4-digit hexadecimal value (S) with values in table consisting of 16 words (TB is the first word of the comparison table). If the value of S equals the content of a word in the table, the corresponding bit in result word (R) is set (1 for agreement, and 0 for disagreement). The table must be entirely within one data area.  1: agreement 0: disagreement	**S:** CIO, G, A, T/C, #, DM, DR, IR **TB:** CIO, G, A, T/C, DM **R:** CIO, G, A, T/C, DM, DR, IR	210
MULTIPLE COMPARE MCMP, ↑MCMP (024) MCMP TB_1 TB_2 R	Compares the contents of the 16 words TB_1 through TB_1+15 to the contents of the 16 words TB_2 through TB_2+15, and turns ON the corresponding bit in word R when the contents are not equal. (In the example below, the contents of TB_1 and TB_2 are not equal, and the contents of TB_1+1 and TB_2+1 are equal.) The table must be entirely within one data area.  0: agreement 1: disagreement	**TB_1:** CIO, G, A, T/C, DM **TB_2:** CIO, G, A, T/C, DM **R:** CIO, G, A, DM, DR, IR	211

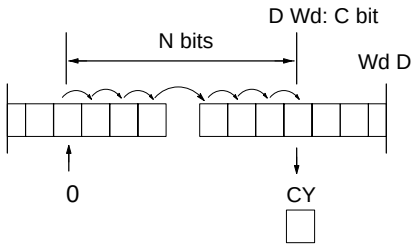
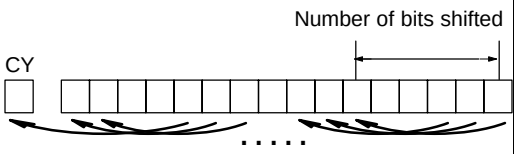
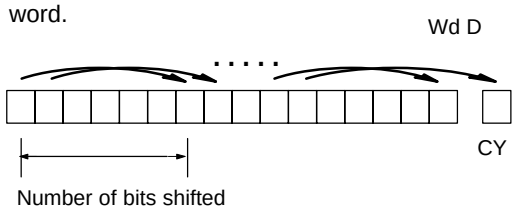
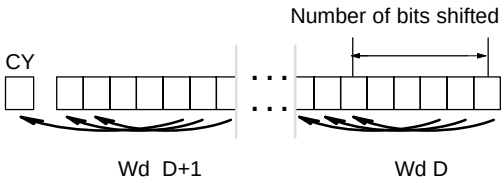
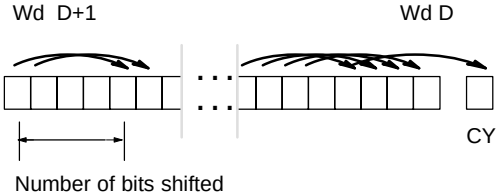
Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
EQUAL EQU, ↑EQU (025) EQU Cp_1 Cp_2	Compares the data in two 4-digit hexadecimal words (Cp_1 and Cp_2) and produces an ON execution condition if the contents are the same. EQU(025) cannot be placed at the end of an instruction line, only between conditions or between a condition and a right-hand instruction.	**Cp_1:** CIO G A T/C # DM DR IR **Cp_2:** CIO G A T/C # DM DR IR	212
SIGNED BINARY COMPARE (V2 only) CPS, !CPS (026) CPS S_1 S_2	Compares word data and constants in signed 16-bit binary. The content of S_1 is compared to that of S_2.	**S_1:** CIO G A T/C # DM DR IR **S_2:** CIO G A T/C # DM DR IR	215
DOUBLE SIGNED BINARY COMPARE (V2 only) CPSL, !CPSL (027) CPSL S_1 S_2	Compares word data and constants in signed 32-bit binary. The content of S_1 and S_1+1 is compared to that of S_2 and S_2+1.	**S_1:** CIO G A T/C # DM **S_2:** CIO G A T/C # DM	216
UNSIGNED COMPARE (V2 only) CMP, !CMP (028) CMP S_1 S_2	Compares word data and constants in four digits hexadecimal. The content of S_1 is compared to that of S_2.	**S_1:** CIO G A T/C # DM DR IR **S_2:** CIO G A T/C # DM DR IR	217
DOUBLE UNSIGNED COMPARE (V2 only) CMPL, !CMPL (029) CMPL S_1 S_2	Compares word data and constants in eight digits hexadecimal. The content of S_1 and S_1+1 is compared to that of S_2 and S_2+1.	**S_1:** CIO G A T/C # DM **S_2:** CIO G A T/C # DM	217
MOVE MOV, !MOV, ↑MOV, !↑MOV (030) MOV S D	Copies data from the source word (S) to the destination word (D).	**S:** CIO G A T/C # DM DR IR **D:** CIO G A T/C # DM DR IR	187
MOVE NOT MVN, ↑MVN (031) MVN S D	Copies the inverse of the data in the source word (S) to the destination word (D).	**S:** CIO G A T/C # DM DR IR **D:** CIO G A T/C # DM DR IR	188

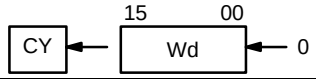
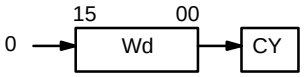
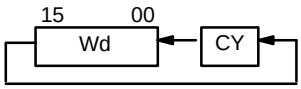
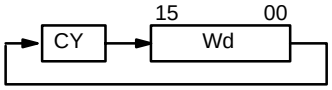
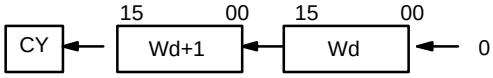
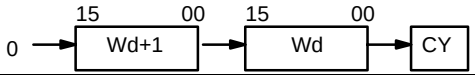
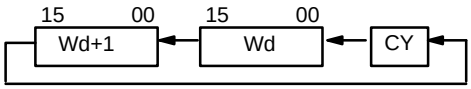
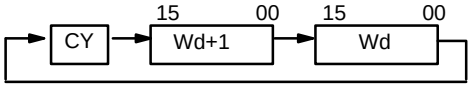
Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
DOUBLE MOVE MOVL, ↑MOVL (032) MOVL S D	Copies data from the source words (S and S+1) to the destination words (D and D+1).	**S:** CIO, G, A, T/C, #, DM **D:** CIO, G, A, T/C, DM	189
DOUBLE MOVE NOT MVNL, ↑MVNL (033) MVNL S D	Copies the inverse of the data in the source words (S and S+1) to destination words (D and D+1).	**S:** CIO, G, A, T/C, #, DM **D:** CIO, G, A, T/C, DM	190
DATA EXCHANGE XCHG, ↑XCHG (034) XCHG E_1 E_2	Exchanges the contents of two words (E_1 and E_2). 	**E_1:** CIO, G, A, T/C, DM, DR, IR **E_2:** CIO, G, A, T/C, DM, DR, IR	191
DOUBLE DATA EXCHANGE XCGL, ↑XCGL (035) XCGL E_1 E_2	Exchanges the contents of E_1 and E_1+1 with the contents of E_2 and E_2+1. 	**E_1:** CIO, G, A, T/C, DM **E_2:** CIO, G, A, T/C, DM	192
MOVE TO REGISTER MOVR, ↑MOVR (036) MOVR S D	Copies the 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 address of the timer or counter Completion Flag is copied to the Index Register.	**S:** CIO, G, A, T/C, TN, ST, DM **D:** IR**	193
MOVE QUICK MOVQ (037) MOVQ S D	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).	**S:** CIO, G, A, T/C, #, DM* **D:** CIO, G, A, T/C, DM*	194
MULTIPLE BIT TRANSFER (V2 only) XFRB, ↑XFRB (038) XFRB C S D	Transfers specified consecutive bits to a destination beginning with a specified bit in a specified word. 	**C:** CIO, G, A, T/C, #, DM, DR, IR **S:** CIO, G, A, T/C, DM **D:** CIO, G, A, DM	195

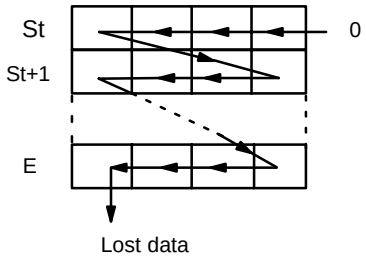
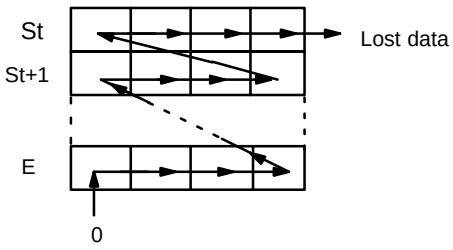
Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
BLOCK TRANSFER XFER, ↑XFER (040) XFER N S D	Moves the content of several consecutive source words (S gives the address of the starting source word) to consecutive destination words (D is the starting destination word). All source words must be in the same data area, as must all destination words. Transfers can be within one data area or between two data areas, but the source and destination words must not overlap. 	**N:** CIO, G, A, T/C, #, DM, DR, IR **S:** CIO, G, A, T/C, DM, DR*, IR* **D:** CIO, G, A, T/C, DM, DR*, IR*	197
BLOCK SET BSET, ↑BSET (041) BSET S St E	Copies the content of one word or constant (S) to several consecutive words (from the starting word, St, through to the ending word, E). St and E must be in the same data area. 	**S:** CIO, G, A, T/C, #, DM, DR, IR **St:** CIO, G, A, T/C, DM, DR*, IR* **E:** CIO, G, A, T/C, DM, DR*, IR*	198
MOVE BIT MOVB, ↑MOVB (042) MOVB S Bi D	Copies the data from the bit in the source word (S) specified in the bit designator (Bi) to the bit in the destination word (D) specified in the bit designator (Bi). The rightmost two digits of Bi designate the source bit and the leftmost two digits designate the destination bit. 	**S:** CIO, G, A, #, DM, DR, IR **Bi:** CIO, G, A, #, T/C, DM, DR, IR **D:** CIO, G, A, DM, DR, IR	199
MOVE DIGIT MOVD, ↑MOVD (043) MOVD S Di D	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. The rightmost digit in Di determines the first digit in S to be transferred. The next digit determines the number of digits to be transferred, and the third digit determines the first digit in D to which data will be transferred. 	**S:** CIO, G, A, T/C, #, DM, DR, IR **Di:** CIO, G, A, T/C, #, DM, DR, IR **D:** CIO, G, A, T/C, DM, DR, IR	201

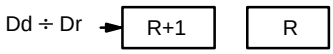
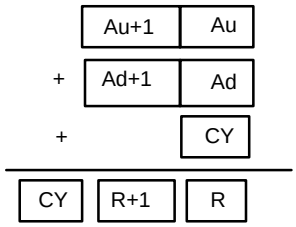
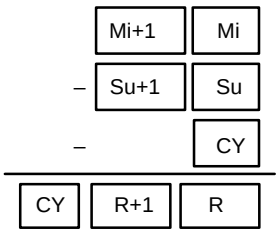
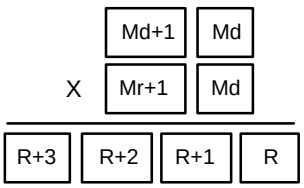
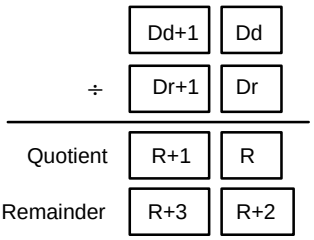
Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
SINGLE WORD DISTRIBUTE DIST, ↑DIST (044) DIST S DBs Of	Copies one word of source data (S) to the destination word whose address is given by the destination base word (DBs) plus the offset (Of). 	**S:** CIO, G, A, T/C, #, DM, DR, IR **DBs:** CIO, G, A, T/C, DM, DR*, IR* **Of:** CIO, G, A, T/C, DM, DR*, IR*	202
DATA COLLECT COLL, ↑COLL (045) COLL SBs Of D	Copies data from the source word and writes it to the destination word (D). The source word is determined by adding the offset (Of) to the address of the source base word (SBs). 	**SBs:** CIO, G, A, T/C, DM, DR*, IR* **Of:** CIO, G, A, T/C, #, DM, DR, IR **D:** CIO, G, A, T/C, DM, DR, IR	203
INTERBANK BLOCK TRANSFER (V2 only) BXFR, ↑BXFR (046) BXFR C S D	Transfers specified consecutive bits from the source bank to a destination beginning with a specified word in a specified bank. Words to transfer: C: Dest. bank no. | Source bank no. C+1: $x10^3$ | $x10^2$ | $x10^1$ | $x10^0$ C+2: 0 | 0 | 0 | $x10^4$	**C:** CIO, G, A, T/C, DM **S:** CIO, G, A, T/C, DM **D:** CIO, G, A, T/C, DM	204
MULTIPLE BIT SET (V2 only) SETA, ↑SETA (047) SETA D N_1 N_2	Turns ON a designated number of continuous bits, beginning from the designated bit of the designated word.	**D:** CIO, G, A **N_1:** CIO, G, A, T/C, #, DM, DR, IR **N2:** CIO, G, A, T/C, #, DM, DR, IR	130
MULTIPLE BIT RESET (V2 only) RSTA, ↑RSTA (048) RSTA D N_1 N_2	Turns OFF a designated number of continuous bits, beginning from the designated bit of the designated word.	**D:** CIO, G, A **N_1:** CIO, G, A, T/C, #, DM, DR, IR **N2:** CIO, G, A, T/C, #, DM, DR, IR	130
SHIFT REGISTER SFT I, P, R (050) SFT St E	Creates a bit shift register for data from the starting word (St) through to the ending word (E). I: input bit; P: shift pulse; R: reset input. St must be less than or equal to E. St and E must be in the same data area. 	**St:** CIO, G, A **E:** CIO, G, A	159

Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
REVERSIBLE SHIFT REGISTER SFTR, ↑SFTR (051) SFTR C St E	Shifts bits in the specified word or series of words either left or right. Starting (St) and ending words (E) must be specified. Control word (C) contains shift direction, reset input, and data input. (Bit 12: 0 = shift right, 1 = shift left. Bit 13 is the value shifted in, with the bit at the opposite end being moved to CY. Bit 14: 1 = shift enabled, 0 = shift disabled. If bit 15 is ON when SFTR(051) is executed with an ON condition, the entire shift register and CY will be set to zero.) St and E must be in the same data area and St must be less than or equal to E. 	**C:** CIO, G, A, DM, DR, IR **St:** CIO, G, A, DM **E:** CIO, G, A, DM	162
ASYNCHRONOUS SHIFT REGISTER ASFT, ↑ASFT (052) ASFT C St E	Exchanges the contents of adjacent words when the contents of one of the words is zero and the other is non-zero. 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 C contains 4000, non-zero words are shifted to the next higher address; if C contains 6000, non-zero words are shifted to the next lower address; and if C contains 8000, all words in the register are set to zero.	**C:** CIO, G, A, DM, DR, IR **St:** CIO, G, A, DM **E:** CIO, G, A, DM	163
WORD SHIFT WSFT, ↑WSFT (053) WSFT S St E	The data in the source word (S) is transferred into the starting word (St), and the data in the words from the starting word (St) through to the ending word (E) is shifted left in word units. The data in the ending word is lost. St must be less than or equal to E, and St and E must be in the same data area.	**S:** CIO, G, A, T/C, #, DM, DR, IR **St:** CIO, G, A, DM **E:** CIO, G, A, DM	165
SHIFT N-BIT DATA LEFT (V2 only) NSFL, ↑NSFL (054) NSFL D C N	Shifts the specified number of bits (i.e., the shift data length), from the beginning bit of the beginning word, one bit at a time to the left. A “0” is entered for the beginning bit (bit C of word D). The status of the Nth bit is then shifted to CY. 	**D:** CIO, G, A **C:** CIO, G, A, T/C, #, DM, DR, IR **N:** CIO, G, A, T/C, #, DM, DR, IR	166

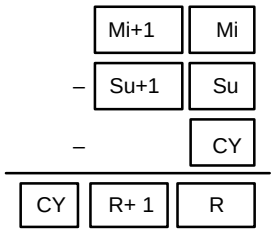
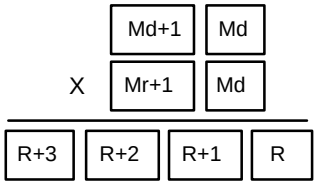
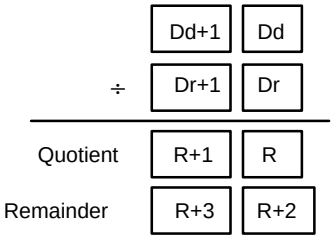
Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
SHIFT N-BIT DATA RIGHT (V2 only) NSFR, ↑NSFR (055) NSFR D C N	Shifts the specified number of bits (i.e., the shift data length), from the beginning bit of the beginning word, one bit at a time to the right. A "0" is entered for the beginning bit (bit C of word D). The status of the Nth bit is then shifted to CY. 	**D:** CIO, G, A **C:** CIO, G, A, T/C, #, DM, DR, IR **N:** CIO, G, A, T/C, #, DM, DR, IR	167
SHIFT N-BITS LEFT (V2 only) NASL, ↑NASL (056) NASL D C	Shifts to the left, for the specified number of bits, the status of the 16 bits in the specified word. 	**D:** CIO, G, A, DM, DR, IR **C:** CIO, G, A, T/C, #, DM, DR, IR	168
SHIFT N-BITS RIGHT (V2 only) NASR, ↑NASR (057) NASR D C	Shifts to the right, for the specified number of bits, the status of the 16 bits in the specified word. 	**D:** CIO, G, A, DM, DR, IR **C:** CIO, G, A, T/C, #, DM, DR, IR	169
DOUBLE SHIFT N-BITS LEFT (V2 only) NSLL, ↑NSLL (058) NSLL D C	Shifts to the left, for the specified number of bits, the status of the 32 bits in the specified words. 	**D:** CIO, G, A, DM **C:** CIO, G, A, T/C, #, DM	170
DOUBLE SHIFT N-BITS RIGHT (V2 only) NSRL, ↑NSRL (059) NSRL D C	Shifts to the right, for the specified number of bits, the status of the 32 bits in the specified words. 	**D:** CIO, G, A, DM **C:** CIO, G, A, T/C, #, DM	172

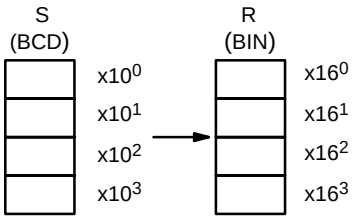
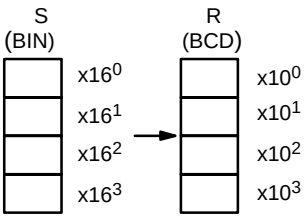
Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
ARITHMETIC SHIFT LEFT ASL, ↑ASL (060) ASL Wd	Each bit within a single word of data (Wd) is shifted one bit to the left, with zero written to bit 00 and bit 15 moved to CY. 	**Wd:** CIO G A DM	173
ARITHMETIC SHIFT RIGHT ASR, ↑ASR (061) ASR Wd	Each bit within a single word of data (Wd) is shifted one bit to the right, with zero written to bit 15 and bit 00 moved to CY. 	**Wd:** CIO G A DM DR IR	174
ROTATE LEFT ROL, ↑ROL (062) ROL Wd	Each bit within a single word of data (Wd) is moved one bit to the left, with bit 15 moving to carry (CY) and CY moving to bit 00. 	**Wd:** CIO G A DM DR IR	175
ROTATE RIGHT ROR, ↑ROR (063) ROR Wd	Each bit within a single word of data (Wd) is moved one bit to the right, with bit 00 moving to carry (CY) and CY moving to bit 15. 	**Wd:** CIO G A DM DR IR	176
DOUBLE SHIFT LEFT ASLL, ↑ASLL (064) ASLL Wd	Each bit within two consecutive words of data (Wd and Wd+1) is shifted one bit to the left, with zero written to bit 00 of Wd and bit 15 of Wd+1 moved to CY. 	**Wd:** CIO G A DM	177
DOUBLE SHIFT RIGHT ASRL, ↑ASRL (065) ASRL Wd	Each bit within two consecutive words of data (Wd and Wd+1) is shifted one bit to the right, with zero written to bit 15 of Wd+1 and bit 00 of Wd moved to CY. 	**Wd:** CIO G A DM	178
DOUBLE ROTATE LEFT ROLL, ↑ROLL (066) ROLL Wd	Each bit within two consecutive words of data (Wd and Wd+1) is moved one bit to the left, with bit 15 of Wd+1 moving to carry (CY) and CY moving to bit 00 of Wd. 	**Wd:** CIO G A DM	179
DOUBLE ROTATE RIGHT RORL, ↑RORL (067) RORL Wd	Each bit within two consecutive words of data (Wd and Wd+1) is moved one bit to the right, with bit 00 of Wd moving to carry (CY) and CY moving to bit 15 of Wd+1. 	**Wd:** CIO G A DM	182

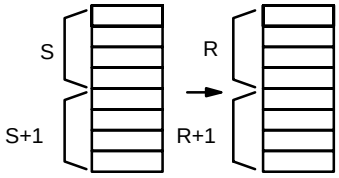
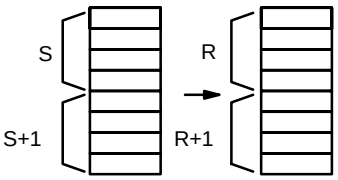
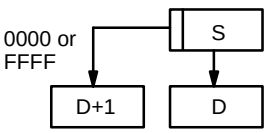
Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
SHIFT DIGIT LEFT SLD, ↑SLD (068) SLD St E	Shifts all data between the starting word (St) and ending word (E) one digit (four bits) to the left, writing zero into the rightmost digit of the starting word. St and E must be in the same data area. 	**St:** CIO G A DM **E:** CIO G A DM	185
SHIFT DIGIT RIGHT SRD, ↑SRD (069) SRD St E	Shifts all data between starting word (St) and ending word (E) one digit (four bits) to the right, writing zero into the leftmost digit of the ending word. St and E must be in the same data area. 	**St:** CIO G A DM **E:** CIO G A DM	186
BCD ADD ADD, ↑ADD (070) ADD Au Ad R	Adds two 4-digit BCD values (Au and Ad) and the content of CY, and outputs the result to the specified result word (R). Au + Ad + CY → CY R	**Au:** CIO G A T/C # DM DR IR **Ad:** CIO G A T/C # DM DR IR **R:** CIO G A DM DR IR	250
BCD SUBTRACT SUB, ↑SUB (071) SUB Mi Su R	Subtracts both the 4-digit BCD subtrahend (Su) and the content of CY from the 4-digit BCD minuend (Mi) and outputs the result to the specified result word (R). Mi – Su – CY → CY R	**Mi:** CIO G A T/C # DM DR IR **Su:** CIO G A T/C # DM DR IR **R:** CIO G A DM DR IR	251
BCD MULTIPLY MUL, ↑MUL (072) MUL Md Mr R	Multiplies the 4-digit BCD multiplicand (Md) and 4-digit BCD multiplier (Mr) and outputs the result to the specified result words (R and R+1). R and R+1 must be in the same data area. Md x Mr → R+1 R	**Md:** CIO G A T/C # DM DR IR **Mr:** CIO G A T/C # DM DR IR **R:** CIO G A DM	253

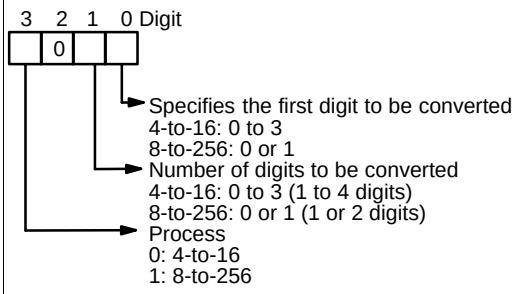
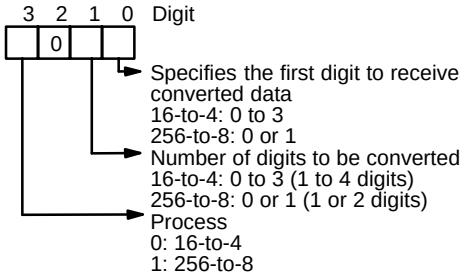
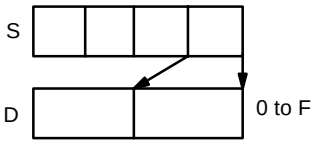
Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
BCD DIVIDE DIV, ↑DIV (073) DIV Dd Dr R	Divides the 4-digit BCD dividend (Dd) by the 4-digit BCD divisor (Dr) and outputs the result to the specified result words. R receives the quotient; R+1 receives the remainder. R and R+1 must be in the same data area. 	**Dd:** CIO G A T/C # DM DR IR **Dr:** CIO G A T/C # DM DR IR **R:** CIO G A DM	254
DOUBLE BCD ADD ADDL, ↑ADDL (074) ADDL Au Ad R	Adds two 8-digit BCD values (2 words each) and the content of CY and outputs the result to the specified result words. All words for any one operand must be in the same data area. 	**Au:** CIO G A T/C # DM **Ad:** CIO G A T/C # DM **R:** CIO G A DM DR IR	255
DOUBLE BCD SUBTRACT SUBL, ↑SUBL (075) SUBL Mi Su R	Subtracts both the 8-digit BCD subtrahend and the content of CY from an 8-digit BCD minuend, and outputs the result to the specified result words. All words for any one operand must be in the same data area. 	**Mi:** CIO G A T/C # DM **Su:** CIO G A T/C # DM **R:** CIO G A DM DR IR	256
DOUBLE BCD MULTIPLY MULL, ↑MULL (076) MULL Md Mr R	Multiplies the 8-digit BCD multiplicand and 8-digit BCD multiplier, and outputs the result to the specified result words. All words for any one operand must be in the same data area. 	**Md:** CIO G A T/C # DM **Mr:** CIO G A T/C # DM **R:** CIO G A DM	257
DOUBLE BCD DIVIDE DIVL, ↑DIVL (077) DIVL Dd Dr R	Divides the 8-digit BCD dividend by an 8-digit BCD divisor, and outputs the result to the specified result words. All words for any one operand must be in the same data area. 	**Dd:** CIO G A T/C # DM **Dr:** CIO G A T/C # DM **R:** CIO G A DM	258
SET CARRY STC, ↑STC (078) STC	Sets the Carry Flag (i.e., turns ON A50004).	None	250

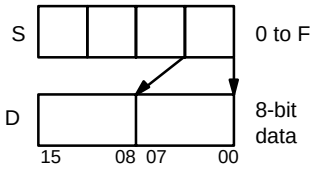
Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
CLEAR CARRY CLC, ↑CLC (079) CLC	Clears the Carry Flag (i.e., turns OFF A50004).	None	250
BINARY ADD ADB, ↑ADB (080) ADB Au Ad R	Adds two 4-digit hexadecimal values (Au and Ad) and content of CY and outputs the result to the specified result word (R). Au + Ad + CY → CY R	**Au:** CIO G A T/C # DM DR IR **Ad:** CIO G A T/C # DM DR IR **R:** CIO G A DM DR IR	261
BINARY SUBTRACT SBB, ↑SBB (081) SBB Mi Su R	Subtracts both the 4-digit hexadecimal subtrahend (Su) and content of CY from the 4-digit hexadecimal minuend (Mi) and outputs the result to the specified result word (R). Mi – Su – CY → CY R	**Mi:** CIO G A T/C # DM DR IR **Su:** CIO G A T/C # DM DR IR **R:** CIO G A DM DR IR	262
BINARY MULTIPLY MLB, ↑MLB (082) MLB Md Mr R	Multiplies the 4-digit hexadecimal multiplicand (Md) and 4-digit hexadecimal multiplier (Mr) and outputs the result to the specified result words (R and R+1). R and R+1 must be in the same data area. Md x Mr → R+1 R	**Md:** CIO G A T/C # DM DR IR **Mr:** CIO G A T/C # DM DR IR **R:** CIO G A DM	264
BINARY DIVIDE DVB, ↑DVB (083) DVB Dd Dr R	Divides the 4-digit hexadecimal dividend (Dd) by the 4-digit hexadecimal divisor (Dr) and outputs the result to the specified result words. R receives the quotient; R+1 receives the remainder. R and R+1 must be in the same data area. Dd ÷ Dr → R+1 R	**Dd:** CIO G A T/C # DM DR IR **Dr:** CIO G A T/C # DM DR IR **R:** CIO G A DM	265
DOUBLE BINARY ADD ADBL, ↑ADBL (084) ADBL Au Ad R	Adds two 8-digit hexadecimal values (2 words each) and the content of CY, and outputs the result to the specified result words. All words for any one operand must be in the same data area. CY will be set (acting as the 9th digit) if the result is greater than FFFF. Au+1 Au + Ad+1 Ad + CY CY R+1 R	**Au:** CIO G A T/C # DM **Ad:** CIO G A T/C # DM **R:** CIO G A DM DR IR	266

Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
DOUBLE BINARY SUBTRACT SBBL, ↑SBBL (085) SBBL Mi Su R	Subtracts both the 8-digit hexadecimal subtrahend and the content of CY from an 8-digit hexadecimal minuend and outputs the result to the specified result words. All words for any one operand must be in the same data area. The Carry Flag will be set to indicate a negative result. 	**Mi:** CIO G A T/C # DM **Su:** CIO G A T/C # DM **R:** CIO G A DM DR IR	268
DOUBLE BINARY MULTIPLY MLBL, ↑MLBL (086) MLBL Md Mr R	Multiplies the 8-digit hexadecimal multiplicand and 8-digit hexadecimal multiplier and outputs the result to the specified result words. All words for any one operand must be in the same data area. 	**Md:** CIO G A T/C # DM **Mr:** CIO G A T/C # DM **R:** CIO G A DM	269
DOUBLE BINARY DIVIDE DVBL, ↑DVBL (087) DVBL Dd Dr R	Divides the 8-digit hexadecimal dividend by an 8-digit hexadecimal divisor and outputs the result to the specified result words. All words for any one operand must be in the same data area. 	**Dd:** CIO G A T/C # DM **Dr:** CIO G A T/C # DM **R:** CIO G A DM	270
INCREMENT BCD INC, ↑INC (090) INC Wd	Increments the value of a 4-digit BCD word (Wd) by one, without affecting carry (CY).	**Wd:** CIO G A DM DR IR	314
DECREMENT BCD DEC, ↑DEC (091) DEC Wd	Decrements the value of a 4-digit BCD word (Wd) by one, without affecting carry (CY).	**Wd:** CIO G A DM DR IR	314
INCREMENT BINARY INCB, ↑INCB (092) INCB Wd	Increments the value of a 4-digit hexadecimal word (Wd) by one, without affecting carry (CY).	**Wd:** CIO G A DM DR IR	315

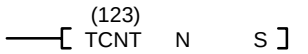
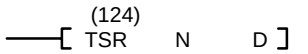
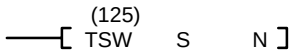
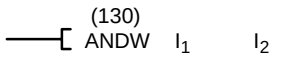
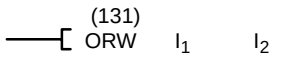
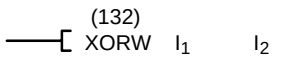
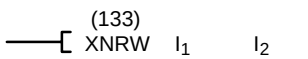
Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
DECREMENT BINARY DECB, ↑DECB (093) DECB Wd	Decrements the value of a 4-digit hexadecimal word (Wd) by one, without affecting carry (CY).	**Wd:** CIO G A DM DR IR	316
DOUBLE INCREMENT BCD INCL, ↑INCL (094) INCL Wd	Increments the 8-digit BCD value contained in Wd+1 and Wd, without affecting carry (CY).	**Wd:** CIO G A DM	316
DOUBLE DECREMENT BCD DECL, ↑DECL (095) DECL Wd	Decrements the 8-digit BCD value contained in Wd+1 and Wd, without affecting carry (CY).	**Wd:** CIO G A DM	317
DOUBLE INCREMENT BINARY INBL, ↑INBL (096) INBL Wd	Increments the 8-digit hexadecimal value contained in Wd+1 and Wd, without affecting carry (CY).	**Wd:** CIO G A DM	317
DOUBLE DECREMENT BINARY DCBL, ↑DCBL (097) DCBL Wd	Decrements the 8-digit hexadecimal value contained in Wd+1 and Wd, without affecting carry (CY).	**Wd:** CIO G A DM	318
BCD-TO-BINARY BIN, ↑BIN (100) BIN S R	Converts 4-digit, BCD data in source word (S) into 16-bit binary data, and outputs converted data to result word (R). 	**S:** CIO G A T/C DM DR IR **R:** CIO G A DM DR IR	219
BINARY-TO-BCD BCD, ↑BCD (101) BCD S R	Converts binary data in source word (S) into BCD, and outputs converted data to result word (R). 	**S:** CIO G A T/C DM DR IR **R:** CIO G A DM DR IR	220

Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
DOUBLE BCD-TO-DOUBLE BINARY BINL, ↑BINL (102) BINL S R	Converts the BCD value of the two source words (S: starting word) into binary and outputs the converted data to the two result words (R: starting word). All words for any one operand must be in the same data area. 	**S:** CIO, G, A, T/C, DM **R:** CIO, G, A, DM	221
DOUBLE BINARY-TO-DOUBLE BCD BCDL, ↑BCDL (103) BCDL S R	Converts the binary value of the two source words (S: starting word) into eight digits of BCD data, and outputs the converted data to the two result words (R: starting result word). Both words for any one operand must be in the same data area. 	**S:** CIO, G, A, T/C, DM **R:** CIO, G, A, DM	222
2'S COMPLEMENT NEG, ↑NEG (104) NEG S D	Takes the 2's complement of the 4-digit hexadecimal content of the source word (S) 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. GR (A50005) is ON when the content of S is $8000. EQ (A50006) is ON when the content of S is 0. N (A50008) is the same as the status of bit 15 of R after execution.	**S:** CIO, G, A, T/C, #, DM, DR, IR **D:** CIO, G, A, DM, DR, IR	223
DOUBLE 2'S COMPLEMENT NEGL, ↑NEGL (105) NEGL S D	Takes the 2's complement of the 8-digit hexadecimal content of the source words (S and S+1) and outputs the result to the result words (R and R+1). This operation is effectively the same as subtracting S and S+1 from $0000 0000 and outputting the result to R and R+1. GR (A50005) is ON when the content of S is $8000. EQ (A50006) is ON when the content of S is 0. N (A50008) is the same as the status of bit 15 of R after execution.	**S:** CIO, G, A, T/C, #, DM, DR, IR **D:** CIO, G, A, DM, DR, IR	224
SIGN SIGN, ↑SIGN (106) SIGN S D	Places FFFF into D+1 if the sign bit (bit 15) of S is 1, places 0000 into D+1 if the sign bit is 0, and copies the content of S to D. 	**S:** CIO, G, A, T/C, #, DM, DR, IR **D:** CIO, G, A, DM	225

Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
DATA DECODER MLPX, ↑MLPX (110) MLPX S Di R	Converts up to four hexadecimal digits in the source word (S) into decimal values from 0 to 15 and turns ON the corresponding bit(s) in the result word(s) (R), or converts up to two 8-bit values in the source word (S) into decimal values from 0 to 256 and turns on the corresponding bit in sets of 16 result words. There is one result word or set of 16 result words for each converted value. Operation is controlled by Di as follows:  	**S:** CIO G A T/C DM DR IR **Di:** CIO G A T/C # DM DR IR **R:** CIO G A DM	226
DATA ENCODER DMPX, ↑DMPX (111) DMPX SB R Di	Determines the position of the leftmost ON bit in the source word(s) or set of 16 source words (starting word: SB) and then turns ON the corresponding bit(s) in the specified digit of the result word (R) or outputs the sequential value of the bit as an 8-bit value to R. One 4-bit or 8-bit value is used for each source word or set of 16 source words. Operation is controlled by Di as follows:  	**SB:** CIO G A T/C DM **R:** CIO G A DM DR IR **Di:** CIO G A T/C # DM DR IR	228
7-SEGMENT DECODER SDEC, ↑SDEC (112) SDEC S Di D	Converts hexadecimal digits from the source word (S) into 7-segment display data. Results are placed in consecutive half-words, starting at the first destination word (D). Di gives digit and destination details. The rightmost digit gives the first digit to be converted. The next digit to the left gives the number of digits to be converted minus 1. If the third digit is 1, the first converted data is transferred to left half of the first destination word. If it is 0, the transfer is to the right half.  	**S:** CIO G A T/C DM DR IR **Di:** CIO G A T/C # DM DR IR **D:** CIO G A DM	231

Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
ASCII CONVERT ASC, ↑ASC (113) ASC S Di D	Converts hexadecimal digits from the source word (S) into 8-bit ASCII values, starting at leftmost or rightmost half of the starting destination word (D). The rightmost digit of Di designates the first source digit. The next digit to the left gives the number of digits to be converted. The third digit specifies the whether the data is to be transferred to the rightmost (0) or leftmost (1) half of the first destination word. The leftmost digit specifies parity: 0: none, 1: even, or 2: odd.  	**S:** CIO G A T/C DM DR IR **Di:** CIO G A T/C # DM DR IR **D:** CIO G A DM	234
BIT COUNTER BCNT, ↑BCNT (114) BCNT N St R	Counts the number of ON bits in one or more words (St is the beginning source word) and outputs the result to the specified result word (R). N must be in BCD and gives the number of words to be counted. All words in which bits are to be counted must be in the same data area.	**N:** CIO G A T/C DM # DR IR **St:** CIO G A T/C DM **R:** CIO G A T/C DM DR IR	236
COLUMN TO LINE LINE, ↑LINE (115) LINE S Bi D	Copies bit column Bi from the 16 word set S through S+15 to the 16 bits of word D (00 to 15). In other words, bit Bi of S+n is copied to bit n of D for n = 00 to 15.	**S:** CIO G A T/C DM **Bi:** CIO G A T/C # DM DR IR **D:** CIO G A DM DR IR	237
LINE TO COLUMN COLM, ↑COLM (116) COLM S D Bi	Copies the 16 bits of word S (00 to 15) to bit column Bi of the 16 word set D through D+15. In other words, bit n of S is copied to bit Bi of D+n, for n=00 to 15.	**S:** CIO G A T/C # DM **D:** CIO G A DM **Bi:** CIO G A T/C # DM DR IR	238

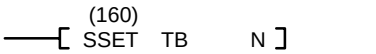
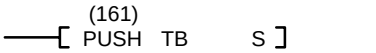
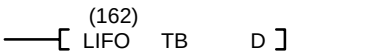
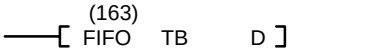
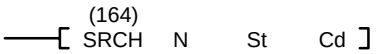
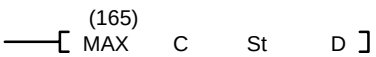
Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
ASCII TO HEX (V2 only) HEX, ↑HEX (117) HEX S C D	Converts the data in specified words from ASCII to hexadecimal, and outputs the result to a specified destination word. MSB LSB C Specifies the first digit to be output. Number of digits to be converted. 0: 1 digit 1: 2 digits 2: 3 digits 3: 4 digits First ASCII data to be converted. 0: Rightmost 8 bits 1: Leftmost 8 bits Parity 0: None 1: Even 2: Odd	**S:** CIO, G, A, T/C, DM **C:** CIO, G, A, T/C, #, DM, DR, IR **D:** CIO, G, A, DM	239
ACCUMULATIVE TIMER TTIM I (120) TTIM N S R	Creates a totalizing timer. I is the timer input; R is the reset input. The timer PV is incremented 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. The time is incremented from zero in units of 0.1 second, and accuracy is +0.0/–0.1 second. The set value (S) must be between 000.0 and 999.9 s. The decimal point is not entered. Each timer number (BCD) can be used in only one timer instruction (TIM, TIMH(015), and TTIM(120)), unless the timers are never active simultaneously. The timer number can be designated as a constant or indirectly addressed by placing the address of the present value for the timer in an Index Register.	**N:** # **S:** CIO, G, A, T/C, #, DM, DR, IR	148
LONG TIMER TIML (121) TIML D_1 D_2 S	Creates a decrementing ON-delay timer that can time up to 9,999,999.9 s (approx. 115 days). The timer is activated when its execution condition goes ON and is reset (to SV) when the execution condition goes OFF. Once activated, TIML(121) measures in units of 0.1 second from the SV, and accuracy is +0.0/–0.1 second. The Completion Flag is bit 00 of D_1. The PV is output to D_2 and D_2+1, and the SV is set in S and S+1.	**D_1:** CIO, G, A, T/C, DM **D_2:** CIO, G, A, T/C, DM **S:** CIO, G, A, T/C, #, DM	150
MULTI-OUTPUT TIMER MTIM (122) MTIM D_1 D_2 S	Creates a totalizing timer that can have up to eight pairs of set values and Completion Flags. The timer is activated when the execution condition is ON, and the reset bit (bit 08 of D_1) goes from ON to OFF. Each time the instruction is executed, the PV (content of D_2) 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 (bits 00 through 07 of D_1) is turned ON. The same MTIM(122) can be input into the program several times to increase comparison frequency and therefore accuracy.	**D_1:** CIO, G, A, T/C, DM **D_2:** CIO, G, A, T/C, DM, DR, IR **S:** CIO, G, A, T/C, #, DM	151

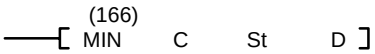
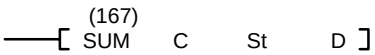
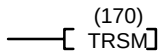
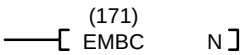
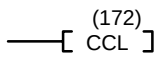
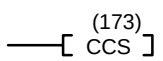
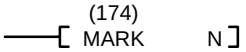
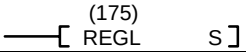
Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
TRANSITION COUNTER TCNT 	Computes the number of times that a transition program is executed and turns ON the Transition Flag when the preset count is reached. Each counter number (BCD) can be used in only one counter instruction (CNT, CNTR(012), and TCNT(123)), unless the counters are never active simultaneously.	**N:** # **S:** CIO G A T/C DM # DR IR	434
READ STEP TIMER TSR, ↑TSR 	Reads the present value of the step timer for step N and writes to D.	**N:** CIO G A T/C DM # DR IR **D:** CIO G A T/C DM DR IR	435
WRITE STEP TIMER TSW, ↑TSW 	Changes the present value of the step timer for step N to the value in S.	**S:** CIO G A T/C DM DR IR **N:** CIO G A T/C DM # DR IR	436
LOGICAL AND ANDW, ↑ANDW 	Logically ANDs two 16-bit input words (I_1 and I_2) and sets the bits in the result word (R) if the corresponding bits in the input words are both ON.	**I_1:** CIO G A T/C # DM DR IR **I_2:** CIO G A T/C # DM DR IR **R:** CIO G A DM DR IR	341
LOGICAL OR ORW, ↑ORW 	Logically ORs two 16-bit input words (I_1 and I_2) and sets the bits in the result word (R) when one or both of the corresponding bits in the input words is/are ON.	**I_1:** CIO G A T/C # DM DR IR **I_2:** CIO G A T/C # DM DR IR **R:** CIO G A DM DR IR	342
EXCLUSIVE OR XORW, ↑XORW 	Exclusively ORs two 16-bit input words (I_1 and I_2) and sets the bits in the result word (R) when the corresponding bits in input words differ in status.	**I_1:** CIO G A T/C # DM DR IR **I_2:** CIO G A T/C # DM DR IR **R:** CIO G A DM DR IR	343
EXCLUSIVE NOR XNRW, ↑XNRW 	Exclusively NORs two 16-bit input words (I_1 and I_2) and sets the bits in the result word (R) when the corresponding bits in both input words have the same status.	**I_1:** CIO G A T/C # DM DR IR **I_2:** CIO G A T/C # DM DR IR **R:** CIO G A DM DR IR	343

Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
DOUBLE LOGICAL AND ANDL, ↑ANDL (134) ANDL I_1 I_2 R	Logically ANDs the contents of I_1 and I_1+1 with the contents of I_2 and I_2+1 and sets the bits in the result words (R and R+1) if the corresponding bits in the input words are both ON.	**I_1:** CIO, G, A, T/C, #, DM **I_2:** CIO, G, A, T/C, #, DM **R:** CIO, G, A, DM	344
DOUBLE LOGICAL OR ORWL, ↑ORWL (135) ORWL I_1 I_2 R	Logically ORs the contents of I_1 and I_1+1 with the contents of I_2 and I_2+1 and sets the bits in the result words (R and R+1) when one or both of the corresponding bits in the input words are ON.	**I_1:** CIO, G, A, T/C, #, DM **I_2:** CIO, G, A, T/C, #, DM **R:** CIO, G, A, DM	345
DOUBLE EXCLUSIVE OR XORL, ↑XORL (136) XORL I_1 I_2 R	Exclusively ORs the contents of I_1 and I_1+1 with the contents of I_2 and I_2+1 and sets the bits in the result words (R and R+1) when the corresponding bits in input words differ in status.	**I_1:** CIO, G, A, T/C, #, DM **I_2:** CIO, G, A, T/C, #, DM **R:** CIO, G, A, DM	346
DOUBLE EXCLUSIVE NOR XNRL, ↑XNRL (137) XNRL I_1 I_2 R	Exclusively NORs the contents of I_1 and I_1+1 with the contents of I_2 and I_2+1 and sets the bits in the result words (R and R+1) when the corresponding bits in both input words have the same status.	**I_1:** CIO, G, A, T/C, #, DM **I_2:** CIO, G, A, T/C, #, DM **R:** CIO, G, A, DM	346
COMPLEMENT COM, ↑COM (138) COM Wd	Inverts the bit status of one word (Wd) of data, changing 0s to 1s and 1s to 0s. Wd → $\overline{Wd}$	**Wd:** CIO, G, A, DM, DR, IR	347
DOUBLE COMPLEMENT COML, ↑COML (139) COML Wd	Inverts the bit status of two consecutive words of data (Wd and Wd+1), changing 0s to 1s and 1s to 0s. Wd and Wd+1 → $\overline{Wd}$ and $\overline{Wd+1}$	**Wd:** CIO, G, A, DM	348
SQUARE ROOT ROOT, ↑ROOT (140) ROOT Sq R	Computes the square root of an 8-digit BCD value (Sq and Sq+1) and outputs the truncated 4-digit integer result to the specified result word (R). Sq and Sq+1 must be in the same data area. $\sqrt{\text{Sq+1 \| Sq}}$ → R	**Sq:** CIO, G, A, T/C, #, DM **R:** CIO, G, A, DM, DR, IR	323

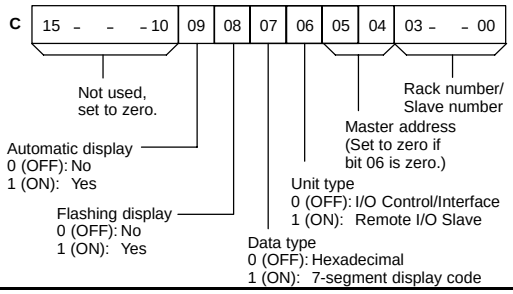
Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
FLOATING POINT DIVIDE FDIV, ↑FDIV (141) FDIV Dd Dr R	Divides one floating point value by another and outputs a floating point result. The rightmost seven digits of each set of two words (eight digits) are used for mantissa, and the leftmost digit is used for the exponent and its sign. Bits 12 to 14 give the exponent value between 0 and 7. If bit 15 is 0, the exponent is positive; if it's 1, the exponent is negative. Dd+ 1 Dd ÷ Dr+ 1 Dr R+1 R	**Dd:** CIO G A T/C DM **Dr:** CIO G A T/C DM **R:** CIO G A DM	326
ARITHMETIC PROCESS APR, ↑APR (142) APR C S R	The operation of APR(142) depends on the control word C. If C is #0000 or #0001, APR(142) computes sin(Θ) or cos(Θ) and outputs the result to R. Θ is contained in S in units of tenths of degrees ($0° \leq \Theta \leq 90°$). If C is an address, APR(142) computes the value of a function from the data in S and outputs the result to R. The function is a series of line segments (which can approximate a curve) entered in advance in a table beginning at C.	**C:** CIO G A # DM DR IR **S:** CIO G A T/C # DM DR IR **R:** CIO G A DM DR IR	328
HOURS TO SECONDS SEC, ↑SEC (143) SEC S R	Converts a time given in hours, minutes, and seconds (S and S+1) to an equivalent time in seconds only (R and R+1). S and S+1 must be BCD and within one data area. R and R+1 must also be within one data area.	**S:** CIO G A T/C # DM **R:** CIO G A DM	349
SECONDS TO HOURS HMS, ↑HMS (144) HMS S R	Converts a time given in seconds (S and S+1) to an equivalent time in hours, minutes, and seconds (R and R+1). S and S+1 must be BCD between 0 and 35,999,999 and within the same data area. R and R+1 must also be within one data area.	**S:** CIO G A T/C # DM **R:** CIO G A DM	350
CALENDAR ADD CADD, ↑CADD (145) CADD C T R	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. **Word / Bits / Contents** C: 00 to 07 Seconds; 08 to 15 Minutes C+1: 00 to 07 Hours; 08 to 15 Day of month C+2: 00 to 07 Month; 08 to 15 Year **Word / Bits / Contents** T: 00 to 07 Seconds; 08 to 15 Minutes T+1: 00 to 07 Hours	**C:** CIO G A T/C DM **S:** CIO G A T/C # DM **R:** CIO G A DM	350
CALENDAR SUBTRACT CSUB, ↑CSUB (146) CSUB C T R	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. The time and calendar formats are the same as in CADD(145).	**C:** CIO G A T/C DM **S:** CIO G A T/C # DM **R:** CIO G A DM	352

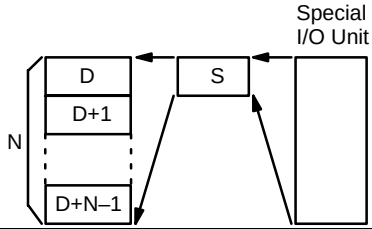
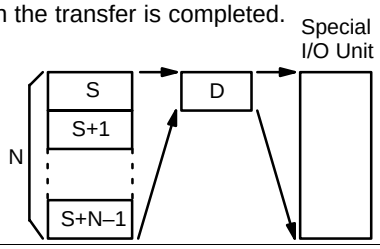
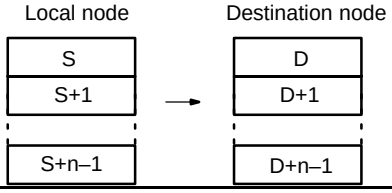
Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
SUBROUTINE ENTRY SBN (150) SBN N	Calls subroutine N. Moves program operation to the specified subroutine. N must be BCD between 000 and 999 for the CV1000, CV2000, or CVM1-CPU11/21-EV2 or between 000 and 099 for the CV500 or CVM1-CPU01-EV2.	**N:** #	377
SUBROUTINE CALL SBS, ↑SBS (151) SBS N	Marks the start of subroutine N. N must be BCD between 000 and 999 for the CV1000, CV2000, or CVM1-CPU11/21-EV2 or between 000 and 099 for the CV500 or CVM1-CPU01-EV2.	**N:** #	378
SUBROUTINE RETURN RET (152) RET	Marks the end of a subroutine and returns control to the main program.	None	377
INTERRUPT MASK MSKS, ↑MSKS (153) MSKS N S	N must be between 0 and 5 for the CV1000, CV2000, or CVM1-CPU11/21-EV2 or between 0 and 4 for the CV500 or CVM1-CPU01-EV2. If N is 0 to 3, it designates the unit number of the Interrupt Input Unit 0 to 3, the bits of the designated Interrupt Input Unit corresponding to ON bits in S are masked, and bits corresponding to OFF bits in S are unmasked. If N is 4 or 5, a scheduled interrupt is designated and the time interval for the scheduled interrupt is set according to the value in S and the time unit set in the PC Setup. CLI(154) should be used to set the time to the first scheduled interrupt. Unstable operation might result if the time to the first interrupt is not set.	**N:** # **S:** CIO G A T/C # DM DR IR	385
CLEAR INTERRUPT CLI, ↑CLI (154) CLI N S	N must be between 0 and 5 for the CV1000, CV2000, or CVM1-CPU11/21-EV2 or between 0 and 4 for the CV500 or CVM1-CPU01-EV2. If N is 0 to 3, it designates the unit number of the Interrupt Input Unit and the interrupt inputs from the designated Interrupt Input Unit corresponding to ON bits in S are cleared. If N is 4 or 5, a scheduled interrupt is designated and the time to the first interrupt is set according to the value in S and the time unit set in the PC Setup.	**N:** # **S:** CIO G A T/C # DM DR IR	386
READ MASK MSKR, ↑MSKR (155) MSKR N D	If N is 0 to 3, writes the current mask status of the designated Interrupt Input Unit into D. If N is 4 or 5, writes the scheduled interrupt interval into D. N must be between 0 and 5 for the CV1000, CV2000, or CVM1-CPU11/21-EV2 or between 0 and 4 for the CV500 or CVM1-CPU01-EV2.	**N:** # **D:** CIO G A DM DR IR	388
MACRO (V2 only) MCRO, ↑MCRO (156) MCRO N S D	MCRO(156) allows a single subroutine to replace several subroutines that have identical structure but different operands. The subroutine number (N) range is 000 to 999.	**S:** CIO G A T/C DM **D:** CIO G A T/C DM	*380*

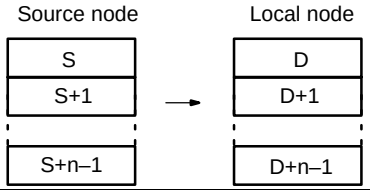
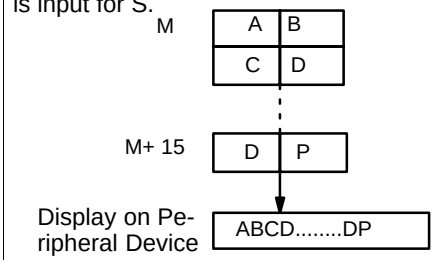
Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
SET STACK SSET, ↑SSET 	Defines a stack from TB to TB+N–1 and resets to zero all words from TB+2 to TB+N–1. TB contains the memory address for TB+N–1, and TB+1 contains the memory address of TB+2. TB+1 is called the stack pointer, and contains the memory address for TB+2 after SSET(160) is executed. N must be BCD between #0003 and #9999.	**TB:** CIO G A DM **N:** CIO G A T/C # DM DR IR	389
PUSH ONTO STACK PUSH, ↑PUSH 	Copies the data from the source word (S) to the word indicated by the stack pointer (TB+1). The memory address in the stack pointer is then incremented by one. TB must be the first address of a stack defined using SSET(160).	**TB:** CIO G A DM **S:** CIO G A T/C # DM DR IR	390
LAST IN FIRST OUT LIFO, ↑LIFO 	Decrements the memory address in the stack pointer (TB+1) by one and then copies the data from the word indicated by the stack pointer (the last written to the table) to the destination word (D). The stack pointer is the only word changed in the stack. TB must be the first address of a stack defined using SSET(160).	**TB:** CIO G A DM **D:** CIO G A DM DR IR	391
FIRST IN FIRST OUT FIFO, ↑FIFO 	Writes zero 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 TB+2 (the first written to the stack) to the destination word (D). The memory address in the stack pointer (TB+1) is then decremented by one. TB must be the first address of a stack defined using SSET(160).	**TB:** CIO G A DM **D:** CIO G A DM DR IR	392
DATA SEARCH SRCH, ↑SRCH 	Searches the range of memory from St to St+N–1 for addresses that contain the comparison data (Cd). If the content of an address within the range matches 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, only the lowest address containing the comparison data is written to IR0. If none of the addresses within the range contains the comparison data, the EQ Flag (A50006) is turned OFF and IR0 is left unchanged.	**N:** CIO G A T/C # DM DR IR **St:** CIO G A T/C DM **Cd:** CIO G A T/C # DM DR IR	365
FIND MAXIMUM MAX, ↑MAX 	Searches the range of memory from St to St+N–1 for the address that contains the maximum value and outputs that value to the destination word (D). 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 hexadecimal values, and when it is ON the data is treated as signed hexadecimal values. When bit 14 of C is OFF, the address of the word containing the maximum value will not be output to IR0; when it is ON, the value will be output to IR0.	**C:** CIO G A # DM DR IR **St:** CIO G A T/C DM **D:** CIO G A DM DR IR	319

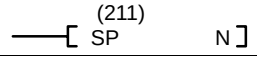
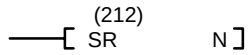
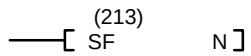
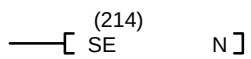
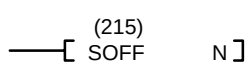
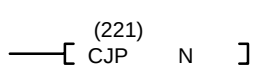
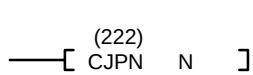
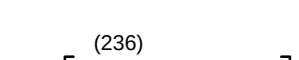
Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
FIND MINIMUM MIN, ↑MIN 	Searches the range of memory from St to St+N–1 for the address that contains the minimum value and outputs that value to the destination word (D). The number of words within the range (N) is contained in the 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 hexadecimal values, and when it is ON the data is treated as signed hexadecimal values. When bit 14 of C is OFF, the address of the word containing the minimum value will not be output to IR0; when it is ON, the value will be output to IR0.	**C:** CIO G A # DM DR IR **St:** CIO G A T/C DM **D:** CIO G A DM DR IR	320
SUM SUM, ↑SUM 	Computes the sum of the contents of words from St to St+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 values, and when it is ON the data is treated as signed values. 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 hexadecimal.	**C:** CIO G A # DM DR IR **St:** CIO G A T/C DM **D:** CIO G A DM	322
TRACE MEMORY TRSM 	Marks the start of tracing. This instruction is effective only when tracing is being executed from a Peripheral Device. Changes in status of the bits and words specified from the Peripheral Device are stored in trace memory.	None	393
SELECT EM BANK EMBC, ↑EMBC 	Changes the current EM bank to the one indicated by the EM bank number (N). N must be between #0000 and #0007. The current EM bank number is recorded in the least significant (rightmost) digit of A511. Bit A51115 is ON when an Expansion Data Memory Unit is mounted to the CPU. EM is optional and available with various numbers of banks.	**N:** CIO G A # DM DR IR	364
LOAD FLAGS CCL, ↑CCL 	Changes the Arithmetic Flags to the status recorded by the last CCS(173) instruction. Arithmetic Flags include the following: ER (A50003), CY (A50004), GR (A50005), EQ (A50006), LE (A50007), and N (A50008).	None	366
SAVE FLAGS CCS, ↑CCS 	Records the current status of the Arithmetic Flags in the CPU for later retrieval by the CCS(173) instruction.	None	367
MARK TRACE MARK 	Marks the location for sampling when executing a mark trace from a Peripheral Device or when measuring the execution time between MARK(174) instructions. When executing a mark trace, the status of the words specified from the Peripheral Devices are stored in trace memory.	**N:** #	395
LOAD REGISTER REGL, ↑REGL 	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. S to S+5 must be in the same data area.	**S:** CIO G A DM	367

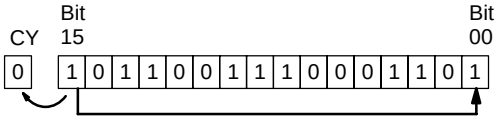
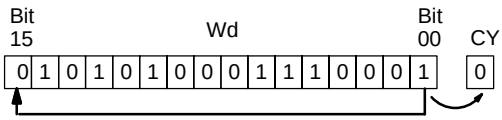
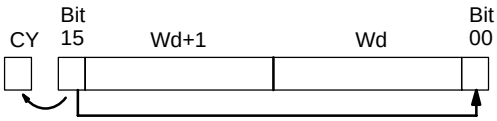
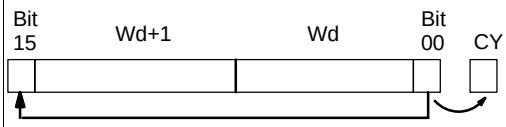
Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
SAVE REGISTER REGS, ↑REGS (176) REGS D	Copies the data from Data Registers DR0, DR1, and DR2 to D, D+1, and D+2, and copies the data from Index Registers IR0, IR1, and IR2 to D+3, D+4, and D+5. D to D+5 must be in the same data area.	**D:** CIO G A DM	368
FAILURE POINT DETECTION (V2 only) FPD (177) FPD C W D	Monitors the execution of an instruction block according to specified conditions, detects errors, and determines the input conditions responsible for the errors.	**C:** # Ti **W:** CIO G A DM **D:** CIO G A DM	356
MAXIMUM CYCLE TIME EXTEND WDT, ↑WDT (V2 only) (178) WDT T	Extends the setting of the watchdog timer by 10 ms times T.	# (0000 to 3999)	361
CLOCK COMPENSATION (V2 only) DATE, ↑DATE (179) DATE C	Changes the internal clock setting according to the clock data in four consecutive specified words beginning with C	**C:** CIO G A T/C DM	353
READ DATA FILE FILR, ↑FILR (180) FILR N D C	Reads N words of data from the Memory Card data file specified in C+1 to C+4 and outputs the data to the designated data area beginning at D. The name of the file from which the data is read is specified by eight ASCII characters in C+1 to C+4. Data will be read from the word indicated in C+5 if bit 04 of C (the Offset Enable Bit) is ON.	**N:** CIO G A T/C DM DR IR **D:** CIO G A T/C DM **C:** CIO G A T/C DM	396
WRITE DATA FILE FILW, ↑FILW (181) FILW N S C	Writes the data in S to S+N–1 to a Memory Card data file specified in C+1 to C+4. The data will replace data in the file if bit 07 of C (the Write-over Bit) is ON, or will be added to the end of the file if bit 07 of C is OFF. Data will be written from the word indicated in C+5 if bit 04 of C (the Offset Enable Bit) is ON. If the specified file name does not exist, a new file by that name will be created and the data will be written from the beginning of the file, regardless of the status of the Offset Enable Bit.	**N:** CIO G A T/C DM DR IR **S:** CIO G A T/C DM **C:** CIO G A T/C DM	398
READ PROGRAM FILE FILP, ↑FILP (182) FILP C	Reads the ladder program file (extension .LDP) named in C+1 to C+4 and either overwrites or replaces the program following the FILP(182) instruction with it. The program file must be written to the Memory Card beforehand with CVSS. The file name is given in eight ASCII characters starting in the leftmost half of C+1. The extension is not required. When the Write Method Bit (C bit 07) is ON, FILP(182) will overwrite just enough of the current ladder program to accommodate the program file. When C bit 07 is OFF, FILP(182) will erase the current ladder program from the instruction just after FILP(182) to END(001), then insert the program file. The program will be executed from the beginning when FILP(182) has been completed.	**C:** CIO G A T/C DM	400

Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
CHANGE STEP PROGRAM FLSP, ↑FLSP (183) FLSP N C	Reads from the Memory Card the action block in the step program file (extension .SFC) specified in C and replaces the action block for step N with it. The new action block must have the same number of actions as the one being replaced. The file name is given in eight ASCII characters starting in the leftmost half of C+1. The extension is not required. The step program file must be written to the Memory Card beforehand with CVSS.	**N:** CIO, G, A, T/C, DM, #, DR, IR **C:** CIO, G, A, T/C, DM	402
I/O REFRESH IORF, ↑IORF (184) IORF St E	Refreshes all I/O words between the start (St) and end (E) words. Only I/O words may be designated. Normally these words are refreshed only once per cycle, but refreshing words before use in an instruction can increase processing speed. St must be less than or equal to E. St and E must be between CIO 0000 and CIO 0511.	**St:** CIO, DR*, IR* **E:** CIO	362
DISABLE ACCESS IOSP, ↑IOSP (187) IOSP	Disables both read and write access to PC memory from Peripheral Devices, SYSMAC NET Link Units, SYSMAC LINK Units, Host Link Systems, BASIC Units, etc. Access to memory is disabled until either END(001) or IORS(188) is executed or PC operation is stopped.	None	413
ENABLE ACCESS IORS (188) IORS	Enables both read and write access to PC memory from Peripheral Devices, SYSMAC NET Link Units, SYSMAC LINK Units, Host Link Systems, BASIC Units, etc.	None	414
I/O DISPLAY IODP, ↑IODP (189) IODP C S	Outputs the message in S to the 7-segment display on a Remote I/O Slave Unit, I/O Control Unit, or I/O Interface Unit. The four display characters can be in 7-segment display code in source words S and S+1, or converted automatically from a single hexadecimal source word (S). The Unit and the display attributes are specified in C.  	**C:** CIO, G, A, T/C, DM, DR, IR **S:** CIO, G, A, T/C, DM, #	362

Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
I/O READ READ (190) READ N S D	Reads data from memory area of a Special I/O Unit through a word (S) allocated to the Special I/O Unit to destination words (D gives the address of the first destination word). N must be in BCD and is the number of words to be transferred. If the Special I/O Unit is busy and unable to receive data, the data will be written during the next cycle. The EQ Flag is set when read transfer is completed. 	**N:** CIO G A T/C DM # DR IR **S:** CIO G A T/C DM DR* IR* **D:** CIO G A T/C DM	404
I/O WRITE WRIT (191) WRIT N S D	Writes data through the I/O word (D) allocated to a Special I/O Unit to the memory area of the Special I/O Unit. N must be in BCD and is the number of words to be transferred. If the Special I/O Unit is busy and unable to receive data, the data will be written during the next cycle. S is the address of the first PC source word to be transferred. The EQ Flag is set when the transfer is completed. 	**N:** CIO G A T/C DM # DR IR **S:** CIO G A T/C DM **D:** CIO	408
NETWORK SEND SEND, ↑SEND (192) SEND S D C	Sends data from n source words (S is the starting word) to the destination words (D is the first word) in the specified node of the specified network in a SYSMAC LINK or SYSMAC NET Link System. The destination node can be a PC, a BASIC Unit, or a computer. The number of words to be transferred, the source (local) node and network addresses, the destination node and network addresses, and other parameters are given in C to C+4. 	**S/D/C:** CIO G A T/C DM	415

Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
NETWORK RECEIVE RECV, ↑RECV (193) RECV S D C	Receives data from n source words (S is the starting word) from the specified node of the specified network and writes it to the destination words (D is the first word) in a SYSMAC LINK or SYSMAC NET Link System. The source node can be a PC, a BASIC Unit, or a computer. The number of words to be transferred, the destination (local) node and network addresses, the source node and network addresses, and other parameters are given in C to C+4. 	**S/D/C:** CIO G A T/C DM	418
DELIVER COMMAND CMND, ↑CMND (194) CMND S D C	Sends the FINS command starting in S to the specified node of the specified network in a SYSMAC LINK or SYSMAC NET Link System and stores the response starting at D. The number of words to be transferred, the destination node and network addresses, and other parameters are given in C to C+5. The FINS commands used here are the same as those used in the Host Link System. If the destination node number is $FF, the command will be broadcast to all nodes in the designated network.	**S/D/C:** CIO G A T/C DM	420
MESSAGE MSG, ↑MSG (195) MSG N S	Displays 32 ASCII characters from 16 words starting at S on the Peripheral Device. If not all sixteen words are required for the message, it can be stopped at any point by inputting "OD." The message number (N) must be registered in the System Setup of the Peripheral Device. N must be in BCD and must be between 0000 and 0007. The message is cleared when the next message is displayed or when a constant is input for S. 	**N:** CIO G A T/C DM # DR IR **S:** CIO G A T/C DM #	414
TRANSITION OUTPUT TOUT (202) TOUT	Outputs the result of a transition program to the Transition Flag.	None	433
ACTIVATE STEP SA, ↑SA (210) SA N_1 N_2	Places step N_1 in subchart N_2 in execute status to start the execution of actions. N_1 and N_2 are input as the numeric portions of ST addresses. Specify 9999 for N_2 if the step is not in a subchart. To activate a subchart, specify the subchart dummy step for N_1.	**N_1/N_2:** #	427

Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
PAUSE STEP SP, ↑SP 	Changes a step or subchart from execute to pause status. To pause a subchart, specify the subchart dummy step for N_1. Actions with S-type action qualifiers will continue to be executed.	**N:** #	428
RESTART STEP SR, ↑SR 	Changes step N from pause to execute status. To restart a subchart, specify the subchart dummy step for N.	**N:** #	429
END STEP SF, ↑SF 	Changes step N from execute or pause to halt status. To halt a subchart, specify the subchart dummy step for N. Actions with S-type action qualifiers will continue to be executed.	**N:** #	430
DEACTIVATE STEP SE, ↑SE 	Changes step N from active (execute, pause, or halt) to inactive status. To deactivate a subchart, specify the subchart dummy step for N. Actions with S-type action qualifiers and those with action qualifier with the H option will not be reset.	**N:** #	431
RESET STEP SOFF, ↑SOFF 	Resets step N (regardless of its status) to inactive status. It also resets actions with action qualifiers S, SD, DS, and SL. To reset a subchart, specify the subchart dummy step for N.	**N:** #	432
CONDITIONAL JUMP (V2 only) CJP 	When the execution condition is ON, the program jumps directly to JME(005).	**N:** CIO G A T/C # DM DR IR	138
CONDITIONAL JUMP (V2 only) CJPN 	When the execution condition is OFF, the program jumps directly to JME(005).	**N:** CIO G A T/C # DM DR IR	138
RESET TIMER/COUNTER CNR, ↑CNR 	When D_1 and D_2 are timer or counter numbers, CNR(236) resets the PV of timers from D_1 through D_2 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_1 and D_2. When D_1 and D_2 are addresses in a data area, CNR(236) sets the content of words D_1 through D_2 to zero. D_1 must be less than or equal to D_2.	**D_1:** **D_2:** CIO CIO G G A A T/C T/C DM DM	158

Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
BLOCK PROGRAM (V2 only) BPRG (250) BPRG N	Indicates the beginning of the designated block program.		439
ROTATE LEFT WITHOUT CARRY (V2 only) RLNC, ↑RLNC (260) RLNC Wd	Shifts all Wd bits one bit to the left, shifting the status of bit 15 of Wd simultaneously into bit 00 and CY. 	**Wd:** CIO G A DM DR IR	180
ROTATE RIGHT WITHOUT CARRY (V2 only) RRNC, ↑RRNC (261) RRNC Wd	Shifts all Wd bits one bit to the right, shifting the status of bit 00 simultaneously into bit 15 of Wd and into CY. 	**Wd:** CIO G A DM DR IR	181
DOUBLE ROTATE LEFT WITHOUT CARRY (V2 only) RLNL, ↑RLNL (262) RLNL Wd	Shifts all bits previously in Wd and Wd+1 to the left, and shifts bit 15 of Wd+1 simultaneously into bit 00 or Wd and into CY. 	**Wd:** CIO G A DM	182
DOUBLE ROTATE RIGHT WITHOUT CARRY (V2 only) RRNL, ↑RRNL (263) RRNL Wd	Shifts all bits previously in Wd and Wd+1 to the right, and shifts bit 00 of Wd simultaneously into bit 15 of Wd+1 and into CY. 	**Wd:** CIO G A DM	183
PID CONTROL (V2 only) PID (270) PID S C D	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 operation according to the parameters that are set. The results are then stored as the operation output amount in output word D.	**S:** CIO G A DM DR IR **C:** CIO G A DM **D:** CIO G A DM DR IR	*330*
LIMIT CONTROL (V2 only) LMT, ↑LMT (271) LMT S C D	Controls output data according to whether or not the specified input data (signed 16-bit binary) is within the upper and lower limits.	**S:** CIO G A T/C DM DR IR **C:** CIO G A T/C DM **D:** CIO G A T/C DM DR IR	337
DEAD-BAND CONTROL (V2 only) BAND, ↑BAND (272) BAND S C D	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).	**S:** CIO G A T/C DM DR IR **C:** CIO G A T/C DM **D:** CIO G A T/C DM DR IR	338

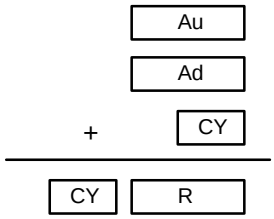
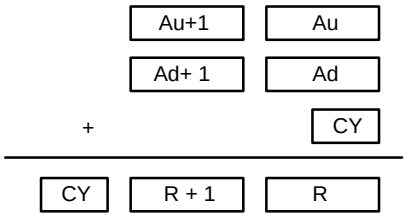
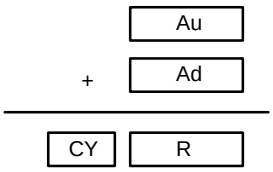
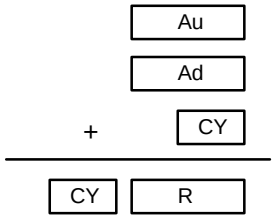
Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
DEAD-ZONE CONTROL (V2 only) ZONE, ↑ZONE (273) ZONE S C D	Adds the specified bias value to the specified input data (signed 16-bit binary) and places the result in a specified word.	**S:** CIO G A T/C DM DR IR **C:** CIO G A T/C DM **D:** CIO G A T/C DM DR IR	340
BINARY ROOT (V2 only) ROTB, ↑ROTB (274) ROTB S R	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).	**S:** CIO G A T/C DM # **R:** CIO G A DM DR IR	325
SIGNED BCD-TO-BINARY (V2 only) BINS, ↑BINS (275) BINS C S D	Converts the data in specified words from signed BCD to signed binary, and outputs the result to a specified destination word.	**C:** CIO G A T/C # DM DR IR **S:** CIO G A T/C DM DR IR **D:** CIO G A DM DR IR	242
SIGNED BINARY-TO-BCD (V2 only) BCDS, ↑BCDS (276) BCDS C S D	Converts the data in specified words from signed binary to signed BCD, and outputs the result to a specified destination word.	**C:** CIO G A T/C # DM DR IR **S:** CIO G A T/C DM DR IR **D:** CIO G A DM DR IR	244
DOUBLE SIGNED BCD-TO-BINARY (V2 only) BISL, ↑BISL (277) BINS C S D	Converts the data in specified words from double signed BCD to double signed binary, and outputs the result to specified destination words.	**C:** CIO G A T/C # DM **S:** CIO G A T/C DM **D:** CIO G A DM	246
DOUBLE SIGNED BINARY-TO-BCD (V2 only) BDSL, ↑BDSL (278) BDSL C S D	Converts the data in specified words from double signed binary to double signed BCD, and outputs the result to specified destination words.	**C:** CIO G A T/C # DM **S:** CIO G A T/C DM **D:** CIO G A DM	248
I/O READ 2 (V2 only) RD2, ↑RD2 (280) RD2 C S D	Reads the memory contents of a Special I/O Unit to specified words (beginning with D) in the Programmable Controller, via a specified Special I/O Unit interface word (S) in the PC's memory. The words that are to be transferred are specified by the control word (C).	**C:** CIO G A T/C # DM DR IR **S:** CIO **D:** CIO G A T/C DM	406

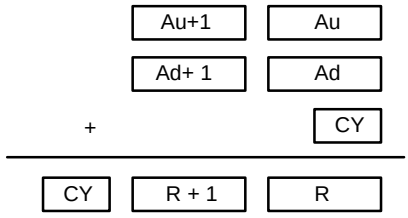
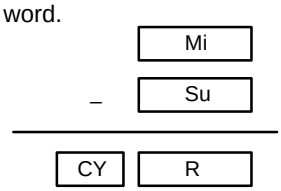
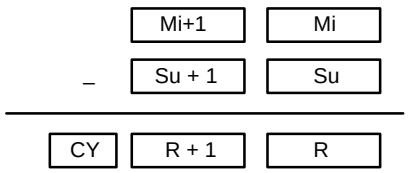
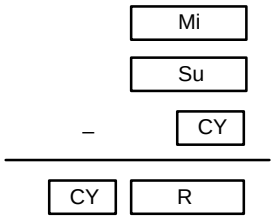
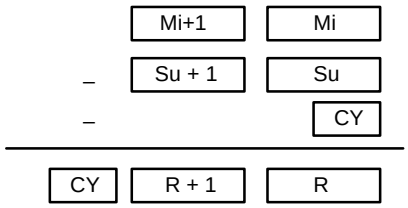
Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
I/O WRITE 2 (V2 only) WR2, ↑WR2 (281) WR2 C S D	Writes the specified number of words to a specified address in a Special I/O Unit, via a specified Special I/O Unit interface word (S) in the PC's memory. The words that are to be transferred are specified by the control word (C).	**C:** CIO G A T/C # DM DR IR **S:** CIO G A T/C DM **D:** CIO	411
EQUAL (V2 only) = (300) = S_1 S_2	Compares word data and constants in four digits hexadecimal, and turns ON the execution condition if the result is true (i.e., if $S_1 = S_2$).	**S_1:** CIO G A T/C # DM DR IR **S_2:** CIO G A T/C # DM DR IR	213
DOUBLE EQUAL (V2 only) =L (301) =L S_1 S_2	Compares word data and constants in eight digits hexadecimal, and turns ON the execution condition if the result is true (i.e., if $S_1 = S_2$).	**S_1:** CIO G A T/C # DM **S_2:** CIO G A T/C # DM	213
SIGNED EQUAL (V2 only) =S (302) =S S_1 S_2	Compares word data and constants in four digits signed hexadecimal, and turns ON the execution condition if the result is true (i.e., if $S_1 = S_2$).	**S_1:** CIO G A T/C # DM DR IR **S_2:** CIO G A T/C # DM DR IR	213
DOUBLE SIGNED EQUAL (V2 only) =SL (303) =SL S_1 S_2	Compares word data and constants in eight digits of signed hexadecimal, and turns ON the execution condition if the result is true (i.e., if $S_1 = S_2$).	**S_1:** CIO G A T/C # DM **S_2:** CIO G A T/C # DM	213
DOUBLE NOT EQUAL (V2 only) < >L (306) < >L S_1 S_2	Compares word data and constants in eight digits hexadecimal, and turns ON the execution condition if the result is true (i.e., if $S_1 <> S_2$).	**S_1:** CIO G A T/C # DM **S_2:** CIO G A T/C # DM	213
SIGNED NOT EQUAL (V2 only) < >S (307) < >S S_1 S_2	Compares word data and constants in four digits signed hexadecimal, and turns ON the execution condition if the result is true (i.e., if $S_1 <> S_2$).	**S_1:** CIO G A T/C # DM DR IR **S_2:** CIO G A T/C # DM DR IR	213

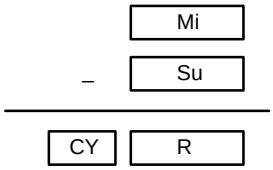
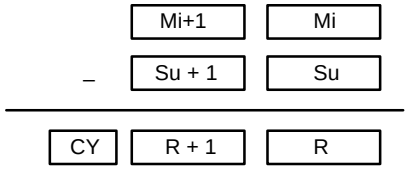
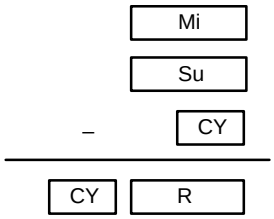
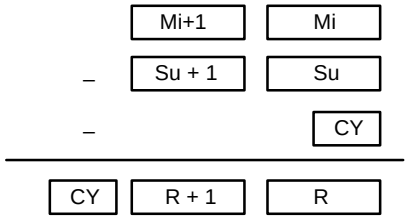
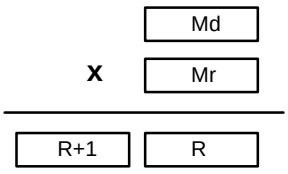
Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
DOUBLE SIGNED NOT EQUAL (V2 only) < >SL (308) < >SL S_1 S_2	Compares word data and constants in eight digits signed hexadecimal, and turns ON the execution condition if the result is true (i.e., if S_1 <> S_2).	**S_1:** CIO, G, A, T/C, #, DM **S_2:** CIO, G, A, T/C, #, DM	213
LESS THAN (V2 only) (310) < S_1 S_2	Compares word data and constants in four digits hexadecimal, and turns ON the execution condition if the result is true (i.e., if S_1 < S_2).	**S_1:** CIO, G, A, T/C, #, DM, DR, IR **S_2:** CIO, G, A, T/C, #, DM, DR, IR	213
DOUBLE LESS THAN (V2 only) <L (311) <L S_1 S_2	Compares word data and constants in eight digits hexadecimal, and turns ON the execution condition if the result is true (i.e., if S_1 < S_2).	**S_1:** CIO, G, A, T/C, #, DM **S_2:** CIO, G, A, T/C, #, DM	213
SIGNED LESS THAN (V2 only) <S (312) <S S_1 S_2	Compares word data and constants in four digits signed hexadecimal, and turns ON the execution condition if the result is true (i.e., if S_1 < S_2).	**S_1:** CIO, G, A, T/C, #, DM, DR, IR **S_2:** CIO, G, A, T/C, #, DM, DR, IR	213
DOUBLE SIGNED LESS THAN (V2 only) <SL (313) <SL S_1 S_2	Compares word data and constants in eight digits signed hexadecimal, and turns ON the execution condition if the result is true (i.e., if S_1 < S_2).	**S_1:** CIO, G, A, T/C, #, DM **S_2:** CIO, G, A, T/C, #, DM	213
LESS THAN OR EQUAL (V2 only) < = (315) <= S_1 S_2	Compares word data and constants in four digits hexadecimal, and turns ON the execution condition if the result is true (i.e., if S_1 □ S_2).	**S_1:** CIO, G, A, T/C, #, DM, DR, IR **S_2:** CIO, G, A, T/C, #, DM, DR, IR	213
DOUBLE LESS THAN OR EQUAL (V2 only) < =L (316) <=L S_1 S_2	Compares word data and constants in eight digits hexadecimal, and turns ON the execution condition if the result is true (i.e., if S_1 □ S_2).	**S_1:** CIO, G, A, T/C, #, DM **S_2:** CIO, G, A, T/C, #, DM	213

Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
SIGNED LESS THAN OR EQUAL (V2 only) < =S (317) ——[<=S S_1 S_2]	Compares word data and constants in four digits signed hexadecimal, and turns ON the execution condition if the result is true (i.e., if S_1 ◻ S_2).	**S_1:** CIO G A T/C # DM DR IR **S_2:** CIO G A T/C # DM DR IR	213
DOUBLE SIGNED LESS THAN OR EQUAL (V2 only) < =SL (318) ——[<=SL S_1 S_2]	Compares word data and constants in eight digits signed hexadecimal, and turns ON the execution condition if the result is true (i.e., if S_1 ◻ S_2).	**S_1:** CIO G A T/C # DM **S_2:** CIO G A T/C # DM	213
GREATER THAN (V2 only) > (320) ——[> S_1 S_2]	Compares word data and constants in four digits hexadecimal, and turns ON the execution condition if the result is true (i.e., if $S_1 > S_2$).	**S_1:** CIO G A T/C # DM DR IR **S_2:** CIO G A T/C # DM DR IR	213
DOUBLE GREATER THAN (V2 only) >L (321) ——[>L S_1 S_2]	Compares word data and constants in eight digits hexadecimal, and turns ON the execution condition if the result is true (i.e., if $S_1 > S_2$).	**S_1:** CIO G A T/C # DM **S_2:** CIO G A T/C # DM	213
SIGNED GREATER THAN (V2 only) >S (322) ——[>S S_1 S_2]	Compares word data and constants in four digits signed hexadecimal, and turns ON the execution condition if the result is true (i.e., if $S_1 > S_2$).	**S_1:** CIO G A T/C # DM DR IR **S_2:** CIO G A T/C # DM DR IR	213
DOUBLE SIGNED GREATER THAN (V2 only) >SL (323) ——[>SL S_1 S_2]	Compares word data and constants in eight digits signed hexadecimal, and turns ON the execution condition if the result is true (i.e., if $S_1 > S_2$).	**S_1:** CIO G A T/C # DM **S_2:** CIO G A T/C # DM	213
GREATER THAN OR EQUAL (V2 only) > = (325) ——[>= S_1 S_2]	Compares word data and constants in four digits hexadecimal, and turns ON the execution condition if the result is true (i.e., if S_1 ◻ S_2).	**S_1:** CIO G A T/C # DM DR IR **S_2:** CIO G A T/C # DM DR IR	213

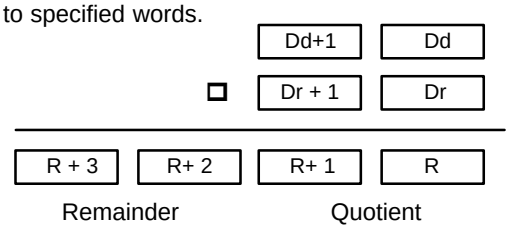
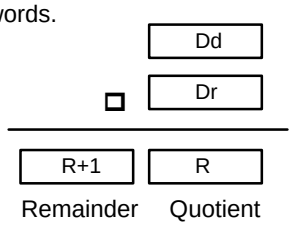
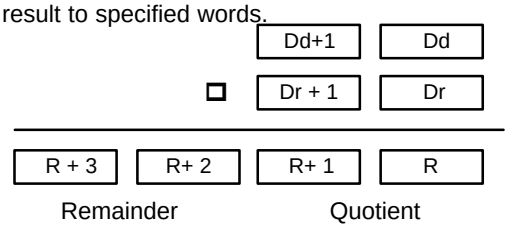
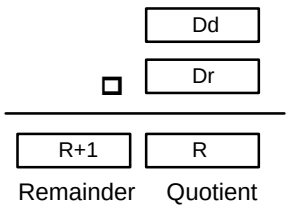
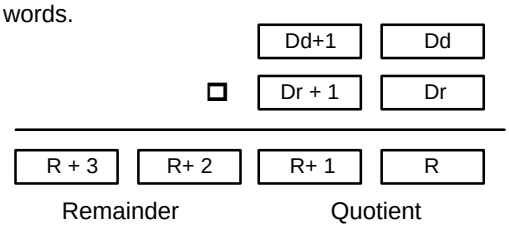
Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
DOUBLE GREATER THAN OR EQUAL (V2 only) > =L (326) >=L S_1 S_2	Compares word data and constants in eight digits hexadecimal, and turns ON the execution condition if the result is true (i.e., if S_1 ◻ S_2).	**S_1:** CIO G A T/C # DM **S_2:** CIO G A T/C # DM	213
SIGNED GREATER THAN OR EQUAL (V2 only) > =S (327) >=S S_1 S_2	Compares word data and constants in four digits signed hexadecimal, and turns ON the execution condition if the result is true (i.e., if S_1 ◻ S_2).	**S_1:** CIO G A T/C # DM DR IR **S_2:** CIO G A T/C # DM DR IR	213
DOUBLE SIGNED GREATER THAN OR EQUAL (V2 only) > =SL (328) >=SL S_1 S_2	Compares word data and constants in eight digits signed hexadecimal, and turns ON the execution condition if the result is true (i.e., if S_1 ◻ S_2).	**S_1:** CIO G A T/C # DM **S_2:** CIO G A T/C # DM	213
BIT TEST (V2 only) TST (350) TST S N	Turns ON the execution condition when a designated bit in a designated word turns ON.	**S:** CIO G A DM DR IR **N:** CIO G A T/C # DM DR IR	124
BIT TEST (V2 only) TSTN (351) TSTN S N	Turns ON the execution condition when a designated bit in a designated word turns OFF.	**S:** CIO G A DM DR IR **N:** CIO G A T/C # DM DR IR	124
SIGNED BINARY ADD WITHOUT CARRY (V2 only) +, ↑+ (400) + Au Ad R Au + Ad CY R	Adds word data and constants in four digits hexadecimal with sign, and outputs the result to a specified word.	**Au:** CIO G A T/C # DM DR IR **Ad:** CIO G A T/C # DM DR IR **R:** CIO G A DM DR IR	272
DOUBLE SIGNED BINARY ADD WITHOUT CARRY (V2 only) +L, ↑+L (401) +L Au Ad R Au+1 Au + Ad + 1 Ad CY R + 1 R	Adds word data and constants in eight digits hexadecimal with sign, and outputs the result to specified words.	**Au:** CIO G A T/C # DM **Ad:** CIO G A T/C # DM **R:** CIO G A DM	272

Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
SIGNED BINARY ADD WITH CARRY (V2 only) +C, ↑+C (402) +C Au Ad R	Adds word data and constants, including carry, in four digits hexadecimal with sign, and outputs the result to a specified word. 	**Au:** CIO G A T/C # DM DR IR **Ad:** CIO G A T/C # DM DR IR **R:** CIO G A DM DR IR	272
DOUBLE SIGNED BINARY ADD WITH CARRY (V2 only) +CL, ↑+CL (403) +CL Au Ad R	Adds word data and constants, including carry, in eight digits hexadecimal with sign, and outputs the result to specified words. 	**Au:** CIO G A T/C # DM **Ad:** CIO G A T/C # DM **R:** CIO G A DM	272
BCD ADD WITHOUT CARRY (V2 only) +B, ↑+B (404) +B Au Ad R	Adds word data and constants in four digits BCD, and outputs the result to a specified word. 	**Au:** CIO G A T/C # DM DR IR **Ad:** CIO G A T/C # DM DR IR **R:** CIO G A DM DR IR	274
DOUBLE BCD ADD WITHOUT CARRY (V2 only) +BL, ↑+BL (405) +BL Au Ad R	Adds word data and constants in eight digits BCD, and outputs the result to specified words. 	**Au:** CIO G A T/C # DM **Ad:** CIO G A T/C # DM **R:** CIO G A DM	274
BCD ADD WITH CARRY (V2 only) +BC, ↑+BC (406) +BC Au Ad R	Adds word data and constants, including carry, in four digits BCD, and outputs the result to a specified word. 	**Au:** CIO G A T/C # DM DR IR **Ad:** CIO G A T/C # DM DR IR **R:** CIO G A DM DR IR	274

Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
DOUBLE BCD ADD WITH CARRY (V2 only) +BCL, ↑+BCL (407) +BCL Au Ad R	Adds word data and constants, including carry, in eight digits BCD, and outputs the result to specified words. 	**Au:** CIO G A T/C # DM **Ad:** CIO G A T/C # DM **R:** CIO G A DM	274
SIGNED BINARY SUBTRACT WITHOUT CARRY (V2 only) –, ↑– (410) – Mi Su R	Subtracts word data and constants in four digits hexadecimal with sign, and outputs the result to a specified word. 	**Mi:** CIO G A T/C # DM DR IR **Su:** CIO G A T/C # DM DR IR **R:** CIO G A DM DR IR	276
DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY (V2 only) –L, ↑–L (411) –L Mi Su R	Subtracts word data and constants in eight digits hexadecimal with sign, and outputs the result to specified words. 	**Mi:** CIO G A T/C # DM **Su:** CIO G A T/C # DM **R:** CIO G A DM	276
SIGNED BINARY SUBTRACT WITH CARRY (V2 only) –C, ↑–C (412) –C Mi Su R	Subtracts word data and constants in four digits hexadecimal with sign, including carry, and outputs the result to a specified word. 	**Mi:** CIO G A T/C # DM DR IR **Su:** CIO G A T/C # DM DR IR **R:** CIO G A DM DR IR	276
DOUBLE SIGNED BINARY SUBTRACT WITH CARRY (V2 only) –CL, ↑–CL (413) –CL Mi Su R	Subtracts word data and constants in eight digits hexadecimal with sign, including carry, and outputs the result to specified words. 	**Mi:** CIO G A T/C # DM **Su:** CIO G A T/C # DM **R:** CIO G A DM	276

Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
BCD SUBTRACT WITHOUT CARRY (V2 only) –B, ↑–B (414) –B Mi Su R	Subtracts word data and constants in four digits BCD, and outputs the result to a specified word. 	**Mi:** CIO G A T/C # DM DR IR **Su:** CIO G A T/C # DM DR IR **R:** CIO G A DM DR IR	281
DOUBLE BCD SUBTRACT WITHOUT CARRY (V2 only) –BL, ↑–BL (415) –BL Mi Su R	Subtracts word data and constants in eight digits BCD, and outputs the result to specified words. 	**Mi:** CIO G A T/C # DM **Su:** CIO G A T/C # DM **R:** CIO G A DM	281
BCD SUBTRACT WITH CARRY (V2 only) –BC, ↑–BC (416) –BC Mi Su R	Subtracts word data and constants in four digits BCD, including carry, and outputs the result to a specified word. 	**Mi:** CIO G A T/C # DM DR IR **Su:** CIO G A T/C # DM DR IR **R:** CIO G A DM DR IR	281
DOUBLE BCD SUBTRACT WITH CARRY (V2 only) –BCL, ↑–BCL (417) –BCL Mi Su R	Subtracts word data and constants in eight digits BCD, including carry, and outputs the result to specified words. 	**Mi:** CIO G A T/C # DM **Su:** CIO G A T/C # DM **R:** CIO G A DM	281
SIGNED BINARY MULTIPLY (V2 only) *, ↑* (420) * Md Mr R	Multiplies word data and constants in four digits hexadecimal with sign, and outputs the result to specified words. 	**Md:** CIO G A T/C # DM DR IR **Mr:** CIO G A T/C # DM DR IR **R:** CIO G A DM	285

Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
DOUBLE SIGNED BINARY MULTIPLY (V2 only) *L, ↑*L (421) *L Md Mr R	Multiplies word data and constants in eight digits hexadecimal with sign, and outputs the result to specified words. Md+1 Md X Mr + 1 Mr R + 3 R+ 2 R+ 1 R	**Md:** CIO G A T/C # DM **Mr:** CIO G A T/C # DM **R:** CIO G A DM	285
UNSIGNED BINARY MULTIPLY (V2 only) *U, ↑*U (422) *U Md Mr R	Multiplies word data and constants in four digits hexadecimal without sign, and outputs the result to specified words. Md X Mr R+1 R	**Md:** CIO G A T/C # DM DR IR **Mr:** CIO G A T/C # DM DR IR **R:** CIO G A DM	285
DOUBLE UNSIGNED BINARY MULTIPLY (V2 only) *UL, ↑*UL (423) *UL Md Mr R	Multiplies word data and constants in eight digits hexadecimal without sign, and outputs the result to specified words. Md+1 Md X Mr + 1 Mr R + 3 R+ 2 R+ 1 R	**Md:** CIO G A T/C # DM **Mr:** CIO G A T/C # DM **R:** CIO G A DM	285
BCD MULTIPLY (V2 only) *B, ↑*B (424) *B Md Mr R	Multiplies word data and constants in four digits BCD, and outputs the result to specified words. Md X Mr R+1 R	**Md:** CIO G A T/C # DM DR IR **Mr:** CIO G A T/C # DM DR IR **R:** CIO G A DM	287
DOUBLE BCD MULTIPLY (V2 only) *BL, ↑*BL (425) *BL Md Mr R	Multiplies word data and constants in eight digits BCD, and outputs the result to specified words. Md+1 Md X Mr + 1 Mr R + 3 R+ 2 R+ 1 R	**Md:** CIO G A T/C # DM **Mr:** CIO G A T/C # DM **R:** CIO G A DM	287
SIGNED BINARY DIVIDE (V2 only) /, ↑/ (430) / Dd Dr R	Divides word data and constants in four digits hexadecimal with sign, and outputs the result to specified words. Dd □ Dr R+1 R Remainder Quotient	**Dd:** CIO G A T/C # DM DR IR **Dr:** CIO G A T/C # DM DR IR **R:** CIO G A DM	289

Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
DOUBLE SIGNED BINARY DIVIDE (V2 only) /L, ↑/L (431) /L Dd Dr R	Divides word data and constants in eight digits hexadecimal with sign, and outputs the result to specified words. 	**Dd:** CIO G A T/C # DM **Dr:** CIO G A T/C # DM **R:** CIO G A DM	289
UNSIGNED BINARY DIVIDE (V2 only) /U, ↑/U (432) /U Dd Dr R	Divides word data and constants in four digits hexadecimal without sign, and outputs the result to specified words. 	**Dd:** CIO G A T/C # DM DR IR **Dr:** CIO G A T/C # DM DR IR **R:** CIO G A DM	289
DOUBLE UNSIGNED BINARY DIVIDE (V2 only) /UL, ↑/UL (433) /UL Dd Dr R	Divides word data and constants in eight digits hexadecimal without sign, and outputs the result to specified words. 	**Dd:** CIO G A T/C # DM **Dr:** CIO G A T/C # DM **R:** CIO G A DM	289
BCD DIVIDE (V2 only) /B, ↑/B (434) /B Dd Dr R	Divides word data and constants in four digits BCD, and outputs the result to specified words. 	**Dd:** CIO G A T/C # DM DR IR **Dr:** CIO G A T/C # DM DR IR **R:** CIO G A DM	291
DOUBLE BCD DIVIDE (V2 only) /BL, ↑/BL (435) /BL Dd Dr R	Divides word data and constants in eight digits BCD, and outputs the result to specified words. 	**Dd:** CIO G A T/C # DM **Dr:** CIO G A T/C # DM **R:** CIO G A DM	291
FLOATING TO 16-BIT (V2 only) FIX, ↑FIX (450) FIX S R	Converts specified 32-bit floating-point data to 16-bit binary data, and places the result in a specified word.	**S:** CIO G A T/C # DM **R:** CIO G A DM DR IR	296

Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
FLOATING TO 32-BIT (V2 only) FIXL, ↑FIXL (451) FIXL S R	Converts specified 32-bit floating-point data to 32-bit binary data, and places the result in a specified word.	**S:** CIO G A T/C # DM **R:** CIO G A DM	297
16-BIT TO FLOATING (V2 only) FLT, ↑FLT (452) FLT S R	Converts specified 16-bit binary data to 32-bit floating-point data, and places the result in specified words.	**S:** CIO G A T/C # DM DR IR **R:** CIO G A DM	298
32-BIT TO FLOATING (V2 only) FLTL, ↑FLTL (453) FLTL S R	Converts specified 32-bit binary data to 32-bit floating-point data, and places the result in specified words.	**S:** CIO G A T/C # DM **R:** CIO G A DM	298
FLOATING-POINT ADD (V2 only) +F, ↑+F (454) +F Au Ad R	Adds specified 32-bit floating-point data and places the result in specified words. Au+1 Au + Ad + 1 Ad R + 1 R	**Au:** CIO G A T/C # DM **Ad:** CIO G A T/C # DM **R:** CIO G A DM	299
FLOATING-POINT SUBTRACT (V2 only) –F, ↑–F (455) –F Mi Su R	Subtracts specified 32-bit floating-point data and places the result in specified words. Mi+1 Mi – Su + 1 Su R + 1 R	**Mi:** CIO G A T/C # DM **Su:** CIO G A T/C # DM **R:** CIO G A DM	300
FLOATING-POINT MULTIPLY (V2 only) *F, ↑*F (456) *F Md Mr R	Multiplies specified 32-bit floating-point data and places the result in specified words. Md+1 Md X Mr + 1 Mr R+ 1 R	**Md:** CIO G A T/C # DM **Mr:** CIO G A T/C # DM **R:** CIO G A DM	301
FLOATING-POINT DIVIDE (V2 only) /F, ↑/F (457) /F Dd Dr R	Divides specified 32-bit floating-point data and places the result in specified words. Dd+1 Dd □ Dr + 1 Dr R+ 1 R	**Dd:** CIO G A T/C # DM **Dr:** CIO G A T/C # DM **R:** CIO G A DM	302

Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
DEGREES TO RADIANS (V2 only) RAD, ↑RAD (458) RAD S R	Converts specified 32-bit floating-point data from degrees to radians, and places the result in specified words.	**S:** CIO G A T/C # DM **R:** CIO G A DM	303
RADIANS TO DEGREES (V2 only) DEG, ↑DEG (459) DEG S R	Converts specified 32-bit floating-point data from radians to degrees, and places the result in specified words.	**S:** CIO G A T/C # DM **R:** CIO G A DM	304
SINE (V2 only) SIN, ↑SIN (460) SIN S R	Computes the sine value for angle data (in radian units) expressed as specified 32-bit floating-point data, and places the result in specified words.	**S:** CIO G A T/C # DM **R:** CIO G A DM	305
COSINE (V2 only) COS, ↑COS (461) COS S R	Computes the cosine value for angle data (in radian units) expressed as specified 32-bit floating-point data, and places the result in specified words.	**S:** CIO G A T/C # DM **R:** CIO G A DM	306
TANGENT (V2 only) TAN, ↑TAN (462) TAN S R	Computes the tangent value for angle data (in radian units) expressed as specified 32-bit floating-point data, and places the result in specified words.	**S:** CIO G A T/C # DM **R:** CIO G A DM	307
SINE TO ANGLE (V2 only) ASIN, ↑ASIN (463) ASIN S R	Computes angle data (in radian units) from a sine value expressed as specified 32-bit floating-point data, and places the result in specified words.	**S:** CIO G A T/C # DM **R:** CIO G A DM	308
COSINE TO ANGLE (V2 only) ACOS, ↑ACOS (464) ACOS S R	Computes angle data (in radian units) from a cosine value expressed as specified 32-bit floating-point data, and places the result in specified words.	**S:** CIO G A T/C # DM **R:** CIO G A DM	309
TANGENT TO ANGLE (V2 only) ATAN, ↑ATAN (465) ATAN S R	Computes angle data (in radian units) from a tangent value expressed as specified 32-bit floating-point data, and places the result in specified words.	**S:** CIO G A T/C # DM **R:** CIO G A DM	310
SQUARE ROOT (V2 only) SQRT, ↑SQRT (466) SQRT S R	Computes the square root of specified 32-bit floating-point data, and places the result in specified words.	**S:** CIO G A T/C # DM **R:** CIO G A DM	311

Name, mnemonic, variations, and symbol	Function	Operand data areas	Page
EXPONENT (V2 only) EXP, ↑EXP (467) EXP S R	Computes the exponent for specified 32-bit floating-point data, and places the result in specified words.	**S:** CIO, G, A, T/C, #, DM **R:** CIO, G, A, DM	312
LOGARITHM (V2 only) LOG, ↑LOG (468) LOG S R	Computes the natural logarithm for specified 32-bit floating-point data, and places the result in specified words.	**S:** CIO, G, A, T/C, #, DM **R:** CIO, G, A, DM	313

Block Programming Instructions

The following instructions are supported by version-2 CVM1 CPUs only.

Name Mnemonic	Function	Operand Data Areas	Page
BLOCK PROGRAM END BEND<001>	Indicates the end of a block program.	None	439
CONDITIONAL BRANCH IF<002> IF<002> B IF<002> NOT B	Indicates the part of the program that is to be executed when a given condition is satisfied.	**B**: IR SR HR AR LR TC	440
NO BRANCH ELSE<003>	Specifies the part of the program that is to be executed when the IF condition is not satisfied.	None	440
BRANCH END IEND<004>	Defines the end of the program portion that has started with IF<002>.	None	440
ONE CYCLE AND WAIT WAIT<005> WAIT<005> B WAIT<005> NOT B	Halts execution of a block program until a specified condition is satisfied.	**B:** IR SR HR AR LR TC	443
CONDITIONAL BLOCK EXIT EXIT<006> EXIT<006 B EXIT<006> NOT B	Exits a block program if a given condition is satisfied.	**B**: IR SR HR AR LR TC	444
LOOP LOOP<009>	Defines the beginning of section to be repeated until a specified terminal condition is satisfied.	None	445
LOOP END LEND<010> LEND<010> B LEND<010> NOT B	Defines the end of the section to be repeated. Execution of the specified section continues until the terminal condition is satisfied.	**B**: IR SR HR AR LR TC	445
BLOCK PROGRAM PAUSE BPPS<011> N	Causes the execution of designated block program to pause until a specified condition is satisfied (often used in conjunction with a timer or counter).	**N**: 0 to 99	446
BLOCK PROGRAM RESTART BPRS<012> N	Restarts execution of the designated block program.	**N**: 0 to 99	446
TIMER WAIT TIMW<013> N SV	The execution of the block program between the TIMW<013> instruction and BEND<004> is not executed until the set value of the specified timer has been reached. SV: 000.0 to 999.9 s	**SV**: **N**: IR TC AR DM HR LR #	447
COUNTER WAIT CNTW<014> N SV I	The portion of block program between the CNTW<014> instruction and BEND<004> is not executed until the set value of the specified counter has been reached.	**SV**: **N**: IR TC AR DM HR LR #	448

Name Mnemonic	Function	Operand Data Areas	Page
HIGH-SPEED TIMER WAIT TMHW<015> N SV	The portion of program between the TIMH<015> instruction and BEND<004> is not executed until the set value of the high-speed timer has been reached. SV: 00.00 to 99.99 s	**SV**: IR AR DM HR LR # **N**: TC	447

Appendix B
Error and Arithmetic Flag Operation

The following table shows the instructions that affect the ER, CY, GR, LE, EQ, OF, UF, and N flags. In general, ER indicates that operand data is not within requirements. CY indicates arithmetic or data shift results. GR indicates that a compared value is larger than some standard, LE that it is smaller, EQ that it is the same. EQ also indicates a result of zero for arithmetic operations. N generally indicates that bit 15 of the result word is ON. OF indicates that the results was greater than could be stored in memory; UF indicates that the results was less than could be stored in memory. Refer to subsections of *Section 5 Instruction Set* for details.

A set value (SV) BCD check is carried out at the time of resetting (TIM, CNT, TIMH(015), TIML(121), TCNT(123), TIMW<013>, CNTW<014>, and TMHW<015>), or counting (CNTR(012), TTIM(120), and MTIM(122)).

When CCL(172) is executed, the status of these flags will return to the status when CCS(173) was last executed. CCL(172), STC(078), and CLC(079) are the only instructions that can directly change the status of the arithmetic flags.

“ON/OFF” in the table indicate the flags that are turned ON and OFF according to the result of the instruction.

Although ladder diagram instructions, TIM, CNT, TIMH(015), CNTR(012), TIML(121), TIMW<013>, CNTW<014>, and TMHW <015> are executed when ER is ON, other instructions with “ON/OFF” under the Error Flag (A50003) column are not executed if ER is ON. All of the other flags in the following table will also not operate when ER is ON.

Instructions not shown do not affect any of the flags in the table. Although only the non-differentiated form of each instruction is shown, all variations of an instructions affect flags in exactly the same way.

Instructions marked with an asterisk (*) are supported by version-2 CVM1 CPUs only.

The status of the ER, CY, GT, LT and EQ Flags is affected by instruction execution and will change each time an instruction that affects them is executed. Differentiated instructions are executed only once when their execution condition changes (ON to OFF or OFF to ON) and are not executed again until the next specified change in their execution condition. The status of the ER, CY, GT, LT and EQ Flags is thus affected by a differentiated instruction only when the execution condition changes and is not affected during scans when the instruction is not executed, i.e., when the specified change does not occur in the execution condition. When a differentiated instruction is not executed, the status of the ER, CY, GT, LT and EQ Flags will not change and will maintain the status produced by the last instruction that was executed.

Instructions	A50003 (ER)	A50004 (CY)	A50005 (GR)	A50006 (EQ)	A50007 (LE)	A50008 (N)	A50009 (OF)	A50010 (UF)
TIM	ON/OFF							
CNT								
JMP(004)	ON/OFF							
FAL(006)								
FALS(007)								
CNTR(012)								
TIMH(015)								
CMP(020)	ON/OFF		ON/OFF	ON/OFF	ON/OFF			
CMPL(021)								
BCMP(022)	ON/OFF			ON/OFF				
TCMP(023)								
MCMP(024)								
EQU(025)	ON/OFF							
CPS(026)*	ON/OFF		ON/OFF	ON/OFF	ON/OFF			
CPSL(027)*								
CMP(028)*								
CMPL(029)*								
Note “ON/OFF” means that the flag is affected by the result of instruction execution.								

Instructions	A50003 (ER)	A50004 (CY)	A50005 (GR)	A50006 (EQ)	A50007 (LE)	A50008 (N)	A50009 (OF)	A50010 (UF)
MOV(030)	ON/OFF			ON/OFF		ON/OFF		
MVN(031)								
MOVL(032)								
MVNL(033)								
XCHG(034)	ON/OFF							
XCGL(035)								
MOVR(036)	ON/OFF			ON/OFF				
XFRB(038)*	ON/OFF							
XFER(040)	ON/OFF							
BSET(041)								
MOVB(042)								
MOVD(043)								
DIST(044)	ON/OFF			ON/OFF		ON/OFF		
COLL(045)								
BXFR(046)*	ON/OFF							
SETA(047)*								
RSTA(048)*								
SFTR(051)	ON/OFF	ON/OFF						
ASFT(052)	ON/OFF							
WSFT(053)								
NSFL(054)*	ON/OFF	ON/OFF						
NSFR(055)*								
NASL(056)*	ON/OFF	ON/OFF		ON/OFF		ON/OFF		
NASR(057)*								
NSLL(058)*								
NSRL(059)*								
ASL(060)	ON/OFF	ON/OFF		ON/OFF		ON/OFF		
ASR(061)	ON/OFF	ON/OFF		ON/OFF		OFF		
ROL(062)	ON/OFF	ON/OFF		ON/OFF		ON/OFF		
ROR(063)								
ASLL(064)								
ASRL(065)	ON/OFF	ON/OFF		ON/OFF		OFF		
ROLL(066)	ON/OFF	ON/OFF		ON/OFF		ON/OFF		
RORL(067)								
SLD(068)	ON/OFF							
SRD(069)								
ADD(070)	ON/OFF	ON/OFF		ON/OFF				
SUB(071)								
MUL(072)	ON/OFF			ON/OFF				
DIV(073)								
ADDL(074)	ON/OFF	ON/OFF		ON/OFF				
SUBL(075)								
MULL(076)	ON/OFF			ON/OFF				
DIVL(077)								
STC(078)		ON						
CLC(079)		OFF						

Note "ON/OFF" means that the flag is affected by the result of instruction execution.

Instructions	A50003 (ER)	A50004 (CY)	A50005 (GR)	A50006 (EQ)	A50007 (LE)	A50008 (N)	A50009 (OF)	A50010 (UF)
ADB(080)	ON/OFF	ON/OFF		ON/OFF		ON/OFF	ON/OFF	ON/OFF
SBB(081)								
MLB(082)	ON/OFF			ON/OFF		ON/OFF		
DVB(083)								
ADBL(084)	ON/OFF	ON/OFF		ON/OFF		ON/OFF	ON/OFF	ON/OFF
SBBL(085)								
MLBL(086)	ON/OFF			ON/OFF		ON/OFF		
DVBL(087)								
INC(090)	ON/OFF			ON/OFF				
DEC(091)								
INCB(092)	ON/OFF			ON/OFF		ON/OFF		
DECB(093)								
INCL(094)	ON/OFF			ON/OFF				
DECL(095)								
INBL(096)	ON/OFF			ON/OFF		ON/OFF		
DCBL(097)								
BIN(100)	ON/OFF			ON/OFF		OFF		
BCD(101)	ON/OFF			ON/OFF				
BINL(102)	ON/OFF			ON/OFF		OFF		
BCDL(103)	ON/OFF			ON/OFF				
NEG(104)	ON/OFF			ON/OFF		ON/OFF		
NEGL(105)								
SIGN(106)								
MLPX(110)	ON/OFF							
DMPX(111)								
SDEC(112)								
ASC(113)								
BCNT(114)	ON/OFF			ON/OFF				
LINE(115)								
COLM(116)								
HEX(117)*	ON/OFF							
TTIM(120)								
TIML(121)								
MTIM(122)								
TCNT(123)								
TSR(124)								
TSW(125)								
ANDW(130)	ON/OFF			ON/OFF		ON/OFF		
ORW(131)								
XORW(132)								
XNRW(133)								
ANDL(134)								
ORWL(135)								
XORL(136)								
XNRL(137)								
COM(138)								
COML(139)								

Note "ON/OFF" means that the flag is affected by the result of instruction execution.

Instructions	A50003 (ER)	A50004 (CY)	A50005 (GR)	A50006 (EQ)	A50007 (LE)	A50008 (N)	A50009 (OF)	A50010 (UF)
ROOT(140)	ON/OFF			ON/OFF				
FDIV(141)								
APR(142)	ON/OFF			ON/OFF		ON/OFF		
SEC(143)	ON/OFF			ON/OFF				
HMS(144)								
CADD(145)								
CSUB(146)								
SBS(151)	ON/OFF							
MSKS(153)								
CLI(154)								
MSKR(155)								
MCRO(156)*								
SSET(160)								
PUSH(161)								
LIFO(162)								
FIFO(163)								
SRCH(164)	ON/OFF			ON/OFF				
MAX(165)	ON/OFF			ON/OFF		ON/OFF		
MIN(166)								
SUM(167)								
EMBC(171)	ON/OFF							
CCL(172)	ON/OFF	ON/OFF	ON/OFF	ON/OFF	ON/OFF	ON/OFF	ON/OFF	ON/OFF
REGL(175)	ON/OFF							
REGS(176)								
FPD(177)*	ON/OFF	ON/OFF						
DATE(179)*	ON/OFF							
FILR(180)								
FILW(181)								
FILP(182)								
FLSP(183)								
IODP(189)								
READ(190)	ON/OFF	ON/OFF		ON/OFF				
WRIT(191)								
SEND(192)	ON/OFF							
RECV(193)								
CMND(194)								
MSG(195)								
SA(210)								
CJP(221)*								
CJPN(222)*								
CNR(236)								
RLNC(260)*	ON/OFF	ON/OFF		ON/OFF		ON/OFF		
RRNC(261)*								
RLNL(262)*								
RRNL(263)*								
PID(270)*	ON/OFF	ON/OFF	ON/OFF		ON/OFF			

Note "ON/OFF" means that the flag is affected by the result of instruction execution.

Instructions	A50003 (ER)	A50004 (CY)	A50005 (GR)	A50006 (EQ)	A50007 (LE)	A50008 (N)	A50009 (OF)	A50010 (UF)
LMT(271)*	ON/OFF		ON/OFF	ON/OFF	ON/OFF	ON/OFF		
BAND(272)*								
ZONE(273)*								
ROTB(274)*	ON/OFF			ON/OFF		OFF	ON/OFF	OFF
BINS(275)*	ON/OFF			ON/OFF		ON/OFF		
BCDS(276)*								
BISL(277)*								
BDSL(278)*								
RD2(280)*	ON/OFF	ON/OFF		ON/OFF				
WR2(281)*								
+(400)*	ON/OFF	ON/OFF		ON/OFF		ON/OFF	ON/OFF	ON/OFF
+L(401)*								
+C(402)*								
+CL(403)*								
+B(404)*	ON/OFF	ON/OFF		ON/OFF				
+BL(405)*								
+BC(406)*								
+BCL(407)*								
–(410)*	ON/OFF	ON/OFF		ON/OFF		ON/OFF	ON/OFF	ON/OFF
–L(411)*								
–C(412)*								
–CL(413)*								
–B(414)*	ON/OFF	ON/OFF		ON/OFF				
–BL(415)*								
–BC(416)*								
–BCL(417)*								
∗(420)*	ON/OFF			ON/OFF		ON/OFF		
∗L(421)*								
∗U(422)*								
∗UL(423)*								
∗B(424)*	ON/OFF			ON/OFF				
∗BL(425)*								
/(430)*	ON/OFF			ON/OFF		ON/OFF		
/L(431)*								
/U(432)*								
/UL(433)*								
/B(434)*	ON/OFF			ON/OFF				
/BL(435)*								
FIX(450)*	ON/OFF			ON/OFF		ON/OFF		
FIXL(451)*								
FLT(452)*								
FLTL(453)*								
+F(454)*	ON/OFF			ON/OFF		ON/OFF	ON/OFF	ON/OFF
–F(455)*								
∗F(456)*								
/F(457)*								
RAD(458)*								
DEG(459)*								

Note "ON/OFF" means that the flag is affected by the result of instruction execution.

Instructions	A50003 (ER)	A50004 (CY)	A50005 (GR)	A50006 (EQ)	A50007 (LE)	A50008 (N)	A50009 (OF)	A50010 (UF)
SIN(460)*	ON/OFF			ON/OFF		ON/OFF	OFF	OFF
COS(461)*								
TAN(462)*	ON/OFF			ON/OFF		ON/OFF	ON/OFF	OFF
ASIN(463)*	ON/OFF			ON/OFF		ON/OFF	OFF	OFF
ACOS(464)*	ON/OFF			ON/OFF		OFF	OFF	OFF
ATAN(465)*	ON/OFF			ON/OFF		ON/OFF	OFF	OFF
SQRT(466)*	ON/OFF			ON/OFF		OFF	ON/OFF	OFF
EXP(467)*	ON/OFF			ON/OFF		OFF	ON/OFF	ON/OFF
LOG(468)*	ON/OFF			ON/OFF		ON/OFF	ON/OFF	OFF
TIMW<013>*	ON/OFF							
CNTW<014>*								
TMHW<015>*								

Note “ON/OFF” means that the flag is affected by the result of instruction execution.

Appendix C
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	
	D:Power on flag	
C:Startup mode		PROGRAM
D:Startup processing		Don't transfer program.
E:I/O refresh		Cyclic refreshing
F:Execute control 1	B:Detect low battery	Detect
	S:Error on power off	Fatal
	T:CPU standby	CPU waits
	K:Measure CPU SIOU cycle	Don't measure cycle.
G:Execute control 2	C:Execute process	Asynchronous
	I:I/O interrupt	Nesting
	D:Power OFF interrupt	Disable
	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		10.0 ms
K:1st Rack addr (First words for local racks)		0 for CPU Rack
L:Group 1,2 1st addr (First words for SYSMAC BUS/2 Slaves)		RM0 RM1 RM2 RM3 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)		RM0 RM1 RM2 RM3 CIO 2300 CIO2332 CIO 2364 CIO2396 RM4 RM5 RM6 RM7 CIO 2428 CIO 2460 CIO 2492 CIO 2524
N:Group 3, RT 1st addr (First words for group-3 Slave Racks)		Group 3 (SYSMAC BUS/2): RM0 RM1 RM2 RM3 CIO 0300 CIO 0500 CIO 0700 CIO 0900 Words allocated to Units in order under each Master. 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:Power break (Momentary power interruption time)		0 ms
Q:Cycle time		Cycle variable
R:Watch cycle time (Cycle time monitoring time)		1,000 ms

Parameter	Default value
S:Error log	20 records in A100 through A199
T:IOIF, RT display (Slave display modes at startup)	Mode 1

Appendix D
Data Areas

The data areas in the CV-series PCs are summarized below. These are the same for all PCs unless specified. Only dedicated bits are shown specifically. The use of all other bits is determined either by the System the PC is involved in, e.g., SYSMAC LINK Systems use the Link Area, or by the programmer, e.g., storage of data in the Dm Area.

Area	PC	Range	Function
CIO Area (Core I/O)	CV500/ CVM1-CPU01-EV2	Words: CIO 0000 to CIO 2427 Bits: CIO 000000 to CIO 242715 ($0000 to $097B)	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 *3-3 CIO (Core I/O) Area* for details.
	CV1000/CV2000/ CVM1-CPU11-EV2 CVM1-CPU21-EV2	Words: CIO 0000 to CIO 2555 Bits: CIO 000000 to CIO 255515 ($0000 to $09FB)	
Temporary Relay Area	All	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	All	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	All	Words: A000 to A511 Bits: A00000 to A51115 ($0B00 to $0CFF)	Contains flags and bits with special functions.
Transition Area	CV500	TN0000 to TN0511 ($0D00 to $0D1F)	Transition Flags for the transitions in the SFC program.
	CV1000/CV2000	TN0000 to TN1023 ($0D00 to $0D3F)	
Step Area	CV500	ST0000 to ST0511 ($0E00 to $0E1F)	Step Flags for steps in the SFC program. A step is active when its flag is ON.
	CV1000/CV2000	ST0000 to ST1023 ($0E00 to $0E3F)	
Timer Area	CV500/ CVM1-CPU01-EV2	T0000 to T0511 (Completion Flags: $0F00 to $0F1F Present Values: $1000 to $11FF)	Used to define timers (normal, high-speed, and totalizing) and to access Completion Flags, PV, and SV.
	CV1000/CV2000/ CVM1-CPU11-EV2 CVM1-CPU21-EV2	T0000 to T1023 (Completion Flags: $0F00 to $0F3F Present Values: $1000 to $13FF)	
Counter Area	CV500/ CVM1-CPU01-EV2	C0000 to C0511 (Completion Flags: $0F80 to $0F9F Present Values: $1800 to $19FF)	Used to define counters (normal, reversible, and transition) and to access Completion Flags, PV, and SV.
	CV1000/CV2000/ CVM1-CPU11-EV2 CVM1-CPU21-EV2	C0000 to C1023 (Completion Flags: $0F80 to $0FBF Present Values: $1800 to $1BFF)	
DM Area	CV500/ CVM1-CPU01-EV2	D00000 to D08191 ($2000 to $3FFF)	Used for internal data storage and manipulation.
	CV1000/CV2000/ CVM1-CPU11-EV2 CVM1-CPU21-EV2	D00000 to D24575 ($2000 to $7FFF)	
EM Area	CV1000/CV2000 CVM1-CPU21-EV2	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	All	IR0 to IR2	Used for indirect addressing.
Data registers	All	DR0 to DR2	Generally used for indirect addressing.

Dedicated Bits

Some of the bits in the CPU Bus Link Area and most of the bits in the Auxiliary Area and are dedicated for specific purposes. These are summarized in the following tables. Refer to *3-5 CPU Bus Link Area* and *3-6 Auxiliary Area* for details.

CPU Bus Link Area

Most CPU Bus Link Area bits are in the Data Link Area, used to transfer information between the CPU and CPU Bus Units, but G000 contains flags and control bits relating to PC status and G001 to G004 is the Clock/ Calendar Area.

Word(s)	Bit(s)	Function
G000	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.
	05	Not used.
	06	ON when a non-fatal error has occured. (PC operation continues.)
	07	ON when a fatal error has occured. (PC stops.)
	08 to 10	Not used.
	11	UM Protect Bit. Prevents both reading out and writing to Program Memory when turned ON. Set with the CVSS.
	12	Memory Card Protect Bit. Prevents writing to Memory Card 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.
G 001	00 to 07	Seconds (00 to 59)
	08 to 15	Minutes (00 to 59)
G 002	00 to 07	Hours (00 to 23)
	08 to 15	Day of month (01 to 31)
G 003	00 to 07	Month (1 to 12)
	08 to 15	Year (00 to 99)
G 004	00 to 07	Day of week (00 to 06, 00 = Sun.)

Auxiliary Area

As a rule, Auxiliary Area bits can be used only for the purposes for which they are dedicated. A256 to A511 are read only.

Word(s)	Bit(s)	Function
A000	00 to 10	Not used.
	11	Restart Continuation Bit
	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.
A005	00 to 07	SYSMAC BUS Error Check Bits
	08 to 15	Not used.
A006	00 to 15	Not used.
A007	00 to 15	Momentary Power Interruption Time (BCD)
A008	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	00 to 15	Startup Time (BCD)
A012 to A013	00 to 15	Power Interruption Time (BCD)
A014	00 to 15	Number of Power Interruptions (BCD)
A015	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	Peripheral Service Disable Bit
	05	I/O 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.
A256 to A299	00 to 15	Not used.
A300	00 to 15	Error Log Pointer (binary)
A301	00 to 15	Not used.

Word(s)	Bit(s)	Function
A302	00 to 15	CPU Bus Unit Initializing Flags
A303 to A305	00 to 15	Not used.
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 to 07	Not used.
	08 to 11	Connected Device Code 2: GPC 3: Programming Console
	12 to 14	Not used.
	15	Peripheral Connected Flag
A307	00 to 07	Peripheral Connected Flags for RT #0 to RT #7 of RM/2 #0
	08 to 15	Peripheral Connected Flags for RT #0 to RT #7 of RM/2 #1
A308	00 to 07	Peripheral Connected Flags for RT #0 to RT #7 of RM/2 #2
	08 to 15	Peripheral Connected Flags for RT #0 to RT #7 of RM/2 #3
A309	00 to 15	Peripheral 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	00 to 02	Memory Card Type
	03 to 06	Not used.
	07	Memory Card Format Error Flag
	08	Memory Card Transfer Error Flag
	09	Memory Card Write Error Flag
	10	Memory Card Read Error Flag
	11	FIle Missing Flag
	12	Memory Card Write Flag
	13	Memory Card Instruction Flag
	14	Accessing Memory Card Flag
	15	Memory Card Protected Flag
A344 to 345	00 to 15	Not used.
A346	00 to 15	Number of Words Remaining to transfer to memory card for a file read/write instruction (BCD)
A347 to A399	00 to 15	Not used.
A400	00 to 15	Error Code
A401	00 to 05	Not used.
	06	FALS Error Flag
	07	SFC Fatal Error Flag
	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

Word(s)	Bit(s)	Function
A402	00 to 01	Not used.
	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	Not used.
	11	SFC Non-fatal Error Flag
	12	Indirect DM Error Flag
	13	Jump Error Flag
	14	Not used.
	15	FAL Error Flag
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 A413	00 to 15	Not used.
A414	00 to 15	SFC Fatal Error Code
A415 to A417	00 to 15	Not used.
A418	00 to 15	SFC Non-fatal Error Code
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 13	Not used
	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

Word(s)	Bit(s)	Function
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)
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	0.1-s Clock Pulse
	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

Appendix E
I/O Assignment Sheets

This appendix contains sheets that can be copied by the programmer to record I/O bit allocations and terminal assignments on the Racks, as well as details of work bits, data storage areas, timers, and counters.

Programmer: **Program:** **Date:** **Page:**

Word:	**Unit:**	
Bit	**Field device**	**Notes**
00		
01		
02		
03		
04		
05		
06		
07		
08		
09		
10		
11		
12		
13		
14		
15		

Word:	**Unit:**	
Bit	**Field device**	**Notes**
00		
01		
02		
03		
04		
05		
06		
07		
08		
09		
10		
11		
12		
13		
14		
15		

Word:	**Unit:**	
Bit	**Field device**	**Notes**
00		
01		
02		
03		
04		
05		
06		
07		
08		
09		
10		
11		
12		
13		
14		
15		

Word:	**Unit:**	
Bit	**Field device**	**Notes**
00		
01		
02		
03		
04		
05		
06		
07		
08		
09		
10		
11		
12		
13		
14		
15		

Programmer: **Program:** **Date:** **Page:**

Area:	**Word:**	
Bit	**Usage**	**Notes**
00		
01		
02		
03		
04		
05		
06		
07		
08		
09		
10		
11		
12		
13		
14		
15		

Area:	**Word:**	
Bit	**Usage**	**Notes**
00		
01		
02		
03		
04		
05		
06		
07		
08		
09		
10		
11		
12		
13		
14		
15		

Area:	**Word:**	
Bit	**Usage**	**Notes**
00		
01		
02		
03		
04		
05		
06		
07		
08		
09		
10		
11		
12		
13		
14		
15		

Area:	**Word:**	
Bit	**Usage**	**Notes**
00		
01		
02		
03		
04		
05		
06		
07		
08		
09		
10		
11		
12		
13		
14		
15		

Programmer: **Program:** **Date:** **Page:**

Word	Contents	Notes

Word	Contents	Notes

Programmer: **Program:** **Date:** **Page:**

Timer	Set value	Notes

Counter	Set value	Notes

Appendix F
Program Coding Sheet

The following page can be copied for use in coding ladder diagram programs. It is designed for flexibility, allowing the user to input all required addresses and instructions.

When coding programs, be sure to specify all function codes for instructions and data areas (or # for constant) for operands. These will be necessary when inputting programs though a Programming Console or other Peripheral Device.

Programmer: **Program:** **Date:** **Page:**

Address	Instruction	Operand(s)

Address	Instruction	Operand(s)

Programmer: **Program:** **Date:** **Page:**

Address	Instruction	Operand(s)	Address	Instruction	Operand(s)

Appendix G
Data Conversion Table

Decimal	BCD	Hex	Binary
00	00000000	00	00000000
01	00000001	01	00000001
02	00000010	02	00000010
03	00000011	03	00000011
04	00000100	04	00000100
05	00000101	05	00000101
06	00000110	06	00000110
07	00000111	07	00000111
08	00001000	08	00001000
09	00001001	09	00001001
10	00010000	0A	00001010
11	00010001	0B	00001011
12	00010010	0C	00001100
13	00010011	0D	00001101
14	00010100	0E	00001110
15	00010101	0F	00001111
16	00010110	10	00010000
17	00010111	11	00010001
18	00011000	12	00010010
19	00011001	13	00010011
20	00100000	14	00010100
21	00100001	15	00010101
22	00100010	16	00010110
23	00100011	17	00010111
24	00100100	18	00011000
25	00100101	19	00011001
26	00100110	1A	00011010
27	00100111	1B	00011011
28	00101000	1C	00011100
29	00101001	1D	00011101
30	00110000	1E	00011110
31	00110001	1F	00011111
32	00110010	20	00100000

Appendix H
Extended ASCII

Bits 0 to 3		Bits 4 to 7													
Binary		**0000**	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
	Hex	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**A**	**B**	**C**	**D**	**E**	**F**
0000	**0**	NUL	DLE	Space	0	@	P	`	p						
0001	**1**	SOH	DC_1	!	1	A	Q	a	q						
0010	**2**	STX	DC_2	"	2	B	R	b	r						
0011	**3**	ETX	DC_3	#	3	C	S	c	s						
0100	**4**	EOT	DC_4	$	4	D	T	d	t						
0101	**5**	ENQ	NAK	%	5	E	U	e	u						
0110	**6**	ACK	SYN	&	6	F	V	f	v						
0111	**7**	BEL	ETB	'	7	G	W	g	w						
1000	**8**	BS	CAN	(	8	H	X	h	x						
1001	**9**	HT	EM	)	9	I	Y	i	y						
1010	**A**	LF	SUB	*	:	J	Z	j	z						
1011	**B**	VT	ESC	+	;	K	[	k	{						
1100	**C**	FF	FS	,	<	L	¥	l	|						
1101	**D**	CR	GS	-	=	M	]	m	}						
1110	**E**	S0	RS	.	>	N	^	n	←						
1111	**F**	S1	US	/	?	O	_	o	→						▒

Glossary

action — In SFC programs, the individual executable elements in an action block. An action can be defined either as a ladder diagram or as a single bit in memory.

Action Area — A memory area that contains flags that indicate when actions are active.

action block — A collection of all the actions for a single step in an SFC program. Each action is accompanied by its action qualifier, set value, and feedback variable.

action number — A number assigned to an action. Each action has a unique number. These numbers are used to access and to control the status of the action.

action program — A ladder diagram program written to define an action.

action qualifier — A designation made for a action to control when the action is to be executed in respect to the status of the step.

active status — One of the two main statuses that a step can be in. Active status includes pause, halt, and execute status.

active step — A step that is in either pause, halt, or execute status. There can be more than one active step.

address — A number used to identify the location of data or programming instructions in memory or to identify the location of a network or a unit in a network.

advanced instruction — An instruction input with a function code that handles data processing operations within ladder diagrams, as opposed to a basic instruction, which makes up the fundamental portion of a ladder diagram.

allocation — The process by which the PC assigns certain bits or words in memory for various functions. This includes pairing I/O bits to I/O points on Units.

analog — Something that represents or can process a continuous range of values as opposed to values that can be represented in distinct increments. Something that represents or can process values represented in distinct increments is called digital.

Analog I/O Unit — I/O Units that convert I/O between analog and digital values. An Analog Input Input converts an analog input to a digital value for processing by the PC. An Analog Output Unit converts a digital value to an analog output.

AND — A logic operation whereby the result is true if and only if both premises are true. In ladder-diagram programming the premises are usually ON/OFF states of bits or the logical combination of such states called execution conditions.

AQ — See *action qualifier*.

area — See *data area* and *memory area*.

area prefix — A one or two letter prefix used to identify a memory area in the PC. All memory areas except the CIO area require prefixes to identify addresses in them.

arithmetic shift — A shift operation wherein the carry flag is included in the shift.

ASCII Short for American Standard Code for Information Interchange. ASCII is used to code characters for output to printers and other external devices.

asynchronous execution Execution of programs and servicing operations in which program execution and servicing are not synchronized with each other.

auto-decrement A process that can be used when addressing memory through index registers so that the address in the register is automatically reduced by 1 before each use.

auto-increment A process that can be used when addressing memory through index registers so that the address in the register is automatically increased by 1 after each use.

Auxiliary Area A PC data area allocated to flags and control bits.

auxiliary bit A bit in the Auxiliary Area.

Backplane A base to which Units are mounted to form a Rack. Backplanes provide a series of connectors for these Units along with buses to connect them to the CPU and other Units and wiring to connect them to the Power Supply Unit. Backplanes also provide connectors used to connect them to other Backplanes.

back-up A copy made of existing data to ensure that the data will not be lost even if the original data is corrupted or erased.

bank One of multiple sections of a storage area for data or settings. The EM Area is divided into banks each of which is accessed using the same addresses, but different bank numbers.

BASIC A common programming language. BASIC Units are programmed in BASIC.

basic instruction A fundamental instruction used in a ladder diagram. See *advanced instruction*.

Basic Rack Any of the following Racks: CPU Rack, Expansion CPU Rack, or Expansion I/O Rack.

BASIC Unit A CPU Bus Unit used to run programs in BASIC.

baud rate The data transmission speed between two devices in a system measured in bits per second.

BCD Short for binary-coded decimal.

BCD calculation An arithmetic calculation that uses numbers expressed in binary-coded decimal.

binary A number system where all numbers are expressed in base 2, i.e., numbers are written using only 0's and 1's. Each group of four binary bits is equivalent to one hexadecimal digit. Binary data in memory is thus often expressed in hexadecimal for convenience.

binary calculation An arithmetic calculation that uses numbers expressed in binary.

binary-coded decimal A system used to represent numbers so that every four binary bits is numerically equivalent to one decimal digit.

bit The smallest piece of information that can be represented on a computer. A bit has the value of either zero or one, corresponding to the electrical signals ON

	and OFF. A bit represents one binary digit. Some bits at particular addresses are allocated to special purposes, such as holding the status of input from external devices, while other bits are available for general use in programming.
bit address	The location in memory where a bit of data is stored. A bit address specifies the data area and word that is being addressed as well as the number of the bit within the word.
bit designator	An operand that is used to designate the bit or bits of a word to be used by an instruction.
bit number	A number that indicates the location of a bit within a word. Bit 00 is the rightmost (least-significant) bit; bit 15 is the leftmost (most-significant) bit.
bit-control instruction	An instruction that is used to control the status of an individual bit as opposed to the status of an entire word.
block	See *logic block* and *instruction block*.
buffer	A temporary storage space for data in a computerized device.
building-block PC	A PC that is constructed from individual components, or “building blocks.” With building-block PCs, there is no one Unit that is independently identifiable as a PC. The PC is rather a functional assembly of Units.
bus	A communications path used to pass data between any of the Units connected to it.
bus bar	The line leading down the left and sometimes right side of a ladder diagram. Instruction execution proceeds down the bus bar, which is the starting point for all instruction lines.
bus link	A data link that passed data between two Units across a bus.
byte	A unit of data equivalent to 8 bits, i.e., half a word.
call	A process by which instruction execution shifts from the main program to a subroutine. The subroutine may be called by an instruction or by an interrupt.
Carry Flag	A flag that is used with arithmetic operations to hold a carry from an addition or multiplication operation, or to indicate that the result is negative in a subtraction operation. The carry flag is also used with certain types of shift operations.
central processing unit	A device that is capable of storing programs and data, and executing the instructions contained in the programs. In a PC System, the central processing unit executes the program, processes I/O signals, communicates with external devices, etc.
channel	See *word*.
character code	A numeric (usually binary) code used to represent an alphanumeric character.
checksum	A sum transmitted with a data pack in communications. The checksum can be recalculated from the received data to confirm that the data in the transmission has not been corrupted.
CIO Area	A memory area used to control I/O and to store and manipulate data. CIO Area addresses do not require prefixes.

clock pulse A pulse available at specific bits in memory for use in timing operations. Various clock pulses are available with different pulse widths, and therefore different frequencies.

clock pulse bit A bit in memory that supplies a pulse that can be used to time operations. Various clock pulse bits are available with different pulse widths, and therefore different frequencies.

comparison instruction An instruction used to compare data at different locations in memory to determine the relationship between the data.

Completion Flag A flag used with a timer or counter that turns ON when the timer has timed out or the counter has reached its set value.

condition A symbol placed on an instruction line to indicate an instruction that controls the execution condition for the terminal instruction. Each condition is assigned a bit in memory that determines its status. The status of the bit assigned to each condition determines the next execution condition. Conditions correspond to LOAD, LOAD NOT, AND, AND NOT, OR, or OR NOT instructions.

constant An input for an operand in which the actual numeric value is specified. Constants can be input for certain operands in place of memory area addresses. Some operands must be input as constants.

control bit A bit in a memory area that is set either through the program or via a Programming Device to achieve a specific purpose, e.g., a Restart Bit is turned ON and OFF to restart a Unit.

control data An operand that specifies how an instruction is to be executed. The control data may specify the part of a word is to be used as the operand, it may specify the destination for a data transfer instructions, it may specify the size of a data table used in an instruction, etc.

control signal A signal sent from the PC to effect the operation of the controlled system.

Control System All of the hardware and software components used to control other devices. A Control System includes the PC System, the PC programs, and all I/O devices that are used to control or obtain feedback from the controlled system.

controlled system The devices that are being controlled by a PC System.

count pulse The signal counted by a counter.

counter A dedicated group of digits or words in memory used to count the number of times a specific process has occurred, or a location in memory accessed through a TC bit and used to count the number of times the status of a bit or an execution condition has changed from OFF to ON.

CPU See *central processing unit*.

CPU Backplane A Backplane used to create a CPU Rack.

CPU Bus Unit A special Unit used with CV-series PCs that mounts to the CPU bus. This connection to the CPU bus enables special data links, data transfers, and processing.

CPU Rack The main Rack in a building-block PC, the CPU Rack contains the CPU, a Power Supply, and other Units. The CPU Rack, along with the Expansion CPU Rack, provides both an I/O bus and a CPU bus.

C-series PC Any of the following PCs: C2000H, C1000H, C500, C200H, C40H, C28H, C20H, C60K, C60P, C40K, C40P, C28K, C28P, C20K, C20P, C120, or C20.

custom data area A data area defined by the user within the CIO Area. Custom data areas can be set from the CVSS and certain other Programming Devices.

CV Support Software A programming package run on an IBM PC/AT or compatible to serve as a Programming Device for CV-series PCs.

CV-series PC Any of the following PCs: CV500, CV1000, CV2000, or CVM1

CVSS See *CV Support Software*.

cycle One unit of processing performed by the CPU, including SFC/ladder program execution, peripheral servicing, I/O refreshing, etc. The cycle is called the scan with C-series PCs.

cycle time The time required to complete one cycle of CPU processing.

cyclic interrupt See *scheduled interrupt.*

data area An area in the PC's memory that is designed to hold a specific type of data.

data area boundary The highest address available within a data area. When designating an operand that requires multiple words, it is necessary to ensure that the highest address in the data area is not exceeded.

data link An automatic data transmission operation that allows PCs or Units within PC to pass data back and forth via common data areas.

data movement instruction An instruction used to move data from one location in memory to another. The data in the original memory location is left unchanged.

data register A storage location in memory used to hold data. In CV-series PCs, data registers are used with or without index registers to hold data used in indirect addressing.

data trace A process in which changes in the contents of specific memory locations are recorded during program execution.

data transfer Moving data from one memory location to another, either within the same device or between different devices connected via a communications line or network.

debug A process by which a draft program is corrected until it operates as intended. Debugging includes both the removal of syntax errors, as well as the fine-tuning of timing and coordination of control operations.

DEBUG mode A mode of PC operation which enables basic debugging of user programs.

decimal A number system where numbers are expressed to the base 10. In a PC all data is ultimately stored in binary form, four binary bits are often used to represent one decimal digit, via a system called binary-coded decimal.

decrement Decreasing a numeric value, usually by 1.

default A value automatically set by the PC when the user does not specifically set another value. Many devices will assume such default conditions upon the application of power.

definer A number used as an operand for an instruction but that serves to define the instruction itself, rather that the data on which the instruction is to operate. Definers include jump numbers, subroutine numbers, etc.

destination The location where an instruction places the data on which it is operating, as opposed to the location from which data is taken for use in the instruction. The location from which data is taken is called the source.

differentiated instruction An instruction that is executed only once each time its execution condition goes from OFF to ON. Non-differentiated instructions are executed for each scan as long as the execution condition stays ON.

differentiation instruction An instruction used to ensure that the operand bit is never turned ON for more than one scan after the execution condition goes either from OFF to ON for a Differentiate Up instruction or from ON to OFF for a Differentiate Down instruction.

digit A unit of storage in memory that consists of four bits.

digit designator An operand that is used to designate the digit or digits of a word to be used by an instruction.

DIP switch Dual in-line package switch, an array of pins in a signal package that is mounted to a circuit board and is used to set operating parameters.

distributed control A automation concept in which control of each portion of an automated system is located near the devices actually being controlled, i.e., control is decentralized and 'distributed' over the system. Distributed control is a concept basic to PC Systems.

DM Area A data area used to hold only word data. Words in the DM area cannot be accessed bit by bit.

DM word A word in the DM Area.

downloading The process of transferring a program or data from a higher-level or host computer to a lower-level or slave computer. If a Programming Device is involved, the Programming Device is considered the host computer.

DR See *data register*.

Dummy I/O Unit An I/O Unit that has no functional capabilities but that can be mounted to a slot on a Rack so that words can be allocated to that slot. Dummy I/O Units can be used to avoid changing operand addresses in programs by reserving words for a slot for future use or by filling a slot vacated by a Unit to which words have already been allocated.

EEPROM Electrically erasable programmable read-only memory; a type of ROM in which stored data can be erased and reprogrammed. This is accomplished using a special control lead connected to the EEPROM chip and can be done without having to remove the EEPROM chip from the device in which it is mounted.

electrical noise Random variations of one or more electrical characteristics such as voltage, current, and data, which might interfere with the normal operation of a device.

EM Area Extended Data Memory Area; an area that can be optionally added to certain PCs to enable greater data storage. Functionally, the EM Area operates like the

DM Area. Area addresses are prefixes with E and only words can be accessed. The EM Area is separated into multiple banks.

EM card A card mounted inside certain PCs to added an EM Area.

EPROM Erasable programmable read-only memory; a type of ROM in which stored data can be erased, by ultraviolet light or other means, and reprogrammed.

error code A numeric code generated to indicate that an error exists, and something about the nature of the error. Some error codes are generated by the system; others are defined in the program by the operator.

Error Log Area An area in System DM that is used to store records indicating the time and nature of errors that have occurred in the system.

event processing Processing that is performed in response to an event, e.g., an interrupt signal.

exclusive NOR A logic operation whereby the result is true if both of the premises are true or both of the premises are false. In ladder-diagram programming, the premises are usually the ON/OFF states of bits, or the logical combination of such states, called execution conditions.

exclusive OR A logic operation whereby the result is true if one, and only one, of the premises is true. In ladder-diagram programming the premises are usually the ON/OFF states of bits, or the logical combination of such states, called execution conditions.

execution condition The ON or OFF status under which an instruction is executed. The execution condition is determined by the logical combination of conditions on the same instruction line and up to the instruction currently being executed.

execution cycle The cycle used to execute all processes required by the CPU, including program execution, I/O refreshing, peripheral servicing, etc.

execution time The time required for the CPU to execute either an individual instruction or an entire program.

Expansion CPU Rack A Rack connected to the CPU Rack to increase the virtual size of the CPU Rack. Units that may be mounted to the CPU Backplane may also be mounted to the Expansion CPU Backplane.

Expansion Data Memory Unit A card mounted inside certain PCs to added an EM Area.

Expansion I/O Rack A Rack used to increase the I/O capacity of a PC. In CV-Series PC, either one Expansion I/O Rack can be connected directly to the CPU or Expansion CPU Rack or multiple Expansion I/O Racks can be connected by using an I/O Control and I/O Interface Units.

extended counter A counter created in a program by using two or more count instructions in succession. Such a counter is capable of counting higher than any of the standard counters provided by the individual instructions.

extended timer A timer created in a program by using two or more timers in succession. Such a timer is capable of timing longer than any of the standard timers provided by the individual instructions.

FA Factory automation.

factory computer A general-purpose computer, usually quite similar to a business computer, that is used in automated factory control.

FAL error An error generated from the user program by execution of an FAL(006) instruction.

FALS error An error generated from the user program by execution of an FALS(007) instruction or an error generated by the system.

fatal error An error that stops PC operation and requires correction before operation can continue.

FINS See *CV-mode*.

flag A dedicated bit in memory that is set by the system to indicate some type of operating status. Some flags, such as the carry flag, can also be set by the operator or via the program.

flicker bit A bit that is programmed to turn ON and OFF at a specific frequency.

floating-point decimal A decimal number expressed as a number (the mantissa) multiplied by a power of 10, e.g., 0.538×10^{-5}.

force reset The process of forcibly turning OFF a bit via a programming device. Bits are usually turned OFF as a result of program execution.

force set The process of forcibly turning ON a bit via a programming device. Bits are usually turned ON as a result of program execution.

forced status The status of bits that have been force reset or force set.

frame checksum The results of exclusive ORing all data within a specified calculation range. The frame checksum can be calculated on both the sending and receiving end of a data transfer to confirm that data was transmitted correctly.

function code A two-digit number used to input an instruction into the PC.

GPC An acronym for Graphic Programming Console.

Graphic Programming Console A programming device with advanced programming and debugging capabilities to facilitate PC operation. A Graphic Programming Console is provided with a large display onto which ladder-diagram programs can be written directly in ladder-diagram symbols for input into the PC without conversion to mnemonic form.

hardware error An error originating in the hardware structure (electronic components) of the PC, as opposed to a software error, which originates in software (i.e., programs).

hexadecimal A number system where all numbers are expressed to the base 16. In a PC all data is ultimately stored in binary form, however, displays and inputs on Programming Devices are often expressed in hexadecimal to simplify operation. Each group of four binary bits is numerically equivalent to one hexadecimal digit.

hold bit A bit in memory designated to maintain status when the PC's operating mode is changed or power is turned off and then back on.

hold Rack A Rack designated to maintain output status when the PC's operating mode is changed or power is turned off and then back on.

holding area Words in memory designated to maintain status when the PC's operating mode is changed or power is turned off and then back on.

host interface An interface that allows communications with a host computer.

Host Link System A system with one or more host computers connected to one or more PCs via Host Link Units or host interfaces so that the host computer can be used to transfer data to and from the PC(s). Host Link Systems enable centralized management and control of PC Systems.

Host Link Unit An interface used to connect a C-series PC to a host computer in a Host Link System.

I/O allocation The process by which the PC assigns certain bits in memory for various functions. This includes pairing I/O bits to I/O points on Units.

I/O bit A bit in memory used to hold I/O status. Input bits reflect the status of input terminals; output bits hold the status for output terminals.

I/O Block Either an Input Block or an Output Block. I/O Blocks provide mounting positions for replaceable relays.

I/O capacity The number of inputs and outputs that a PC is able to handle. This number ranges from around one hundred for smaller PCs to two thousand for the largest ones.

I/O Control Unit A Unit mounted to the CPU Rack to monitor and control I/O points on Expansion CPU Racks or Expansion I/O Racks.

I/O delay The delay in time from when a signal is sent to an output to when the status of the output is actually in effect or the delay in time from when the status of an input changes until the signal indicating the change in the status is received.

I/O device A device connected to the I/O terminals on I/O Units, Special I/O Units, etc. I/O devices may be either part of the Control System, if they function to help control other devices, or they may be part of the controlled system.

I/O Interface Unit A Unit mounted to an Expansion CPU Rack or Expansion I/O Rack to interface the Rack to the CPU Rack.

I/O interrupt An interrupt generated by a signal from I/O.

I/O point The place at which an input signal enters the PC System, or at which an output signal leaves the PC System. In physical terms, I/O points correspond to terminals or connector pins on a Unit; in terms of programming, an I/O points correspond to I/O bits in the IR area.

I/O refreshing The process of updating output status sent to external devices so that it agrees with the status of output bits held in memory and of updating input bits in memory so that they agree with the status of inputs from external devices.

I/O response time The time required for an output signal to be sent from the PC in response to an input signal received from an external device.

I/O table A table created within the memory of the PC that lists the I/O words allocated to each Unit in the PC System. The I/O table can be created by, or modified from, a Programming Device.

I/O Terminal A Remote I/O Unit connected in a Wired Remote I/O System to provide a limited number of I/O points at one location. There are several types of I/O Terminals.

I/O Unit The most basic type of Unit mounted to a Backplane. I/O Units include Input Units and Output Units, each of which is available in a range of specifications. I/O Units do not include Special I/O Units, Link Units, etc.

I/O verification error A error generated by a disagreement between the Units registered in the I/O table and the Units actually mounted to the PC.

I/O word A word in the CIO area that is allocated to a Unit in the PC System and is used to hold I/O status for that Unit.

IBM PC/AT or compatible A computer that has similar architecture to, that is logically compatible with, and that can run software designed for an IBM PC/AT computer.

immediate refreshing A form of I/O refreshing that is executed by certain types of instruction when the instruction is executed to ensure that the most current input status is used for an operand or to ensure that an output is effective immediately.

increment Increasing a numeric value, usually by 1.

index register A data storage location used with or without a data register in indirect addressing.

indirect address An address whose contents indicates another address. The contents of the second address will be used as the actual operand.

initialization error An error that occurs either in hardware or software during the PC System startup, i.e., during initialization.

initialize Part of the startup process whereby some memory areas are cleared, system setup is checked, and default values are set.

input The signal coming from an external device into the PC. The term input is often used abstractly or collectively to refer to incoming signals.

input bit A bit in the CIO area that is allocated to hold the status of an input.

Input Block A Unit used in combination with a Remote Interface to create an I/O Terminal. An Input Block provides mounting positions for replaceable relays. Each relay can be selected according to specific input requirements.

input device An external device that sends signals into the PC System.

input point The point at which an input enters the PC System. Input points correspond physically to terminals or connector pins.

input signal A change in the status of a connection entering the PC. Generally an input signal is said to exist when, for example, a connection point goes from low to high voltage or from a nonconductive to a conductive state.

Input Terminal An I/O Terminal that provides input points.

instruction A direction given in the program that tells the PC of the action to be carried out, and the data to be used in carrying out the action. Instructions can be used to simply turn a bit ON or OFF, or they can perform much more complex actions, such as converting and/or transferring large blocks of data.

instruction block A group of instructions that is logically related in a ladder-diagram program. A logic block includes all of the instruction lines that interconnect with each other from one or more line connecting to the left bus bar to one or more right-hand instructions connecting to the right bus bar.

instruction execution time The time required to execute an instruction. The execution time for any one instruction can vary with the execution conditions for the instruction and the operands used in it.

instruction line A group of conditions that lie together on the same horizontal line of a ladder diagram. Instruction lines can branch apart or join together to form instruction blocks. Also called a rung.

interface An interface is the conceptual boundary between systems or devices and usually involves changes in the way the communicated data is represented. Interface devices such as NSBs perform operations like changing the coding, format, or speed of the data.

interlock A programming method used to treat a number of instructions as a group so that the entire group can be reset together when individual execution is not required. An interlocked program section is executed normally for an ON execution condition and partially reset for an OFF execution condition.

intermediate instruction An instruction other than one corresponding to a condition that appears in the middle of an instruction line and requires at least one more instruction between it and the right bus bar.

interrupt (signal) A signal that stops normal program execution and causes a subroutine to be run or other processing to take place.

Interrupt Input Unit A Rack-mounting Unit used to input external interrupts into a PC System.

interrupt program A program that is executed in response to an interrupt.

inverse condition See *normally closed condition*.

IOIF An acronym for I/O Interface Unit.

IOM (Area) A collective memory area containing all of the memory areas that can be accessed by bit, including timer and counter Completion Flags. The IOM Area includes all memory area memory addresses between 0000 and 0FFF.

JIS An acronym for Japanese Industrial Standards.

jump A type of programming where execution moves directly from one point in a program to another, without sequentially executing any instructions in between. Jumps in ladder diagrams are usually conditional on an execution condition; jumps in SFC programs are conditional on the step status and transition condition status before the jump.

jump number A definer used with a jump that defines the points from and to which a jump is to be made.

ladder diagram (program) A form of program arising out of relay-based control systems that uses circuit-type diagrams to represent the logic flow of programming instructions. The appearance of the program is similar to a ladder, and thus the name.

ladder diagram symbol A symbol used in drawing a ladder-diagram program.

ladder instruction An instruction that represents the conditions on a ladder-diagram program. The other instructions in a ladder diagram fall along the right side of the diagram and are called terminal instructions.

least-significant (bit/word) See *rightmost (bit/word)*.

LED Acronym for light-emitting diode; a device used as for indicators or displays.

leftmost (bit/word) The highest numbered bits of a group of bits, generally of an entire word, or the highest numbered words of a group of words. These bits/words are often called most-significant bits/words.

link A hardware or software connection formed between two Units. “Link” can refer either to a part of the physical connection between two Units or a software connection created to data existing at another location (i.e., data links).

Link Area A data area that is designed for use in data links.

Link System A system used to connect remote I/O or to connect multiple PCs in a network. Link Systems include the following: SYSMAC BUS Remote I/O Systems, SYSMAC BUS/2 Remote I/O Systems, SYSMAC LINK Systems, Host Link Systems, and SYSMAC NET Link Systems.

Link Unit Any of the Units used to connect a PC to a Link System. These include Remote I/O Units, SYSMAC LINK Units, and SYSMAC NET Link Units.

load The processes of copying data either from an external device or from a storage area to an active portion of the system such as a display buffer. Also, an output device connected to the PC is called a load.

logic block A group of instructions that is logically related in a ladder-diagram program and that requires logic block instructions to relate it to other instructions or logic blocks.

logic block instruction An instruction used to locally combine the execution condition resulting from a logic block with a current execution condition. The current execution condition could be the result of a single condition, or of another logic block. AND Load and OR Load are the two logic block instructions.

logic instruction Instructions used to logically combine the content of two words and output the logical results to a specified result word. The logic instructions combine all the same-numbered bits in the two words and output the result to the bit of the same number in the specified result word.

loop A group of instructions that can be executed more than once in succession (i.e., repeated) depending on an execution condition or bit status.

main program All of a program except for subroutine and interrupt programs.

mark trace A process in which changes in the contents of specific memory locations are recorded during program execution using MARK (174) instructions.

masked bit A bit whose status has been temporarily made ineffective.

masking ‘Covering’ an interrupt signal so that the interrupt is not effective until the mask is removed.

MCR Unit Magnetic Card Reader Unit.

megabyte A unit of storage equal to one million bytes.

memory area Any of the areas in the PC used to hold data or programs.

memory card A data storage media similar to a floppy disk.

message number A number assigned to a message generated with the MSG(195) instruction.

mnemonic code A form of a ladder-diagram program that consists of a sequential list of the instructions without using a ladder diagram.

MONITOR mode A mode of PC operation in which normal program execution is possible, and which allows modification of data held in memory. Used for monitoring or debugging the PC.

most-significant (bit/word) See *leftmost (bit/word).*

NC input An input that is normally closed, i.e., the input signal is considered to be present when the circuit connected to the input opens.

negative delay A delay set for a data trace in which recording data begins before the trace signal by a specified amount.

nesting Programming one loop within another loop, programming a call to a subroutine within another subroutine, or programming an IF-ELSE programming section within another IF-ELSE section.

Network Service Board A device with an interface to connect devices other than PCs to a SYSMAC NET Link System.

Network Service Unit A Unit that provides two interfaces to connect peripheral devices to a SYSMAC NET Link System.

NO input An input that is normally open, i.e., the input signal is considered to be present when the circuit connected to the input closes.

noise interference Disturbances in signals caused by electrical noise.

nonfatal error A hardware or software error that produces a warning but does not stop the PC from operating.

normal condition See *normally open condition*.

normally closed condition A condition that produces an ON execution condition when the bit assigned to it is OFF, and an OFF execution condition when the bit assigned to it is ON.

normally open condition A condition that produces an ON execution condition when the bit assigned to it is ON, and an OFF execution condition when the bit assigned to it is OFF.

NOT A logic operation which inverts the status of the operand. For example, AND NOT indicates an AND operation with the opposite of the actual status of the operand bit.

octal A number system where all numbers are expressed in base 8, i.e., numbers are written using only numerals 0 through 7.

OFF The status of an input or output when a signal is said not to be present. The OFF state is generally represented by a low voltage or by non-conductivity, but can be defined as the opposite of either.

OFF delay The delay between the time when a signal is switched OFF (e.g., by an input device or PC) and the time when the signal reaches a state readable as an OFF signal (i.e., as no signal) by a receiving party (e.g., output device or PC).

offset A positive or negative value added to a base value such as an address to specify a desired value.

ON The status of an input or output when a signal is said to be present. The ON state is generally represented by a high voltage or by conductivity, but can be defined as the opposite of either.

ON delay The delay between the time when an ON signal is initiated (e.g., by an input device or PC) and the time when the signal reaches a state readable as an ON signal by a receiving party (e.g., output device or PC).

one-shot bit A bit that is turned ON or OFF for a specified interval of time which is longer than one scan.

operand The values designated as the data to be used for an instruction. An operand can be input as a constant expressing the actual numeric value to be used or as an address to express the location in memory of the data to be used.

operand bit A bit designated as an operand for an instruction.

operand word A word designated as an operand for an instruction.

operating error An error that occurs during actual PC operation as opposed to an initialization error, which occurs before actual operations can begin.

OR A logic operation whereby the result is true if either of two premises is true, or if both are true. In ladder-diagram programming the premises are usually ON/OFF states of bits or the logical combination of such states called execution conditions.

output The signal sent from the PC to an external device. The term output is often used abstractly or collectively to refer to outgoing signals.

output bit A bit in the IR area that is allocated to hold the status to be sent to an output device.

Output Block A Unit used in combination with a Remote Interface to create an I/O Terminal. An Output Block provides mounting positions for replaceable relays. Each relay can be selected according to specific output requirements.

output device An external device that receives signals from the PC System.

output point The point at which an output leaves the PC System. Output points correspond physically to terminals or connector pins.

output signal A signal being sent to an external device. Generally an output signal is said to exist when, for example, a connection point goes from low to high voltage or from a nonconductive to a conductive state.

Output Terminal An I/O Terminal that provides output points.

overflow The state where the capacity of a data storage location has been exceeded.

overseeing Part of the processing performed by the CPU that includes general tasks required to operate the PC.

overwrite Changing the content of a memory location so that the previous content is lost.

Parameter Area A part of System DM used to designate various PC operating parameters.

Parameter Backup Area A part of System DM used to back up the Parameter Area.

parity Adjustment of the number of ON bits in a word or other unit of data so that the total is always an even number or always an odd number. Parity is generally used to check the accuracy of data after being transmitted by confirming that the number of ON bits is still even or still odd.

parity check Checking parity to ensure that transmitted data has not been corrupted.

PC An acronym for Programmable Controller.

PC configuration The arrangement and interconnections of the Units that are put together to form a functional PC.

PC System With building-block PCs, all of the Racks and independent Units connected directly to them up to, but not including the I/O devices. The boundaries of a PC System are the PC and the program in its CPU at the upper end; and the I/O Units, Special I/O Units, Optical I/O Units, Remote Terminals, etc., at the lower end.

PCB An acronym for printed circuit board.

PC Setup A group of operating parameters set in the PC from a Programming Device to control PC operation.

Peripheral Device Devices connected to a PC System to aid in system operation. Peripheral devices include printers, programming devices, external storage media, etc.

peripheral servicing Processing signals to and from peripheral devices, including refreshing, communications processing, interrupts, etc.

PID Unit A Unit designed for PID control.

pointer A variable or register which contains the address of some object in memory.

positive delay A delay set for a data trace in which recording data begins after the trace signal by a specified amount.

power-off interrupt An interrupt executed when power to the PC is turned off.

power-on interrupt An interrupt executed when power to the PC is turned on.

present value The current value registered in a device at any instant during its operation. Present value is abbreviated as PV. The use of this term is generally restricted to timers and counters.

printed circuit board A board onto which electrical circuits are printed for mounting into a computer or electrical device.

PROGRAM mode A mode of operation that allows inputting and debugging of programs to be carried out, but that does not permit normal execution of the program.

Programmable Controller A computerized device that can accept inputs from external devices and generate outputs to external devices according to a program held in memory. Pro-

grammable Controllers are used to automate control of external devices. Although single-unit Programmable Controllers are available, building-block Programmable Controllers are constructed from separate components. Such Programmable Controllers are formed only when enough of these separate components are assembled to form a functional assembly, i.e., there is no one individual Unit called a PC.

programmed alarm An alarm given as a result of execution of an instruction designed to generate the alarm in the program, as opposed to one generated by the system.

programmed error An error arising as a result of the execution of an instruction designed to generate the error in the program, as opposed to one generated by the system.

programmed message A message generated as a result of execution of an instruction designed to generate the message in the program, as opposed to one generated by the system.

Programming Console The simplest form or programming device available for a PC. Programming Consoles are available both as hand-held models and as CPU-mounting models.

Programming Device A Peripheral Device used to input a program into a PC or to alter or monitor a program already held in the PC. There are dedicated programming devices, such as Programming Consoles, and there are non-dedicated devices, such as a host computer.

PROM Programmable read-only memory; a type of ROM into which the program or data may be written after manufacture, by a customer, but which is fixed from that time on.

PROM Writer A peripheral device used to write programs and other data into a ROM for permanent storage and application.

prompt A message or symbol that appears on a display to request input from the operator.

protocol The parameters and procedures that are standardized to enable two devices to communicate or to enable a programmer or operator to communicate with a device.

PV See *present value*.

Rack An assembly that forms a functional unit in a Rack PC System. A Rack consists of a Backplane and the Units mounted to it. These Units include the Power Supply, CPU, and I/O Units. Racks include CPU Racks, Expansion I/O Racks, and I/O Racks. The CPU Rack is the Rack with the CPU mounted to it. An Expansion I/O Rack is an additional Rack that holds extra I/O Units. An I/O Rack is used in the C2000H Duplex System, because there is no room for any I/O Units on the CPU Rack in this System.

rack number A number assigned to a Rack according to the order that it is connected to the CPU Rack, with the CPU Rack generally being rack number 0.

Rack PC A PC that is composed of Units mounted to one or more Racks. This configuration is the most flexible, and most large PCs are Rack PCs. A Rack PC is the opposite of a Package-type PC, which has all of the basic I/O, storage, and control functions built into a single package.

RAM Random access memory; a data storage media. RAM will not retain data when power is disconnected.

RAS An acronym for reliability, assurance, safety.

read-only area A memory area from which the user can read status but to which data cannot be written.

refresh The process of updating output status sent to external devices so that it agrees with the status of output bits held in memory and of updating input bits in memory so that they agree with the status of inputs from external devices.

refresh period The period during which memory status is actual read and written, e.g., during the I/O refresh period, input bit status is written to agree with input terminal status and output terminals are changes to agree with output bit status.

Register Area A memory are that contains both index registers and data registers.

relay-based control The forerunner of PCs. In relay-based control, groups of relays are interconnected to form control circuits. In a PC, these are replaced by programmable circuits.

remote I/O word An I/O word allocated to a Unit in a Remote I/O System.

reserved bit A bit that is not available for user application.

reserved word A word in memory that is reserved for a special purpose and cannot be accessed by the user.

reset The process of turning a bit or signal OFF or of changing the present value of a timer or counter to its set value or to zero.

Restart Bit A bit used to restart a Unit mounted to a PC.

restart continuation A process which allows memory and program execution status to be maintained so that PC operation can be restarted from the state it was in when operation was stopped by a power interruption.

result word A word used to hold the results from the execution of an instruction.

retrieve The processes of copying data either from an external device or from a storage area to an active portion of the system such as a display buffer. Also, an output device connected to the PC is called a load.

retry The process whereby a device will re-transmit data which has resulted in an error message from the receiving device.

return The process by which instruction execution shifts from a subroutine back to the main program (usually the point from which the subroutine was called).

reversible counter A counter that can be both incremented and decremented depending on the specified conditions.

reversible shift register A shift register that can shift data in either direction depending on the specified conditions.

right-hand instruction See *terminal instruction*.

rightmost (bit/word) The lowest numbered bits of a group of bits, generally of an entire word, or the lowest numbered words of a group of words. These bits/words are often called least-significant bits/words.

rising edge The point where a signal actually changes from an OFF to an ON status.

ROM Read only memory; a type of digital storage that cannot be written to. A ROM chip is manufactured with its program or data already stored in it and can never be changed. However, the program or data can be read as many times as desired.

rotate register A shift register in which the data moved out from one end is placed back into the shift register at the other end.

RS-232C interface An industry standard for serial communications.

RS-422 interface An industry standard for serial communications.

RUN mode The operating mode used by the PC for normal control operations.

rung See *instruction line*.

scan The process used to execute a ladder-diagram program. The program is examined sequentially from start to finish and each instruction is executed in turn based on execution conditions. The scan also includes peripheral processing, I/O refreshing, etc. The scan is called the cycle with CV-series PCs.

scan time The time required for a single scan of a ladder-diagram program.

scheduled interrupt An interrupt that is automatically generated by the system at a specific time or program location specified by the operator. Scheduled interrupts result in the execution of specific subroutines that can be used for instructions that must be executed repeatedly at a specified interval of time.

seal See *self-maintaining bit*.

self diagnosis A process whereby the system checks its own operation and generates a warning or error if an abnormality is discovered.

self-maintaining bit A bit that is programmed to maintain either an OFF or ON status until set or reset by specified conditions.

series A wiring method in which Units are wired consecutively in a string. In Link Systems wired through Link Adapters, the Units are still functionally wired in series, even though Units are placed on branch lines.

servicing The process whereby the PC provides data to or receives data from external devices or remote I/O Units, or otherwise handles data transactions for Link Systems.

set The process of turning a bit or signal ON.

set value The value from which a decrementing counter starts counting down or to which an incrementing counter counts up (i.e., the maximum count), or the time from which or for which a timer starts timing. Set value is abbreviated SV.

shift register One or more words in which data is shifted a specified number of units to the right or left in bit, digit, or word units. In a rotate register, data shifted out one end is shifted back into the other end. In other shift registers, new data (either specified data, zero(s) or one(s)) is shifted into one end and the data shifted out at the other end is lost.

signed binary A binary value that is stored in memory along with a bit that indicates whether the value is positive or negative.

software error An error that originates in a software program.

software protect A means of protecting data from being changed that uses software as opposed to a physical switch or other hardware setting.

software switch See *memory switch*.

source (word) The location from which data is taken for use in an instruction, as opposed to the location to which the result of an instruction is to be written. The latter is called the destination.

Special I/O Unit A Unit that is designed for a specific purpose. Special I/O Units include Position Control Units, High-speed Counter Units, Analog I/O Units, etc.

SRAM Static random access memory; a data storage media.

stack instruction An instruction that manipulates data in a stack.

stack pointer The register which points to the top of the stack. See *stack* and *pointer*.

step A basic unit of execution in an SFC program. Steps are used to organize an SFC program by process and control the overall flow of program execution.

Step Area A memory area that contains a flag that indicates the status of steps in an SFC program.

subroutine A group of instructions placed separate from the main program and executed only when called from the main program or activated by an interrupt.

subroutine number A definer used to identify the subroutine that a subroutine call or interrupt activates.

SV Abbreviation for set value.

synchronous execution Execution of programs and servicing operations in which program execution and servicing are synchronized so that all servicing operations are executed each time the programs are executed.

syntax The form of a program statement (as opposed to its meaning). For example, the two statements, LET A=B+B and LET A=B*2 use different syntaxes, but have the same meaning.

syntax error An error in the way in which a program is written. Syntax errors can include 'spelling' mistakes (i.e., a function code that does not exist), mistakes in specifying operands within acceptable parameters (e.g., specifying read-only bits as a destination), and mistakes in actual application of instructions (e.g., a call to a subroutine that does not exist).

system configuration The arrangement in which Units in a System are connected. This term refers to the conceptual arrangement and wiring together of all the devices needed to comprise the System. In OMRON terminology, system configuration is used to describe the arrangement and connection of the Units comprising a Control System that includes one or more PCs.

System DM A dedicated portion of the DM area that is used for special purposes in controlling and managing the PC. Includes the Program Version, Parameter Area, Parameter Backup Area, User Program Header, and Error Log Area.

system error An error generated by the system, as opposed to one resulting from execution of an instruction designed to generate an error.

system error message An error message generated by the system, as opposed to one resulting from execution of an instruction designed to generate a message.

terminal instruction An instruction placed on the right side of a ladder diagram that uses the final execution conditions of an instruction line.

terminator The code comprising an asterisk and a carriage return (* CR) which indicates the end of a block of data in communications between devices. Frames within a multi-frame block are separated by delimiters. Also a Unit in a Link System designated as the last Unit on the communications line.

timer A location in memory accessed through a TC bit and used to time down from the timer's set value. Timers are turned ON and reset according to their execution conditions.

TR Area A data area used to store execution conditions so that they can be reloaded later for use with other instructions.

TR bit A bit in the TR Area.

trace An operation whereby the program is executed and the resulting data is stored to enable step-by-step analysis and debugging.

trace memory A memory area used to store the results of trace operations.

transfer The process of moving data from one location to another within the PC, or between the PC and external devices. When data is transferred, generally a copy of the data is sent to the destination, i.e., the content of the source of the transfer is not changed.

transition A status in a SFC program that determines when active status is transferred from one step to another. Transitions can be defined either as the status of a bit or as an execution condition resulting from a ladder diagram.

Transition Area A memory area that contains Transition Flags.

Transition Flag A flag that indicates when a transition is ON or OFF.

transition number A number assigned to a transition and used to access its Transition Flag.

transmission distance The distance that a signal can be transmitted.

trigger A signal used to activate some process, e.g., the execution of a trace operation.

trigger address An address in the program that defines the beginning point for tracing. The actual beginning point can be altered from the trigger by defining either a positive or negative delay.

UM area The memory area used to hold the active program, i.e., the program that is being currently executed.

Unit In OMRON PC terminology, the word Unit is capitalized to indicate any product sold for a PC System. Though most of the names of these products end with the word Unit, not all do, e.g., a Remote Terminal is referred to in a collective sense as a Unit. Context generally makes any limitations of this word clear.

unit address A number used to control network communications. Unit addresses are computed for Units in various ways, e.g., 10 hex is added to the unit number to determine the unit address for a CPU Bus Unit.

unit number A number assigned to some Link Units, Special I/O Units, and CPU Bus Units to facilitate identification when assigning words or other operating parameters.

unmasked bit A bit whose status is effective. See *masked bit*.

unsigned binary A binary value that is stored in memory without any indication of whether it is positive or negative.

uploading The process of transferring a program or data from a lower-level or slave computer to a higher-level or host computer. If a Programming Devices is involved, the Programming Device is considered the host computer.

watchdog timer A timer within the system that ensures that the scan time stays within specified limits. When limits are reached, either warnings are given or PC operation is stopped depending on the particular limit that is reached.

WDT See *watchdog timer*.

wire communications A communications method in which signals are sent over wire cable. Although noise resistance and transmission distance can sometimes be a problem with wire communications, they are still the cheapest and the most common, and perfectly adequate for many applications.

word A unit of data storage in memory that consists of 16 bits. All data areas consists of words. Some data areas can be accessed only by words; others, by either words or bits.

word address The location in memory where a word of data is stored. A word address must specify (sometimes by default) the data area and the number of the word that is being addressed.

word allocation The process of assigning I/O words and bits in memory to I/O Units and terminals in a PC System to create an I/O Table.

Word Grouping See *custom data area*.

work area A part of memory containing work words/bits.

work bit A bit in a work word.

work word A word that can be used for data calculation or other manipulation in programming, i.e., a ‘work space’ in memory. A large portion of the IR area is always reserved for work words. Parts of other areas not required for special purposes may also be used as work words.

write protect switch A switch used to write-protect the contents of a storage device, e.g., a floppy disk. If the hole on the upper left of a floppy disk is open, the information on this floppy disk cannot be altered.

write-protect A state in which the contents of a storage device can be read but cannot be altered.

zero-cross refresh An I/O refresh process in which I/O status is refreshed when the voltage of an AC power supply is at zero volts.

Index

A

B

C

D

E

F

G

H

I

J–L

M

N

O

P

R

S

T

U

V–Z

Revision History

A manual revision code appears as a suffix to the catalog number on the front cover of the manual.

Cat. No. W202-E1-5
Revision code

The following table outlines changes made to the manual. Page numbers refer to the previous version.

Revision code	Date	Revised content
1	June 1992	Original production
1A	July 1992	**Page 58:** Descriptions for “Greater Than Flag, GR” and “Less Than Flag, LE” changed.
2	January 1993	PC setup defaults added to appendices and following corrections made. **Page 7 and 8:** Personal Computer Unit added to table. **Page 8:** Unclear entries removed from table and maximum number of words transferable in SYSMAC NET System corrected to 990. **Page 17:** Notes added to table. **Page 20:** “EEPROM” removed from last paragraph. **Page 24:** Note on restrictions on Peripheral Devices connected to Slaves added. **Page 42 and others:** Maximum value for seconds corrected to 59. **Page 56:** “Group-2” corrected to “Group–1,” “Group-3” corrected to “Group–2,” and “(Group-3 Slaves)” added to fifth line of bottom table. **Page 58:** “First” and “second” reversed for GR and LE. **Page 65:** Last address in second line corrected to CIO 1897. **Page 71:** First operand in table corrected to 000000. **Page 72:** First instruction in second mnemonic table corrected to LD NOT. **Page 75:** Vertical line between conditions removed. **Page 76:** Mnemonic tables corrected for top ladder diagram. **Page 81 and others:** Leading zeros added to make six-digit addresses for CIO Area. **Page 89:** “, and certain unused bits in the AR area,” removed from second paragraph in section 4-9. **Page 99/100:** Missing text restored. **Page 102 and others:** Leading zeros cut to make four-digit timer addresses. **Page 103 and 104:** DM removed as a possible operand. **Page 109:** Timing chart lines for ↑SET and ↓SET reversed. **Page 117:** Temporary bits removed from program; MOVR added; mnemonics corrected. **Page 118, 119, 121, 122,:** Precaution added; example programs modified accordingly. **Page 122:** Second LD in second program changed to AND. **Page 125:** LD added to mnemonics and precaution added. **Page 142 and 143:** Description of ASFT(052) corrected and revised, including reversing direction of data shift. **Page 145:** Precaution added. **Page 155:** Operands for OUT in top diagram corrected. **Page 159:** “Upper limits” changed to “Compare table” in diagram. **Page 179:** Second operand for second SUB(071) corrected. **Page 187:** Positions of “=R” and other callouts lowered one line. **Page 187, 189, 210, 253, 271, 272:** “025504” changed to A50004,” “025515” changed to A50015,” and “025512” changed to A50012.” **Page 207:** “02AE” changed to “E02A.” **Page 214:** Second and next to last lines of bottom table corrected. **Page 217:** Description for *Example* corrected. **Page 230, 232, and 234:** Ladder diagrams corrected and note added (page 232). **Page 254:** Direction of arrow reversed. **Page 263, 266, 268:** Precaution added. **Page 264 and 266:** Control data and node number corrected in *Example*. **Page 280:** “D100000” corrected to “D10000” in first paragraph of *Example*. **Page 291:** Program modified. **Page 298 and 300:** Input and output refreshing actions modified. **Page 301:** Timer refreshing actions modified. **Page 325:** “(SYSMAC BUS/2)” added to L and M settings and “(SYSMAC BUS/2 and SYSMAC BUS)” added to N setting. **Page 328:** First “SYSMAC BUS” corrected to “SYSMAC BUS/2” in N setting. **Page 329 and 330:** “IORF” corrected to “IOIF” in T setting. **Page 344:** Following mnemonics corrected: 030: MOVE to MOV; 078: SLD to STC; and 079: SRD to CLC. **Page 357 and 358:** DM removed as possible operand.

Revision code	Date	Revised content
3	March 1993	CV2000, CVM1, and Personal Computer Unit added. **Page 18:** Program area capacity corrected in tables. **Page 36:** Notes concerning Units mounted to Slave Racks and use of Intelligent I/O Read/Write instructions corrected. **Page 116:** Description of refreshing of Completion Flags corrected. **Page 119:** Precaution on using Completion Flag to reset timer removed. **Page 143:** Instruction added for example. **Pages 259 and 260:** CY Flag operation added to I/O READ and I/O WRITE instructions. **Pages 302 and 303:** "Data area write" removed from list of events. **Page 329:** Designation for momentary power interruption time corrected to 0 to 9 ms.
3A	December 1993	Several new functions have been added to the CPUs of CV-series PCs (CVM1, CV500, CV1000, and CV2000). The new CPUs have an EV1 suffix. **Page 15:** Improved Specifications 5, 6, and 7 added. **Page 20 :** "See note 2" deleted from table at bottom of page. **Page 247:** Program example corrected and sentence added to Precautions.
4	February 1995	The following major revisions were implemented: Information added on version-2 CVM1 CPUs (see end of section 1 for overview). Programming examples were added for most instructions. Information in section 5 was reorganized to aid understanding and reduce the redundancy of information (mostly precautionary). SSS added.
4A	July 1995	The following additions/corrections were made. **Page 7:** Motion Control Unit added to table. **Page 16:** Special I/O Units readable/writeable on Slave Racks revised. **Page 21:** Default communications setting modified. **Pages 40 and 41:** CT021 and Motion Control Units added. **Pages 52 and 115:** Note added. **Page 91:** Instruction corrected in note. **Page 139:** Jump number changed and caution added. **Page 323:** Minimum results value corrected and precaution added. **Page 328:** Caution added. **Page 330:** Number of zeros in note 3 corrected. **Page 380:** Note added. **Pages 518 to 567:** Note removed from bottom of table. **Page 569:** Paragraph added before table.
5	August 1998	PLP section added to beginning of manual. **Pages 8, 41:** CompoBus/D added. **Page 16:** Corrected first paragraph is 1-13-1. **Page 18:** Section added. **Pages 21, 25, 26:** Information added on simplified backup function. **Page 23:** Information added on using commercial memory cards. **Pages 24, 25:** Note modified. **Page 38:** Added CompoBus/D Area. **Page 43:** Changed work area addresses. **Page 47:** Note on clock accuracy added. **Pages 63, 64:** Description of A50015 corrected. **Page 147:** Precaution added on long cycle times. **Page 331:** Corrected graphic and description of P on lower left of page. **Page 334:** Corrected graphics at top of page. **Page 405, 408:** Precautions added and "The Slave is not set to 54MH" moved to Error Flag section for READ(190) and WRIT(191). **Pages 412, 415:** Descriptions corrected for D and S, respectively. **Page 418:** Description clarified for item 7. **Page 484:** Corrected equation at bottom of page. **Page 504:** Descriptions for SFC fatal error corrected.