Cat. No. W235-E1-05

SYSMAC
Programmable Controllers
C200HS

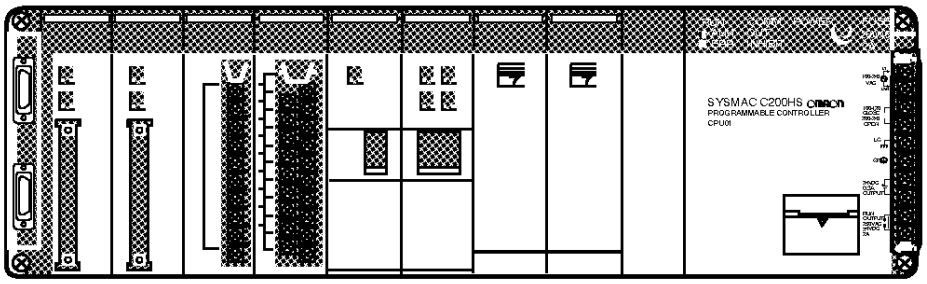
OPERATION MANUAL

OMRON

C200HS Programmable Controllers

Operation Manual

Revised February 2002



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 the operation of the C200HS C-series Programmable Controllers, and it includes the sections described below. Installation information is provided in the *C200HS Programmable Controller Installation Guide*. A table of other manuals that can be used in conjunction with this manual is provided in *Section 1 Introduction*. Provided in *Section 2 Hardware Considerations* is a description of the differences between the older CPUs and the new CPUs described in this manual.

Please read this manual completely and be sure you understand the information provided before attempting to operate the C200HS. **Be sure to read the precautions in the following section.**

Section 1 Introduction explains the background and some of the basic terms 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. Descriptions of Peripheral Devices used with the C200HS PCs and a table of other manuals available to use with this manual for special PC applications are also provided.

Section 2 Hardware Considerations explains basic aspects of the overall PC configuration and describes the indicators that are referred to in other sections of this manual.

Section 3 Memory Areas takes a look at the way memory is divided and allocated and explains the information provided there to aid in programming. It explains how I/O is managed in memory and how bits in memory correspond to specific I/O points. It also provides information on System DM, a special area in C200HS PCs that provides the user with flexible control of PC operating parameters.

Section 4 Writing and Entering Programs explains the basics of ladder-diagram programming, looking at the elements that make up the parts of a ladder-diagram program and explaining how execution of this program is controlled. It also explains how to convert ladder diagrams into mnemonic code so that the programs can be entered using a Programming Console.

Section 5 Instruction Set describes all of the instructions used in programming.

Section 6 Program Execution Timing explains the cycling process used to execute the program and tells how to coordinate inputs and outputs so that they occur at the proper times.

Section 7 Program Debugging and Execution explains the Programming Console procedures used to input and debug the program and to monitor and control operation.

Section 8 Communications provides an overview of the communications features provided by the C200HS.

Section 9 Memory Cassette Operations describes how to manage both UM Area and IOM data via Memory Cassettes. mounted in the CPU.

Section 10 Troubleshooting provides information on error indications and other means of reducing downtime. Information in this section is also useful when debugging programs.

Section 11 Host Link Commands explains the methods and procedures for using host link commands, which can be used for host link communications via the C200HS ports.

The ***Appendices*** provide tables of standard OMRON products available for the C200HS PCs, reference tables of instructions and Programming Console operations, coding sheet to help in programming and parameter input, and other information helpful in PC operation.

⚠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 PC. 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 OMRON PCs. Be sure to read this manual before attempting to use the software 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 abovementioned applications.

3 Safety Precautions

⚠ WARNING Never attempt to disassemble any Units while power is being supplied. Doing so may result in serious electrical shock or electrocution.

⚠ WARNING Never touch any of the terminals while power is being supplied. Doing so may result in serious electrical shock or electrocution.

4 Operating Environment Precautions

Do not operate the control system in the following places.

- Where the PC is exposed to direct sunlight.
- Where the ambient temperature is below 0°C or over 55°C.
- Where the PC may be affected by condensation due to radical temperature changes.
- Where the ambient humidity is below 10% or over 90%.
- Where there is any corrosive or inflammable gas.
- Where there is excessive dust, saline air, or metal powder.
- Where the PC is affected by vibration or shock.
- Where any water, oil, or chemical may splash on the PC.

⚠ 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.

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

- Always ground the system to 100 Ω or less when installing the system to protect against electrical shock.
- Always turn off the power supply to the PC before attempting any of the following. Performing any of the following with the power supply turned on may lead to electrical shock:
  - Mounting or dismounting Power Supply Units, I/O Units, CPU Units, Memory Units, or any other Units.
  - Assembling any devices or racks.
  - Connecting or disconnecting any cables or wiring.

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

- Use the Units only with the power supplies and voltages specified in the operation manuals. Other power supplies and voltages may damage the Units.
- Take measures to stabilize the power supply to conform to the rated supply if it is not stable.
- Provide circuit breakers and other safety measures to provide protection against shorts in external wiring.
- Do not apply voltages exceeding the rated input voltage to Input Units. The Input Units may be destroyed.
- Do not apply voltages exceeding the maximum switching capacity to Output Units. The Output Units may be destroyed.
- Always disconnect the LG terminal when performing withstand voltage tests.
- Install all Units according to instructions in the operation manuals. Improper installation may cause faulty operation.
- Provide proper shielding when installing in the following locations:
  - Locations subject to static electricity or other sources of noise.
  - Locations subject to strong electromagnetic fields.
  - Locations subject to possible exposure to radiation.
  - Locations near to power supply lines.
- Be sure to tighten Backplane screws, terminal screws, and cable connector screws securely.
- Do not attempt to take any Units apart, to repair any Units, or to modify any Units in any way.

⚠ Caution The following precautions are necessary to ensure the general safety of the system. Always heed these precautions.

- Provide double safety mechanisms to handle incorrect signals that can be generated by broken signal lines or momentary power interruptions.
- Provide external interlock circuits, limit circuits, and other safety circuits in addition to any provided within the PC to ensure safety.

6 Conformance to EC Directives

Observe the following precautions when installing the C200HS-CPU01-EC and C200HS-CPU21-EC that conform to the EC Directives.

Provide reinforced insulation or double insulation for the DC power source connected to the DC I/O Unit and for the Power Supply Unit.
Use a separate power source for the DC I/O Unit from the external power supply for the Relay Output Unit.

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. Descriptions of peripheral devices used with the C200HS, a table of other manuals available to use with this manual for special PC applications, and a description of the new features of the C200HS are 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, the appropriate response is made. The response normally involves turning ON 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, switches activating 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, however, are also possible. The problem is how to get the desired control signals from available inputs at appropriate times.

To achieve proper control, the C200HS uses a form of PC logic called ladder-diagram programming. This manual is written to explain ladder-diagram programming and to prepare the reader to program and operate the C200HS.

1-2 The Origins 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 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 *4-4 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 explained here.

PC

Because the C200HS is a Rack PC, there is no one product that is a C200HS 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.

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* list 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 C200HS 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.

High-density I/O Units are designed to provide high-density I/O capability and include Group 2 High-density I/O Units and Special I/O High-density I/O Units.

Special I/O Units are dedicated Units that are designed to meet specific needs. These include some of the High-density I/O Units, Position Control Units, High-speed Counter Units, and Analog I/O Units.

Link Units are used to create Link Systems that link more than one PC or link a single PC to remote I/O points. Link Units include Remote I/O Units, PC Link Units, Host Link Units, SYSMAC NET Link Units, and SYSMAC LINK Units. SYSMAC NET Link and SYSMAC LINK Units can be used with the CPU11 only.

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

1-5 Overview of PC Operation

The following are the basic steps involved in programming and operating a C200HS. 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. This manual is written to explain steps three through six, eight, and nine. The relevant sections of this manual that provide more information are listed with each of these 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 *C200HS 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 or Link Systems, refer to the individual *Operation Manuals* or *System Manuals* for details on I/O bit allocation. (*Section 3 Memory Areas)*
4. Using relay ladder symbols, write a program that represents the sequence of required operations and their inter-relationships. Be sure to also program appropriate responses for all possible emergency situations. (*Section 4 Writing and Inputting the Program, Section 5 Instruction Set, Section 6 Program Execution Timing)*
5. Input the program and all required operating parameters into the PC. (*Section 4-7 Inputting, Modifying, and Checking the Program.*)
6. Debug the program, first to eliminate any syntax errors, and then to find execution errors. (*Section 4-7 Inputting, Modifying, and Checking the Program, Section 7 Program Monitoring and Execution,* and *Section 10 Troubleshooting*)
7. Wire the PC to the controlled system. This step can actually be started as soon as step 3 has been completed. Refer to the *C200HS Installation Guide* and to *Operation Manuals* and *System Manuals* for details on individual Units.
8. Test the program in an actual control situation and carry out fine tuning as required. (*Section 7 Program Monitoring and Execution* and *Section 10 Troubleshooting*)
9. Record two copies of the finished program on masters and store them safely in different locations. (*Section 4-7 Inputting, Modifying, and Checking the Program*)

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. Refer to *3-3 IR Area* for details on I/O capacity and the allocation of I/O bits to I/O points.

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 or Link Systems will need to be considered. In many cases, Special I/O Units, Intelligent I/O 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 Peripheral Devices

The following peripheral devices can be used in programming, either to input/debug/monitor the PC program or to interface the PC to external devices to output the program or memory area data. Model numbers for all devices listed below are provided in *Appendix A Standard Models*. OMRON product names have been placed in bold when introduced in the following descriptions.

Programming Console

A Programming Console is the simplest form of programming device for OMRON PCs. All Programming Consoles are connected directly to the CPU without requiring a separate interface.

Ladder Support Software: LSS

LSS is designed to run on IBM AT/XT compatibles and allows you to perform all the operations of the Programming Console as well as many additional ones. PC programs can be written on-screen in ladder-diagram form as well as in mnemonic form. As the program is written, it is displayed on a display, making confirmation and modification quick and easy. Syntax checks may also be performed on the programs before they are downloaded to the PC.

The LSS is available on either 5” or 3.5” disks.

A computer running the LSS is connected to the C200HS via the Peripheral Port on the CPU using the CQM1-CIF02 cable.

1-7 Available Manuals

The following table lists other manuals that may be required to program and/or operate the C200HS. *Operation Manuals* and/or *Operation Guides* are also provided with individual Units and are required for wiring and other specifications.

Name	Cat. No.	Contents
GPC Operation Manual	W84	Programming procedures for the GPC (Graphics Programming Console)
FIT Operation Manual	W150	Programming procedures for using the FIT (Factory Intelligent Terminal

Name	Cat. No.	Contents
SYSMAC Support Software Operation Manuals	W247/W248	Programming procedures for using the SSS
Data Access Console Operation Guide	W173	Data area monitoring and data modification procedures for the Data Access Console
Printer Interface Unit Operation Guide	W107	Procedures for interfacing a PC to a printer
PROM Writer Operation Guide	W155	Procedures for writing programs to EPROM chips
Floppy Disk Interface Unit Operation Guide	W119	Procedures for interfacing PCs to floppy disk drives
Wired Remote I/O System Manual (SYSMAC BUS)	W120	Information on building a Wired Remote I/O System to enable remote I/O capability
Optical Remote I/O System Manual (SYSMAC BUS)	W136	Information on building an Optical Remote I/O System to enable remote I/O capability
PC Link System Manual	W135	Information on building a PC Link System to automatically transfer data between PCs
Host Link System Manual (SYSMAC WAY)	W143	Information on building a Host Link System to manage PCs from a 'host' computer
SYSMAC NET Link Unit Operation Manual	W114	Information on building a SYSMAC NET Link System and thus create an optical LAN integrating PCs with computers and other peripheral devices
SYSMAC LINK System Manual	W174	Information on building a SYSMAC LINK System to enable automatic data transfer, programming, and programmed data transfer between the PCs in the System
High-speed Counter Unit Operation Manual	W141	Information on High-speed Counter Unit
Position Control Unit Operation Manuals	NC111: W137 NC112: W128 NC211: W166	Information on Position Control Unit
Analog I/O Units Operation Guide	W127	Information on the C200H-AD001, C200H-DA001 Analog I/O Units
Analog Input Unit Operation Manual	W229	Information on the C200H-AD002 Analog Input Unit
Temperature Sensor Unit Operation Guide	W124	Information on Temperature Sensor Unit
ASCII Unit Operation Manual	W165	Information on ASCII Unit
ID Sensor Unit Operation Guide	W153	Information on ID Sensor Unit
Voice Unit Operation Manual	W172	Information on Voice Unit
Fuzzy Logic Unit Operation Manual	W208	Information on Fuzzy Logic Unit
Fuzzy Support Software Operation Manual	W210	Information on the Fuzzy Support Software which supports the Fuzzy Logic Units
Temperature Control Unit Operation Manual	W225	Information on Temperature Control Unit
Heat/Cool Temperature Control Unit Operation Manual	W240	Information on Heating and Cooling Temperature Control Unit
PID Control Unit Operation Manual	W241	Information on PID Control Unit
Cam Positioner Unit Operation Manual	W224	Information on Cam Positioner Unit

1-8 New C200HS Features

The C200HS CPUs (C200HS-CPU01-E, C200HS-CPU03-E, C200HS-CPU21-E, C200HS-CPU23-E, C200HS-CPU31-E, and C200HS-CPU33-E) have a number of new features that the C200H CPUs lacked. The new C200HS features are described briefly in this section. The C200HS-CPU01-E, C200HS-CPU21-E, C200HS-CPU31-E use an AC power supply and the C200HS-CPU03-E, C200HS-CPU23-E, and C200HS-CPU33-E use DC.

In addition, the C200HS-CPU21-E, C200HS-CPU23-E, C200HS-CPU31-E, and C200HS-CPU33-E CPUs have an RS-232C connector. The C200HS-CPU31-E and C200HS-CPU33-E CPUs support the SYSMAC NET Link Unit and SYSMAC LINK Unit.

1-8-1 Improved Memory Capabilities

Internal Memory (UM)

The C200HS CPUs come equipped with 16 KW of RAM in the PC itself, so a very large memory capacity is available without purchasing a separate Memory Unit. Furthermore, the Ladder Program Area has been increased to 15.2 KW.

Memory Cassettes

Two types of Memory Cassettes are available for storage of data such as the program. The PC can be set to transfer data from the Memory Cassette to UM automatically when the PC is turned on.

Model	Specifications
C200HS-ME16K	16-K Word EEPROM
C200HS-MP16K	16-K Word EPROM

Note C200H Memory Cassettes cannot be used in the C200HS.

Clock Function

The C200HS CPUs have a built-in clock. It is not necessary to purchase a Memory Unit equipped with a clock, as it was with the C200H-CPU21-E.

Increased SR Area

In addition to the conventional areas of the C200H, the following areas have been added for the internal auxiliary relays and special auxiliary relays of the C200H. The SR area has been increased substantially to provide more work words and words dedicated to new instructions. The SR area now ranges from SR 236 to SR 299. (The SR area ends at SR 255 in C200H CPUs.) By using additional areas, the user can use Special I/O Units and Remote I/O Units without worrying the empty areas.

Conventional areas	IR Area 1 (without I/O area):	IR 030 to 235
	SR Area 1:	SR 236 to 255
Additional areas	IR Area 2:	IR 300 to 511
	SR Area 2:	SR 256 to 299

The number of operands and instruction execution time will be increased when SR 256 to SR 511 are used in basic instructions.

Increased DM Area

The Read/Write DM area has been increased substantially, too. It now ranges from DM 0000 to DM 6143, compared to DM 0000 to DM 0999 in C200H CPUs.

The 6000 words from DM 0000 to DM 5999 are available for use in the program. (DM 6000 to DM 6143 are used for the History Log and other functions.)

Fixed DM and Expansion DM Areas

The Fixed DM Area, used to store initializing data for Special I/O Units, has been decreased in size. It now contains the 512 words from DM 6144 to DM 6655, compared to 1000 words (DM 1000 to DM 1999) in C200H CPUs.

On the other hand, up to 3000 words of UM can be allocated as expansion DM. Expansion DM is allocated in 1000-word units in DM 7000 to DM 9999.

C200H data stored in words DM 1000 to DM 1999 can be used in C200HS PCs by converting these 1000 words to ROM in the C200HS's DM area (DM 7000 to DM 7999) and then automatically transferring them to DM 1000 to DM 1999 when the C200HS is turned on.

1-8-2 Faster Execution Times

Instruction Execution Time

Basic instructions in the C200HS are executed in ½ of the time required in the C200H. Other instructions are executed in just ¼ of the time.

END Processing Time

The time required for the cycle's overhead processes depend on the system configuration, but these processes are executed in about ¼ of the time required in the C200H.

I/O Refreshing Time

The I/O refreshing time has been reduced for all units, as shown in the following table.

I/O Unit	Time Required for Refreshing
Standard I/O Units	1/3 of the C200H I/O refreshing time
Group-2 High-density I/O Units	1/3 of the C200H I/O refreshing time
Special I/O Units	4/5 of the C200H I/O refreshing time

1-8-3 Larger Instruction Set

Advanced programming is facilitated by the 225 application instructions available with the C200HS-CPU01-E, C200HS-CPU03-E, C200HS-CPU21-E, and C200HS-CPU23-E, or the 229 application instructions available with the C200HS-CPU31-E and C200HS-CPU33-E. In addition, programming has been simplified by the addition of convenient instructions and macro functions. The new instructions and functions are covered in detail in *Section 5 Instruction Set*.

Improved Instructions

Additional functions have been added to the 7 instructions in the following table.

Instruction	Additional Function(s)
DIST(80)	Stack operation. The stack can contain up to 999 words.
COLL(81)	FIFO/LIFO stack operation. The stack can contain up to 999 words.
MLPX(76)	4-to-256 decoder capability.
DMPX(77)	256-to-8 encoder capability.
ADB(50)	Signed binary data can be added.
SBB(51)	Signed binary data can be subtracted.
INT(89)	Can be used to set scheduled interrupts in 1 ms units and control input interrupts.

Expansion Instructions

A group of 47 instructions have been designated as expansion instructions. An expansion instruction does not have a fixed function code; one of the 18 expansion instruction function codes must be assigned to it before it can be used in a program. An instructions tables, which allocates functions codes to expansion instructions, must be transferred to the C200HS before the expansion instructions can be used.

New Instructions

A total of 36 new instructions have been added to the C200HS. These instructions are listed below. (Instructions with (--) for function codes are expansion instructions, which do not have fixed function codes. Some expansion instruction do have default function codes. The SET and RESET instructions are basic instructions, MACRO and TRACE MEMORY SAMPLE instructions are applied instructions, and the other instructions are expansion applied instructions. A default function number is assigned to the TOTALIZING TIMER, TRANSFER BITS, AREA RANGE COMPARE, MACRO, AND TRACE MEMORY SAMPLE instructions.

TRSM(45)	TRACE MEMORY SAMPLE	MBS(--)	SIGNED BINARY MULTIPLY
MCRO(99)	MACRO	DBS(--)	SIGNED BINARY DIVIDE
MAX(--)	FIND MAXIMUM	FCS(--)	FRAME CHECKSUM
MIN(--)	FIND MINIMUM	7SEG(--)	7-SEGMENT DISPLAY OUTPUT
SUM(--)	SUM	RXD(--)	RECEIVE
SRCH(--)	DATA SEARCH	TXD(--)	TRANSMIT
FPD(--)	FAILURE POINT DETECTION	CPS(--)	SIGNED BINARY COMPARE
PID(--)	PID CONTROL	CPSL(--)	SIGNED DOUBLE BINARY COMPARE
HEX(--)	ASCII TO HEX	NEG(--)	2'S COMPLEMENT
XDMR(--)	EXPANSION DM READ	NEGL(--)	DOUBLE 2'S COMPLEMENT
DSW(--)	DIGITAL SWITCH INPUT	ZCPL(--)	DOUBLE AREA RANGE COMPARE
TKY(--)	TEN-KEY INPUT	AVG(--)	AVERAGE VALUE
MTR(--)	MATRIX INPUT	SCL(--)	SCALE
HKY(--)	16-KEY INPUT	SET	SET
ADBL(--)	DOUBLE BINARY ADD	RSET	RESET
SBBL(--)	DOUBLE BINARY SUBTRACT	TTIM(87)	TOTALIZING TIMER
MBSL(--)	DOUBLE SIGNED BINARY MULTIPLY	XFRB(62)	TRANSFER BITS
DBSL(--)	DOUBLE SIGNED BINARY DIVIDE	ZCP(88)	AREA RANGE COMPARE

1-8-4 Wide Selection of Special I/O Units

C200HS Systems can be configured in a variety of ways, using High-density I/O Units, High-speed Counters, Position Control Units, Analog I/O Units, Temperature Sensor Units, ASCII Units, Voice Units, ID Sensor Units, Fuzzy Logic Units, Cam Positioner Units, and so on.

1-8-5 Improved Interrupt Functions

Scheduled Interrupts

The C200HS's scheduled interrupt function has been improved so that the interrupt interval can be set in 1 ms units rather than the 10 ms units in the C200H. When the interrupt mode is set to C200HS mode, the interrupt response time is only 1 ms max. (excluding the input ON/OFF delays). When a Communications Unit is used with the C200HS-CPU31-E/CPU33-E CPU, the interrupt response time is 10 ms max.

Input Interrupts

Up to 8 interrupt subroutines can be executed by inputs to a C200HS-INT01 Interrupt Input Unit mounted to the C200HS. When the interrupt mode is set to C200HS mode, the interrupt response time is only 1 ms max. (excluding the input ON/OFF delays). When a Communications Unit is used with the C200HS-CPU31-E/CPU33-E CPU, the interrupt response time is 10 ms max.

1-8-6 SYSMAC NET Link and SYSMAC LINK Capabilities

The SYSMAC NET Link and SYSMAC LINK Systems are high-speed FA networks which can be used with the C200HS-CPU31-E and C200HS-CPU33-E CPUs and the following Units:

SYSMAC NET Link Unit:	C200HS-SNT32
SYSMAC LINK Units:	C200HS-SLK12 (optical fiber cable) C200HS-SLK22 (coaxial cable)

Data can be exchanged with the PCs in a SYSMAC NET Link or SYSMAC LINK System using the SEND and RECV instructions.

1-8-7 Built-in RS-232C Connector

Host link communications are possible using the RS-232C connector built into the C200HS-CPU21-E/CPU23-E/CPU31-E/CPU33-E CPU. By using the TXD and RXD instructions, RS-232C communications is possible without using time-consuming procedures. A 1-to-1 link using the LR Area or an NT link with the Programmable Terminal (PT) allows high-speed communications.

1-8-8 More Flexible PC Settings

With its default settings, the C200HS can be used like a C200H PC, but the C200HS's new settings provide more flexibility and allow it to be adjusted to fit particular applications. These new settings are described below.

DIP Switch Settings The 6 pins on the C200HS's DIP switch are used to write-protect part of UM, set the CPU to automatically transfer Memory Card data to UM, and other functions.

UM Area Allocation Portions of the UM area can be allocated for use as the Expansion DM Area and I/O Comment Area. (Most of the UM area is used to store the ladder program.)

PC Setup DM 6600 to DM 6655 is set aside for PC Setup data. The PC Setup determines many operating parameters, including the startup mode and initial Special I/O Unit area.

1-8-9 Debugging and Maintenance

Data Trace A data trace function has been added, allowing bit status or word content to be traced in real time.

Differential Monitor The C200HS supports differential monitoring from either the Programming Console or LSS. The operator can detect OFF-to-ON or ON-to-OFF transition in a specified bit.

Error Log Area The C200HS supports all of the C200H-CPU31-E error history area functions and also records the time and date of power interruptions. The C200HS's error log area is DM 6000 to DM 6030 (not DM 0969 to DM 0999 as in the C200H-CPU31-E).

1-8-10 New Programming Console Operations

The following Programming Console operations are supported by the C200HS in addition to those supported by the C200H.

- Constants can be input in decimal form.
- Monitor displays can be switched between hexadecimal and normal or long decimal form.
- OFF to ON and ON to OFF transitions in bit status can be monitored (differential monitoring).
- Function codes can be allocated to expansion instructions and current function code allocations can be read.
- UM area allocations can be set.
- The clock in the C200HS can be read and set.
- In addition to the TERMINAL mode supported in the C200H, the C200HS has an EXTENDED TERMINAL mode in which all of the Programming Console's keys can be used to the status of Key Bits.
- The memory clear operation has been separated into an operation to clear the user program excluding I/O comments and UM area allocation information, and one to clear the user program, I/O comments , and UM area allocation information.

1-8-11 Peripheral Devices

Peripheral Device Connection With the C200H a Peripheral Device had to be connected through a Peripheral Interface Unit or Host Link Unit, but with the C200HS Peripheral Devices can be connected to the PC through a CQM1-CIF02 Connecting Cable.

I/O Comments Stored in PC By allocating a part of UM as the I/O Comment area, it is no longer necessary to read I/O Comments from a Peripheral Device's floppy disk. If the Peripheral Device is connected to the C200HS online, the ladder diagram can be viewed with I/O comments.

Online Editing A "CYCLE TIME OVER" error will no longer be generated when the program in the PC itself is being edited online.

1-8-12 Using C200H Programs

Programs developed for the C200H can be very easily transferred for use in the C200HS. This section provides the steps necessary to achieve this. Two procedures are provided: one for transferring using only internal CPU memory and one for transferring via Memory Cassettes.

Detailed procedures for the individual steps involved in transferring programs can be found in the Version-3 LSS Operation Manuals. You will also require a CQM1-CIF02 Connecting Cable to connect the computer running LSS to the C200HS.

Precautions Observe the following precautions when transferring C200H programs to the C200HS.

- If a C200H program including the SET SYSTEM instruction (SYS(49)) is transferred to the C200HS, the operating parameters set by this instruction will be transferred to the C200HS's PC Setup area (DM 6600, DM 6601, and DM 6655) and overwrite any current settings. Be sure to confirm that the settings in these words are correct before using the C200HS after program transfer.
- If the C200H program accesses the C200H's error log in DM 0969 to DM 0999, the addresses of the words being accessed must be changed to DM 6000 to DM 6030, which is the error log area for the C200HS.
- Any programs that rely on the execution cycle time (i.e., on the time require to execute any one part of all of the program) must be adjusted when used on the C200HS, which provides a much faster cycle time.

Using Internal Memory The following procedure outlines the steps to transfer C200H programs to the user memory inside the C200HS.

1, 2, 3...

1. Transfer the program and any other required data to the LSS work area. This data can be transferred from a C200H CPU, from floppy disk, or from a C200HS Memory Unit.

   To transfer from a C200H CPU, set the PC for the LSS to the C200H, connect the LSS to the C200H, go online, and transfer the program and any other require data to the LSS work area. You will probably want to transfer DM data and the I/O table, if you have created an I/O table for the C200H.

or To transfer from floppy disk, set the PC for the LSS to the C200H in the offline mode and load the program and any other require data to the LSS work area. You will probably want to load DM data and the I/O table, if you have created an I/O table for the C200H.

or To transfer from a C200H-MP831, set the PC for the LSS to the C200H in the offline mode and read data from the Memory Unit into the LSS work area.

2. Go offline if the LSS is not already offline.
3. Change the PC setting for the LSS to the C200HS.
4. If you want to transfer I/O comments together with the program to the C200HS, allocate UM area for I/O comments.
5. Connect the LSS to the C200HS and go online.
6. Make sure that pin 1 on the C200HS's CPU is OFF to enable writing to the UM area.

7. Transfer the program and and any other require data to the C200HS. You will probably want to transfer DM data and the I/O table, if you have created an I/O table for the C200H.
8. Turn the C200HS off and then back on to reset it.
9. Test program execution before attempting actual operation.

Using Memory Cassettes

The following procedure outlines the steps to transfer C200H programs to the C200HS via EEPROM or EPROM Memory Cassettes. This will allow you to read the program data from the Memory Cassette automatically at C200HS startup. The first four steps of this procedure is the same as those used for transferring directly to the C200HS's internal memory (UM area).

1, 2, 3...

1. Transfer the program and any other required data to the LSS work area. This data can be transferred from a C200H CPU, from floppy disk, or from a C200HS Memory Unit.

   To transfer from a C200H CPU, set the PC for the LSS to the C200H, connect the LSS to the C200H, go online, and transfer the program and any other require data to the LSS work area. You will probably want to transfer DM data and the I/O table, if you have created an I/O table for the C200H.

or To transfer from floppy disk, set the PC for the LSS to the C200H in the offline mode and load the program and any other require data to the LSS work area. You will probably want to load DM data and the I/O table, if you have created an I/O table for the C200H.

or To transfer from a C200H-MP831, set the PC for the LSS to the C200H in the offline mode and read data from the Memory Unit into the LSS work area.

2. Go offline if the LSS is not already offline.
3. Change the PC setting for the LSS to the C200HS.
4. If you want to transfer I/O comments together with the program to the C200HS, allocate UM area for I/O comments.
5. Allocate expansion DM words DM 7000 to DM 7999 in the UM area using the UM allocation operation from the LSS.
6. Copy DM 1000 through DM 1999 to DM 7000 through DM 7999.
7. Write "0100" to DM 6602 to automatically transfer the contents of DM 7000 through DM 7999 to DM 1000 through DM 1999 at startup.
8. To transfer to an EEPROM Memory Cassette, use the following procedure.
   a) Connect the LSS to the C200HS and go online.
   b) Make sure that pin 1 on the C200HS's CPU is OFF to enable writing to the UM area.
   c) Transfer the program and any other require data to the C200HS. You will probably want to transfer DM data and the I/O table, if you have created an I/O table for the C200H. Make sure you specify transfer of the Expansion DM Area and, if desired, the I/O Comment Area.
   d) Turn ON SR 27000 from the LSS to transfer UM data to the Memory Cassette and continue from step 9.

or To transfer to an EPROM Memory Cassette, use the following procedure.
   a) Connect an PROM Writer to the LSS and write the data to the EPROM chip using the LSS EPROM writing operation.
   e) Set the ROM type selector on the Memory Cassette to the correct capacity.
   f) Mount the ROM chip to the Memory Cassette.
   g) Mount a EPROM Memory Cassette to the C200HS.
9. Turn ON pin 2 on the C200HS's DIP switch to enable automatic transfer of Memory Cassette data to the CPU at startup.

10. Turn the C200HS off and then back on to reset it and transfer data from the Memory Cassette to the CPU.
11. Test program execution before attempting actual operation.

SECTION 2
Hardware Considerations

This section provides information on hardware aspects of the C200HS that are relevant to programming and software operation. These include CPU Components, basic PC configuration, CPU capabilities, and Memory Cassettes. This information is covered in detail in the *C200HS Installation Guide.*

2-1 CPU Components

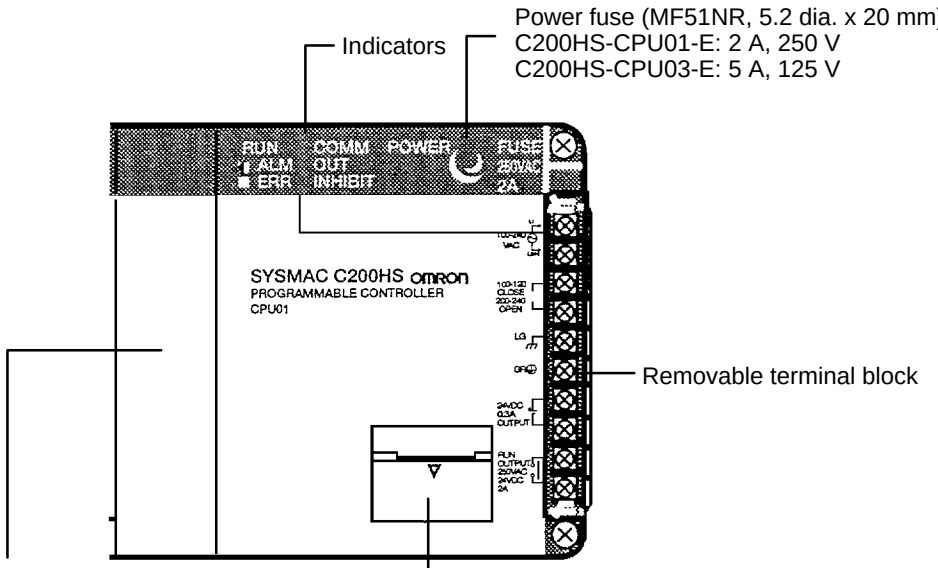
There are two groups of CPUs available, one that uses an AC power supply, and one that uses a DC power supply. Select one of the models shown below according to requirements of your control system.

CPU model	Power supply voltage
C200HS-CPU01-E/CPU21-E/CPU31-E	100 to 120 VAC or 200 to 240 VAC (voltage selector)
C200HS-CPU03-E/CPU23-E/CPU33-E	24 VDC

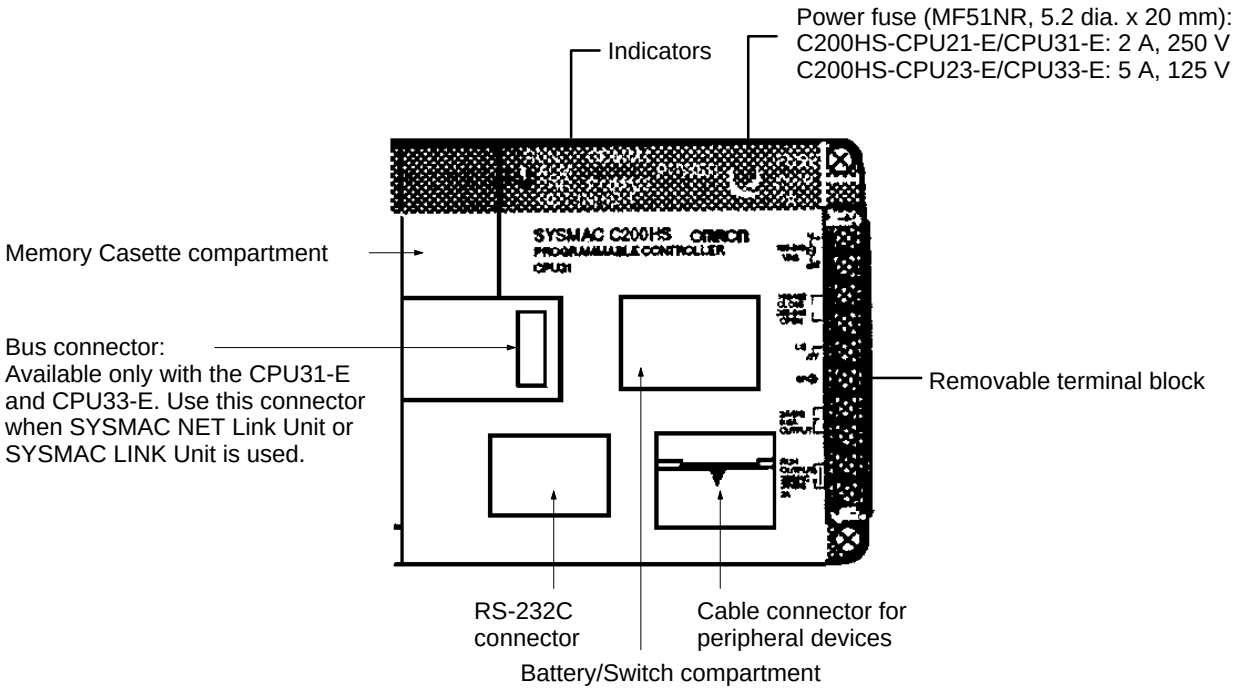
The CPU21-E, CPU23-E, CPU31-E, and CPU33-E CPUs have an RS-232C connector. The CPU31-E and CPU33-E CPUs support the SYSMAC NET Link Unit and SYSMAC LINK Unit.

⚠ **Caution** Be sure to check the power supply used by the CPU. Absolutely do not provide an AC power supply to a DC-type CPU.

The following diagram shows the main CPU components.



C200HS-CPU21-E/CPU23-E/CPU31-E/CPU33-E

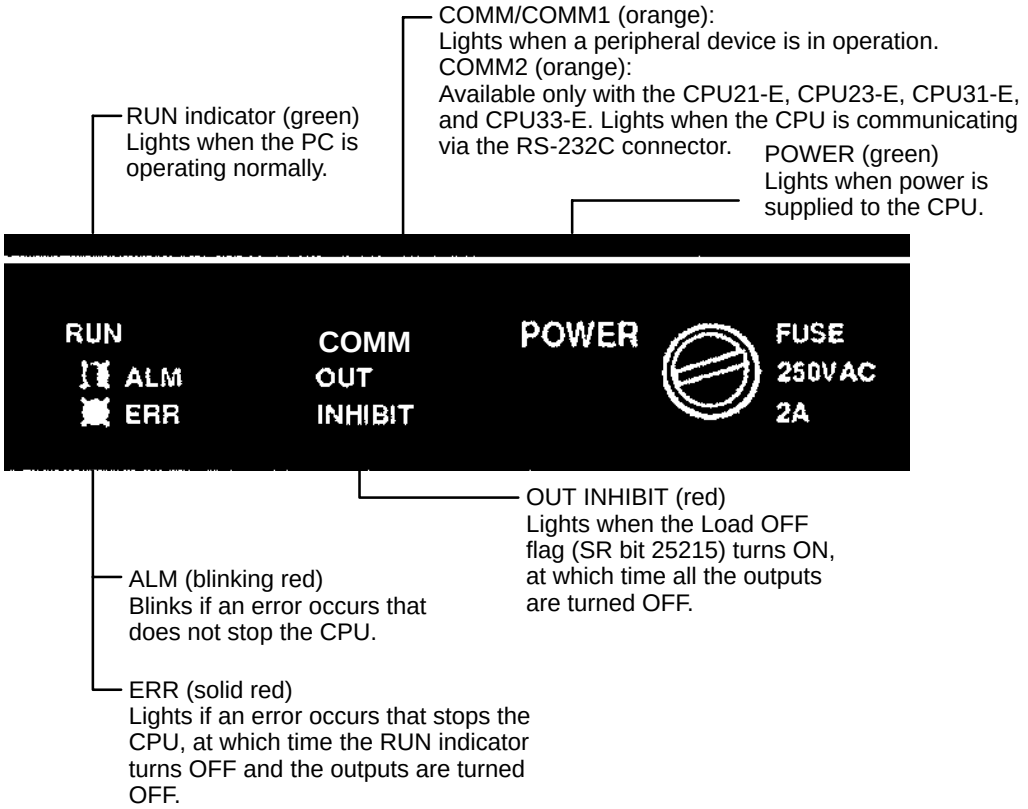


2-1-1 CPU Indicators

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

CPU indicators are shown and described below. (CPU01-E/03-E shown below.)

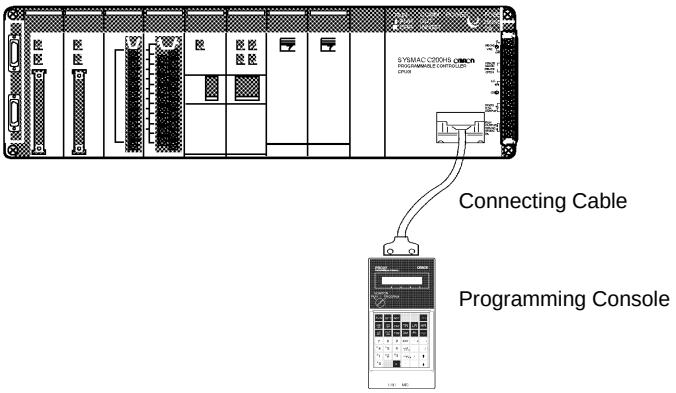


2-1-2 Peripheral Device Connection

A Programming Console or IBM PC/AT running LSS can be used to program and monitor the C200HS PCs.

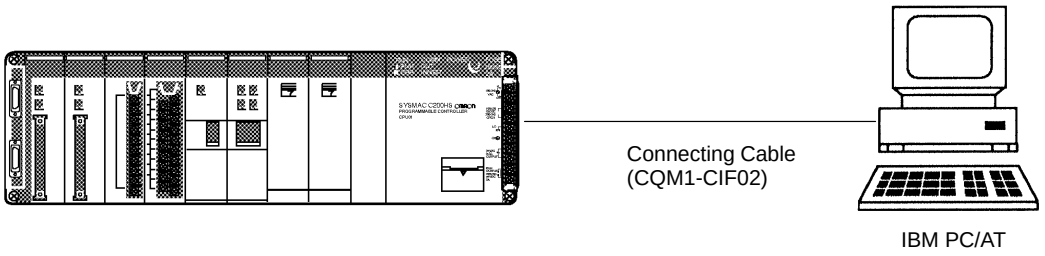
Programming Console

A C200H-PR027-E or CQM1-PRO01-E Programming Console can be connected as shown in the following diagram. The C200H-PR027-E is connected via the C200H-CN222 or C200H-CN422 Programming Console Connecting Cable, which must be purchased separately. A Connecting Cable is provided with the CQM1-PRO01-E.



IBM PC/AT with LSS

An IBM PC/AT or compatible computer can be connected as shown in the following diagram. The LSS is available on either 3.5" disks (C500-SF312-EV3) or 5.25" disks (C500-SF711-EV3). Only version 3 or later of the LSS supports C200HS functionality.



2-2 PC Configuration

The basic PC configuration consists of two types of Rack: a CPU Rack and Expansion I/O Racks. The Expansion I/O Racks are not a required part of the basic system. They are used to increase the number of I/O points. An illustration of these Racks is provided in *3-3 IR Area.* A third type of Rack, called a Slave Rack, can be used when the PC is provided with a Remote I/O System.

CPU Racks

A C200HS CPU Rack consists of three components: (1) The CPU Backplane, to which the CPU 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.

A C200HS CPU Rack can be used alone or it can be connected to other Racks to provide additional I/O points. The CPU Rack provides three, five, eight, or ten slots to which these other Units can be mounted depending on the backplane used.

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 can be mounted. It is built onto an Expansion I/O Backplane to which a Power Supply and up to ten other Units are mounted.

An Expansion I/O Rack is always connected to the CPU via the connectors on the Backplanes, allowing communication between the two Racks. Up to two Expansion I/O Racks can be connected in series to the CPU Rack.

Unit Mounting Position

Only I/O Units and Special I/O Units can be mounted to Slave Racks. All I/O Units, Special I/O Units, Group-2 High-density I/O Units, Remote I/O Master Units, PC and Host Link Units, can be mounted to any slot on all other Racks. Interrupt Input Units must be mounted to C200H-BC□□1-V2 Backplanes.

Refer to the *C200HS Installation Guide* for details about which slots can be used for which Units and other details about PC configuration. The way in which I/O points on Units are allocated in memory is described in *3-3 IR Area*.

2-3 CPU Capabilities

The following tables compare the capabilities of C200H and C200HS CPUs.

C200H

Function	C200H		
	CPU21-E	CPU23-E	CPU31-E
Built-in clock/calendar	No (see note)	No (see note)	Yes
Error log	No	No	Yes
Data Trace	No	No	No
Differential Monitor	No	No	No
Expansion DM	No	No	No
General-use DM	1000 words		970 words
Ladder Program capacity	6974 words (in Memory Unit)		
SR Area	SR 236 to SR 255		
New instructions: (See *1-8-3 Larger Instruction Set* for a list of the 36 new instructions.)	No	No	No
Network Instructions: NETWORK SEND - SEND(90) NETWORK RECEIVE - RECV(98)	No	No	Yes
Power Supply	AC	DC	AC

Note The C200H-CPU21-E/CPU23-E can use the C200H-MR433/MR833/ME432/ME832 Memory Units' clock.

C200HS

Function	C200HS					
	CPU01-E	CPU21-E	CPU31-E	CPU03-E	CPU23-E	CPU33-E
Built-in clock/calendar	Yes					
Error log	Yes[1]					
Data Trace	Yes					
Differential Monitor	Yes					
Expansion DM	3K words max.[2]					
General-use DM	6K words					
Ladder Program capacity	15.2K words max[2]					
SR Area	SR 236 to SR 255 and SR 256 to SR 299					
New instructions: (See *1-8-3 Larger Instruction Set* for a list of the 36 new instructions.)	Yes					
Network Instructions: NETWORK SEND - SEND(90) NETWORK RECEIVE - RECV(98)	No	No	Yes	No	No	Yes
Power Supply	AC			DC		

Note 1. The C200HS CPUs record the time and date of power interruptions.

2. Part of the 16K-word UM can be allocated to Expansion DM and I/O comments.

2-4 Memory Cassettes

The C200HS comes equipped with a built-in RAM for the user's program, so a normal program be created even without installing a Memory Cassette. An optional Memory Cassette, however, can be used. There are two types of Memory Cassette available, each with a capacity of 16K words. For instructions on installing Memory Cassettes, refer to *2-5 Installing Memory Cassettes*.

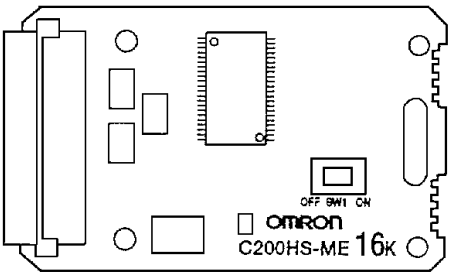
The following table shows the Memory Cassettes which can be used with the C200HS PCs. These Memory Cassettes cannot be used in C200H PCs.

Memory	Capacity	Model number	Comments
EEPROM	16K words	C200HS-ME16K	---
EPROM	16K words	C200HS-MP16K	The EPROM chip is not included with the Memory Cassette; it must be purchased separately.

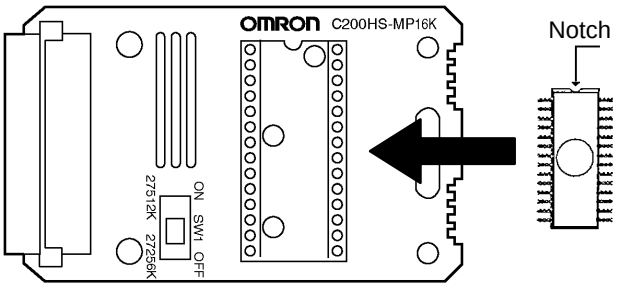
Note Memory Cassettes for the C200HS cannot be used with the C200H, and Memory Units for the C200H cannot be used with the C200HS.

C200HS-ME□16K (EEPROM)

When a Memory Cassette is installed in the CPU, reading and writing of the user memory (UM) and I/O data is made possible. There is no need for a backup power supply. The Memory Cassette can be removed from the CPU and used for storing data.



C200HS-MP□16K (EPROM) The program is written using a PROM Writer. The ROM is mounted to the Memory Casette and then installed in the CPU. I/O data cannot be stored.



2-5 Installing Memory Cassettes

An optional Memory Cassette can be installed in the C200HS. (The C200H Memory Unit cannot be used with the C200HS.) The two types of Memory Cassettes are described in *2-4 Memory Cassettes*. To install a Memory Cassette, follow the procedure outlined below.

⚠ **Caution** Be careful to always turn the power off before inserting or removing a Memory Cassette. If a Memory Cassette is inserted into or removed from the CPU with the power on, it may cause the CPU to malfunction or cause damage to the memory.

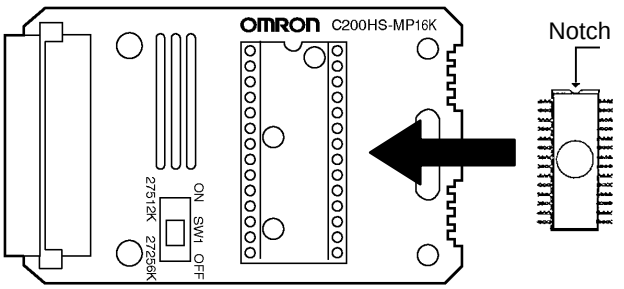
1, 2, 3... 1. Set the DIP switch. For an EEPROM Memory Cassette, set pin no. 1 (write protect) to either ON or OFF. Setting it to ON will protect the program in the memory from being overwritten. Setting it to OFF will allow the program to be overwritten. (The factory setting is OFF.)

For an EPROM Memory Cassette, set pin no. 1 (ROM Type Selector) according to the type of ROM that is to be mounted.

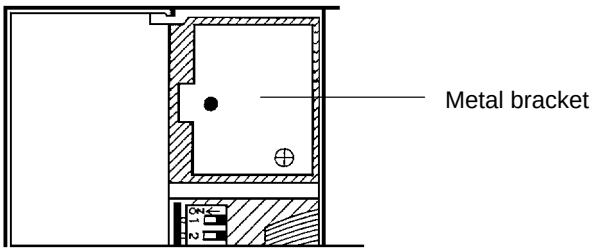
Pin no. 1	ROM type	Model	Capacity	Access speed
OFF	27256	ROM-JD-B	16K words	150 ns
ON	27512	ROM-KD-B	32K words*	150 ns

Note *Only 16K words accessible.

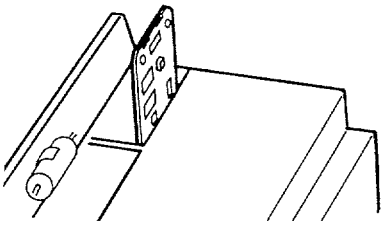
2. Write to EPROM (if using an EPROM Memory Cassette). Using a PROM Writer, write the program to EPROM. Then mount the EPROM chip to the Memory Cassette, with the notched end facing upwards as shown in the illustration below.



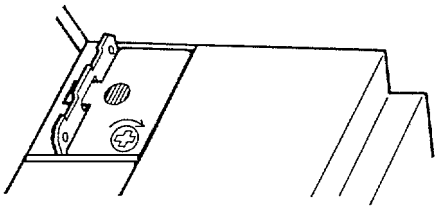
3. Remove the bracket from the Memory Cassette, as shown in the illustration below.



4. Check that the connector side goes in first and that the Cassette's circuit components face right and then insert the Cassette into the CPU. The Cassette slides in along a track in the CPU.



5. Replace the Memory Cassette bracket over the Cassette and tighten the screw that holds the bracket.



2-6 CPU DIP Switch

The DIP switch on C200HS CPUs is located between the Memory Cassette compartment and battery.

The 6 pins on the DIP switch control 6 of the CPU's operating parameters.

Pin no.	Item	Setting	Function
1	Memory protect	ON	Program Memory and read-only DM (DM 6144 to DM 6655) data cannot be overwritten from a Peripheral Device.
		OFF	Program Memory and read-only DM (DM 6144 to DM 6655) data can be overwritten from a Peripheral Device.
2	Automatic transfer of Memory Cassette contents	ON	The contents of the Memory Cassette will be automatically transferred to the internal RAM at start-up.
		OFF	The contents will not be automatically transferred.
3	Message language	ON	Programming Console messages will be displayed in English.
		OFF	Programming Console messages will be displayed in the language stored in system ROM. (Messages will be displayed in Japanese with the Japanese version of system ROM.)
4	Expansion instruction setting	ON	Expansion instructions set by user. Normally ON when using a host computer for programming/monitoring.
		OFF	Expansion instructions set to defaults.
5	Communications parameters	ON	Standard communications parameters (see note) will be set for the following serial communications ports. • Built-in RS-232C port • Peripheral port (only when a CQM1-CIF01/-CIF02 Cable is connected. Does not apply to Programming Console.) **Note** 1. Standard communications parameters are as follows: Serial communications mode: Host Link or peripheral bus; start bits: 1; data length: 7 bits; parity: even; stop bits: 2; baud rate: 9,600 bps 2. The CX-Programmer running on a personal computer can be connected to the peripheral port via the peripheral bus using the above standard communications parameters.
		OFF	The communications parameters for the following serial communications ports will be set in PC Setup as follows: • Built-in RS-232C port: DM 6645 and DM 6646 • Peripheral port: DM 6650 and DM 6651 **Note** When the CX-Programmer is connected to the peripheral port with the peripheral bus, either set bits 00 to 03 of DM 6650 to 0 Hex (for standard parameters), or set bits 12 to 15 of DM 6650 to 0 Hex and bits 00 to 03 of DM 6650 to 1 Hex (for Host Link or peripheral bus) separately.
6	Expansion TERMINAL mode setting when AR 0709 is ON	ON	Expansion TERMINAL mode; AR 0712 ON.
		OFF	Normal mode; AR 0712: OFF

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.

SECTION 3
Memory Areas

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**. The other memory area is the UM Area, where the user's program is actually stored. This section describes these areas individually and provides information that will be necessary to use them. As a matter of convention, the TR area is described in this section, even though it is not strictly a memory area.

3-1 Introduction

Details, including the name, size, and range of each area are summarized in the following table. Data and memory areas are normally referred to by their acronyms, e.g., the IR Area, the SR Area, etc.

Area	Size	Range	Comments
I/O Area	480 bits	IR 000 to IR 029	I/O words are allocated to the CPU Rack and Expansion I/O Racks by slot position.[1]
Group-2 High-density I/O Unit and B7A Interface Unit Area	320 bits	IR 030 to IR 049	Allocated to Group-2 High-density I/O Units and to Group-2 B7A Interface Units 0 to 9[1]
SYSMAC BUS Area	800 bits	IR 050 to IR 099	Allocated to Remote I/O Slave Racks 0 to 4.[1]
Special I/O Unit Area	1,600 bits	IR 100 to IR 199	Allocated to Special I/O Units 0 to 9.[1]
Optical I/O Unit and I/O Terminal Area	512 bits	IR 200 to IR 231	Allocated to Optical I/O Units and I/O Terminals.[1]
Work Area 1	64 bits	IR 232 to IR 235	For use as work bits in the program.
Special Relay Area 1	312 bits	SR 23600 to SR 25507	Contains system clocks, flags, control bits, and status information.
Special Relay Area 2	704 bits	SR 256 to SR 299 (298 to 299 reserved by system)	Contains flags, control bits, and status information.
Macro Area	64 bits	SR 290 to SR 293	Inputs
	64 bits	SR 294 to SR 297	Outputs
Work Area 2	3,392 bits	IR 300 to IR 511	For use as work bits in the program.
Temporary Relay Area	8 bits	TR 00 to TR 07	Used to temporarily store and retrieve execution conditions when programming certain types of branching ladder diagrams.
Holding Relay Area	1,600 bits	HR 00 to HR 99	Used to store data and to retain the data values when the power to the PC is turned off.
Auxiliary Relay Area	448 buts	AR 00 to AR 27	Contains flags and bits for special functions. Retains status during power failure.
Link Relay Area	1,024 bits	LR 00 to LR 63	Used for data links in the PC Link System.[1]
Timer/Counter Area	512 counters/ timers	TC 000 to TC 511	Used to define timers and counters, and to access completion flags, PV, and SV. Interval timers 0 through 2 and high-speed counters 0 through 2 provided in separate area. TIM 000 through TIM 015 can be refreshed via interrupt processing as high-speed timers.
Data Memory Area	6,144 words	DM 0000 to DM 6143	Read/Write
	1,000 words	DM 0000 to DM 0999	Normal DM.
	1,000 words	DM 1000 to DM 1999	Special I/O Unit Area[2]
	4,000 words	DM 2000 to DM 5999	Normal DM.
	31 words	DM 6000 to DM 6030	History Log
	(44 words)	DM 6100 to DM 6143	Link test area (reserved)
Fixed DM Area	512 words	DM 6144 to DM 6599	Fixed DM Area (read only)
	56 words	DM 6600 to DM 6655	PC Setup
Expansion DM Area	3,000 words max.	DM 7000 to DM 9999	Read only

Note 1. These can be used as work words and bits when not used for their allocated purposes.

2. The PC Setup can be set to use DM 7000 through DM 7999 as the Special I/O Area.

Work Bits and Words

When some bits and words in certain data areas are not being used for their intended purpose, they can be used in programming as required to control other bits. Words and bits available for use in this fashion are called work words and work bits. Most, but not all, unused bits can be used as work bits. Those that can be used are described area-by-area in the remainder of this section. Actual application of work bits and work words is described in *Section 4 Writing and Inputting the Program*.

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 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

When designating a data area, the acronym for the area is always required for any but the IR and SR areas. Although the acronyms for the IR and SR areas are often given for clarity in text explanations, they are not required, and not entered, when programming. Any data area designation without an acronym is assumed to be in either the IR or SR area. Because IR and SR addresses run consecutively, the word or bit addresses are sufficient to differentiate these two areas.

An actual data location within any data area but the TC area is designated by its address. The address designates the bit or word within the area where the desired data is located. The TC area consists of TC numbers, each of which is used for a specific timer or counter defined in the program. Refer to *3-8 TC Area* for more details on TC numbers and to *5-14 Timer and Counter Instructions* for information on their application.

The rest of the data areas (i.e., the IR, SR, HR, DM, AR, and LR areas) consist of words, each of which consists of 16 bits numbered 00 through 15 from right to left. IR words 000 and 001 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. These terms are not used in this manual because a single data word is often split into two or more parts, with each part used for different parameters or operands. When this is done, the rightmost bits of a word may actually become the most significant bits, i.e., the leftmost bits in another word,when combined with other bits to form a new word.

Bit number	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
IR word 000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IR word 001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DM area is accessible by word only; you cannot designate an individual bit within a DM word. Data in the IR, SR, HR, AR, and LR 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- or five-digit address. The following table show examples of this. The two rightmost digits of a bit designation must indicate a bit between 00 and 15, i.e., the rightmost digit must be 5 or less the next digit to the left, either 0 or 1.

The same TC number can be used to designate either the present value (PV) of the timer or counter, or a bit that functions as the Completion Flag for the timer or counter. This is explained in more detail in *3-8 TC Area*.

Area	Word designation	Bit designation
IR	000	00015 (leftmost bit in word 000)
SR	252	25200 (rightmost bit in word 252)
DM	DM 1250	Not possible
TC	TC 215 (designates PV)	TC 215 (designates completion flag)
LR	LR 12	LR 1200

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 represents 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 into data areas, it must be input in the proper form for the intended purpose. This is no problem when designating individual bits, which are merely 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 x 5 + 16^2 x 15 + 16 x 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 of 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 x 5 + 16^2 x 7 + 16 x 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 then 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 are a equivalent to the hexadecimal digit C.

There are instructions provided to convert data either direction between BCD and hexadecimal. Refer to *5-18 Data Conversion* for details. Tables of binary equivalents to hexadecimal and BCD digits are provided in the appendices for reference.

Decimal Points

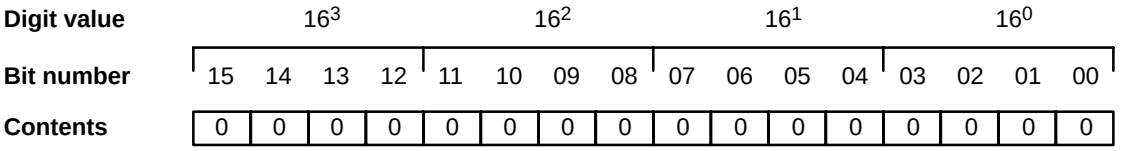
Decimal points are used in timers only. The least significant digit represents tenths of a second. All arithmetic instructions operate on integers only.

Signed and Unsigned Binary Data

This section explains signed and unsigned binary data formats. Many instructions can use either signed or unsigned data and a few (CPS(—), CPSL(—), DBS(—), DBSL(—), MBS(—), and MBSL(—)) use signed data exclusively.

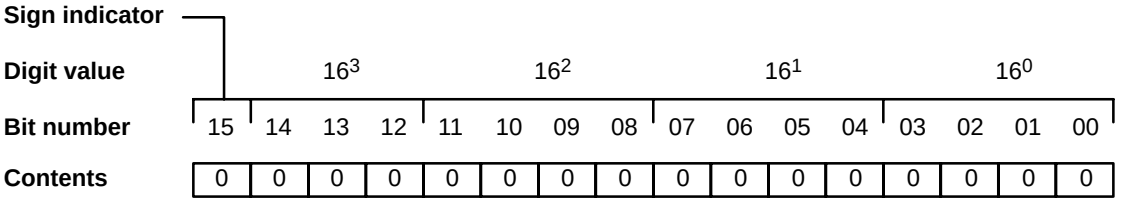
Unsigned binary

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).

The following table shows the corresponding decimal, 16-bit hexadecimal, and 32-bit hexadecimal values.

Decimal	16-bit Hex	32-bit Hex
2147483647	——	7FFFFFFF
2147483646	——	7FFFFFFE
.	.	.
.	.	.
.	.	.
32768	——	00008000
32767	7FFF	00007FFF
32766	7FFE	00007FFE
.	.	.
.	.	.
.	.	.
2	0002	00000002
1	0001	00000001
0	0000	00000000
–1	FFFF	FFFFFFFF
–2	FFFE	FFFFFFFE
.	.	.
.	.	.
.	.	.
–32767	8001	FFFF8001
–32768	8000	FFFF8000
–32769	——	FFFF7FFF
.	.	.
.	.	.
.	.	.
–2147483647	——	80000001
–2147483648	——	80000000

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 IR (Internal Relay) Area

The IR area is used both as data to control I/O points, and as work bits to manipulate and store data internally. It is accessible both by bit and by word. In the C200HS PC, the IR area is comprised of words 000 to 235 and 298 to 511.

Words in the IR area that are used to control I/O points are called I/O words. Bits in I/O words are called I/O bits. Bits in the IR area which are not assigned as I/O bits can be used as work bits. IR area work bits are reset when power is interrupted or PC operation is stopped.

I/O Words

If a Unit brings inputs into the PC, the bit assigned to it is an input bit; if the Unit sends an output from the PC, the bit is an output bit. To turn on an output, the output bit assigned to it must be turned ON. When an input turns on, the input bit assigned to it also turns ON. These facts can be used in the program to access input status and control output status through I/O bits.

Input Bit Usage

Input bits can be used to directly input external signals 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 in instructions that control bit status, e.g., the OUTPUT, DIFFERENTIATION UP, and KEEP instructions.

Output Bit Usage

Output bits are used to output program execution results and can be used in any order in programming. Because outputs are refreshed only once during each cycle (i.e., once each time the program is executed), any output bit can 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.

See *5-15-1 Shift Register – SFT(10)* for an example that uses an output bit in two 'bit-control' instructions.

Word Allocation for Racks

I/O words are allocated to the CPU Rack and Expansion I/O Racks by slot position. One I/O word is allocated to each slot, as shown in the following table. Since each slot is allocated only one I/O word, a 3-slot rack uses only the first 3 words, a 5-slot rack uses only the first 5 words, and an 8-slot rack uses only the first 8 words. Words that are allocated to unused or nonexistent slots are available as work words.

← Left side of rack | Right side of a 10-slot rack →

Rack	Slot 1	Slot 2	Slot 3	Slot 4	Slot 5	Slot 6	Slot 7	Slot 8	Slot 9	Slot 10
CPU	IR 000	IR 001	IR 002	IR 003	IR 004	IR 005	IR 006	IR 007	IR 008	IR 009
1st Expansion	IR 010	IR 011	IR 012	IR 013	IR 014	IR 015	IR 016	IR 017	IR 018	IR 019
2nd Expansion	IR 020	IR 021	IR 022	IR 023	IR 024	IR 025	IR 026	IR 027	IR 028	IR 029

Unused Words

Any words allocated to a Unit that does not use them can be used in programming as work words and bits. Units that do not used the words assigned to the slot they are mounted to include Link Units (e.g., Host Link Units, PC Link Units, SYSMAC NET Link Units, etc.), Remote I/O Master Units, Special I/O Units, Group-2 High-density I/O Units, Group-2 B7A Interface Units, and Auxiliary Power Supply Units.

Allocation for Special I/O Units and Slave Racks

Up to ten Special I/O Units may be mounted in any slot of the CPU Rack or Expansion I/O Racks. Up to five Slave Racks may be used, whether one or two Masters are used. IR area words are allocated to Special I/O Units and Slave Racks by the unit number on the Unit, as shown in the following tables.

Special I/O Units

Unit number	IR address
0	100 to 109
1	110 to 119
2	120 to 129
3	130 to 139
4	140 to 149
5	150 to 159
6	160 to 169
7	170 to 179
8	180 to 189
9	190 to 199

Slave Racks

Unit number	IR address
0	050 to 059
1	060 to 069
2	070 to 079
3	080 to 089
4	090 to 099

The C500-RT001/002-(P)V1 Remote I/O Slave Rack may be used, but it requires 20 I/O words, not 10, and therefore occupies the I/O words allocated to 2 C200H Slave Racks, both the words allocated to the unit number set on the rack and the words allocated to the following unit number. When using a C200HS CPU, do not set the unit number on a C500 Slave Rack to 4, because there is no unit number 5. I/O words are allocated only to installed Units, from left to right, and not to slots as in the C200HS system.

Allocation for Optical I/O Units and I/O Terminals

I/O words between IR 200 and IR 231 are allocated to Optical I/O Units and I/O Terminals by unit number. The I/O word allocated to each Unit is IR 200+n, where n is the unit number set on the Unit.

Allocation for Remote I/O Master and Link Units

Remote Master I/O Units and Host Link Units do not use I/O words, and the PC Link Units use the LR area, so words allocated to the slots in which these Units are mounted are available as work words.

Bit Allocation for I/O Units

An I/O Unit may require anywhere from 8 to 16 bits, depending on the model. With most I/O Units, any bits not used for input or output are available as work bits. Transistor Output Units C200H-OD213 and C200H-OD411, as well as Triac Output Unit C200H-OA221, however, uses bit 08 for the Blown Fuse Flag. Transistor Output Unit C200H-OD214 uses bits 08 to 11 for the Alarm Flag. Bits 08 to 15 of any word allocated to these Units, therefore, cannot be used as work bits.

Bit Allocation for Interrupt Input Units

The Interrupt Input Unit uses the 8 bits of the first I/O word allocated to its slot in the CPU Rack. (An Interrupt Input Unit will operate as a normal Input Unit when installed in an Expansion I/O Rack.) The other 24 bits allocated to its slot in the CPU Rack can be used as work bits.

Allocation for Group-2 High-density I/O Units and B7 Interface Units

Group-2 High-density I/O Units and B7A Interface Units are allocated words between IR 030 and IR 049 according to I/O number settings made on them and do not use the words allocated to the slots in which they are mounted. For 32-point Units, each Unit is allocated two words; for 64-point Units, each Unit is allocated four words. The words allocated for each I/O number are in the following tables. Any words or part of words not used for I/O can be used as work words or bits in programming.

32-point Units		64-point Units	
I/O number	**Words**	**I/O number**	**Words**
0	IR 30 to IR 31	0	IR 30 to IR 33
1	IR 32 to IR 33	1	IR 32 to IR 35
2	IR 34 to IR 35	2	IR 34 to IR 37
3	IR 36 to IR 37	3	IR 36 to IR 39
4	IR 38 to IR 39	4	IR 38 to IR 41
5	IR 40 to IR 41	5	IR 40 to IR 43
6	IR 42 to IR 43	6	IR 42 to IR 45
7	IR 44 to IR 45	7	IR 44 to IR 47
8	IR 46 to IR 47	8	IR 46 to IR 49
9	IR 48 to IR 49	9	Cannot be used.

When setting I/O numbers on the High-density I/O Units and B7A Interface Units, be sure that the settings will not cause the same words to be allocated to more than one Unit. For example, if I/O number 0 is allocated to a 64-point Unit, I/O number 1 cannot be used for any Unit in the system.

Group-2 High-density I/O Units and B7A Interface Units are not considered Special I/O Units and do not affect the limit to the number of Special I/O Units allowed in the System, regardless of the number used.

The words allocated to Group-2 High-density I/O Units correspond to the connectors on the Units as shown in the following table.

Unit	Word	Connector/row
32-point Units	First	Row A
	Second	Row B
64-point Units	First	CN1, row A
	Second	CN1, row B
	Third	CN2, row A
	Fourth	CN2, row B

Note Group-2 High-density I/O Units and B7A Interface Units cannot be mounted to Slave Racks.

3-4 SR (Special Relay) Area

The SR area contains flags and control bits used for monitoring PC operation, accessing clock pulses, and signalling errors. SR area word addresses range from 236 through 511; bit addresses, from 23600 through 51115.

The following table lists the functions of SR area flags and control bits. Most of these bits are described in more detail following the table. Descriptions are in order by bit number except that Link System bits are grouped together.

Unless otherwise stated, flags are OFF until the specified condition arises, when they are turned ON. Restart bits are usually OFF, but when the user turns one ON then OFF, the specified Link Unit will be restarted. Other control bits are OFF until set by the user.

Note all SR words and bits are writeable by the user. Be sure to check the function of a bit or word before attempting to use it in programming.

Word(s)	Bit(s)	Function
236	00 to 07	Node loop status output area for operating level 0 of SYSMAC NET Link System
	08 to 15	Node loop status output area for operating level 1 of SYSMAC NET Link System
237	00 to 07	Completion code output area for operating level 0 following execution of SEND(90)/RECV(98) SYSMAC LINK/SYSMAC NET Link System
	08 to 15	Completion code output area for operating level 1 following execution of SEND(90)/RECV(98) SYSMAC LINK/SYSMAC NET Link System
238 and 241	00 to 15	Data link status output area for operating level 0 of SYSMAC LINK or SYSMAC NET Link System
242 and 245	00 to 15	Data link status output area for operating level 1 of SYSMAC LINK or SYSMAC NET Link System
246	00 to 15	Reserved by system
247 and 248	00 to 07	PC Link Unit Run Flags for Units 16 through 31 or data link status for operating level 1
	08 to 15	PC Link Unit Error Flags for Units 16 through 31 or data link status for operating level 1
249 and 250	00 to 07	PC Link Unit Run Flags for Units 00 through 15 or data link status for operating level 0
	08 to 15	PC Link Unit Error Flags for Units 00 through 15 or data link status for operating level 0
251 Writeable	00	Remote I/O Error Read Bit
	01 and 02	Not used
	03	Remote I/O Error Flag
	04 to 06	Unit number of Remote I/O Unit, Optical I/O Unit, or I/O Terminal with error
	07	Not used
	08 to 15	Word allocated to Remote I/O Unit, Optical I/O Unit, or I/O Terminal with error (BCD)
252	00	SEND(90)/RECV(98) Error Flag for operating level 0 of SYSMAC LINK or SYSMAC NET Link System
	01	SEND(90)/RECV(98) Enable Flag for operating level 0 of SYSMAC LINK or SYSMAC NET Link System
	02	Operating Level 0 Data Link Operating Flag
	03	SEND(90)/RECV(98) Error Flag for operating level 1 of SYSMAC LINK or SYSMAC NET Link System
	04	SEND(90)/RECV(98) Enable Flag for operating level 1 of SYSMAC LINK or SYSMAC NET Link System
	05	Operating Level 1 Data Link Operating Flag
	06	Rack-mounting Host Link Unit Level 1 Communications Error Flag
	07	Rack-mounting Host Link Unit Level 1 Restart Bit
	08	Peripheral Port Restart Bit
	09	RS-232C Port Restart Bit
	10	PC Setup Clear Bit
	11	Forced Status Hold Bit
	12	Data Retention Control Bit
	13	Rack-mounting Host Link Unit Level 0 Restart Bit
	14	Not used.
	15	Output OFF Bit
253	00 to 07	FAL number output area (see error information provided elsewhere)
	08	Low Battery Flag
	09	Cycle Time Error Flag
	10	I/O Verification Error Flag
	11	Rack-mounting Host Link Unit Level 0 Communications Error Flag
	12	Remote I/O Error Flag
	13	Always ON Flag
	14	Always OFF Flag
	15	First Cycle Flag

Word(s)	Bit(s)	Function
254	00	1-minute clock pulse bit
	01	0.02-second clock pulse bit
	02 and 03	Reserved for function expansion. Do not use.
	04	Overflow Flag (for signed binary calculations)
	05	Underflow Flag (for signed binary calculations)
	06	Differential Monitor End Flag
	07	Step Flag
	08	MTR Execution Flag
	09	7SEG Execution Flag
	10	DSW Execution Flag
	11	Interrupt Input Unit Error Flag
	12	Reserved by system
	13	Interrupt Programming Error Flag
	14	Group-2 Error Flag
	15	Special Unit Error Flag (includes Special I/O, PC Link, Host Link, Remote I/O Master Units)
255	00	0.1-second clock pulse bit
	01	0.2-second clock pulse bit
	02	1.0-second clock pulse bit
	03	Instruction Execution Error (ER) Flag
	04	Carry (CY) Flag
	05	Greater Than (GR) Flag
	06	Equals (EQ) Flag
	07	Less Than (LE) Flag
	08 to 15	Reserved by system (used for TR bits)
256 to 261	00 to 15	Reserved by system
262	00 to 15	Longest interrupt subroutine (action) execution time (0.1 ms)
263	00 to 15	Number of interrupt subroutine (action) with longest execution time. (8000 to 8512) 8000 to 8007, 8099 Bit 15: Interrupt Flag
264	00 to 03	RS-232C Port Error Code 0: No error 1: Parity error 2: Framing error 3: Overrun error
	04	RS-232C Port Communications Error
	05	RS-232C Port Send Ready Flag
	06	RS-232C Port Reception Completed Flag
	07	RS-232C Port Reception Overflow Flag
	08 to 11	Peripheral Port Error Code in General I/O Mode 0: No error 1: Parity error 2: Framing error 3: Overrun error F: When in Peripheral Bus Mode
	12	Peripheral Port Communications Error in General I/O Mode
	13	Peripheral Port Send Ready Flag in General I/O Mode
	14	Peripheral Port Reception Completed Flag in General I/O Mode
	15	Peripheral Port Reception Overflow Flag in General I/O Mode
265	00 to 15	RS-232C Port Reception Counter in General I/O Mode
266	00 to 15	Peripheral Reception Counter in General I/O Mode (BCD)

Word(s)	Bit(s)	Function	
267	00 to 04	Reserved by system (not accessible by user)	
	05	Host Link Level 0 Send Ready Flag	
	06 to 12	Reserved by system (not accessible by user)	
	13	Host Link Level 1 Send Ready Flag	
	14 and 15	Reserved by system (not accessible by user)	
268	00 to 15	Reserved by system (not accessible by user)	
269	00 to 07	Memory Cassette Contents 00: Nothing; 01: UM; 02: IOM (03: HIS)	
	08 to 10	Memory Cassette Capacity 0: 0 KW (no cassette); 3: 16 KW	
	11 to 13	Reserved by system (not accessible by user)	
	14	EEPROM Memory Cassette Protected or EPROM Memory Cassette Mounted Flag	
	15	Memory Cassette Flag	
270	00	Save UM to Cassette Bit	Data transferred to Memory Cassette when Bit is turned ON in PROGRAM mode. Bit will automatically turn OFF. An error will be produced if turned ON in any other mode.
	01	Load UM from Cassette Bit	
	02	Compare UM to Cassette Bit	
	03	Comparison Results 0: Contents identical; 1: Contents differ or comparison not possible	
	04 to 10	Reserved by system (not accessible by user)	
	11	Transfer Error Flag: Transferring SYSMAC NET data link table on UM during active data link.	Data will not be transferred from UM to the Memory Cassette if an error occurs (except for Board Checksum Error). Detailed information on checksum errors occurring in the Memory Cassette will not be output to SR 272 because the information is not needed. Repeat the transmission if SR 27015 is ON.
	12	Transfer Error Flag: Not PROGRAM mode	
	13	Transfer Error Flag: Read Only	
	14	Transfer Error Flag: Insufficient Capacity or No UM	
	15	Transfer Error Flag: Board Checksum Error	
271	00 to 07	Ladder program size stored in Memory Cassette Ladder-only File: 04: 4 KW; 08: 8 KW; 12: 12 KW; ... (64: 64 KW) 00: No ladder program or no file Data updated at data transfer from CPU at startup. The file must begin in segment 0.	
	08 to 15	Ladder program size and type in CPU (Specifications are the same as for bits 00 to 07.) Data updated when indexes generated. Default value (after clearing memory) is 16.	
272	00 to 10	Reserved by system (not accessible by user)	
	11	Memory Error Flag: PC Setup Checksum Error	
	12	Memory Error Flag: Ladder Checksum Error	
	13	Memory Error Flag: Instruction Change Vector Area Checksum Error	
	14	Memory Error Flag: Memory Cassette Online Disconnection	
	15	Memory Error Flag: Autoboot Error	

Word(s)	Bit(s)	Function	
273	00	Save IOM to Cassette Bit	Data transferred to Memory Cassette when Bit is turned ON in PROGRAM mode. Bit will automatically turn OFF. An error will be produced if turned ON in any other mode.
	01	Load IOM from Cassette Bit	
	02 to 11	Reserved by system (not accessible by user)	
	12	Transfer Error Flag: Not PROGRAM mode	Data will not be transferred from IOM to the Memory Cassette if an error occurs (except for Read Only Error).
	13	Transfer Error Flag: Read Only	
	14	Transfer Error Flag: Insufficient Capacity or No IOM	
	15	Transfer Error Flag: Checksum Error	
274	00	Special I/O Unit #0 Restart Flag	These flags will turn ON during restart processing. These flags will not turn ON for Units on Slave Racks.
	01	Special I/O Unit #1 Restart Flag	
	02	Special I/O Unit #2 Restart Flag	
	03	Special I/O Unit #3 Restart Flag	
	04	Special I/O Unit #4 Restart Flag	
	05	Special I/O Unit #5 Restart Flag	
	06	Special I/O Unit #6 Restart Flag	
	07	Special I/O Unit #7 Restart Flag	
	08	Special I/O Unit #8 Restart Flag	
	09	Special I/O Unit #9 Restart Flag	
	10 to 15	Reserved by system (not accessible by user)	
275	00	PC Setup Startup Error (DM 6600 to DM 6614)	
	01	PC Setup RUN Error (DM 6615 to DM 6644)	
	02	PC Setup Communications/Error Setting/Misc. Error (DM 6645 to DM 6655)	
	03 to 15	Reserved by system (not accessible by user)	
276	00 to 07	Minutes (00 to 59)	Used for time increments.
	08 to 15	Hours (00 to 23)	
277 to 279	00 to 15	Used for keyboard mapping. See page 368.	
280 to 289	00 to 15	Reserved by system (not accessible by user)	
290 to 293	00 to 15	Macro Area inputs.	
294 to 297	00 to 15	Macro Area outputs.	
298 to 299	00 to 15	Reserved by system (not accessible by user)	

3-4-1 SYSMAC NET/SYSMAC LINK System

Loop Status

SR 236 provides the local node loop status for SYSMAC NET Systems, as shown below.

–––	Bit in SR 236							
Level 0	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
Level 1	**15**	**14**	**13**	**12**	**11**	**10**	**09**	**08**
Status/ Meaning	1	1	Central Power Supply 0: Connected 1: Not connected	1	Loop Status 11: Normal loop 10: Downstream backloop 01: Upstream backloop 00: Loop error		Reception Status 0: Reception enabled 1: Reception disabled	1

Completion Codes

SR 23700 to SR23707 provide the SEND/RECV completion code for operating level 0 and SR 23708 to SR 23215 provide the SEND/RECV completion code for operating level 1. The completion codes are as given in the following tables.

SYSMAC LINK

Code	Item	Meaning
00	Normal end	Processing ended normally.
01	Parameter error	Parameters for network communication instruction is not within acceptable ranges.
02	Unable to send	Unit reset during command processing or local node in not in network.
03	Destination not in network	Destination node is not in network.
04	Busy error	The destination node is processing data and cannot receive the command.
05	Response timeout	The response monitoring time was exceeded.
06	Response error	There was an error in the response received from the destination node.
07	Communications controller error	An error occurred in the communications controller.
08	Setting error	There is an error in the node address settings.
09	PC error	An error occurred in the CPU of the destination node.

SYSMAC NET

Code	Item	Meaning
00	Normal end	Processing ended normally.
01	Parameter error	Parameters for network communication instruction is not within acceptable ranges.
02	Routing error	There is a mistake in the routing tables for connection to a remote network.
03	Busy error	The destination node is processing data and cannot receive the command.
04	Send error (token lost)	The token was not received from the Line Server.
05	Loop error	An error occurred in the communications loop.
06	No response	The destination node does not exist or the response monitoring time was exceeded.
07	Response error	There is an error in the response format.

Data Link Status

SR 238 to SR 245 contain the data link status for SYSMAC LINK/SYSMAC NET Systems. The data structure depends on the system used to create the data link.

SYSMAC LINK

Operating level 0	Operating level 1	Bit			
		12 to 15	11 to 08	04 to 07	00 to 03
SR 238	SR 242	Node 4	Node 3	Node 2	Node 1
SR 239	SR 243	Node 8	Node 7	Node 6	Node 5
SR 240	SR 244	Node 12	Node 11	Node 10	Node 9
SR 241	SR 245	Node 16	Node 15	Node 14	Node 13

Leftmost bit			Rightmost bit
1: PC RUN status	1: PC CPU error	1: Communica-tions error	1: Data link operating

SYSMAC NET

Operating level 0	Operating level 1	Bit (Node numbers below)															
		15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
SR 238	SR 242	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1
SR 239	SR 243	16	15	14	13	12	11	10	9	16	15	14	13	12	11	10	9
SR 240	SR 244	24	23	22	21	20	19	18	17	24	23	22	21	20	19	18	17
SR 241	SR 245	32	31	30	29	28	27	26	25	32	31	30	29	28	27	26	25

1: PC CPU error

1: PC RUN status

3-4-2 Remote I/O Systems

SR 25312 turns ON to indicate an error has occurred in Remote I/O Systems. The ALM/ERR indicator will flash, but PC operation will continue. SR 251, as well as AR 0014 and AR 0015, contain information on the source and type of error. The function of each bit is described below. Refer to *Optical* and *Wired Remote I/O System Manuals* for details.

Bit 00 – Error Check Bit

If there are errors in more than one Remote I/O Unit, word 251 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 the Error Check bit ON and OFF. Be sure to record data for the first error, which will be cleared when data for the next error is displayed.

Bits 01 and 02

Not used.

Bit 03

Remote I/O Error Flag: Bit 03 turns ON when an error has occurred in a Remote I/O Unit.

Bits 04 to 15

The content of bits 04 to 06 is a 3-digit binary number (04: 2^0, 05: 2^1, 06: 2^2) and the content of bits 08 to 15 is a 2-digit BCD number (08 to 11: 10^0, 12 to 15: 10^1).

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 unit number, either 0 or 1, of the Master involved. In this case, bits 04 to 06 contain the unit number of the Slave Rack involved.

If the content of bits 12 through 15 is a number from 0 to 31, an error has occurred in an Optical I/O Unit or I/O Terminal. The number is the unit number of the Optical I/O Unit or I/O Terminal involved, and bit 04 will be ON if the Unit is assigned leftmost word bits (08 through 15), and OFF if it is assigned rightmost word bits (00 through 07).

3-4-3 Link System Flags and Control Bits

Use of the following SR bits depends on the configuration of any Link Systems to which your PC belongs. These flags and control bits are used when Link Units, such as PC Link Units, Remote I/O Units, or Host Link Units, are mounted to the PC Racks or to the CPU. For additional information, consult the System Manual for the particular Units involved.

The following bits can be employed as work bits when the PC does not belong to the Link System associated with them.

Host Link Systems

Both Error flags and Restart bits are provided for Host Link Systems. Error flags turn ON to indicate errors in Host Link Units. Restart bits are turned ON and then OFF to restart a Host Link Unit. SR bits used with Host Link Systems are summarized in the following table. **Rack-mounting Host Link Unit Restart bits are not effective for the Multilevel Rack-mounting Host Link Units.** Refer to the *Host Link System Manual* for details.

Bit	Flag
25206	Rack-mounting Host Link Unit Level 1 Error Flag
25207	Rack-mounting Host Link Unit Level 1 Restart Bit
25213	Rack-mounting Host Link Unit Level 0 Restart Bit
25311	Rack-mounting Host Link Unit Level 0 Error Flag

PC Link Systems

PC Link Unit Error and Run Flags

When the PC belongs to a PC Link System, words 247 through 250 are used to monitor the operating status of all PC Link Units connected to the PC Link System. This includes a maximum of 32 PC Link Units. If the PC is in a Multilevel PC Link System, half of the PC Link Units will be in a PC Link Subsystem in operating level 0; the other half, in a Subsystem in operating level 1. The actual bit assignments depend on whether the PC is in a Single-level PC Link System or a Multi-level PC Link System. Refer to the *PC Link System Manual* for details. Error and Run Flag bit assignments are described below.

Bits 00 through 07 of each word are the Run flags, which are ON when the PC Link Unit is in RUN mode. Bits 08 through 15 are the Error flags, which are ON when an error has occurred in the PC Link Unit. The following table shows bit assignments for Single-level and Multi-level PC Link Systems.

Single-level PC Link Systems

Flag type	Bit no.	SR 247	SR 248	SR 249	SR 250
Run flags	00	Unit #24	Unit #16	Unit #8	Unit #0
	01	Unit #25	Unit #17	Unit #9	Unit #1
	02	Unit #26	Unit #18	Unit #10	Unit #2
	03	Unit #27	Unit #19	Unit #11	Unit #3
	04	Unit #28	Unit #20	Unit #12	Unit #4
	05	Unit #29	Unit #21	Unit #13	Unit #5
	06	Unit #30	Unit #22	Unit #14	Unit #6
	07	Unit #31	Unit #23	Unit #15	Unit #7
Error flags	08	Unit #24	Unit #16	Unit #8	Unit #0
	09	Unit #25	Unit #17	Unit #9	Unit #1
	10	Unit #26	Unit #18	Unit #10	Unit #2
	11	Unit #27	Unit #19	Unit #11	Unit #3
	12	Unit #28	Unit #20	Unit #12	Unit #4
	13	Unit #29	Unit #21	Unit #13	Unit #5
	14	Unit #30	Unit #22	Unit #14	Unit #6
	15	Unit #31	Unit #23	Unit #15	Unit #7

Multilevel PC Link Systems

Flag type	Bit no.	SR 247	SR 248	SR 249	SR 250
Run flags	00	Unit #8, level 1	Unit #0, level 1	Unit #8, level 0	Unit #0, level 0
	01	Unit #9, level 1	Unit #1, level 1	Unit #9, level 0	Unit #1, level 0
	02	Unit #10, level 1	Unit #2, level 1	Unit #10, level 0	Unit #2, level 0
	03	Unit #11, level 1	Unit #3, level 1	Unit #11, level 0	Unit #3, level 0
	04	Unit #12, level 1	Unit #4, level 1	Unit #12, level 0	Unit #4, level 0
	05	Unit #13, level 1	Unit #5, level 1	Unit #13, level 0	Unit #5, level 0
	06	Unit #14, level 1	Unit #6, level 1	Unit #14, level 0	Unit #6, level 0
	07	Unit #15, level 1	Unit #7, level 1	Unit #15, level 0	Unit #7, level 0
Error flags	08	Unit #8, level 1	Unit #0, level 1	Unit #8, level 0	Unit #0, level 0
	09	Unit #9, level 1	Unit #1, level 1	Unit #9, level 0	Unit #1, level 0
	10	Unit #10, level 1	Unit #2, level 1	Unit #10, level 0	Unit #2, level 0
	11	Unit #11, level 1	Unit #3, level 1	Unit #11, level 0	Unit #3, level 0
	12	Unit #12, level 1	Unit #4, level 1	Unit #12, level 0	Unit #4, level 0
	13	Unit #13, level 1	Unit #5, level 1	Unit #13, level 0	Unit #5, level 0
	14	Unit #14, level 1	Unit #6, level 1	Unit #14, level 0	Unit #6, level 0
	15	Unit #15, level 1	Unit #7, level 1	Unit #15, level 0	Unit #7, level 0

Application Example

If the PC is in a Multilevel PC Link System and the content of word 248 is 02FF, then PC Link Units #0 through #7 of in the PC Link Subsystem assigned operating level 1 would be in RUN mode, and PC Link Unit #1 in the same Subsystem would have an error. The hexadecimal digits and corresponding binary bits of word 248 would be as shown below.

Bit no.	15 00
Binary	0 0 0 0 0 0 1 0 1 1 1 1 1 1 1 1
Hex	0 2 F F

3-4-4 Forced Status Hold Bit

SR 25211 determines whether or not the status of bits that have been force-set or force-reset is maintained when switching between PROGRAM and MONITOR mode to start or stop operation. If SR 25211 is ON, bit status will be maintained; if SR 25211 is OFF, all bits will return to default status when operation is started or stopped. The Forced Status Hold Bit is only effective when enabled in the PC Setup.

The status of SR 25211 in not affected by a power interruption unless the I/O table is registered; in that case, SR 25211 will go OFF.

SR 25211 is not effective when switching to RUN mode.

SR 25211 should be manipulated from a Peripheral Device, e.g., a Programming Console or LSS.

Maintaining Status during Startup

The status of SR 25211 and thus the status of force-set and force-reset bits can be maintained when power is turned off and on by enabling the Forced Status Hold Bit in the PC Setup. If the Forced Status Hold Bit is enabled, the status of SR 25211 will be preserved when power is turned off and on. If this is done and SR 25211 is ON, then the status of force-set and force-reset bits will also be preserved, as shown in the following table.

Status before shutdown	Status at next startup	
SR 25211	SR 25211	Force-set/reset bits
ON	ON	Status maintained
OFF	OFF	Reset

Note Refer to *3-6-4 PC Setup* for details on enabling the Forced Status Hold Bit.

3-4-5 I/O Status Hold Bit

SR 25212 determines whether or not the status of IR and LR area bits is maintained when operation is started or stopped, when operation begins by switching from PROGRAM mode to MONITOR or RUN modes. If SR 25212 is ON, bit status will be maintained; if SR 25212 is OFF, all IR and LR area bits will be reset. The I/O Status Hold Bit is effective only if enabled in the PC Setup.

The status of SR 25212 in not affected by a power interruption unless the I/O table is registered; in that case, SR 25212 will go OFF.

SR 25212 should be manipulated from a Peripheral Device, e.g., a Programming Console or LSS.

Maintaining Status during Startup

The status of SR 25212 and thus the status of IR and LR area bits can be maintained when power is turned off and on by enabling the I/O Status Hold Bit in the PC Setup. If the I/O Status Hold Bit is enabled, the status of SR 25212 will be preserved when power is turned off and on. If this is done and SR 25212 is ON, then the status of IR and LR area bits will also be preserved, as shown in the following table.

Status before shutdown	Status at next startup	
SR 25212	SR 25212	IR and LR bits
ON	ON	Status maintained
OFF	OFF	Reset

Note Refer to *3-6-4 PC Setup* for details on enabling the I/O Status Hold Bit.

3-4-6 Output OFF Bit

SR bit 25215 is turned ON to turn OFF all outputs from the PC. The OUT INHIBIT indicator on the front panel of the CPU will light. When the Output OFF Bit is OFF, all output bits will be refreshed in the usual way.

The status of the Output OFF Bit is maintained for power interruptions or when PC operation is stopped, unless the I/O table has been registered, or the I/O table has been registered and either the Forced Status Hold Bit or the I/O Status Hold Bit has not been enabled in the PC Setup.

3-4-7 FAL (Failure Alarm) Area

A 2-digit BCD FAL code is output to bits 25300 to 25307 when the FAL or FALS instruction is executed. These codes are user defined for use in error diagnosis, although the PC also outputs FAL codes to these bits, such as one caused by battery voltage drop.

This area can be reset by executing the FAL instruction with an operand of 00 or by performing a Failure Read Operation from the Programming Console.

3-4-8 Low Battery Flag

SR bit 25308 turns ON if the voltage of the CPU's backup battery drops. The ALM/ERR indicator on the front of the CPU will also flash.

This bit can be programmed to activate an external warning for a low battery voltage.

The operation of the battery alarm can be disabled in the PC Setup if desired. Refer to *3-6-4 PC Setup* for details.

3-4-9 Cycle Time Error Flag

SR bit 25309 turns ON if the cycle time exceeds 100 ms. The ALM/ERR indicator on the front of the CPU will also flash. Program execution will not stop, however, unless the maximum time limit set for the watchdog timer is exceeded. Timing may become inaccurate after the cycle time exceeds 100 ms.

3-4-10 I/O Verification Error Flag

SR bit 25310 turns ON when the Units mounted in the system disagree with the I/O table registered in the CPU. The ALM/ERR indicator on the front of the CPU will also flash, but PC operation will continue.

To ensure proper operation, PC operation should be stopped, Units checked, and the I/O table corrected whenever this flag goes ON.

3-4-11 First Cycle Flag

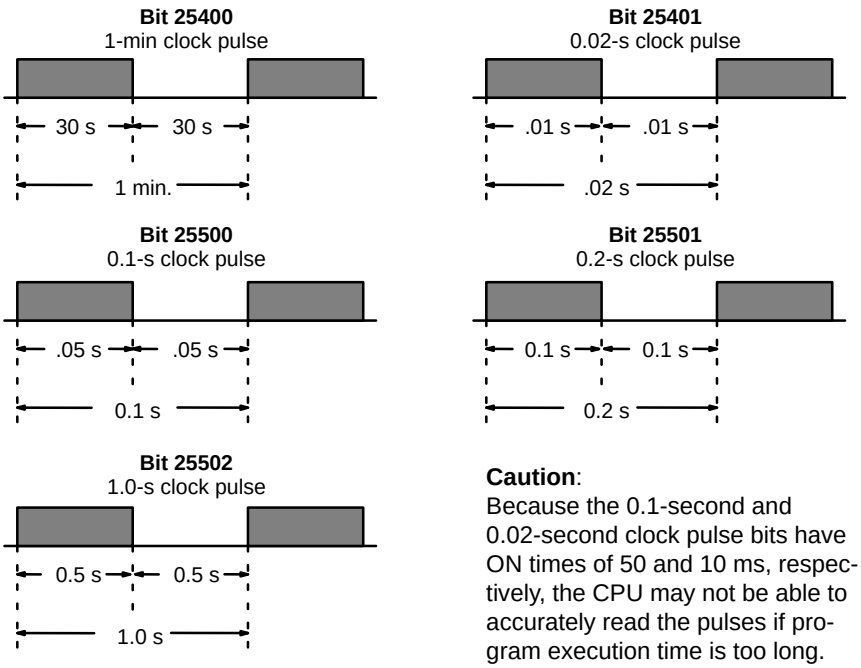
SR bit 25315 turns ON when PC operation begins and then turns OFF after one cycle of the program. The First Cycle Flag is useful in initializing counter values and other operations. An example of this is provided in *5-14 Timer and Counter Instructions*.

3-4-12 Clock Pulse Bits

Five 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 *5-14 Timer and Counter Instructions* for an example of this.

Pulse width	1 min	0.02 s	0.1 s	0.2 s	1.0 s
Bit	25400	25401	25500	25501	25502



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.

3-4-13 Step Flag

SR bit 25407 turns ON for one cycle when step execution is started with the STEP(08) instruction.

3-4-14 Group-2 Error Flag

SR bit 25414 turns ON for any of the following errors for Group-2 High-density I/O Units and B7A Interface Units: the same I/O number set twice, the same words allocated to more than one Unit, refresh errors. If one of these errors occurs, the Unit will stop operation and the ALARM indicator will flash, but the overall PC will continue operation.

When the Group-2 Error Flag is ON, the number of the Unit with the error will be provided in AR 0205 to AR 0214. If the Unit cannot be started properly even though the I/O number is set correctly and the Unit is installed properly, a fuse may be blown or the Unit may contain a hardware failure. If this should occur, replace the Unit with a spare and try to start the system again.

There is also an error flag for High-density I/O Units and B7A Interface Units in the AR area, AR 0215.

3-4-15 Special Unit Error Flag

SR bit 25415 turns ON to indicate errors in the following Units: Special I/O, PC Link, Host Link, and Remote I/O Master Units. SR bit 25415 will turn ON for any of the following errors.

- When more than one Special I/O Unit is set to the same unit number.
- When an error occurs in refreshing data between a Special I/O Unit and the PC's CPU.
- When an error occurs between a Host Link Unit and the PC's CPU.
- When an error occurs in a Remote I/O Master Unit.

Although the PC will continue operation if SR 25415 turns ON, the Units causing the error will stop operation and the ALM indicator will flash. Check the status of AR 0000 to AR 0015 to obtain the unit numbers of the Units for which the error occurred and investigate the cause of the error.

Unit operation can be restarted by using the Restart Bits (AR 0100 to AR 0115, SR 25207, and SR 25213), but will not be effective if the same unit number is set for more than one Special I/O Unit. Turn off the power supply, correct the unit number settings, and turn of the power supply again to restart.

SR 25415 will not turn OFF even if AR 0100 to AR 0115 (Restart Bits) are turned ON. It can be turned OFF by reading errors from a Programming Device or by executing FAL(06) 00 from the ladder program.

3-4-16 Instruction Execution Error Flag, ER

SR bit 25503 turns 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.**

3-4-17 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(01) instruction is executed, and therefore cannot be monitored from a programming device.

Refer to *5-15 Data Shifting*, *5-17 Data Comparison*, *5-19 BCD Calculations*, and *5-20 Binary Calculations* for details.

Overflow Flag, OF SR bit 25404 turns ON when the result of a binary addition or subtraction exceeds 7FFF or 7FFFFFFF.

Underflow Flag, UF SR bit 25405 turns ON when the result of a signed binary addition or subtraction exceeds 8000 or 80000000.

Carry Flag, CY SR bit 25504 turns 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.

Greater Than Flag, GR SR bit 25505 turns ON when the result of a comparison shows the first of two operands to be greater than the second.

Equal Flag, EQ SR bit 25506 turns 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 SR bit 25507 turns ON when the result of a comparison shows the first of two operands to be less than the second.

Note The four arithmetic flags are turned OFF when END(01) is executed.

3-4-18 Interrupt Subroutine Areas

The following areas are used in subroutine interrupt processing.

Interrupt Subroutine Maximum Processing Time Area SR bits 26200 to 26215 are used to set the maximum processing time of the interrupt subroutine. Processing times are determined to within 0.1 ms increments.

Maximum Processing Time Interrupt Subroutine Number Area SR bits 26300 to 26315 contain the maximum processing time interrupt subroutine number. Bit 15 will be ON if there is an interruption.

3-4-19 RS-232C Port Communications Areas

RS-232C Port Error Code SR bits 26400 to 26403 set when there is a RS-232C port error.

Setting	Error type
0	No error
1	Parity error
2	Framing error
3	Overrun error

RS-232C Port Communication Error Bit SR bit 26404 turns ON when there is a RS-232C port communication error.

RS-232C Port Send Ready Flag SR bit 26405 turns ON when the C200HS is ready to transmit data.

RS-232C Port Reception Completed Flag SR bit 26406 turns ON when the C200HS has completed reading data from a RS-232C device.

RS-232C Port Reception Overflow Flag SR bit 26407 turns ON when data overflow occurs following the reception of data.

RS-232C Reception Counter SR areas 26500 to 26515 contains the number of RS-232C port receptions in General I/O Mode.

Host Link Level 0 Send Ready Flag SR bit 26705 turns ON when the C200HS is ready to transmit to the Host Link Unit.

Host Link Level 1 Send Ready Flag SR bit 26713 turns ON when the C200HS is ready to transmit to the Host Link.

3-4-20 Peripheral Port Communications Areas

Peripheral Port Error Code

SR bits 26408 to 26411 are set when there is a peripheral port error in the General I/O Mode.

Setting	Error type
0	No error
1	Parity error
2	Framing error
3	Overrun error
F	Connected in Peripheral Mode

Peripheral Port Communication Error Bit

SR bit 26412 turns ON when there is a peripheral port communication error (effective in General I/O Mode).

Peripheral Port Send Ready Flag

SR bit 26413 turns ON when the C200HS is ready to transmit data in General I/O Mode.

Peripheral Port Reception Completed Flag

SR bit 26414 turns ON when the C200HS has completed reading data from a peripheral device. Effective in General I/O Mode.

Peripheral Port Reception Overflow Flag

SR bit 26415 turns ON when data overflow occurs following the reception of data. Effective in General I/O Mode.

Peripheral Reception Counter

SR areas 26600 to 26615 contains the number of peripheral port receptions in General I/O Mode (BCD).

Host Link Level 0 Send Ready Flag

SR bit 26705 turns ON when the C200HS is ready to transmit to the Host Link Unit.

Host Link Level 1 Receive Ready Flag

SR bit 26713 turns ON when the C200HS is ready to receive data from the Host Link.

3-4-21 Memory Cassette Areas

Memory Cassette Contents

SR areas 26900 to 26907 indicate memory type contained on the Memory Cassette.

Memory Type	Code
Nothing	00
UM	01
IOM	02

Memory Cassette Capacity

SR areas 26908 to 26910 indicate memory capacity of the Memory Cassette.

Capacity	Code
0 KW (no board mounted)	0
16 KW	3

EEPROM/EPROM Memory Cassette Mounted Flag

SR bit 26914 turns ON when EEPROM Memory Cassette is protected or EPROM Memory Cassette is mounted.

Memory Cassette Flag

SR bit 26915 turns ON when a Memory Cassette is mounted.

Save UM to Cassette Flag

SR bit 27000 turns ON when UM data is read to a Memory Cassette in Program Mode. Bit will automatically turn OFF. An error will be produced if turned ON in any other mode.

Load UM from Cassette Flag

SR bit 27001 turns ON when data is loaded into UM from a Memory Cassette in Program Mode. Bit will automatically turn OFF. An error will be produced if turned ON in any other mode.

Collation (Between DM and Memory Cassette)

SR bit 27002 turns ON when data is verified between DM and a Memory Cassette. SR bit 27003 turns OFF when the contents of the verification coincide and turns ON when the contents of the verification do not coincide.

3-4-22 Data Transfer Error Bits

Data will not be transferred from UM to the Memory Cassette if an error occurs (except for Board Checksum Error). Detailed information on checksum errors occurring in the Memory Cassette will not be output to SR 272 because the information is not needed. Repeat the transmission if SR 27015 is ON

Transfer Error Flag: Not PROGRAM Mode

SR bit 27012 turns ON when the C200HS is not in Program Mode and data transfer is attempted.

Transfer Error Flag: Read Only

SR bit 27013 turns ON when the C200HS is in Read-only Mode and data transfer is attempted.

Transfer Error Flag: Insufficient Capacity or No UM

SR bit 27014 turns ON when data transfer is attempted and available UM is insufficient.

Transfer Error Flag: Board Checksum Error

SR bit 27015 turns ON when data transfer is attempted and a Board Checksum error occurs.

3-4-23 Ladder Diagram Memory Areas

Memory Cassette Ladder Diagram Size Area

SR areas 27100 to 27107 indicate the amount of ladder program stored in a Memory Cassette.

Ladder-only File: 04: 4 KW; 08: 8 KW; 12: 12 KW; ... (64: 64 KW)
(Ladder File (Bit 07 will be ON): 84: 4 KW; 88: 8 KW; 92: 12 KW;
... (E4: 64 KW))

00: No ladder program or no file

Data updated at data transfer from CPU at startup. The file must begin in segment 0.

CPU Ladder Diagram Size and Type

SR areas 27108 to 27115 indicate the CPU's ladder program size and type. Specifications are the same as for bits 00 to 07.

3-4-24 Memory Error Flags

Memory Error Flag: PC Setup Error

SR bit 27211 turns ON when a PC Setup Checksum error occurs.

Memory Error Flag: Ladder Checksum Error

SR bit 27212 turns ON when a Ladder Checksum error occurs.

Memory Error Flag: Instruction Change Error

SR bit 27213 turns ON when an instruction change vector area error occurs.

Memory Error Flag: Memory Cassette Disconnect Error

SR bit 27214 turns ON when a Memory Cassette is connected or disconnected during operations.

Memory Error Flag: Autoboot Error

SR bit 27215 turns ON when an autoboot error occurs.

3-4-25 Data Save Flags

Data transferred to Memory Cassette when Bit is turned ON in PROGRAM mode. Bit will automatically turn OFF. An error will be produced if turned ON in any other mode.

Save IOM to Cassette Bit SR bit 27300 turns ON when IOM is saved to a Memory Cassette.

Load IOM from Cassette Bit SR bit 27301 turns ON when loading to IOM from a Memory Cassette.

3-4-26 Transfer Error Flags

Data will not be transferred from IOM to the Memory Cassette if an error occurs (except for Read Only Error).

Transfer Error Flag: Not PROGRAM mode SR bit 27312 turns ON when attempting to transfer data in other than Program Mode.

Transfer Error Flag SR bit 27313 turns ON when attempting to transfer data in Read-only Mode.

Transfer Error Flag SR bit 27314 turns ON when attempting to transfer data and IOM capacity is insufficient.

3-4-27 PC Setup Error Flags

PC Setup Startup Error SR bit 27500 turns ON when a PC Setup Startup error occurs (DM6600 to DM6614).

PC Setup RUN Error SR bit 27501 turns ON when a PC Setup Run error occurs (DM6615 to DM6644).

PC Setup Communications/Error Setting/Misc. Error SR bit 27501 turns ON when a PC Setup Communications, Error setting or Miscellaneous error occurs (DM6645 to DM6655).

Minutes (00 to 59) SR bits 27600 to 27607 set the PC Clock to minutes (00 to 59).

Hours (00 to 23) SR bits 27608 to 27615 set the PC Clock to hours (0 to 23).

Keyboard Map Used for keyboard mapping.

3-5 AR (Auxiliary Relay) Area

AR word addresses extend from AR 00 to AR 27; AR bit addresses extend from AR 0000 to AR 2715. Most AR area words and bits are dedicated to specific uses, such as transmission counters, flags, and control bits, and words AR 00 through AR 07 and AR 23 through AR 27 cannot be used for any other purpose. Words and bits from AR 08 to AR 22 are available as work words and work bits if not used for the following assigned purposes.

Word	Use
AR 0713 to AR 0715	Error History Area
AR 07 to 15	SYSMAC LINK Units
AR 16, AR 17	SYSMAC LINK and SYSMAC NET Link Units
AR 18 to AR 21	Calendar/clock Area
AR 0708, AR 0709, and AR 22	TERMINAL Mode Key Bits

The AR area retains status during power interruptions, when switching from MONITOR or RUN mode to PROGRAM mode, or when PC operation is stopped. Bit allocations are shown in the following table and described in the following pages in order of bit number.

AR Area Flags and Control Bits

Word(s)	Bit(s)	Function
00	00 to 09	Error Flags for Special I/O Units 0 to 9 (also function as Error Flags for PC Link Units)
	10	Error Flag for operating level 1 of SYSMAC LINK or SYSMAC NET Link System
	11	Error Flag for operating level 0 of SYSMAC LINK or SYSMAC NET Link System
	12	Host Computer to Rack-mounting Host Link Unit Level 1 Error Flag
	13	Host Computer to Rack-mounting Host Link Unit Level 0 Error Flag
	14	Remote I/O Master Unit 1 Error Flag
	15	Remote I/O Master Unit 0 Error Flag

Word(s)	Bit(s)	Function
01	00 to 09	Restart Bits for Special I/O Units 0 to 9 (also function as Restart Bits for PC Link Units)
	10	Restart Bit for operating level 1 of SYSMAC LINK or SYSMAC NET Link System
	11	Restart Bit for operating level 0 of SYSMAC LINK or SYSMAC NET Link System
	12, 13	Not used.
	14	Remote I/O Master Unit 1 Restart Flag.
	15	Remote I/O Master Unit 0 Restart Flag.
02	00 to 04	Slave Rack Error Flags (#0 to #4)
	05 to 14	Group-2 Error Flags
	15	Group-2 Error Flag
03	00 to 15	Error Flags for Optical I/O Units and I/O Terminals 0 to 7
04	00 to 15	Error Flags for Optical I/O Units and I/O Terminals 8 to 15
05	00 to 15	Error Flags for Optical I/O Units and I/O Terminals 16 to 23
06	00 to 15	Error Flags for Optical I/O Units and I/O Terminals 24 to 31
07	00 to 03	Data Link setting for operating level 0 of SYSMAC LINK System
	04 to 07	Data Link setting for operating level 1 of SYSMAC LINK System
	08	Normal TERMINAL Mode/Expansion TERMINAL Mode Input Cancel Bit
	09	Expansion TERMINAL Mode Changeover Flag
	10 and 11	Reserved by system.
	12	Terminal Mode Flag ON: Expansion; OFF: Normal (Same as status of pin 6 on CPU's DIP switch)
	13	Error History Overwrite Bit
	14	Error History Reset Bit
	15	Error History Enable Bit
08 to 11	00 to 15	Active Node Flags for SYSMAC LINK System nodes of operating level 0
12 to 15	00 to 15	Active Node Flags for SYSMAC LINK System nodes of operating level 1
16	00 to 15	SYSMAC LINK/SYSMAC NET Link System operating level 0 service time per cycle
17	00 to 15	SYSMAC LINK/SYSMAC NET Link System operating level 1 service time per cycle
18 Writeable	00 to 07	Seconds: 00 to 99
	08 to 15	Minutes: 00 to 59
19 Writeable	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)
20 Writeable	00 to 07	Month: 1 to 12
	08 to 15	Year: 00 to 99 (Rightmost two digits of year)
21 Writeable	00 to 07	Day of Week: 00 to 06 (00: Sunday; 01: Monday; 02: Tuesday; 03: Wednesday; 04: Thursday; 05: Friday; 06: Saturday)
	08 to 12	Not used.
	13	30-second Compensation Bit
	14	Clock Stop Bit
	15	Clock Set Bit
22	00 to 15	Keyboard Mapping
23	00 to 15	Power Off Counter (BCD)

Word(s)	Bit(s)	Function
24	00 to 04	Reserved by system.
	05	Cycle Time Flag
	06	SYSMAC LINK System Network Parameter Flag for operating level 1
	07	SYSMAC LINK System Network Parameter Flag for operating level 0
	08	SYSMAC/SYSMAC NET Link Unit Level 1 Mounted Flag
	09	SYSMAC/SYSMAC NET Link Unit Level 0 Mounted Flag
	10	Reserved by system.
	11 and 12	PC Link Level
	13	Rack-mounting Host Link Unit Level 1 Mounted Flag
	14	Rack-mounting Host Link Unit Level 0 Mounted Flag
	15	CPU-mounting Device Mounted Flag
25	00 to 11	Reserved by system.
	12	Trace End Flag
	13	Tracing Flag
	14	Trace Trigger Bit (writeable)
	15	Trace Start Bit (writeable)
26	00 to 15	Maximum Cycle Time (0.1 ms)
27	00 to 15	Present Cycle Time (0.1 ms)

3-5-1 Restarting Special I/O Units

To restart Special I/O Units (including PC Link Units) turn the corresponding bit ON and OFF (or turn power ON and OFF). Do not access data refreshed for Special I/O Units during restart processing (see SR 27400 to SR 27409 on page 37).

3-5-2 Slave Rack Error Flags

AR bits 0200 to AR 0204 correspond to the unit numbers of Remote I/O Slave Units #0 to #4 and AR bits 0710 to AR 0712 correspond to the unit numbers of Remote I/O Slave Units #5 to #7. These flags will turn ON if the same number is allocated to more then one Slave or if a transmission error occurs when starting the System. Refer to SR 251 for errors that occur after the System has started normally.

3-5-3 Group-2 Error Flags

AR bits 0205 to AR 0215 correspond to Group-2 High-density I/O Units and B7A Interface Units 0 to 9 (I/O numbers) and will turn ON when the same number is set for more than one Unit, when the same word is allocated to more than one Unit, when I/O number 9 is set for a 64-point Unit, or when the fuse burns out in a Transistor High-density I/O Unit. AR bit 0215 will turn ON when a Unit is not recognized as a Group-2 High-density I/O Unit.

3-5-4 Optical I/O Unit and I/O Terminal Error Flags

AR 03 through AR 06 contain the Error Flags for Optical I/O Units and I/O Terminals. An error indicates a duplication of a unit number. Up to 64 Optical I/O Units and I/O Terminals can be connected to the PC. Units are distinguished by unit

number, 0 through 31, and a letter, L or H. Bits are allocated as shown in the following table.

Optical I/O Unit and I/O Terminal Error Flags

Bits	AR03 allocation	AR04 allocation	AR05 allocation	AR06 allocation
00	0 L	8 L	16 L	24 L
01	0 H	8 H	16 H	24 H
02	1 L	9 L	17 L	25 L
03	1 H	9 H	17 H	25 H
04	2 L	10 L	18 L	26 L
05	2 H	10 H	18 H	26 H
06	3 L	11 L	19 L	27 L
07	3 H	11 H	19 H	27 H
08	4 L	12 L	20 L	28 L
09	4 H	12 H	20 H	28 H
10	5 L	13 L	21 L	29 L
11	5 H	13 H	21 H	29 H
12	6 L	14 L	22 L	30 L
13	6 H	14 H	22 H	30 H
14	7 L	15 L	23 L	31 L
15	7 H	15 H	23 H	31 H

3-5-5 SYSMAC LINK System Data Link Settings

AR 0700 to AR 0703 and AR 0704 to AR 0707 are used to designate word allocations for operating levels 0 and 1 of the SYSMAC LINK System. Allocation can be set to occur either according to settings from an FIT or automatically in the LR and/or DM areas. If automatic allocation is designated, the number of words to be allocated to each node is also designated. These settings are shown below.

External/Automatic Allocation

Operating level 0		Operating level 1		Setting	
AR 0700	AR 0701	AR 0704	AR 0705		
0	0	0	0	Words set externally (FIT)	
1	0	1	0	Automatic allocation	LR area only
0	1	0	1		DM area only
1	1	1	1		LR and DM areas

Words per Node

The following setting is necessary if automatic allocation is designated above.

Operating level 0		Operating level 1		Words per node		Max. no. of nodes
AR 0702	AR 0703	AR 0706	AR 0707	LR area	DM area	
0	0	0	0	4	8	16
1	0	1	0	8	16	8
0	1	0	1	16	32	4
1	1	1	1	32	64	2

The above settings are read every cycle while the SYSMAC LINK System is in operation.

3-5-6 Error History Bits

AR 0713 (Error History Overwrite Bit) is turned ON or OFF by the user to control overwriting of records in the Error History Area in the DM area. Turn AR 0713 ON to overwrite the oldest error record each time an error occurs after 10 have been recorded. Turn OFF AR 0713 to store only the first 10 records that occur each time after the history area is cleared.

AR 0714 (Error History Reset Bit) is turned ON and then OFF by the user to reset the Error Record Pointer (DM 0969) and thus restart recording error records at the beginning of the history area.

AR 0715 (Error History Enable Bit) is turned ON by the user to enable error history storage and turned OFF to disable error history storage.

Refer to *3-6 DM Area* for details on the Error History Area.

Error history bits are refreshed each cycle.

3-5-7 Active Node Flags

AR 08 through AR 11 and AR 12 through AR 15 provide flags that indicate which nodes are active in the SYSMAC LINK System at the current time. These flags are refreshed every cycle while the SYSMAC LINK System is operating.

The body of the following table show the node number assigned to each bit. If the bit is ON, the node is currently active.

Level 0	Level 1	Bit (body of table shows node numbers)															
		00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
AR 08	AR 12	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
AR 09	AR 13	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
AR 10	AR 14	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
AR 11	AR 15	49	50	51	52	53	54	55	56	57	58	59	60	61	62	*	**

*Communication Controller Error Flag
**EEPROM Error Flag

3-5-8 SYSMAC LINK/SYSMAC NET Link System Service Time

AR 16 provides the time allocated to servicing operating level 0 of the SYSMAC LINK System and/or SYSMAC NET Link System during each cycle when a SYSMAC LINK Unit and/or SYSMAC NET Link Unit is mounted to a Rack.

AR 17 provides the time allocated to servicing operating level 1 of the SYSMAC LINK System and/or SYSMAC NET Link System during each cycle when a SYSMAC LINK Unit and/or SYSMAC NET Link Unit is mounted to a Rack.

These times are recorded in 4-digit BCD to tenths of a millisecond (000.0 ms to 999.9 ms) and are refreshed every cycle.

Bits			
15 to 12	11 to 08	07 to 04	03 to 00
10^2	10^1	10^0	10^{-1}

3-5-9 Calendar/Clock Area and Bits

Calendar/Clock Area

A clock is built into the C200HS CPUs. If AR 2114 (Clock Stop Bit) is OFF, then the date, day, and time will be available in BCD in AR 18 to AR 20 and AR 2100 to AR 2108 as shown below. This area can also be controlled with AR 2113 (30-second Compensation Bit) and AR 2115 (Clock Set Bit).

Calendar/Clock Bits

Bits	Contents	Possible values
AR 1800 to AR 1807	Seconds	00 to 59
AR 1808 to AR 1815	Minutes	00 to 59
AR 1900 to AR 1907	Hours	00 to 23 (24-hour system)
AR 1908 to AR 1915	Day of month	01 to 31 (adjusted by month and for leap year)
AR 2000 to AR 2007	Month	1 to 12
AR 2008 to AR 2015	Year	00 to 99 (Rightmost two digits of year)
AR 2100 to AR 2107	Day of week	00 to 06 (00: Sunday; 01: Monday; 02: Tuesday; 03: Wednesday; 04: Thursday; 05: Friday; 06: Saturday)

30-second Compensation Bit AR 2113 is turned ON to round the seconds of the Calendar/clock Area to zero, i.e., if the seconds is 29 or less, it is merely set to 00; if the seconds is 30 or greater, the minutes is incremented by 1 and the seconds is set to 00.

Clock Stop Bit AR 2114 is turned OFF to enable the operation of the Calendar/clock Area and ON to stop the operation.

Clock Set Bit AR 2115 is used to set the Calendar/clock Area as described below. This data must be in BCD and must be set within the limits for the Calendar/clock Area given above.

1, 2, 3...

1. Turn ON AR 2114 (Stop Bit).
2. Set the desired date, day, and time, being careful not to turn OFF AR 2114 (Clock Stop Bit) when setting the day of the week (they're in the same word). (On the Programming Console, the Bit/Digit Monitor and Force Set/Reset Operations are the easiest ways to set this data.)
3. Turn ON AR 2115 (Clock Set Bit). The Calendar/clock will automatically start operating with the designated settings and AR 2114 and AR 2115 will both be turned OFF.

The Calendar/clock Area and Bits are refreshed each cycle while operational.

Clock Accuracy Clock accuracy is affected by the ambient temperature as shown in the following table.

Ambient temperature	Accuracy (loss or gain per month)
55°C	–3 to 0 minutes
25°C	±1 minute
0°C	–2 to 0 minutes

3-5-10 TERMINAL Mode Key Bits

If the Programming Console is mounted to the PC and is in TERMINAL mode, any inputs on keys 0 through 9 (including characters A through F, i.e, keys 0 through 5 with SHIFT) will turn on a corresponding bit in AR 22. TERMINAL mode is entered by a Programming Console operation.

The bits in AR 22 correspond to Programming Console inputs as follows:

Bit	Programming Console input
AR 2200	0
AR 2201	1
AR 2202	2
AR 2203	3
AR 2204	4
AR 2205	5
AR 2206	6
AR 2207	7
AR 2208	8
AR 2209	9
AR 2210	A
AR 2211	B
AR 2212	C
AR 2213	D
AR 2214	E
AR 2215	F

Refer to *Section 7 Program Monitoring and Execution* for details on the TERMINAL mode.

3-5-11 Power OFF Counter

AR 23 provides in 4-digit BCD the number of times that the PC power has been turned off. This counter can be reset as necessary using the PV Change 1 operation from the Programming Console. (Refer to *7-1-4 Hexadecimal/BCD Data Modification* for details.) The Power OFF Counter is refreshed every time power is turned on.

3-5-12 Cycle Time Flag

AR 2405 turns ON when the cycle time set with SCAN(18) is shorter than the actual cycle time.

AR 2405 is refreshed every cycle while the PC is in RUN or MONITOR mode.

3-5-13 Link Unit Mounted Flags

The following flags indicate when the specified Link Units are mounted to the Racks. (Refer to *3-5-14 CPU-mounting Device Mounted Flag* for CPU-mounting Host Link Units.) These flags are refreshed every cycle.

Name	Bit	Link Unit
PC Link Unit Level 1	AR 2411	PC Link Unit in operating level 1
PC Link Unit Level 0	AR 2412	PC Link Unit in operating level 0
Rack-mounting Host Link Unit Level 1	AR 2413	Rack-mounting Host Link Unit in operating level 1
Rack-mounting Host Link Unit Level 0	AR 2414	Rack-mounting Host Link Unit in operating level 0

3-5-14 CPU-mounting Device Mounted Flag

AR 2415 turns ON when any device is mounted directly to the CPU. This includes CPU-mounting Host Link Units, Programming Consoles, and Interface Units. This flag is refreshed every cycle.

3-5-15 FPD Trigger Bit

AR 2508 is used to adjust the monitoring time of FPD(—) automatically. Refer to *5-25-12 FAILURE POINT DETECT – FPD(—)* for details.

3-5-16 Data Tracing Flags and Control Bits

The following control bits and flags are used during data tracing with TRSM(45). 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.

Bit	Name
AR 2512	Trace Completed Flag
AR 2513	Tracing Flag
AR 2514	Trace Trigger Bit (writeable)
AR 2515	Sampling Start Bit (writeable)

Note Refer to *5-25-3 TRACE MEMORY SAMPLING – TRSM(45)* for details.

3-5-17 Cycle Time Indicators

AR 26 contains the maximum cycle time that has occurred since program execution was begun. AR 27 contains the present cycle time.

Both times are to tenths of a millisecond in 4-digit BCD (000.0 ms to 999.9 ms), and are refreshed every cycle.

3-6 DM (Data Memory) Area

The DM area is divided into various parts as described in the following table. A portion of UM (up to 3,000 words in 1,000-word increments) can be allocated as Expansion DM.

Addresses	User read/write	Usage
DM 0000 to DM 0999	Read/Write	Normal DM.
DM 1000 to DM 1999		Special I/O Unit Area[1]
DM 2000 to DM 5999		Normal DM.
DM 6000 to DM 6030		History Log
DM 6100 to DM 6143		Link test area (reserved)
DM 6144 to DM 6599	Read only	System Settings
DM 6600 to DM 6655		PC Setup
DM 7000 to DM 9999		Expansion DM[2]

Note

1. The PC Setup can be set to use DM 7000 through DM 7999 as the Special I/O Area instead of DM 1000 to DM 1999. Refer to *3-6-4 PC Setup* for details.
2. The UM ALLOCATION Programming Console operation can be used to allocate up to 3000 words of UM as Expansion DM.

Although composed of 16-bit words like any other data area, data in the DM area cannot be specified by bit for use in instructions with bit operands. DM 0000 to DM 6143 can be written to by the program, but DM 6144 to DM 6655 can be overwritten only from a Peripheral Device, such as a Programming Console or host computer with LSS.

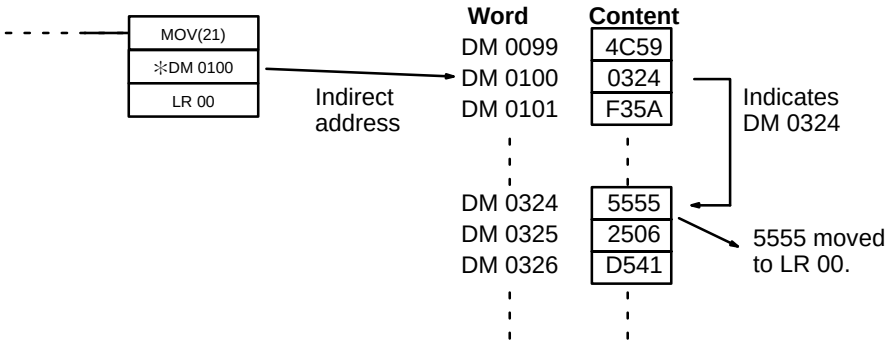
The DM area retains status during power interruptions.

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 MOV(21) is performed with DM 0100 as the first operand and LR 20 as the second operand. When this instruction is executed, the content of DM 0100 is moved to LR 20.

Note Expansion DM cannot be used for indirect addressing.

It is possible, however, to use indirect DM addresses as the operands for many instructions. To indicate an indirect DM address, *DM is input with the address of the operand. With an indirect address, with content of this operand does not contain the actual data to be used. Instead, it's contents is assumed to hold the address of another DM word, the content of which will actually be used in the instruction. If *DM 0100 was used in our example above and the content of DM 0100 is 0324, then *DM 0100 actually means that the content of DM 0324 is to be used as the operand in the instruction, and the content of DM 0324 will be moved to LR 20.



3-6-1 Expansion DM Area

The expansion DM area is designed to provide memory space for storing operating parameters and other operating data for Link Units and Special I/O Units. Up to 3,000 words of UM can be allocated as Expansion DM (in 1K-word increments) using the UM ALLOCATION operation in the Programming Console or LSS. Expansion DM area addresses run from DM 7000 to DM 9999.

The data in the expansion DM area can be transferred to the Special I/O Unit Default Area (DM 1000 to DM 1999) when starting the PC or via programming instruction to easily change operating parameters, enabling rapid switching between control processes. The expansion DM area can also be used to store parameters for other devices connected in the PC system, e.g., Programmable Terminal character string or numeral tables.

The expansion DM area is used to store operating parameters and cannot be used in programming like the normal DM area. Expansion DM can only be overwritten from a Peripheral Device, retains status during power interruptions, and cannot be used for indirect addressing.

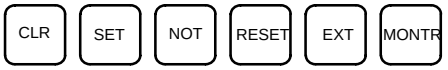
The UM area can be allocated as expansion DM area in increments of 1K words. Once expansion DM area has been created, it is saved and transferred as part of the program, i.e., no special procedures are required when saving or transferring the program.

UM ALLOCATION Operation

The procedure for the Programming Console's UM ALLOCATION operation is shown below. Refer to *1-8-10 New Programming Console Operations* for details on the DATA CLEAR and UM ALLOCATION instructions.

1, 2, 3...

1. Clear memory.



Note UM allocation is not possible unless memory is cleared first.

2. The expansion DM area can be set to 0, 1, 2, or 3 K words. The following key sequence creates a 2-KW expansion DM area (DM 7000 to DM 8999).

Press the 0 Key to eliminate the expansion DM area (0 KW).

or Press the 1 Key to allocate DM 7000 to DM 7999 (1 KW).

or Press the 2 Key to allocate DM 7000 to DM 8999 (2 KW).

or Press the 3 Key to allocate DM 7000 to DM 9999 (3 KW).

3-6-2 Special I/O Unit Data

Special I/O Units are allocated 1000 words in the DM Area as shown in the following table. The value set in the PC Setup (DM 6602 bits 08 to 15) determines

whether DM 1000 to DM 1999 or DM 7000 to 7999 will be used. Refer to *3-6-4 PC Setup* for details.

Unit	Addresses
0	DM 1000 to DM 1099 or DM 7000 to DM 7099
1	DM 1100 to DM 1199 or DM 7100 to DM 7199
2	DM 1200 to DM 1299 or DM 7200 to DM 7299
3	DM 1300 to DM 1399 or DM 7300 to DM 7399
4	DM 1400 to DM 1499 or DM 7400 to DM 7499
5	DM 1500 to DM 1599 or DM 7500 to DM 7599
6	DM 1600 to DM 1699 or DM 7600 to DM 7699
7	DM 1700 to DM 1799 or DM 7700 to DM 7799
8	DM 1800 to DM 1899 or DM 7800 to DM 7899
9	DM 1900 to DM 1999 or DM 7900 to DM 7999

Note These DM words can be used for other purposes when not allocated to Special I/O Units.

3-6-3 Error History Area

DM 6000 to DM 6030 are used to store up to 10 records that show the nature, time, and date of errors that have occurred in the PC.

The Error History Area will store system-generated or FAL(06)/FALS(07)-generated error codes whenever AR 0715 (Error History Enable Bit) is ON. Refer to *Section 10 Troubleshooting* for details on error codes.

Area Structure

Error records occupy three words each stored between DM 6001 and DM 6030. The last record that was stored can be obtained via the content of DM 6000 (Error Record Pointer). The record number, DM words, and pointer value for each of the ten records are as follows:

Record	Addresses	Pointer value
None	N.A.	0000
1	DM 6001 to DM 6003	0001
2	DM 6004 to DM 6006	0002
3	DM 6007 to DM 6009	0003
4	DM 6010 to DM 6012	0004
5	DM 6013 to DM 6015	0005
6	DM 6016 to DM 6018	0006
7	DM 6019 to DM 6021	0007
8	DM 6022 to DM 6024	0008
9	DM 6025 to DM 6027	0009
10	DM 6028 to DM 6030	000A

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 and third words, the day and time. The error code will be either one generated by the system or by FAL(06)/FALS(07); the time and date will be the date and time from AR 18 and AR 19 (Calender/date Area). Also recorded with the error code is an indication of whether the error is fatal (08) or non-fatal (00). This structure is shown below.

Word	Bit	Content
First	00 to 07	Error code
	08 to 15	00 (non-fatal) or 80 (fatal)
Second	00 to 07	Seconds
	08 to 15	Minutes
Third	00 to 07	Hours
	08 to 15	Day of month

The following table lists the possible error codes and corresponding errors.

Error severity	Error code	Error
Fatal errors	00	Power Interruption
	01 to 99	System error (FALS)
	9F	Cycle time error
	C0 to C2	I/O bus error
	E0	Input-output I/O table error
	E1	Too many Units
	F0	No END(01) instruction
	F1	Memory error
Non-fatal errors	01 to 99	System error (FAL)
	8A	Interrupt Input error
	8B	Interrupt program error
	9A	Group 2 High-density I/O error
	9B	PC Setup error
	9D	UM Memory Cassette transfer error
	B0 to B1	Remote I/O error
	D0	Special I/O error
	E7	I/O table verification error
	F7	Battery error
	F8	Cycle time overrun

Operation

When the first error code is generated with AR 0715 (Error History Enable Bit) turned ON, the relevant data will be placed in the error record after the one indicated by the History 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. Processing of further error records is based on the status of AR 0713 (Error History Overwrite Bit).

If AR 0713 is ON and the Pointer contains 000A, the next error will be written into record 10, the contents of record 10 will be moved to record 9, 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. The Record Pointer will remain set to 000A.

If AR 0713 is OFF and the Pointer reaches 000A, the contents of the Error History Error will remain as it is and any error codes generate thereafter will not be recorded until AR 0713 is turned OFF or until the Error History Area is reset.

The Error History Area can be reset by turning ON and then OFF AR 0714 (Error History Reset Bit). When this is done, the Record Pointer will be reset to 0000, the Error History Area will be reset (i.e., cleared), and any further error codes will be recorded from the beginning of the Error History Area. AR 0715 (Error History Enable Bit) must be ON to reset the Error History Area.

3-6-4 PC Setup

The PC Setup contains settings that determine C200HS operation. Data in the PC Setup can be changed with a Programming Console or LSS if UM is not write-protected by pin 1 of the CPU's DIP switch. Refer to page 23 for details on changing DIP switch pin settings.

The PC can be operated with the default PC Setup, which requires changing only when customizing the PC's operating environment to application needs. The PC Setup parameters are described in the following table. Refer to *Appendix E PC Setup* for more details on these parameters.

The PC Setup is allocated to DM 6600 through DM 6655.

Parameter	Default	Settings	Remarks
STARTUP MODE			
STARTUP MODE	Programming Console mode selector	Programming Console mode selector, previous mode (i.e., the mode in use last time power was interrupted), PROGRAM, MONITOR, or RUN	Determines the operating mode the PC will start in when power is turned ON. This setting is required for restart continuation. Setting is effective from next time power is turned on to the PC.
FORCED STATUS	Don't hold	Hold or don't hold	Determines whether or not the status of the Forced Status Hold Bit is maintained after power interruptions. If the status of the Forced Status Hold Bit is not set to be held, it will be turned OFF the next time the PC is started and forced status will be cleared. Setting is effective from next time power is turned on to the PC.
IOM HOLD BIT STATUS	Don't hold	Hold or don't hold	Determines whether or not the status of the IOM Hold Bit is maintained after power interruptions. If the status of the IOM Hold Bit is not set to be held, it will be turned OFF the next time the PC is started and I/O status will be cleared. This setting is required for restart continuation. Setting is effective from next time power is turned on to the PC.
CYCLE TIME	Variable	Variable or minimum Minimum setting: 1 to 9,999 ms	Determines whether or not a minimum cycle time is to be used for user program execution. If a minimum time is set, the PC will wait until the minimum time has expired before starting program execution again. The entire program will be executed even if the minimum time is exceeded. This setting can be used to reduce variations in I/O response times. An error of approximately 3 to 4 ms, plus the execution time required for any interrupt programs, can occur. Setting is effective immediately.
Detect Long Cycles	120 ms	0 to 99,000 ms	If the set time is exceeded, the Cycle Time Exceeded Flag will turn ON and a fatal error will occur. An error of approximately 3 to 4 ms can occur. Setting is effective immediately.
RS-232C SETUP			
METHOD	Host link	Host Link, RS-232C with no protocol, 1:1 link master, or 1:1 link slave	Determines the settings used when a device, such as a Programmable Terminal or bar code reader is connected to the RS-232C port. Do not set the node number to a number already used by another Unit connected in the same communications system (e.g., Host Link System). All other settings must match those of the device being communicated with. Settings are effective immediately.
NODE NO	0	00 to 31	
DELAY	0	0 to 9,999 ms	
START CODE	None	00 to FF	
END CODE	None	00 to FF or CR, LF	
DATA LINK AREAS	None	LR 00 to LR 63, LR 00 to LR 31, or LR 00 to LR 15	
BAUD RATE	9,600 bps	1200, 2400, 4800, 9600, or 19200	
STOP BITS	2 bits	1 or 2 bits	
PARITY	Even parity	Even, odd, or none	
DATA LENGTH	7 bits	7 or 8 bits	
PC SETUP, HEX INPUT	Used to set the above parameters on a binary display.		

3-7 HR (Holding Relay) Area

The HR area is used to store/manipulate various kinds of data and can be accessed either by word or by bit. Word addresses range from HR 00 through HR 99; bit addresses, from HR 0000 through HR 9915. HR bits can be used in any order required and can be programmed as often as required.

The HR area retains status when the system operating mode is changed, when power is interrupted, or when PC operation is stopped.

HR area bits and words can be used to preserve data whenever PC operation is stopped. HR 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 and Inputting the Program* and *Section 5 Instruction Set*.

Note The required number of words is allocated between HR 00 and HR 42 for routing tables and to monitor timers when using SYSMAC NET Systems.

3-8 TC (Timer/Counter) Area

The TC area is used to create and program timers and counters and holds the Completion flags, set values (SV), and present values (PV) for all timers and counters. All of these are accessed through TC numbers ranging from TC 000 through TC 511. Each TC number is defined as either a timer or counter using one of the following instructions: TIM, TIMH, CNT, CNTR(12), and TTIM(87). No prefix is required when using a TC number in a timer or counter instruction.

Once a TC number has been defined using one of these instructions, it cannot be redefined elsewhere in the program either using the same or a different instruction. If the same TC 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. There are no restrictions on the order in which TC numbers can be used.

Once defined, a TC number can be designated as an operand in one or more of certain set of instructions other than those listed above. When defined as a timer, a TC number designated as an operand takes a TIM prefix. The TIM prefix is used regardless of the timer instruction that was used to define the timer. Once defined as a counter, the TC number designated as an operand takes a CNT prefix. The CNT is also used regardless of the counter instruction that was used to define the counter.

TC 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 TC number accesses the completion flag of the timer or counter. When designated as an operand that requires word data, the TC number accesses a memory location that holds the PV of the timer or counter.

TC numbers are also used to access the SV of timers and counters from a Programming Device. The procedures for doing so using the Programming Console are provided in *7-1 Monitoring Operation and Modifying Data.*

The TC area retains the SVs of both timers and counters during power interruptions. The PVs of timers are reset when PC operation is begun and when reset in interlocked program sections. Refer to *5-10 INTERLOCK and INTERLOCK CLEAR – IL(02) and ILC(03)* for details on timer and counter operation in interlocked program sections. The PVs of counters are not reset at these times.

Note that in programming “TIM 000” is used to designate three things: the Timer instruction defined with TC number 000, the completion flag for this timer, and the PV of this timer. The meaning in context should be clear, i.e., the first is always an instruction, the second is always a bit, and the third is always a word. The same is true of all other TC numbers prefixed with TIM or CNT.

3-9 LR (Link Relay) Area

The LR area is used as a common data area to transfer information between PCs. This data transfer is achieved through a PC Link System.

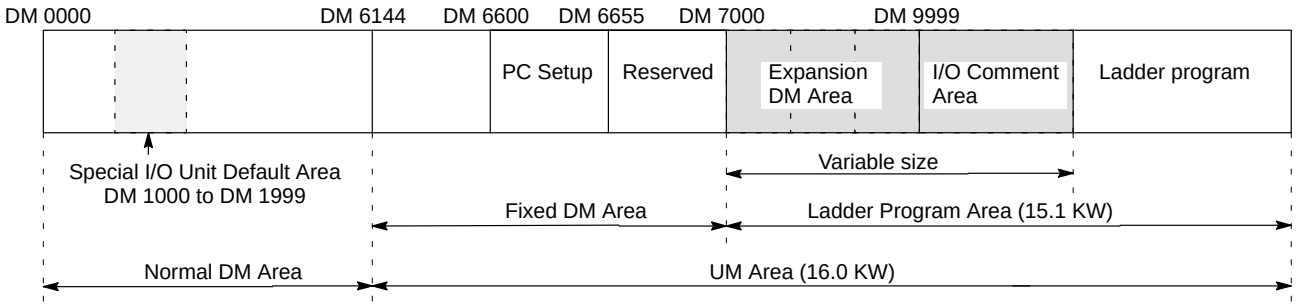
Certain words will be allocated as the write words of each PC. These words are written by the PC and automatically transferred to the same LR words in the other PCs in the System. The write words of the other PCs are transferred in as read words so that each PC can access the data written by the other PCs in the PC Link System. Only the write words allocated to the particular PC will be available for writing; all other words may be read only. Refer to the *PC Link System Manual* for details.

The LR area is accessible either by bit or by word. LR area word addresses range from LR 00 to LR 63; LR area bit addresses, from LR 0000 to LR 6315. Any part of the LR area that is not used by the PC Link System can be used as work words or work bits.

LR area data is not retained when the power is interrupted, when the PC is changed to PROGRAM mode, or when it is reset in an interlocked program section. Refer to *5-10 INTERLOCK and INTERLOCK CLEAR – IL(02) and ILC(03)* for details on interlocks.

3-10 UM Area

With the C200HS, the UM area is defined as the part of memory that can be converted and transferred to ROM. The UM area is 16 KW of RAM which is backed up by the CPU's battery. Some of the UM area is reserved to system use, so 15,488 words can be used by the operator. The structure of the C200HS DM and UM areas is shown in the following illustration.



Note Allocating UM area for an expansion DM and/or I/O Comment Area will reduce program capacity. Check program capacity requirements before allocating the UM area.

3-11 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. The use of TR bits is described in *Section 4 Writing and Inputting the Program*.

TR addresses range from TR 0 though TR 7. Each of these bits can be used as many times as required and in any order required as long as the same LR bit is not used twice in the same instruction block.

SECTION 4
Writing and Inputting the Program

This section explains the basic steps and concepts involved in writing a basic ladder diagram program, inputting the program into memory, and executing it. It introduces the instructions that are used to build the basic structure of the ladder diagram and control its execution. 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 F Word Assignment Recording Sheets* and *Appendix G 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 IR area or are allocated IR 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 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 TC numbers and jump numbers so that you can allocate these as you use them. Remember, the function of a TC number can be defined only once within the program; jump numbers 01 through 99 can be used only once each. (TC number are described in *5-14 Timer and Counter Instructions*; jump numbers are described later in this section.)
5. Draw the ladder diagram.
6. Input the program into the CPU. When using the Programming Console, this will involve converting the program to 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.
10. Make a backup copy of the program.

The basics of ladder-diagram programming and conversion to mnemonic code are described in *4-4 Basic Ladder Diagrams*. Preparing for and inputting the program via the Programming Console are described in *4-5 The Programming Console* through *4-7 Inputting, Modifying, and Checking the Program*. The rest of Section 4 covers more advanced programming, programming precautions, and program execution. All special application instructions are covered in *Section 5 Instruction Set*. Debugging is described in *Section 7 Program Monitoring and Execution*. *Section 10 Troubleshooting* also provides information required for debugging.

4-2 Instruction Terminology

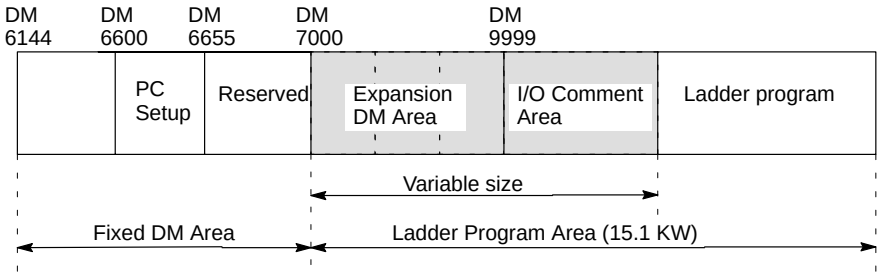
There are basically two types of instructions used in ladder-diagram programming: instructions that correspond to the conditions on the ladder diagram and are used in instruction form only when converting a program to mnemonic code and instructions that are used on the right side of the ladder diagram 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 to be 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 IR 000 designated as the source operand will move the contents of IR 000 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 actual value is entered as a constant, it is preceded by # to indicate that it is not an address.

Other terms used in describing instructions are introduced in *Section 5 Instruction Set*.

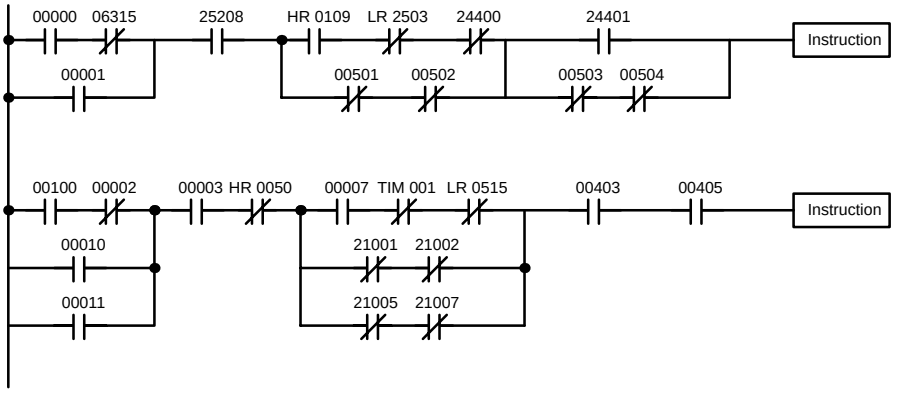
4-3 Program Capacity

The maximum user program size varies with the amount of UM allocated to expansion DM and the I/O Comment Area. Approximately 10.1 KW are available for the ladder program when 3 KW are allocated to expansion DM and 2 KW are allocated to I/O comments as shown below. Refer to the *3-10 UM Area* for further information on UM allocation.



4-4 Basic Ladder Diagrams

A ladder diagram consists of one line running down the left side with lines branching off to the right. The line on the left is called the bus bar; the branching lines, instruction lines or rungs. Along the instruction lines are placed conditions that lead to other instructions on the right side. The logical combinations of these conditions determine when and how the instructions at the right are executed. A ladder diagram is shown below.



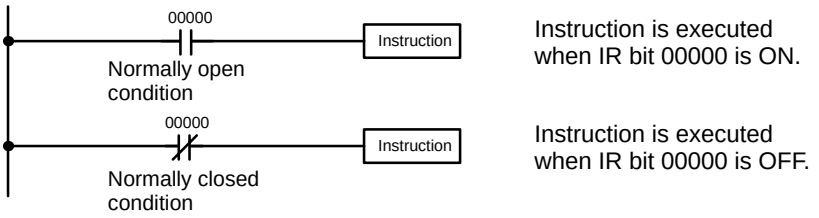
As shown in the diagram above, instruction lines can branch apart and they can join back together. The vertical pairs of lines are called conditions. Conditions without diagonal lines through them are called normally open conditions and correspond to a LOAD, AND, or OR instruction. The conditions with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

Note When displaying ladder diagrams with LSS, a second bus bar will be shown on the right side of the ladder diagram and will be connected to all instructions on the right side. This does not change the ladder-diagram program in any functional sense. No conditions can be placed between the instructions on the right side and the right bus bar, i.e., all instructions on the right must be connected directly to the right bus bar. Refer to the *LSS Operation Manual* for details.

4-4-1 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.

Operand Bits

The operands designated for any of the ladder instructions can be any bit in the IR, SR, HR, AR, LR, or TC 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 *4-7-7 Branching Instruction Lines* for details.

Logic Blocks

The way that conditions correspond to what instructions is determined by the relationship between the conditions within the instruction lines that connect them. Any group of conditions that go together to create a logic result 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.

4-4-2 Mnemonic Code

The ladder diagram cannot be directly input into the PC via a Programming Console; LSS is required. To input from a Programming Console, it is necessary to convert the ladder diagram to mnemonic code. The mnemonic code provides exactly the same information as the ladder diagram, but in a form that can be typed directly into the PC. Actually you can program directly in mnemonic code, although it in not recommended for beginners or for complex programs. Also, regardless of the Programming Device used, the program is stored in memory in mnemonic form, making it important to understand mnemonic code.

Because of the importance of the Programming Console as a peripheral device and because of the importance of mnemonic code in complete understanding of a program, we will introduce and describe the mnemonic code along with the ladder diagram. Remember, you will not need to use the mnemonic code if you are inputting via LSS (although you can use it with LSS too, if you prefer).

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. Because some instructions require no operands, while others require up to three operands, Program Memory addresses can be from one to four words long.

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 also contained in the first word. Also, if an instruction requires only a single bit operand (with no definer), the bit operand is also programmed 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. When converting to mnemonic code, all but ladder diagram instructions are written in the same form, one word to a line, just as they appear in the ladder diagram symbols. An example of mnemonic code is shown below. The instructions used in it are described later in the manual.

Address	Instruction	Operands	
00000	LD	HR	0001
00001	AND		00001
00002	OR		00002
00003	LD NOT		00100
00004	AND		00101
00005	AND LD		00102
00006	MOV(21)		
			000
		DM	0000
00007	CMP(20)		
		DM	0000
		HR	00
00008	LD		25505
00009	OUT		00501
00010	MOV(21)		
		DM	0000
		DM	0500
00011	DIFU(13)		00502
00012	AND		00005
00013	OUT		00503

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.

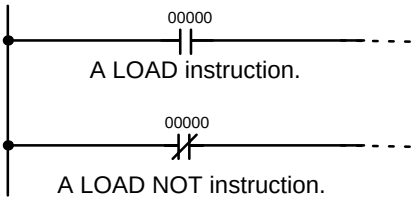
When programming, addresses are automatically displayed and do not have to be input unless for some reason a different location is desired for the instruction. When converting to mnemonic code, it is best to start at Program Memory address 00000 unless there is a specific reason for starting elsewhere.

4-4-3 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 next, 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 instruction requires 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.

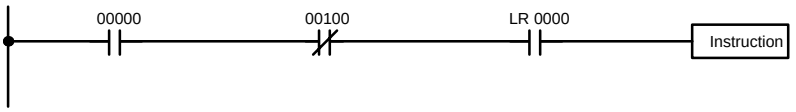


Address	Instruction	Operands
00000	LD	00000
00001	Instruction	
00002	LD NOT	00000
00003	Instruction	

When this is the only condition on the instruction line, the execution condition for the instruction at the right is ON when the condition is ON. For the LOAD instruction (i.e., a normally open condition), the execution condition will be ON when IR 00000 is ON; for the LOAD NOT instruction (i.e., a normally closed condition), it will be ON when 00000 is 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 requires one line of mnemonic code.



Address	Instruction	Operands
00000	LD	00000
00001	AND NOT	00100
00002	AND	LR 0000
00003	Instruction	

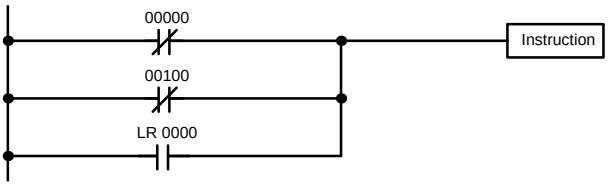
The instruction will have an ON execution condition only when all three conditions are ON, i.e., when IR 00000 is ON, IR 00100 is OFF, and LR 0000 is ON.

AND instructions in series can be considered individually, with each taking the logical AND of the execution condition (i.e., the total of all conditions up to that point) 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 result will also be OFF. The execution condition for the first AND instruction in a series is the first condition on the instruction line.

Each AND NOT instruction in series takes the logical AND of its execution condition and the inverse of its operand bit.

OR and OR NOT

When two or more conditions lie on separate instruction lines which run in parallel and then join together, the first condition corresponds to a LOAD or LOAD NOT instruction; the other 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.



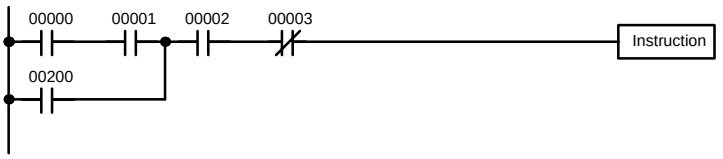
Address	Instruction	Operands
00000	LD	00000
00001	OR NOT	00100
00002	OR	LR 0000
00003	Instruction	

The instruction will have an ON execution condition when any one of the three conditions is ON, i.e., when IR 00000 is OFF, when IR 00100 is OFF, or when LR 0000 is ON.

OR and OR NOT instructions can be considered individually, each taking the logical OR between its execution condition and the status of the OR instruction's operand bit. If either one of these were ON, an ON execution condition will 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 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.



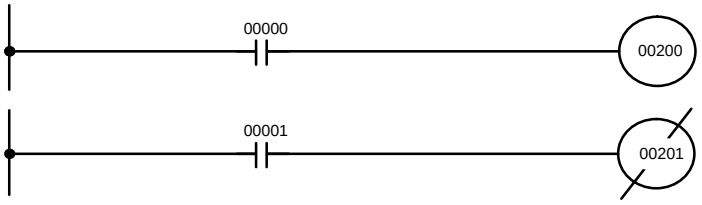
Address	Instruction	Operands
00000	LD	00000
00001	AND	00001
00002	OR	00200
00003	AND	00002
00004	AND NOT	00003
00005	Instruction	

Here, an AND is taken between the status of IR 00000 and that of IR 00001 to determine the execution condition for an OR with the status of IR 00200. The result of this operation determines the execution condition for an AND with the status of IR 00002, which in turn determines the execution condition for an AND with the inverse (i.e., and AND NOT) of the status of IR 00003.

In more complicated diagrams, however, 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.

4-4-4 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	00000
00001	OUT	00200

Address	Instruction	Operands
00000	LD	00001
00001	OUT NOT	00201

In the above examples, IR 00200 will be ON as long as IR 00000 is ON and IR 00201 will be OFF as long as IR 00001 is ON. Here, IR 00000 and IR 00001 will be input bits and IR 00200 and IR 00201 output bits assigned to the Units controlled by the PC, i.e., the signals coming in through the input points assigned IR 00000 and IR 00001 are controlling the output points assigned IR 00200 and IR 00201, respectively.

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 *5-14-1 TIMER – TIM* for details.

4-4-5 The END Instruction

The last instruction required to complete a simple program is the END instruction. When the CPU cycles the program, it executes all instruction 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 until it is removed. The number following the END instruction in the mnemonic code is its function code, which is used when inputted most instruction into the PC. These are described later. The END instruction requires no operands and no conditions can be placed on the same instruction line with it.

Address	Instruction	Operands
00000	LD	00000
00001	AND NOT	00001
00002	Instruction	
00003	END(01)	---

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

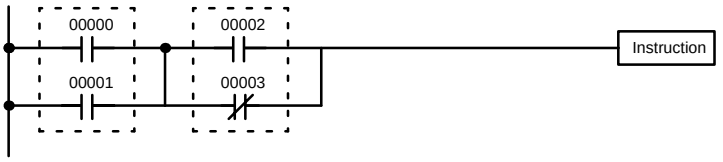
Now you have all of the instructions required to write simple input-output programs. Before we finish with ladder diagram basic and go onto inputting the program into the PC, let's look at logic block instruction (AND LOAD and OR LOAD), which are sometimes necessary even with simple diagrams.

4-4-6 Logic Block Instructions

Logic block instructions do not correspond to specific conditions on the ladder diagram; rather, they describe relationships between logic blocks. The AND LOAD instruction logically ANDs the execution conditions produced by two logic blocks. 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	00000
00001	OR	00001
00002	LD	00002
00003	OR NOT	00003
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 IR 00000 or IR 00001 is ON), **and** when either of the conditions in the right logic block is ON (i.e., when either IR 00002 is ON or IR 00003 is OFF).

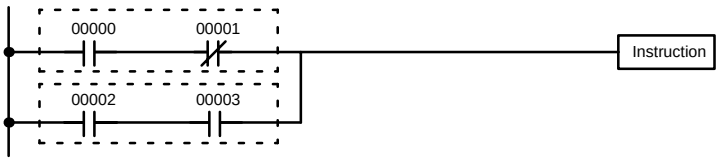
The above ladder diagram cannot, however, be converted to mnemonic code using AND and OR instructions alone. If an AND between IR 00002 and the results of an OR between IR 00000 and IR 00001 is attempted, the OR NOT between IR 00002 and IR 00003 is lost and the OR NOT ends up being an OR NOT between just IR 00003 and the result of an AND between IR 00002 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 a special buffer and the logic process is restarted. 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 IR 00000 is a LOAD instruction and the condition below it is an OR instruction between the status of IR 00000 and that of IR 00001. The condition at IR 00002 is another LOAD instruction and the condition below is an OR NOT instruction, i.e., an OR between the status of IR 00002 and the inverse of the status of IR 00003. To arrive at the execution condition for the instruction at the right, the logical AND of the execution conditions resulting from these two blocks will have to be taken. AND LOAD does this. The mnemonic code for the ladder diagram is shown below. 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 will be produced for the instruction at the right either when IR 00000 is ON and IR 00001 is OFF, or when IR 00002 and IR 00003 are both ON. The operation of the OR LOAD instruction and its mnemonic code is exactly the same as that for an AND LOAD instruction, except that the current execution condition is **ORed** with the last unused execution condition.



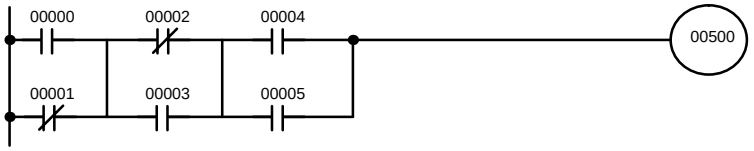
Address	Instruction	Operands
00000	LD	00000
00001	AND NOT	00001
00002	LD	00002
00003	AND	00003
00004	OR LD	---

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 options for coding the programs are also shown.



Address	Instruction	Operands
00000	LD	00000
00001	OR NOT	00001
00002	LD NOT	00002
00003	OR	00003
00004	AND LD	—
00005	LD	00004
00006	OR	00005
00007	AND LD	—
00008	OUT	00500

Address	Instruction	Operands
00000	LD	00000
00001	OR NOT	00001
00002	LD NOT	00002
00003	OR	00003
00004	LD	00004
00005	OR	00005
00006	AND LD	—
00007	AND LD	—
00008	OUT	00500

Again, with the method on the right, 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 series conditions lie in parallel to each other.

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 codes for both methods are shown below.

Address	Instruction	Operands
00000	LD	00000
00001	AND NOT	00001
00002	LD NOT	00002
00003	AND NOT	00003
00004	OR LD	—
00005	LD	00004
00006	AND	00005
00007	OR LD	—
00008	OUT	00501

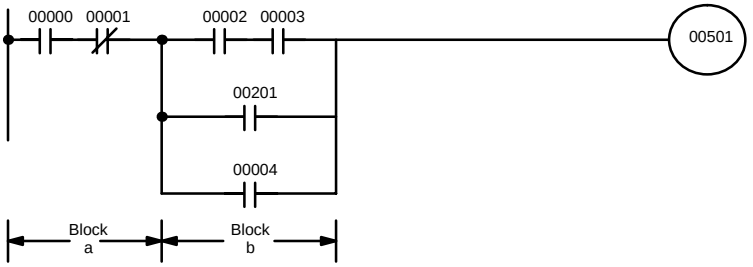
Address	Instruction	Operands
00000	LD	00000
00001	AND NOT	00001
00002	LD NOT	00002
00003	AND NOT	00003
00004	LD	00004
00005	AND	00005
00006	OR LD	—
00007	OR LD	—
00008	OUT	00501

Again, with the method on the right, 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.

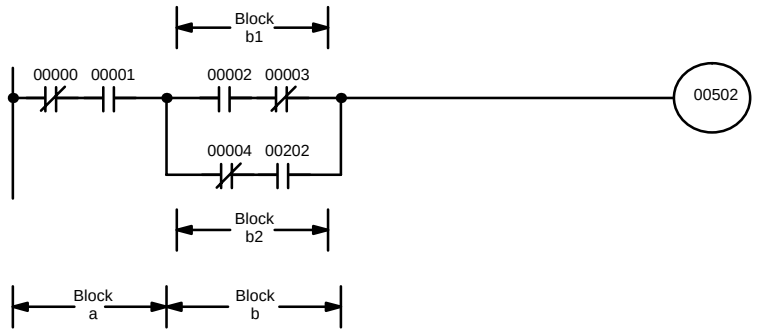
The following diagram contains only two logic blocks as shown. It is not necessary to further separate block b components, because it can be coded directly using only AND and OR.



Address	Instruction	Operands
00000	LD	00000
00001	AND NOT	00001
00002	LD	00002
00003	AND	00003
00004	OR	00201
00005	OR	00004
00006	AND LD	—
00007	OUT	00501

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.



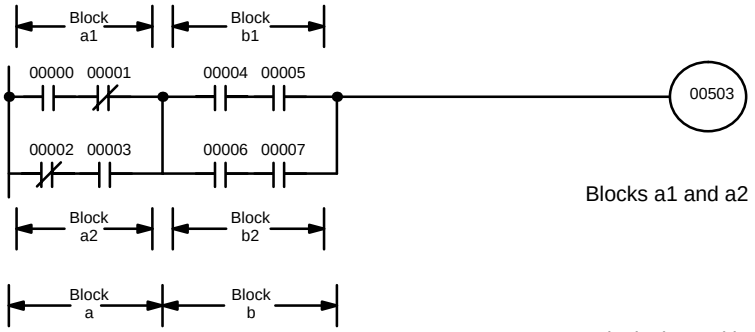
Address	Instruction	Operands
00000	LD NOT	00000
00001	AND	00001
00002	LD	00002
00003	AND NOT	00003
00004	LD NOT	00004
00005	AND	00202
00006	OR LD	—
00007	AND LD	—
00008	OUT	00502

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.



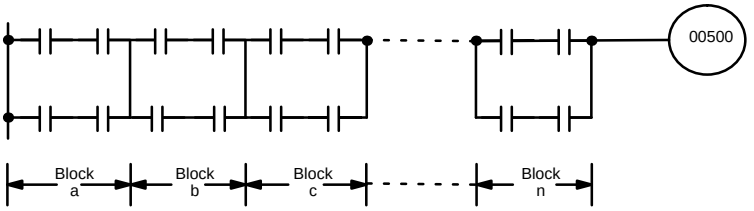
Blocks a1 and a2

Blocks b1 and b2

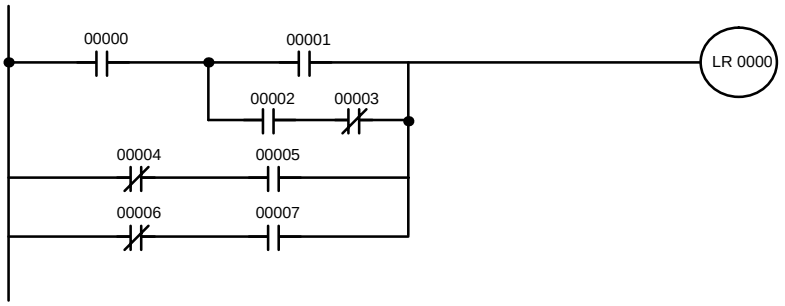
Blocks a and b

Address	Instruction	Operands
00000	LD	00000
00001	AND NOT	00001
00002	LD NOT	00002
00003	AND	00003
00004	OR LD	—
00005	LD	00004
00006	AND	00005
00007	LD	00006
00008	AND	00007
00009	OR LD	—
00010	AND LD	—
00011	OUT	00503

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.

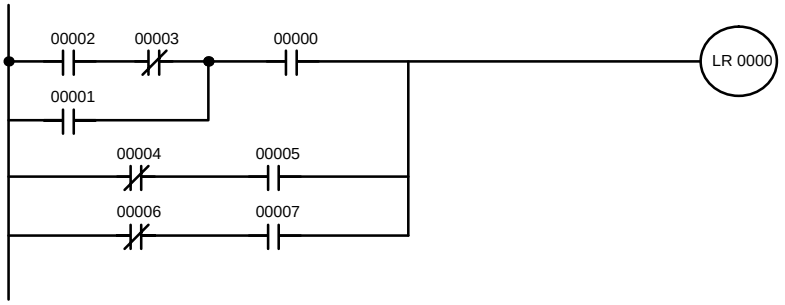


The following diagram requires an OR LOAD followed by an AND LOAD to code the top of the three blocks, and then two more OR LOADs to complete the mnemonic code.



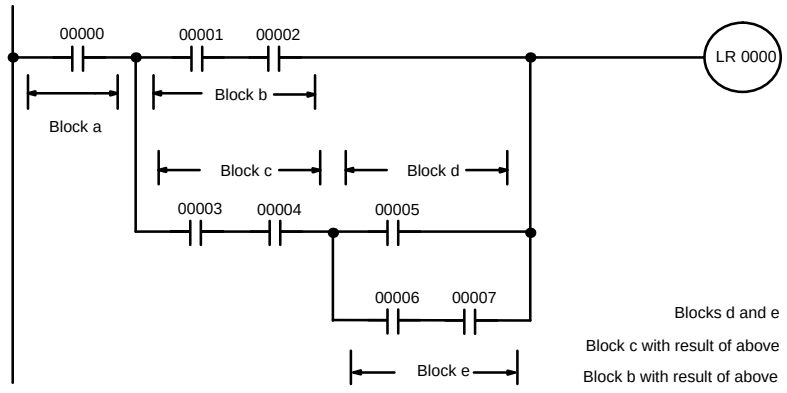
Address	Instruction	Operands
00000	LD	00000
00001	LD	00001
00002	LD	00002
00003	AND NOT	00003
00004	OR LD	—
00005	AND LD	—
00006	LD NOT	00004
00007	AND	00005
00008	OR LD	—
00009	LD NOT	00006
00010	AND	00007
00011	OR LD	—
00012	OUT	LR 0000

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.



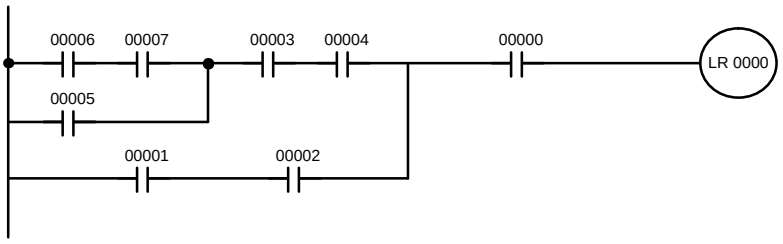
Address	Instruction	Operands
00000	LD	00002
00001	AND NOT	00003
00002	OR	00001
00003	AND	00000
00004	LD NOT	00004
00005	AND	00005
00006	OR LD	—
00007	LD NOT	00006
00008	AND	00007
00009	OR LD	—
00010	OUT	LR 0000

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.



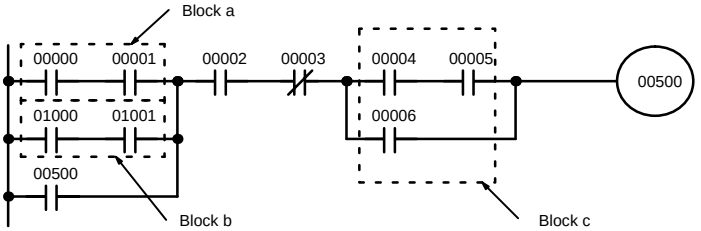
Address	Instruction	Operands
00000	LD	00000
00001	LD	00001
00002	AND	00002
00003	LD	00003
00004	AND	00004
00005	LD	00005
00006	LD	00006
00007	AND	00007
00008	OR LD	—
00009	AND LD	—
00010	OR LD	—
00011	AND LD	—
00012	OUT	LR 0000

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

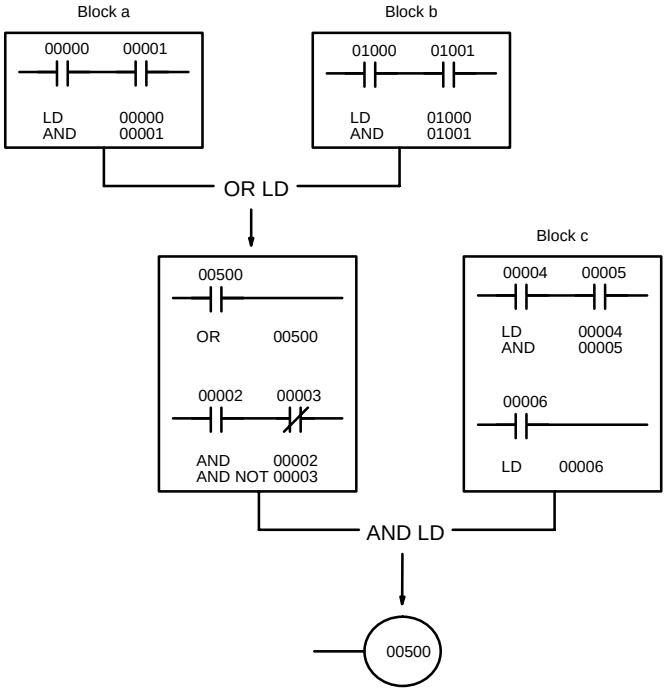


Address	Instruction	Operands
00000	LD	00006
00001	AND	00007
00002	OR	00005
00003	AND	00003
00004	AND	00004
00005	LD	00001
00006	AND	00002
00007	OR LD	—
00008	AND	00000
00009	OUT	LR 0000

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:



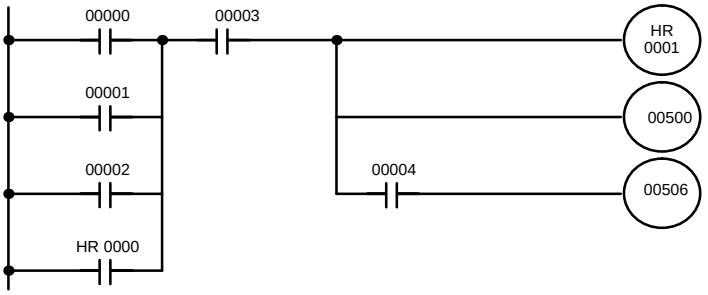
The first logic block instruction is used to combine the execution conditions resulting from blocks a and b, and the second one is to combine the execution condition of block c with the execution condition resulting from the normally closed condition assigned IR 00003. The rest of the diagram can be 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	00000
00001	AND	00001
00002	LD	01000
00003	AND	01001
00004	OR LD	—
00005	OR	00500
00006	AND	00002
00007	AND NOT	00003
00008	LD	00004
00009	AND	00005
00010	OR	00006
00011	AND LD	—
00012	OUT	00500

4-4-7 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 IR 00004.



Address	Instruction	Operands
00000	LD	00000
00001	OR	00001
00002	OR	00002
00003	OR	HR 0000
00004	AND	00003
00005	OUT	HR 0001
00006	OUT	00500
00007	AND	00004
00008	OUT	00506

4-5 The Programming Console

This and the next section describe the Programming Console and the operations necessary to prepare for program input. *4-7 Inputting, Modifying, and Checking the Program* describes actual procedures for inputting the program into memory.

Although the Programming Console can be used to write ladder programs, it is primarily used to support LSS operations and is very useful for on-site editing and maintenance. The main Programming Console functions are listed below.

1, 2, 3...

1. Displaying operating messages and the results of diagnostic checks.
2. Writing and reading ladder programs, inserting and deleting instructions, searching for data or instructions, and monitoring I/O bit status.
3. Monitoring I/O status, force-setting/resetting bits.
4. The Programming Console can be connected to or disconnected from the PC with the power on.
5. The Programming Console can be used with C-series PCs.
6. Supports TERMINAL mode, which allows the display of a 32-character message, as well as operation of the keyboard mapping function. Refer to *5-25-6 TERMINAL MODE – TERM(—)* for details.

Note The Programming Console does not support all of the LSS operations, only those required for on-site editing and maintenance.

4-5-1 The Keyboard

The keyboard of the Programming Console is functionally divided by key color into the following four areas:

White: Numeric Keys The ten white keys are used to input numeric program data such as program addresses, data area addresses, and operand values. The numeric keys are also used in combination with the function key (FUN) to enter instructions with function codes.

Red: CLR Key The CLR key clears the display and cancels current Programming Console operations. It is also used when you key in the password at the beginning of programming operations. Any Programming Console operation can be cancelled by pressing the CLR key, although the CLR key may have to be pressed two or three times to cancel the operation and clear the display.

Yellow: Operation Keys The yellow keys are used for writing and correcting programs. Detailed explanations of their functions are given later in this section.

Gray: Instruction and Data Area Keys

Except for the SHIFT key on the upper right, the gray keys are used to input instructions and designate data area prefixes when inputting or changing a program. The SHIFT key is similar to the shift key of a typewriter, and is used to alter the function of the next key pressed. (It is not necessary to hold the SHIFT key down; just press it once and then press the key to be used with it.)

The gray keys other than the SHIFT key have either the mnemonic name of the instruction or the abbreviation of the data area written on them. The functions of these keys are described below.

FUN	Pressed before the function code when inputting an instruction via its function code.
SFT	Pressed to enter SFT (the Shift Register instruction).
NOT	Input either after a function code to designate the differentiated form of an instruction or after a ladder instruction to designate an inverse condition.
AND ⊣⊢	Pressed to enter AND (the AND instruction) or used with NOT to enter AND NOT.
OR ⊣⊢	Pressed to enter OR (the OR instruction) or used with NOT to enter OR NOT.
CNT	Pressed to enter CNT (the Counter instruction) or to designate a TC number that has already been defined as a counter.
LD ⊢⊢	Pressed to enter LD (the Load instruction) or used with NOT to enter LD NOT. Also pressed to indicate an input bit.
OUT —○⊢	Pressed to enter OUT (the Output instruction) or used with NOT to enter OUT NOT. Also pressed to indicate an output bit.
TIM	Pressed to enter TIM (the Timer instruction) or to designate a TC number that has already been defined as a timer.
TR	Pressed before designating an address in the TR area.
LR	Pressed before designating an address in the LR area.
HR	Pressed before designating an address in the HR area.
SHIFT + HR	Pressed before designating an address in the AR area.
DM	Pressed before designating an address in the DM area.
CH / *	Pressed before designating an indirect DM address.
SHIFT + CH / *	Pressed before designating a word address.
CONT / #	Pressed before designating an operand as a constant.
SHIFT + CONT / #	Pressed before designating a bit address.
SHIFT	Pressed before function codes for block programming instructions, i.e., those placed between pointed parentheses <>.

4-5-2 PC Modes

The Programming Console is equipped with a switch to control the PC mode. To select one of the three operating modes—RUN, MONITOR, or PROGRAM—use the mode switch. The mode that you select will determine PC operation as well as the procedures that are possible from the Programming Console.

RUN mode is the mode used for normal program execution. When the switch is set to RUN and the START input on the CPU Power Supply Unit is ON, the CPU will begin executing the program according to the program written in its Program Memory. Although monitoring PC operation from the Programming Console is possible in RUN mode, no data in any of the memory areas can be input or changed.

MONITOR mode allows you to visually monitor in-progress program execution while controlling I/O status, changing PV (present values) or SV (set values), etc. In MONITOR mode, I/O processing is handled in the same way as in RUN mode. MONITOR mode is generally used for trial system operation and final program adjustments.

In PROGRAM mode, the PC does not execute the program. PROGRAM mode is for creating and changing programs, clearing memory areas, and registering and changing the I/O table. A special Debug operation is also available within PROGRAM mode that enables checking a program for correct execution before trial operation of the system.

⚠ DANGER Do not leave the Programming Console connected to the PC by an extension cable when in RUN mode. Noise picked up by the extension cable can enter the PC, affecting the program and thus the controlled system.

4-5-3 The Display Message Switch

Pin 3 of the CPU's DIP switch determines whether Japanese or English language messages will be displayed on the Programming Console. It is factory set to ON, which causes English language messages to be displayed.

4-6 Preparation for Operation

This section describes the procedures required to begin Programming Console operation. These include password entry, clearing memory, error message clearing, and I/O table operations. I/O table operations are also necessary at other times, e.g., when changes are to be made in Units used in the PC configuration.

⚠ DANGER Always confirm that the Programming Console is in PROGRAM mode when turning on the PC with a Programming Console connected unless another mode is desired for a specific purpose. If the Programming Console is in RUN mode when PC power is turned on, any program in Program Memory will be executed, possibly causing a PC-controlled system to begin operation.

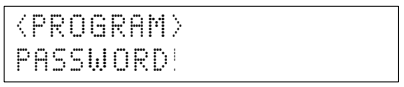
The following sequence of operations must be performed before beginning initial program input.

1, 2, 3...

1. Insert the mode key into the Programming Console.
2. Set the mode switch to PROGRAM mode. (The mode key cannot be removed while set to PROGRAM mode.)
3. Turn on PC power.

Note When I/O Units are installed, turn on those Units also. The Programming Console will not operate if these Units are not turned on.

4. Confirm that the CPU's POWER LED is lit and the following display appears on the Programming Console screen. (If the ALM/ERR LED is lit or flashing or an error message is displayed, clear the error that has occurred.)



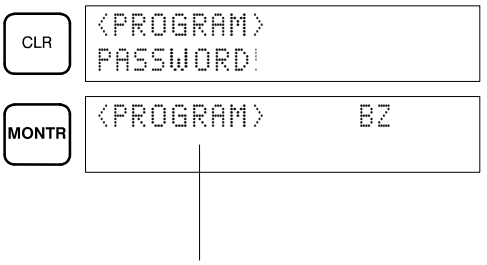
5. Enter the password. See *4-6-1 Entering the Password* for details.
6. Clear memory. Skip this step if the program does not need to be cleared. See *4-6-3 Clearing Memory* for details.

4-6-1 Entering the Password

To gain access to the PC's programming functions, you must first enter the password. The password prevents unauthorized access to the program.

The PC prompts you for a password when PC power is turned on or, if PC power is already on, after the Programming Console has been connected to the PC. To gain access to the system when the "Password!" message appears, press CLR and then MONTR. Then press CLR to clear the display.

If the Programming Console is connected to the PC when PC power is already on, the first display below will indicate the mode the PC was in before the Programming Console was connected. **Ensure that the PC is in PROGRAM mode before you enter the password.** When the password is entered, the PC will shift to the mode set on the mode switch, causing PC operation to begin if the mode is set to RUN or MONITOR. The mode can be changed to RUN or MONITOR with the mode switch after entering the password.



Indicates the mode set by the mode selector switch.

4-6-2 Buzzer

Immediately after the password is input or anytime immediately after the mode has been changed, SHIFT and then the 1 key can be pressed to turn on and off the buzzer that sounds when Programming Console keys are pressed. If BZ is displayed in the upper right corner, the buzzer is operative. If BZ is not displayed, the buzzer is not operative.

This buzzer also will also sound whenever an error occurs during PC operation. Buzzer operation for errors is not affected by the above setting.

4-6-3 Clearing Memory

Using the Memory Clear operation it is possible to clear all or part of the UM area (RAM or EEPROM), and the IR, HR, AR, DM and TC areas. Unless otherwise specified, the clear operation will clear all of the above memory areas. The UM area will not be cleared if the write-protect switch (pin 1 of the CPU's DIP switch) is set to ON.

Before beginning to programming for the first time or when installing a new program, all areas should normally be cleared. Before clearing memory, check to see if a program is already loaded that you need. If you need the program, clear only the memory areas that you do not need, and be sure to check the existing program with the program check key sequence before using it. The check sequence is provided later in this section. Further debugging methods are provided in *Section 7 Program Monitoring and Execution*. To clear all memory areas press CLR until all zeros are displayed, and then input the keystrokes given in the top line of the following key sequence. The branch lines shown in the sequence are used only when performing a partial memory clear, which is described below.

Memory can be cleared in PROGRAM mode only. The following table shows which memory areas will be cleared for the 3 memory clearing operations (all clear, partial clear, memory clear).

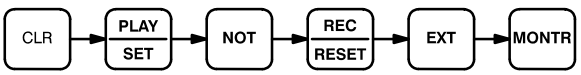
Memory Area	All clear	Partial clear	Memory clear
I/O words	Cleared	Cleared	Cleared
Work words	Cleared	---	Cleared
HR, AR, TC, DM, fixed DM	Cleared	Cleared	Cleared
Expansion DM	Cleared	---	Cleared
I/O comments	Cleared	---	---
Ladder program	Cleared	Cleared	Cleared
UM Allocation information	Cleared	---	---

Note

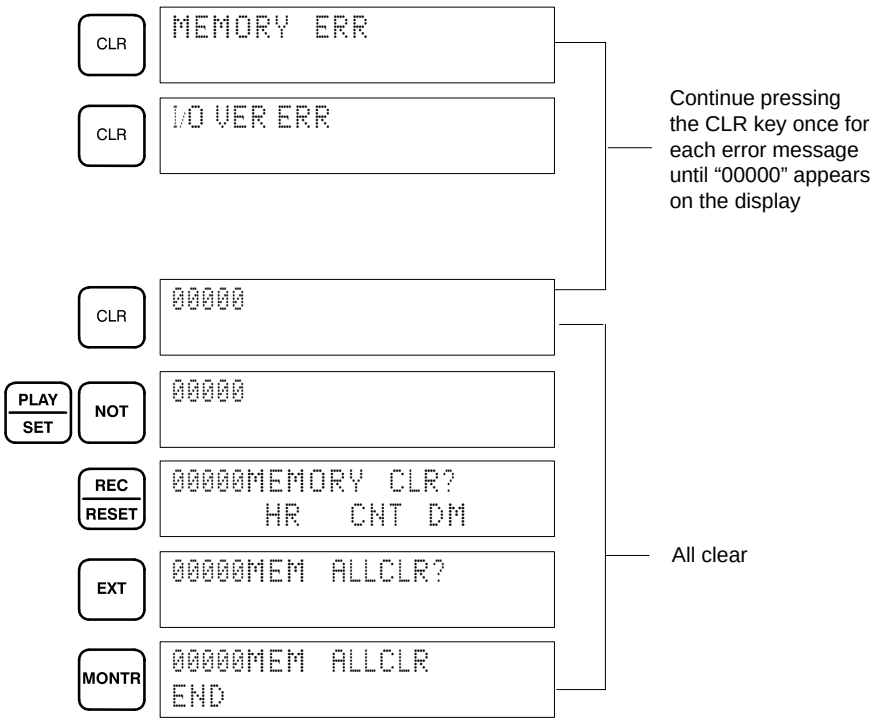
1. The error history area (DM 6000 to DM 6030) will not be cleared when the DM area is cleared.
2. When the PC Setup area (DM 6600 to DM 6655 in fixed DM) is cleared, the settings will be returned to their factory-set defaults.
3. When the All Clear operation is executed, the ladder program area will be allocated entirely to the ladder program. (The expansion DM and I/O comment areas will be set to 0 KW.)

All Clear

The key sequence for all clear is shown below.



The following procedure is used to clear memory completely.

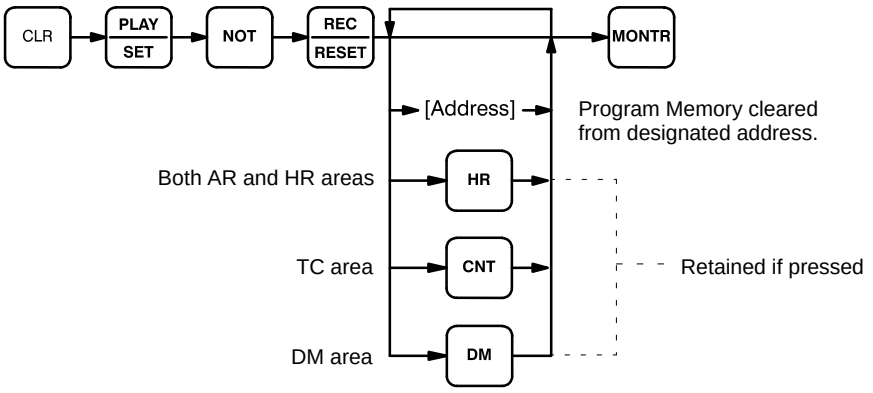


Partial Clear

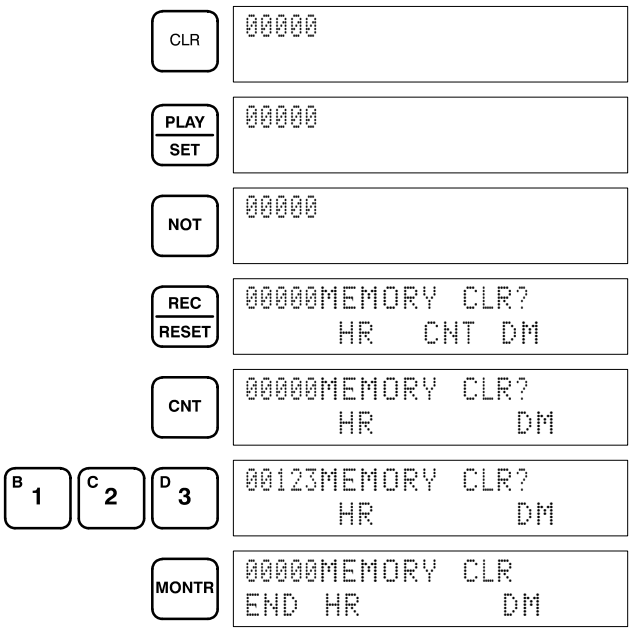
It is possible to retain the data in specified areas or part of the ladder program. To retain the data in the HR and AR, TC, and/or DM areas, press the appropriate key after entering REC/RESET. HR is pressed to designate both the HR and AR areas. In other words, specifying that HR is to be retained will ensure that AR is retained also. If not specified for retention, both areas will be cleared. CNT is used for the entire TC area. The display will show those areas that will be cleared.

It is also possible to retain a portion of the ladder program from the beginning to a specified address. After designating the data areas to be retained, specify the first program address to be cleared. For example, to leave addresses 00000 to 00122 untouched, but to clear addresses from 00123 to the end of Program Memory, input 00123.

The key sequence for a partial memory clear is shown below.



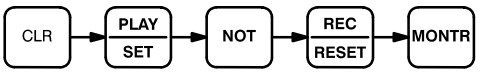
To leave the TC area uncleared and retain Program Memory addresses 00000 through 00122, input as follows:



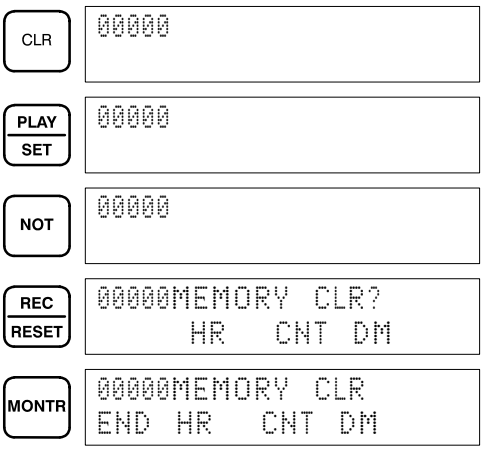
Memory Clear

The memory clear operation clears all memory areas except the I/O comments and UM Allocation information.

The key sequence for a partial memory clear is shown below.



The Programming Console will display the following screens:



Note When the write-protect switch (pin 1 of the CPU's DIP switch) is set to ON the UM area (from DM 6144 through the ladder program) will not be cleared. Other data areas, such as HR, AR, CNT, and DM from DM 0000 to DM 6143 will be cleared.

4-6-4 Registering the I/O Table

The I/O Table Registration operation records the types of I/O Units controlled by the PC and the Rack locations of the I/O Units. It also clears all I/O bits.

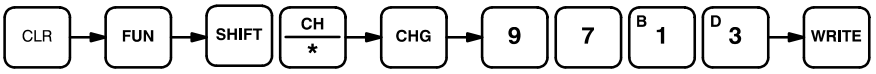
It is not absolutely necessary to register the I/O table with the C200HS. When the I/O table has not been registered, the PC will operate according to the I/O Units mounted when power is applied. The I/O verification/setting error will not occur.

It is necessary to register the I/O table if I/O Units are changed, otherwise an I/O verification error message, “I/O VER ERR” or “I/O SET ERROR”, will appear when starting programming operations.

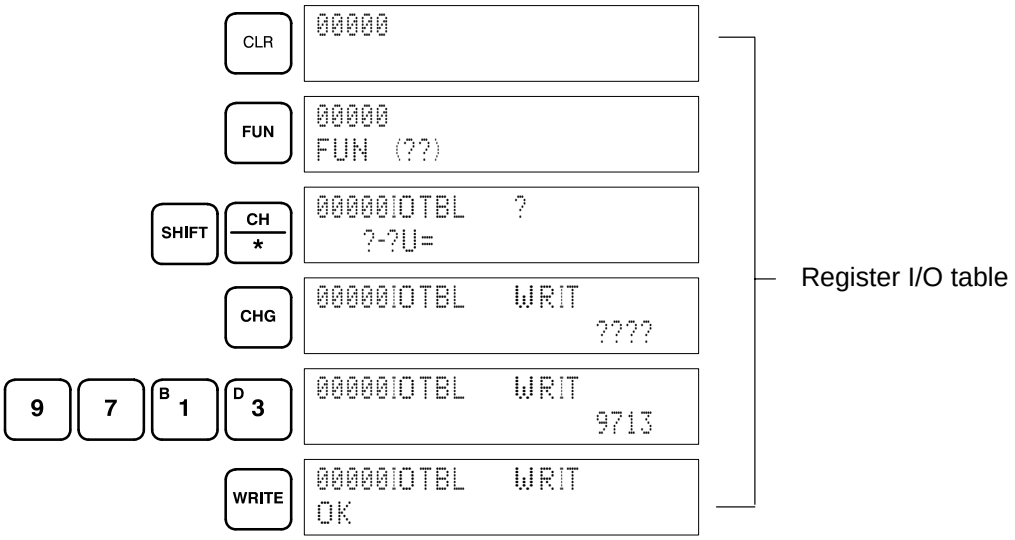
I/O Table Registration can be performed only in PROGRAM mode with the write-protection switch (pin 1 of the CPU’s DIP switch) set to OFF (OFF=“WRITE”).

Group-2 HIgh-density I/O Units will not be displayed in the I/O table when it is displayed using a host computer. Four asterisks (∗∗∗∗), indicating no Unit, will be displayed instead.

Key Sequence



Initial I/O Table Registration



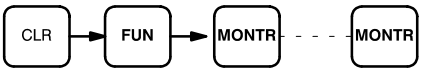
4-6-5 Clearing Error Messages

After the I/O table has been registered, any error messages recorded in memory should be cleared. It is assumed here that the causes of any of the errors for which error messages appear have already been taken care of. If the beeper sounds when an attempt is made to clear an error message, eliminate the cause of the error, and then clear the error message (refer to *Section 10 Troubleshooting*).

To display any recorded error messages, press CLR, FUN, and then MONTR. The first message will appear. Pressing MONTR again will clear the present message and display the next error message. Continue pressing MONTR until all messages have been cleared.

Although error messages can be accessed in any mode, they can be cleared only in PROGRAM mode.

Key Sequence

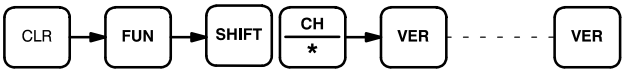


4-6-6 Verifying the I/O Table

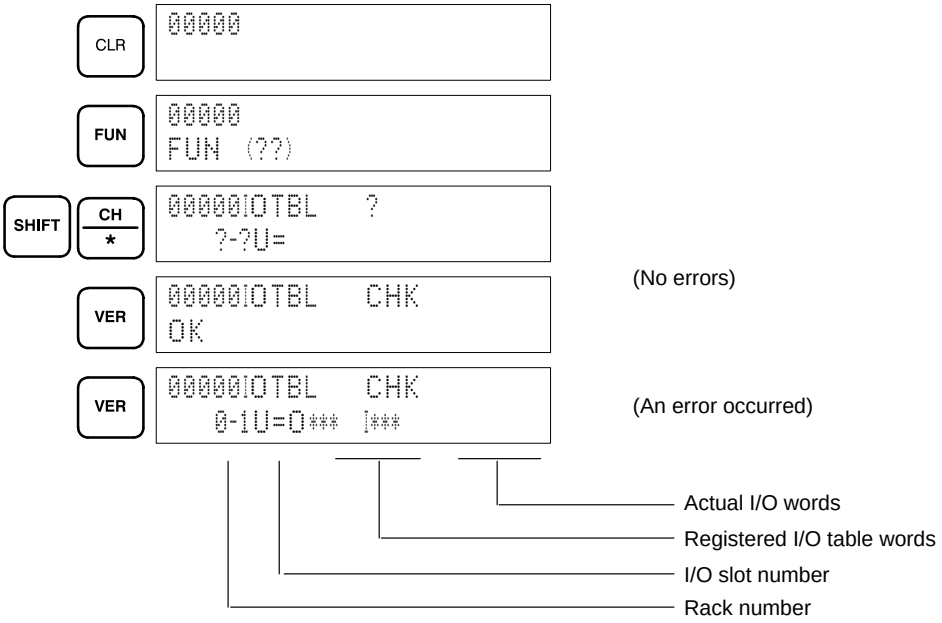
The I/O Table Verification operation is used to check the I/O table registered in memory to see if it matches the actual sequence of I/O Units mounted. The first inconsistency discovered will be displayed as shown below. Every subsequent pressing of **VER** displays the next inconsistency.

Note This operation can be executed only when the I/O table has been registered.

Key Sequence

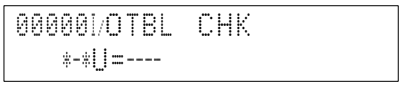


Example

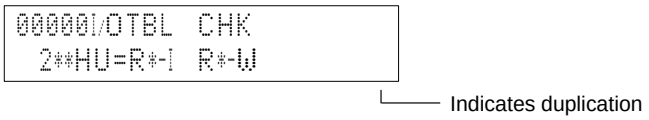


Meaning of Displays

The following display indicates a C500, C1000H, or C2000H and C200H or C200HS have the same unit number on a Remote I/O Slave Rack.



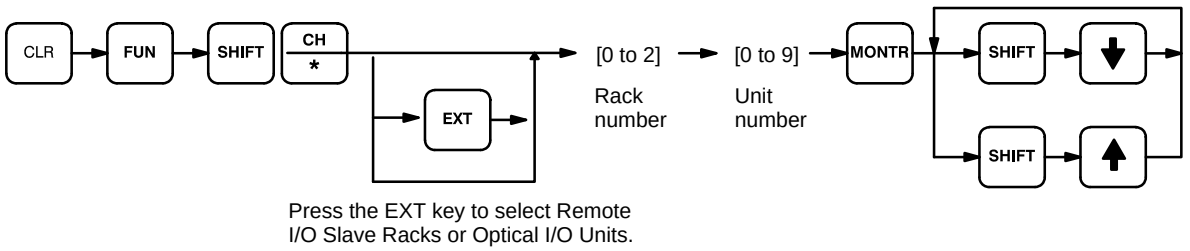
The following display indicates a duplication in Optical I/O Unit unit numbers.



4-6-7 Reading the I/O Table

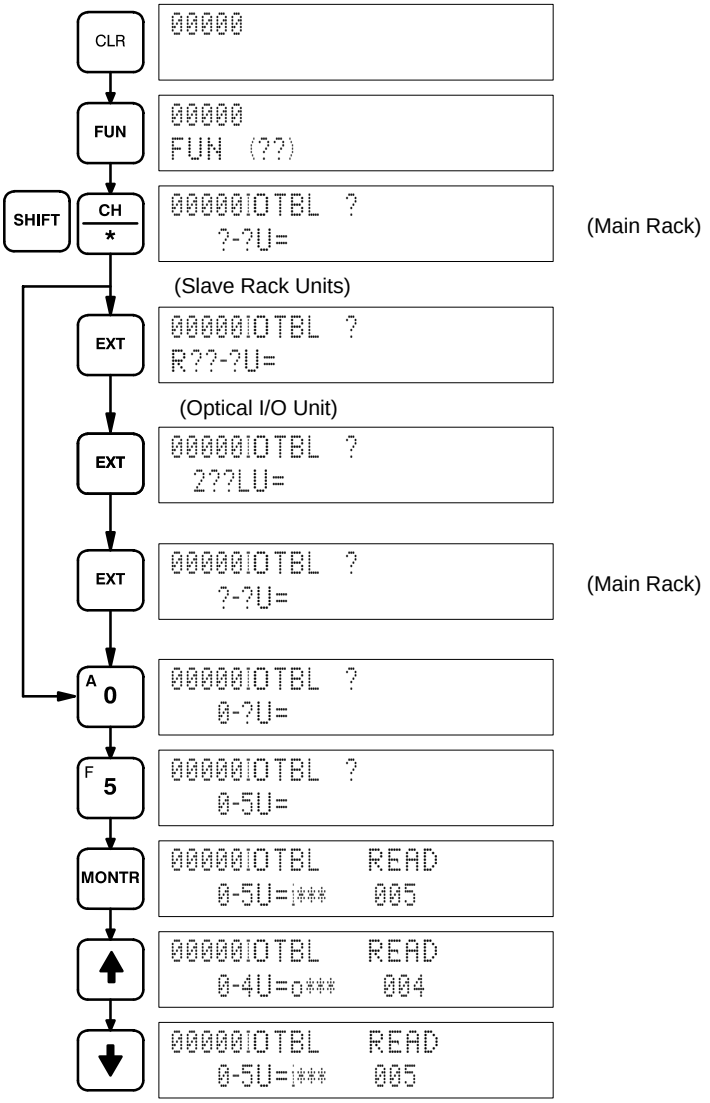
The I/O Table Read operation is used to access the I/O table that is currently registered in the CPU memory. This operation can be performed in any PC mode.

Key Sequence



Press the EXT key to select Remote I/O Slave Racks or Optical I/O Units.

Example



Meaning of Displays

I/O Unit Designations for Displays
(see *I/O Units Mounted in Remote Slave Racks*, page 89)

C500, 1000H/C2000H I/O Units

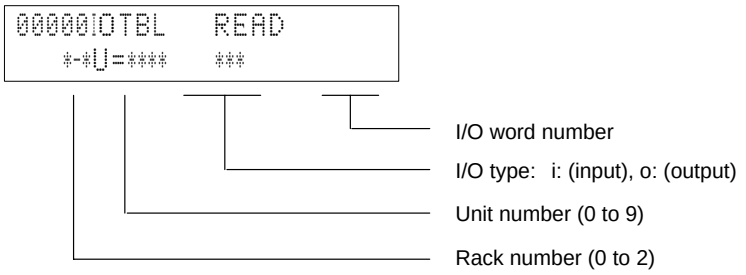
No. of points	Input Unit	Output Unit
16	I * * *	O * * *
32	I I * *	O O * *
64	I I I I	O O O O

C200H I/O Units

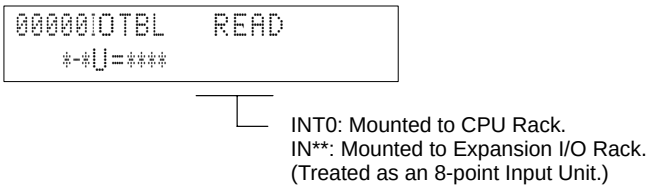
No. of points	Input Unit	Output Unit
8	i(*)* *	o * * *
16	i i * *	o o * *

Note: (*) is i for non-fatal errors or F_

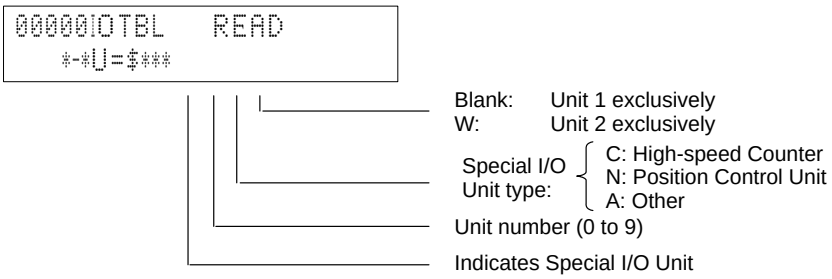
I/O Units



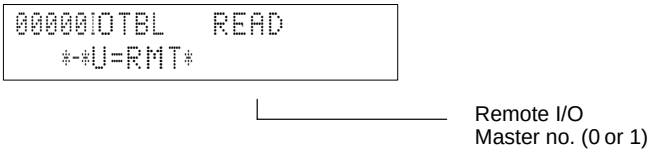
Interrupt Input Units



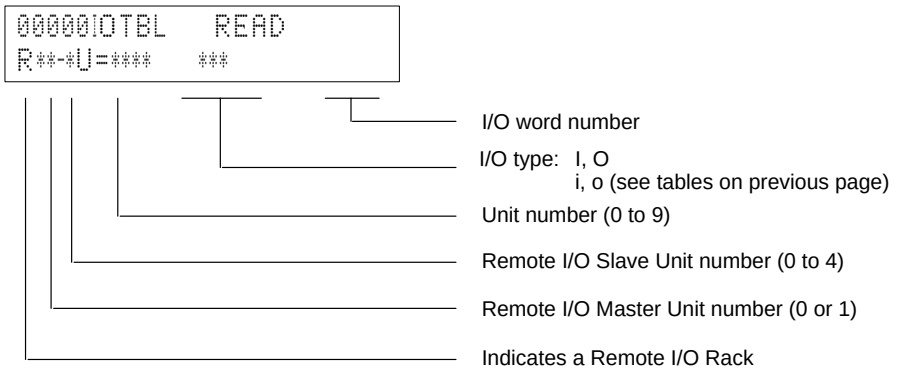
Special I/O Units



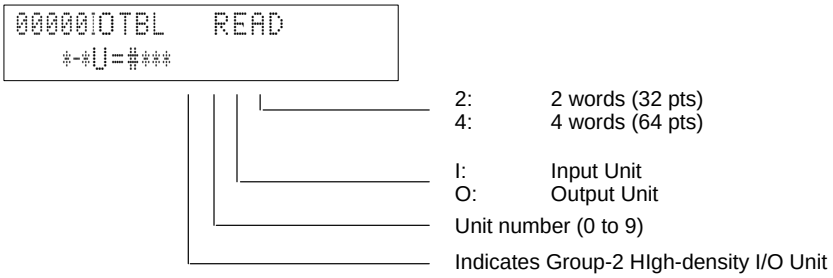
Remote I/O Master Units



Remote I/O Slave Racks

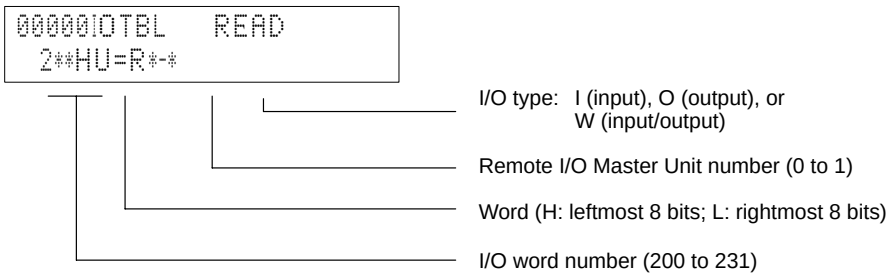


Group-2 HIgh-density I/O Units



Note Group-2 HIgh-density I/O Units will not be displayed in the I/O table when it is displayed using LSS (host computer). Four asterisks (****), indicating no Unit, will be displayed instead.

Optical I/O Units and Remote Terminals



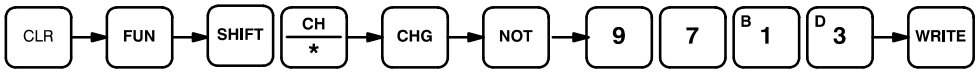
4-6-8 Clearing the I/O Table

The I/O Table Clear operation is used to delete the contents of the I/O table that is currently registered in the CPU memory. The PC will be set for operation based on the I/O Units mounted when the I/O Table Clear operation is performed.

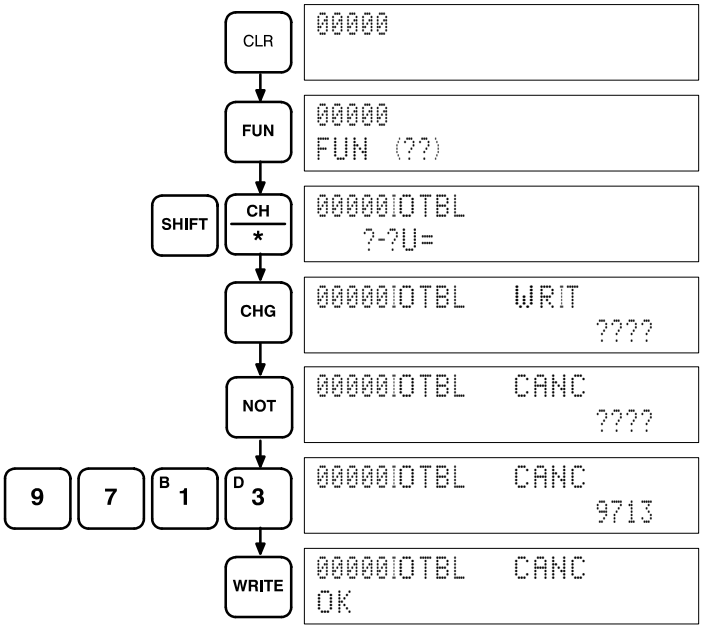
The I/O Table Clear operation will reset all Special I/O Units and Link Units mounted at the time. Do not perform the I/O Table Clear operation when a Host or PC Link Unit, Remote I/O Master Unit, High-speed Counter Unit, Position Control Unit, or other Special I/O Unit is in operation.

Note This operation can be performed only in PROGRAM mode with the write-protection switch (pin 1 of the CPU's DIP switch) set to OFF (OFF="WRITE").

Key Sequence



Example



4-6-9 SYSMAC NET Link Table Transfer (CPU31/33-E Only)

The SYSMAC NET Link Table Transfer operation transfers a copy of the SYSMAC NET Link Data Link table to RAM or EEPROM program memory.This allows the user program and SYSMAC NET Link table to be written into EPROM together. This operation is applicable to the CPU31-E and CPU33-E only.

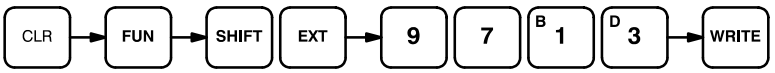
Note When power is applied to a PC which has a copy of a SYSMAC NET Link table stored in its program memory, the SYSMAC NET Link table of the CPU will be overwritten. Changes made in the SYSMAC NET Link table do not affect the copy of the SYSMAC NET Link table in program memory; SYSMAC NET Link Table Transfer must be repeated to change the copy in program memory.

The SYSMAC NET Link Table Transfer operation will not work if:

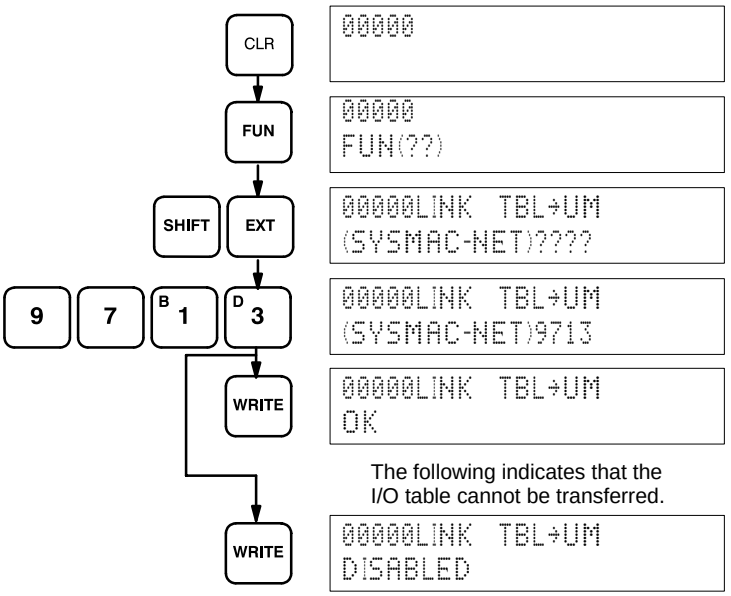
- The Memory Unit is not RAM or EEPROM, or the write protect switch is not set to write.
- There isn't an END(01) instruction.
- The contents of program memory exceeds 14.7K words. (To find the size of the contents of program memory, do an instruction search for END(01).)

SYSMAC NET Link table transfer can only be done in PROGRAM mode.

Key Sequence



Example



4-7 Inputting, Modifying, and Checking the Program

Once a program is written in mnemonic code, it can be input directly into the PC from a Programming Console. Mnemonic code is keyed into Program Memory addresses from the Programming Console. Checking the program involves a syntax check to see that the program has been written according to syntax rules. Once syntax errors are corrected, a trial execution can begin and, finally, correction under actual operating conditions can be made.

The operations required to input a program are explained below. Operations to modify programs that already exist in memory are also provided in this section, as well as the procedure to obtain the current cycle time.

Before starting to input a program, check to see whether there is a program already loaded. If there is a program loaded that you do not need, clear it first using the program memory clear key sequence, then input the new program. If you need the previous program, be sure to check it with the program check key sequence and correct it as required. Further debugging methods are provided in *Section 7 Program Monitoring and Execution*.

4-7-1 Setting and Reading from Program Memory Address

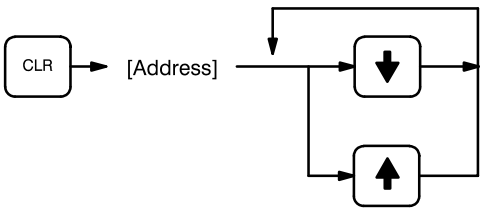
When inputting a program for the first time, it is generally written to Program Memory starting from address 00000. Because this address appears when the display is cleared, it is not necessary to specify it.

When inputting a program starting from other than 00000 or to read or modify a program that already exists in memory, the desired address must be designated. To designate an address, press CLR and then input the desired address. Leading zeros of the address need not be input, i.e., when specifying an address such as 00053 you need to enter only 53. The contents of the designated address will not be displayed until the down key is pressed.

Once the down key has been pressed to display the contents of the designated address, the up and down keys can be used to scroll through Program Memory. Each time one of these keys is pressed, the next or previous word in Program Memory will be displayed.

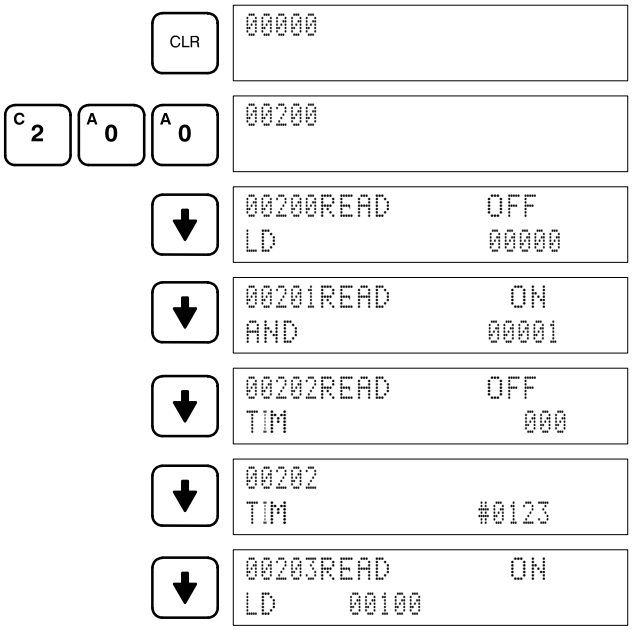
If Program Memory is read in RUN or MONITOR mode, the ON/OFF status of any displayed bit will also be shown.

Key Sequence



Example

If the following mnemonic code has already been input into Program Memory, the key inputs below would produce the displays shown.



Address	Instruction	Operands
00200	LD	00000
00201	AND	00001
00202	TIM	000
		# 0123
00203	LD	00100

4-7-2 Entering and Editing Programs

Programs can be entered and edited only in PROGRAM mode with the write-protect switch (pin 1 of the CPU's DIP switch) set to OFF (OFF="WRITE").

The same procedure is used to either input a program for the first time or to edit a program that already exists. In either case, the current contents of Program Memory is overwritten, i.e., if there is no previous program, the NOP(00) instruction, which will be written at every address, will be overwritten.

To enter a program, input the mnemonic code that was produced from the ladder diagram step-by-step, ensuring that the correct address is set before starting. Once the correct address is displayed, enter the first instruction word and press WRITE. Next, enter the required operands, pressing WRITE after each, i.e., WRITE is pressed at the end of each line of the mnemonic code. When WRITE is pressed at the end of each line, the designated instruction or operand is entered and the next display will appear. If the instruction requires two or more words, the next display will indicate the next operand required and provide a default value for it. If the instruction requires only one word, the next address will be displayed. Continue inputting each line of the mnemonic code until the entire program has been entered.

When inputting numeric values for operands, it is not necessary to input leading zeros. Leading zeros are required only when inputting function codes (see below). When designating operands, be sure to designate the data area for all but IR and SR addresses by pressing the corresponding data area key, and to designate each constant by pressing CONT/#. CONT/# is not required for counter or timer SVs (see below). The AR area is designated by pressing SHIFT and then HR. TC numbers as bit operands (i.e., completion flags) are designated by pressing either TIM or CNT before the address, depending on whether the TC number has been used to define a timer or a counter. To designate an indirect DM address, press CH/∗ before the address (pressing DM is not necessary for an indirect DM address).

Inputting SV for Counters and Timers

The SV (set value) for a timer or counter is generally entered as a constant, although inputting the address of a word that holds the SV is also possible. When inputting an SV as a constant, CONT/# is not required; just input the numeric value and press WRITE. To designate a word, press CLR and then input the word address as described above.

Designating Instructions

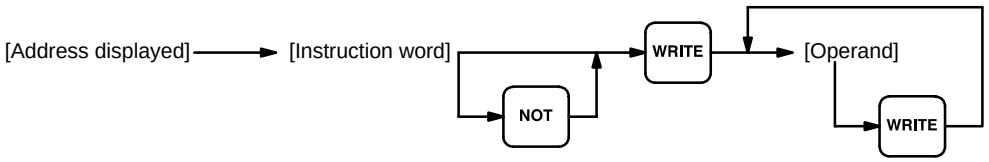
The most basic instructions are input using the Programming Console keys provided for them. All other instructions are entered using function codes. These function codes are always written after the instruction's mnemonic. If no function code is given, there should be a Programming Console key for that instruction.

To designate the differentiated form of an instruction, press NOT after the function code.

To input an instruction using a function code, set the address, press FUN, input the function code including any leading zeros, press NOT if the differentiated form of the instruction is desired, input any bit operands or definers required for the instruction, and then press WRITE.

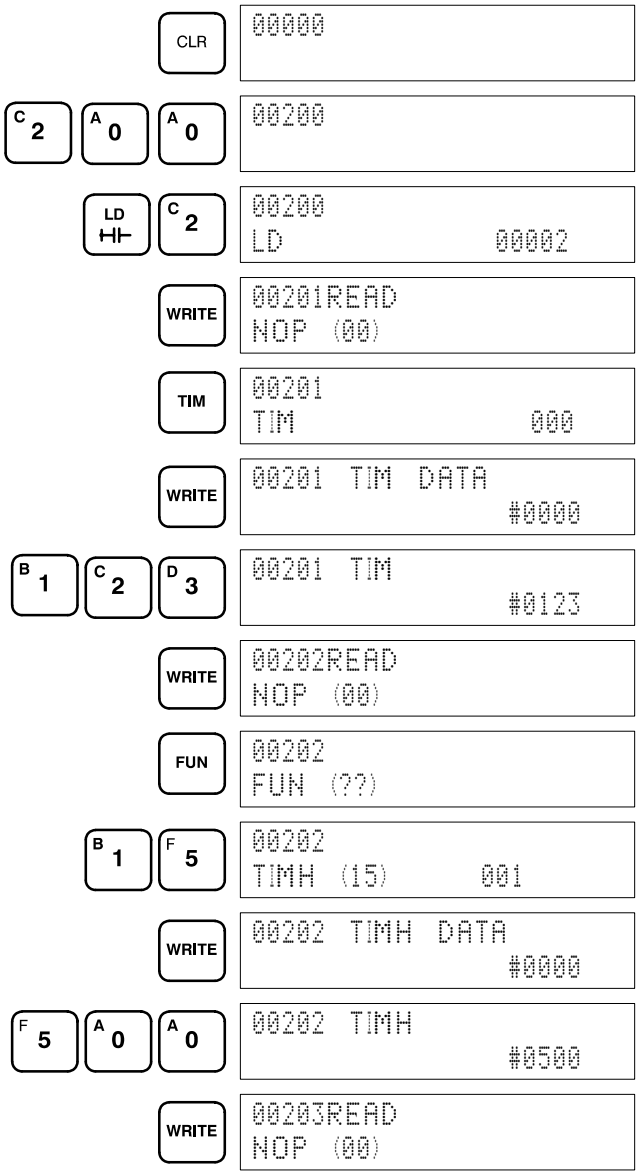
⚠ **Caution** Enter function codes with care and be sure to press SHIFT when required.

Key Sequence



Example

The following program can be entered using the key inputs shown below. Displays will appear as indicated.



Address	Instruction	Operands
00200	LD	00002
00201	TIM	000
		# 0123
00202	TIMH(15)	001
		# 0500

Error Messages

The following error messages may appear when inputting a program. Correct the error as indicated and continue with the input operation. The asterisks in the displays shown below will be replaced with numeric data, normally an address, in the actual display.

Message	Cause and correction
****REPL ROM	An attempt was made to write to write-protected RAM or EEPROM. Ensure that the write-protect switch is set to OFF.
****PROG OVER	The instruction at the last address in memory is not NOP(00). Erase all unnecessary instructions at the end of the program.
****ADDR OVER	An address was set that is larger than the highest memory address in the UM area. Input a smaller address
****SETDATA ERR	Data has been input in the wrong format or beyond defined limits, e.g., a hexadecimal value has been input for BCD. Re-enter the data. This error will generate a FALS 00 error.
****I/O NO. ERR	A data area address has been designated that exceeds the limit of the data area, e.g., an address is too large. Confirm the requirements for the instruction and re-enter the address.

4-7-3 Checking the Program

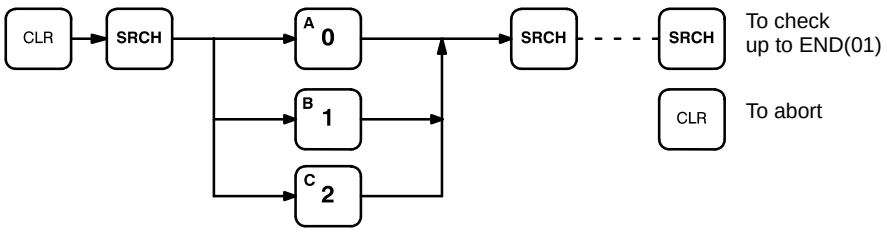
Once a program has been entered, the syntax should be checked to verify that no programming rules have been violated. This check should also be performed if the program has been changed in any way that might create a syntax error.

To check the program, input the key sequence shown below. The numbers indicate the desired check level (see below). When the check level is entered, the program check will start. If an error is discovered, the check will stop and a display indicating the error will appear. Press SRCH to continue the check. If an error is not found, the program will be checked through to the first END(01), with a display indicating when each 64 instructions have been checked (e.g., display #1 of the example after the following table).

CLR can be pressed to cancel the check after it has been started, and a display like display #2, in the example, will appear. When the check has reached the first END, a display like display #3 will appear.

A syntax check can be performed on a program only in PROGRAM mode.

Key Sequence



(0, 1, 2, Check levels)

Check Levels and Error Messages

Three levels of program checking are available. The desired level must be designated to indicate the type of errors that are to be detected. The following table provides the error types, displays, and explanations of all syntax errors. Check level 0 checks for type A, B, and C errors; check level 1, for type A and B errors; and check level 2, for type A errors only.

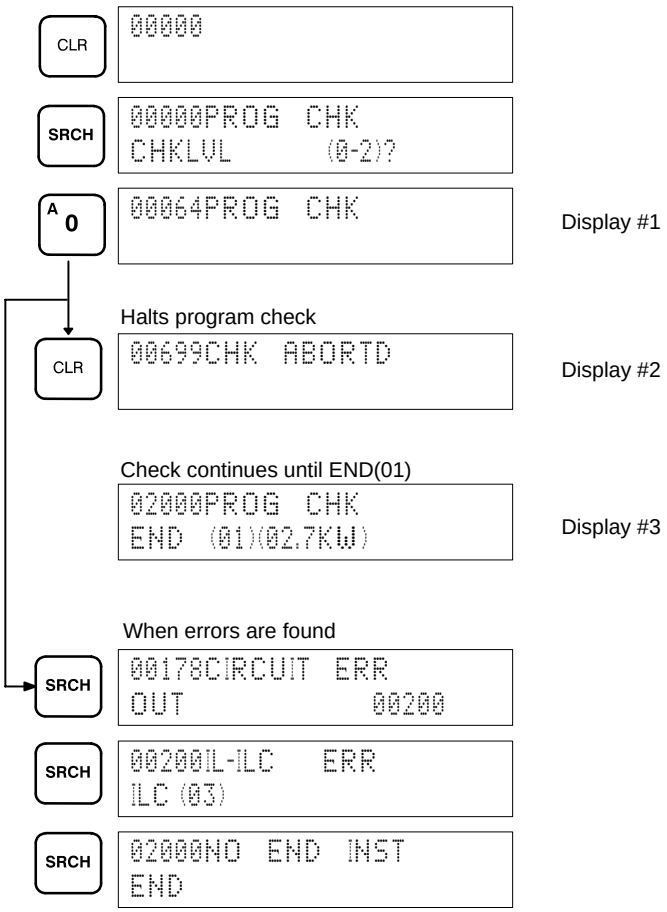
The address where the error was generated will also be displayed.

Many of the following errors are for instructions that have not yet been described yet. Refer to *4-8 Controlling Bit Status* or to *Section 5 Instruction Set* for details on these.

Type	Message	Meaning and appropriate response
Type A	?????	The program has been lost. Re-enter the program.
	NO END INSTR	There is no END(01) in the program. Write END(01) at the final address in the program.
	CIRCUIT ERR	The number of logic blocks and logic block instructions does not agree, i.e., either LD or LD NOT has been used to start a logic block whose execution condition has not been used by another instruction, or a logic block instruction has been used that does not have the required number of logic blocks. Check your program.
	LOCN ERR	An instruction is in the wrong place in the program. Check instruction requirements and correct the program.
	DUPL	The same jump number or subroutine number has been used twice. Correct the program so that the same number is only used once for each. (Jump number 00 may be used as often as required.)
	SBN UNDEFD	SBS(91) has been programmed for a subroutine number that does not exist. Correct the subroutine number or program the required subroutine.
	JME UNDEFD	A JME(04) is missing for a JMP(05). Correct the jump number or insert the proper JME(04).
	OPERAND ERR	A constant entered for the instruction is not within defined values. Change the constant so that it lies within the proper range.
	STEP ERR	STEP(08) with a section number and STEP(08) without a section number have been used correctly. Check STEP(08) programming requirements and correct the program.
Type B	IL-ILC ERR	IL(02) and ILC(03) are not used in pairs. Correct the program so that each IL(02) has a unique ILC(03). Although this error message will appear if more than one IL(02) is used with the same ILC(03), the program will executed as written. Make sure your program is written as desired before proceeding.
	JMP-JME ERR	JMP(04) 00 and JME(05) 00 are not used in pairs. Although this error message will appear if more than one JMP(04) 00 is used with the same JME(05) 00, the program will be executed as written. Make sure your program is written as desired before proceeding.
	SBN-RET ERR	If the displayed address is that of SBN(92), two different subroutines have been defined with the same subroutine number. Change one of the subroutine numbers or delete one of the subroutines. If the displayed address is that of RET(93), RET(93) has not been used properly. Check requirements for RET(93) and correct the program.
Type C	JMP UNDEFD	JME(05) has been used with no JMP(04) with the same jump number. Add a JMP(04) with the same number or delete the JME(05) that is not being used.
	SBS UNDEFD	A subroutine exists that is not called by SBS(91). Program a subroutine call in the proper place, or delete the subroutine if it is not required.
	COIL DUPL	The same bit is being controlled (i.e., turned ON and/or OFF) by more than one instruction (e.g., OUT, OUT NOT, DIFU(13), DIFD(14), KEEP(11), SFT(10)). Although this is allowed for certain instructions, check instruction requirements to confirm that the program is correct or rewrite the program so that each bit is controlled by only one instruction.

Example

The following example shows some of the displays that can appear as a result of a program check.

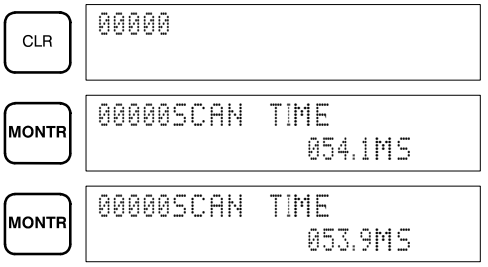


4-7-4 Displaying the Cycle Time

Once the program has been cleared of syntax errors, the cycle time should be checked. This is possible only in RUN or MONITOR mode while the program is being executed. See *Section 6 Program Execution Timing* for details on the cycle time.

To display the current average cycle time, press CLR then MONTR. The time displayed by this operation is a typical cycle time. The differences in displayed values depend on the execution conditions that exist when MONTR is pressed.

Example



4-7-5 Program Searches

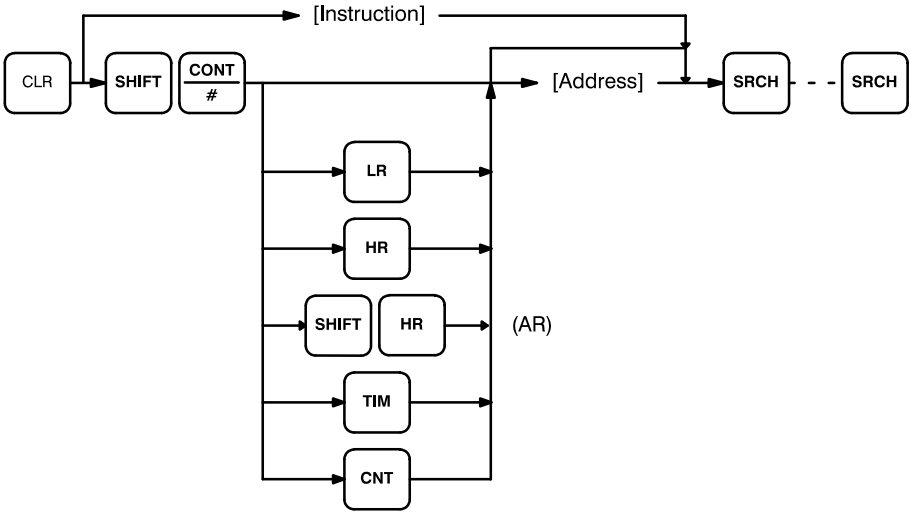
The program can be searched for occurrences of any designated instruction or data area address used in an instruction. Searches can be performed from any currently displayed address or from a cleared display.

To designate a bit address, press SHIFT, press CONT/#, then input the address, including any data area designation required, and press SRCH. To designate an instruction, input the instruction just as when inputting the program and press SRCH. Once an occurrence of an instruction or bit address has been found, any additional occurrences of the same instruction or bit can be found by pressing SRCH again. SRCH'G will be displayed while a search is in progress.

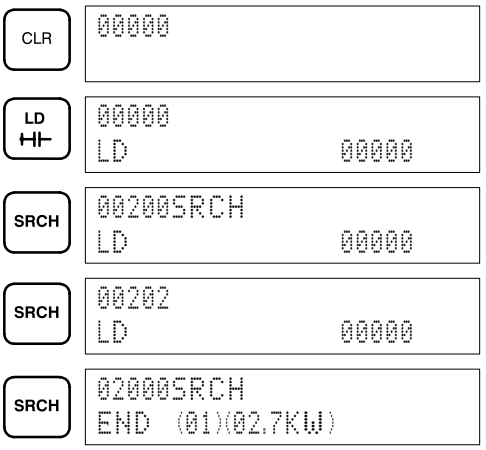
When the first word of a multiword instruction is displayed for a search operation, the other words of the instruction can be displayed by pressing the down key before continuing the search.

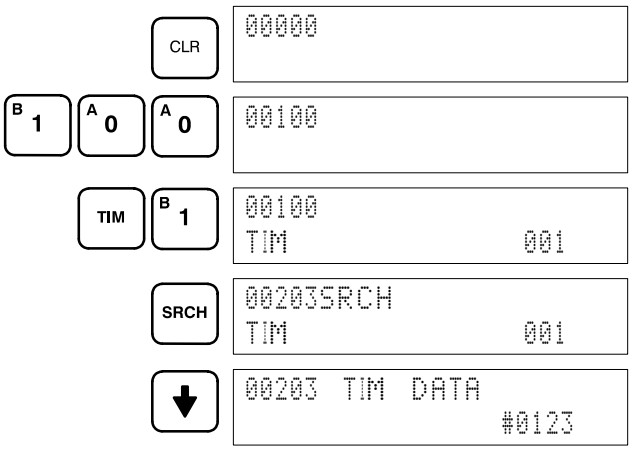
If Program Memory is read in RUN or MONITOR mode, the ON/OFF status of any bit displayed will also be shown.

Key Sequence

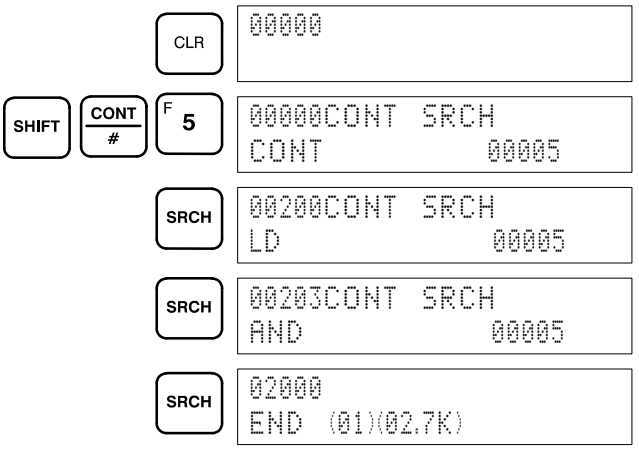


Example:
Instruction Search





Example:
Bit Search



4-7-6 Inserting and Deleting Instructions

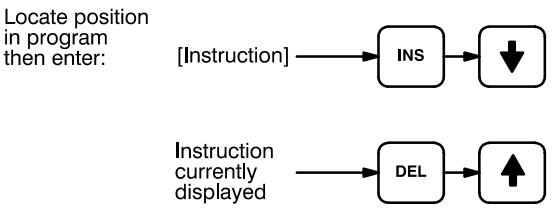
In PROGRAM mode, any instruction that is currently displayed can be deleted or another instruction can be inserted before it. These operations are possible only in PROGRAM mode with the write-protect switch (pin 1 of the CPU's DIP switch) set to OFF (OFF="WRITE").

To insert an instruction, display the instruction before which you want the new instruction to be placed, input the instruction word in the same way as when inputting a program initially, and then press INS and the down key. If other words are required for the instruction, input these in the same way as when inputting the program initially.

To delete an instruction, display the instruction word of the instruction to be deleted and then press DEL and the up key. All the words for the designated instruction will be deleted.

⚠ **Caution** Be careful not to inadvertently delete instructions; there is no way to recover them without reinputting them completely.

Key Sequences



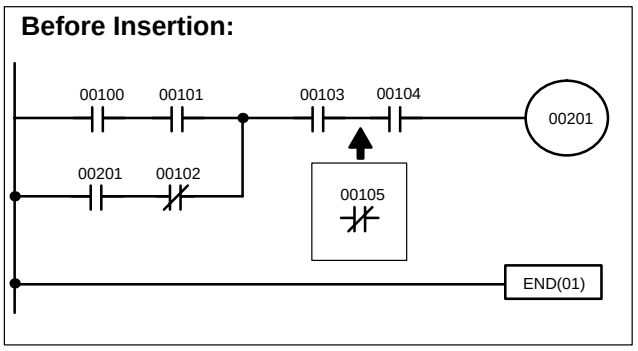
When an instruction is inserted or deleted, all addresses in Program Memory following the operation are adjusted automatically so that there are no blank addresses or no unaddressed instructions.

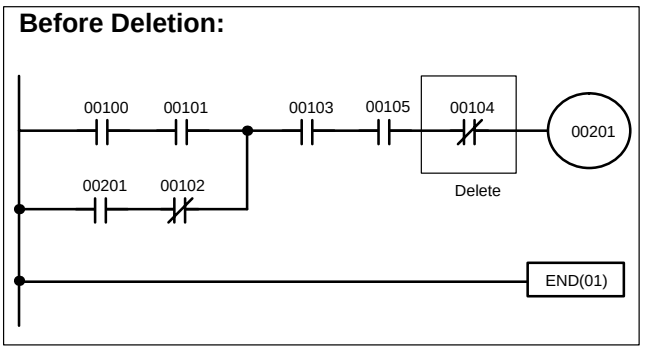
Example

The following mnemonic code shows the changes that are achieved in a program through the key sequences and displays shown below.

Original Program

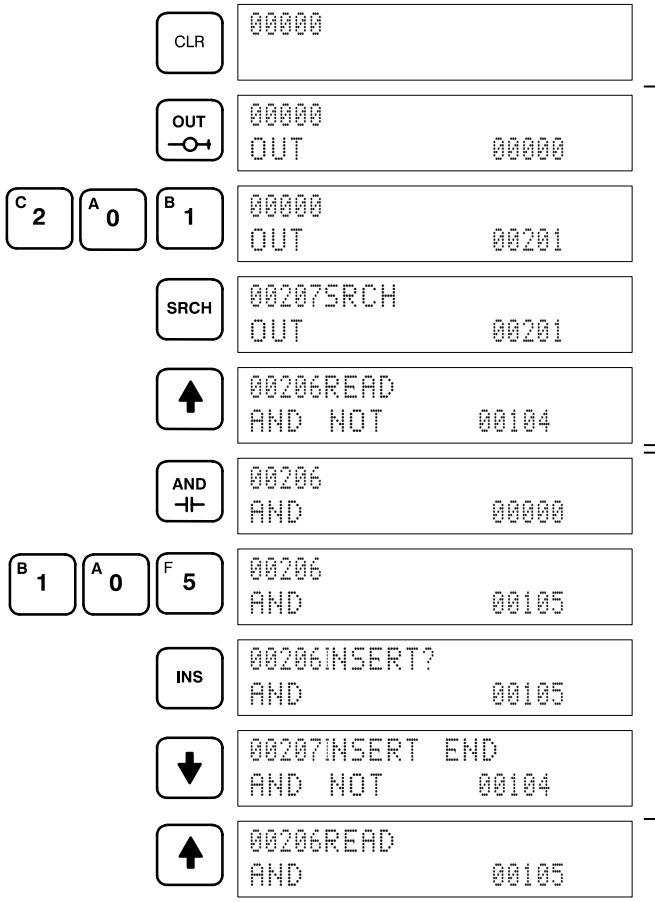
Address	Instruction	Operands
00000	LD	00100
00001	AND	00101
00002	LD	00201
00003	AND NOT	00102
00004	OR LD	—
00005	AND	00103
00006	AND NOT	00104
00007	OUT	00201
00008	END(01)	—





The following key inputs and displays show the procedure for achieving the program changes shown above.

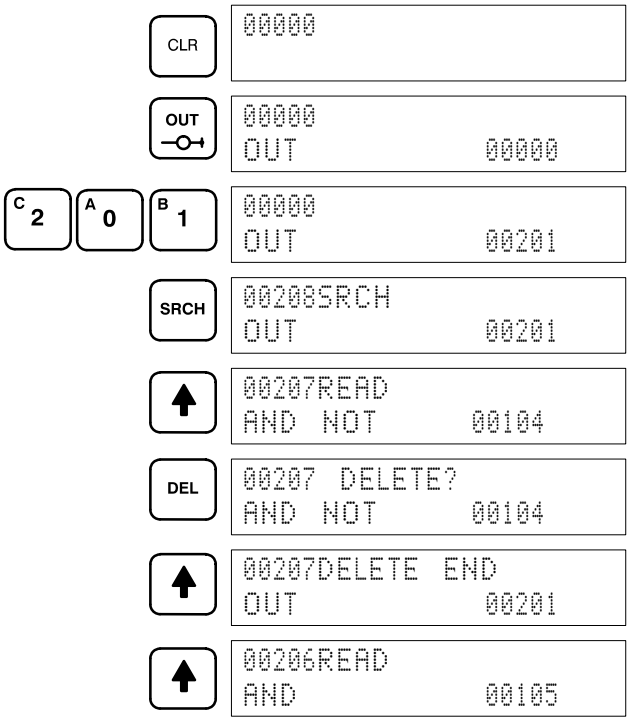
Inserting an Instruction



Program After Insertion

Address	Instruction	Operands
00000	LD	00100
00001	AND	00101
00002	LD	00201
00003	AND NOT	00102
00004	OR LD	—
00005	AND	00103
00006	AND	00105
00007	AND NOT	00104
00008	OUT	00201
00009	END(01)	—

Deleting an Instruction



Program After Deletion

Address	Instruction	Operands
00000	LD	00100
00001	AND NOT	00101
00002	LD	00201
00003	AND NOT	00102
00004	OR LD	—
00005	AND	00103
00006	AND	00105
00007	AND NOT	00104
00008	OUT	00201

4-7-7 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 on 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.

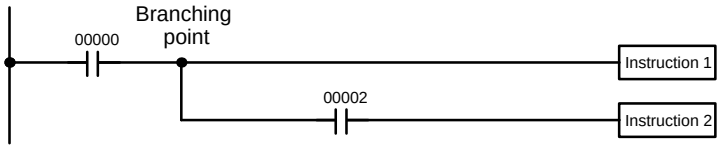


Diagram A: Correct Operation

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

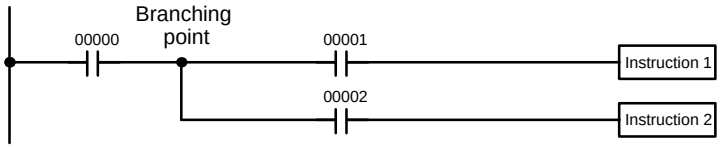


Diagram B: Incorrect Operation

Address	Instruction	Operands
00000	LD	00000
00001	AND	00001
00002	Instruction 1	
00003	AND	00002
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(02)/IL(03)).

TR Bits

The TR area provides eight bits, TR 0 through TR 7, 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.

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

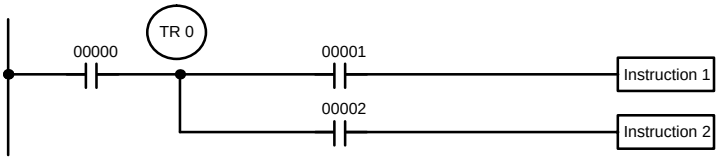
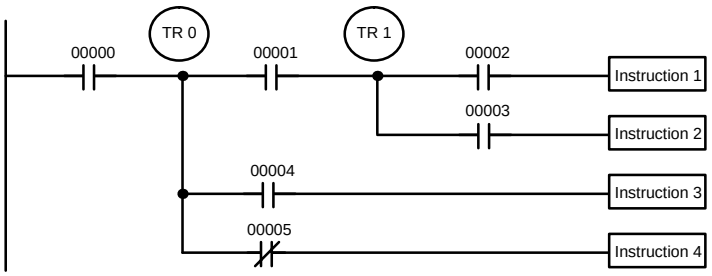


Diagram B: Corrected Using a TR bit

Address	Instruction	Operands
00000	LD	00000
00001	OUT	TR 0
00002	AND	00001
00003	Instruction 1	
00004	LD	TR 0
00005	AND	00002
00006	Instruction 2	

In terms of actual instructions the above diagram would be as follows: The status of IR 00000 is loaded (a LOAD instruction) to establish the initial execution condition. This execution condition is then output using an OUTPUT instruction to TR 0 to store the execution condition at the branching point. The execution condition is then ANDed with the status of IR 00001 and instruction 1 is executed accordingly. The execution condition that was stored at the branching point is then re-loaded (a LOAD instruction with TR 0 as the operand), this is ANDed with the status of IR 00002, and instruction 2 is executed accordingly.

The following example shows an application using two TR bits.



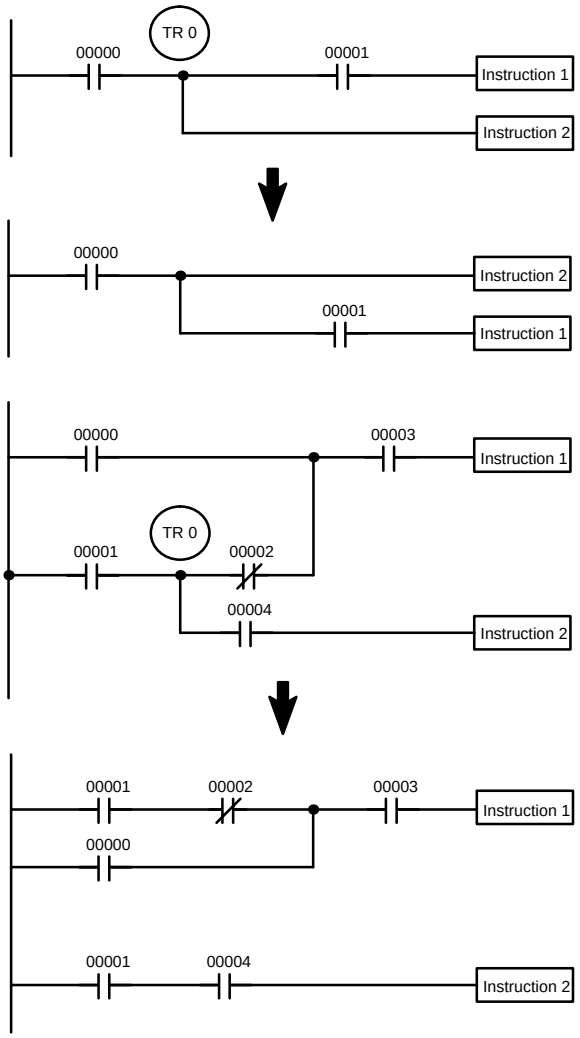
Address	Instruction	Operands
00000	LD	00000
00001	OUT	TR 0
00002	AND	00001
00003	OUT	TR 1
00004	AND	00002
00005	OUT	00500
00006	LD	TR 1
00007	AND	00003
00008	OUT	00501
00009	LD	TR 0
00010	AND	00004
00011	OUT	00502
00012	LD	TR 0
00013	AND NOT	00005
00014	OUT	00503

In this example, TR 0 and TR 1 are used to store the execution conditions at the branching points. After executing instruction 1, the execution condition stored in TR 1 is loaded for an AND with the status IR 00003. The execution condition stored in TR 0 is loaded twice, the first time for an AND with the status of IR 00004 and the second time for an AND with the inverse of the status of IR 00005.

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. Here, 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.



Note TR bits are only used when programming using mnemonic code. They are not necessary when inputting ladder diagrams directly. The above limitations on the number of branching points requiring TR bits, and considerations on methods to reduce the number of programming instructions, still hold.

Interlocks

The problem of storing execution conditions at branching points can also be handled by using the INTERLOCK (IL(02)) and INTERLOCK CLEAR (ILC(03)) 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.

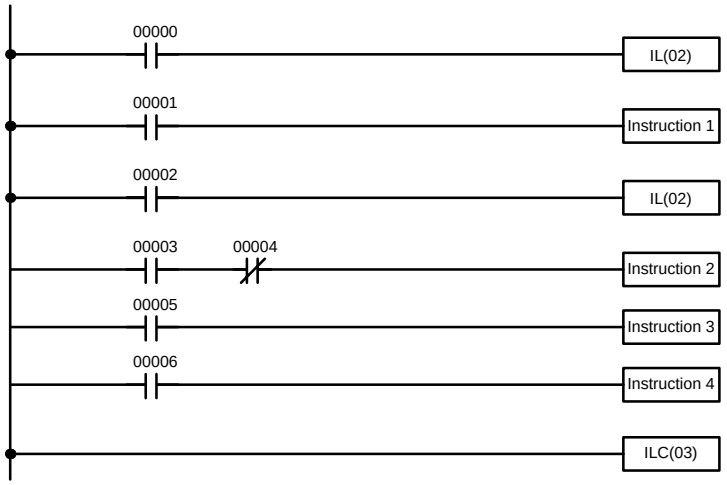
When an INTERLOCK instruction is placed before a section of a ladder program, the execution condition for the INTERLOCK instruction will control the execution of all instruction up to the next INTERLOCK CLEAR instruction. If the execution condition for the INTERLOCK instruction is OFF, all right-hand 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 *5-10 INTERLOCK and INTERLOCK CLEAR – IL(02) and ILC(03).*

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. Note that neither INTERLOCK nor INTERLOCK CLEAR requires an operand.

Address	Instruction	Operands
00000	LD	00000
00001	IL(02)	---
00002	LD	00001
00003	Instruction 1	
00004	LD	00002
00005	Instruction 2	
00006	ILC(03)	---

If IR 00000 is ON in the revised version of diagram B, above, the status of IR 00001 and that of IR 00002 would determine the execution conditions for instructions 1 and 2, respectively. Because IR 00000 is ON, this would produce the same results as ANDing the status of each of these bits. If IR 00000 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 in the following diagram, more than one INTERLOCK instruction can be used within one instruction block; each is effective through the next INTERLOCK CLEAR instruction.



Address	Instruction	Operands
00000	LD	00000
00001	IL(02)	---
00002	LD	00001
00003	Instruction 1	
00004	LD	00002
00005	IL(02)	---
00006	LD	00003
00007	AND NOT	00004
00008	Instruction 2	
00009	LD	00005
00010	Instruction 3	
00011	LD	00006
00012	Instruction 4	
00013	ILC(03)	---

If IR 00000 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 IR 00000 is ON, the status of IR 00001 would be loaded as the execution condition for instruction 1 and then the status of IR 00002 would be loaded to form the execution condition for the second INTERLOCK instruction. If IR 00002 is OFF, instructions 2 through 4 will be executed with OFF execution conditions. If IR 00002 is ON, IR 00003, IR 00005, and IR 00006 will determine the first execution condition in new instruction lines.

4-7-8 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(04)) and JUMP END (JME(05)) 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 00 and 99. There are two types of jumps. The jump number used determines the type of jump.

A jump can be defined using jump numbers 01 through 99 only once, i.e., each of these numbers can be used once in a JUMP instruction and once in a JUMP END instruction. 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. Diagram B from the TR bit and interlock example could be redrawn as shown below using a jump. Although 01 has been used as the jump number, any number between 01 and 99 could be used as long as it has not already been used in a different part of the program. JUMP and JUMP END require no other operand and JUMP END never has conditions on the instruction line leading to it.

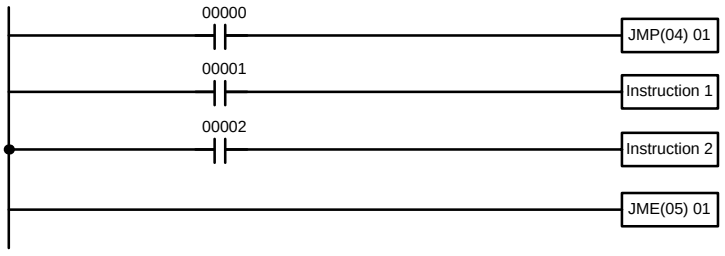


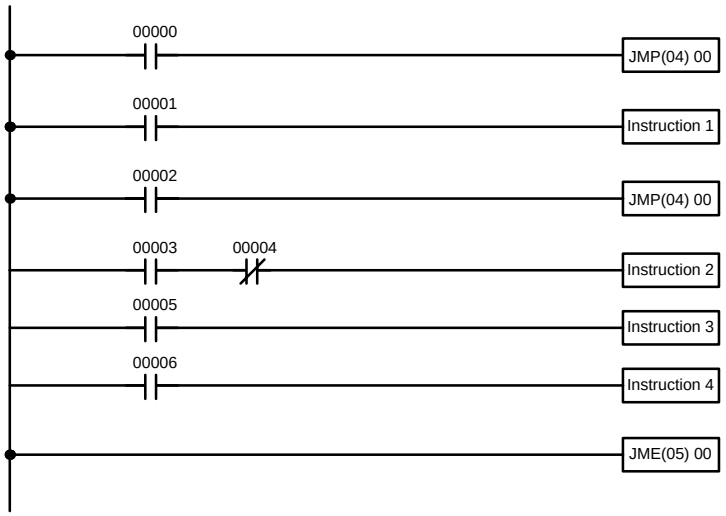
Diagram B: Corrected with a Jump

Address	Instruction	Operands
00000	LD	00000
00001	JMP(04)	01
00002	LD	00001
00003	Instruction 1	
00004	LD	00002
00005	Instruction 2	
00006	JME(05)	015

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

The other type of jump is created with a jump number of 00. As many jumps as desired can be created using jump number 00 and JUMP instructions using 00 can be used consecutively without a JUMP END using 00 between them. It is even possible for all JUMP 00 instructions to move program execution to the same JUMP END 00, i.e., only one JUMP END 00 instruction is required for all JUMP 00 instruction in the program. When 00 is used as the jump number for a JUMP instruction, program execution moves to the instruction following the next JUMP END instruction with a jump number of 00. Although, as in all jumps, no status is changed and no instructions are executed between the JUMP 00 and JUMP END 00 instructions, the program must search for the next JUMP END 00 instruction, producing a slightly longer execution time.

Execution of programs containing multiple JUMP 00 instructions for one JUMP END 00 instruction is similar to that of interlocked sections. The following diagram is the same as that used for the interlock example above, except redrawn with jumps. The execution of this diagram would differ from that of the diagram described above (e.g., in the previous diagram interlocks would reset certain parts of the interlocked section, however, jumps do not affect the status of any bit between the JUMP and JUMP END instructions).



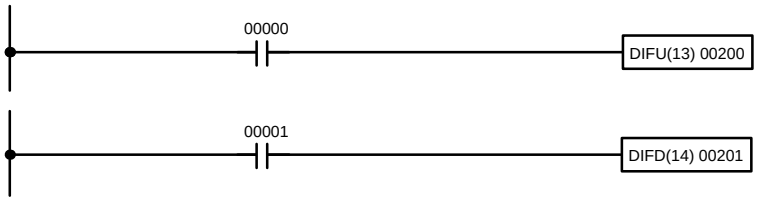
Address	Instruction	Operands
00000	LD	00000
00001	JMP(04)	00
00002	LD	00001
00003	Instruction 1	
00004	LD	00002
00005	JMP(04)	00
00006	LD	00003
00007	AND NOT	00004
00008	Instruction 2	
00009	LD	00005
00010	Instruction 3	
00011	LD	00006
00012	Instruction 4	
00013	JME(05)	00

4-8 Controlling Bit Status

There are five instructions that can be used generally to control individual bit status. These are the OUTPUT, OUTPUT NOT, DIFFERENTIATE UP, DIFFERENTIATE DOWN, and KEEP instructions. All of these instructions appear as the last instruction in an instruction line and take a bit address for an operand. Although details are provided in *5-9 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 IR area (i.e., to send or stop output signals to external devices), they are also used to control the status of other bits in the IR area or in other data areas.

4-8-1 DIFFERENTIATE UP and DIFFERENTIATE DOWN

DIFFERENTIATE UP and DIFFERENTIATE DOWN instructions are used to turn the operand bit ON for one cycle at a time. The DIFFERENTIATE UP instruction turns ON the operand bit for one cycle after the execution condition for it goes from OFF to ON; the DIFFERENTIATE DOWN instruction turns ON the operand bit for one cycle 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	00000
00001	DIFU(13)	00200

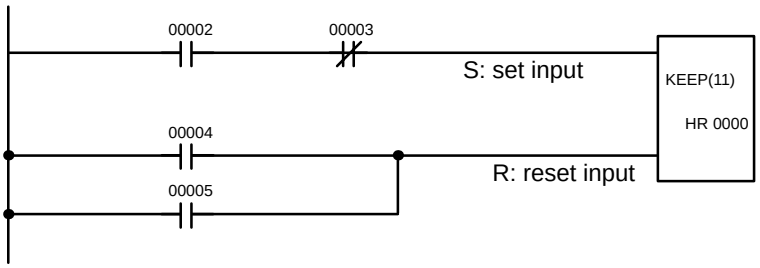
Address	Instruction	Operands
00000	LD	00001
00001	DIFD(14)	00201

Here, IR 00200 will be turned ON for one cycle after IR 00000 goes ON. The next time DIFU(13) 00200 is executed, IR 00200 will be turned OFF, regardless of the status of IR 00000. With the DIFFERENTIATE DOWN instruction, IR 00201 will be turned ON for one cycle after IR 00001 goes OFF (IR 00201 will be kept OFF until then), and will be turned OFF the next time DIFD(14) 00201 is executed.

4-8-2 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, HR 0000 will be turned ON when IR 00002 is ON and IR 00003 is OFF. HR 0000 will then remain ON until either IR 00004 or IR 00005 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.



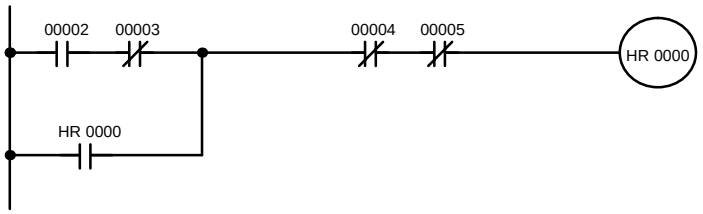
Address	Instruction	Operands
00000	LD	00002
00001	AND NOT	00003
00002	LD	00004
00003	OR	00005
00004	KEEP(11)	HR 0000

4-8-3 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., HR 0000 can be turned OFF by turning ON either IR 00004 or IR 00005.



Address	Instruction	Operands
00000	LD	00002
00001	AND NOT	00003
00002	OR	HR 0000
00003	AND NOT	00004
00004	AND NOT	00005
00005	OUT	HR 0000

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 IR area that are not allocated as I/O bits, and certain unused bits in the AR area, 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

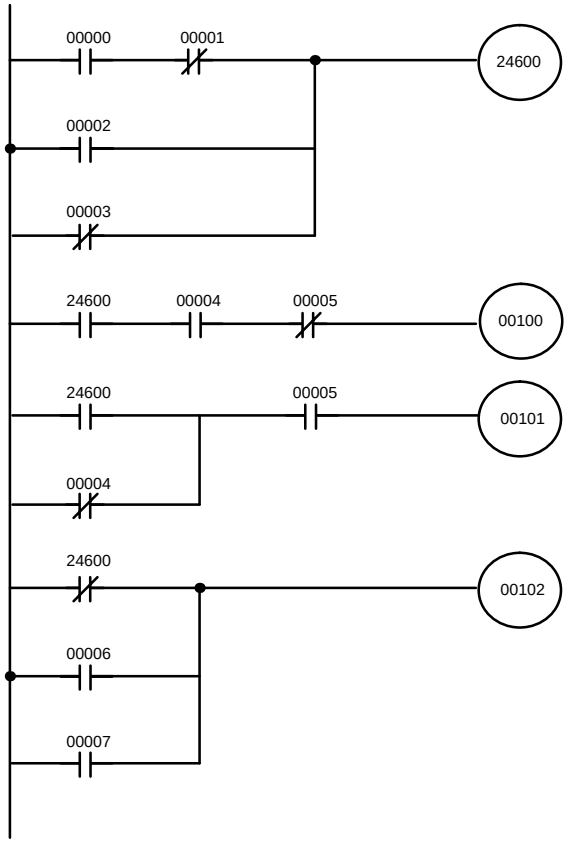
Examples given later in this subsection show two of the most common ways to employ work bits. These should act as a guide to the almost limitless number of ways in which the work bits can be used. Whenever difficulties arise in programming a control action, consideration should be given to work bits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

Work bits are often used with the OUTPUT, OUTPUT NOT, DIFFERENTIATE UP, DIFFERENTIATE DOWN, and KEEP instructions. 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(10)). An example of the use of work words and bits with the SHIFT REGISTER instruction is provided in *5-15-1 SHIFT REGISTER – SFT(10)*.

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, IR 00000, IR 00001, IR 00002, and IR 00003 are combined in a logic block that stores the resulting execution condition as the status of IR 24600. IR 24600 is then combined with various other conditions to determine output conditions for IR 00100, IR 00101, and IR 00102, i.e., to turn the outputs allocated to these bits ON or OFF.

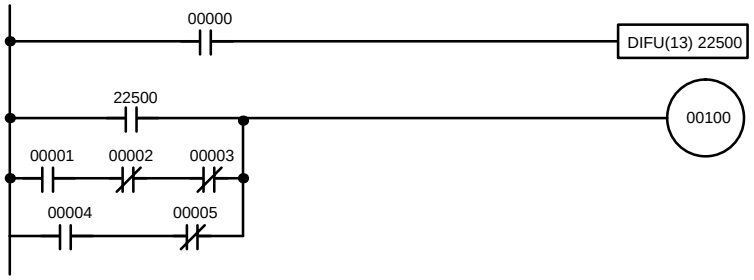


Address	Instruction	Operands
00000	LD	00000
00001	AND NOT	00001
00002	OR	00002
00003	OR NOT	00003
00004	OUT	24600
00005	LD	24600
00006	AND	00004
00007	AND NOT	00005
00008	OUT	00100
00009	LD	24600
00010	OR NOT	00004
00011	AND	00005
00012	OUT	00101
00013	LD NOT	24600
00014	OR	00006
00015	OR	00007
00016	OUT	00102

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, IR 00100 must be left ON continuously as long as IR 00001 is ON and both IR 00002 and IR 00003 are OFF, or as long as IR 00004 is ON and IR 00005 is OFF. It must be turned ON for only one cycle each time IR 00000 turns ON (unless one of the preceding conditions is keeping it ON continuously).

This action is easily programmed by using IR 22500 as a work bit as the operand of the DIFFERENTIATE UP instruction (DIFU(13)). When IR 00000 turns ON, IR 22500 will be turned ON for one cycle and then be turned OFF the next cycle by DIFU(13). Assuming the other conditions controlling IR 00100 are not keeping it ON, the work bit IR 22500 will turn IR 00100 ON for one cycle only.



Address	Instruction	Operands
00000	LD	00000
00001	DIFU(13)	22500
00002	LD	22500
00003	LD	00001
00004	AND NOT	00002
00005	AND NOT	00003
00006	OR LD	---
00007	LD	00004
00008	AND NOT	00005
00009	OR LD	---
00010	OUT	00100

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 a clear diagram. 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.

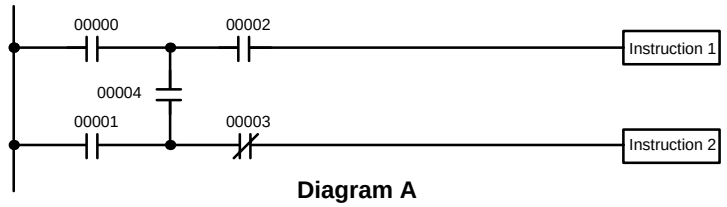


Diagram A

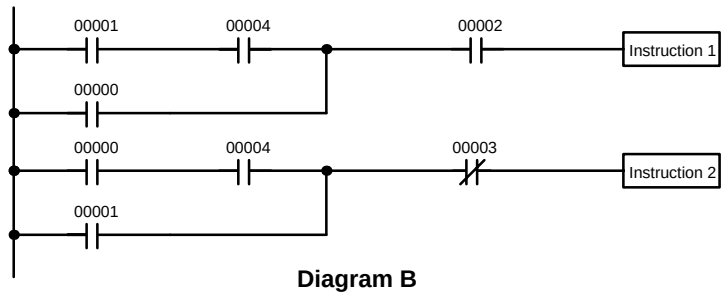


Diagram B

Address	Instruction	Operands
00000	LD	00001
00001	AND	00004
00002	OR	00000
00003	AND	00002
00004	Instruction 1	
00005	LD	00000
00006	AND	00004
00007	OR	00001
00008	AND NOT	00003
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 (SR 25313) in the SR area can be used.

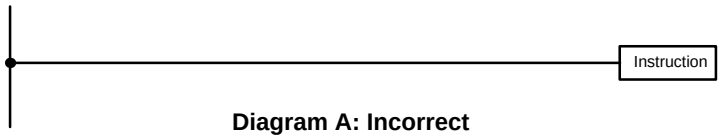


Diagram A: Incorrect

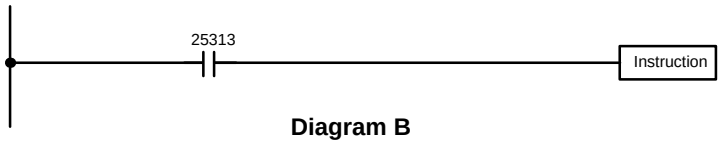


Diagram B

Address	Instruction	Operands
00000	LD	25313
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 *5-8-2 AND LOAD and OR LOAD* for more details and *Section 7 Program Monitoring and Execution* for further examples.

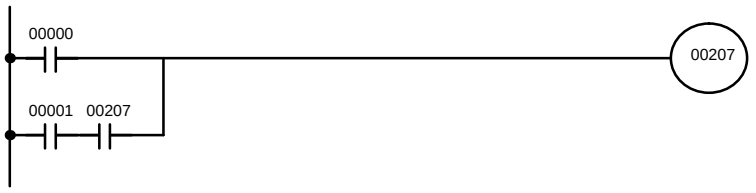


Diagram A

Address	Instruction	Operands
00000	LD	00000
00001	LD	00001
00002	AND	00207
00003	OR LD	---
00004	OUT	00207

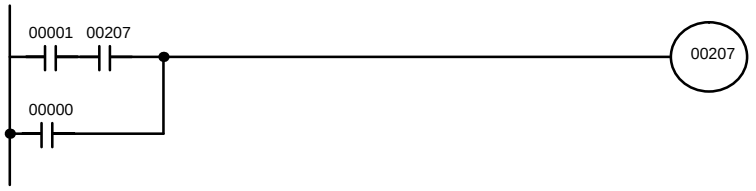


Diagram B

Address	Instruction	Operands
00000	LD	00001
00001	AND	00207
00002	OR	00000
00003	OUT	002

4-11 Program Execution

When program execution is started, the CPU cycles 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 an instruction lines branching from the first instruction line to other terminal instructions at the right.

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

SECTION 5
Instruction Set

The C200HS PC has a large programming instruction set that allows for easy programming of complicated control processes. This section explains instructions individually and provides the ladder diagram symbol, data areas, and flags used with each. The C200HS can process more than 100 instructions that require function codes, but only 100 function codes (00 to 99) are available. Some instructions, called expansion instructions, do not have fixed function codes and must be assigned function codes from the 18 function codes set aside for expansion instructions before they can be used.

The many instructions provided by the C200HS are organized in the following subsections by instruction group. These groups include Ladder Diagram Instructions, Bit Control Instructions, Timer and Counter Instructions, Data Shifting Instructions, Data Movement Instructions, Data Comparison Instructions, Data Conversion Instructions, BCD Calculation Instructions, Binary Calculation Instructions, Logic Instructions, Subroutines, Special Instructions, and Network Instructions.

Some instructions, such as Timer and Counter instructions, are used to control execution of other instructions, e.g., a TIM 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, all 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 used 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 2-digit decimal numbers, are used to input most instructions into the CPU and are described briefly below and in more detail in *4-7 Inputting, Modifying, and Checking the Program*. A table of instructions listed in order of function codes, is also provided in *Appendix B*.

An @ before a mnemonic indicates the differentiated version of that instruction. Differentiated instructions are explained in *Section 5-4*.

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 TC 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, Definer Values, and Flags

In this section, each instruction description includes its ladder diagram symbol, the data areas that can be used by its operands, and the values that can be used as definers. Details for the data areas are also specified by the operand names and the type of data required for each operand (i.e., word or bit and, for words, hexadecimal or BCD).

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. Also, not all words in the SR and DM areas are writeable as operands (see *Section 3 Memory Areas* for details.) Other specific limitations are given in a *Limitations* subsection. Refer to *Section 3 Memory Areas* for addressing conventions and the addresses of flags and control bits.

⚠ Caution The IR and SR areas are considered as separate data areas. If an operand has access to one area, it doesn't necessarily mean that the same operand will have access to the other area. The border between the IR and SR areas can, however, be crossed for a single operand, i.e., the last bit in the IR area may be specified for an operand that requires more than one word as long as the SR area is also allowed for that operand.

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

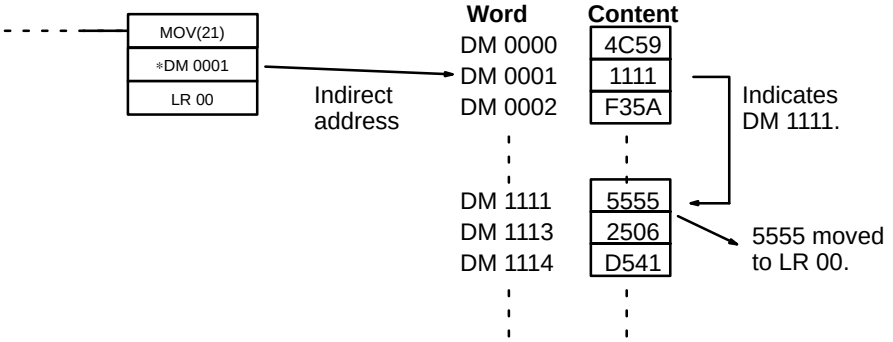
Abbreviation	Name	Bit
ER	Instruction Execution Error Flag	25503
CY	Carry Flag	25504
GR	Greater Than Flag	25505
EQ	Equals Flag	25506
LE	Less Than Flag	25507
OF	Overflow Flag	25404
UF	Underflow Flag	25405

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 the current instruction. The *Flags* subsection of each instruction lists possible reasons for ER being ON. ER will turn ON if operands are not entered correctly. Instructions are not executed when ER is ON. A table of instructions and the flags they affect is provided in *Appendix C Error and Arithmetic Flag Operation*.

Indirect Addressing

When the DM area is specified for an operand, an indirect address can be used. Indirect DM addressing is specified by placing an asterisk before the DM: *DM.

When an indirect DM address is specified, the designated DM word will contain the address of the DM word that contains the data that will be used as the operand of the instruction. If, for example, *DM 0001 was designated as the first operand and LR 00 as the second operand of MOV(21), the contents of DM 0001 was 1111, and DM 1111 contained 5555, the value 5555 would be moved to LR 00.



When using indirect addressing, the address of the desired word must be in BCD and it must specify a word within the DM area. In the above example, the content of *DM 0000 would have to be in BCD between 0000 and 1999.

Designating Constants

Although data area addresses are most often given as operands, many operands and all definers are input as constants. The available value range for a given definer or 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.

5-4 Differentiated Instructions

Most instructions are provided in both differentiated and non-differentiated forms. Differentiated instructions are distinguished by an @ in front of the instruction mnemonic.

A non-differentiated instruction is executed each time it is cycled as long as its execution condition is ON. A differentiated 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 cycled, the instruction will not be executed. The following two examples show how this works with MOV(21) and @MOV(21), which are used to move the data in the address designated by the first operand to the address designated by the second operand.

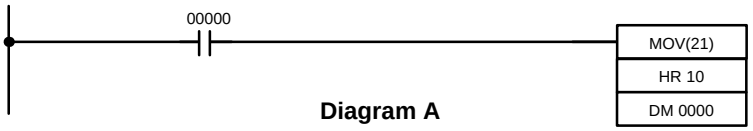


Diagram A

Address	Instruction	Operands	
00000	LD		00000
00001	MOV(21)		
		HR	10
		DM	0000

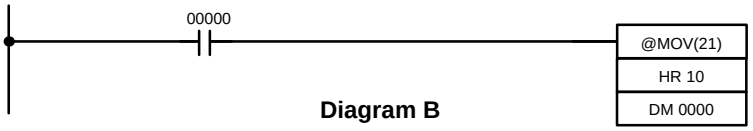


Diagram B

Address	Instruction	Operands	
00000	LD		00000
00001	@MOV(21)		
		HR	10
		DM	0000

In diagram A, the non-differentiated MOV(21) will move the content of HR 10 to DM 0000 whenever it is cycled with 00000. If the cycle time is 80 ms and 00000 remains ON for 2.0 seconds, this move operation will be performed 25 times and only the last value moved to DM 0000 will be preserved there.

In diagram B, the differentiated @MOV(21) will move the content of HR 10 to DM 0000 only once after 00000 goes ON. Even if 00000 remains ON for 2.0 seconds with the same 80 ms cycle time, the move operation will be executed only once during the first cycle in which 00000 has changed from OFF to ON. Because the content of HR 10 could very well change during the 2 seconds while 00000 is ON, the final content of DM 0000 after the 2 seconds could be different depending on whether MOV(21) or @MOV(21) was used.

All operands, ladder diagram symbols, 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, but NOT is input after the function code to designate the differentiated form of an instruction. Most, but not all, instructions have differentiated forms.

Refer to *5-10 INTERLOCK and INTERLOCK CLEAR – IL(02) and IL(03)* for the effects of interlocks on differentiated instructions.

The C200HS also provides differentiation instructions: DIFU(13) and DIFD(14). DIFU(13) operates the same as a differentiated instruction, but is used to turn ON a bit for one cycle. DIFD(14) also turns ON a bit for one cycle, but does it when the execution condition has changed from ON to OFF. Refer to *5-9-2 DIFFERENTIATE UP and DOWN - DIFU(13) and DIFD(14)* for details.

⚠ Caution Do not use SR 25313 and SR25315 for differentiated instructions. These bits never change status and will not trigger differentiated instructions

5-5 Expansion Instructions

The C200HS has more instructions that require function codes (121) than function codes (100), so some instructions do not have fixed function codes. These instructions, called expansion instructions, are listed in the following table. Default function codes are given for the instructions that have them.

An expansion instruction can be assigned one of 18 function codes using the Programming Console's Expansion Instruction Function Code Assignments operation. The 18 function codes are: 17, 18, 19, 47, 48, 60 to 69, 87, 88, and 89. Refer to *7-1-14 Expansion Instruction Function Code Assignments* for details on assigning function codes.

Code	Mnemonic	Name	Page
17	(@)ASFT	ASYNCHRONOUS SHIFT REGISTER	157
18	(@)SCAN	CYCLE TIME	276
19	(@)MCMP	MULTI-WORD COMPARE	169
47	(@)LMSG	32-CHARACTER MESSAGE	279
48	(@)TERM	TERMINAL MODE	280
60	CMPL	DOUBLE COMPARE	172
61	(@)MPRF	GROUP-2 HIGH-DENSITY I/O REFRESH	282
62	(@)XFRB	TRANSFER BITS	168
63	(@)LINE	COLUMN TO LINE	200
64	(@)COLM	LINE TO COLUMN	201
65	(@)SEC	HOURS TO SECONDS	183
66	(@)HMS	SECONDS TO HOURS	184
67	(@)BCNT	BIT COUNTER	283
68	(@)BCMP	BLOCK COMPARE	174
69	(@)APR	ARITHMETIC PROCESS	239
87	TTIM	TOTALIZING TIMER	144
88	ZCP	AREA RANGE COMPARE	176
89	(@)INT	INTERRUPT CONTROL	262
---	7SEG	7-SEGMENT DISPLAY OUTPUT	301
---	(@)ADBL	DOUBLE BINARY ADD	225
---	AVG	AVERAGE VALUE	235
---	CPS	SIGNED BINARY COMPARE	178
---	CPSL	DOUBLE SIGNED BINARY COMPARE	179
---	(@)DBS	SIGNED BINARY DIVIDE	231
---	(@)DBSL	DOUBLE SIGNED BINARY DIVIDE	232
---	DSW	DIGITAL SWITCH INPUT	304
---	(@)FCS	FCS CALCULATE	283
---	FPD	FAILURE POINT DETECT	285
---	(@)HEX	ASCII-TO-HEXADECIMAL	195
---	HKY	HEXADECIMAL KEY INPUT	308
---	(@)MAX	FIND MAXIMUM	233
---	(@)MBS	SIGNED BINARY MULTIPLY	229
---	(@)MBSL	DOUBLE SIGNED BINARY MULTIPLY	230
---	(@)MIN	FIND MINIMUM	234
---	MTR	MATRIX INPUT	313
---	(@)NEG	2'S COMPLEMENT	202
---	(@)NEGL	DOUBLE 2'S COMPLEMENT	203
---	PID	PID CONTROL	242
---	(@)RXD	RECEIVE	297
---	(@)SBBL	DOUBLE BINARY SUBTRACT	227
---	(@)SCL	SCALING	198
---	(@)SRCH	DATA SEARCH	289
---	(@)SUM	SUM CALCULATE	237
---	(@)TKY	TEN KEY INPUT	311
---	(@)TXD	TRANSMIT	299
---	(@)XDMR	EXPANSION DM READ	290
---	ZCPL	DOUBLE AREA RANGE COMPARE	177

5-6 Coding Right-hand Instructions

Writing mnemonic code for ladder instructions is described in *Section 4 Writing and Inputting the Program*. 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.

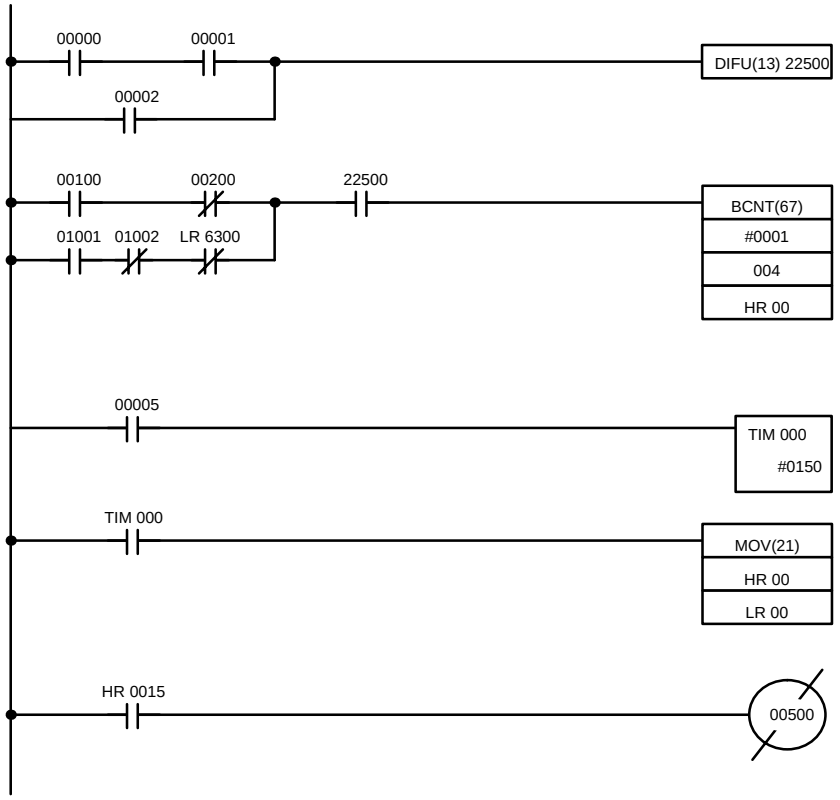
The first word of any instruction defines the instruction and provides any definers. If the instruction requires only a signal bit operand with no definer, the bit operand is also placed on the same line as the mnemonic. All other operands are placed on lines after the instruction line, one operand per 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 an IR or SR 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 place on the right side. If a constant 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. TC bits, once defined as a timer or counter, take a TIM (timer) or CNT (counter) prefix.

When coding an instruction that has a function code, be sure to write in the function code, which will be necessary when inputting the instruction via the Programming Console. Also be sure to designate the differentiated instruction with the @ symbol.

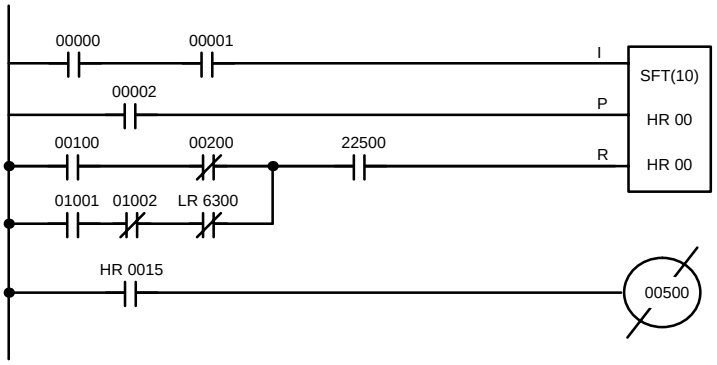
The following diagram and corresponding mnemonic code illustrates the points described above.



Address	Instruction	Data	
00000	LD		00000
00001	AND		00001
00002	OR		00002
00003	DIFU(13)		22500
00004	LD		00100
00005	AND NOT		00200
00006	LD		01001
00007	AND NOT		01002
00008	AND NOT	LR	6300
00009	OR LD		—
00010	AND		22500
00011	BCNT(67)		—
		#	0001
			004
		HR	00
00012	LD		00005
00013	TIM		000
		#	0150
00014	LD	TIM	000
00015	MOV(21)		—
		HR	00
		LR	00
00016	LD	HR	0015
00017	OUT NOT		00500

Multiple Instruction Lines

If a right-hand instruction requires multiple instruction lines (such as KEEP(11)), all of the lines for the instruction are entered before the right-hand instruction. Each of the lines for the instruction is coded, starting with LD or LD NOT, to form 'logic blocks' that are combined by the right-hand instruction. An example of this for SFT(10) is shown below.



Address	Instruction	Data	
00000	LD		00000
00001	AND		00001
00002	LD		00002
00003	LD		00100
00004	AND NOT		00200
00005	LD		01001
00006	AND NOT		01002
00007	AND NOT	LR	6300
00008	OR LD		—
00009	AND		22500
00010	SFT(10)		—
		HR	00
		HR	00
00011	LD	HR	0015
00012	OUT NOT		00500

END(01)

When you have finished coding the program, make sure you have placed END(01) at the last address.

5-7 Instruction Set Lists

This section provides tables of the instructions available in the C200HS. The first table can be used to find instructions by function code. The second table can be used to find instruction by mnemonic. In both tables, the @ symbol indicates instructions with differentiated variations.

Note Refer to *5-5 Expansion Instructions* for a list of the expansion instructions.

5-7-1 Function Codes

The following table lists the instructions that have fixed function codes. Each instruction is listed by mnemonic and by instruction name. Use the numbers in the leftmost column as the left digit and the number in the column heading as the right digit of the function code.

Left digit	Right digit									
	0	1	2	3	4	5	6	7	8	9
0	**NOP** NO OPERATION	**END** END	**IL** INTERLOCK	**ILC** INTERLOCK CLEAR	**JMP** JUMP	**JME** JUMP END	**(@) FAL** FAILURE ALARM AND RESET	**FALS** SEVERE FAILURE ALARM	**STEP** STEP DEFINE	**SNXT** STEP START
1	**SFT** SHIFT REGISTER	**KEEP** KEEP	**CNTR** REVERSIBLE COUNTER	**DIFU** DIFFERENTIATE UP	**DIFD** DIFFERENTIATE DOWN	**TIMH** HIGH-SPEED TIMER	**(@) WSFT** WORD SHIFT	**(@) ASFT** ASYNCHRONOUS SHIFT REGISTER	**(@) SCAN** CYCLE TIME	**(@) MCMP** MULTI-WORD COMPARE
2	**CMP** COMPARE	**(@) MOV** MOVE	**(@) MVN** MOVE NOT	**(@) BIN** BCD TO BINARY	**(@) BCD** BINARY TO BCD	**(@) ASL** SHIFT LEFT	**(@) ASR** SHIFT RIGHT	**(@) ROL** ROTATE LEFT	**(@) ROR** ROTATE RIGHT	**(@) COM** COMPLEMENT
3	**(@) ADD** BCD ADD	**(@) SUB** BCD SUBTRACT	**(@) MUL** BCD MULTIPLY	**(@) DIV** BCD DIVIDE	**(@) ANDW** LOGICAL AND	**(@) ORW** LOGICAL OR	**(@) XORW** EXCLUSIVE OR	**(@) XNRW** EXCLUSIVE NOR	**(@) INC** INCREMENT	**(@) DEC** DECREMENT
4	**(@) STC** SET CARRY	**(@) CLC** CLEAR CARRY	---	---	---	**TRSM** TRACE MEMORY SAMPLE	**(@) MSG** MESSAGE DISPLAY	**(@) LMSG** LONG MESSAGE	**(@) TERM** TERMINAL MODE	---
5	**(@) ADB** BINARY ADD	**(@) SBB** BINARY SUBTRACT	**(@) MLB** BINARY MULTIPLY	**(@) DVB** BINARY DIVIDE	**(@) ADDL** DOUBLE BCD ADD	**(@) SUBL** DOUBLE BCD SUBTRACT	**(@) MULL** DOUBLE BCD MULTIPLY	**(@) DIVL** DOUBLE BCD DIVIDE	**(@) BINL** DOUBLE BCD-TO-DOUBLE BINARY	**(@) BCDL** DOUBLE BINARY-TO-DOUBLE BCD
6	**CMPL** DOUBLE COMPARE	**(@) MPRF** TRANSFER BITS	**(@) XFRB** TRANSFER BITS	**(@) LINE** COLUMN TO LINE	**(@) COLM** LINE TO COLUMN	**(@) SEC** HOURS-TO-SECONDS	**(@) HMS** SECONDS-TO-HOURS	**(@) BCNT** BIT COUNTER	**(@) BCMP** BLOCK COMPARE	**(@) APR** ARITHMETIC PROCESS
7	**(@) XFER** BLOCK TRANSFER	**(@) BSET** BLOCK SET	**(@) ROOT** SQUARE ROOT	**(@) XCHG** DATA EXCHANGE	**(@) SLD** ONE DIGIT SHIFT LEFT	**(@) SRD** ONE DIGIT SHIFT RIGHT	**(@) MLPX** 4-TO-16 DECODER	**(@) DMPX** 16-TO-4 ENCODER	**(@) SDEC** 7-SEGMENT DECODER	**(@) FDIV** FLOATING POINT DIVIDE
8	**(@) DIST** SINGLE WORD DISTRIBUTE	**(@) COLL** DATA COLLECT	**(@) MOVB** MOVE BIT	**(@) MOVD** MOVE DIGIT	**(@) SFTR** REVERSIBLE SHIFT REGISTER	**(@) TCMP** TABLE COMPARE	**(@) ASC** ASCII CONVERT	**TTIM** TOTALIZING COUNTER	**ZCP** AREA RANGE COMPARE	**(@) INT** INTERRUPT CONTROL
9	**(@) SEND** NETWORK SEND	**(@) SBS** SUBROUTINE ENTRY	**SBN** SUBROUTINE DEFINE	**RET** SUBROUTINE RETURN	**(@) WDT** WATCHDOG TIMER REFRESH	---	---	**(@) IORF** I/O REFRESH	**(@) RECV** NETWORK RECEIVE	**(@) MCRO** MACRO

5-7-2 Alphabetic List by Mnemonic

Mnemonic	Code	Words	Name	Page
7SEG	—	4	7-SEGMENT DISPLAY OUTPUT	301
ADB (@)	50	4	BINARY ADD	219
ADBL (@)	—	4	DOUBLE BINARY ADD	225
ADD (@)	30	4	BCD ADD	205
ADDL (@)	54	4	DOUBLE BCD ADD	206
AND	None	1	AND	129
AND LD	None	1	AND LOAD	130
AND NOT	None	1	AND NOT	129
ANDW (@)	34	4	LOGICAL AND	250
APR (@)	69	4	ARITHMETIC PROCESS	239
ASC (@)	86	4	ASCII CONVERT	194
ASFT(@)	17	4	ASYNCHRONOUS SHIFT REGISTER	157

Mnemonic	Code	Words	Name	Page
ASL (@)	25	2	ARITHMETIC SHIFT LEFT	154
ASR (@)	26	2	ARITHMETIC SHIFT RIGHT	154
AVG (@)	—	4	AVERAGE VALUE	235
BCD (@)	24	3	BINARY TO BCD	181
BCDL (@)	59	3	DOUBLE BINARY-TO-DOUBLE BCD	182
BCMP (@)	68	4	BLOCK COMPARE	174
BCNT (@)	67	4	BIT COUNTER	283
BIN (@)	23	3	BCD-TO-BINARY	180
BINL (@)	58	3	DOUBLE BCD-TO-DOUBLE BINARY	181
BSET (@)	71	4	BLOCK SET	160
CLC (@)	41	1	CLEAR CARRY	205
CMP	20	3	COMPARE	170
CMPL	60	4	DOUBLE COMPARE	172
CNT	None	2	COUNTER	145
CNTR	12	3	REVERSIBLE COUNTER	148
COLL (@)	81	4	DATA COLLECT	164
COLM(@)	64	4	LINE TO COLUMN	201
COM (@)	29	2	COMPLEMENT	249
CPS	—	4	SIGNED BINARY COMPARE	178
CPSL	—	4	DOUBLE SIGNED BINARY COMPARE	179
DBS (@)	—	4	SIGNED BINARY DIVIDE	231
DBSL (@)	—	4	DOUBLE SIGNED BINARY DIVIDE	232
DEC (@)	39	2	BCD DECREMENT	204
DIFD	14	2	DIFFERENTIATE DOWN	131
DIFU	13	2	DIFFERENTIATE UP	131
DIST (@)	80	4	SINGLE WORD DISTRIBUTE	162
DIV (@)	33	4	BCD DIVIDE	212
DIVL (@)	57	4	DOUBLE BCD DIVIDE	213
DMPX (@)	77	4	16-TO-4 ENCODER	188
DSW	—	4	DIGITAL SWITCH	304
DVB (@)	53	4	BINARY DIVIDE	224
END	01	1	END	138
FAL (@)	06	2	FAILURE ALARM AND RESET	275
FALS	07	2	SEVERE FAILURE ALARM	275
FCS (@)	—	4	FCS CALCULATE	283
FDIV (@)	79	4	FLOATING POINT DIVIDE	214
FPD	—	4	FAILURE POINT DETECT	285
HEX (@)	—	4	ASCII-TO-HEXADECIMAL	195
HKY	—	4	HEXADECIMAL KEY INPUT	308
HMS (@)	66	4	SECONDS TO HOURS	184
IL	02	1	INTERLOCK	135
ILC	03	1	INTERLOCK CLEAR	135
INC (@)	38	2	INCREMENT	204
INT (@)	89	4	INTERRUPT CONTROL	262
IORF (@)	97	3	I/O REFRESH	281
JME	05	2	JUMP END	137
JMP	04	2	JUMP	137
KEEP	11	2	KEEP	133
LD	None	1	LOAD	129

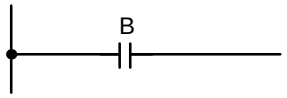
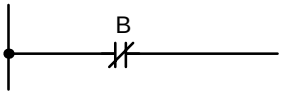
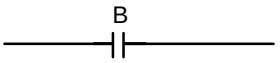
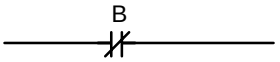
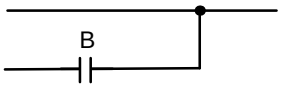
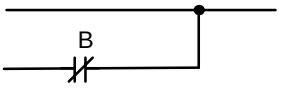
Mnemonic	Code	Words	Name	Page
LD NOT	None	1	LOAD NOT	129
LINE (@)	63	4	COLUMN TO LINE	200
LMSG (@)	47	4	32-CHARACTER MESSAGE	279
MAX (@)	—	4	FIND MAXIMUM	233
MBS (@)	—	4	SIGNED BINARY MULTIPLY	229
MBSL (@)	—	4	DOUBLE SIGNED BINARY MULTIPLY	230
MCMP (@)	19	4	MULTI-WORD COMPARE	169
MCRO (@)	99	4	MACRO	260
MIN (@)	—	4	FIND MINIMUM	234
MLB (@)	52	4	BINARY MULTIPLY	224
MLPX (@)	76	4	4-TO-16 DECODER	185
MOV (@)	21	3	MOVE	159
MOVB (@)	82	4	MOVE BIT	166
MOVD (@)	83	4	MOVE DIGIT	167
MPRF (@)	61	4	GROUP-2 HIGH-DENSITY I/O REFRESH	282
MSG (@)	46	2	MESSAGE	278
MTR	—	4	MATRIX INPUT	313
MUL (@)	32	4	BCD MULTIPLY	211
MULL (@)	56	4	DOUBLE BCD MULTIPLY	212
MVN (@)	22	3	MOVE NOT	159
NEG (@)	—	4	2'S COMPLEMENT	202
NEGL (@)	—	4	DOUBLE 2'S COMPLEMENT	203
NOP	00	1	NO OPERATION	138
OR	None	1	OR	129
OR LD	None	1	OR LOAD	130
OR NOT	None	1	OR NOT	129
ORW (@)	35	4	LOGICAL OR	251
OUT	None	2	OUTPUT	130
OUT NOT	None	2	OUTPUT NOT	130
PID (@)	—	4	PID CONTROL	242
RECV (@)	98	4	NETWORK RECEIVE (CPU31-E/33-E only)	293
RET	93	1	SUBROUTINE RETURN	259
ROL (@)	27	2	ROTATE LEFT	155
ROOT (@)	72	3	SQUARE ROOT	217
ROR (@)	28	2	ROTATE RIGHT	155
RSET	None	2	RESET	133
RXD(@)	—	4	RECEIVE	297
SBB (@)	51	4	BINARY SUBTRACT	221
SBBL (@)	—	4	DOUBLE BINARY SUBTRACT	227
SBN	92	2	SUBROUTINE DEFINE	259
SBS (@)	91	2	SUBROUTINE ENTRY	257
SCAN (@)	18	4	CYCLE TIME	276
SCL (@)	—	4	SCALING	198
SDEC (@)	78	4	7-SEGMENT DECODER	191
SEC (@)	65	4	HOURS TO SECONDS	183
SEND (@)	90	4	NETWORK SEND (CPU31-E/33-E only)	291
SET	None	2	SET	133
SFT	10	3	SHIFT REGISTER	150
SFTR (@)	84	4	REVERSIBLE SHIFT REGISTER	152

Mnemonic	Code	Words	Name	Page
SLD (@)	74	3	ONE DIGIT SHIFT LEFT	156
SNXT	09	2	STEP START	266
SRCH (@)	—	4	DATA SEARCH	289
SRD (@)	75	3	ONE DIGIT SHIFT RIGHT	156
STC (@)	40	1	SET CARRY	205
STEP	08	2	STEP DEFINE	266
SUB (@)	31	4	BCD SUBTRACT	207
SUBL (@)	55	4	DOUBLE BCD SUBTRACT	209
SUM (@)	—	4	SUM CALCULATION	237
TCMP (@)	85	4	TABLE COMPARE	175
TERM (@)	48	4	TERMINAL MODE	280
TIM	None	2	TIMER	139
TIMH	15	3	HIGH-SPEED TIMER	143
TKY (@)	—	4	TEN KEY INPUT	311
TRSM	45	1	TRACE MEMORY SAMPLE	277
TTIM	87	4	TOTALIZING TIMER	144
TXD (@)	—	4	TRANSMIT	299
WDT (@)	94	2	WATCHDOG TIMER REFRESH	281
WSFT (@)	16	3	WORD SHIFT	157
XCHG (@)	73	3	DATA EXCHANGE	162
XDMR (@)	—	4	EXPANSION DM READ	290
XFER (@)	70	4	BLOCK TRANSFER	161
XFRB (@)	62	4	TRANSFER BITS	168
XNRW (@)	37	4	EXCLUSIVE NOR	253
XORW (@)	36	4	EXCLUSIVE OR	252
ZCP	88	4	AREA RANGE COMPARE	176
ZCPL	—	4	DOUBLE AREA RANGE COMPARE	177

5-8 Ladder Diagram Instructions

Ladder Diagram instructions include Ladder instructions and Logic Block instructions and correspond to the conditions on the ladder diagram. Logic block instructions are used to relate more complex parts.

5-8-1 LOAD, LOAD NOT, AND, AND NOT, OR, and OR NOT

	Ladder Symbols	Operand Data Areas
LOAD – LD		**B**: Bit IR, SR, AR, HR, TC, LR, TR
LOAD NOT – LD NOT		**B**: Bit IR, SR, AR, HR, TC, LR
AND – AND		**B**: Bit IR, SR, AR, HR, TC, LR
AND NOT – AND NOT		**B**: Bit IR, SR, AR, HR, TC, LR
OR – OR		**B**: Bit IR, SR, AR, HR, TC, LR
OR NOT – OR NOT		**B**: Bit IR, SR, AR, HR, TC, LR

Limitations

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 memory capacity of the PC is not exceeded.

Description

These six basic instructions correspond to the conditions on a ladder diagram. As described in *Section 4 Writing and Inputting the Program*, 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 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. The ladder symbol for loading TR bits is different from that shown above. Refer to *4-4-3 Ladder Instructions* for details.

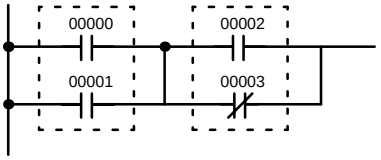
Flags

There are no flags affected by these instructions.

5-8-2 AND LOAD and OR LOAD

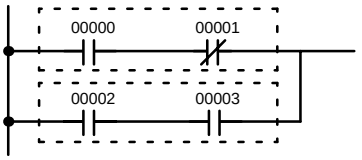
AND LOAD – AND LD

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.

In order to draw ladder diagrams, it is not necessary to use AND LD and OR LD instructions, nor are they necessary when inputting ladder diagrams directly, as is possible from LSS. They are required, however, to convert the program to and input it in mnemonic form. The procedures for these, limitations for different procedures, and examples are provided in *4-7 Inputting, Modifying, and Checking the Program*.

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 *4-4-6 Logic Block Instructions*.

Flags

There are no flags affected by these instructions.

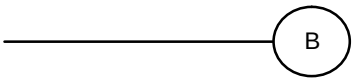
5-9 Bit Control Instructions

There are five instructions that can be used generally to control individual bit status. These are OUT, OUT NOT, DIFU(13), DIFD(14), and KEEP(11). These instructions are used to turn bits ON and OFF in different ways.

5-9-1 OUTPUT and OUTPUT NOT – OUT and OUT NOT

OUTPUT – OUT

Ladder Symbol

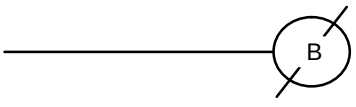


Operand Data Areas

B: Bit
IR, SR, AR, HR, TC, LR, TR

OUTPUT NOT – OUT NOT

Ladder Symbol



Operand Data Areas

B: Bit
IR, SR, AR, HR, TC, LR

Limitations

Any output bit can generally be used in only one instruction that controls its status. Refer to *3-3 IR Area* for details.

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 *4-7-7 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 *5-14-1 TIMER – TIM* for details.

Flags

There are no flags affected by these instructions.

5-9-2 DIFFERENTIATE UP and DOWN – DIFU(13) and DIFD(14)

Ladder Symbols	Operand Data Areas
DIFU(13) B	**B**: Bit IR, AR, HR, LR
DIFD(14) B	**B**: Bit IR, AR, HR, LR

Limitations

Any output bit can generally be used in only one instruction that controls its status. Refer to *3-3 IR Area* for details.

Description

DIFU(13) and DIFD(14) are used to turn the designated bit ON for one cycle only.

Whenever executed, DIFU(13) compares its current execution with the previous execution condition. If the previous execution condition was OFF and the current one is ON, DIFU(13) will turn ON the designated bit. If the previous execution condition was ON and the current execution condition is either ON or OFF, DIFU(13) 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(14) compares its current execution with the previous execution condition. If the previous execution condition was ON and the current one is OFF, DIFD(14) will turn ON the designated bit. If the previous execution condition was OFF and the current execution condition is either ON or OFF, DIFD(14) 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 an @) 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.

Flags

There are no flags affected by these instructions.

Precautions

DIFU(13) and DIFD(14) operation can be uncertain when the instructions are programmed between IL and ILC, between JMP and JME, or in subroutines. Refer to *5-10 INTERLOCK and INTERLOCK CLEAR – IL(02) and ILC(03)*, *5-11 JUMP and JUMP END – JMP(04) and JME(05)*, and *5-23 Subroutines and Interrupt Control* for details.

Example 1: When There is No Differentiated Instruction

In diagram A, below, whenever CMP(20) is executed with an ON execution condition it will compare the contents of the two operand words (HR 10 and DM 0000) and set the arithmetic flags (GR, EQ, and LE) accordingly. If the execution condition remains ON, flag status may be changed each cycle if the content of one or both operands change. Diagram B, however, is an example of how DIFU(13) can be used to ensure that CMP(20) is executed only once each time the desired execution condition goes ON.

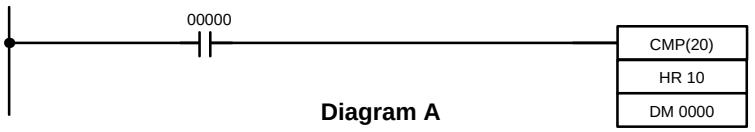


Diagram A

Address	Instruction	Operands
00000	LD	00000
00001	CMP(20)	
		HR 10
		DM 0000

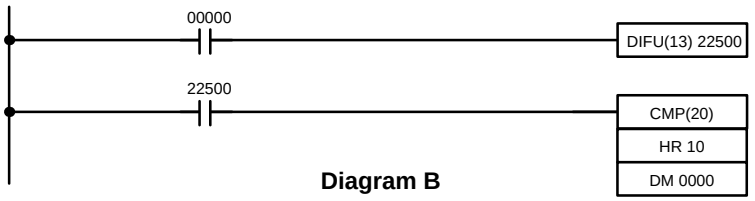
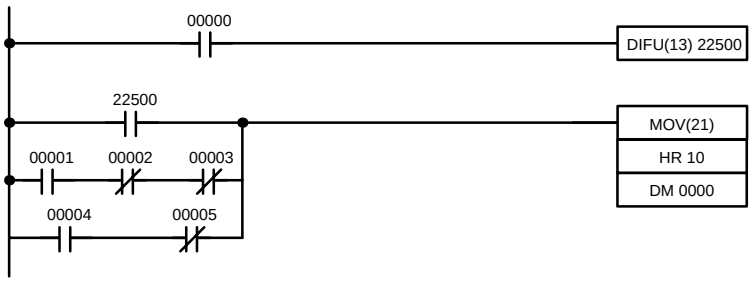


Diagram B

Address	Instruction	Operands
00000	LD	00000
00001	DIFU(13)	22500
00002	LD	22500
00003	CMP(20)	
		HR 10
		DM 0000

Example 2: Simplifying Programming

Although a differentiated form of MOV(21) 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(21) requires differentiated treatment.



Address	Instruction	Operands
00000	LD	00000
00001	DIFU(13)	22500
00002	LD	22500
00003	LD	00001
00004	AND NOT	00002
00005	AND NOT	00003
00006	OR LD	---
00007	LD	00004
00008	AND NOT	00005
00009	OR LD	---
00010	MOV(21)	
		HR 10
		DM 0000

5-9-3 SET and RESET – SET and RSET

Ladder Symbols

SET B

RSET B

Operand Data Areas

B: Bit
IR, SR, AR, HR, LR

B: Bit
IR, SR, AR, HR, LR

Description

SET 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 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 differs from that of OUT because the OUT instruction turns the operand bit OFF when its execution condition is OFF. Likewise, RSET differs from OUT NOT because OUT NOT turns the operand bit ON when its execution condition is OFF.

Precautions

The status of operand bits for SET and RSET programmed between IL(02) and ILC(03) or JMP(04) and JME(05) will not change when the interlock or jump condition is met (i.e., when IL(02) or JMP(04) is executed with an OFF execution condition).

Flags

There are no flags affected by these instructions.

Examples

The following examples demonstrate the difference between OUT and SET/RSET. In the first example (Diagram A), IR 10000 will be turned ON or OFF whenever IR 00000 goes ON or OFF.

In the second example (Diagram B), IR 10000 will be turned ON when IR 00001 goes ON and will remain ON (even if IR 00001 goes OFF) until IR 00002 goes ON.

Diagram A

Address	Instruction	Operands
00000	LD	00000
00001	OUT	10000

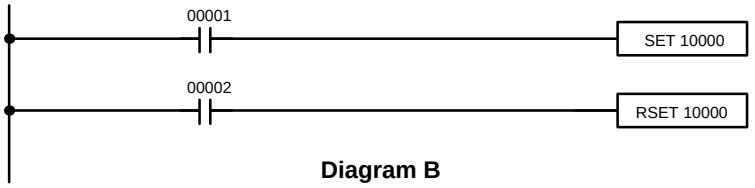
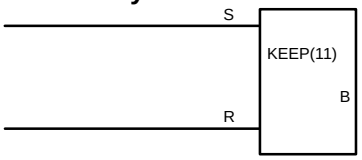


Diagram B

Address	Instruction	Operands
00000	LD	00001
00001	SET	10000
00002	LD	00002
00003	RSET	10000

5-9-4 KEEP – KEEP(11)

Ladder Symbol



Operand Data Areas

B: Bit
IR, AR, HR, LR

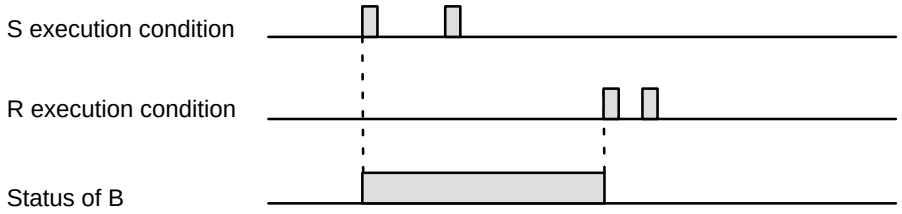
Limitations

Any output bit can generally be used in only one instruction that controls its status. Refer to *3-3 IR Area* for details.

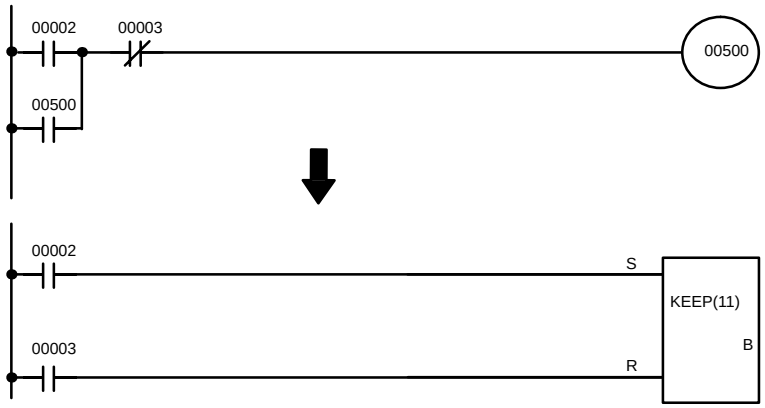
Description

KEEP(11) 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(11) 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(11) bit status is shown below.



KEEP(11) operates like the self-maintaining bit described in *4-8-3 Self-maintaining Bits*. The following two diagrams would function identically, though the one using KEEP(11) requires one less instruction to program and would maintain status even in an interlocked program section.



Address	Instruction	Operands
00000	LD	00002
00001	OR	00500
00002	AND NOT	00003
00003	OUT	00500

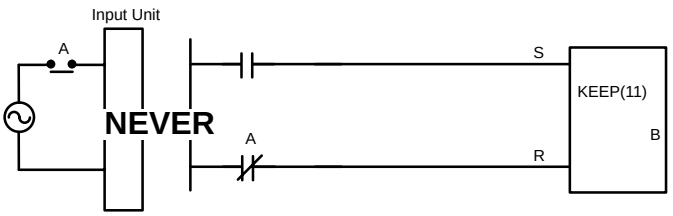
Address	Instruction	Operands
00000	LD	00002
00001	LD	00003
00002	KEEP(11)	00500

Flags

There are no flags affected by this instruction.

Precautions

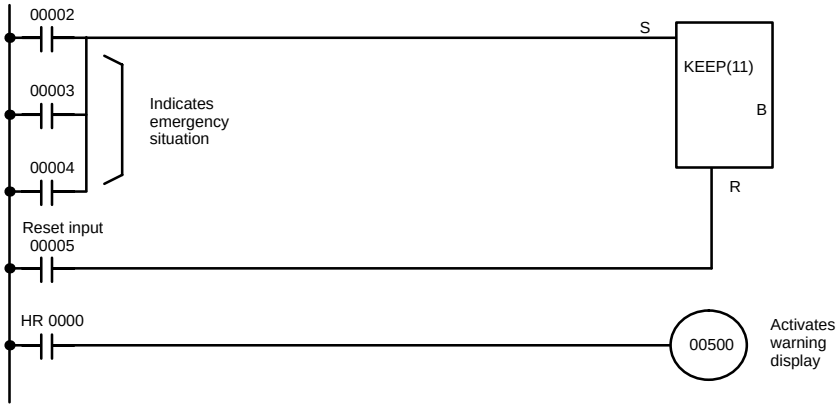
Exercise caution when using a KEEP reset line that is controlled by an external normally closed device. Never use an input bit in an inverse condition on the reset (R) for KEEP(11) when the input device uses an AC power supply. The delay in shutting down the PC's DC power supply (relative to the AC power supply to the input device) can cause the designated bit of KEEP(11) to be reset. This situation is shown below.



Bits used in KEEP are not reset in interlocks. Refer to the *5-10 INTERLOCK – and INTERLOCK CLEAR IL(02) and ILC(03)* for details.

Example

If a HR bit or an AR bit is used, bit status will be retained even during a power interruption. KEEP(11) 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 00002, 00003, and 00004 would be turned ON to indicate some type of error. Bit 00005 would be turned ON to reset the warning display. HR 0000, which is turned ON when any one of the three bits indicates an emergency situation, is used to turn ON the warning indicator through 00500.

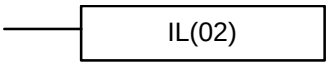


Address	Instruction	Operands
00000	LD	00002
00001	OR	00003
00002	OR	00004
00003	LD	00005
00004	KEEP(11)	HR 0000
00005	LD	HR 0000
00006	OUT	00500

KEEP(11) can also be combined with TIM to produce delays in turning bits ON and OFF. Refer to *5-14-1 TIMER – TIM* for details.

5-10 INTERLOCK and INTERLOCK CLEAR – IL(02) and ILC(03)

Ladder Symbol



Ladder Symbol

Description

IL(02) is always used in conjunction with ILC(03) to create interlocks. Interlocks are used to enable branching in the same way as can be achieved with TR bits, but treatment of instructions between IL(02) and ILC(03) differs from that with TR bits when the execution condition for IL(02) is OFF. If the execution condition of IL(02) is ON, the program will be executed as written, with an ON execution condition used to start each instruction line from the point where IL(02) is located through the next ILC(03). Refer to *4-7-7 Branching Instruction Lines* for basic descriptions of both methods.

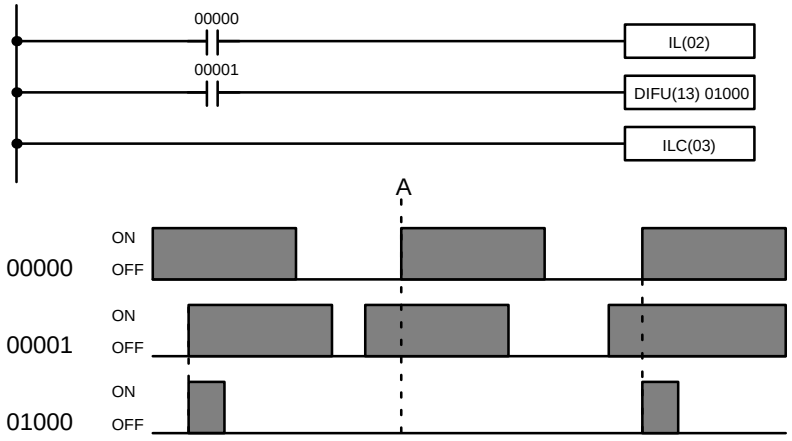
If the execution condition for IL(02) is OFF, the interlocked section between IL(02) and ILC(03) will be treated as shown in the following table:

Instruction	Treatment
OUT and OUT NOT	Designated bit turned OFF.
SET and RSET	Bit status maintained.
TIM and TIMH(15)	Reset.
TTIM(87)	PV maintained.
CNT, CNTR(12)	PV maintained.
KEEP(11)	Bit status maintained.
DIFU(13) and DIFD(14)	Not executed (see below).
All others	Not executed.

IL(02) and ILC(03) do not necessarily have to be used in pairs. IL(02) can be used several times in a row, with each IL(02) creating an interlocked section through the next ILC(03). ILC(03) cannot be used unless there is at least one IL(02) between it and any previous ILC(03).

DIFU(13) and DIFD(14) in Interlocks

Changes in the execution condition for a DIFU(13) or DIFD(14) are not recorded if the DIFU(13) or DIFD(14) is in an interlocked section and the execution condition for the IL(02) is OFF. When DIFU(13) or DIFD(14) is execution in an interlocked section immediately after the execution condition for the IL(02) has gone ON, the execution condition for the DIFU(13) or DIFD(14) will be compared to the execution condition that existed before the interlock became effective (i.e., before the interlock condition for IL(02) went OFF). The ladder diagram and bit status changes for this are shown below. The interlock is in effect while 00000 is OFF. Notice that 01000 is not turned ON at the point labeled A even though 00001 has turned OFF and then back ON.



Address	Instruction	Operands
00000	LD	00000
00001	IL(02)	
00002	LD	00001
00003	DIFU(13)	01000
00004	ILC(03)	

Precautions

There must be an ILC(03) following any one or more IL(02).

Although as many IL(02) instructions as are necessary can be used with one ILC(03), ILC(03) instructions cannot be used consecutively without at least one IL(02) in between, i.e., nesting is not possible. Whenever a ILC(03) is executed, all interlocks between the active ILC(03) and the preceding ILC(03) are cleared.

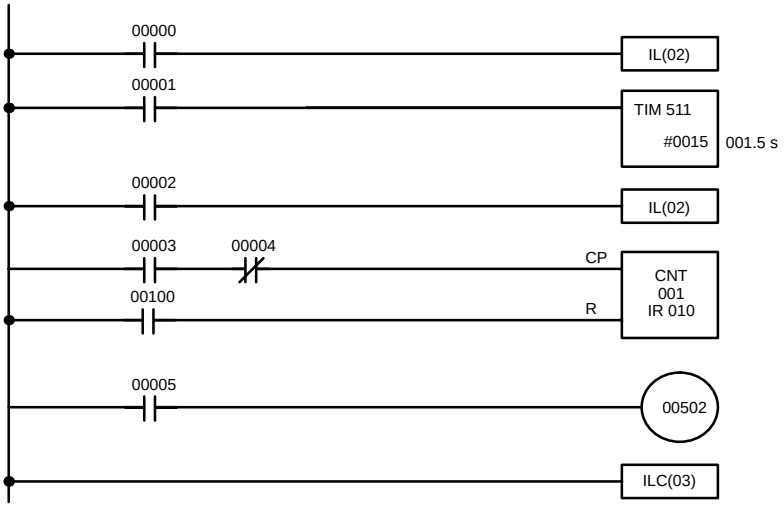
When more than one IL(02) is used with a single ILC(03), an error message will appear when the program check is performed, but execution will proceed normally.

Flags

There are no flags affected by these instructions.

Example

The following diagram shows IL(02) being used twice with one ILC(03).



Address	Instruction	Operands
00000	LD	00000
00001	IL(02)	
00002	LD	00001
00003	TIM	511
		# 0015
00004	LD	00002
00005	IL(02)	
00006	LD	00003
00007	AND NOT	00004
00008	CNT	001
		010
00009	LD	00005
00010	OUT	00502
00011	ILC(03)	

When the execution condition for the first IL(02) is OFF, TIM 511 will be reset to 1.5 s, CNT 001 will not be changed, and 00502 will be turned OFF. When the execution condition for the first IL(02) is ON and the execution condition for the second IL(02) is OFF, TIM 511 will be executed according to the status of 00001, CNT 001 will not be changed, and 00502 will be turned OFF. When the execution conditions for both the IL(02) are ON, the program will execute as written.

5-11 JUMP and JUMP END – JMP(04) and JME(05)

Ladder Symbols	Definer Values
JMP(04) N	**N**: Jump number # (00 to 99)
JME(05) N	**N**: Jump number # (00 to 99)

Limitations

Jump numbers 01 through 99 may be used only once in JMP(04) and once in JME(05), i.e., each can be used to define one jump only. Jump number 00 can be used as many times as desired.

Description

JMP(04) is always used in conjunction with JME(05) to create jumps, i.e., to skip from one point in a ladder diagram to another point. JMP(04) defines the point from which the jump will be made; JME(05) defines the destination of the jump. When the execution condition for JMP(04) in ON, no jump is made and the program is executed consecutively as written. When the execution condition for JMP(04) is OFF, a jump is made to the JME(05) with the same jump number and the instruction following JME(05) is executed next.

If the jump number for JMP(04) is between 01 and 99, jumps, when made, will go immediately to JME(05) with the same jump number without executing any instructions in between. The status of timers, counters, bits used in OUT, bits used in OUT NOT, and all other status bits controlled by the instructions between JMP(04) and JMP(05) will not be changed. Each of these jump numbers can be used to define only one jump. Because all of instructions between JMP(04) and JME(05) are skipped, jump numbers 01 through 99 can be used to reduce cycle time.

If the jump number for JMP(04) is 00, the CPU will look for the next JME(05) with a jump number of 00. To do so, it must search through the program, causing a longer cycle time (when the execution condition is OFF) than for other jumps. The status of timers, counters, bits used in OUT, bits used in OUT NOT, and all other status controlled by the instructions between JMP(04) 00 and JMP(05) 00 will not be changed. jump number 00 can be used as many times as desired. A jump from JMP(04) 00 will always go to the next JME(05) 00 in the program. It is thus possible to use JMP(04) 00 consecutively and match them all with the same JME(05) 00. It makes no sense, however, to use JME(05) 00 consecutively, because all jumps made to them will end at the first JME(05) 00.

DIFU(13) and DIFD(14) in Jumps

Although DIFU(13) and DIFD(14) are designed to turn ON the designated bit for one cycle, they will not necessarily do so when written between JMP(04) and JMP (05). Once either DIFU(13) or DIFD(14) has turned ON a bit, it will remain ON until the next time DIFU(13) or DIFD(14) is executed again. In normal programming, this means the next cycle. In a jump, this means the next time the jump from JMP(04) to JME(05) is not made, i.e., if a bit is turned ON by DIFU(13) or DIFD(14) and then a jump is made in the next cycle so that DIFU(13) or DIFD(14) are skipped, the designated bit will remain ON until the next time the execution condition for the JMP(04) controlling the jump is ON.

Precautions

When JMP(04) and JME(05) are not used in pairs, an error message will appear when the program check is performed. Although this message also appears if JMP(04) 00 and JME(05) 00 are not used in pairs, the program will execute properly as written.

Flags

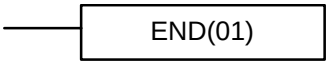
There are no flags affected by these instructions.

Examples

Examples of jump programs are provided in *4-7-8 Jumps*.

5-12 END – END(01)

Ladder Symbol



Description

END(01) is required as the last instruction in any program. If there are subroutines, END(01) is placed after the last subroutine. No instruction written after END(01) will be executed. END(01) can be placed anywhere in the program to execute all instructions up to that point, as is sometimes done to debug a program, but it must be removed to execute the remainder of the program.

If there is no END(01) in the program, no instructions will be executed and the error message “NO END INST” will appear.

Flags

END(01) turns OFF the ER, CY, GR, EQ, and LE Flags.

5-13 NO OPERATION – NOP(00)

Description

NOP(00) is not generally required in programming and there is no ladder symbol for it. When NOP(00) 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(00) is written at all addresses. NOP(00) can be input through the 00 function code.

Flags

There are no flags affected by NOP(00).

5-14 Timer and Counter Instructions

TIM and TIMH are decrementing ON-delay timer instructions which require a TC number and a set value (SV).

CNT is a decrementing counter instruction and CNTR is a reversible counter instruction. Both require a TC number and a SV. Both are also connected to multiple instruction lines which serve as an input signal(s) and a reset.

Any one TC number cannot be defined twice, i.e., once it has been used as the definer in any of the timer or counter instructions, it cannot be used again. Once defined, TC numbers can be used as many times as required as operands in instructions other than timer and counter instructions.

TC numbers run from 000 through 511. No prefix is required when using a TC number as a definer in a timer or counter instruction. Once defined as a timer, a TC number can be prefixed with TIM for use as an operand in certain instructions. The TIM prefix is used regardless of the timer instruction that was used to define the timer. Once defined as a counter, a TC number can be prefixed with CNT for use as an operand in certain instructions. The CNT is also used regardless of the counter instruction that was used to define the counter.

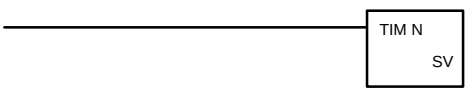
TC numbers can be designated as operands that require either bit or word data. When designated as an operand that requires bit data, the TC number accesses a bit that functions as a 'Completion Flag' that indicates when the time/count has expired, i.e., the bit, which is normally OFF, will turn ON when the designated SV has expired. When designated as an operand that requires word data, the TC 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(20), or any other instruction for which the TC area is allowed. This is done by designating the TC number used to define that timer or counter to access the memory location that holds the PV.

Note that "TIM 000" is used to designate the TIMER instruction defined with TC number 000, to designate the Completion Flag for this timer, and to designate the PV of this timer. The meaning of the term in context should be clear, i.e., the first is always an instruction, the second is always a bit operand, and the third is always a word operand. The same is true of all other TC numbers prefixed with TIM or CNT.

An SV can be input as a constant or as a word address in a data area. If an IR 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 and counters wired in this way can only be set externally during RUN or MONITOR mode. All SVs, including those set externally, must be in BCD.

5-14-1 TIMER – TIM

Ladder Symbol



Definer Values

N: TC number
(000 through 511)

Operand Data Areas

SV: Set value (word, BCD)
IR, AR, DM, HR, LR, #

Limitations

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

Each TC number can be used as the definer in only one TIMER or COUNTER instruction.

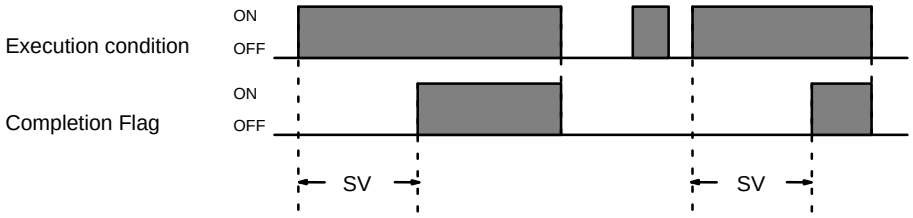
TC 000 through TC 015 should not be used in TIM if they are required for TIMH(15). Refer to *5-14-2 HIGH-SPEED TIMER – TIMH(15)* for details.

Description

A timer is activated when its execution condition goes ON and is reset (to SV) when the execution condition goes OFF. Once activated, TIM measures in units of 0.1 second from the SV.

If the execution condition remains ON long enough for TIM to time down to zero, the Completion Flag for the TC number used will turn ON and will remain ON until TIM is reset (i.e., until its execution condition is goes OFF).

The following figure illustrates the relationship between the execution condition for TIM and the Completion Flag assigned to it.



Precautions

Timers in interlocked program sections are reset when the execution condition for IL(02) is OFF. Power interruptions also reset timers. If a timer that is not reset under these conditions is desired, SR area clock pulse bits can be counted to produce timers using CNT. Refer to *5-14-4 COUNTER – CNT* for details.

Program execution will continue even if a non-BCD SV is used, but timing will not be accurate.

The SV of the timers can be set in the range #0000 to #9999 (BCD). If the SV for a timer is set to #0000 or #0001, it will operate in the following way. If the SV is set to #0000, when the timer input goes from OFF to ON, the Completion Flag will turn ON. If the SV is set to #0001, because the timer accuracy is 0 to –0.1 s, the actual time will be a value between 0 and 0.1 s, and the Completion Flag may turn ON as soon as the timer input goes from OFF to ON. With other values also, allow for a timer accuracy of 0 to –0.1 s when setting the SV.

Flags

ER: SV is not in BCD.

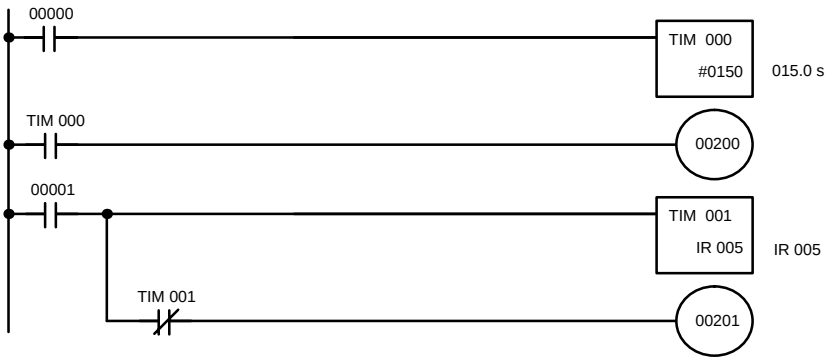
Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

Examples

All of the following examples use OUT in diagrams that would generally be used to control output bits in the IR area. There is no reason, however, why these diagrams cannot be modified to control execution of other instructions.

Example 1: Basic Application

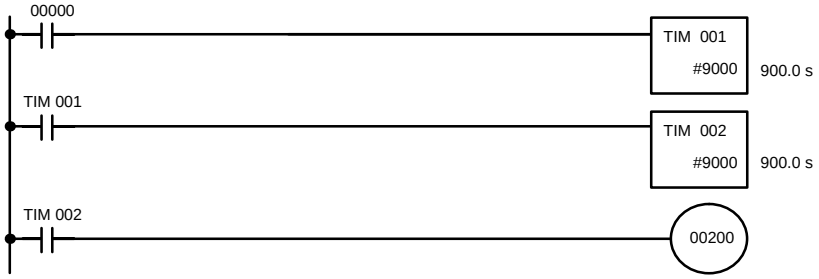
The following example shows two timers, one set with a constant and one set via input word 005. Here, 00200 will be turned ON after 00000 goes ON and stays ON for at least 15 seconds. When 00000 goes OFF, the timer will be reset and 00200 will be turned OFF. When 00001 goes ON, TIM 001 is started from the SV provided through IR word 005. Bit 00201 is also turned ON when 00001 goes ON. When the SV in 005 has expired, 00201 is turned OFF. This bit will also be turned OFF when TIM 001 is reset, regardless of whether or not SV has expired.



Address	Instruction	Operands
00000	LD	00000
00001	TIM	000
		# 0150
00002	LD	TIM 000
00003	OUT	00200
00004	LD	00001
00005	TIM	001
		005
00006	AND NOT	TIM 001
00007	OUT	00200

Example 2: Extended Timers

There are two ways to achieve timers that operate for longer than 999.9 seconds. One method 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.



Address	Instruction	Operands
00000	LD	00000
00001	TIM	001
		# 9000
00002	LD	TIM 001
00003	TIM	002
		# 9000
00004	LD	TIM 002
00005	OUT	00200

In this example, 00200 will be turned ON 30 minutes after 00000 goes ON.

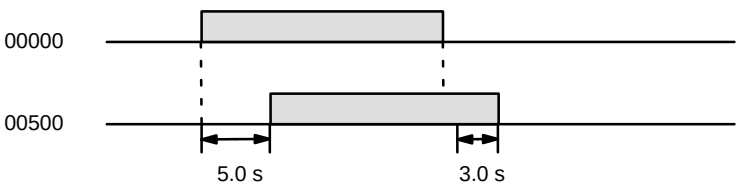
TIM can also be combined with CNT or CNT can be used to count SR area clock pulse bits to produce longer timers. An example is provided in *5-14-4 COUNTER – CNT*.

Example 3: ON/OFF Delays

TIM can be combined with KEEP(11) to delay turning a bit ON and OFF in reference to a desired execution condition. KEEP(11) is described in *5-9-4 KEEP – KEEP(11)*.

To create delays, the Completion Flags for two TIM are used to determine the execution conditions for setting and reset the bit designated for KEEP(11). The bit whose manipulation is to be delayed is used in KEEP(11). Turning ON and OFF the bit designated for KEEP(11) is thus delayed by the SV for the two TIM. The two SV could naturally be the same if desired.

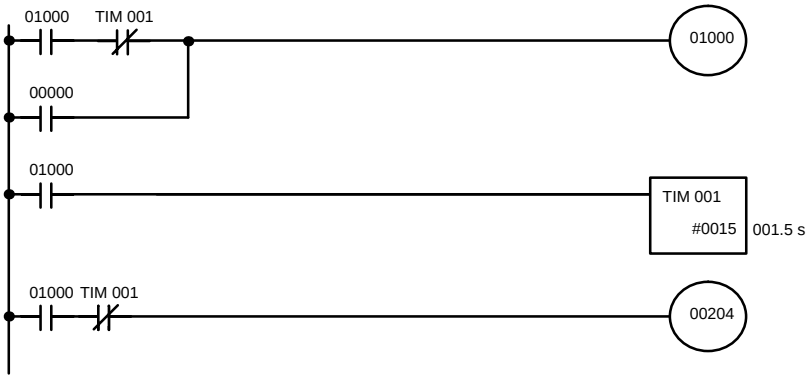
In the following example, 00500 would be turned ON 5.0 seconds after 00000 goes ON and then turned OFF 3.0 seconds after 00000 goes OFF. It is necessary to use both 00500 and 00000 to determine the execution condition for TIM 002; 00000 in an inverse condition is necessary to reset TIM 002 when 00000 goes ON and 00500 is necessary to activate TIM 002 (when 00000 is OFF).



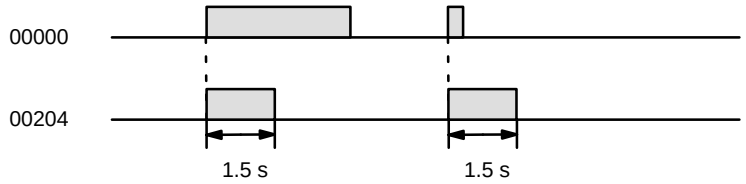
Address	Instruction	Operands
00000	LD	00000
00001	TIM	001
		# 0050
00002	LD	00500
00003	AND NOT	00000
00004	TIM	002
		# 0030
00005	LD	TIM 001
00006	LD	TIM 002
00007	KEEP(11)	00500

Example 4: One-Shot Bits

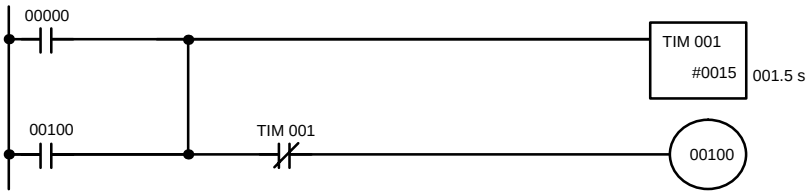
The length of time that a bit is kept ON or OFF can be controlled by combining TIM with OUT or OUT NO. The following diagram demonstrates how this is possible. In this example, 00204 would remain ON for 1.5 seconds after 00000 goes ON regardless of the time 00000 stays ON. This is achieved by using 01000 as a self-maintaining bit activated by 00000 and turning ON 00204 through it. When TIM 001 comes ON (i.e., when the SV of TIM 001 has expired), 00204 will be turned OFF through TIM 001 (i.e., TIM 001 will turn ON which, as an inverse condition, creates an OFF execution condition for OUT 00204).



Address	Instruction	Operands
00000	LD	01000
00001	AND NOT	TIM 001
00002	OR	00000
00003	OUT	01000
00004	LD	01000
00005	TIM	001
		# 0015
00006	LD	01000
00007	AND NOT	TIM 001
00008	OUT	00204



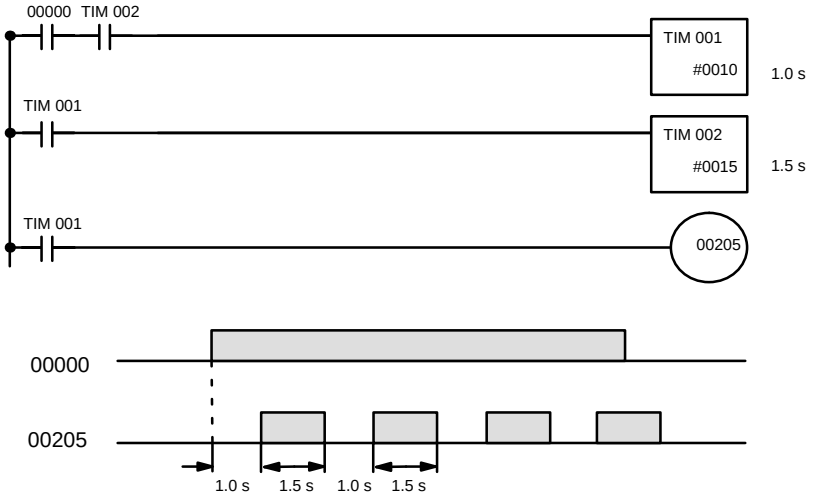
The following one-shot timer may be used to save memory.



Address	Instruction	Operands
00000	LD	00000
00001	OR	00100
00002	TIM	001
		# 0015
00003	AND NOT	TIM 001
00004	OUT	00100

Example 5: Flicker Bits

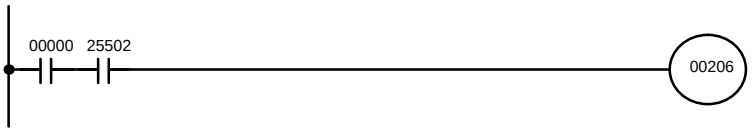
Bits can be programmed to turn ON and OFF at regular intervals while a designated execution condition is ON by using TIM twice. One TIM functions to turn ON and OFF a specified bit, i.e., the Completion Flag of this TIM turns the specified bit ON and OFF. The other TIM functions to control the operation 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	00000
00001	AND	TIM 002
00002	TIM	001
		# 0010
00003	LD	TIM 001
00004	TIM	002
		# 0015
00005	LD	TIM 001
00006	OUT	00205

A simpler but less flexible method of creating a flicker bit is to AND one of the SR 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 SR area.

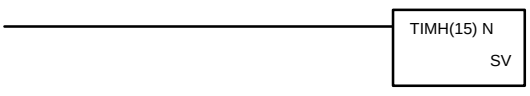
In the following example the 1-second clock pulse is used (25502) so that 00206 would be turned ON and OFF every second, i.e., it would be ON for 0.5 seconds and OFF for 0.5 seconds. Precise timing and the initial status of 00206 would depend on the status of the clock pulse when 00000 goes ON.



Address	Instruction	Operands
00000	LD	00000
00001	LD	25502
00002	OUT	00206

5-14-2 HIGH-SPEED TIMER – TIMH(15)

Ladder Symbol



Definer Values

N: TC number
(000 through 015 preferred)

Operand Data Areas

SV: Set value (word, BCD)
IR, AR, DM, HR, LR, #

Limitations

SV is between 00.00 and 99.99. (Although 00.00 and 00.01 may be set, 00.00 will disable the timer, i.e., turn ON the Completion Flag immediately, and 00.01 is not reliably cycled.) The decimal point is not entered.

Each TC number can be used as the definer in only one TIMER or COUNTER instruction.

If the cycle time is greater than 10 ms, use TC 000 through TC 015.

Description

TIMH(15) operates in the same way as TIM except that TIMH measures in units of 0.01 second.

The cycle time affects TIMH(15) accuracy if TC 016 through TC 511 are used. If the cycle time is greater than 10 ms, use TC 000 through TC 015.

Refer to *5-14-1 TIMER – TIM* for operational details and examples. Except for the above, and all aspects of operation are the same.

Precautions

Timers in interlocked program sections are reset when the execution condition for IL(02) is OFF. Power interruptions also reset timers. If a timer that is not reset under these conditions is desired, SR area clock pulse bits can be counted to produce timers using CNT. Refer to *5-14-4 COUNTER – CNT* for details.

Program execution will continue even if a non-BCD SV is used, but timing will not be accurate.

The SV of the timers can be set in the range #0000 to #9999 (BCD). If the SV for a timer is set to #0000 or #0001, it will operate in the following way. If the SV is set to #0000, when the timer input goes from OFF to ON, the Completion Flag will turn ON. There may be a time delay if TC 000 to TC 003 are used. If the SV is set to #0001, because the timer accuracy is 0 to –0.1 s, the actual time will be a value between 0 and 0.1 s, and the Completion Flag may turn ON as soon as the timer input goes from OFF to ON. With other values also, allow for a timer accuracy of 0 to –0.1 s when setting the SV.

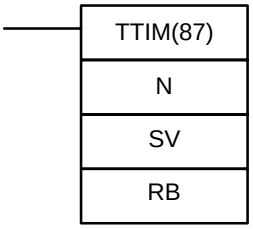
Flags

ER: SV is not in BCD.

Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

5-14-3 TOTALIZING TIMER – TTIM(87)

Ladder Symbol



Definer Values

N: TC number
(000 through 511)

Operand Data Areas

SV: Set value (word, BCD)
IR, AR, DM, HR, LR

RB: Reset bit
IR, SR, AR, HR, LR

Limitations

SV is between 0000 and 9999 (000.0 and 999.9 s) and must be in BCD. The decimal point is not entered.

Each TC number can be used as the definer in only one TIMER or COUNTER instruction.

Description

TTIM(87) is used to create a timer that increments the PV every 0.1 s to time between 0.1 and 999.9 s. TTIM(87) increments in units of 0.1 second from zero. TTIM(87) accuracy is +0.0/–0.1 second. A TTIM(87) timer will time as long as its execute condition is ON until it reaches the SV or until RB turns ON to reset the timer. TIMM(87) timers will time only as long as they are executed every cycle, i.e., they do not time, but maintain the current PV, in interlocked program sections or when they are jumped in the program.

Precautions

The PVs of totalizing timers in interlocked program sections are maintained when the execution condition for IL(02) is OFF. Unlike timers and high-speed timers, totalizing timers in jumped program sections do not continue timing, but maintain the PV.

Power interruptions will reset timers.

Totalizing timers will not restart after timing out unless the PV is changed to a value below the SV or the reset input is turned ON.

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

The SV of the timers can be set in the range #0000 to #9999 (BCD). If the SV for a timer is set to #0000 or #0001, it will operate in the following way. If the SV is set to #0000, when the timer input goes from OFF to ON, the Completion Flag will turn ON. If the SV is set to #0001, because the timer accuracy is 0 to –0.1 s, the actual time will be a value between 0 and 0.1 s, and the Completion Flag may turn ON as soon as the timer input goes from OFF to ON. With other values also, allow for a timer accuracy of 0 to –0.1 s when setting the SV.

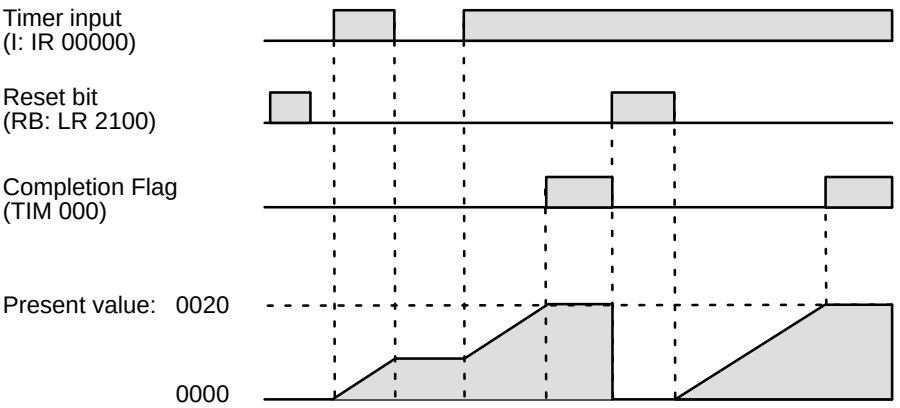
Flags

ER (SR 25503): Content of ∗DM word is not BCD when set for BCD.
SV is not BCD.

Example

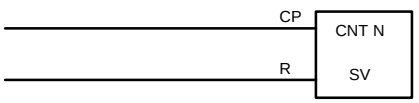
The following figure illustrates the relationship between the execution conditions for a totalizing timer with a set value of 2 s, its PV, and the Completion Flag.

Address	Instruction	Operands	
00000	LD		00000
00001	TTIM(87)		
		TIM	000
		#	0020
		LR	2100



5-14-4 COUNTER – CNT

Ladder Symbol



Definer Values

N: TC number
(000 through 511)

Operand Data Areas

SV: Set value (word, BCD)
IR, AR, DM, HR, LR, #

Limitations

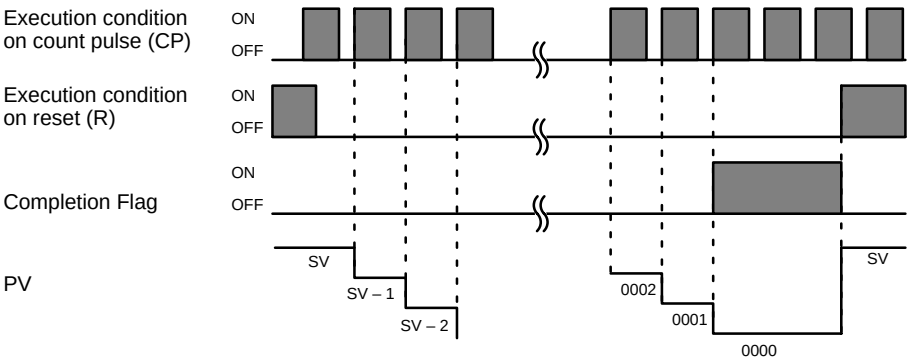
Each TC number can be used as the definer in only one TIMER or COUNTER instruction.

Description

CNT is used to count down from 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 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.

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.



Precautions

Program execution will continue even if a non-BCD SV is used, but the SV will not be correct.

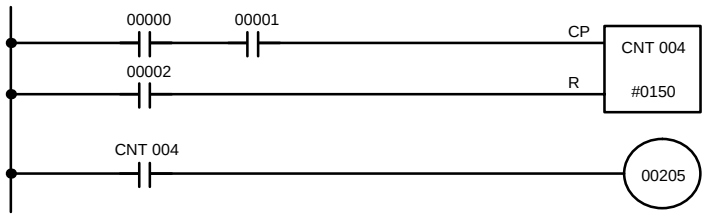
Flags

ER: SV is not in BCD.

Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

Example 1: Basic Application

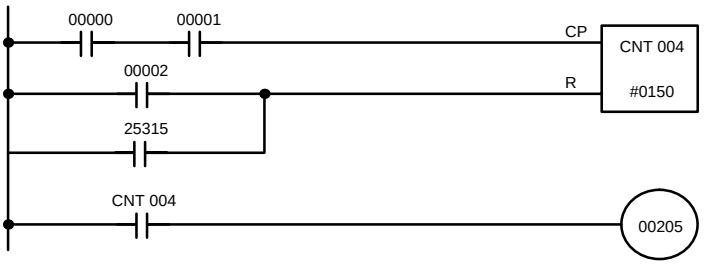
In the following example, the PV will be decremented whenever both 00000 and 00001 are ON provided that 00002 is OFF and either 00000 or 00001 was OFF the last time CNT 004 was executed. When 150 pulses have been counted down (i.e., when PV reaches zero), 00205 will be turned ON.



Address	Instruction	Operands
00000	LD	00000
00001	AND	00001
00002	LD	00002
00003	CNT	0004
		# 0150
00004	LD	CNT 004
00005	OUT	00205

Here, 00000 can be used to control when CNT is operative and 00001 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 SR area (25315) to reset CNT as shown below.



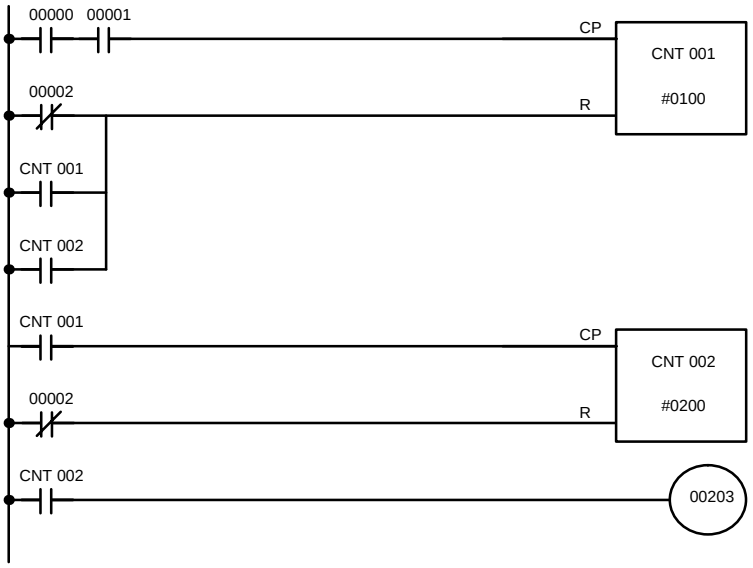
Address	Instruction	Operands
00000	LD	00000
00001	AND	00001
00002	LD	00002
00003	OR	25315
00004	CNT	004
		# 0150
00005	LD	CNT 004
00006	OUT	00205

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, 00000 is used to control when CNT 001 operates. CNT 001, when 00000 is ON, counts down the number of OFF to ON changes in 00001. CNT 001 is reset by its Completion Flag, i.e., it starts counting again as soon as its PV reaches zero. CNT 002 counts the number of times the Completion Flag for CNT 001 goes ON. Bit 00002 serves as a reset for the entire extended counter, resetting both CNT 001 and CNT 002 when it is OFF. The Completion Flag for CNT 002 is also used to reset CNT 001 to inhibit CNT 001 operation, once SV for CNT 002 has been reached, until the entire extended counter is reset via 00002.

Because in this example the SV for CNT 001 is 100 and the SV for CNT 002 is 200, the Completion Flag for CNT 002 turns ON when 100 x 200 or 20,000 OFF to ON changes have been counted in 00001. This would result in 00203 being turned ON.



Address	Instruction	Operands
00000	LD	00000
00001	AND	00001
00002	LD NOT	00002
00003	OR	CNT 001
00004	OR	CNT 002
00005	CNT	001
		# 0100
00006	LD	CNT 001
00007	LD NOT	00002
00008	CNT	002
		# 0200
00009	LD	CNT 002
00010	OUT	00203

CNT can be used in sequence as many times as required to produce counters capable of counting any desired values.

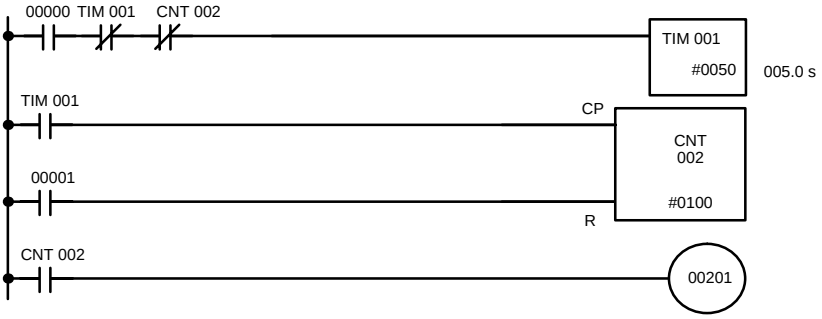
Example 3: Extended Timers

CNT can be used to create extended timers in two ways: by combining TIM with CNT and by counting SR area clock pulse bits.

In the following example, CNT 002 counts the number of times TIM 001 reaches zero from its SV. The Completion Flag for TIM 001 is used to reset TIM 001 so that it runs continuously and CNT 002 counts the number of times the Completion Flag for TIM 001 goes ON (CNT 002 would be executed once each time be-

tween when the Completion Flag for TIM 001 goes ON and TIM 001 is reset by its Completion Flag). TIM 001 is also reset by the Completion Flag for CNT 002 so that the extended timer would not start again until CNT 002 was reset by 00001, which serves as the reset for the entire extended timer.

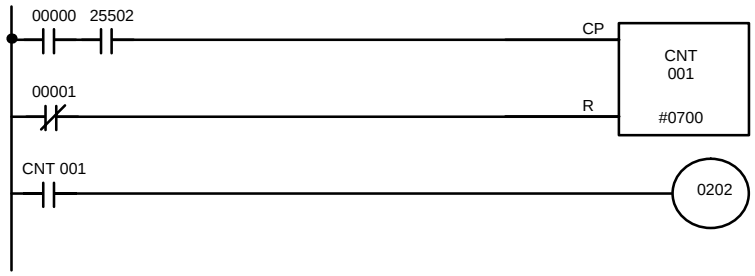
Because in this example the SV for TIM 001 is 5.0 seconds and the SV for CNT 002 is 100, the Completion Flag for CNT 002 turns ON when 5 seconds x 100 times, i.e., 500 seconds (or 8 minutes and 20 seconds) have expired. This would result in 00201 being turned ON.



Address	Instruction	Operands	
00000	LD		00000
00001	AND NOT	TIM	001
00002	AND NOT	CNT	002
00003	TIM		001
		#	0050
00004	LD	TIM	001
00005	LD		00001
00006	CNT		002
		#	0100
00007	LD	CNT	002
00008	OUT		00201

In the following example, CNT 001 counts the number of times the 1-second clock pulse bit (25502) goes from OFF to ON. Here again, 00000 is used to control the times when CNT is operating.

Because in this example the SV for CNT 001 is 700, the Completion Flag for CNT 002 turns ON when 1 second x 700 times, or 11 minutes and 40 seconds have expired. This would result in 00202 being turned ON.

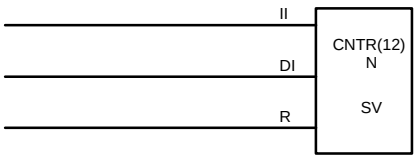


Address	Instruction	Operands	
00000	LD		00000
00001	AND		25502
00002	LD NOT		00001
00003	CNT		001
		#	0700
00004	LD	CNT	001
00005	OUT		00202

⚠ 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 with CNT instructions.

5-14-5 REVERSIBLE COUNTER – CNTR(12)

Ladder Symbol



Definer Values

N: TC number
(000 through 511)

Operand Data Areas

SV: Set value (word, BCD)
IR, AR, DM, HR, LR, #

Limitations

Each TC number can be used as the definer in only one TIMER or COUNTER instruction.

Description

The CNTR(12) 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 in the increment input (II) and those in the decrement input (DI).

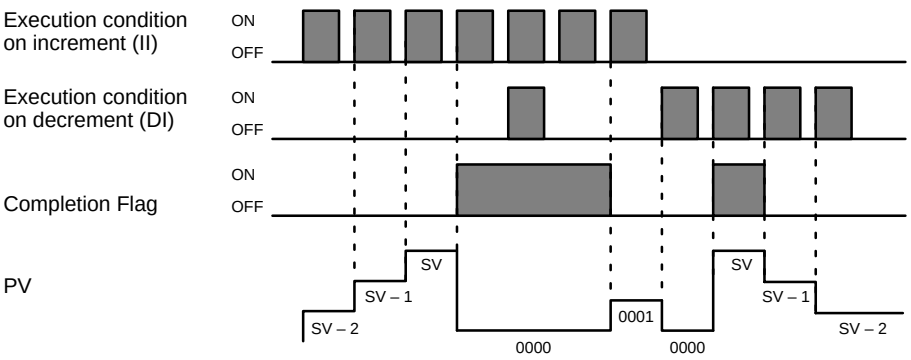
The present value (PV) will be incremented by one whenever CNTR(12) is executed with an ON execution condition for II and the last execution condition for II was OFF. The present value (PV) will be decremented by one whenever CNTR(12) is executed with an ON execution condition for DI and the last execution 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 CNT will not be changed.

When decremented from 0000, the present value is set to SV and the Completion Flag is turned ON until the PV is decremented again. When incremented past the SV, the PV is set to 0000 and the Completion Flag is turned ON until the PV is incremented again.

CNTR(12) 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(12) will not be reset in interlocked program sections or by the effects of power interruptions.

Changes in II and DI execution conditions, the Completion Flag, and the PV are illustrated below starting from part way through CNTR(12) operation (i.e., when reset, counting begins from zero). PV line height is meant to indicate changes in the PV only.



Precautions

Program execution will continue even if a non-BCD SV is used, but the SV will not be correct.

Flags

ER: SV is not in BCD.

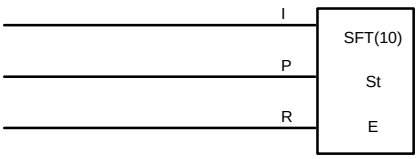
Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

5-15 Data Shifting

All of the instructions described in this section are used to shift data, but in differing amounts and directions. The first shift instruction, SFT(10), shifts an execution condition into a shift register; the rest of the instructions shift data that is already in memory.

5-15-1 SHIFT REGISTER – SFT(10)

Ladder Symbol



Operand Data Areas

St: Starting word	
IR, SR, AR, HR, LR	
E: End word	
IR, SR, AR, HR, LR	

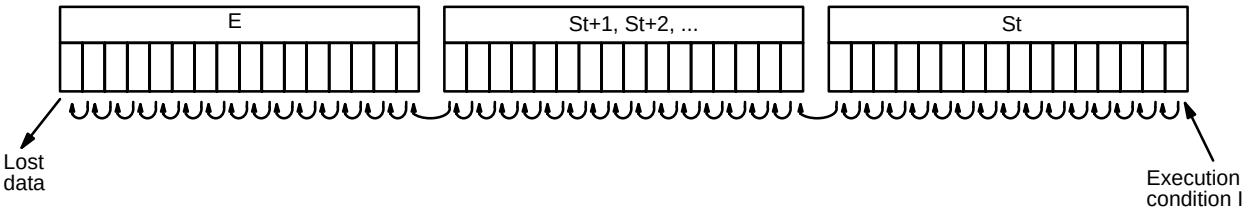
Limitations

St must be less than or equal to E, and 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(11)), an error ("COIL DUPL") will be generated when program syntax is checked on the Programming Console or another Programming Device. The program, however, will be executed as written. See *Example 2: Controlling Bits in Shift Registers* for a programming example that does this.

Description

SFT(10) is controlled by three execution conditions, I, P, and R. If SFT(10) 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(10) 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.

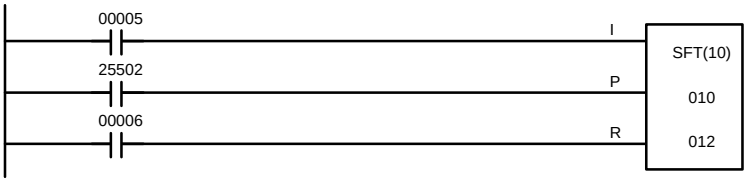
When execution condition R goes ON, all bits in the shift register will be turned OFF (i.e., set to 0) and the shift register will not operate until R goes OFF again.

Flags

There are no flags affected by SFT(10).

Example 1: Basic Application

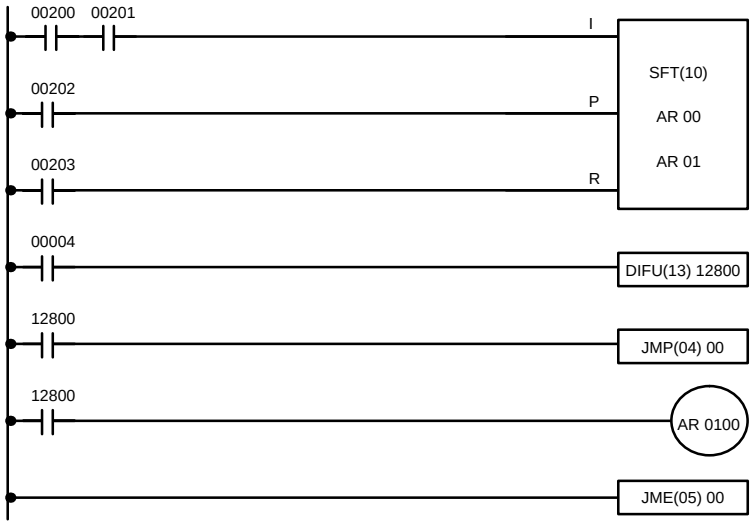
The following example uses the 1-second clock pulse bit (25502) so that the execution condition produced by 00005 is shifted into a 3-word register between IR 010 and IR 012 every second.



Address	Instruction	Operands
00000	LD	00005
00001	LD	25502
00002	LD	00006
00003	SFT(10)	
		010
		012

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 AR 00 through AR 01. When the 17th bit is to be set, 00004 is turned ON. This causes the jump for JMP(04) 00 not to be made for that one cycle, and AR 0100 (the 17th bit) will be turned ON. When 12800 is OFF (i.e., at all times except during the first cycle after 00004 has changed from OFF to ON), the jump is executed and the status of AR 0100 will not be changed.



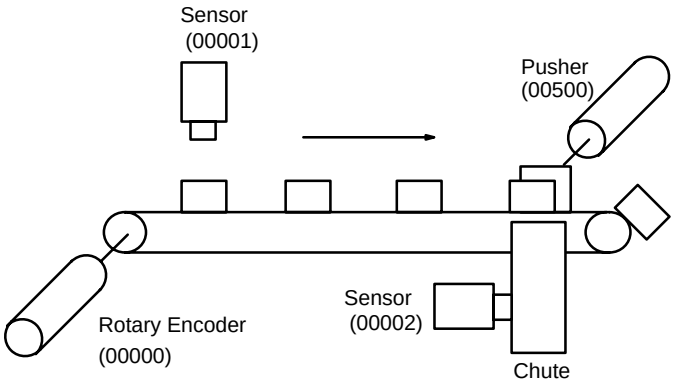
Address	Instruction	Operands
00000	LD	00200
00001	AND	00201
00002	LD	00202
00003	LD	00203
00004	SFT(10)	
		AR 00
		AR 01
00005	LD	00004
00006	DIFU(13)	12800
00007	LD	12800
00008	JMP(04)	00
00009	LD	12800
00010	OUT	AR 0100
00011	JME(05)	00

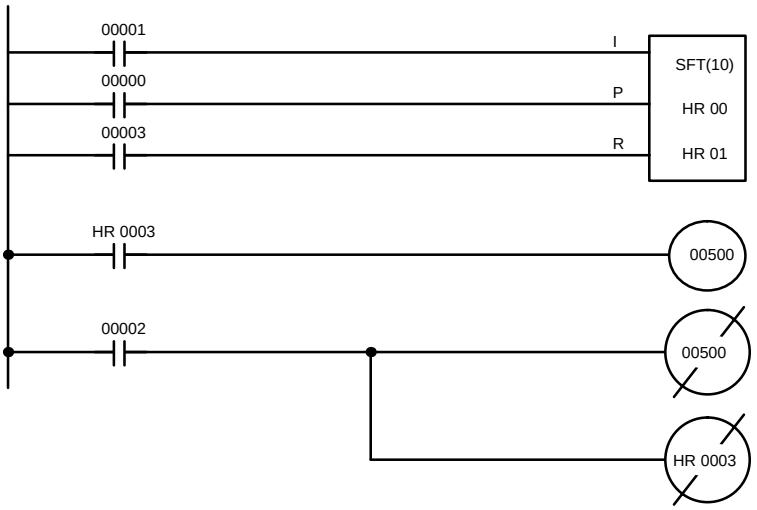
When a bit that is part of a shift register is used in OUT (or any other instruction that controls bit status), a syntax error will be generated during the program check, but the program will executed properly (i.e., as written).

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 shoot. To do this, the execution condition determined by inputs from the first sensor (00001) are stored in a shift register: ON for good products; OFF for faulty ones. Conveyor speed has been adjusted so that HR 0003 of the shift register can be used to activate a pusher (00500) when a faulty product reaches it, i.e., when HR 0003 turns ON, 00500 is turned ON to activate the pusher.

The program is set up so that a rotary encoder (00000) controls execution of SFT(10) through a DIFU(13), the rotary encoder is set up to turn ON and OFF each time a product passes the first sensor. Another sensor (00002) is used to detect faulty products in the shoot so that the pusher output and HR 0003 of the shift register can be reset as required.





Address	Instruction	Operands
00000	LD	00001
00001	LD	00000
00002	LD	00003
00003	SFT(10)	
		HR 00
		HR 01
00004	LD	HR 0003
00005	OUT	00500
00006	LD	00002
00007	OUT NOT	00500
00008	OUT NOT	HR 0003

5-15-2 REVERSIBLE SHIFT REGISTER – SFTR(84)

Ladder Symbols

SFTR(84)
C
St
E

@SFTR(84)
C
St
E

Operand Data Areas

C: Control word
IR, AR, DM, HR, LR

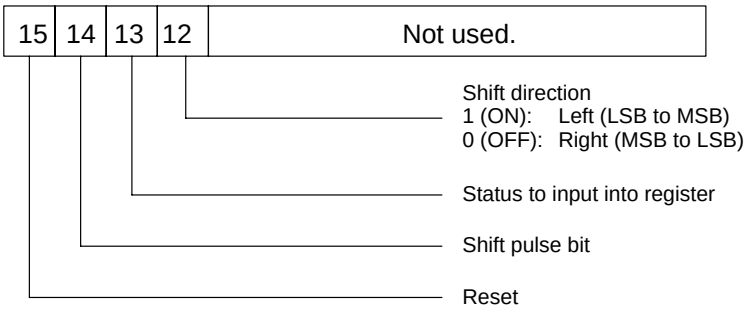
St: Starting word
IR, SR, AR, DM, HR, LR

E: End word
IR, SR, AR, DM, HR LR

Limitations St and E must be in the same data area and St must be less than or equal to E.

Description

SFTR(84) 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(84) 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(84) is executed with an OFF execution condition or if SFTR(84) is executed with bit 14 OFF, the shift register will remain unchanged. If SFTR(84) 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.

Flags

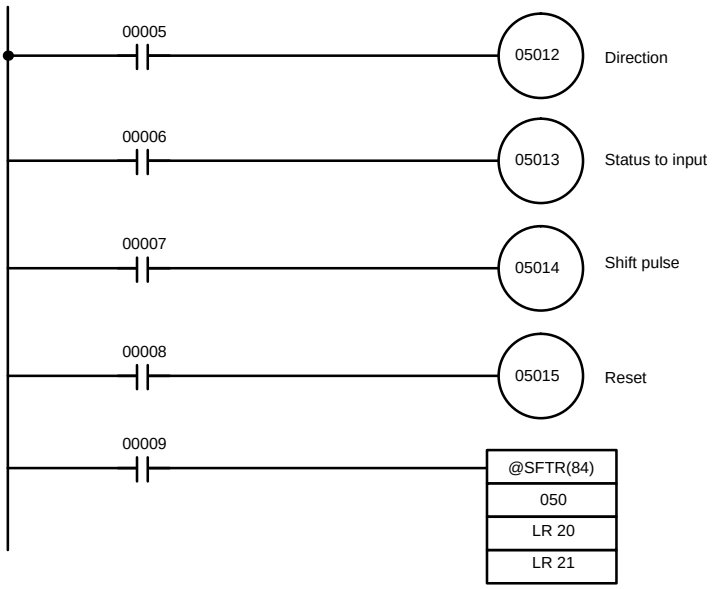
ER: St and E are not in the same data area or ST is greater than E.

Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

CY: Receives the status of bit 00 of St or bit 15 of E, depending on the shift direction.

Example

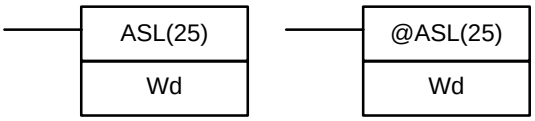
In the following example, IR 00005, IR 00006, IR 00007, and IR 00008 are used to control the bits of C used in @SHIFT(84). The shift register is between LR 20 and LR 21, and it is controlled through IR 00009.



Address	Instruction	Operands
00000	LD	00005
00001	OUT	05012
00002	LD	00006
00003	OUT	05013
00004	LD	00007
00005	OUT	00514
00006	LD	00008
00007	OUT	05015
00008	LD	00009
00009	@SFT(10)	
		050
		LR 20
		LR 21

5-15-3 ARITHMETIC SHIFT LEFT – ASL(25)

Ladder Symbols

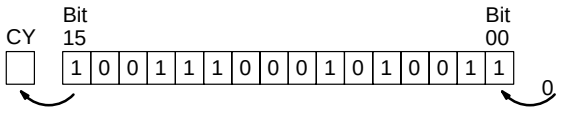


Operand Data Areas

Wd: Shift word
IR, SR, AR, DM, HR, LR

Description

When the execution condition is OFF, ASL(25) is not executed. When the execution condition is ON, ASL(25) 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

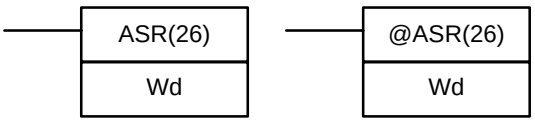
ER: Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

CY: Receives the status of bit 15.

EQ: ON when the content of Wd is zero; otherwise OFF.

5-15-4 ARITHMETIC SHIFT RIGHT – ASR(26)

Ladder Symbols

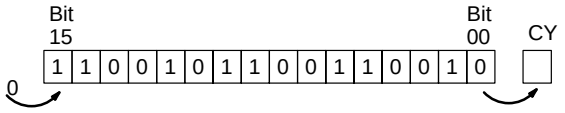


Operand Data Areas

Wd: Shift word
IR, SR, AR, DM, HR, LR

Description

When the execution condition is OFF, ASR(25) is not executed. When the execution condition is ON, ASR(25) 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

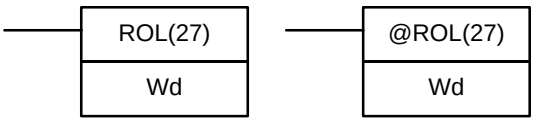
ER: Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

CY: Receives the data of bit 00.

EQ: ON when the content of Wd is zero; otherwise OFF.

5-15-5 ROTATE LEFT – ROL(27)

Ladder Symbols

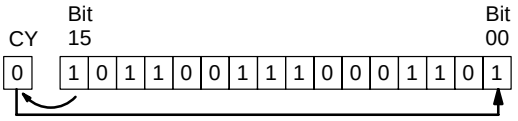


Operand Data Areas

Wd: Rotate word
IR, SR, AR, DM, HR, LR

Description

When the execution condition is OFF, ROL(27) is not executed. When the execution condition is ON, ROL(27) 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(41) to set the status of CY or CLC(41) to clear the status of CY before doing a rotate operation to ensure that CY contains the proper status before execution ROL(27).

The status of CY is cleared at the end of each cycle (when END(01) is executed).

Flags

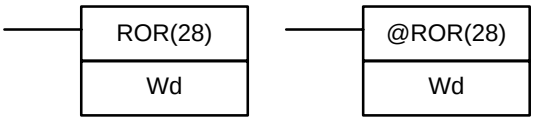
ER: Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

CY: Receives the data of bit 15.

EQ: ON when the content of Wd is zero; otherwise OFF.

5-15-6 ROTATE RIGHT – ROR(28)

Ladder Symbols

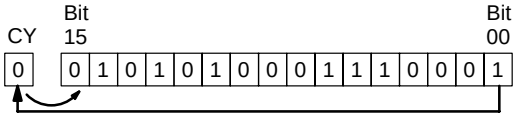


Operand Data Areas

Wd: Rotate word
IR, SR, AR, DM, HR, LR

Description

When the execution condition is OFF, ROR(28) is not executed. When the execution condition is ON, ROR(28) 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(41) to set the status of CY or CLC(41) to clear the status of CY before doing a rotate operation to ensure that CY contains the proper status before execution ROR(28).

The status of CY is cleared at the end of each cycle (when END(01) is executed).

Flags

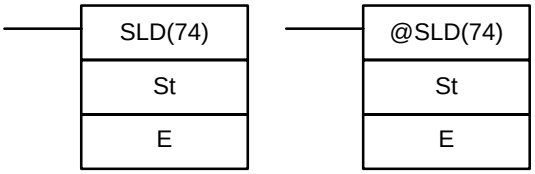
ER: Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

CY: Receives the data of bit 15.

EQ: ON when the content of Wd is zero; otherwise OFF.

5-15-7 ONE DIGIT SHIFT LEFT – SLD(74)

Ladder Symbols



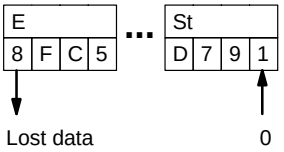
Operand Data Areas

St: Starting word
IR, SR, AR, DM, HR, LR

E: End word
IR, SR, AR, DM, HR, LR

Limitations St and E must be in the same data area, and St must be less than or equal to E.

Description When the execution condition is OFF, SLD(74) is not executed. When the execution condition is ON, SLD(74) 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 If a power failure occurs during a shift operation across more than 50 words, the shift operation might not be completed.

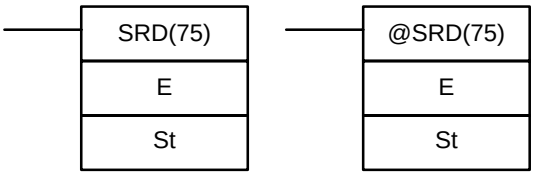
Flags

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

Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

5-15-8 ONE DIGIT SHIFT RIGHT – SRD(75)

Ladder Symbols



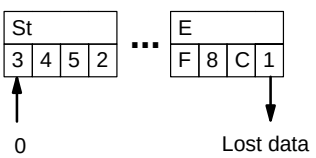
Operand Data Areas

E: End word
IR, SR, AR, DM, HR, LR

St: Starting word
IR, SR, AR, DM, HR, LR

Limitations St and E must be in the same data area, and St must be less than or equal to E.

Description When the execution condition is OFF, SRD(75) is not executed. When the execution condition is ON, SRD(75) shifts data between St and E (inclusive) by one digit (four bits) to the right. 0 is written into the leftmost digit of St and the rightmost digit of E is lost.



Precautions

If a power failure occurs during a shift operation across more than 50 words, the shift operation might not be completed. Set the range between E and St to a maximum of 50 words.

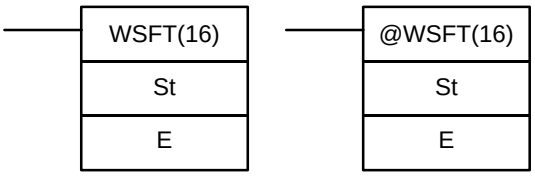
Flags

ER: The St and E words are in different areas, or St is less than E.

Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

5-15-9 WORD SHIFT – WSFT(16)

Ladder Symbols



Operand Data Areas

St: Starting word
IR, SR, AR, DM, HR, LR

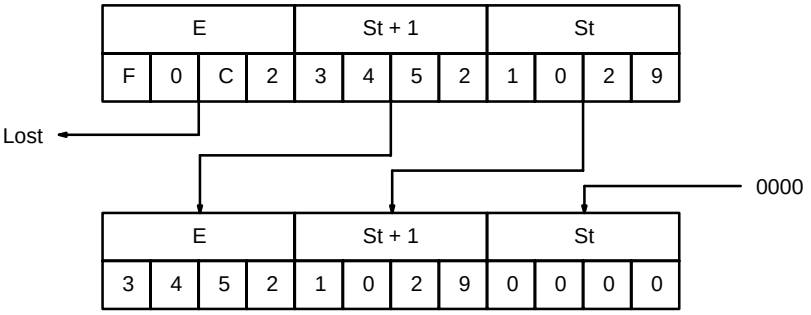
E: End word
IR, SR, AR, DM, HR, LR

Limitations

St and E must be in the same data area, and St must be less than or equal to E.

Description

When the execution condition is OFF, WSFT(16) is not executed. When the execution condition is ON, WSFT(16) shifts data between St and E in word units. Zeros are written into St and the content of E is lost.



Flags

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

Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

5-15-10 ASYNCHRONOUS SHIFT REGISTER – ASFT(17)

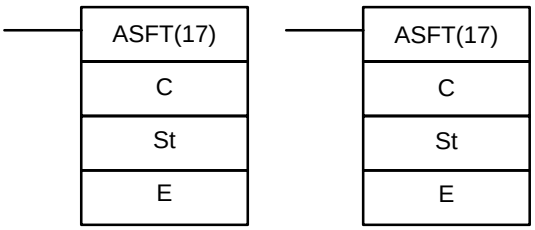
Operand Data Areas

C: Control word
IR, SR, AR, DM, HR, LR

St: Starting word
IR, SR, AR, DM, HR, LR

E: End word
IR, SR, AR, DM, HR, LR

Ladder Symbols



Limitations

St and E must be in the same data area, and St must be less than or equal to E.

Description

When the execution condition is OFF, ASFT(17) does nothing and the program moves to the next instruction. When the execution condition is ON, ASFT(17) is used to create and control a reversible asynchronous word shift register between St and E. This register only shifts words when the next word in the register is zero, e.g., if no words in the register contain zero, nothing is shifted. Also, only one word is shifted for each word in the register that contains zero. When the contents of a word are shifted to the next word, the original word's contents are set to zero. In essence, when the register is shifted, each zero word in the register trades places with the next word. (See *Example* below.)

The shift direction (i.e. whether the "next word" is the next higher or the next lower word) is designated in C. C is also used to reset the register. All of any portion of the register can be reset by designating the desired portion with St and E.

Control Word

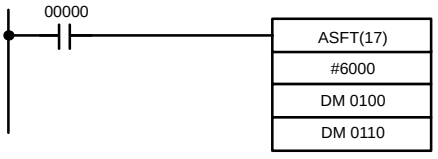
Bits 00 through 12 of C are not used. Bit 13 is the shift direction: turn bit 13 ON to shift down (toward lower addressed words) and OFF to shift up (toward higher addressed words). Bit 14 is the Shift Enable Bit: turn bit 14 ON to enable shift register operation according to bit 13 and OFF to disable the register. Bit 15 is the Reset bit: the register will be reset (set to zero) between St and E when ASFT(17) is executed with bit 15 ON. Turn bit 15 OFF for normal operation.

Flags

ER: The St and E words are in different areas, or St is greater than E.
Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

Example

The following example shows instruction ASFT(17) used to shift words in an 11-word shift register created between DM 0100 and DM 0110 with a control word value of #6000 (bits 13 and 14 ON). The data changes that would occur for the given register and control word contents are also shown.



Address	Instruction	Operands
00100	LD	00000
00101	ASFT(17)	
		# 6000
		DM 0100
		DM 0110

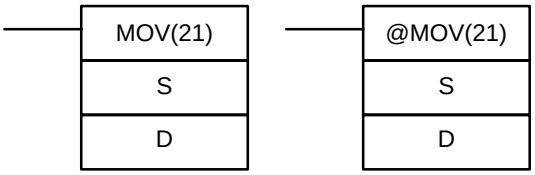
	Before execution	After 1 execution	After 7 executions
DM 0100	1234	1234	1234
DM 0101	0000	0000	2345
DM 0102	0000	2345	3456
DM 0103	2345	0000	4567
DM 0104	3456	3456	5678
DM 0105	0000	4567	6789
DM 0106	4567	0000	789A
DM 0107	5678	5678	0000
DM 0108	6789	6789	0000
DM 0109	0000	789A	0000
DM 0110	789A	0000	0000

5-16 Data Movement

This section describes the instructions 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 Link Systems 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.

5-16-1 MOVE – MOV(21)

Ladder Symbols



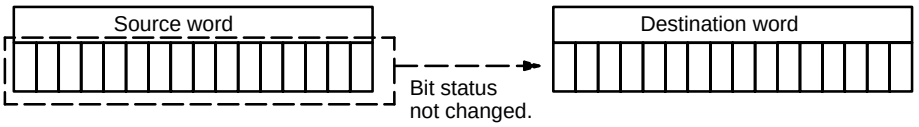
Operand Data Areas

S: Source word
IR, SR, AR, DM, HR, TC, LR, #

D: Destination word
IR, SR, AR, DM, HR, LR

Description

When the execution condition is OFF, MOV(21) is not executed. When the execution condition is ON, MOV(21) copies the content of S to D.



Precautions

TC numbers cannot be designated as D to change the PV of the timer or counter. You can, however, easily change the PV of a timer or a counter by using BSET(71).

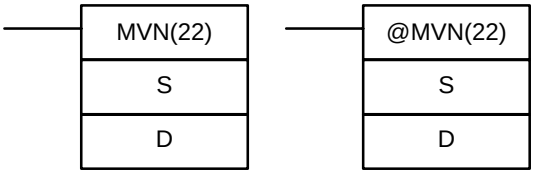
Flags

ER: Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON when all zeros are transferred to D.

5-16-2 MOVE NOT – MVN(22)

Ladder Symbols



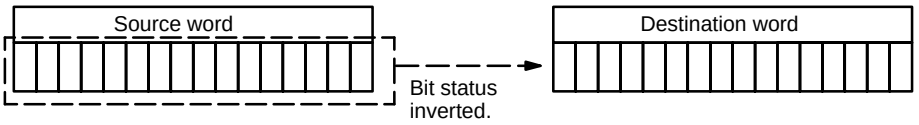
Operand Data Areas

S: Source word
IR, SR, AR, DM, HR, TC, LR, #

D: Destination word
IR, SR, AR, DM, HR, LR

Description

When the execution condition is OFF, MVN(22) is not executed. When the execution condition is ON, MVN(22) transfers the inverted 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.



Precautions

TC numbers cannot be designated as D to change the PV of the timer or counter. However, these can be easily changed using BSET(71).

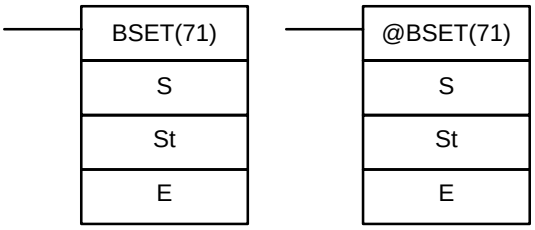
Flags

ER: Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON when all zeros are transferred to D.

5-16-3 BLOCK SET – BSET(71)

Ladder Symbols



Operand Data Areas

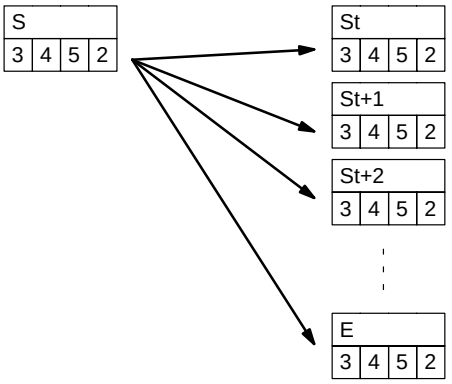
S: Source data
IR, SR, AR, DM, HR, TC, LR, #
St: Starting word
IR, SR, AR, DM, HR, TC, LR
E: End Word
IR, SR, AR, DM, HR, TC, LR

Limitations

St must be less than or equal to E, and St and E must be in the same data area.

Description

When the execution condition is OFF, BSET(71) is not executed. When the execution condition is ON, BSET(71) copies the content of S to all words from St through E.



BSET(71) can be used to change timer/counter PV. (This cannot be done with MOV(21) or MVN(22).) BSET(71) can also be used to clear sections of a data area, i.e., the DM area, to prepare for executing other instructions.

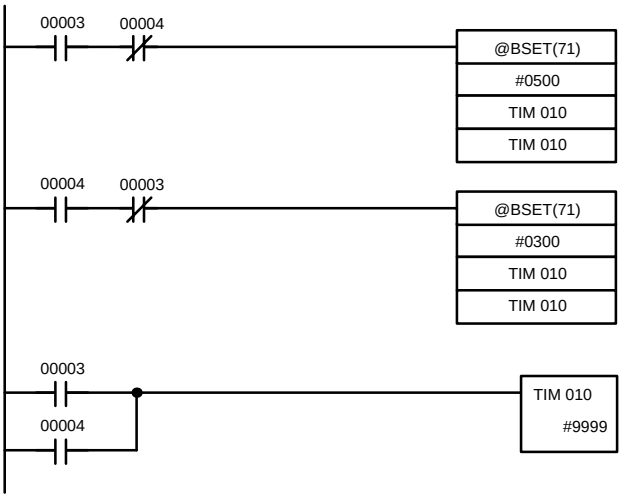
Flags

ER: St and E are not in the same data area or St is greater than E.

Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

Example

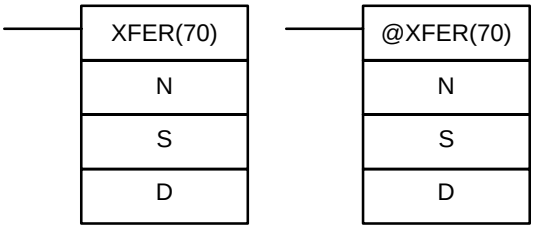
The following example shows how to use BSET(71) to change the PV of a timer depending on the status of IR 00003 and IR 00004. When IR 00003 is ON, TIM 010 will operate as a 50-second timer; when IR 00004 is ON, TIM 010 will operate as a 30-second timer.



Address	Instruction	Operands
00000	LD	00003
00001	AND NOT	00004
00002	@BSET(71)	
		# 0500
		TIM 010
		TIM 010
00003	LD	00004
00004	AND NOT	00003
00005	@BSET(71)	
		# 0300
		TIM 010
		TIM 010
00006	LD	00003
00007	OR	00004
00008	TIM	010
		# 9999

5-16-4 BLOCK TRANSFER – XFER(70)

Ladder Symbols



Operand Data Areas

N: Number of words (BCD)
IR, SR, AR, DM, HR, TC, LR, #

S: Starting source word
IR, SR, AR, DM, HR, TC, LR

D: Starting destination word
IR, SR, AR, DM, HR, TC, LR

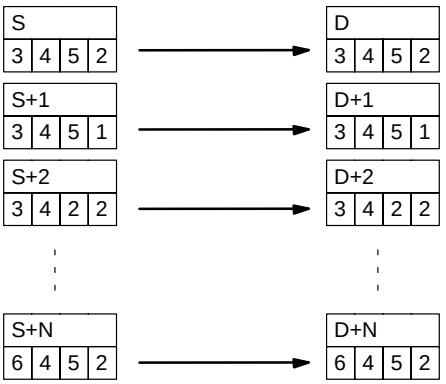
Limitations

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 between 0000 and 6144.

Description

When the execution condition is OFF, XFER(70) is not executed. When the execution condition is ON, XFER(70) copies the contents of S, S+1, ..., S+N to D, D+1, ..., D+N.



Flags

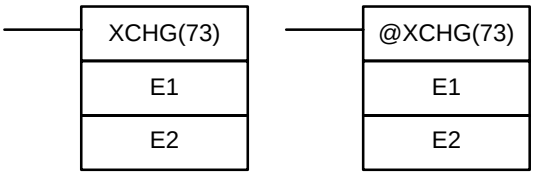
ER: N is not BCD between 0000 and 2000.

S and S+N or D and D+N are not in the same data area.

Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

5-16-5 DATA EXCHANGE – XCHG(73)

Ladder Symbols



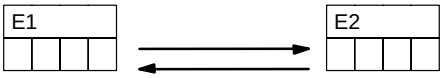
Operand Data Areas

E1: Exchange word 1
IR, SR, AR, DM, HR, TC, LR

E2: Exchange word 2
IR, SR, AR, DM, HR, TC, LR

Description

When the execution condition is OFF, XCHG(73) is not executed. When the execution condition is ON, XCHG(73) exchanges the content of E1 and E2.



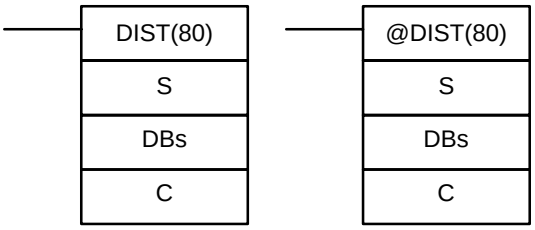
If you want to exchange content of blocks whose size is greater than 1 word, use work words as an intermediate buffer to hold one of the blocks using XFER(70) three times.

Flags

ER: Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

5-16-6 SINGLE WORD DISTRIBUTE – DIST(80)

Ladder Symbols



Operand Data Areas

S: Source data
IR, SR, AR, DM, HR, TC, LR, #

DBs: Destination base word
IR, SR, AR, DM, HR, TC, LR

C: Control word (BCD)
IR, SR, AR, DM, HR, TC, LR, #

Limitations

C must be a BCD. If C≤6655, DBs must be in the same data area as DBs+C. If C≥9000, DBs must be in the same data area as DBs+C–9000.

Description

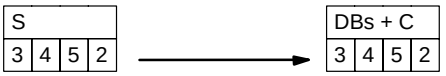
Depending on the value of C, DIST(80) will operate as a data distribution instruction or stack instruction. If C is between 0000 and 6655, DIST(80) will operate as a data distribution instruction and copy the content of S to DBs+C. If the leftmost digit of C is 9, DIST(80) will operate as a stack instruction and create a stack with the number of words specified in the rightmost 3 digits of C.

Precautions

Stack operation will be unreliable if the specified stack length is different from the length specified in the last execution of DIST(80) or COLL(81).

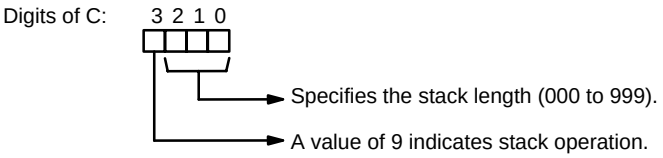
Data Distribution Operation (C=0000 to 6655)

When the execution condition is OFF, DIST(80) is not executed. When the execution condition is ON, DIST(80) copies the content of S to DBs+C, i.e.,C is added to DBs to determine the destination word.



Stack Operation (C=9000 to 9999)

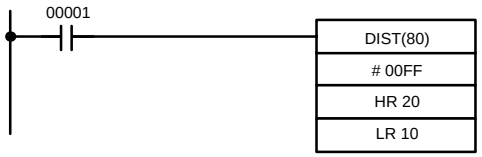
When the execution condition is OFF, DIST(80) is not executed. When the execution condition is ON, DIST(80) operates a stack from DBs to DBs+C–9000. DBs is the stack pointer, so S is copied to the word indicated by DBs and DBs is incremented by 1.



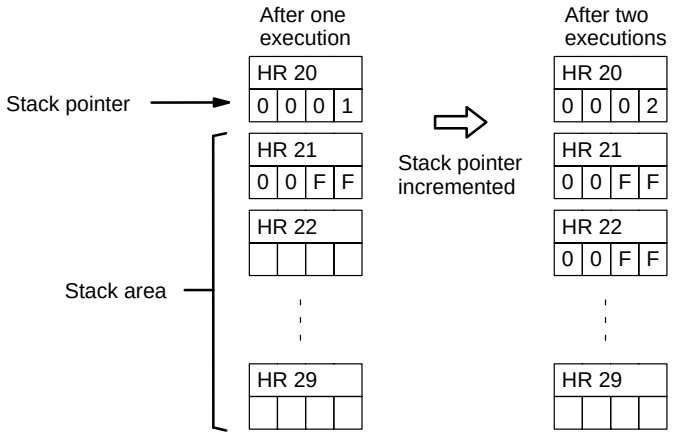
Data can be added to the stack until it is full. DIST(80) is normally used together with COLL(81), which can be set to read from the stack on a FIFO or LIFO basis. Refer to *5-16-7 DATA COLLECT – COLL(81)* for details.

Example of Stack Operation

In the following example, the content of C (LR 10) is 9010, and DIST(80) is used to write the numerical data #00FF to the 10-word stack from HR 20 to HR 29. During the first cycle when IR 00001 is ON, the data is written to DBs+1 (HR 21) and the stack pointer is incremented by 1. In the second cycle the data is written to DBs+2 (HR 22) and the stack pointer is incremented, and so on.



Address	Instruction	Operands
00000	LD	00001
00001	DIST(80)	
		# 00FF
		HR 20
		LR 10



Flags

ER: The content of C is not BCD or 6655<C<9000.

When C≤6655, DBs and DBs+C are not in the same data area.

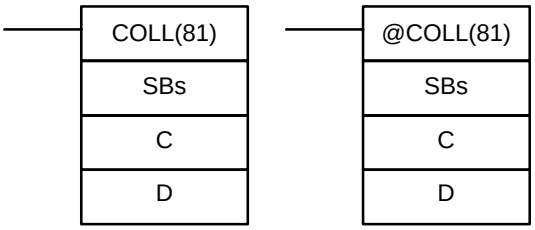
When C≥9000, DBs and DBs+C–9000 are not in the same data area.

Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON when the content of S is zero; otherwise OFF.

5-16-7 DATA COLLECT – COLL(81)

Ladder Symbols



Operand Data Areas

SBs: Source base word	
IR, SR, AR, DM, HR, TC, LR	
C: Offset data (BCD)	
IR, SR, AR, DM, HR, TC, LR, #	
D: Destination word	
IR, SR, AR, DM, HR, TC, LR	

Limitations

C must be a BCD. If C≤6655, SBs must be in the same data area as SBs+C. If the leftmost digit of C is 8 or 9, DBs must be in the same data area as SBs+N (N=the 3 rightmost digits of C).

Description

Depending on the value of C, COLL(81) will operate as a data collection instruction, FIFO stack instruction, or LIFO stack instruction. If C is between 0000 and 6655, COLL(81) will operate as a data collection instruction and copy the content of SBs+C to D.

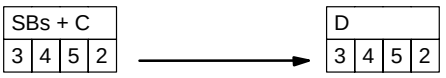
If the leftmost digit of C is 9 , COLL(81) will operate as a FIFO stack instruction. If the leftmost digit of C is 8, COLL(81) will operate as a LIFO stack instruction. Both stack operations use a stack beginning at SBs with a length specified in the rightmost 3 digits of C.

Precautions

Stack operation will be unreliable if the specified stack length is different from the length specified in the last execution of DIST(80) or COLL(81).

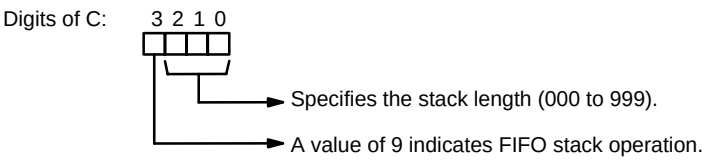
Data Collection Operation (C=0000 to 6655)

When the execution condition is OFF, COLL(81) is not executed. When the execution condition is ON, COLL(81) copies the content of SBs + C to D, i.e., C is added to SBs to determine the source word.



FIFO Stack Operation (C=9000 to 9999)

When the execution condition is OFF, COLL(81) is not executed. When the execution condition is ON, COLL(81) copies the data from the oldest word recorded in the stack to D. The stack pointer, SBs, is then decremented by 1.

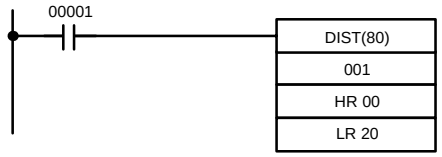


COLL(81) can be used together with DIST(80). Refer to *5-16-6 SINGLE WORD DISTRIBUTE – DIST(80*) for details.

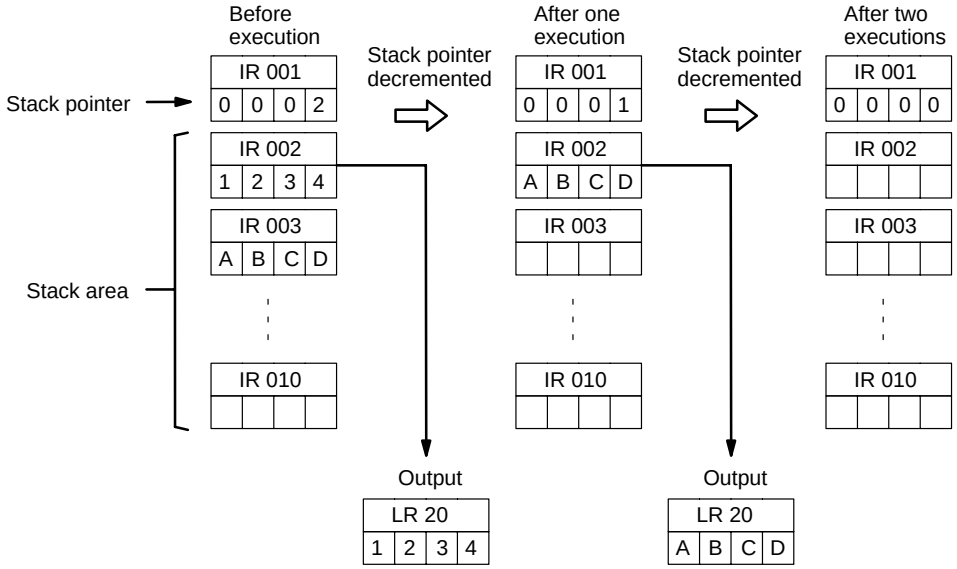
Note FIFO stands for First-In-First-Out.

Example

In the following example, the content of C (HR 00) is 9010, and COLL(81) is used to copy the oldest entries from a10-word stack (IR 001 to IR 010) to LR 20.

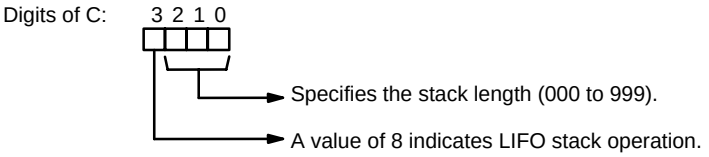


Address	Instruction	Operands
00000	LD	00001
00001	COLL(81)	
		001
		HR 00
		LR 20



LIFO Stack Operation (C=8000 to 8999)

When the execution condition is OFF, COLL(81) is not executed. When the execution condition is ON, COLL(81) copies the data most recently recorded in the stack to D. The stack pointer, SBs, is then decremented by 1.

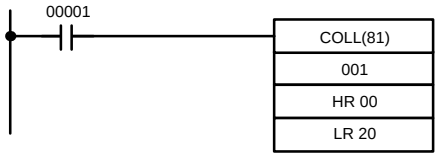


Data can be added to the stack until it is full. DIST(80)'s stack operation can be used together with COLL(81)'s read stack operation. COLL(81) can be set to read on a FIFO or LIFO basis. Refer to *5-16-6 SINGLE WORD DISTRIBUTE (80*) for details.

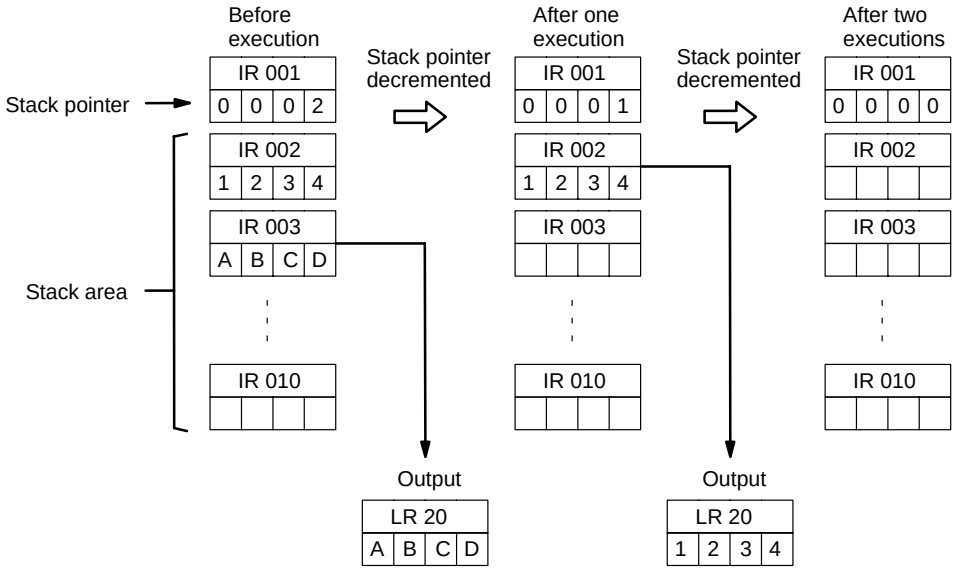
Note LIFO stands for Last-In-First-Out.

Example

In the following example, the content of C (HR 00) is 8010, and COLL(81) is used to copy the most recent entries from a 10-word stack (IR 001 to IR 010) to LR 20.



Address	Instruction	Operands
00000	LD	00001
00001	COLL(81)	
		001
		HR 00
		LR 20



Flags

ER: The content of C is not BCD or 6655<C<8000.

When C≤6655, DBs and DBs+C are not in the same data area.

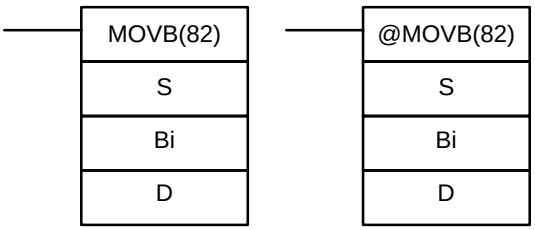
When C≥8000, the beginning and end of the stack are not in the same data area or the value of the stack pointer exceeds the length of the stack.

Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON when the transferred data is zero; otherwise OFF.

5-16-8 MOVE BIT – MOVB(82)

Ladder Symbols



Operand Data Areas

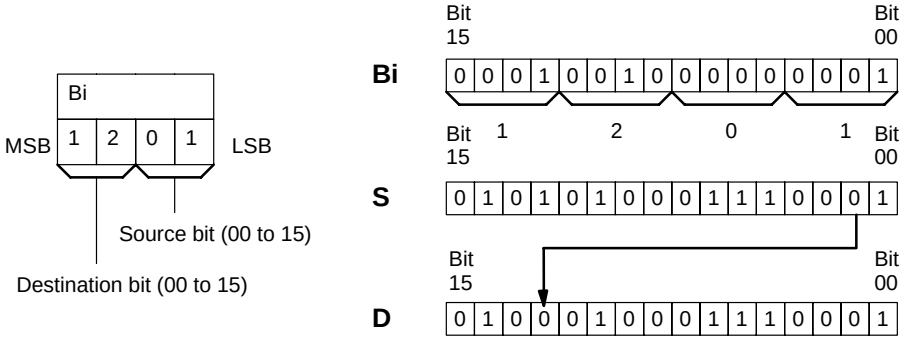
S: Source word
IR, SR, AR, DM, HR, LR, #
Bi: Bit designator (BCD)
IR, SR, AR, DM, HR, TC, LR, #
D: Destination word
IR, SR, AR, DM, HR, LR

Limitations

The rightmost two digits and the leftmost two digits of Bi must each be between 00 and 15.

Description

When the execution condition is OFF, MOVB(82) is not executed. When the execution condition is ON, MOVB(82) copies the specified bit of S to the specified bit in D. The bits in S and D are specified by Bi. The rightmost two digits of Bi designate the source bit; the leftmost two bits designate the destination bit.



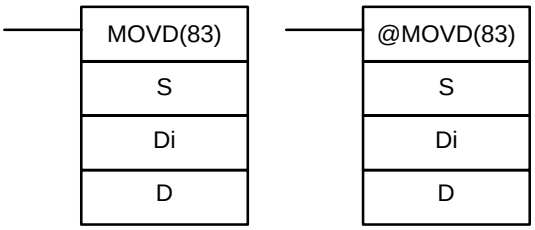
Flags

ER: C is not BCD, or it is specifying a non-existent bit (i.e., bit specification must be between 00 and 15).

Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

5-16-9 MOVE DIGIT – MOVD(83)

Ladder Symbols



Operand Data Areas

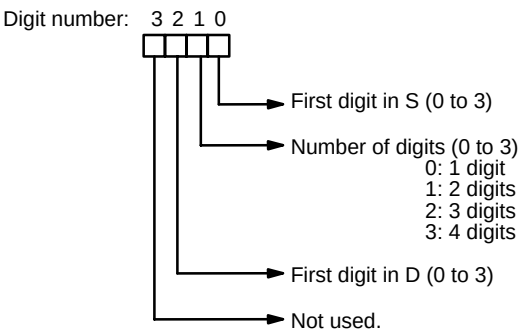
S: Source word
IR, SR, AR, DM, HR, TC, LR, #
Di: Digit designator (BCD)
IR, SR, AR, DM, HR, TC, LR, #
D: Destination word
IR, SR, AR, DM, HR, TC, LR

Limitations

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

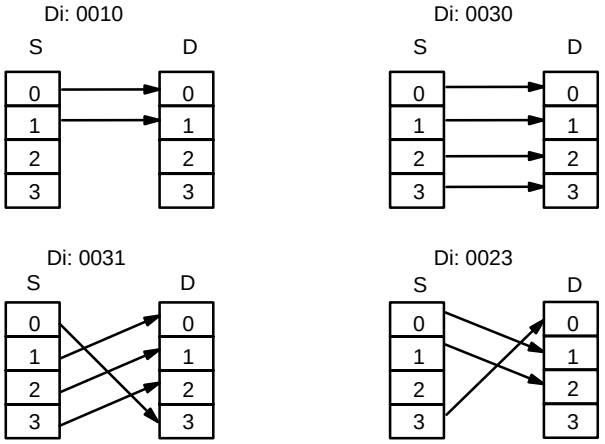
Description

When the execution condition is OFF, MOVD(83) is not executed. When the execution condition is ON, MOVD(83) 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.



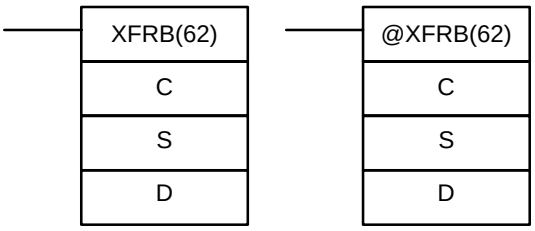
Flags

ER: At least one of the rightmost three digits of Di is not between 0 and 3.

Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

5-16-10 TRANSFER BITS – XFRB(62)

Ladder Symbols



Operand Data Areas

C: Control word	IR, SR, AR, DM, TC, HR, LR, #
S: First source word	IR, SR, AR, DM, HR, TC, LR
D: First destination word	IR, SR, AR, DM, HR, LR

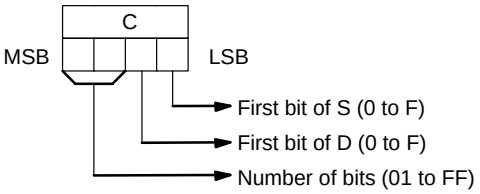
Limitations

The specified source bits must be in the same data area.

The specified destination bits must be in the same data area.

Description

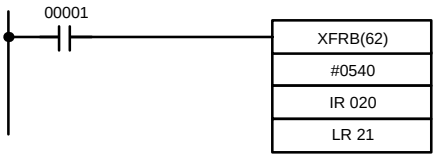
When the execution condition is OFF, XFRB(62) is not executed. When the execution condition is ON, XFRB(62) copies the specified source bits to the specified destination bits. The two rightmost digits of C specify the starting bits in S and D and the leftmost two digits indicate the number of bits that will be copied.



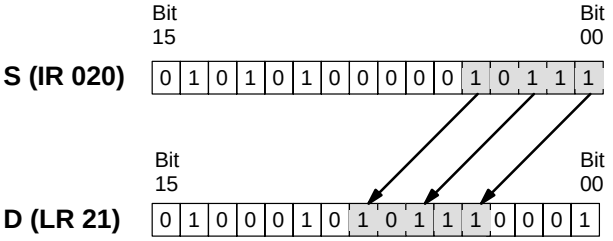
Note 255 (FF) bits or more can not be copied at one time.

Example

In the following example, XFRB(62) is used to transfer 5 bits from IR 020 to LR 21 when IR 00001 is ON. The starting bit in IR 020 is 0, and the starting bit in LR 21 is 4, so IR 02000 to IR 02004 are copied to LR 2104 to LR 2108.



Address	Instruction	Operands
00000	LD	00001
00001	XFRB(62)	
		# 0540
		020
		LR 21



Flags

ER: The specified source bits are not all in the same data area.

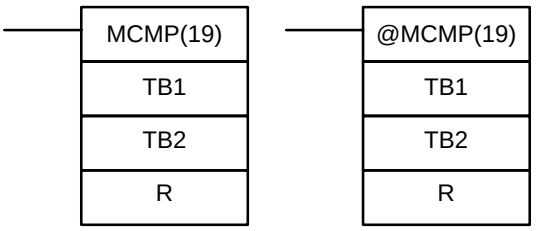
The specified destination bits are not all in the same data area.

Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

5-17 Data Comparison

5-17-1 MULTI-WORD COMPARE – MCMP(19)

Ladder Symbols



Operand Data Areas

TB1: First word of table 1
IR, SR, AR, DM, HR, TC, LR
TB2: First word of table 2
IR, SR, AR, DM, HR, TC, LR
R: Result word
IR, AR, DM, HR, TC, LR

Limitations

TB1 and TB1+15 must be in the same data area, as must TB2 and TB2+15.

Description

When the execution condition is OFF, MCMP(19) is not executed. When the execution condition is ON, MCMP(19) compares the content of TB1 to TB2, TB1+1 to TB2+1, TB1+2 to TB2+2, ..., and TB1+15 to TB2+15. If the first pair is equal, the first bit in R is turned OFF, etc., i.e., if the content of TB1 equals the content of TB2, bit 00 is turned OFF, if the content of TB1+1 equals the content of TB2+1, bit 01 is turned OFF, etc. The rest of the bits in R will be turned ON.

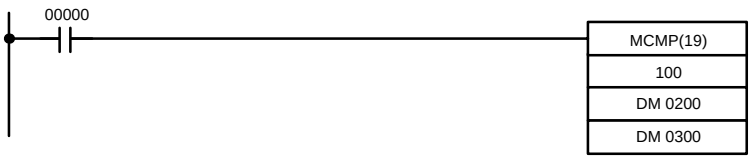
Flags

ER: One of the tables (i.e., TB1 through TB1+15, or TB2 through TB2+15) exceeds the data area.

Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

Example

The following example shows the comparisons made and the results provided for MCMP(19). Here, the comparison is made during each cycle when 00000 is ON.

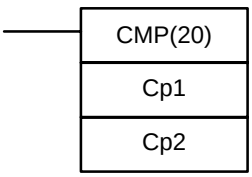


Address	Instruction	Operands	
00000	LD		00000
00001	MCMP(19)		
			100
		DM	0200
		DM	0300

TB1: IR 100		TB2: DM 0200		R: DM 0300	
IR 100	0100	DM 0200	0100	DM 030000	0
IR 101	0200	DM 0201	0200	DM 030001	0
IR 102	0210	DM 0202	0210	DM 030002	0
IR 103	ABCD	DM 0203	0400	DM 030003	1
IR 104	ABCD	DM 0204	0500	DM 030004	1
IR 105	ABCD	DM 0205	0600	DM 030005	1
IR 106	ABCD	DM 0206	0210	DM 030006	1
IR 107	0800	DM 0207	0800	DM 030007	0
IR 108	0900	DM 0208	0900	DM 030008	0
IR 109	1000	DM 0209	1000	DM 030009	0
IR 110	ABCD	DM 0210	0210	DM 030010	1
IR 111	ABCD	DM 0211	1200	DM 030011	1
IR 112	ABCD	DM 0212	1300	DM 030012	1
IR 113	1400	DM 0213	1400	DM 030013	0
IR 114	0210	DM 0214	0210	DM 030014	0
IR 115	1212	DM 0215	1600	DM 030015	1

5-17-2 COMPARE – CMP(20)

Ladder Symbols



Operand Data Areas

Cp1: First compare word
IR, SR, AR, DM, HR, TC, LR, #

Cp2: Second compare word
IR, SR, AR, DM, HR, TC, LR, #

Limitations

When comparing a value to the PV of a timer or counter, the value must be in BCD.

Description

When the execution condition is OFF, CMP(20) is not executed. When the execution condition is ON, CMP(20) compares Cp1 and Cp2 and outputs the result to the GR, EQ, and LE flags in the SR area.

Precautions

Placing other instructions between CMP(20) 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.

CMP(20) cannot be used to compare signed binary data. Use CPS(—) instead. Refer to *5-17-8 SIGNED BINARY COMPARE – CPS(—)* for details.

Flags

ER: Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON if Cp1 equals Cp2.

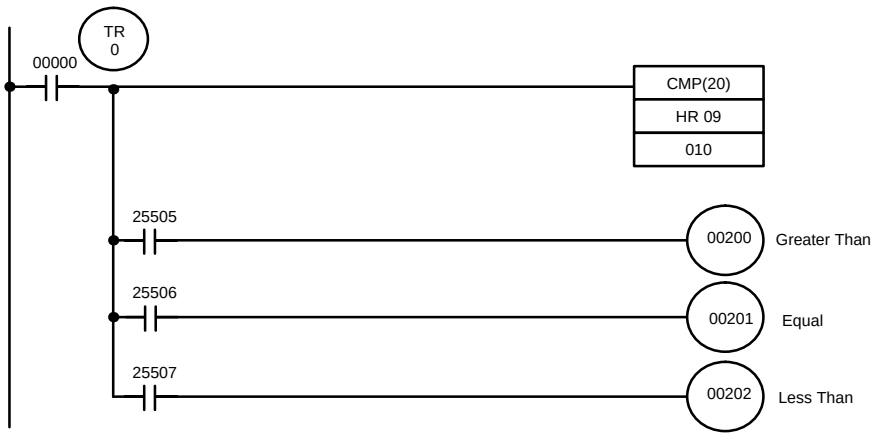
LE: ON if Cp1 is less than Cp2.

GR: ON if Cp1 is greater than Cp2.

Flag	Address	C1 < C2	C1 = C2	C1 > C2
GR	25505	OFF	OFF	ON
EQ	25506	OFF	ON	OFF
LE	25507	ON	OFF	OFF

Example 1: Saving CMP(20) Results

The following example shows how to save the comparison result immediately. If the content of HR 09 is greater than that of 010, 00200 is turned ON; if the two contents are equal, 00201 is turned ON; if content of HR 09 is less than that of 010, 00202 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, 00200, 00201, and 00202 are changed only when CMP(20) is executed.



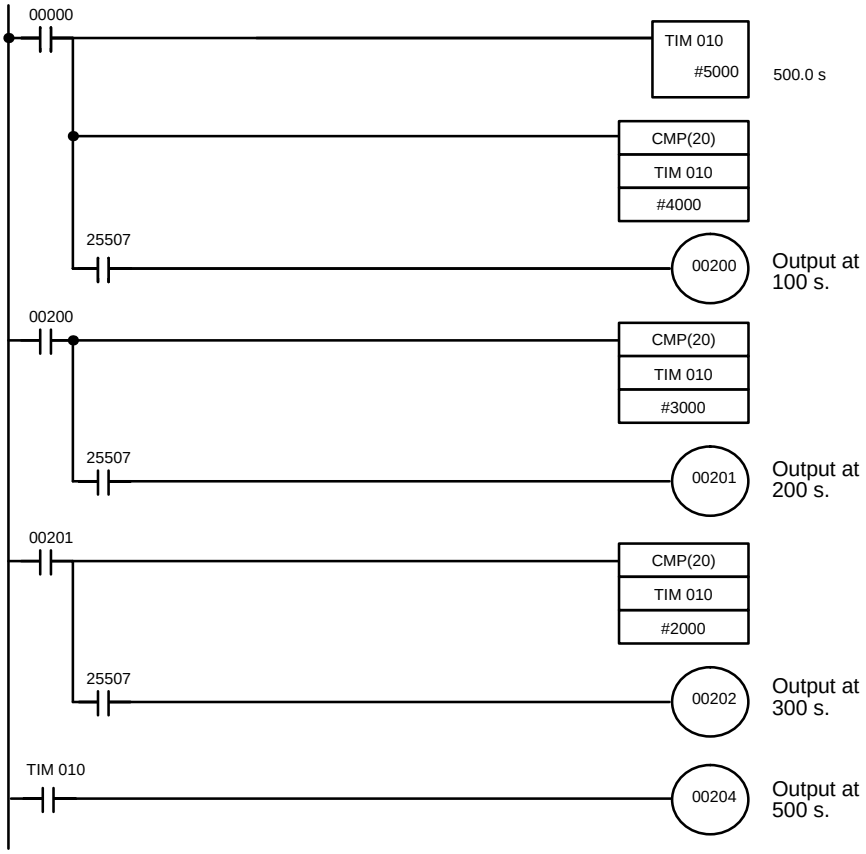
Address	Instruction	Operands	
00000	LD		00000
00001	OUT	TR	0
00002	CMP(20)		
			010
		HR	09
00003	LD	TR	0
00004	AND		25505

Address	Instruction	Operands	
00005	OUT		00200
00006	LD	TR	0
00007	AND		25506
00008	OUT		00201
00009	LD	TR	0
00010	AND		25507
00011	OUT		00202

Example 2: Obtaining Indications during Timer Operation

The following example uses TIM, CMP(20), and the LE flag (25507) to produce outputs at particular times in the timer's countdown. The timer is started by turning ON 00000. When 00000 is OFF, TIM 010 is reset and the second two CMP(20)s are not executed (i.e., executed with OFF execution conditions). Output 00200 is produced after 100 seconds; output 00201, after 200 seconds; output 00202, after 300 seconds; and output 00204, after 500 seconds.

The branching structure of this diagram is important in order to ensure that 00200, 00201, and 00202 are controlled properly as the timer counts down. Because all of the comparisons here use to the timer's PV as reference, the other operand for each CMP(20) must be in 4-digit BCD.

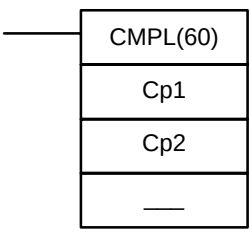


Address	Instruction	Operands
00000	LD	00000
00001	TIM	010
		# 5000
00002	CMP(20)	
		TIM 010
		# 4000
00003	AND	25507
00004	OUT	00200
00005	LD	00200
00006	CMP(20)	
		TIM 010
		# 3000

Address	Instruction	Operands
00007	AND	25507
00008	OUT	00201
00009	LD	00201
00010	CMP(20)	
		TIM 010
		# 2000
00011	AND	25507
00012	OUT	00202
00013	LD	TIM 010
00014	OUT	00204

5-17-3 DOUBLE COMPARE – CMPL(60)

Ladder Symbols



Operand Data Areas

Cp1: First word of first compare word pair
IR, SR, AR, DM, HR, TC, TR

Cp2: First word of second compare word pair
IR, SR, AR, DM, HR, TC, LR

Limitations

Cp1 and Cp1+1 must be in the same data area, as must Cp2 and Cp2+1.

Description

When the execution condition is OFF, CMPL(60) is not executed. When the execution condition is ON, CMPL(60) joins the 4-digit hexadecimal content of Cp1+1 with that of Cp1, and that of Cp2+1 with that of Cp2 to create two 8-digit hexadecimal numbers, Cp+1,Cp1 and Cp2+1,Cp2. The two 8-digit numbers are then compared and the result is output to the GR, EQ, and LE flags in the SR area.

Precautions

Placing other instructions between CMPL(60) 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.

CMPL(60) cannot be used to compare signed binary data. Use CPSL(—) instead. Refer to *5-17-9 DOUBLE SIGNED BINARY COMPARE – CPSL(—)* for details.

Flags

ER: Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

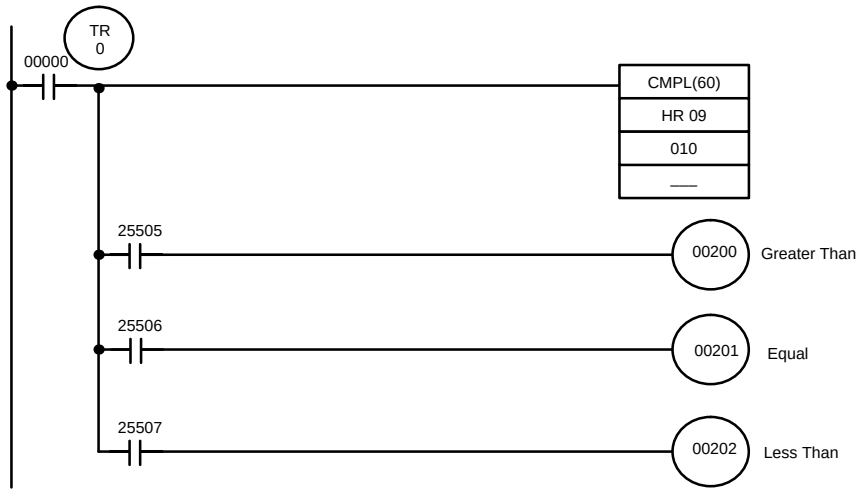
GR: ON if Cp1+1,Cp1 is greater than Cp2+1,Cp2.

EQ: ON if Cp1+1,Cp1 equals Cp2+1,Cp2.

LE: ON if Cp1+1,Cp1 is less than Cp2+1,Cp2.

Example: Saving CMPL(60) Results

The following example shows how to save the comparison result immediately. If the content of HR 10, HR 09 is greater than that of 011, 010, then 00200 is turned ON; if the two contents are equal, 00201 is turned ON; if content of HR 10, HR 09 is less than that of 011, 010, then 00202 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, 00200, 00201, and 00202 are changed only when CMPL(60) is executed.

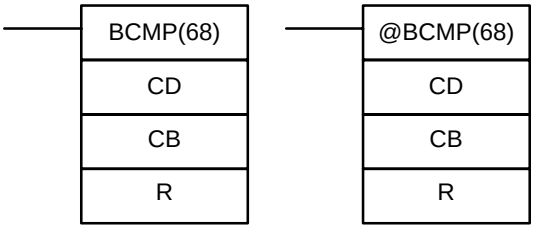


Address	Instruction	Operands
00000	LD	00000
00001	OUT	TR 0
00002	CMPL(60)	
		HR 09
		010
00003	AND	25505

Address	Instruction	Operands
00004	OUT	00200
00005	LD	TR 0
00006	AND	25506
00007	OUT	00201
00008	LD	TR 0
00009	AND	25507
00010	OUT	00202

5-17-4 BLOCK COMPARE – BCMP(68)

Ladder Symbols



Operand Data Areas

CD: Compare data	
IR, SR, AR, DM, HR, TC, LR, #	
CB: First comparison block word	
IR, DM, HR, TC, LR	
R: Result word	
IR, SR, AR, DM, HR, TC, LR	

Limitations

Each lower limit word in the comparison block must be less than or equal to the upper limit.

Description

When the execution condition is OFF, BCMP(68) is not executed. When the execution condition is ON, BCMP(68) compares CD to the ranges defined by a block consisting of of CB, CB+1, CB+2, ..., CB+31. Each range is defined by two words, the first one providing the lower limit and the second word providing the upper limit. If CD 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 ≤ CD ≤ CB+1	Bit 00
CB+2 ≤ CD ≤ CB+3	Bit 01
CB+4 ≤ CD ≤ CB+5	Bit 02
CB+6 ≤ CD ≤ CB+7	Bit 03
CB+8 ≤ CD ≤ CB+9	Bit 04
CB+10 ≤ CD ≤ CB+11	Bit 05
CB+12 ≤ CD ≤ CB+13	Bit 06
CB+14 ≤ CD ≤ CB+15	Bit 07
CB+16 ≤ CD ≤ CB+17	Bit 08
CB+18 ≤ CD ≤ CB+19	Bit 09
CB+20 ≤ CD ≤ CB+21	Bit 10
CB+22 ≤ CD ≤ CB+23	Bit 11
CB+24 ≤ CD ≤ CB+25	Bit 12
CB+26 ≤ CD ≤ CB+27	Bit 13
CB+28 ≤ CD ≤ CB+29	Bit 14
CB+30 ≤ CD ≤ CB+31	Bit 15

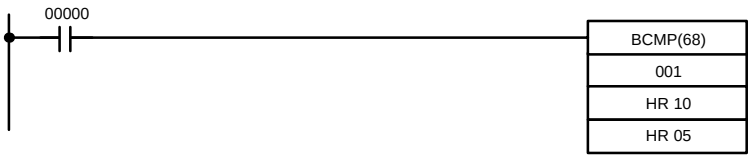
Flags

ER: The comparison block (i.e., CB through CB+31) exceeds the data area.

Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

Example

The following example shows the comparisons made and the results provided for BCMP(68). Here, the comparison is made during each cycle when 00000 is ON.



Address	Instruction	Operands
00000	LD	00000
00001	BCMP(68)	
		001
		HR 10
		HR 05

CD 001

001	0210

Compare data in IR 001

(which contains 0210) with the given ranges.

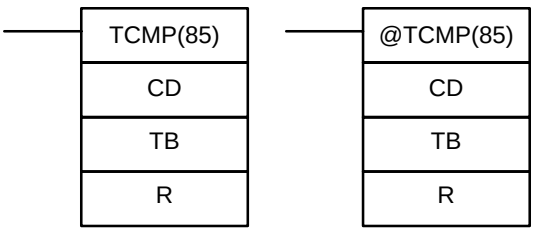
Lower limits	
HR 10	0000
HR 12	0101
HR 14	0201
HR 16	0301
HR 18	0401
HR 20	0501
HR 22	0601
HR 24	0701
HR 26	0801
HR 28	0901
HR 30	1001
HR 32	1101
HR 34	1201
HR 36	1301
HR 38	1401
HR 40	1501

Upper limits	
HR 11	0100
HR 13	0200
HR 15	0300
HR 17	0400
HR 19	0500
HR 21	0600
HR 23	0700
HR 25	0800
HR 27	0900
HR 29	1000
HR 31	1100
HR 33	1200
HR 35	1300
HR 37	1400
HR 39	1500
HR 41	1600

R: HR 05	
HR 0500	0
HR 0501	0
HR 0502	1
HR 0503	0
HR 0504	0
HR 0505	0
HR 0506	0
HR 0507	0
HR 0508	0
HR 0509	0
HR 0510	0
HR 0511	0
HR 0512	0
HR 0513	0
HR 0514	0
HR 0515	0

5-17-5 TABLE COMPARE – TCMP(85)

Ladder Symbols



Operand Data Areas

CD: Compare data
IR, SR, AR, DM, HR, TC, LR, #
TB: First comparison table word
IR, AR, DM, HR, TC, LR
R: Result word
IR, SR, AR, DM, HR, TC, LR

Limitations

TB and TB+15 must be in the same data area.

Description

When the execution condition is OFF, TCMP(85) is not executed. When the execution condition is ON, TCMP(85) compares CD to the content of TB, TB+1, TB+2, ..., and TB+15. If CD is equal to the content of any of these words, the corresponding bit in R is set, e.g., if the CD equals the content of TB, bit 00 is turned ON, if it equals that of TB+1, bit 01 is turned ON, etc. The rest of the bits in R will be turned OFF.

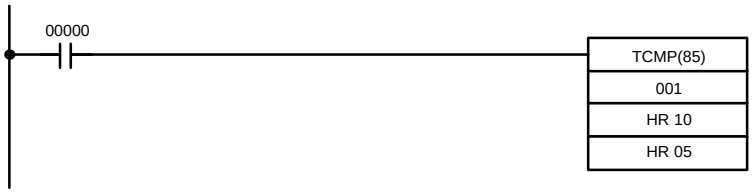
Flags

ER: The comparison table (i.e., TB through TB+15) exceeds the data area.

Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

Example

The following example shows the comparisons made and the results provided for TCMP(85). Here, the comparison is made during each cycle when 00000 is ON.



Address	Instruction	Operands
00000	LD	00000
00001	TCMP(85)	
		001
		HR 10
		HR 05

CD: 001

001	0210

Compare the data in IR 001 with the given ranges.

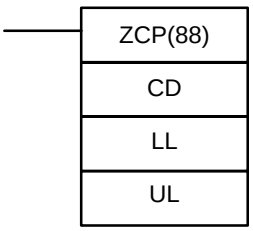
Upper limits	
HR 10	0100
HR 11	0200
HR 12	0210
HR 13	0400
HR 14	0500
HR 15	0600
HR 16	0210
HR 17	0800
HR 18	0900
HR 19	1000
HR 20	0210
HR 21	1200
HR 22	1300
HR 23	1400
HR 24	0210
HR 25	1600

R: HR 05

HR 0500	0
HR 0501	0
HR 0502	1
HR 0503	0
HR 0504	0
HR 0505	0
HR 0506	1
HR 0507	0
HR 0508	0
HR 0509	0
HR 0510	1
HR 0511	0
HR 0512	0
HR 0513	0
HR 0514	1
HR 0515	0

5-17-6 AREA RANGE COMPARE – ZCP(88)

Ladder Symbols



Operand Data Areas

CD: Compare data
IR, SR, AR, DM, HR, TC, LR, #

LL: Lower limit of range
IR, SR, AR, DM, HR, TC, LR, #

UL: Upper limit of range
IR, SR, AR, DM, HR, TC, LR, #

Limitations

LL must be less than or equal to UL.

Description

When the execution condition is OFF, ZCP(88) is not executed. When the execution condition is ON, ZCP(88) compares CD to the range defined by lower limit LL and upper limit UL and outputs the result to the GR, EQ, and LE flags in the SR area. The resulting flag status is shown in the following table.

Comparison result	Flag status		
	GR (SR 25505)	EQ (SR 25506)	LE (SR 25507)
CD < LL	0	0	1
LL ≤ CD ≤ UL	0	1	0
UL < CD	1	0	0

Precautions

Placing other instructions between ZCP(88) 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.

Flags

ER: Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

LL is greater than UL.

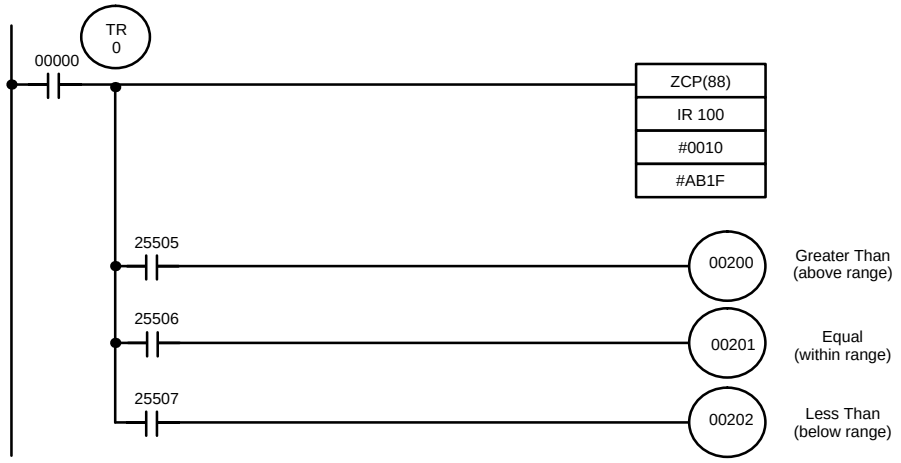
EQ: ON if LL ≤ CD ≤ UL

LE: ON if CD < LL.

GR: ON if CD > UL.

Example: Saving ZCP(88) Results

The following example shows how to save the comparison result immediately. If IR 100 > AB1F, IR 00200 is turned ON; if #0010 ≤ IR 100 ≤ AB1F, IR 00201 is turned ON; if IR 100 < 0010, IR 00202 is turned ON.

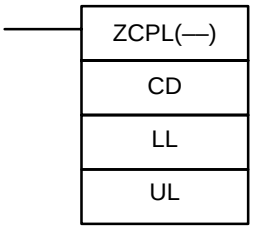


Address	Instruction	Operands
00000	LD	00000
00001	OUT	TR 0
00002	ZCP(88)	
		IR 100
		# 0010
00003	LD	# AB1F
00004	AND	25505

Address	Instruction	Operands
00005	OUT	00200
00006	LD	TR 0
00007	AND	25506
00008	OUT	00201
00009	LD	TR 0
00010	AND	25507
00011	OUT	00202

5-17-7 DOUBLE AREA RANGE COMPARE – ZCPL(––)

Ladder Symbols



Operand Data Areas

CD: Compare data
IR, SR, AR, DM, HR, LR
LL: Lower limit of range
IR, SR, AR, DM, HR, LR
UL: Upper limit of range
IR, SR, AR, DM, HR, LR

Limitations

The 8-digit value in LL+1,LL must be less than or equal to UL+1,UL.

CD and CD+1 must be in the same data area, as must LL and LL+1, and UL and UL+1.

Description

When the execution condition is OFF, ZCPL(—) is not executed. When the execution condition is ON, ZCPL(—) compares the 8-digit value in CD, CD+1 to the range defined by lower limit LL+1,LL and upper limit UL+1,UL and outputs the result to the GR, EQ, and LE flags in the SR area. The resulting flag status is shown in the following table.

Comparison result	Flag status		
	GR (SR 25505)	EQ (SR 25506)	LE (SR 25507)
CD , CD+1< LL+1,LL	0	0	1
LL+1,LL ≤ CD, CD+1 ≤ UL+1,UL	0	1	0
UL+1,UL < CD, CD+1	1	0	0

Precautions

Placing other instructions between ZCPL(—) 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.

Flags

ER: Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

LL+1,LL is greater than UL+1,UL.

EQ: ON if LL+1,LL ≤ CD, CD+1 ≤ UL+1,UL

LE: ON if CD, CD+1 < LL+1,LL.

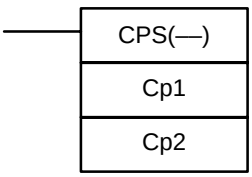
GR: ON if CD, CD+1 > UL+1,UL.

Example

Refer to *5-17-6 AREA RANGE COMPARE – ZCP(88)* for an example. The only difference between ZCP(88) and ZCPL(—) is the number of digits in the comparison data.

5-17-8 SIGNED BINARY COMPARE – CPS(––)

Ladder Symbols



Operand Data Areas

Cp1: First compare word
IR, SR, AR, DM, HR, TC, LR, #

Cp2: Second compare word
IR, SR, AR, DM, HR, TC, LR, #

Description

When the execution condition is OFF, CPS(—) is not executed. When the execution condition is ON, CPS(—) compares the 16-bit (4-digit) signed binary contents in Cp1 and Cp2 and outputs the result to the GR, EQ, and LE flags in the SR area.

Note

1. Refer to page 29 for details on 16-bit signed binary data.
2. Refer to *5-17-2 Compare – CMP(20)* for details on saving comparison results.

Precautions

Placing other instructions between CPS(—) 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.

Flags

ER: Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON if Cp1 equals Cp2.

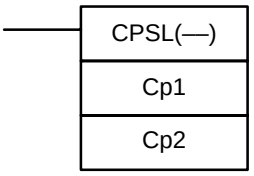
LE: ON if Cp1 is less than Cp2.

GR: ON if Cp1 is greater than Cp2.

Comparison result	Flag status		
	GR (SR 25505)	EQ (SR 25506)	LE (SR 25505)
Cp1 < Cp2	0	0	1
Cp1 = Cp2	0	1	0
Cp1 > Cp2	1	0	0

5-17-9 DOUBLE SIGNED BINARY COMPARE – CPSL(––)

Ladder Symbols



Operand Data Areas

Cp1: First compare word
IR, SR, AR, DM, HR, TC, LR

Cp2: Second compare word
IR, SR, AR, DM, HR, TC, LR

Limitations

Cp1 and Cp1+1 must be in the same data area, as must Cp2 and Cp2+1.

Description

When the execution condition is OFF, CPSL(—) is not executed. When the execution condition is ON, CPSL(—) compares the 32-bit (8-digit) signed binary contents in Cp1+1, Cp1 and Cp2+1, Cp2 and outputs the result to the GR, EQ, and LE flags in the SR area.

Note

1. Refer to page 29 for details on 32-bit signed binary data.
2. Refer to *5-17-2 Compare – CMP(20)* for details on saving comparison results.

Precautions

Placing other instructions between CPSL(—) 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.

Flags

ER: Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON if Cp1+1, Cp1 equals Cp2+1, Cp2.

LE: ON if Cp1+1, Cp1 is less than Cp2+1, Cp2.

GR: ON if Cp1+1, Cp1 is greater than Cp2+1, Cp2.

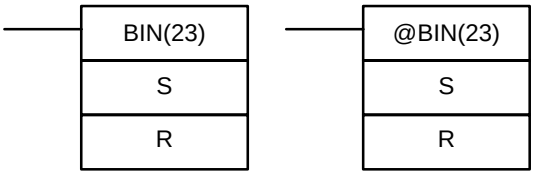
Comparison result	Flag status		
	GR (SR 25505)	EQ (SR 25506)	LE (SR 25507)
Cp1+1, Cp1 < Cp2+1, Cp2	0	0	1
Cp1+1, Cp1 = Cp2+1, Cp2	0	1	0
Cp1+1, Cp1 > Cp2+1, Cp2	1	0	0

5-18 Data Conversion

The conversion instructions convert word data that is in one format into another format and output the converted data to specified result word(s). Conversions are available to convert between binary (hexadecimal) and BCD, to 7-segment display data, to ASCII, and between multiplexed and non-multiplexed data. 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.

5-18-1 BCD-TO-BINARY – BIN(23)

Ladder Symbols

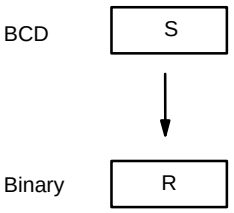


Operand Data Areas

S: Source word (BCD)	IR, SR, AR, DM, HR, TC, LR
R: Result word	IR, SR, AR, DM, HR, LR

Description

When the execution condition is OFF, BIN(23) is not executed. When the execution condition is ON, BIN(23) 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(23) can be used to convert BCD to binary so that displays on the Programming Console or any other programming device 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.

Flags

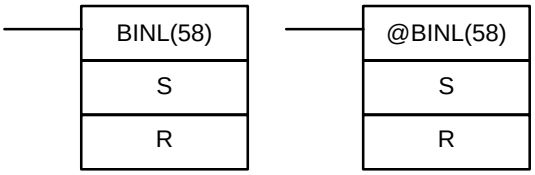
ER: The content of S is not BCD.

Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON when the result is zero.

5-18-2 DOUBLE BCD-TO-DOUBLE BINARY – BINL(58)

Ladder Symbols

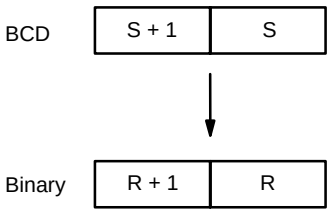


Operand Data Areas

S: First source word (BCD)
IR, SR, AR, DM, HR, TC, LR

R: First result word
IR, SR, AR, DM, HR, LR

Description When the execution condition is OFF, BINL(58) is not executed. When the execution condition is ON, BINL(58) converts an eight-digit number in S and S+1 into 32-bit binary data, and outputs the converted data to R and R+1.



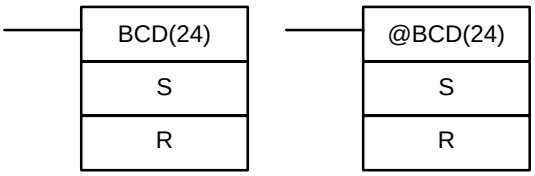
Flags

ER: The contents of S and/or S+1 words are not BCD.
Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON when the result is zero.

5-18-3 BINARY-TO-BCD – BCD(24)

Ladder Symbols

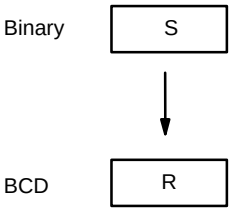


Operand Data Areas

S: Source word (binary)
IR, SR, AR, DM, HR, TC, LR

R: Result word
IR, SR, AR, DM, HR, LR

Description BCD(24) converts the binary (hexadecimal) content of S into the numerically equivalent BCD bits, and outputs the BCD bits to R. Only the content of R is changed; the content of S is left unchanged.



BCD(24) can be used to convert binary to BCD so that displays on the Programming Console or any other programming device will appear in decimal rather than hexadecimal. It can also be used to convert to BCD to perform BCD arithmetic operations rather than binary arithmetic operations, e.g., when BCD and binary values must be added.

Note If the content of S exceeds 270F, the converted result would exceed 9999 and BCD(24) will not be executed. When the instruction is not executed, the content of R remains unchanged.

Signed Binary Data

BCD(24) cannot be used to convert signed binary data directly to BCD. To convert signed binary data, first determine whether the data is positive or negative. If it is positive, BCD(24) can be used to convert the data to BCD. If it is negative, use the 2's COMPLEMENT – NEG(—) instruction to convert the data to unsigned binary before executing BCD(24). Refer to page 29 for details on signed binary data.

Flags

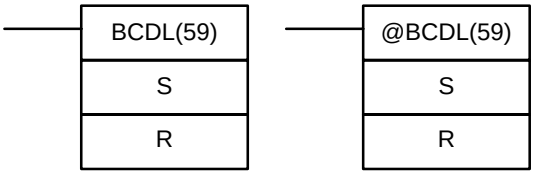
ER: S is greater than 270F.

Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON when the result is zero.

5-18-4 DOUBLE BINARY-TO-DOUBLE BCD – BCDL(59)

Ladder Symbols



Operand Data Areas

S: First source word (binary)
IR, SR, AR, DM, HR, TC, LR

R: First result word
IR, SR, AR, DM, HR, LR

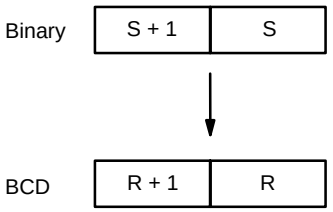
Limitations

If the content of S exceeds 05F5E0FF, the converted result would exceed 99999999 and BCDL(59) will not be executed. When the instruction is not executed, the content of R and R+1 remain unchanged.

S and S+1 must be in the same data area as must R and R+1.

Description

BCDL(59) 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.



Signed Binary Data

BCD(24) cannot be used to convert signed binary data directly to BCD. To convert signed binary data, first determine whether the data is positive or negative. If it is positive, BCD(24) can be used to convert the data to BCD. If it is negative, use the DOUBLE 2's COMPLEMENT – NEGL(—) instruction to convert the data to unsigned binary before executing BCD(24). Refer to page 29 for details on signed binary data.

Flags

ER: Content of R and R+1 exceeds 99999999.

Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON when the result is zero.

5-18-5 HOURS-TO-SECONDS – SEC(65)

Ladder Symbols

SEC(65)
S
R

@SEC(65)
S
R

Operand Data Areas

S: Beginning source word (BCD)
IR, SR, AR, DM, HR, TC, LR

R: Beginning result word (BCD)
IR, SR, AR, DM, HR, TC, LR

---: Not used.

Limitations

S and S+1 must be within the same data area. R and R+1 must be within the same data area. S and S+1 must be BCD and must be in the proper hours/minutes/seconds format.

Description

SEC(65) is used to convert time notation in hours/minutes/seconds to an equivalent in just seconds.

For the source data, the seconds is designated in bits 00 through 07 and the minutes is designated in bits 08 through 15 of S. The hours is designated in S+1. The maximum is thus 9,999 hours, 59 minutes, and 59 seconds.

The result is output to R and R+1. The maximum obtainable value is 35,999,999 seconds.

Flags

ER: S and S+1 or R and R+1 are not in the same data area.

S and/or S+1 do not contain BCD.

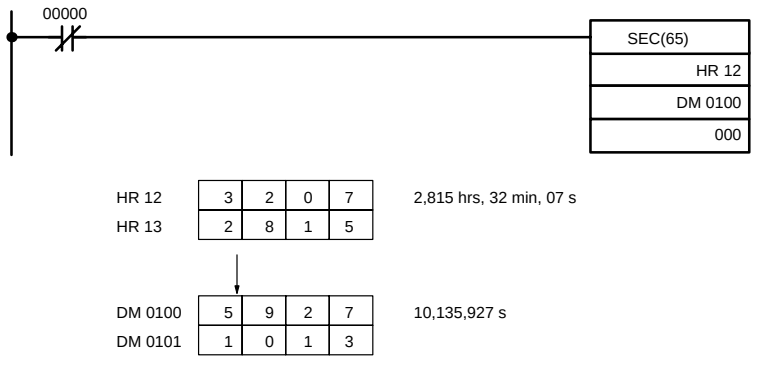
Number of seconds and/or minutes exceeds 59.

Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: Turns ON when the result is zero.

Example

When 00000 is OFF (i.e., when the execution condition is ON), the following instruction would convert the hours, minutes, and seconds given in HR 12 and HR 13 to seconds and store the results in DM 0100 and DM 0101 as shown.



Address	Instruction	Operands
00000	LD NOT	00000
00001	SEC(65)	
		HR 12
		DM 0100
		000

5-18-6 SECONDS-TO-HOURS – HMS(66)

Ladder Symbols

HMS(66)
S
R

@HMS(66)
S
R

Operand Data Areas

S: Beginning source word (BCD)
IR, SR, AR, DM, HR, TC, LR

R: Beginning result word (BCD)
IR, SR, AR, DM, HR, TC, LR

---: Not used.

Limitations

S and S+1 must be within the same data area. R and R+1 must be within the same data area. S and S+1 must be BCD and must be between 0 and 35,999,999 seconds.

Description

HMS(66) is used to convert time notation in seconds to an equivalent 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 is placed in bits 00 through 07 and the minutes is placed in bits 08 through 15 of R. The hours is placed in R+1. The maximum will be 9,999 hours, 59 minutes, and 59 seconds.

Flags

ER: S and S+1 or R and R+1 are not in the same data area.

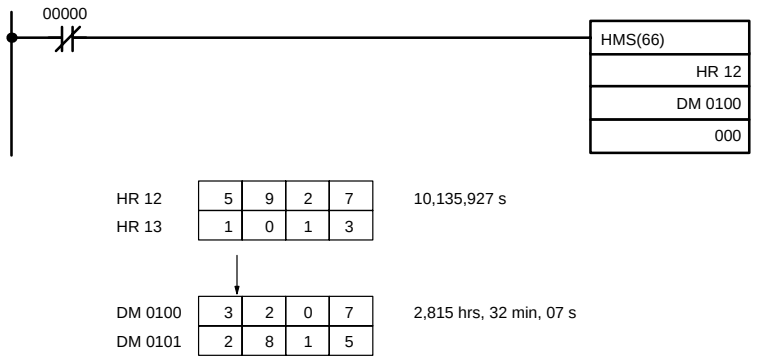
S and/or S+1 do not contain BCD or exceed 36,000,000 seconds.

Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: Turns ON when the result is zero.

Example

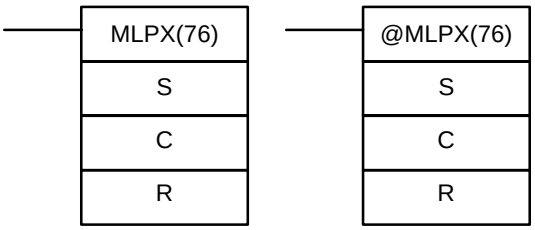
When 00000 is OFF (i.e., when the execution condition is ON), the following instruction would convert the seconds given in HR 12 and HR 13 to hours, minutes, and seconds and store the results in DM 0100 and DM 0101 as shown.



Address	Instruction	Operands	
00000	LD NOT		00000
00001	HMS(66)		
		HR	12
		DM	0100
			000

5-18-7 4-TO-16 DECODER – MLPX(76)

Ladder Symbols



Operand Data Areas

S: Source word	IR, SR, AR, DM, HR, TC, LR
C: Control word	IR, SR, AR, DM, HR, TC, LR, #
R: First result word	IR, SR, AR, DM, HR, LR

Limitations

When the leftmost digit of C is 0, the rightmost two digits of C must each be between 0 and 3.

When the leftmost digit of C is 1, the rightmost two digits of C must each be between 0 and 1.

All result words must be in the same data area.

Description

Depending on the value of C, MLPX(76) operates as a 4-bit to 16-bit decoder or an 8-bit to 256-bit decoder.

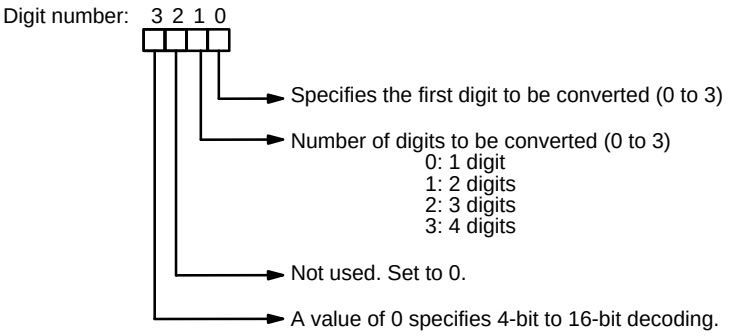
4-bit to 16-bit Decoder

MLPX(76) operates as a 4-bit to 16-bit decoder when the leftmost digit of C is 0. The hexadecimal value of the digits in S are used to specify bits in up to 4 result words. The specified bit in each result word is turned on, and the other 15 bits in each word are turned off.

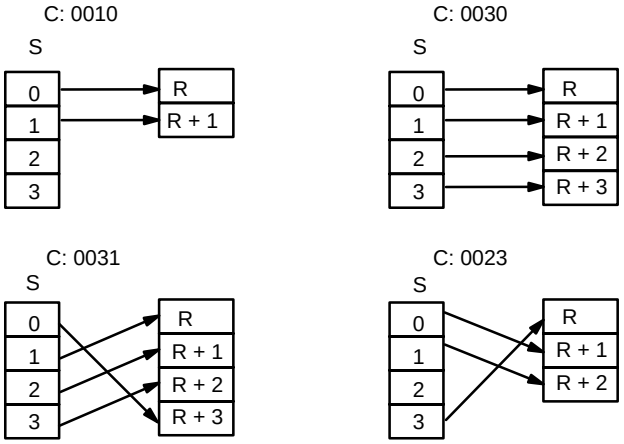
When the execution condition is OFF, MLPX(76) is not executed. When the execution condition is ON, MLPX(76) converts up to four, four-bit hexadecimal digits from S into decimal values from 0 to 15, each of which is used to indicate a bit position. The bit whose number corresponds to each converted value 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.)

Control Word

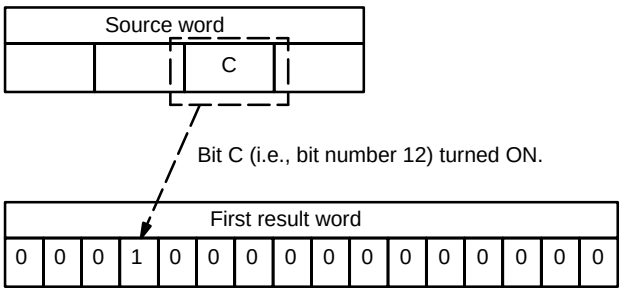
The digits of C are set as shown below. Set the leftmost digit of C to 0 to specify 4-bit to 16-bit decoding.



Some example C values and the digit-to-word conversions that they produce are shown below.



The following is an example of a one-digit decode operation from digit number 1 of S, i.e., here C would be 0001.



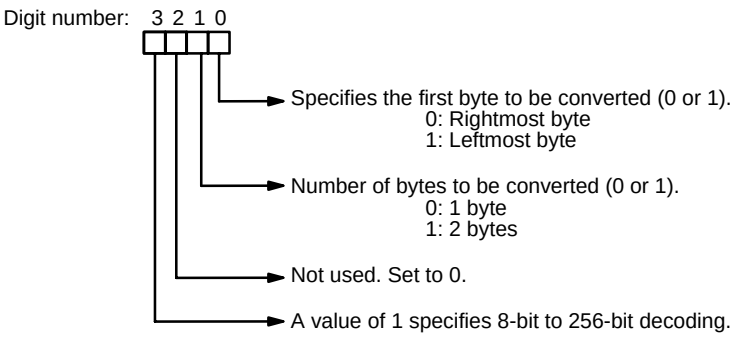
The first digit and the number of digits to be converted are designated in C. If more digits are designated than remain in S (counting from the designated first digit), the remaining digits will be taken starting back at the beginning of S. The final word required to store the converted result (R plus the number of digits to be converted) must be in the same data area as R, e.g., if two digits are converted, the last word address in a data area cannot be designated; if three digits are converted, the last two words in a data area cannot be designated.

8-bit to 256-bit Decoder

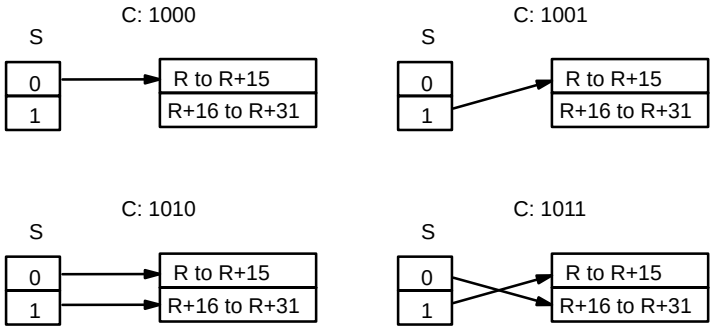
MLPX(76) operates as an 8-bit to 256-bit decoder when the leftmost digit of C is set to 1. The hexadecimal value of the two bytes in S are used to specify a bit in one or two groups of 16 consecutive result words (256 bits). The specified bit in each group is turned on, and the other 255 bits in the group are turned off.

Control Word

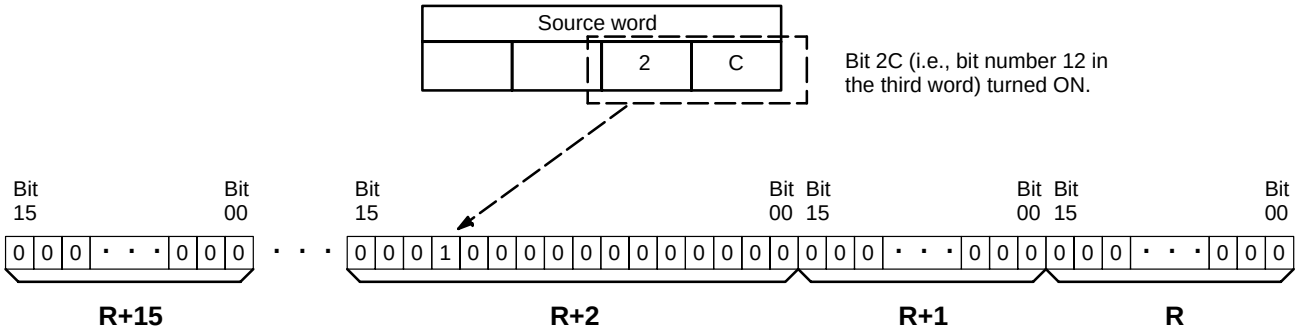
The digits of C are set as shown below. Set the leftmost digit of C to 1 to specify 8-bit to 256-bit decoding.



The 4 possible C values and the conversions that they produce are shown below. (In S, 0 indicates the rightmost byte and 1 indicates the leftmost byte.)



The following is an example of a one-byte decode operation from the rightmost byte of S (C would be 1000 in this case).



Flags

ER: Undefined control word.

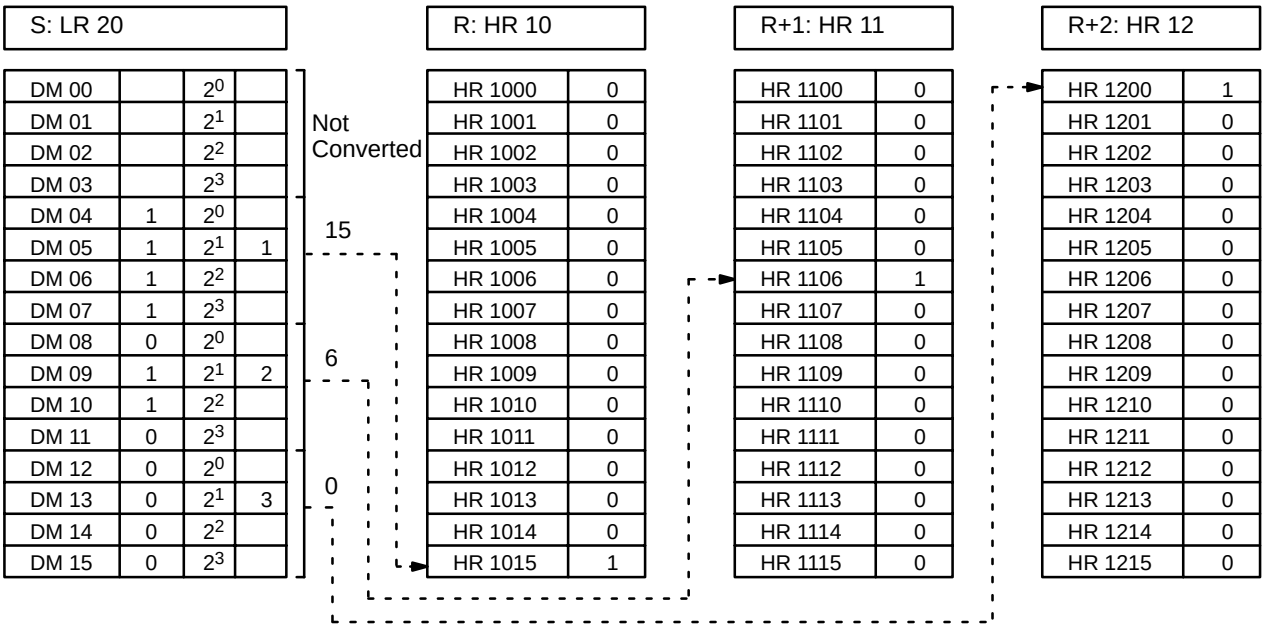
The result words are not all in the same data area.

Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

Example:
4-bit to 16-bit Decoding

The following program converts three digits of data from LR 20 to bit positions and turns ON the corresponding bits in three consecutive words starting with HR 10.

Address	Instruction	Operands	
00000	LD		00000
00001	MLPX(76)		
		LR	20
		#	0021
		HR	10



DM 00		2^0	
DM 01		2^1	
DM 02		2^2	
DM 03		2^3	
DM 04	1	2^0	
DM 05	1	2^1	1
DM 06	1	2^2	
DM 07	1	2^3	
DM 08	0	2^0	
DM 09	1	2^1	2
DM 10	1	2^2	
DM 11	0	2^3	
DM 12	0	2^0	
DM 13	0	2^1	3
DM 14	0	2^2	
DM 15	0	2^3	

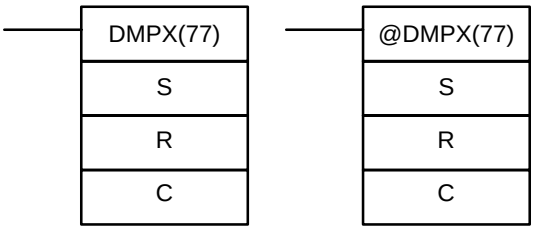
HR 1000	0
HR 1001	0
HR 1002	0
HR 1003	0
HR 1004	0
HR 1005	0
HR 1006	0
HR 1007	0
HR 1008	0
HR 1009	0
HR 1010	0
HR 1011	0
HR 1012	0
HR 1013	0
HR 1014	0
HR 1015	1

HR 1100	0
HR 1101	0
HR 1102	0
HR 1103	0
HR 1104	0
HR 1105	0
HR 1106	1
HR 1107	0
HR 1108	0
HR 1109	0
HR 1110	0
HR 1111	0
HR 1112	0
HR 1113	0
HR 1114	0
HR 1115	0

HR 1200	1
HR 1201	0
HR 1202	0
HR 1203	0
HR 1204	0
HR 1205	0
HR 1206	0
HR 1207	0
HR 1208	0
HR 1209	0
HR 1210	0
HR 1211	0
HR 1212	0
HR 1213	0
HR 1214	0
HR 1215	0

5-18-8 16-TO-4 ENCODER – DMPX(77)

Ladder Symbols



Operand Data Areas

S: First source word
IR, SR, AR, DM, HR, TC, LR

R: Result word
IR, SR, AR, DM, HR, LR

C: Control Word
IR, SR, AR, DM, HR, TC, LR, #

Limitations

When the leftmost digit of C is 0, the rightmost two digits of C must each be between 0 and 3.

When the leftmost digit of C is 1, the rightmost two digits of C must each be between 0 and 1.

All source words must be in the same data area.

Description

Depending on the value of C, MLPX(76) operates as a 16-bit to 4-bit encoder or an 256-bit to 8-bit encoder.

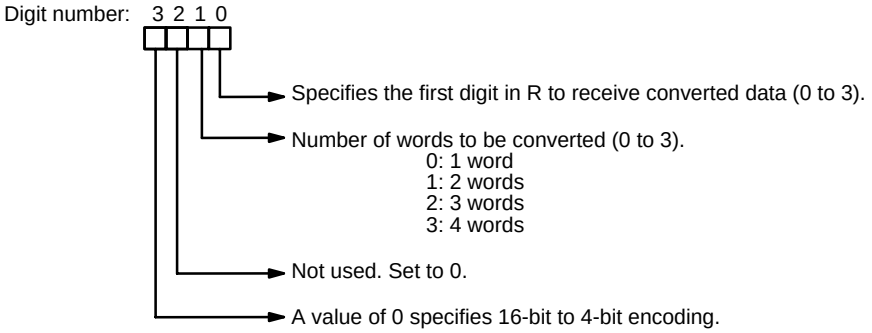
16-bit to 4-bit encoder

DMPX(77) operates as a 16-bit to 4-bit encoder when the leftmost digit of C is 0.

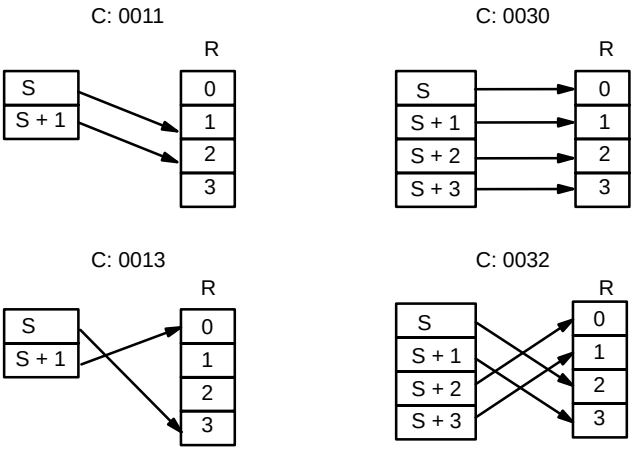
When the execution condition is OFF, DMPX(77) is not executed. When the execution condition is ON, DMPX(77) determines the position of the highest ON bit in S, encodes it into single-digit hexadecimal value corresponding to the bit number, then transfers the hexadecimal value to the specified digit in R. The digits to receive the results are specified in C, which also specifies the number of digits to be encoded.

Control Word

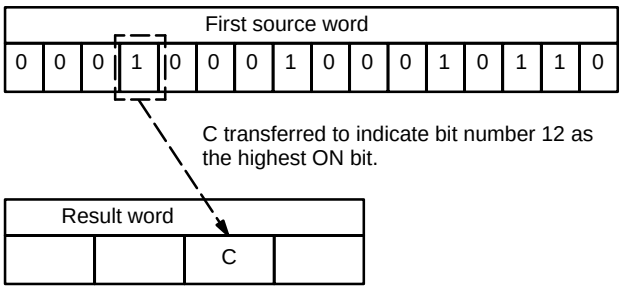
The digits of C are set as shown below. Set the leftmost digit of C to 0 to specify 16-bit to 4-bit encoding.



Some example C values and the word-to-digit conversions that they produce are shown below.



The following is an example of a one-digit encode operation to digit number 1 of R, i.e., here C would be 0001.



Up to four digits from four consecutive source words starting with S may be encoded and the digits written to R in order from the designated first digit. 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 to be converted (S plus the number of digits to be converted) must be in the same data area as SB.

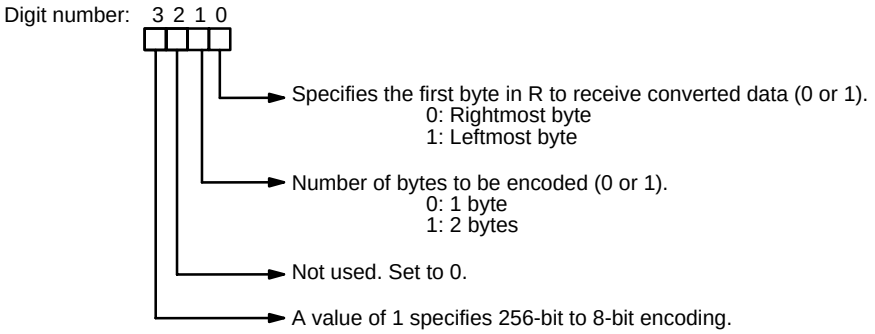
256-bit to 8-bit Encoder

DMPX(77) operates as a 256-bit to 8-bit encoder when the leftmost digit of C is set to 1.

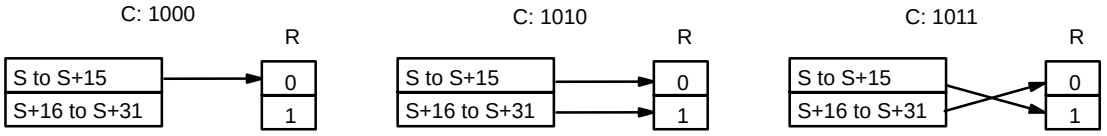
When the execution condition is OFF, DMPX(77) is not executed. When the execution condition is ON, DMPX(77) determines the position of the highest (leftmost) ON bit in the group of 16 source words from S to S+15 or S+16 to S+31, encodes it into a two-digit hexadecimal value corresponding to the location of the bit among the 256 bits in the group, then transfers the hexadecimal value to the specified byte in R. The byte to receive the result is specified in C, which also specifies the number of bytes to be encoded.

Control Word

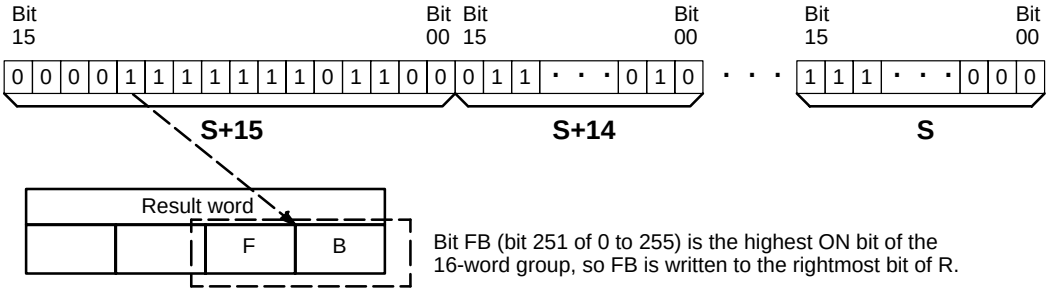
The digits of C are set as shown below. Set the leftmost digit of C to 1 to specify 256-bit to 8-bit decoding.



Three possible C values and the conversions that they produce are shown below. (In R, 0 indicates the rightmost byte and 1 indicates the leftmost byte.)



The following is an example of a one-byte encode operation to the rightmost byte of R (C would be 1000 in this case).



Flags

ER: Undefined control word.

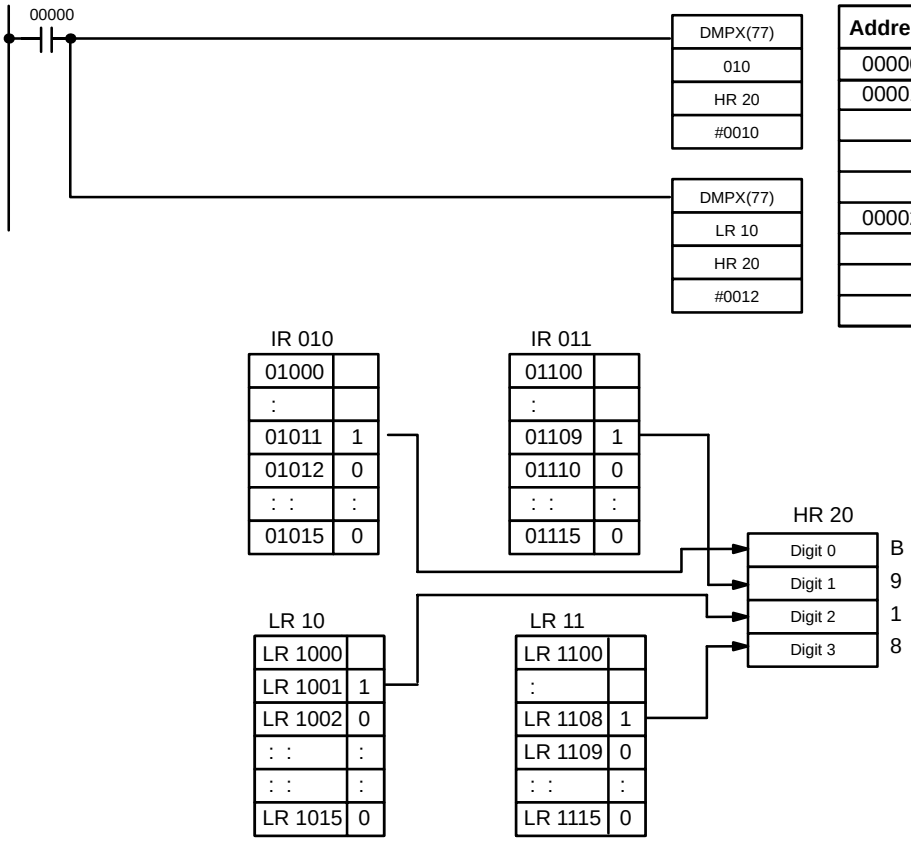
The source words are not all in the same data area.

Content of a source word is zero. (There isn't an ON bit in the source words.)

Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

Example:
16-bit to 4-bit Encoding

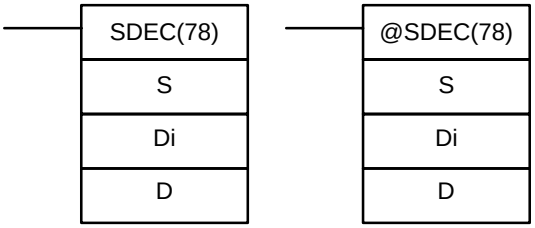
When 00000 is ON, the following diagram encodes IR words 010 and 011 to the first two digits of HR 20 and then encodes LR 10 and 11 to the last two digits of HR 20. Although the status of each source word bit is not shown, it is assumed that the bit with status 1 (ON) shown is the highest bit that is ON in the word.



Address	Instruction	Operands
00000	LD	00000
00001	DMPX(77)	
		010
		HR 20
		# 0010
00002	DMPX(77)	
		LR 10
		HR 20
		# 0012

5-18-9 7-SEGMENT DECODER – SDEC(78)

Ladder Symbols



Operand Data Areas

S: Source word (binary)
IR, SR, AR, DM, HR, TC, LR
Di: Digit designator
IR, SR, AR, DM, HR, TC, LR, #
D: First destination word
IR, SR, AR, DM, HR, LR

Limitations

Di must be within the values given below

All destination words must be in the same data area.

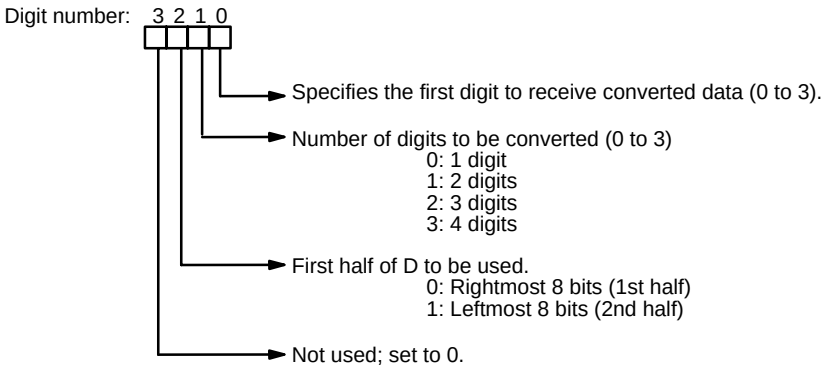
Description

When the execution condition is OFF, SDEC(78) is not executed. When the execution condition is ON, SDEC(78) converts the designated digit(s) of S into the equivalent 8-bit, 7-segment display code and places it into the destination word(s) beginning with D.

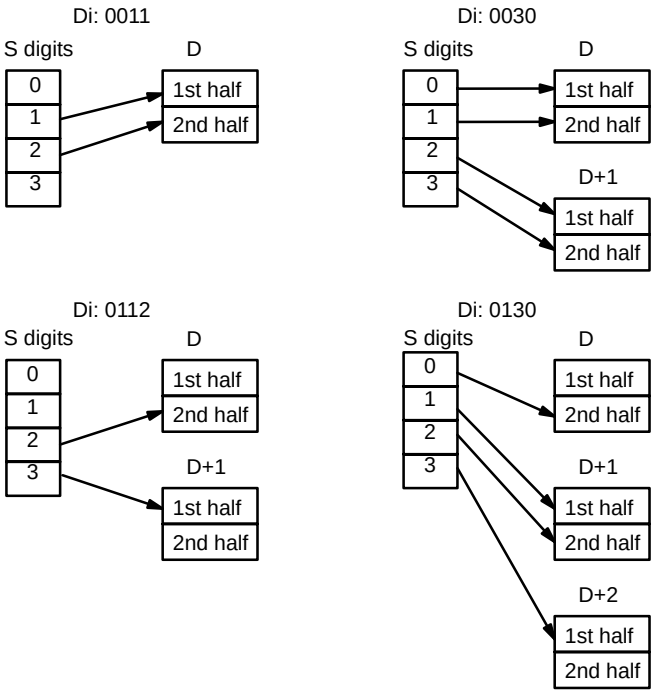
Any or all of the digits in S may be converted in sequence from the designated first digit. The first digit, the number of digits to be converted, and the half of D to receive the first 7-segment display code (rightmost or leftmost 8 bits) are designated in Di. If multiple digits are designated, they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

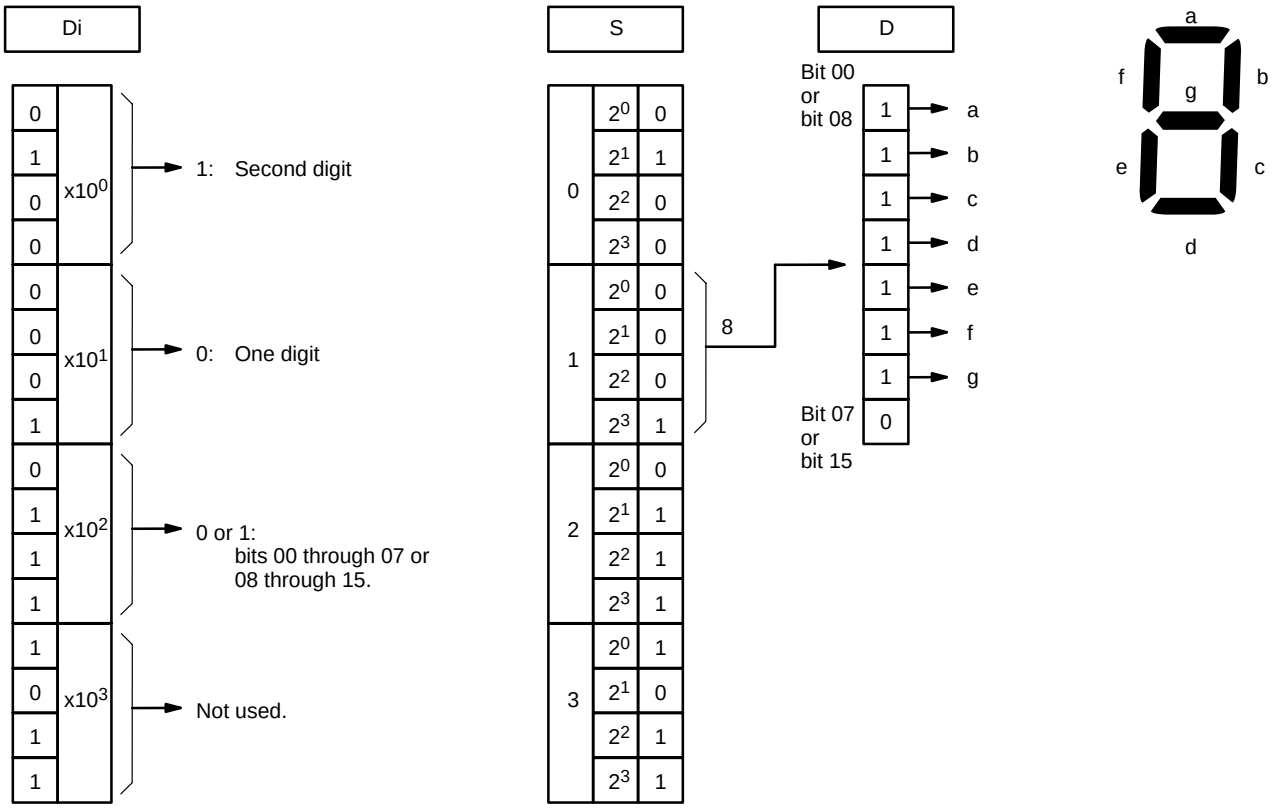


Some example Di values and the 4-bit binary to 7-segment display conversions that they produce are shown below.



Example

The following example shows the data to produce 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

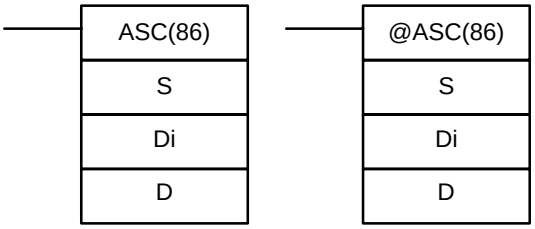
Flags

ER: Incorrect digit designator, or data area for destination exceeded

Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

5-18-10 ASCII CONVERT – ASC(86)

Ladder Symbols



Operand Data Areas

S: Source word	IR, SR, AR, DM, HR, TC, LR
Di: Digit designator	IR, SR, AR, DM, HR, TC, LR, #
D: First destination word	IR, SR, AR, DM, HR, LR

Limitations

Di must be within the values given below

All destination words must be in the same data area.

Description

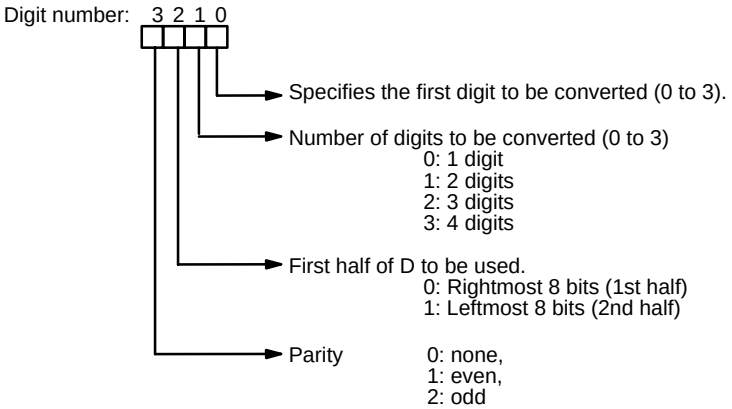
When the execution condition is OFF, ASC(86) is not executed. When the execution condition is ON, ASC(86) converts the designated digit(s) of S into the equivalent 8-bit ASCII code and places it into the destination word(s) beginning with D.

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 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.

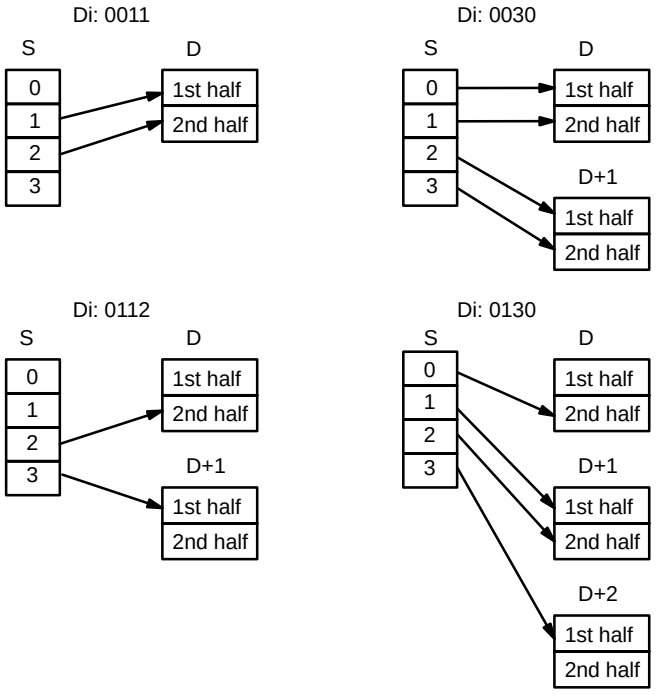
Refer to *Appendix I* for a table of extended ASCII characters.

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.

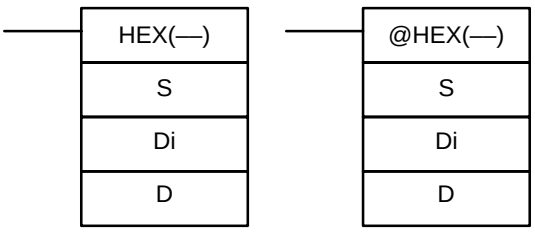
Flags

ER: Incorrect digit designator, or data area for destination exceeded.

Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

5-18-11 ASCII-TO-HEXADECIMAL – HEX(––)

Ladder Symbols



Operand Data Areas

S: First source word
IR, SR, AR, DM, HR, TC, LR

Di: Digit designator
IR, SR, AR, DM, HR, TC, LR

D: Destination word
IR, SR, AR, DM, HR, LR

Limitations

Di must be within the values given below.

All source words must be in the same data area.

Bytes in the source words must contain the ASCII code equivalent of hexadecimal values, i.e., 30 to 39 (0 to 9), 41 to 46 (A to F), or 61 to 66 (a to f).

Description

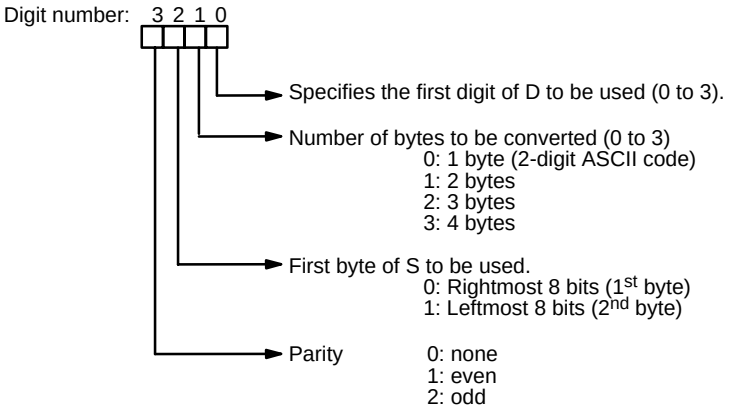
When the execution condition is OFF, HEX(—) is not executed. When the execution condition is ON, HEX(—) converts the designated byte(s) of ASCII code from the source word(s) into the hexadecimal equivalent and places it into D.

Up to 4 ASCII codes may be converted beginning with the designated first byte of S. The converted hexadecimal values are then placed in D in order from the designated digit. The first byte (rightmost or leftmost 8 bits), the number of bytes to be converted, and the digit of D to receive the first hexadecimal value are designated in Di. If multiple bytes are designated, they will be converted in order starting from the designated half of S and continuing to S+1 and S+2, if necessary.

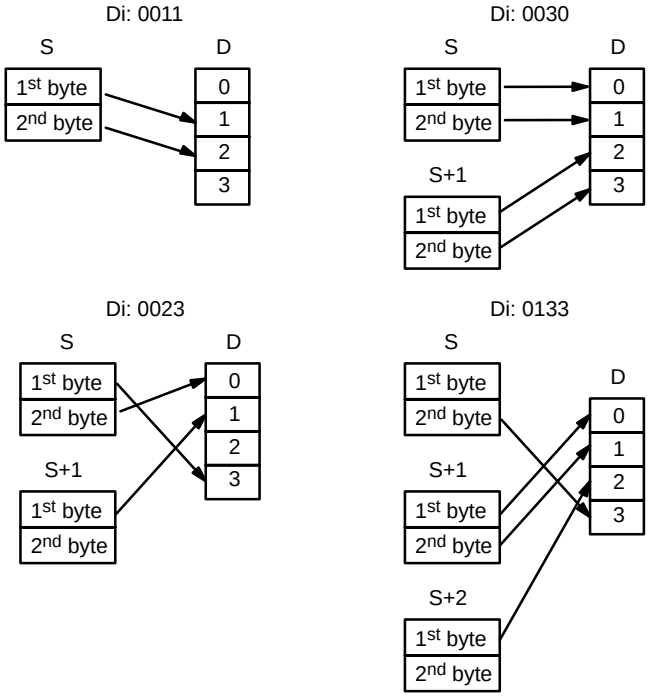
If more digits are designated than remain in D (counting from the designated first digit), further digits will be used starting back at the beginning of D. Digits in D that do not receive converted data will not be changed.

Digit Designator

The digits of Di are set as shown below.



Some examples of Di values and the 8-bit ASCII to 4-bit hexadecimal conversions that they produce are shown below.



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	*	0	1	1	0	0	0	0	0	0	0	0	0
31	*	0	1	1	0	0	0	1	1	0	0	0	1
32	*	0	1	1	0	0	1	0	2	0	0	1	0
33	*	0	1	1	0	0	1	1	3	0	0	1	1
34	*	0	1	1	0	1	0	0	4	0	1	0	0
35	*	0	1	1	0	1	0	1	5	0	1	0	1
36	*	0	1	1	0	1	1	0	6	0	1	1	0
37	*	0	1	1	0	1	1	1	7	0	1	1	1
38	*	0	1	1	1	0	0	0	8	1	0	0	0
39	*	0	1	1	1	0	0	1	9	1	0	0	1
41	*	1	0	1	0	0	0	1	A	1	0	1	0
42	*	1	0	1	0	0	1	0	B	1	0	1	1
43	*	1	0	1	0	0	1	1	C	1	1	0	0
44	*	1	0	1	0	1	0	0	D	1	1	0	1
45	*	1	0	1	0	1	0	1	E	1	1	1	0
46	*	1	0	1	0	1	1	0	F	1	1	1	1

Note The leftmost bit of each ASCII code is adjusted for parity.

Parity

The leftmost bit of each ASCII character (2 digits) is automatically adjusted for either even or odd parity.

With no parity, the leftmost bit should always be zero. With odd or even parity, the leftmost bit of each ASCII character should be adjusted so that there is an odd or even number of ON bits.

If the parity of the ASCII code in S does not agree with the parity specified in Di, the ER Flag (SR 25503) will be turned ON and the instruction will not be executed.

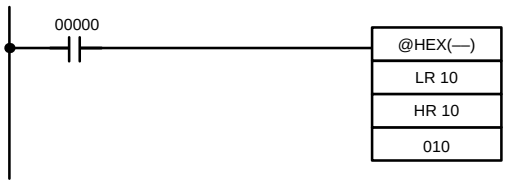
Flags

ER: Incorrect digit designator, or data area for destination exceeded.

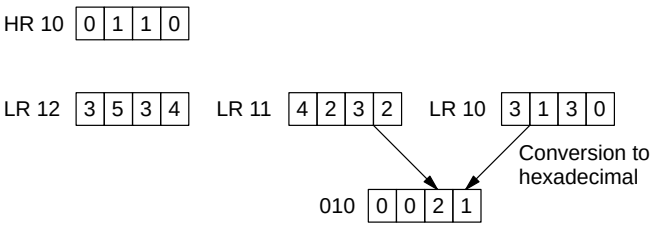
Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

Example

In the following example, the 2nd byte of LR 10 and the 1st byte of LR 11 are converted to hexadecimal values and those values are written to the first and second bytes of IR 010.

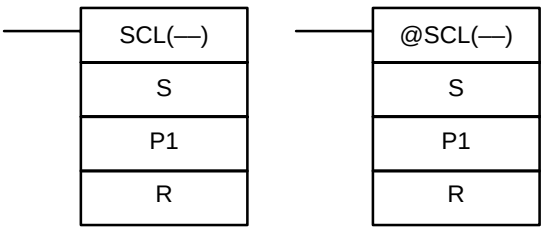


Address	Instruction	Operands
00000	LD	00000
00001	@HEX(—)	
		LR 10
		HR 10
		010



5-18-12 SCALING – SCL(––)

Ladder Symbols



Operand Data Areas

S: Source word
IR, SR, AR, DM, HR, TC, LR, #
P1: First parameter word
IR, SR, AR, DM, HR, TC, LR
R: Result word
IR, SR, AR, DM, HR, LR

Limitations

P1 and P1+2 must be BCD.

P1 through P1+3 must be in the same data area.

P1+1 and P1+3 must not be set to the same value.

Description

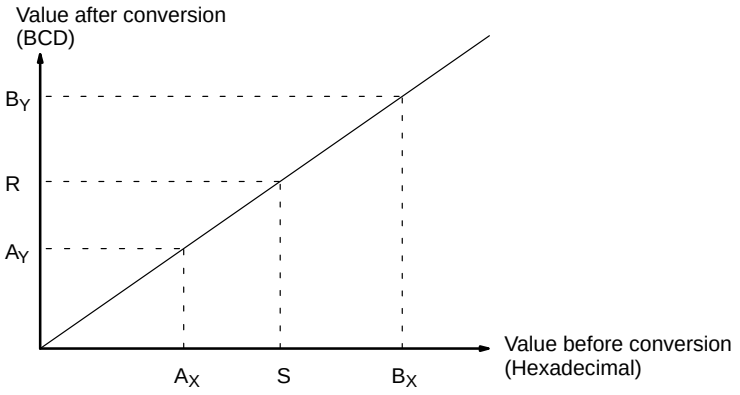
SCL(—) is used to linearly convert a 4-digit hexadecimal value to a 4-digit BCD value. Unlike BCD(24), which converts a 4-digit hexadecimal value to its 4-digit BCD equivalent ($S_{hex} \rightarrow S_{BCD}$), SCL(—) can convert the hexadecimal value according to a specified linear relationship. The conversion line is defined by two points specified in the parameter words P1 to P1+3.

When the execution condition is OFF, SCL(—) is not executed. When the execution condition is ON, SCL(—) converts the 4-digit hexadecimal value in S to the 4-digit BCD value on the line defined by points (P1, P1+1) and (P1+2, P1+3) and places the result in R. The result is rounded off to the nearest integer. If the result is less than 0000, then 0000 is written to R, and if the result is greater than 9999, then 9999 is written to R.

The following table shows the functions and ranges of the parameter words:

Parameter	Function	Range	Comments
P1	BCD point #1 (A_Y)	0000 to 9999	---
P1+1	Hex. point #1 (A_X)	0000 to FFFF	Do not set P1+1=P1+3.
P1+2	BCD point #2 (B_Y)	0000 to 9999	---
P1+3	Hex. point #2 (B_X)	0000 to FFFF	Do not set P1+3=P1+1.

The following diagram shows the source word, S, converted to D according to the line defined by points (A_Y, A_X) and (B_Y, B_X).



The results can be calculated by first converting all values to BCD and then using the following formula.

$$\text{Results} = B_Y - [(B_Y - A_Y)/(B_X - A_X) \times (B_X - S)]$$

Flags

ER: The value in P1+1 equals that in P1+3.

Indirectly addressed DM word is non-existent. (Content of *DM word is not BCD, or the DM area boundary has been exceeded.)

P1 and P1+3 are not in the same data area, or other setting error.

EQ: ON when the result, R, is 0000.

Example

When 00000 is turned ON in the following example, the BCD source data in DM 0100 (#0100) is converted to hexadecimal according to the parameters in DM 0150 to DM 0153. The result (#0512) is then written to DM 0200.

Address	Instruction	Operands	
00000	LD		00000
00001	@SCL(—)		
		DM	0100
		DM	0150
		DM	0200

DM 0150	0010
DM 0151	0005
DM 0152	0050
DM 0153	0019

DM 0100	0100

DM 0200	0512

5-18-13 COLUMN TO LINE – LINE(63)

Ladder Symbols

LINE(63)
S
C
D

@LINE(63)
S
C
D

Operand Data Areas

S: First word of 16 word source set
IR, SR, AR, DM, HR, TC, LR

C: Column bit designator (BCD)
IR, SR, AR, DM, HR, TC, LR, #

D: Destination word
IR, SR, AR, DM, HR, TC, LR

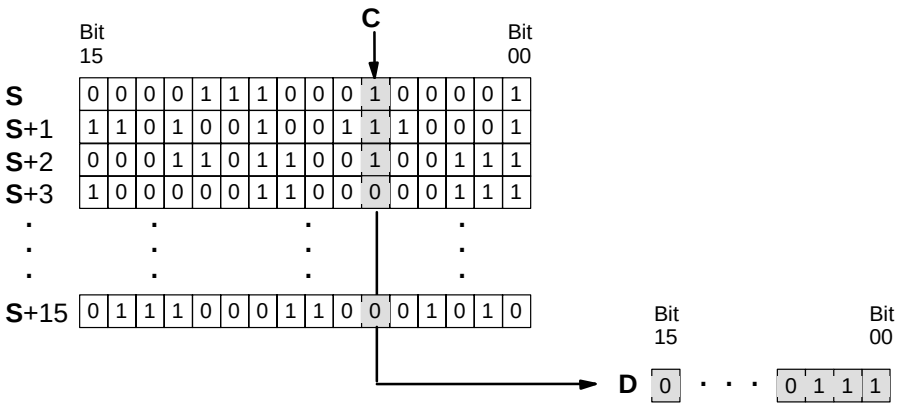
Limitations

S and S+15 must be in the same data area.

C must be between #0000 and #0015.

Description

When the execution condition is OFF, LINE(63) is not executed. When the execution condition is ON, LINE(63) copies bit column C from the 16-word set (S to S+15) to the 16 bits of word D (00 to 15).



Flags

ER: The column bit designator C is not BCD, or it is specifying a non-existent bit (i.e., bit specification must be between 00 and 15).

Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON when the content of D is zero; otherwise OFF.

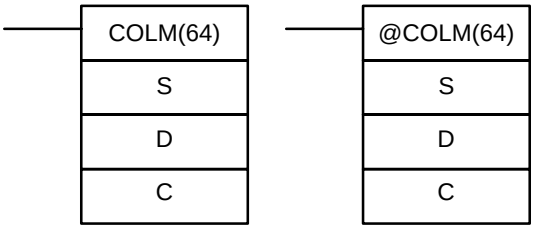
Example

The following example shows how to use LINE(63) to move bit column 07 from the set (IR 100 to IR 115) to DM 0100.

Address	Instruction	Operands
00000	LD	00000
00001	LINE(63)	
		100
		# 0007
		DM 0100

5-18-14 LINE TO COLUMN – COLM(64)

Ladder Symbols



Operand Data Areas

S: Source word
IR, SR, AR, DM, HR, TC, LR
D: First word of the destination set
IR, AR, DM, HR, TC, LR
C: Column bit designator (BCD)
IR, SR, AR, DM, HR, TC, LR, #

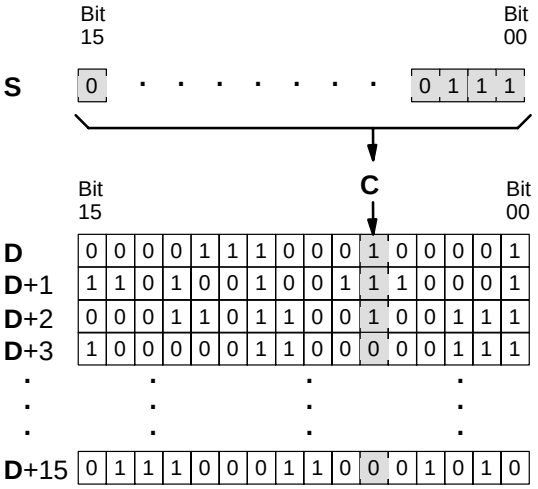
Limitations

D and D+15 must be in the same data area.

C must be between #0000 and #0015.

Description

When the execution condition is OFF, COLM(64) is not executed. When the execution condition is ON, COLM(64) copies the 16 bits of word S (00 to 15) to the column of bits, C, of the 16-word set (D to D+15).



Flags

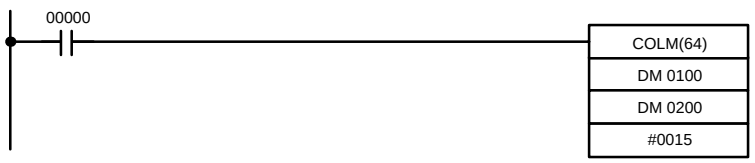
ER: The bit designator C is not BCD, or it is specifying a non-existent bit (i.e., bit specification must be between 00 and 15).

Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON when the content of S is zero; otherwise OFF.

Example

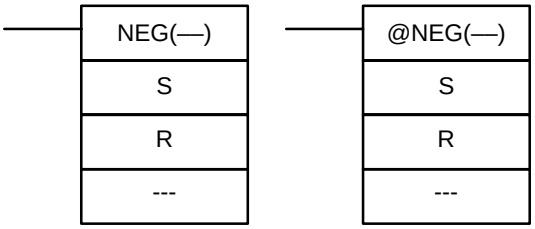
The following example shows how to use COLM(64) to move the contents of word DM 0100 (00 to 15) to bit column 15 of the set (DM 0200 to DM 0215).



Address	Instruction	Operands
00000	LD	00000
00001	COLM(64)	
		DM 0100
		DM 0200
		# 0015

5-18-15 2'S COMPLEMENT – NEG(––)

Ladder Symbols



Operand Data Areas

S: Source word
IR, SR, AR, DM, HR, TC, LR, #

R: Result word
IR, SR, AR, DM, HR, LR

Description

Converts the four-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.

If the content of S is 0000, the content of R will also be 0000 after execution, and EQ (SR 25506) will be turned on.

If the content of S is 8000, the content of R will also be 8000 after execution, and UF (SR 25405) will be turned on.

Note Refer to page 29 for details on 16-bit signed binary data.

Flags

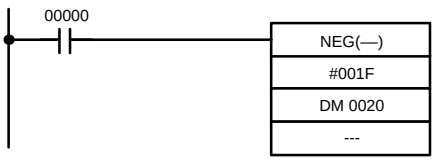
ER: Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON when the content of S is 0000; otherwise OFF.

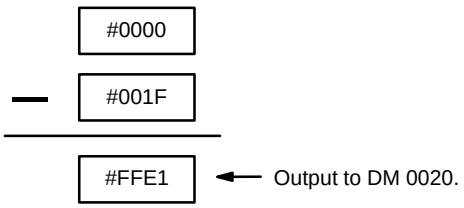
UF: ON when the content of S is 8000; otherwise OFF.

Example

The following example shows how to use NEG(—) to find the 2's complement of the hexadecimal value 001F and output the result to DM 0020.



Address	Instruction	Operands	
00000	LD		00000
00001	NEG(—)		
		#	001F
		DM	0020



5-18-16 DOUBLE 2'S COMPLEMENT – NEGL(––)

Ladder Symbols

NEGL(—)
S
R

@NEGL(—)
S
R

Operand Data Areas

S: First source word
IR, SR, AR, DM, HR, TC, LR

R: First result word
IR, SR, AR, DM, HR, LR

Limitations

S and S+1 must be in the same data area, as must R and R+1.

Description

Converts the eight-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 eight-digit content S and S+1 from $0000 0000 and outputting the result to R and R+1.

If the content of S is 0000 0000, the content of R will also be 0000 0000 after execution and EQ (SR 25506) will be turned on.

If the content of S is 8000 0000, the content of R will also be 8000 0000 after execution and UF (SR 25405) will be turned on.

Note Refer to page 29 for details on 32-bit signed binary data.

Flags

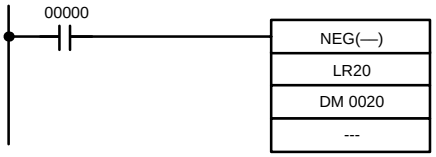
ER: Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON when the content of S+1, S is 0000 0000; otherwise OFF.

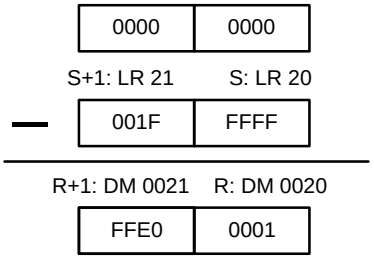
UF: ON when the content of S+1, S is 8000 0000; otherwise OFF.

Example

The following example shows how to use NEGL(—) to find the 2's complement of the hexadecimal value in LR 21, LR 20 (001F FFFF) and output the result to DM 0021, DM 0020.



Address	Instruction	Operands
00000	LD	00000
00001	NEGL(—)	
		LR 20
		DM 0020



5-19 BCD Calculations

The BCD calculation instructions – INC(38), DEC(39), ADD(30), ADDL(54), SUB(31), SUBL(55), MUL(32), MULL(56), DIV(33), DIVL(57), FDIV(79), and ROOT(72) – all perform arithmetic operations on BCD data.

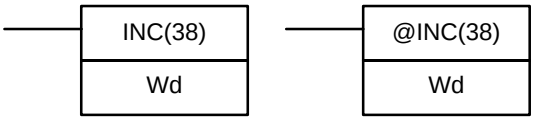
For INC(38) and DEC(39) the source and result words are the same. That is, the content of the source word is overwritten with the instruction result. All other instructions change only the content of the words in which results are placed, i.e., the contents of source words are the same before and after execution of any of the other BCD calculation instructions.

STC(40) and CLC(41), which set and clear the carry flag, are included in this group because most of the BCD operations make use of the Carry Flag (CY) in their results. Binary calculations and shift operations also use CY.

The addition and subtraction instructions include CY in the calculation as well as in the result. Be sure to clear CY if its previous status is not required in the calculation, and to use the result placed in CY, if required, before it is changed by execution of any other instruction.

5-19-1 INCREMENT – INC(38)

Ladder Symbols



Operand Data Areas

Wd: Increment word (BCD)
IR, SR, AR, DM, HR, LR

Description When the execution condition is OFF, INC(38) is not executed. When the execution condition is ON, INC(38) increments Wd, without affecting Carry (CY).

Flags

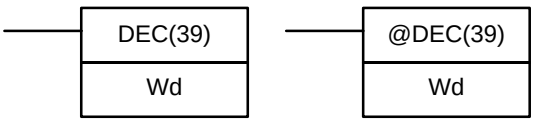
ER: Wd is not BCD

Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON when the incremented result is 0.

5-19-2 DECREMENT – DEC(39)

Ladder Symbols



Operand Data Areas

Wd: Decrement word (BCD)
IR, SR, AR, DM, HR, LR

Description When the execution condition is OFF, DEC(39) is not executed. When the execution condition is ON, DEC(39) decrements Wd, without affecting CY. DEC(39) works the same way as INC(38) except that it decrements the value instead of incrementing it.

Flags

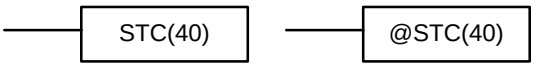
ER: Wd is not BCD

Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON when the decremented result is 0.

5-19-3 SET CARRY – STC(40)

Ladder Symbols

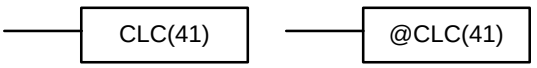


When the execution condition is OFF, STC(40) is not executed.When the execution condition is ON, STC(40) turns ON CY (SR 25504).

Note Refer to *Appendix C Error and Arithmetic Flag Operation* for a table listing the instructions that affect CY.

5-19-4 CLEAR CARRY – CLC(41)

Ladder Symbols



When the execution condition is OFF, CLC(41) is not executed.When the execution condition is ON, CLC(41) turns OFF CY (SR 25504).

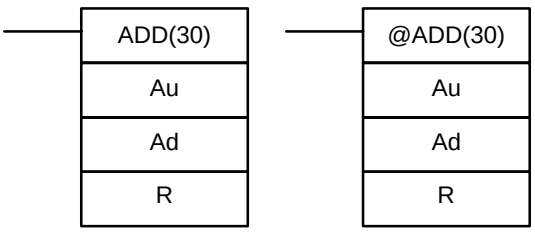
CLEAR CARRY is used to reset (turn OFF) CY (SR 25504) to “0”.

CY is automatically reset to “0” when END(01) is executed at the end of each cycle.

Note Refer to *Appendix C Error and Arithmetic Flag Operation* for a table listing the instructions that affect CY.

5-19-5 BCD ADD – ADD(30)

Ladder Symbols

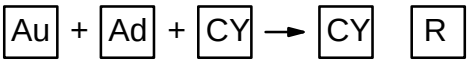


Operand Data Areas

Au: Augend word (BCD)
IR, SR, AR, DM, HR, TC, LR, #
Ad: Addend word (BCD)
IR, SR, AR, DM, HR, TC, LR, #
R: Result word
IR, SR, AR, DM, HR, LR

Description

When the execution condition is OFF, ADD(30) is not executed. When the execution condition is ON, ADD(30) 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.



Flags

ER: Au and/or Ad is not BCD.

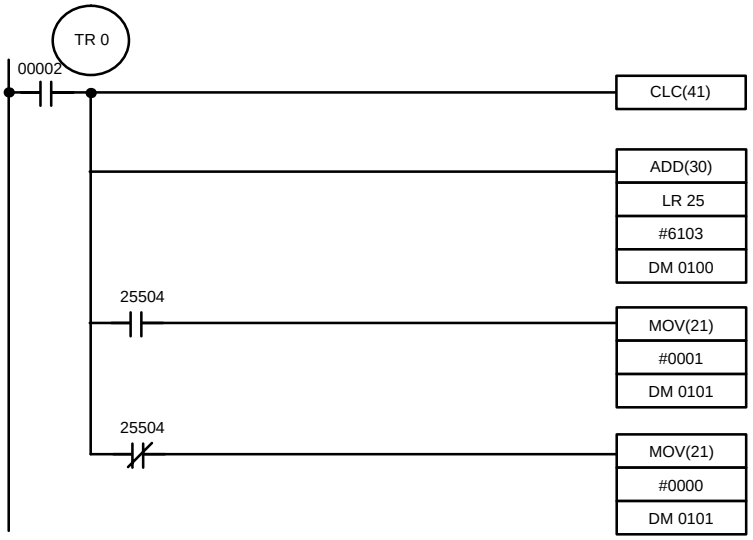
Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

CY: ON when there is a carry in the result.

EQ: ON when the result is 0.

Example

If 00002 is ON, the program represented by the following diagram clears CY with CLC(41), adds the content of LR 25 to a constant (6103), places the result in DM 0100, and then moves either all zeros or 0001 into DM 0101 depending on the status of CY (25504). This ensures that any carry from the last digit is preserved in R+1 so that the entire result can be later handled as eight-digit data.

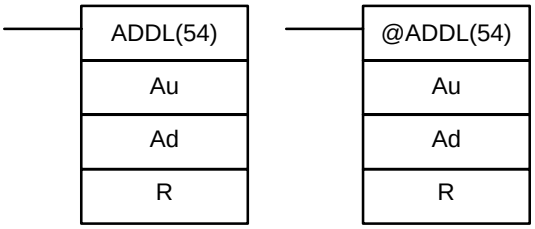


Address	Instruction	Operands	
00000	LR		00002
00001	OUT	TR	0
00002	CLC(41)		
00003	AND(30)		
		LR	25
		#	6103
		DM	0100
00004	AND		25504
00005	MOV(21)		
		#	0001
		DM	0101
00006	LD	TR	0
00007	AND NOT		25504
00008	MOV(21)		
		#	0000
		DM	0101

Although two ADD(30) can be used together to perform eight-digit BCD addition, ADDL(54) is designed specifically for this purpose.

5-19-6 DOUBLE BCD ADD – ADDL(54)

Ladder Symbols



Operand Data Areas

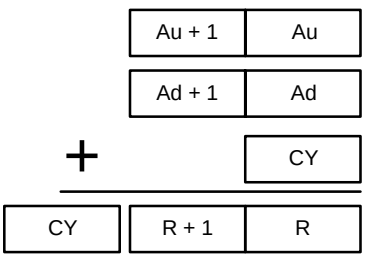
Au: First augend word (BCD)
IR, SR, AR, DM, HR, TC, LR
Ad: First addend word (BCD)
IR, SR, AR, DM, HR, TC, LR
R: First result word
IR, SR, AR, DM, HR, LR

Limitations

Each of the following pairs must be in the same data area: Au and Au+1, Ad and Ad+1, and R and R+1.

Description

When the execution condition is OFF, ADDL(54) is not executed. When the execution condition is ON, ADDL(54) adds the contents 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 99999999.



Flags

ER: Au and/or Ad is not BCD.

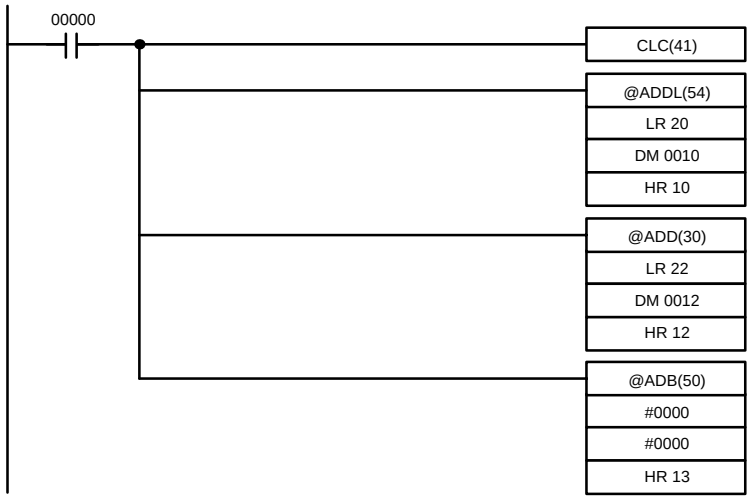
Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

CY: ON when there is a carry in the result.

EQ: ON when the result is 0.

Example

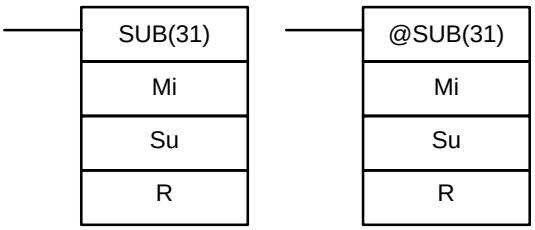
When 00000 is ON, the following program adds two 12-digit numbers, the first contained in LR 20 through LR 22 and the second in DM 0012. The result is placed in LR 10 through HR 13. In the second addition (using ADD(30)), any carry from the first addition is included. The carry from the second addition is placed in HR 13 by using @ADB(50) (see *5-20-1 BINARY ADD – ADB(50)*) with two all-zero constants to indirectly place the content of CY into HR 13.



Address	Instruction	Operands	
00000	LD		00000
00001	CLC(41)		
00002	@ADDL(54)		
		LR	20
		DM	0010
		HR	10
00003	@ADD(30)		
		LR	22
		DM	0012
		HR	12
00004	@ADB(50)		
		#	0000
		#	0000
		HR	13

5-19-7 BCD SUBTRACT – SUB(31)

Ladder Symbols



Operand Data Areas

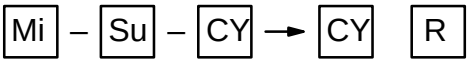
Mi: Minuend word (BCD)
IR, SR, AR, DM, HR, TC, LR, #

Su: Subtrahend word (BCD)
IR, SR, AR, DM, HR, TC, LR, #

R: Result word
IR, SR, AR, DM, HR, LR

Description

When the execution condition is OFF, SUB(31) is not executed. When the execution condition is ON, SUB(31) 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).



Note The 2's COMPLEMENT – NEG(—) instruction can be used to convert binary data only, it cannot be used with BCD data.

Flags

ER: Mi and/or Su 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.)

CY: ON when the result is negative, i.e., when Mi is less than Su plus CY.

EQ: ON when the result is 0.

⚠ **Caution** Be sure to clear the carry flag with CLC(41) before executing SUB(31) if its previous status is not required, and check the status of CY after doing a subtraction with SUB(31). If CY is ON as a result of executing SUB(31) (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.

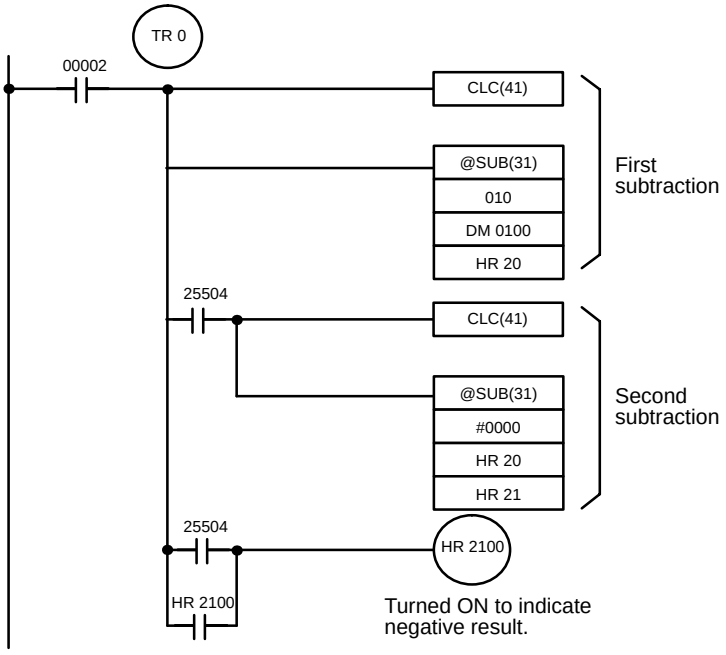
Example

When 00002 is ON, the following ladder program clears CY, subtracts the contents of DM 0100 and CY from the content of 010 and places the result in HR 20.

If CY is set by executing SUB(31), the result in HR 20 is subtracted from zero (note that CLC(41) is again required to obtain an accurate result), the result is placed back in HR 20, and HR 2100 is turned ON to indicate a negative result.

If CY is not set by executing SUB(31), the result is positive, the second subtraction is not performed, and HR 2100 is not turned ON. HR 2100 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 recycled.

In this example, differentiated forms of SUB(31) are used so that the subtraction operation is performed only once each time 00002 is turned ON. When another subtraction operation is to be performed, 00002 will need to be turned OFF for at least one cycle (resetting HR 2100) and then turned back ON.



Address	Instruction	Operands	
00000	LD		00002
00001	OUT	TR	0
00002	CLC(41)		
00003	@SUB(31)		
			010
		DM	0100
		HR	20
00004	AND		25504
00005	CLC(41)		
00006	@SUB(31)		
		#	0000
		HR	20
		HR	20
00007	LD	TR	0
00008	AND		25504
00009	OR	HR	2100
00010	OUT	HR	2100

The first and second subtractions for this diagram are shown below using example data for 010 and DM 0100.

Note The actual SUB(31) 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

IR 010	1029	
DM 0100	– 3452	
CY	– 0	
HR 20	7577	(1029 + (10000 – 3452))
CY	1	(negative result)

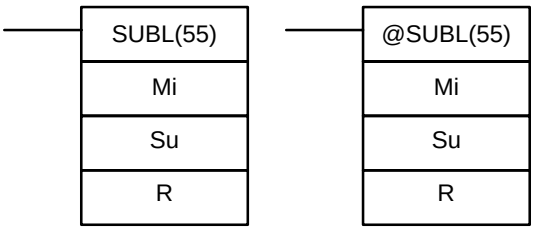
Second Subtraction

	0000	
HR 20	–7577	
CY	–0	
HR 20	2423	(0000 + (10000 – 7577))
CY	1	(negative result)

In the above case, the program would turn ON HR 2100 to indicate that the value held in HR 20 is negative.

5-19-8 DOUBLE BCD SUBTRACT – SUBL(55)

Ladder Symbols

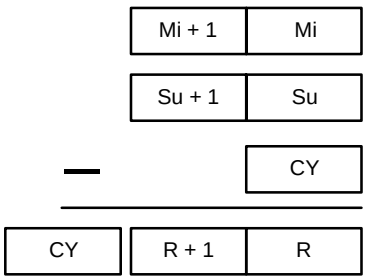


Operand Data Areas

Mi: First minuend word (BCD)	IR, SR, AR, DM, HR, TC, LR
Su: First subtrahend word (BCD)	IR, SR, AR, DM, HR, TC, LR
R: First result word	IR, SR, AR, DM, HR, LR

Limitations Each of the following pairs must be in the same data area: Mi and Mi+1, Su and Su+1, and R and R+1.

Description When the execution condition is OFF, SUBL(55) is not executed. When the execution condition is ON, SUBL(55) subtracts CY and the 8-digit contents 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. Since an 8-digit constant cannot be directly entered, use the BSET(71) instruction (see *5-16-3 BLOCK SET – BSET(71)*) to create an 8-digit constant.



Note The DOUBLE 2's COMPLEMENT – NEGL(—) instruction can be used to convert binary data only, it cannot be used with BCD data.

Flags

ER: Mi, M+1,Su, or Su+1 are not BCD.

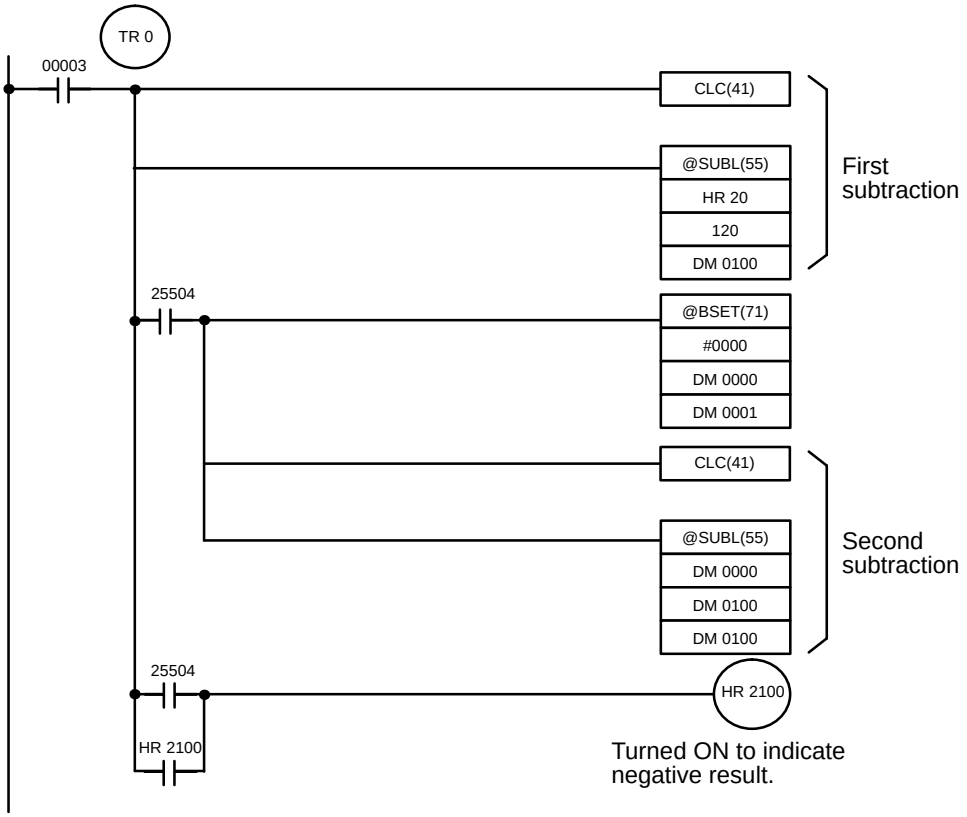
Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

CY: ON when the result is negative, i.e., when Mi is less than Su.

EQ: ON when the result is 0.

The following example works much like that for single-word subtraction. In this example, however, BSET(71) is required to clear the content of DM 0000 and DM 0001 so that a negative result can be subtracted from 0 (inputting an 8-digit constant is not possible).

Example



Address	Instruction	Operands	
00000	LD		00003
00001	OUT	TR	0
00002	CLC(41)		
00003	@SUBL(55)		
		HR	20
			120
		DM	0100
00004	AND		25504
00005	@BSET(71)		
		#	0000
		DM	0000
		DM	0001

Address	Instruction	Operands	
00006	CLC(41)		
00007	@SUBL(55)		
		DM	0000
		DM	0100
		DM	0100
00008	LD	TR	0
00009	AND		25504
00010	OR	HR	2100
00011	OUT	HR	2100

5-19-9 BCD MULTIPLY – MUL(32)

Ladder Symbols

MUL(32)
Md
Mr
R

@MUL(32)
Md
Mr
R

Operand Data Areas

Md: Multiplicand (BCD)
IR, SR, AR, DM, HR, TC, LR, #

Mr: Multiplier (BCD)
IR, SR, AR, DM, HR, TC, LR, #

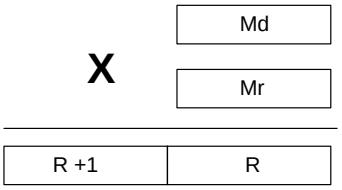
R: First result word
IR, SR, AR, DM, HR LR

Limitations

R and R+1 must be in the same data area.

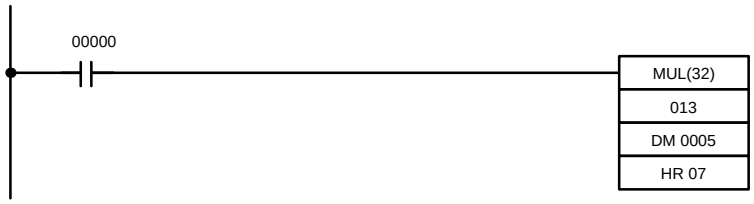
Description

When the execution condition is OFF, MUL(32) is not executed. When the execution condition is ON, MUL(32) multiplies Md by the content of Mr, and places the result In R and R+1.

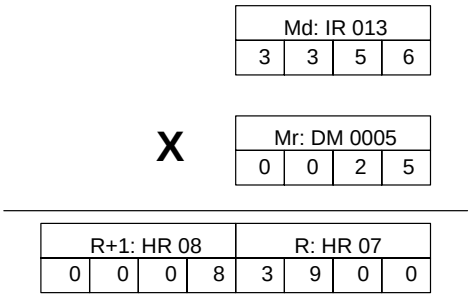


Example

When IR 00000 is ON with the following program, the contents of IR 013 and DM 0005 are multiplied and the result is placed in HR 07 and HR 08. Example data and calculations are shown below the program.



Address	Instruction	Operands
00000	LD	00000
00001	MUL(32)	
		013
		DM 00005
		HR 07



Flags

ER: Md and/or Mr is not BCD.

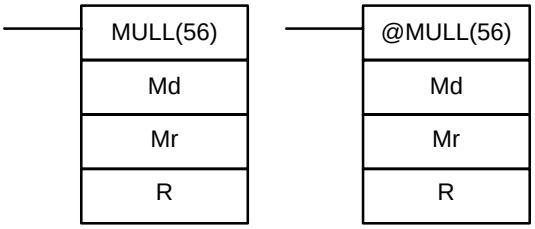
Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

CY: ON when there is a carry in the result.

EQ: ON when the result is 0.

5-19-10 DOUBLE BCD MULTIPLY – MULL(56)

Ladder Symbols



Operand Data Areas

Md: First multiplicand word (BCD)
IR, SR, AR, DM, HR, TC, LR
Mr: First multiplier word (BCD)
IR, SR, AR, DM, HR, TC, LR
R: First result word
IR, SR, AR, DM, HR LR

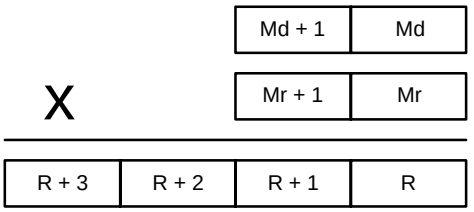
Limitations

Md and Md+1 must be in the same data area, as must Mr and Mr+1.

R and R+3 must be in the same data area.

Description

When the execution condition is OFF, MULL(56) is not executed. When the execution condition is ON, MULL(56) multiplies the eight-digit content of Md and Md+1 by the content of Mr and Mr+1, and places the result in R to R+3.



Flags

ER: Md, Md+1,Mr, or Mr+1 is not BCD.

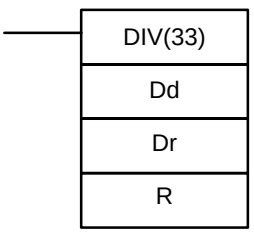
Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

CY: ON when there is a carry in the result.

EQ: ON when the result is 0.

5-19-11 BCD DIVIDE – DIV(33)

Ladder Symbol



Operand Data Areas

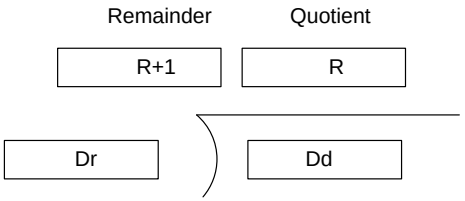
Dd: Dividend word (BCD)
IR, SR, AR, DM, HR, TC, LR, #
Dr: Divisor word (BCD)
IR, SR, AR, DM, HR, TC, LR, #
R: First result word (BCD)
IR, SR, AR, DM, HR, LR

Limitations

R and R+1 must be in the same data area.

Description

When the execution condition is OFF, DIV(33) 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.



Flags

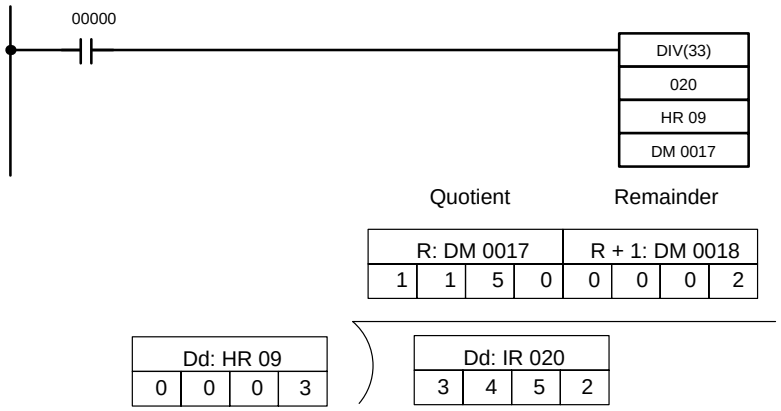
ER: Dd or Dr is not in BCD.

Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON when the result is 0.

Example

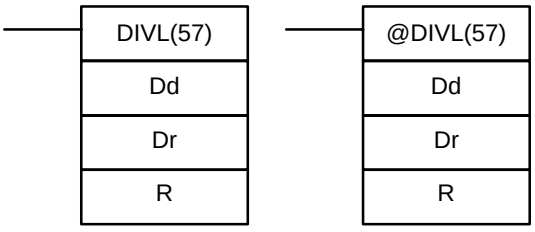
When IR 00000 is ON with the following program, the content of IR 020 is divided by the content of HR 09 and the result is placed in DM 0017 and DM 0018. Example data and calculations are shown below the program.



Address	Instruction	Operands
00000	LD	00000
00001	DIV(33)	
		020
		HR 09
		DM 0017

5-19-12 DOUBLE BCD DIVIDE – DIVL(57)

Ladder Symbols



Operand Data Areas

Dd: First dividend word (BCD)
IR, SR, AR, DM, HR, TC, LR
Dr: First divisor word (BCD)
IR, SR, AR, DM, HR, TC, LR
R: First result word
IR, SR, AR, DM, HR, LR

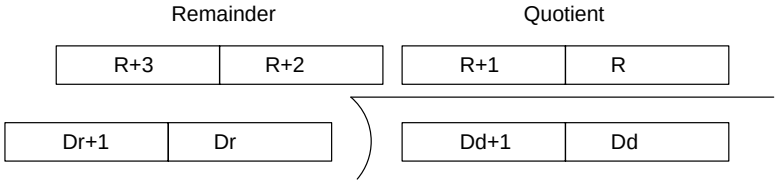
Limitations

Dd and Dd+1 must be in the same data area, as must Dr and Dr+1.

R and R+3 must be in the same data area.

Description

When the execution condition is OFF, DIVL(57) is not executed. When the execution condition is ON, DIVL(57) the eight-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, the remainder in R+2 and R+3.



Flags

ER: Dr and Dr+1 contain 0.

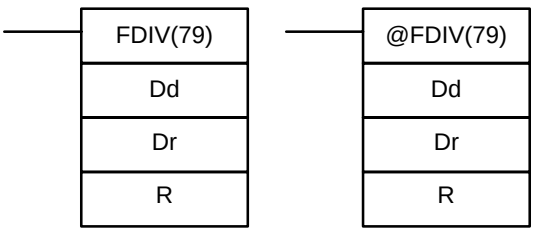
Dd, Dd+1, Dr, or Dr+1 is not BCD.

Indirectly addressed DM word is non-existent. (Content of *DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON when the result is 0.

5-19-13 FLOATING POINT DIVIDE – FDIV(79)

Ladder Symbols



Operand Data Areas

Dd: First dividend word (BCD)
IR, SR, AR, DM, HR, TC, LR
Dr: First divisor word (BCD)
IR, SR, AR, DM, HR, TC, LR
R: First result word
IR, SR, AR, DM, HR, LR

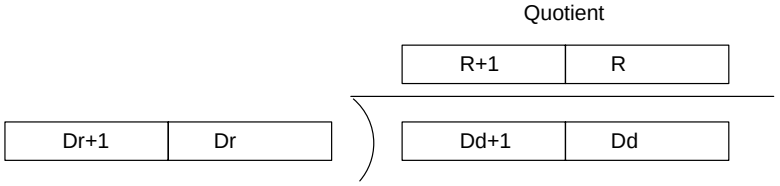
Limitations

Dr and Dr+1 cannot contain zero. Dr and Dr+1 must be in the same data area, as must Dd and Dd+1; R and R+1.

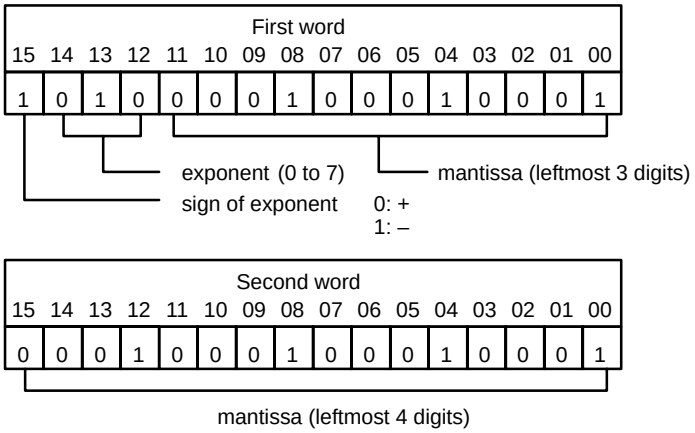
The dividend and divisor must be between 0.0000001×10^{-7} and 0.9999999×10^{7}. The results must be between 0.1×10^{-7} and 0.9999999×10^{7}.

Description

When the execution condition is OFF, FDIV(79) is not executed. When the execution condition is ON, FDIV(79) 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 below. The mantissa is expressed as a value less than one, i.e., to seven decimal places.



$= 0.1111111 \times 10^{-2}$

Flags

ER: 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}.

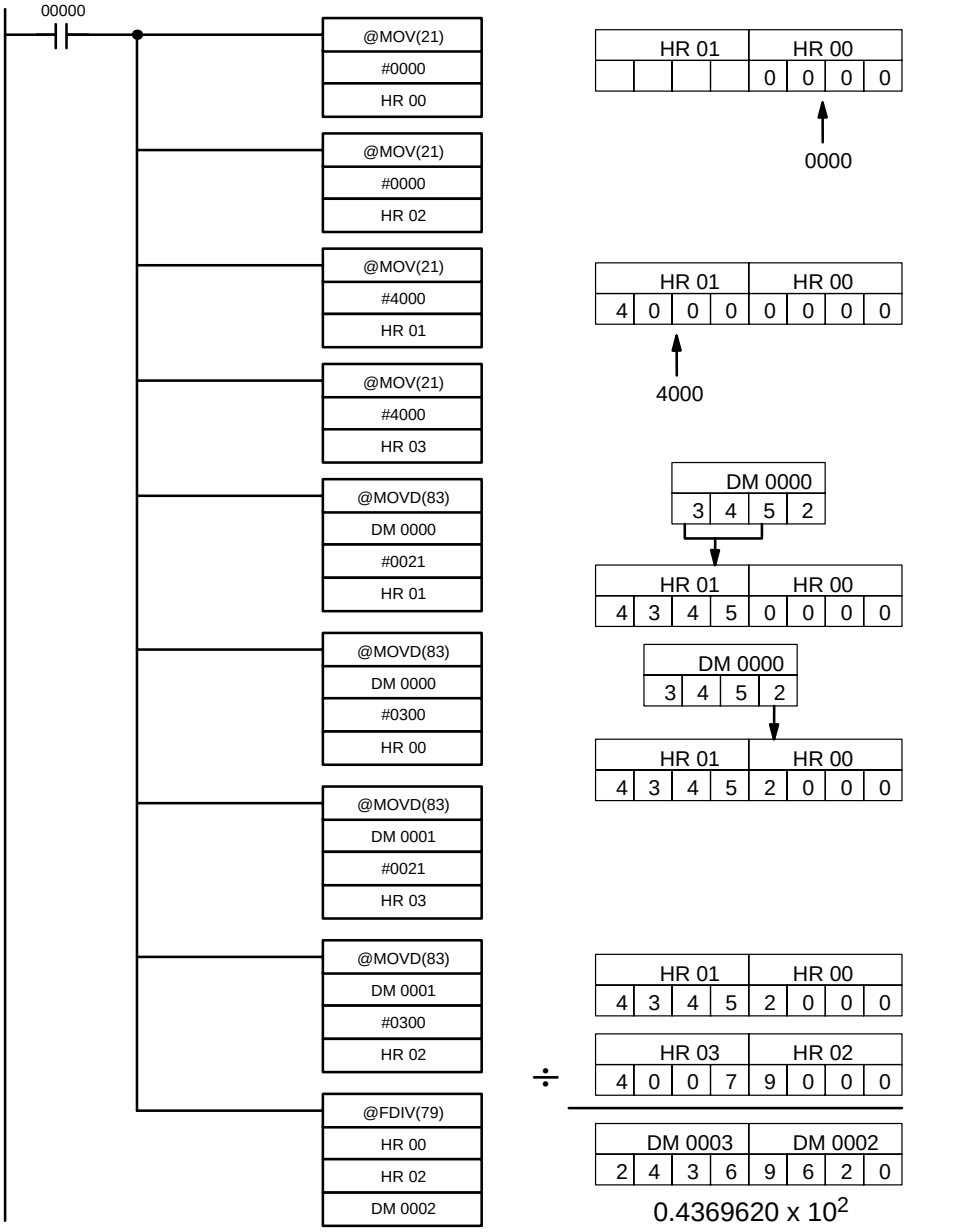
Indirectly addressed DM word is non-existent. (Content of *DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON when the result is 0.

Example

The following example shows how to divide two whole four-digit 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 DM 0000 and DM 0001. The final division is also shown.

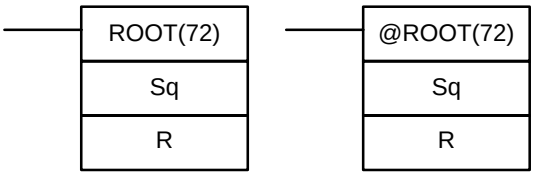


Address	Instruction	Operands
00000	LD	00000
00001	@MOV(21)	
		# 0000
		HR 00
00002	@MOV(21)	
		# 0000
		HR 02
00003	@MOV(21)	
		# 4000
		HR 01
00004	@MOV(21)	
		# 4000
		HR 03
00005	@MOVD(83)	
		DM 0000
		# 0021
		HR 01

Address	Instruction	Operands
00006	@MOVD(83)	
		DM 0000
		# 0300
		HR 00
00007	@MOVD(83)	
		DM 0001
		# 0021
		HR 03
00008	@MOVD(83)	
		DM 0001
		# 0300
		HR 02
00009	@FDIV(79)	
		HR 00
		HR 02
		DM 0002

5-19-14 SQUARE ROOT – ROOT(72)

Ladder Symbols



Operand Data Areas

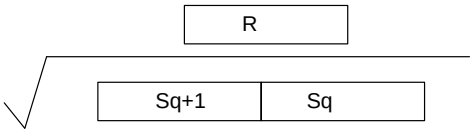
Sq: First source word (BCD)	
IR, SR, AR, DM, HR, TC, LR	
R: Result word	
IR, SR, AR, DM, HR, LR,	

Limitations

Sq and Sq+1 must be in the same data area.

Description

When the execution condition is OFF, ROOT(72) is not executed. When the execution condition is ON, ROOT(72) computes the square root of the eight-digit content of Sq and Sq+1 and places the result in R. The fractional portion is truncated.



Flags

ER: Sq 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.)

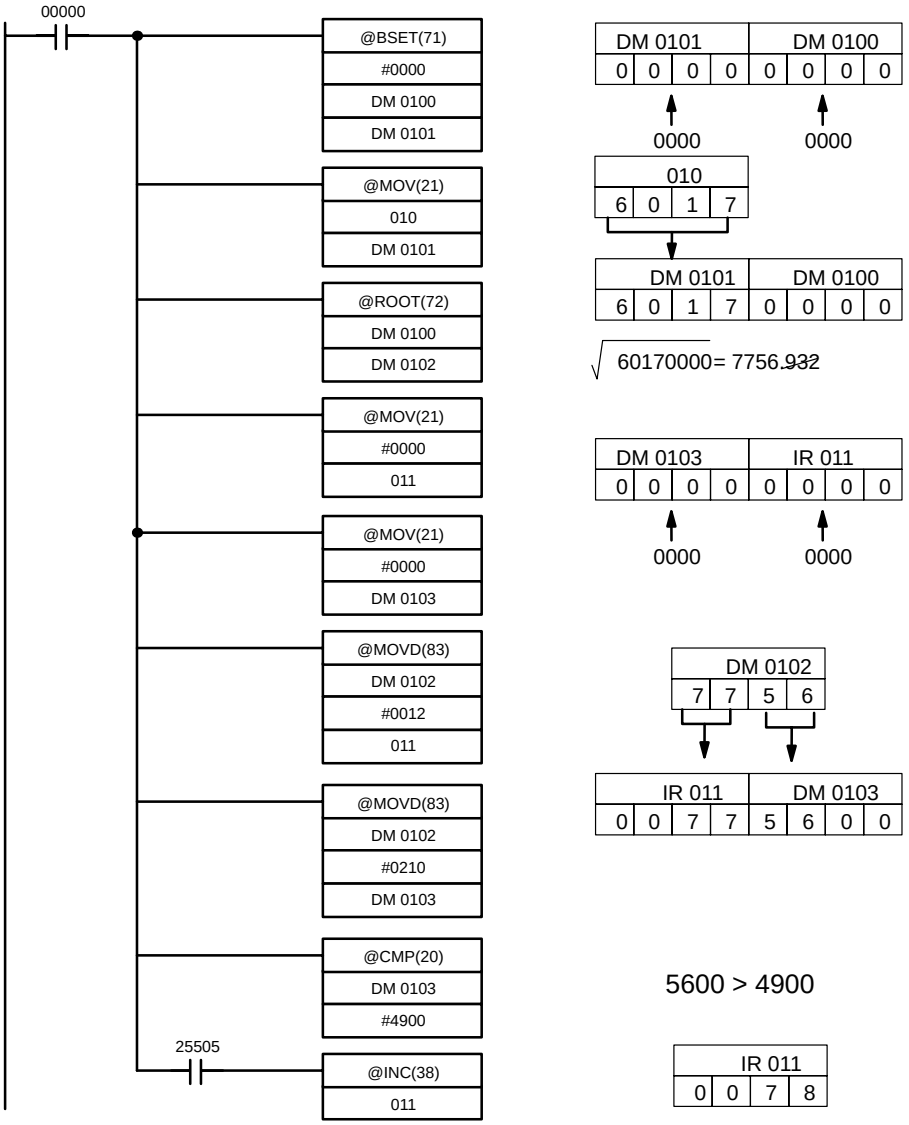
EQ: ON when the result is 0.

Example

The following example shows how to take the square root of a four-digit number and then round the result.

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 required for the answer (because the number of digits in the original value was doubled), is placed in DM 0102, and the digits are split into two different words, the leftmost two digits to IR 011 for the answer and the rightmost two digits to DM 0103 so that the answer in IR 011 can be rounded up if required. The last step is to compare the value in DM 0103 so that IR 011 can be incremented using the Greater Than flag.

In this example, √6017 = 77.56, and 77.56 is rounded off to 78.



Address	Instruction	Operands
00000	LD	00000
00001	@BSET(71)	
		# 0000
		DM 0100
		DM 0101
00002	@MOV(21)	
		010
		DM 0101
00003	@ROOT(72)	
		DM 0100
		DM 0102
00004	@MOV(21)	
		# 0000
		011
00005	@MOV(21)	
		# 0000
		DM 0103

Address	Instruction	Operands
00006	@MOVD(83)	
		DM 0102
		# 0012
		011
00007	@MOVD(83)	
		DM 0102
		# 0210
		DM 0103
00008	@CMP(20)	
		DM 0103
		# 4900
00009	LD	25505
00010	@INC(38)	
		011

5-20 Binary Calculations

Binary calculation instructions — ADB(50), SBB(51), MLB(52), DVB(53), ADBL(—), SBBL(—), MBS(—), MBSL(—), DBS(—), and DBSL(—) — perform arithmetic operations on hexadecimal data.

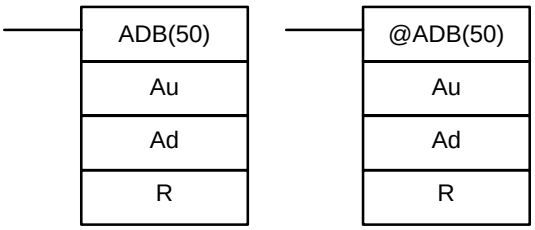
Four of these instructions (ADB(50), SBB(51), ADBL(—), and SBBL(—)) can act on both normal and signed data, two (MLB(52) and DVB(53)) act only on normal data, and four (MBS(—), MBSL(—), DBS(—), and DBSL(—)) act only on signed binary data.

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(40) and CLC(41) can be used to control CY. Refer to *5-19 BCD Calculations*.

Signed binary addition and subtraction instructions use the underflow and overflow flags (UF and OF) to indicate whether the result exceeds the acceptable range for 16-bit or 32-bit signed binary data. Refer to page 29 for details on signed binary data.

5-20-1 BINARY ADD – ADB(50)

Ladder Symbols



Operand Data Areas

Au: Augend word (binary)
IR, SR, AR, DM, HR, TC, LR, #
Ad: Addend word (binary)
IR, SR, AR, DM, HR, TC, LR, #
R: Result word
IR, SR, AR, DM, HR, LR

Description

When the execution condition is OFF, ADB(50) is not executed. When the execution condition is ON, ADB(50) 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.

ADB(50) can also be used to add signed binary data. The underflow and overflow flags (SR 25404 and SR 25405) indicate whether the result has exceeded the lower or upper limits of the 16-bit signed binary data range. Refer to page 29 for details on signed binary data.

Flags

ER: Indirectly addressed DM word is non-existent. (Content of ⁎DM word is not BCD, or the DM area boundary has been exceeded.)

CY: ON when the result is greater than FFFF.

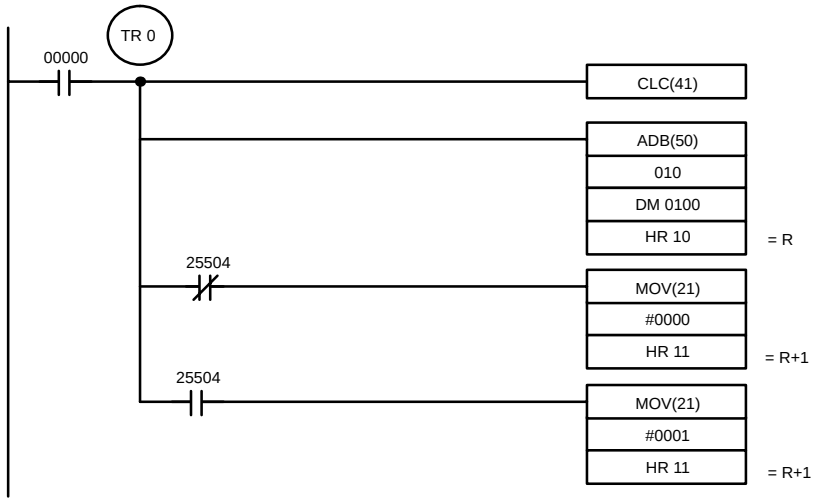
EQ: ON when the result is 0.

OF: ON when the result exceeds +32,767 (7FFF).

UF: ON when the result is below –32,768 (8000).

Example 1:
Adding Normal Data

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		00000
00001	OUT	TR	0
00002	CLC(41)		
00003	ADB(50)		
			010
		DM	0100
		HR	10
00004	AND NOT		25504
00005	MOV(21)		
		#	0000
		HR	11
00006	LD	TR	0
00007	AND		25504
00008	MOV(21)		
		#	00001
		HR	11

In the case below, A6E2 + 80C5 = 127A7. The result is a 5-digit number, so CY (SR 25504) = 1, and the content of R + 1 becomes #0001.

Au: IR 010			
A	6	E	2

\+

Ad: DM 0100			
8	0	C	5

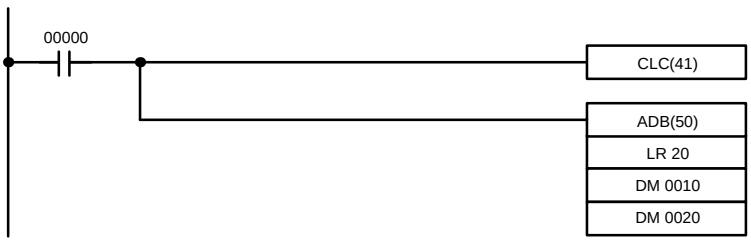
R+1: HR 11				R: HR 10			
0	0	0	1	2	7	A	7

Note The UF and OF flags would also be turned ON during this addition, but they can be ignored since they are relevant only in the addition of signed binary data.

Example 2:
Adding Signed Binary Data

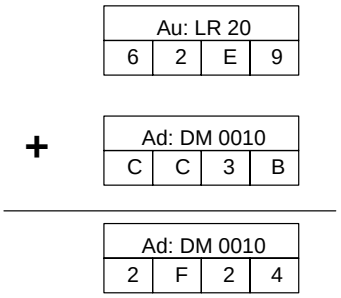
In the following example, ADB(50) is used to add two 16-bit signed binary values. (The 2's complement is used to express negative values.)

The effective range for 16-bit signed binary values is –32,767 (8000) to +32,768 (7FFF). The overflow flag (OF: SR 25404) is turned ON if the result exceeds +32,768 (7FFF) and the underflow flag (UF: SR 25405) is turned ON if the result falls below –32,767 (8000).



Address	Instruction	Operands	
00000	LD		00000
00001	CLC(41)		
00002	ADB(50)		
		LR	20
		DM	0010
		DM	0020

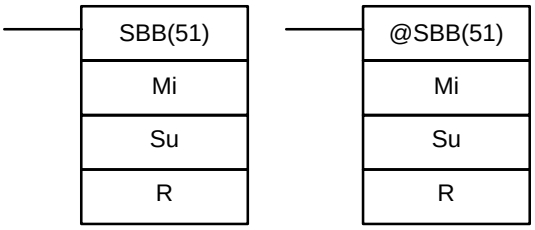
In the case below, 25,321 +(–13,253) = 12,068 (62E9 + CC3B = 2F24). Neither OF nor UF are turned ON.



Note The status of the CY flag can be ignored when adding signed binary data since it is relevant only in the addition of normal hexadecimal values.

5-20-2 BINARY SUBTRACT – SBB(51)

Ladder Symbols



Operand Data Areas

Mi: Minuend word (binary)	
IR, SR, AR, DM, HR, TC, LR, #	
Su: Subtrahend word (binary)	
IR, SR, AR, DM, HR, TC, LR, #	
R: Result word	
IR, SR, AR, DM, HR, LR	

Description

When the execution condition is OFF, SBB(51) is not executed. When the execution condition is ON, SBB(51) 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.

SBB(51) can also be used to subtract signed binary data. The overflow and underflow flags (SR 25404 and SR 25405) indicate whether the result has exceeded the lower or upper limits of the 16-bit signed binary data range. Refer to page 29 for details on signed binary data.

Flags

ER: Indirectly addressed DM word is non-existent. (Content of ⁎DM word is not BCD, or the DM area boundary has been exceeded.)

CY: ON when the result is negative, i.e., when Mi is less than Su plus CY.

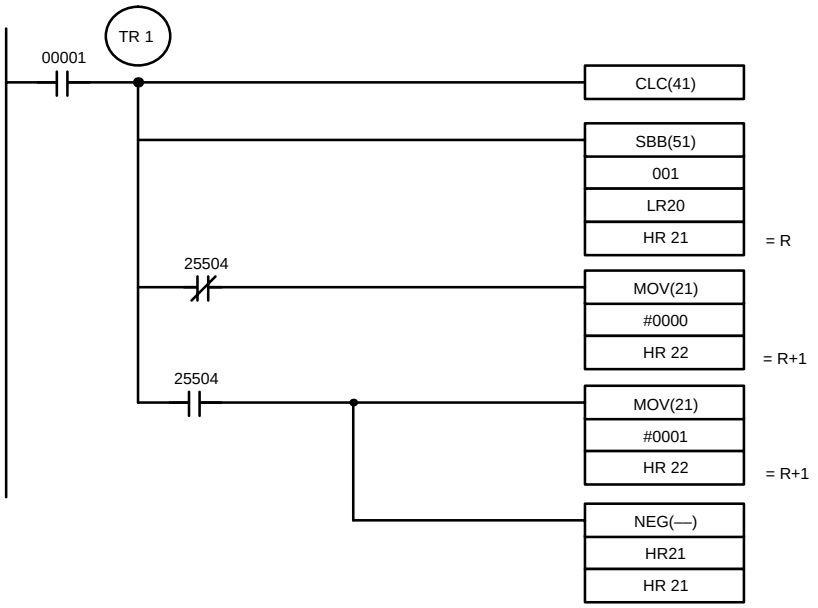
EQ: ON when the result is 0.

OF: ON when the result exceeds +32,767 (7FFF).

UF: ON when the result is below –32,768 (8000).

Example 1: Normal Data

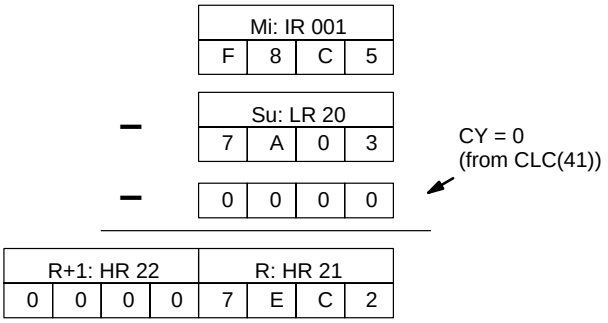
The following example shows a four-digit subtraction with CY used to place either #0000 or #0001 into R+1 to ensure that any carry is preserved.



Address	Instruction	Operands
00000	LD	00001
00001	OUT	TR 1
00002	CLC(41)	
00003	SBB(51)	
		001
		LR 20
		HR 21
00004	AND NOT	25504
00005	MOV(21)	
		# 0000
		HR 22
00006	LD	TR 1
00007	AND	25504
00008	MOV(21)	
		# 0001
		HR 22
00009	NEG(—)	
		HR 21
		HR 21

In the case below, the content of LR 20 (#7A03) and CY are subtracted from IR 001 (#F8C5). The result is stored in HR 21 and the content of HR 22 (#0000) indicates that the result is positive.

If the result had been negative, CY would have been set, #0001 would have been placed in HR 22, and the result would have been converted to its 2's compliment.

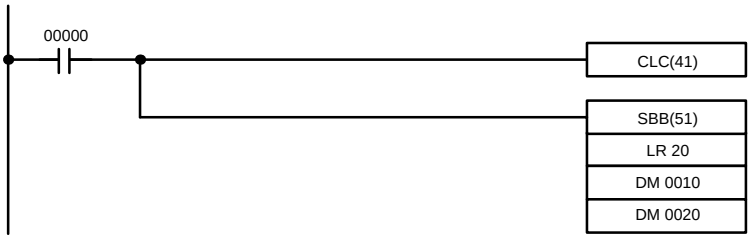


Note The status of the UF and OF flags can be ignored since they are relevant only in the subtraction of signed binary data.

Example 2: Signed Binary Data

In the following example, SBB(51) is used to subtract one 16-bit signed binary value from another. (The 2's complement is used to express negative values).

The effective range for 16-bit signed binary values is –32,768 (8000) to +32,767 (7FFF). The overflow flag (OF: SR 25404) is turned ON if the result exceeds +32,767 (7FFF) and the underflow flag (UF: SR 25405) is turned ON if the result falls below –32,768 (8000).



Address	Instruction	Operands	
00000	LD		00000
00001	CLC(41)		
00002	SBB(51)		
		LR	20
		DM	0010
		DM	0020

In the case shown below, 30,020 – (–15,238) = 45,258 (7544 – C47A = 60CA).The OF flag would be turned ON to indicate that this result exceeds the upper limit of the 16-bit signed binary data range. (In other words, the result is a positive value that exceeds 32,767 (7FFF), not a negative number expressed as signed binary data.)

	Mi: LR 20			
	7	5	4	4
	Su: DM 0010			
–	C	4	7	A
	R: DM 0020			
	B	0	C	A

In the case shown below, –30,000 – 3,000 = –33,000 (8AD0 – 0BB8 = 7F18).The UF flag would be turned ON to indicate that this result is below the lower limit of 16-bit signed binary data range. (In other words, the result is a negative number below –32,768 (8000), not a positive number expressed as signed binary data.)

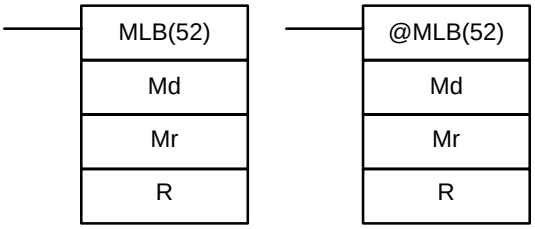
	Mi: LR 20			
	8	A	D	0
	Su: DM 0010			
–	0	B	B	8
	R: DM 0020			
	7	F	1	8

The absolute value of the true result (80E8=33,000) can be obtained by taking the 2's complement of 7F18 using NEG(—).

Note The status of the CY flag can be ignored when adding signed binary data since it is relevant only in the addition of normal hexadecimal values.

5-20-3 BINARY MULTIPLY – MLB(52)

Ladder Symbols



Operand Data Areas

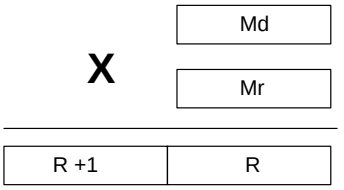
Md: Multiplicand word (binary)
IR, SR, AR, DM, HR, TC, LR, #
Mr: Multiplier word (binary)
IR, SR, AR, DM, HR, TC, LR, #
R: First result word
IR, SR, AR, DM, HR LR

Limitations

R and R+1 must be in the same data area.

Description

When the execution condition is OFF, MLB(52) is not executed. When the execution condition is ON, MLB(52) multiplies the content of Md by the contents of Mr, places the rightmost four digits of the result in R, and places the leftmost four digits in R+1.



Precautions

MLB(52) cannot be used to multiply signed binary data. Use MBS(—) instead. Refer to *5-20-7 SIGNED BINARY MULTIPLY – MBS(—)* for details.

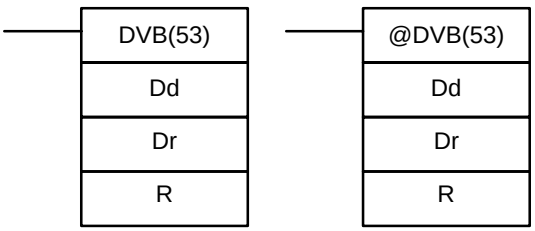
Flags

ER: Indirectly addressed DM word is non-existent. (Content of ⁎DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON when the result is 0.

5-20-4 BINARY DIVIDE – DVB(53)

Ladder Symbols

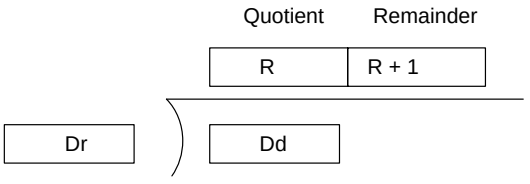


Operand Data Areas

Dd: Dividend word (binary)
IR, SR, AR, DM, HR, TC, LR, #
Dr: Divisor word (binary)
IR, SR, AR, DM, HR, TC, LR, #
R: First result word
IR, SR, AR, DM, HR LR

Description

When the execution condition is OFF, DVB(53) is not executed. When the execution condition is ON, DVB(53) 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.



Precautions

DVB(53) cannot be used to divide signed binary data. Use DBS(—) instead. Refer to *5-20-9 SIGNED BINARY DIVIDE – DBS(—)* for details.

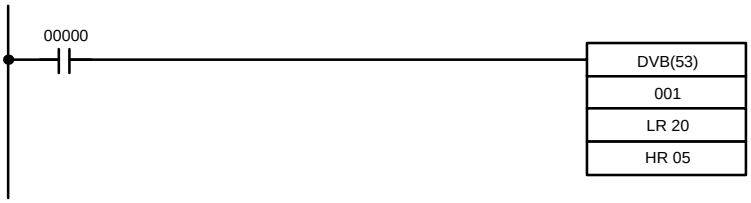
Flags

ER: Dr contains 0.

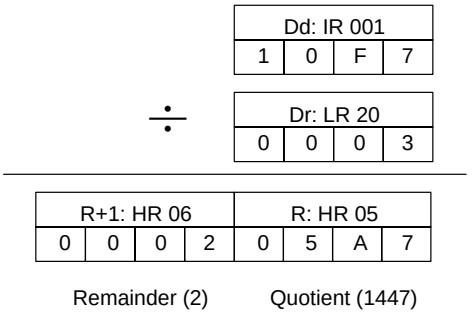
Indirectly addressed DM word is non-existent. (Content of ⁜DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON when the result is 0.

Example

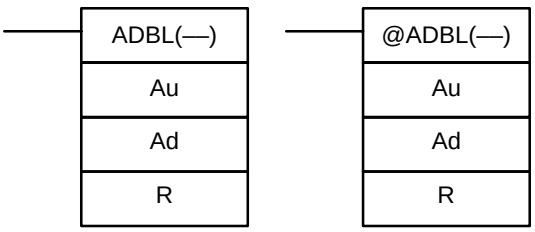


Address	Instruction	Operands	
00000	LD		00000
00001	DVB(53)		
			001
		LR	0020
		HR	05



5-20-5 DOUBLE BINARY ADD – ADBL(––)

Ladder Symbols



Operand Data Areas

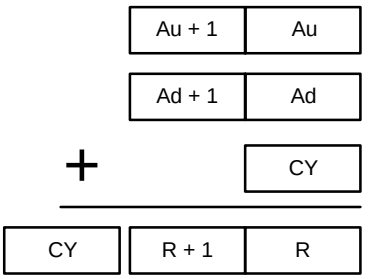
Au: First augend word (binary)
IR, SR, AR, DM, HR, LR
Ad: First addend word (binary)
IR, SR, AR, DM, HR, LR
R: First result word
IR, SR, AR, DM, HR, LR

Limitations

Au and Au+1 must be in the same data area, as must Ad and Ad+1, and R and R+1.

Description

When the execution condition is OFF, ADBL(—) is not executed. When the execution condition is ON, ADBL(—)) adds the eight-digit contents of Au+1 and Au, the eight-digit contents 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.



ADBL(—) can also be used to add signed binary data. The underflow and overflow flags (SR 25404 and SR 25405) indicate whether the result has exceeded the lower or upper limits of the 32-bit signed binary data range. Refer to page 29 for details on signed binary data.

Flags

ER: Indirectly addressed DM word is non-existent. (Content of ✳DM word is not BCD, or the DM area boundary has been exceeded.)

CY: ON when the result is greater than FFFF FFFF.

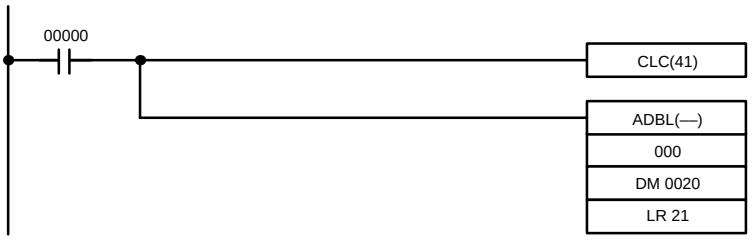
EQ: ON when the result is 0.

OF: ON when the result exceeds +2,147,483,647 (7FFF FFFF).

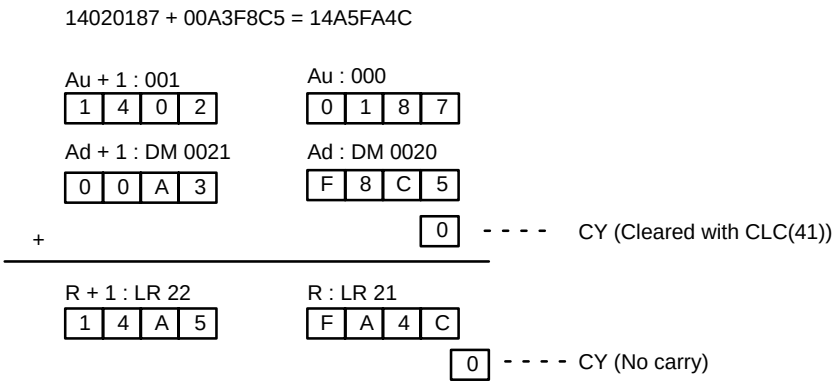
UF: ON when the result is below –2,147,483,648 (8000 0000).

Example 1: Normal Data

The following example shows an eight-digit addition with CY (SR 25504) used to represent the status of the 9th digit.



Address	Instruction	Operands	
00000	LD		00000
00001	CLC(41)		
00002	ADBL(—)		
			000
		DM	0020
		LR	21

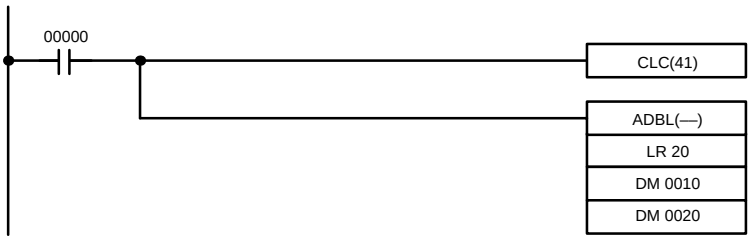


Note The status of the UF and OF flags can be ignored since they are relevant only in the addition of signed binary data.

Example 2: Signed Binary Data

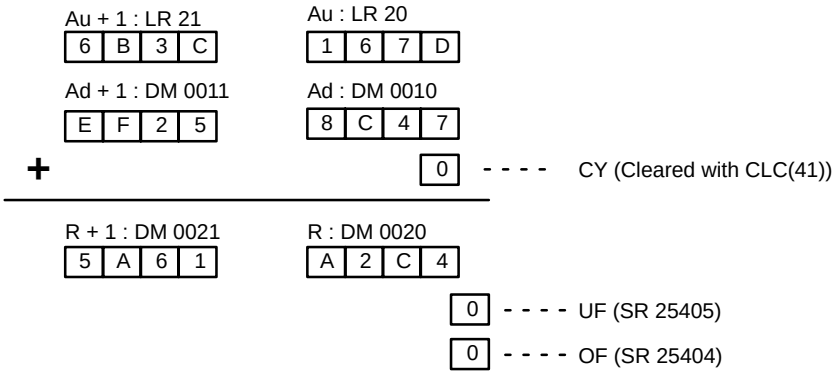
In the following example, ADBL(—) is used to add two 32-bit signed binary values and output the 32-bit signed binary result to R and R+1. (The 2's complement is used to express negative values).

The effective range for 32-bit signed binary values is –2,147,483,648 (8000 0000) to +2,147,483,647 (7FFF FFFF). The overflow flag (OF: SR 25404) is turned ON if the result exceeds +2,147,483,647 (7FFF FFFF) and the underflow flag (UF: SR 25405) is turned ON if the result falls below –2,147,483,648 (8000 0000).



Address	Instruction	Operands	
00000	LD		00000
00001	CLC(41)		
00002	ADBL(—)		
		LR	20
		DM	0010
		DM	0020

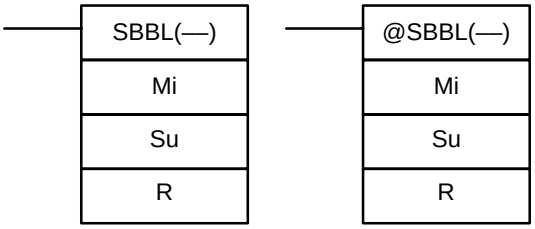
In the case below, 1,799,100,099 + (–282,751,929) = 1,516,348,100 (6B3C167D + EF258C47 = 5A61A2C4). Neither OF nor UF are turned ON.



Note The status of the CY flag can be ignored when adding signed binary data since it is relevant only in the addition of normal hexadecimal values.

5-20-6 DOUBLE BINARY SUBTRACT – SBBL(––)

Ladder Symbols



Operand Data Areas

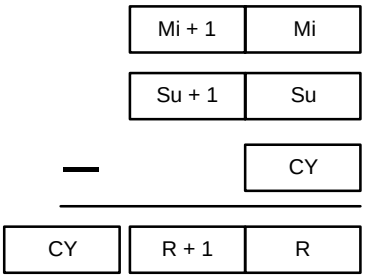
Mi: First minuend word (binary)
IR, SR, AR, DM, HR, TC, LR
Su: First subtrahend word (binary)
IR, SR, AR, DM, HR, TC, LR
R: First result word
IR, SR, AR, DM, HR, LR

Limitations

Mi and Mi+1 must be in the same data area, as must Su and Su+1, and R and R+1.

Description

When the execution condition is OFF, SBBL(—) is not executed. When the execution condition is ON, SBBL(—) subtracts CY and the eight-digit value in Su and Su+1 from the eight-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+1 and R. Use the DOUBLE 2's COMPLEMENT instructions to convert the 2's complement to the true result.



SBBL(—) can also be used to subtract signed binary data. The underflow and overflow flags (SR 25404 and SR 25405) indicate whether the result has exceeded the lower or upper limits of the 32-bit signed binary data range. Refer to page 29 for details on signed binary data.

Flags

ER: Indirectly addressed DM word is non-existent. (Content of ✳DM word is not BCD, or the DM area boundary has been exceeded.)

CY: ON when the result is negative, i.e., when Mi is less than Su plus CY.

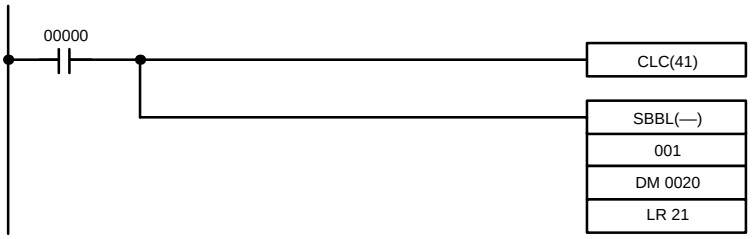
EQ: ON when the result is 0.

OF: ON when the result exceeds +2,147,483,647 (7FFF FFFF).

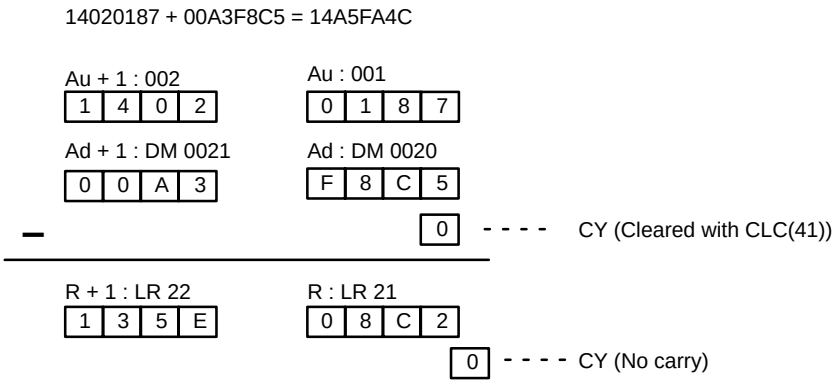
UF: ON when the result is below –2,147,483,648 (8000 0000).

Example 1: Normal Data

In this example, the eight-digit number in IR 002 and IR 001 is subtracted from the eight-digit number in DM 0021 and DM 0020, and the result is output to LR 22 and LR 21. If the result is negative, CY (SR 25504) is turned ON.



Address	Instruction	Operands	
00000	LD		00000
00001	CLC(41)		
00002	SBBL(—)		
			001
		DM	0020
		LR	21

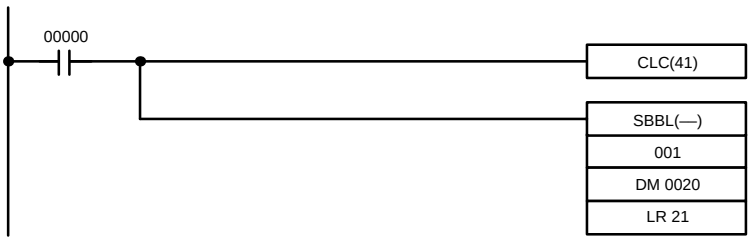


Note The status of the UF and OF flags can be ignored since they are relevant only in the subtraction of signed binary data.

Example 2: Signed Binary Data

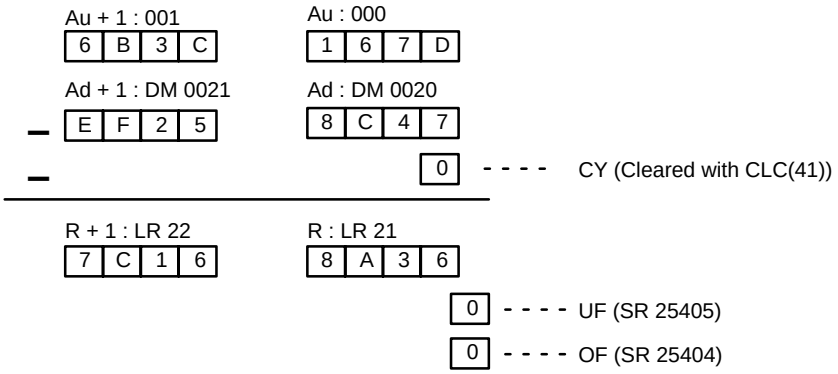
In the following example, SBBL(—) is used to subtract one 32-bit signed binary value from another and output the 32-bit signed binary result to R and R+1.

The effective range for 32-bit signed binary values is –2,147,483,648 (8000 0000) to +2,147,483,647 (7FFF FFFF). The overflow flag (OF: SR 25404) is turned ON if the result exceeds +2,147,483,647 (7FFF FFFF) and the underflow flag (UF: SR 25405) is turned ON is the result falls below –2,147,483,648 (8000 0000).



Address	Instruction	Operands	
00000	LD		00000
00001	CLC(41)		
00002	SBBL(—)		
			001
		DM	0020
		LR	21

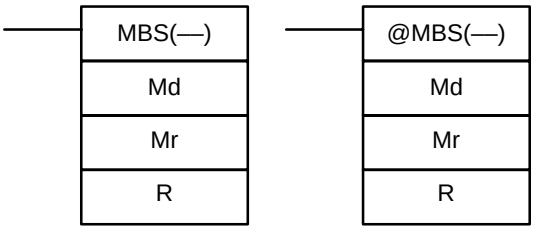
In the case below, 1,799,100,099 – (–282,751,929) = 2,081,851,958 (6B3C 167D – {EF25 8C47 – 1 0000 0000} = 7C16 8A36). Neither OF nor UF are turned ON.



Note The status of the CY flag can be ignored when adding signed binary data since it is relevant only in the addition of normal hexadecimal values.

5-20-7 SIGNED BINARY MULTIPLY – MBS(––)

Ladder Symbols

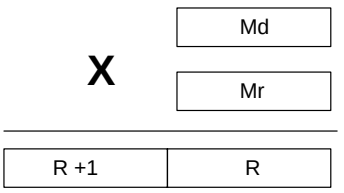


Operand Data Areas

Md: Multiplicand word
IR, SR, AR, DM, HR, TC, LR, #
Mr: Multiplier word
IR, SR, AR, DM, HR, TC, LR, #
R: First result word
IR, SR, AR, DM, HR LR

Limitations R and R+1 must be in the same data area.

Description MBS(—) multiplies the signed binary content of two words and outputs the 8-digit signed binary result to R+1 and R. The rightmost four digits of the result are placed in R, and the leftmost four digits are placed in R+1. Refer to page 29 for details on signed binary data.



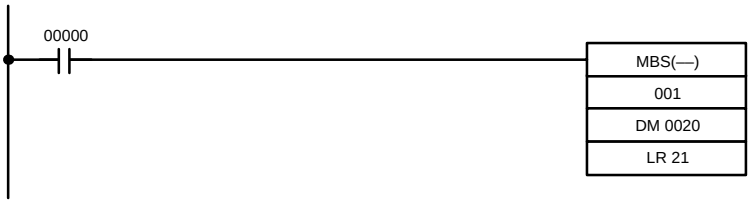
Flags

ER: Indirectly addressed DM word is non-existent. (Content of ⁜DM word is not BCD, or the DM area boundary has been exceeded.)

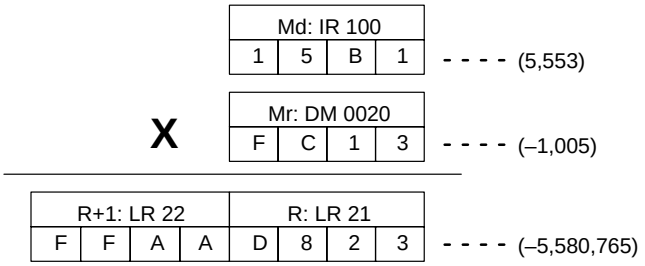
EQ: ON when the result is 0000 0000, otherwise OFF.

Example

In the following example, MBS(—) is used to multiply the signed binary contents of IR 001 with the signed binary contents of DM 0020 and output the result to LR 21 and LR 22.

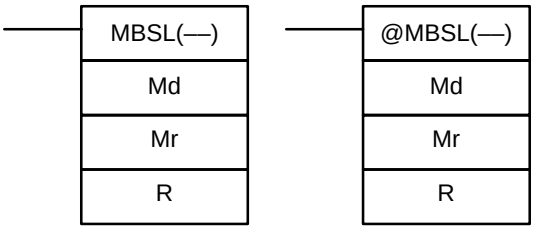


Address	Instruction	Operands
00000	LD	00000
00001	MBS(—)	
		001
		DM 0020
		LR 21



5-20-8 DOUBLE SIGNED BINARY MULTIPLY – MBSL(––)

Ladder Symbols



Operand Data Areas

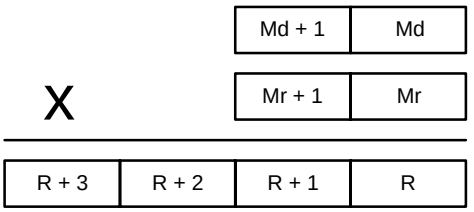
Md: First multiplicand word
IR, SR, AR, DM, HR, TC, LR
Mr: First multiplier word
IR, SR, AR, DM, HR, TC, LR
R: First result word
IR, SR, AR, DM, HR LR

Limitations

Md and Md+1 must be in the same data area, as must Mr and Mr+1, and R and R+3.

Description

MBSL(—) multiplies the 32-bit (8-digit) signed binary data in Md+1 and Md with the 32-bit signed binary data in Mr+1 and Mr, and outputs the 16-digit signed binary result to R+3 through R. Refer to page 29 for details on signed binary data.



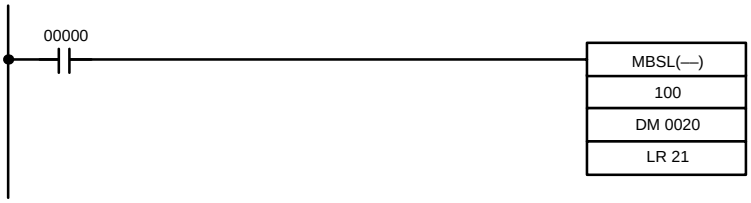
Flags

ER: Indirectly addressed DM word is non-existent. (Content of ⁎DM word is not BCD, or the DM area boundary has been exceeded.)

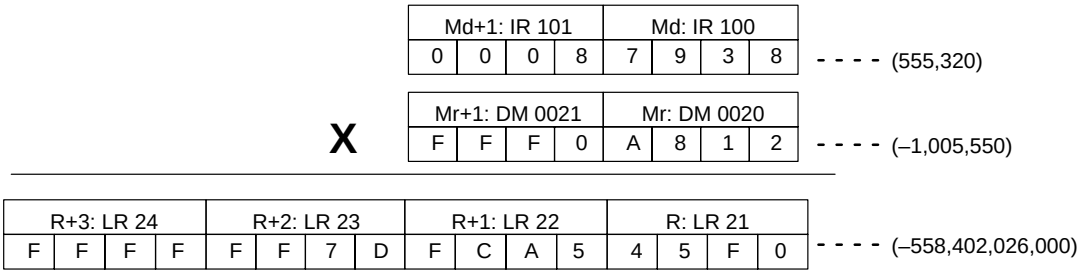
EQ: ON when the result is zero (content of R+3 through R all zeroes), otherwise OFF.

Example

In the following example, MBSL(—) is used to multiply the signed binary contents of IR 101 and IR 100 with the signed binary contents of DM 0021 and DM 0020 and output the result to LR 24 through LR 21.

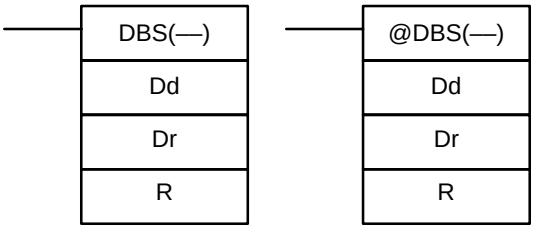


Address	Instruction	Operands
00000	LD	00000
00001	MBSL(—)	
		100
		DM 0020
		LR 21



5-20-9 SIGNED BINARY DIVIDE – DBS(––)

Ladder Symbols



Operand Data Areas

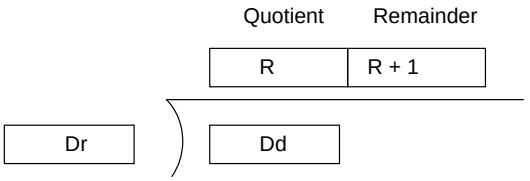
Dd: Dividend word
IR, SR, AR, DM, HR, TC, LR, #
Dr: Divisor word
IR, SR, AR, DM, HR, TC, LR, #
R: First result word
IR, SR, AR, DM, HR LR

Limitations

R and R+1 must be in the same data area.

Description

DBS(—) divides the signed binary content of Dd by the signed binary content of Dr, and outputs the 8-digit signed binary result to R+1 and R. The quotient is placed in R, and the remainder is placed in R+1. Refer to page 29 for details on signed binary data.



Flags

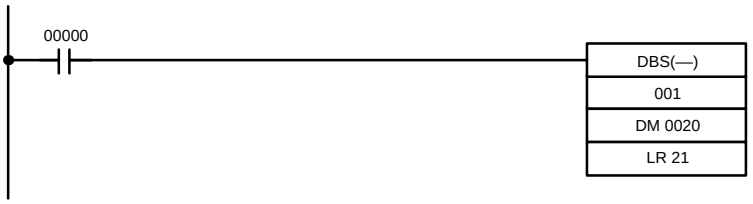
ER: Dr contains 0.

Indirectly addressed DM word is non-existent. (Content of ⁎DM word is not BCD, or the DM area boundary has been exceeded.)

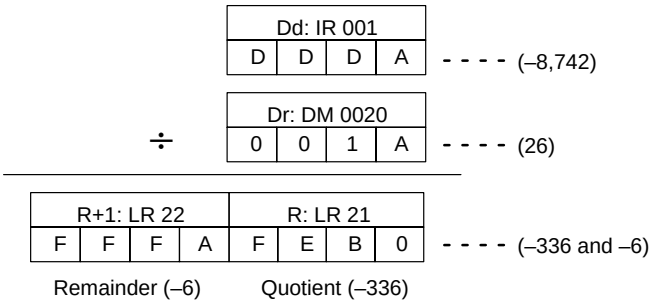
EQ: ON when the content of R (the quotient) is 0000, otherwise OFF.

Example

In the following example, DBS(—) is used to divide the signed binary contents of IR 001 with the signed binary contents of DM 0020 and output the result to LR 21 and LR 22.

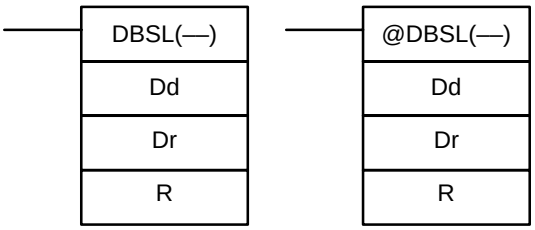


Address	Instruction	Operands
00000	LD	00000
00001	DBS(—)	
		001
		DM 0020
		LR 21



5-20-10 DOUBLE SIGNED BINARY DIVIDE – DBSL(––)

Ladder Symbols



Operand Data Areas

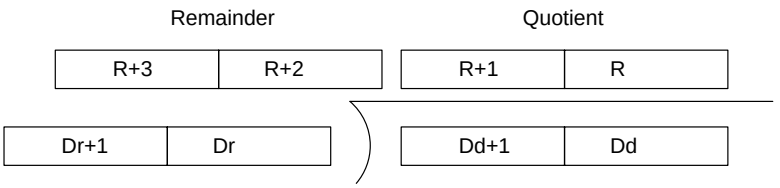
Dd: Dividend word (binary)	IR, SR, AR, DM, HR, TC, LR, #
Dr: Divisor word (binary)	IR, SR, AR, DM, HR, TC, LR, #
R: First result word	IR, SR, AR, DM, HR LR

Limitations

Dd and Dd+1 must be in the same data area, as must Dr and Dr+1, and R and R+3.

Description

DBS(—) divides the 32-bit (8-digit) signed binary data in Dd+1 and Dd by the 32-bit signed binary data in Dr+1 and Dr, and outputs the 16-digit signed binary result to R+3 through R. The quotient is placed in R+1 and R, and the remainder is placed in R+3 and R+2. Refer to page 29 for details on signed binary data.



Flags

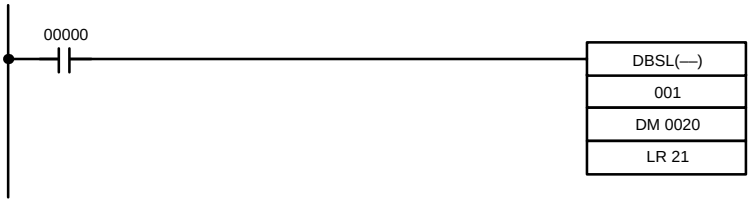
ER: Dr+1 and Dr contain 0.

Indirectly addressed DM word is non-existent. (Content of ⁎DM word is not BCD, or the DM area boundary has been exceeded.)

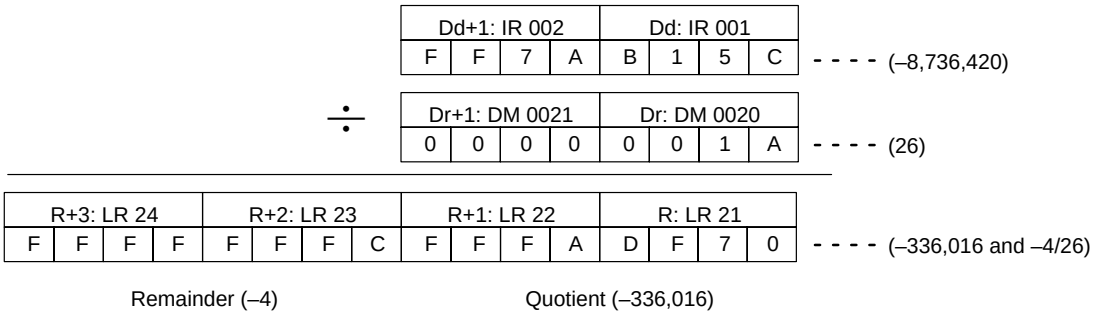
EQ: ON when the content of R+1 and R (the quotient) is 0, otherwise OFF.

Example

In the following example, DBSL(—) is used to divide the signed binary contents of IR 002 and IR 001 with the signed binary contents of DM 0021 and DM 0020 and output the result to LR 24 through LR 21.



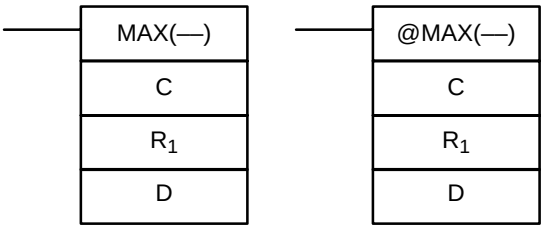
Address	Instruction	Operands
00000	LD	00000
00001	DBSL(—)	
		001
		DM 0020
		LR 21



5-21 Special Math Instructions

5-21-1 FIND MAXIMUM – MAX(––)

Ladder Symbols



Operand Data Areas

C: Control data
IR, SR, AR, DM, HR, LR
R_1: First word in range
IR, SR, AR, DM, HR, TC, LR
D: Destination word
IR, SR, AR, DM, HR, LR

Limitations

N in C must be BCD between 001 to 999.

R_1 and R_1+N–1 must be in the same data area.

Description

When the execution condition is OFF, MAX(—) is not executed. When the execution condition is ON, MAX(—) searches the range of memory from R_1 to R_1+N–1 for the address that contains the maximum value and outputs the maximum value to the destination word (D).

If bit 14 of C is ON, MAX(—) identifies the address of the word containing the maximum value in D+1. The address is identified differently for the DM area:

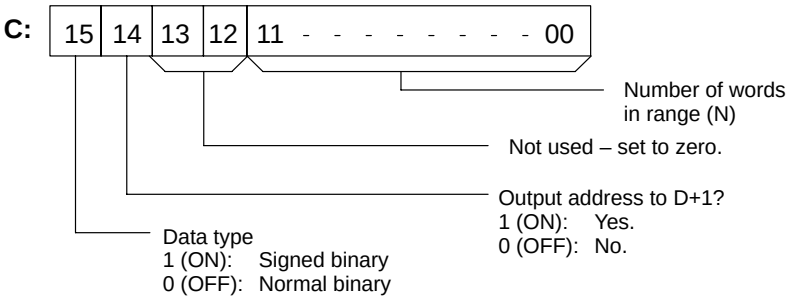
1, 2, 3...

1. For an address in the DM area, the word address is written to D+1. For example, if the address containing the maximum value is DM 0114, then #0114 is written in D+1.
2. For an address in another data area, the number of addresses from the beginning of the search is written to D+1. For example, if the address containing the maximum value is IR 114 and the first word in the search range is IR 014, then #0100 is written in D+1.

If bit 15 of C is ON and more than one address contains the same maximum value, the position of the lowest of the addresses will be output to 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 normal binary and when it is ON the data is treated as signed binary.



Caution If bit 14 of C is ON, values above #8000 are treated as negative numbers, so the results will differ depending on the specified data type. Be sure that the correct data type is specified.

Flags

ER: Indirectly addressed DM word is non-existent. (Content of ⁜DM word is not BCD, or the DM area boundary has been exceeded.)

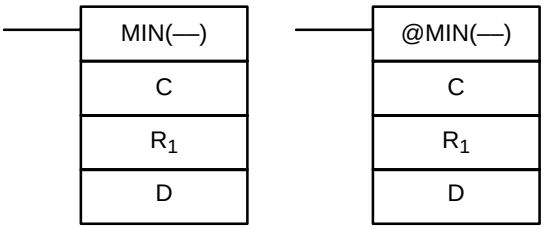
The number of words specified in C is not BCD (000 to 999).

R_1 and R_1+N–1 are not in the same data area.

EQ: ON when the maximum value is #0000.

5-21-2 FIND MINIMUM – MIN(––)

Ladder Symbols



Operand Data Areas

C: Control data
IR, SR, AR, DM, HR, TC, LR
R_1: First word in range
IR, SR, AR, DM, HR, TC, LR
D: Destination word
IR, SR, AR, DM, HR, LR

Limitations

N in C must be BCD between 001 to 999.

R_1 and R_1+N–1 must be in the same data area.

Description

When the execution condition is OFF, MIN(—) is not executed. When the execution condition is ON, MIN(—) searches the range of memory from R_1 to R_1+N–1 for the address that contains the minimum value and outputs the minimum value to the destination word (D).

If bit 14 of C is ON, MIN(—) identifies the address of the word containing the minimum value in D+1. The address is identified differently for the DM area:

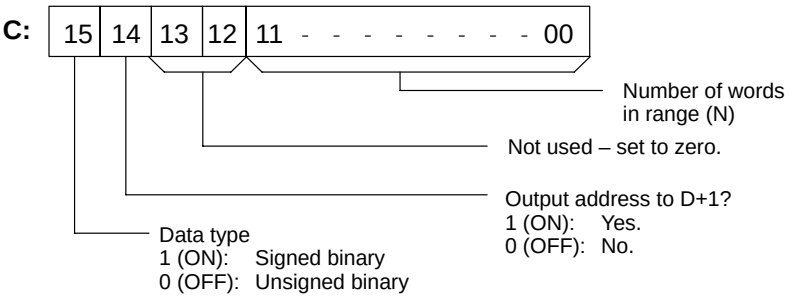
1, 2, 3...

1. For an address in the DM area, the word address is written to D+1. For example, if the address containing the minimum value is DM 0114, then #0114 is written in D+1.
2. For an address in another data area, the number of addresses from the beginning of the search is written to D+1. For example, if the address containing the minimum value is IR 114 and the first word in the search range is IR 014, then #0100 is written in D+1.

If bit 14 of C is ON and more than one address contains the same minimum value, the position of the lowest of the addresses will be output to 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 page 29 for details on signed binary data.



⚠ Caution If bit 14 of C is ON, values above #8000 are treated as negative numbers, so the results will differ depending on the specified data type. Be sure that the correct data type is specified.

Flags

ER: Indirectly addressed DM word is non-existent. (Content of ⁜DM word is not BCD, or the DM area boundary has been exceeded.)

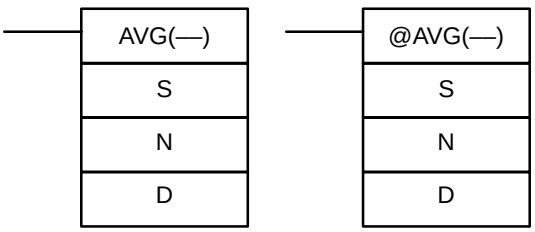
The number of words specified in C is not BCD (000 to 999).

R_1 and R_1+N–1 are not in the same data area.

EQ: ON when the minimum value is #0000.

5-21-3 AVERAGE VALUE – AVG(––)

Ladder Symbols



Operand Data Areas

S: Source word
IR, SR, AR, DM, HR, TC, LR

N: Number of cycles
IR, SR, AR, DM, HR, TC, LR, #

D: First destination word
IR, SR, AR, DM, HR, LR

Limitations

Data of S must be hexadecimal.

N must be BCD from #0001 to #0064.

D and D+N+1 must be in the same data area.

Description

AVG(—) is used to calculate the average value of S over N cycles.

When the execution condition is OFF, AVG(—) is not executed.

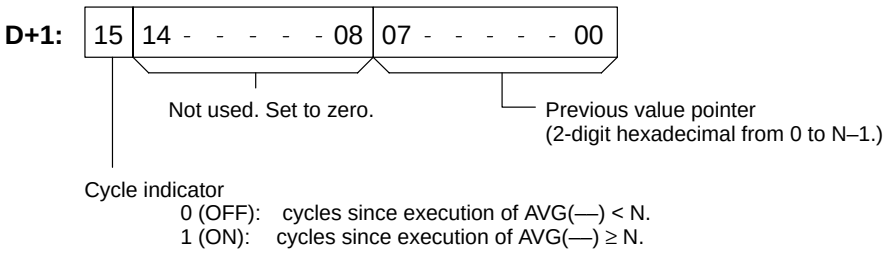
For the first N–1 cycles when the execution condition is ON, AVG(—) writes the value of S to D. Each time that AVG(—) is executed, the previous value of S is stored in words D+2 to D+N+1. The first 2 digits of D+1 are incremented with each execution and act as a pointer to indicate where the previous value is stored. Bit 15 of D+1 remains OFF for the first N–1 cycles.

On the N^{th} cycle, the previous value of S is written to last word in the range D+2 to D+N+1. The average value of the previous values stored in D+2 to D+N+1 is calculated and written to D, bit 15 of D+1 is turned ON, and the previous value pointer (the first 2 digits of D+1) is reset to zero. Each time that AVG(—) is executed, the previous value of S overwrites the content of the word indicated by the pointer and the new average value is calculated and written to D. The pointer will be reset again after reaching N–1.

The following diagram shows the function of words D to D+N+1.

D	Average value (after N or more cycles)
D+1	Previous value pointer and cycle indicator
D+2	Previous value #1
D+3	Previous value #2
⋮	⋮ ⋮ ⋮ ⋮
D+N+1	Previous value #N

The function of bits in D+1 are shown in the following diagram and explained in more detail below.



Previous Value Pointer

The previous value pointer indicates the location where the most recent value of S was stored relative to D+2, i.e., a pointer value of 0 indicates D+2, a value of 1 indicates D+3, etc.

Cycle Indicator

The cycle indicator is turned ON after AVG(—) has been executed N times. At this point, D will contain the average value of the contents of words D+2 through D+N+1. The average value is 4-digit hexadecimal and is rounded off to the nearest integer value.

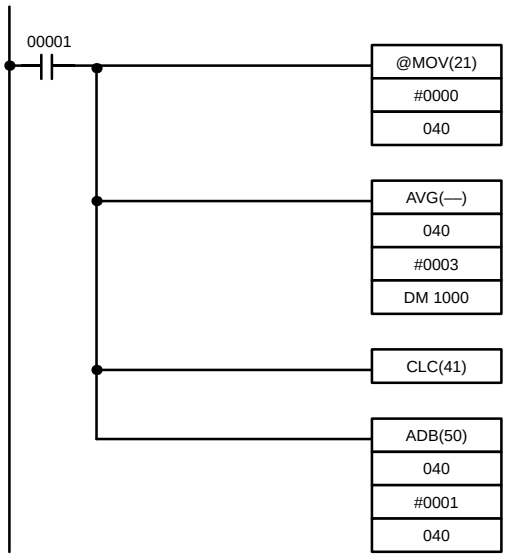
Flags

ER: Indirectly addressed DM word is non-existent. (Content of ✳DM word is not BCD, or the DM area boundary has been exceeded.)

One or more operands have been set incorrectly.

Example

In the following example, the content of IR 040 is set to #0000 and then incremented by 1 each cycle. For the first two cycles, AVG(—) moves the content of IR 040 to DM 1002 and DM 1003. The contents of DM 1001 will also change (which can be used to confirm that the results of AVG(—) has changed). On the third and later cycles AVG(—) calculates the average value of the contents of DM 1002 to DM 1004 and writes that average value to DM 1000.



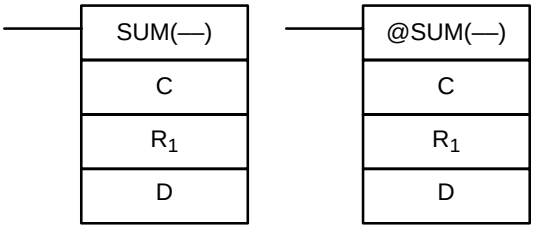
Address	Instruction	Operands
00000	LD	00001
00001	@MOV(21)	
		# 0000
		040
00002	AVG(—)	
		040
		# 0003
		DM 1000
00003	CLC(41)	
00004	ADB(50)	
		040
		# 0001
		040

	1st cycle	2nd cycle	3rd cycle	4th cycle
IR 40	0000	0001	0002	0003

	1st cycle	2nd cycle	3rd cycle	4th cycle	
DM 1000	0000	0001	0001	0002	Average
DM 1001	0001	0002	8000	8000	Pointer
DM 1002	0000	0000	0000	0003	Previous
DM 1003	---	0001	0001	0001	values of
DM 1004	---	---	0002	0002	IR 40

5-21-4 SUM – SUM(––)

Ladder Symbols



Operand Data Areas

C: Control data
IR, SR, AR, DM, HR, LR

R_1: First word in range
IR, SR, AR, DM, HR, TC, LR

D: First destination word
IR, SR, AR, DM, HR, LR

Limitations

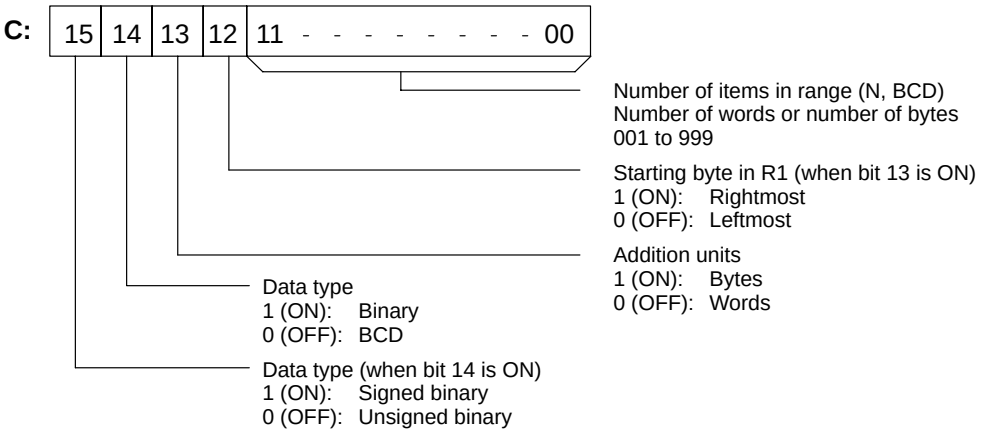
The 3 rightmost digits of C must be BCD between 001 and 999.

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.

Description

When the execution condition is OFF, SUM(—) is not executed. When the execution condition is ON, SUM(—) adds either the contents of words R_1 to R_1+N–1 or the bytes in words R_1 to R_1+N/2–1 and outputs that value to the destination words (D and D+1). The data can be summed as binary or BCD and will be output in the same form. Binary data can be either signed or unsigned.

The function of bits in C are shown in the following diagram and explained in more detail below.



Number of Items in Range

The number of items within the range (N) is contained in the 3 rightmost digits of C, which must be BCD between 001 and 999. This number will indicate the number of words or the number of bytes depending the items being summed.

Addition Units

Words will be added if bit 13 is OFF and bytes will be added if bit 13 is ON.

If bytes are specified, the range can begin with the leftmost or rightmost byte of R_1. The leftmost byte of R_1 will not be added if bit 12 is ON.

	MSB	LSB
R_1	1	2
R_1+1	3	4
R_1+2	5	6
R_1+3	7	8
⋮	⋮	⋮

The bytes will be added in this order when bit 12 is OFF: 1+2+3+4....

The bytes will be added in this order when bit 12 is ON: 2+3+4....

Data Type

Data within the range is treated as unsigned binary when bit 14 of C is ON and bit 15 is OFF, and it is treated as signed binary when both bits 14 and 15 are ON. Refer to page 29 for details on signed binary data.

Data within the range is treated as BCD when bit 14 of C is OFF, regardless of the status of bit 15.

Flags

ER: Indirectly addressed DM word is non-existent. (Content of ✳DM word is not BCD, or the DM area boundary has been exceeded.)

R_1 and R_1+N–1 are not in the same data area.

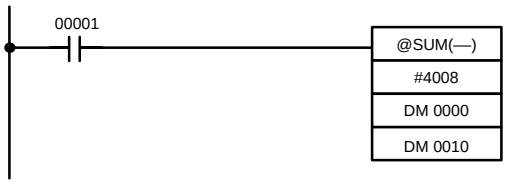
The number of items in C is not BCD between 001 and 999.

The data being summed is not BCD when BCD was designated.

EQ: ON when the result is zero.

Example

In the following example, the BCD contents of the 8 words from DM 0000 to DM 0007 are added when IR 00001 is ON and the result is written to DM 0010 and DM 0011.



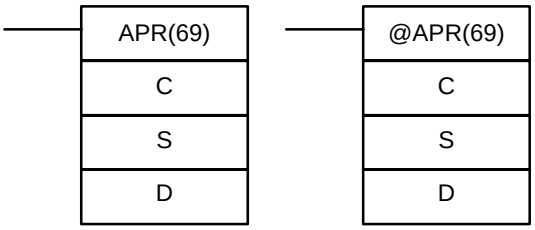
Address	Instruction	Operands
00000	LD	00001
00001	@SUM(—)	
		# 4008
		DM 0000
		DM 0010

DM 0000	0001
DM 0001	0002
DM 0002	0003
DM 0003	0004
DM 0004	0005
DM 0005	0006
DM 0006	0007
DM 0007	0008

DM 0010	0036
DM 0011	0000

5-21-5 ARITHMETIC PROCESS – APR(69)

Ladder Symbols



Operand Data Areas

C: Control word
IR, SR, AR, DM, HR, TC, LR, #
S: Input data source word
IR, SR, AR, DM, HR, TC, LR
D: Result destination word
IR, SR, AR, DM, HR,TC, LR

Limitations

For trigonometric functions S must be BCD from 0000 to 0900 ($0° \leq \theta \leq 90°$).

Description

When the execution condition is OFF, APR(69) is not executed. When the execution condition is ON, the operation of APR(69) depends on the control word C.

If C is #0000 or #0001, APR(69) computes sin(θ) or cos(θ)*. The BCD value of S specifies θ in tenths of degrees.

If C is an address, APR(69) computes f(x) of the function entered in advance beginning at word C. The function is a series of line segments (which can approximate a curve) determined by the operator. The BCD or hexadecimal value of S specifies x.

Flags

ER: Indirectly addressed DM word is non-existent. (Content of ⁜DM word is not BCD, or the DM area boundary has been exceeded.)

For trigonometric functions, x > 0900. (x is the content of S.)

A constant other than #0000 or #0001 was designated for C.

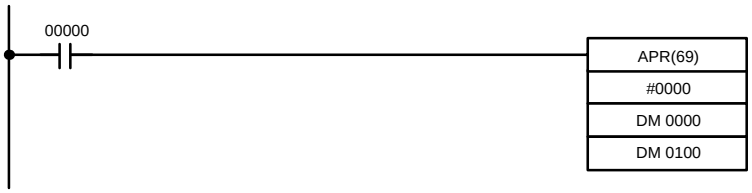
The linear approximation data is not readable.

EQ: The result is 0000.

Examples

Sine Function

The following example demonstrates the use of the APR(69) sine function to calculate the sine of 30°. The sine function is specified when C is #0000.



Address	Instruction	Operands
00000	LD	00000
00001	APR(69)	
		# 0000
		DM 0000
		DM 0100

Input data, x

S: DM 0000			
0	10^1	10^0	10^{-1}
0	3	0	0

Enter input data not exceeding #0900 in BCD.

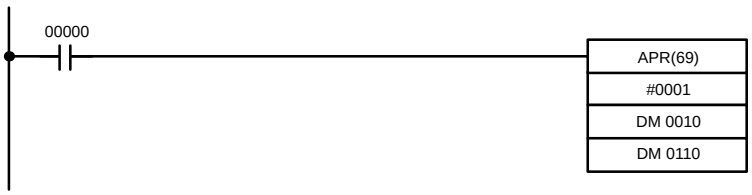
Result data

D: DM 0100			
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 demonstrates the use of the APR(69) cosine function to calculate the cosine of 30°. The cosine function is specified when C is #0001.



Address	Instruction	Operands
00000	LD	00000
00001	APR(69)	
		# 0001
		DM 0010
		DM 0110

Input data, x

S: DM 0010			
0	10^1	10^0	10^{-1}
0	3	0	0

Enter input data not exceeding #0900 in BCD.

Result data

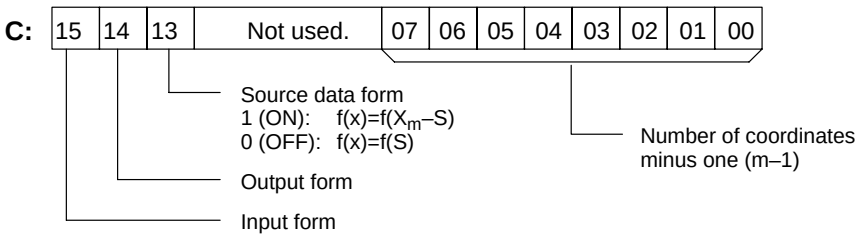
D: DM 0110			
10^{-1}	10^{-2}	10^{-3}	10^{-4}
8	6	6	0

Result data has four significant digits, fifth and higher digits are ignored. The result for cos(0) will be 0.9999, not 1.

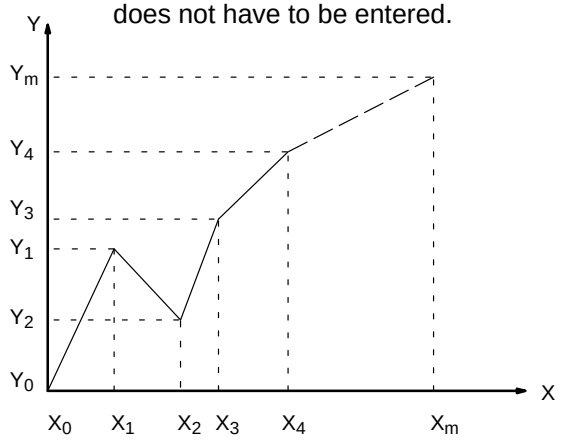
Linear Approximation

APR(69) linear approximation is specified when C is a memory address. Word C is the first word of the continuous block of memory containing the linear approximation data.

The content of word C specifies the number of line segments in the approximation, and whether the input and output are in BCD or BIN form. Bits 00 to 07 contain the number of line segments less 1, m–1, as binary data. Bits 14 and 15 determine, respectively, the output and input forms: 0 specifies BCD and 1 specifies BIN.

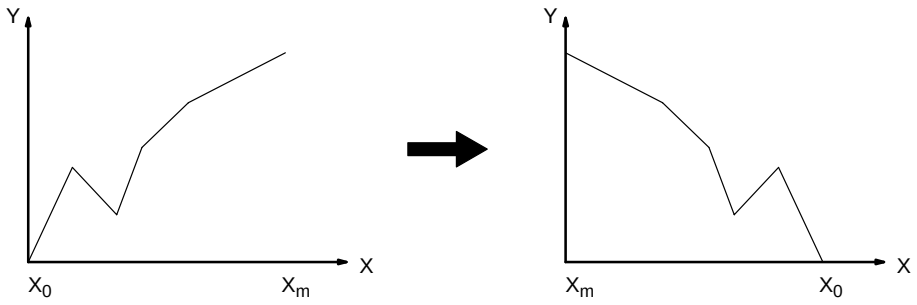


Enter the coordinates of the m+1 end-points, which define the m line segments, as shown in the following table. Enter all coordinates in BIN form. Always enter the coordinates from the lowest X value (X_1) to the highest (X_m). X_0 is 0000, and does not have to be entered.

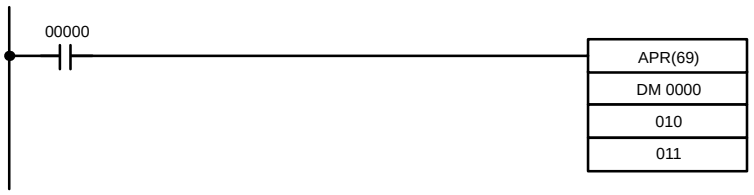


Word	Coordinate
C+1	X_m (max. X value)
C+2	Y_0
C+3	X_1
C+4	Y_1
C+5	X_2
C+6	Y_2
↓	↓
C+(2m+1)	X_m
C+(2m+2)	Y_m

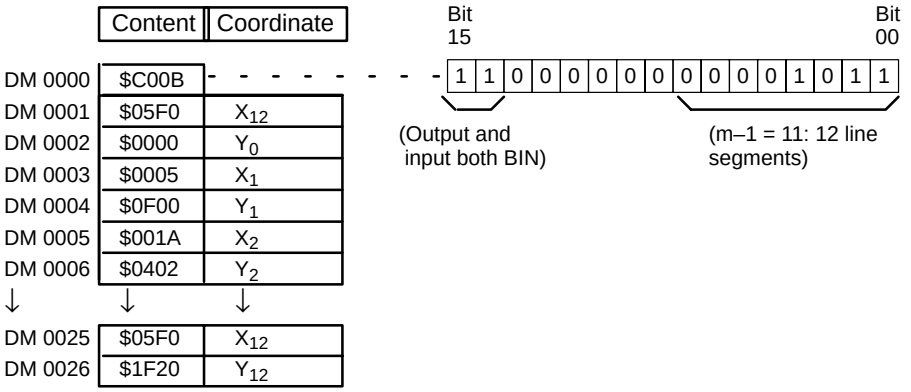
If bit 13 of C is set to 1, the graph will be reflected from left to right, as shown in the following diagram.



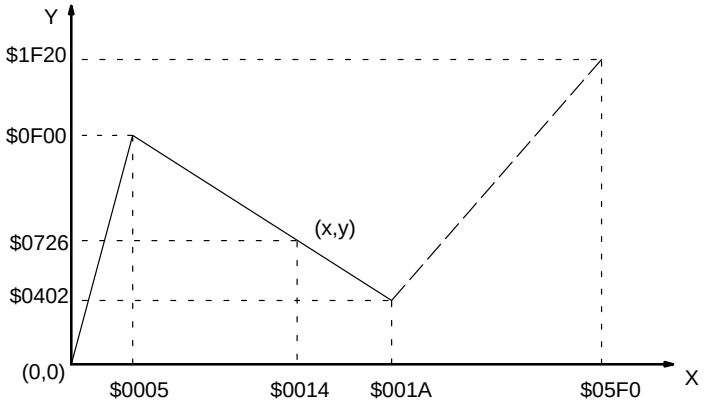
The following example demonstrates the construction of a linear approximation with 12 line segments. The block of data is continuous, as it must be, from DM 0000 to DM 0026 (C to C + (2 × 12 + 2)). The input data is taken from IR 010, and the result is output to IR 011.



Address	Instruction	Operands
00000	LD	00000
00001	APR(69)	
		DM 0000
		010
		011

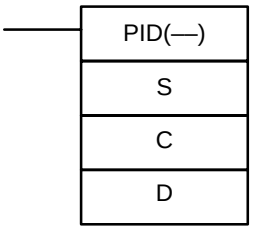


In this case, the input data word, IR 010, contains #0014, and f(0014) = #0726 is output to R, IR 011.



5-21-6 PID CONTROL – PID(––)

Ladder Symbol



Operand Data Areas

S: Input word	IR, SR, AR, DM, HR, LR,
C: First parameter word	IR, SR, DM, HR, LR
D: Output word	IR, SR, AR, DM, HR, LR

Limitations

C and C+32 must be within the same data area.

⚠ Caution

Do not program PID(—) in the following situations. Doing so may produce unexpected behavior: In interrupt programs, in subroutines, between IL(02) and ILC(03), between JMP(04) and JME(05), and in step programming (when using STEP(08) and SNXT(09)).

Description

PID(—) 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.

PID parameter words range from C through C+32. 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 time T_1/sampling period γ (See note 1.)			
C+3	Tdk = Derivative time Td/sampling period γ (See note 1.)			
C+4	Sampling period γ			
C+5	2-PID parameter (α) (See note 2.)			PID forward/ reverse designation
C+6	0	Input range	0	Output range
C+7 to C+32	Work area (Cannot be accessed directly from program.)			

Note

1. What is set in words 2 and 3 is not the actual integral and derivative times, but rather their values divided by the value of the sampling period γ.
2. Setting the 2-PID parameter (α) to 000 yields 0.65, the normal value.

Parameter Settings

Item	Contents	Setting range
Set value (SV)	This is the target value of the process being controlled.	Binary data (of the same number of bits as specified for the input range)
Proportional band	This is the parameter for P control expressing the proportional control range/total control range.	0001 to 9999 (4 digits BCD); (0.1% to 999.9%, in units of 0.1%)
Tik	This is a constant expressing the strength of the integral operation. As this value increases, the integral strength decreases.	0001 to 8191 (4 digits BCD); (9999 = Integral operation not executed)
Tdk	This is a constant expressing the strength of the derivative operation. As this value increases, the derivative strength increases.	0000 to 8191 (4 digits BCD)
Sampling period	This sets the period for executing the PID operation.	0001 to 1023 (4 digits BCD); (0.1 to 102.3 s, in units of 0.1 s)
PID forward/reverse designation	This is the parameter that determines the direction of the proportional operation.	0: Reverse operation 1: Forward operation
2-PID parameter (α)	This is 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.
Input range	This is the number of input data bits.	0: 8 bits 1: 9 bits 2: 10 bits 3: 11 bits 4: 12 bits 5: 13 bits 6: 14 bits 7: 15 bits 8: 16 bits
Output range	This is the number of output data bits. {The number of output bits is automatically the same as the number of input bits.)	

PID CONTROL Operation

Execution Condition OFF

All data that has been set is retained. Then the execution condition is OFF, the operation amount 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 operation is begin. Sudden and radical changes in the operation output amount are not made when starting operation 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 operation 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

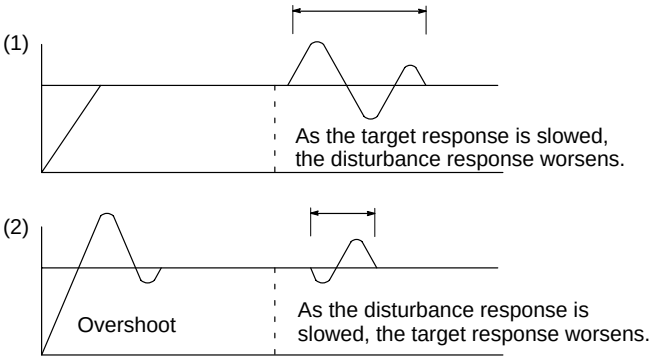
The sampling period is the time interval to retrieve the measurement data for carrying out a PID operation. PID(—), however, is executed according to C200HS cycle time, so there may be cases where the sampling period is exceeded. In such cases, the time interval until the next sampling is reduced.

PID Control Method

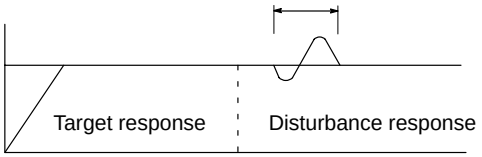
C200HS PID control operations are executed by means of PID control with feedforward control (two degrees of freedom).

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



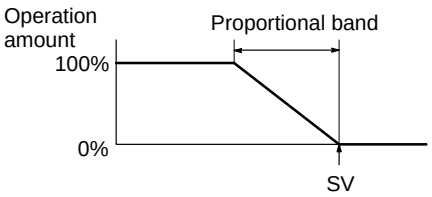
Control Operations

Proportional Operation (P)

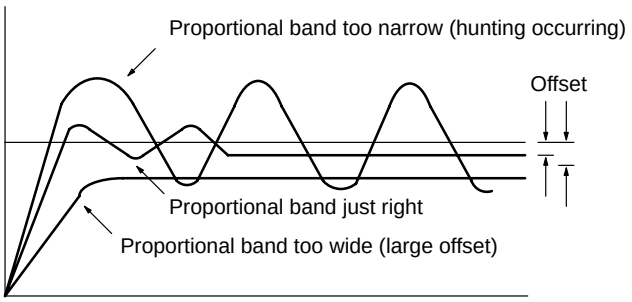
Proportional operation is an operation in which a proportional band is established with respect to the set value (SV), and within that band the operation amount (the control output amount) is made proportional to the deviation. If the present value (PV) is smaller than the proportional band, the operation amount will be 100%. If within the proportional band the operation amount is made proportional to the deviation and gradually decreased until the SV and PV match (i.e., until the deviation is 0), the operation amount will return to the previous value (forward operation).

The proportional band is expressed as a percentage with respect to the total input range. With proportional operation an offset (residual deviation) occurs, and the offset is reduced by making the proportional band smaller. If it is made too small, however, hunting will occur.

Proportional Operation (Forward Operation)



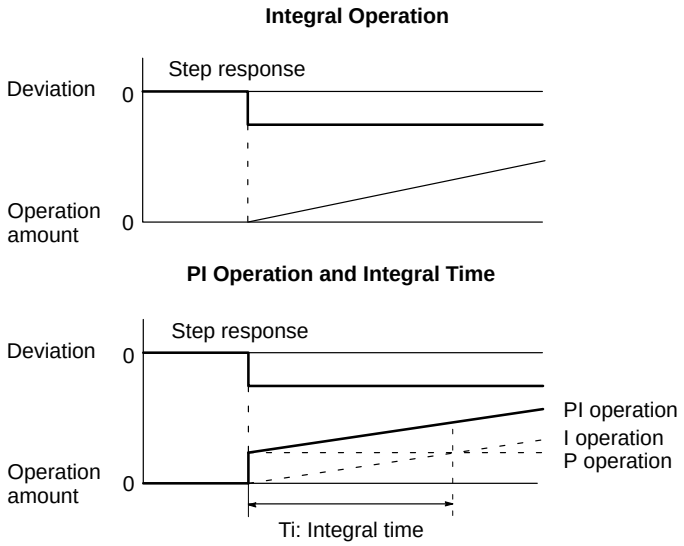
Adjusting the Proportional Band



Integral Operation (I)

Combining integral operation with proportional operation reduces the offset according to the time that has passed. The strength of the integral operation is indicated by the integral time, which is the time required for the integral operation amount to reach the same level as the proportional operation amount with respect to the step deviation, as shown in the following illustration. The shorter the integral time, the stronger the correction by the integral operation will be. If the

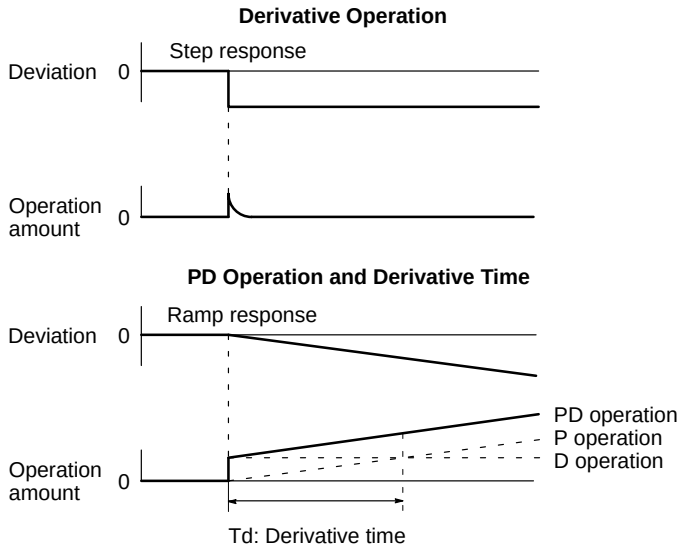
integral time is too short, the correction will be too strong and will cause hunting to occur.



Derivative Operation (D)

Proportional operation and integral operation both make corrections with respect to the control results, so there is inevitably a response delay. Derivative operation compensates for that drawback. In response to a sudden disturbance it delivers a large operation amount and rapidly restores the original status. A correction is executed with the operation amount made proportional to the incline (derivative coefficient) caused by the deviation.

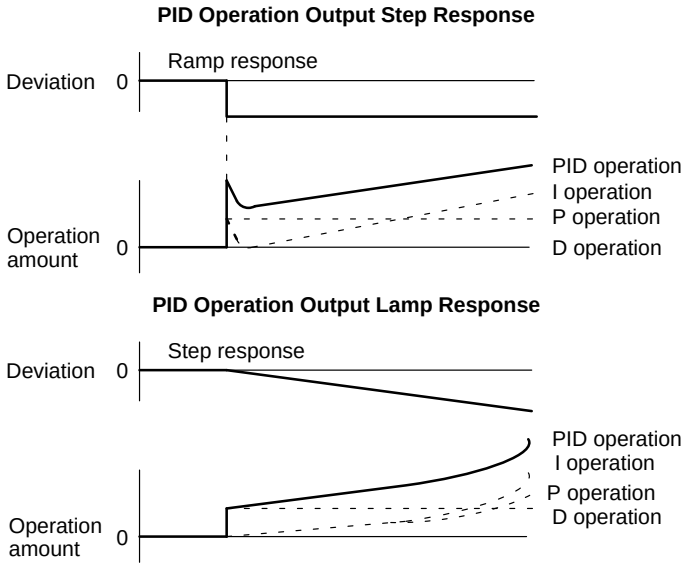
The strength of the derivative operation is indicated by the derivative time, which is the time required for the derivative operation amount to reach the same level as the proportional operation amount with respect to the step deviation, as shown in the following illustration. The longer the derivative time, the stronger the correction by the derivative operation will be.



PID Operation

PID operation combines proportional operation (P), integral operation (I), and derivative operation (D). It produces superior control results even for control objects with dead time. It employs proportional operation to provide smooth control

without hunting, integral operation to automatically correct any offset, and derivative operation to speed up the response to disturbances.



Direction of Operation

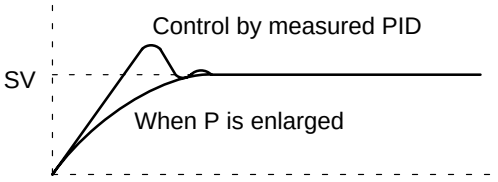
When using PID operation, select either of the following two control directions. In either direction, the operation amount increases as the difference between the SV and the PV increases.

- Forward operation: Control amount is increased when the PV is larger than the SV.
- Reverse operation: Control amount is increased when the PV is smaller than the SV.

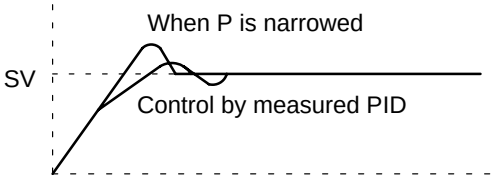
Adjusting PID Parameters

The general relationship between PID parameters and control status is shown below.

- When it is not a problem if a certain amount of time is required for stabilization (settlement time), but it is important not to cause overshooting, then enlarge the proportional band.

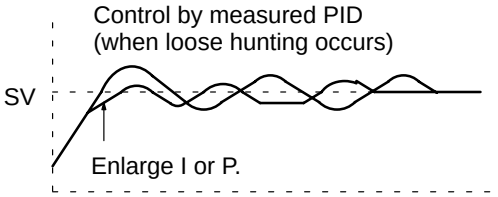


- When overshooting is not a problem but it is desirable to quickly stabilize control, then narrow the proportional band. If the proportional band is narrowed too much, however, then hunting may occur.

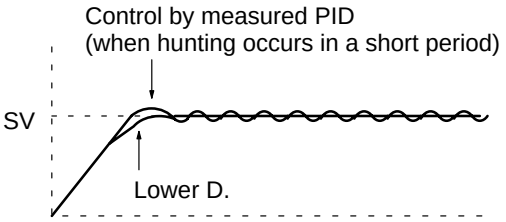


- When there is broad hunting, or when operation is tied up by overshooting and undershooting, it is probably because integral operation is too strong. The

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 operation is too strong. In that case, set the derivative operation lower.



Flags

ER: Content of ⁜DM word is not BCD, or the DM area boundary has been exceeded.

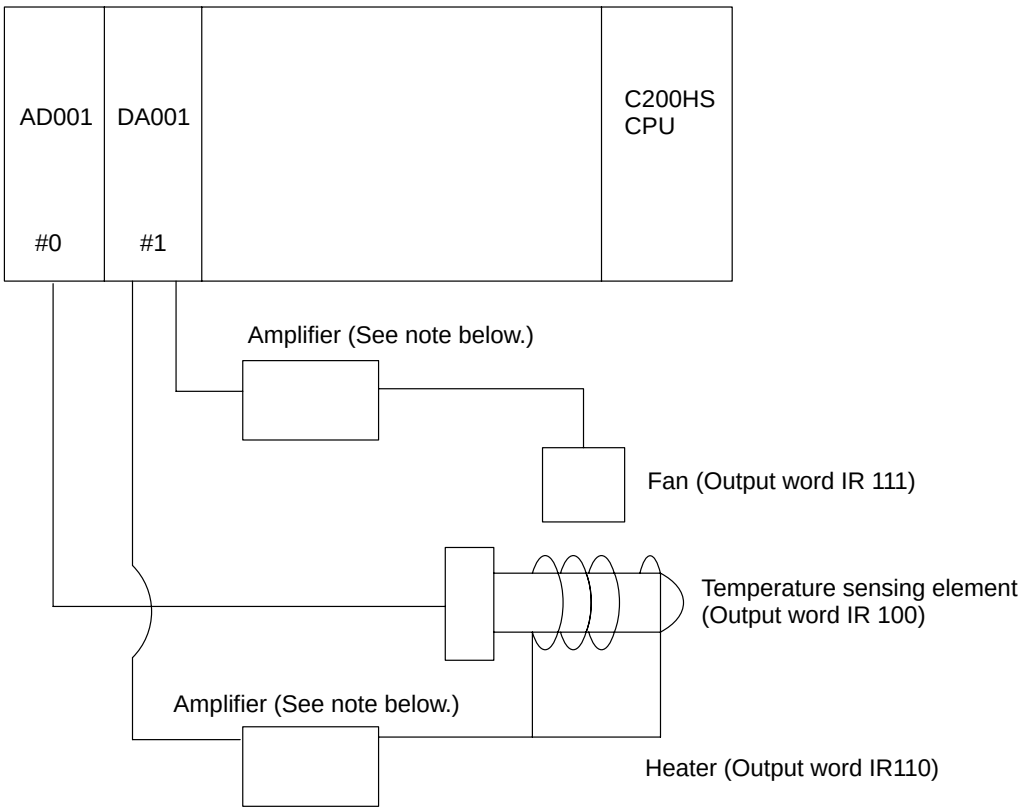
A PID parameter SV is out of range.

The PID operation was executed but the cycle time was two times the sampling period. PID(—) will be executed for this error only even when ER (SR25503) is ON.

CY: The PID operation is being executed.

Example

This example shows a PID control program using PID(—).



Note Motors and heaters cannot be directly connected from a Analog Output Unit. An amplifier (i.e., a current amplification circuit) is required.

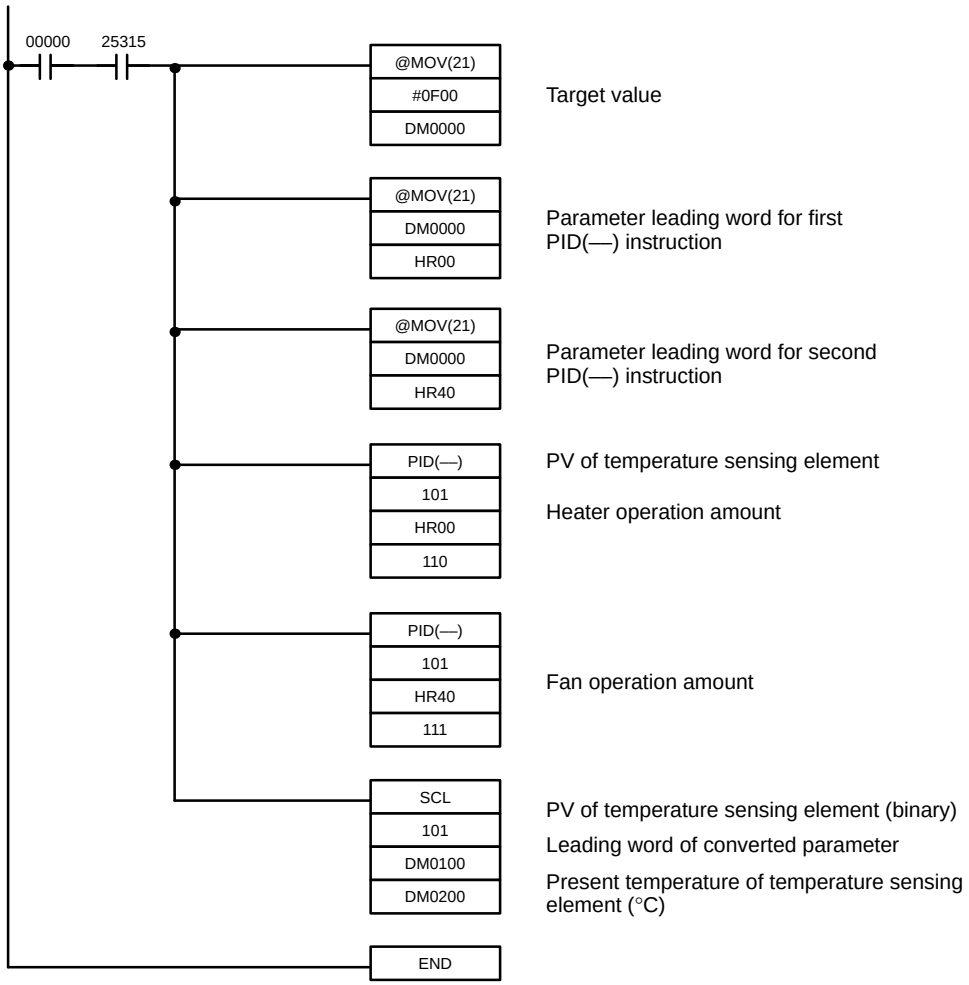
Creating the Program

Follow the procedure outlined below in creating the program.

1, 2, 3...

1. Set the target value (binary 0000 to 0FFF) in DM 0000.
2. Input the PV of the temperature sensing element (binary 000 to 0FFF) in bits 0 to 11 of word 101.
3. Output the operation amount of the heater to bits 0 to 11 of word 110 by means of the first PID(—) instruction in the following program.
4. Output the operation amount of the fan to bits 0 to 11 of word 111 by means of the second PID(—) instruction in the following program.
5. Convert the PV of the temperature sensing element (binary 000 to FFF) to temperature data (0000°C to 0200°C) by means of SCL(—), and output it to DM 0200.

Program



Note When using PID(—) or SCL(—), make the data settings in advance with a Peripheral Device such as the Programming Console or LSS.

Heater

HR 00	(DM0000)	Target value HR
HR 01	0080	Proportional band
HR 02	0200	Integral time/sampling period
HR 03	0100	Derivative time/sampling period
HR 04	0001	Sampling period
HR 05	0000	Forward/reverse designation/ PID parameters
HR 06	0404	I/O range

SCL Parameters

DM 0100	0000
DM 0101	0000
DM 0102	0200
DM 0103	0FFF

Fan

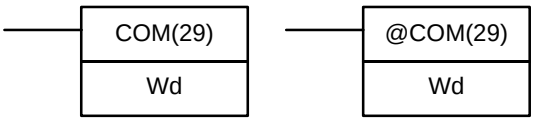
HR 40	(DM0000)
HR 41	0060
HR 42	0150
HR 43	0100
HR 44	0001
HR 45	0001
HR 46	0404

5-22 Logic Instructions

The logic instructions – COM(29), ANDW(34), ORW(35), XORW(36), and XNRW(37) – perform logic operations on word data.

5-22-1 COMPLEMENT – COM(29)

Ladder Symbols

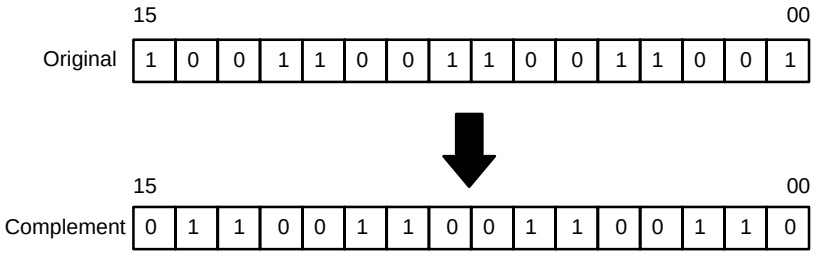


Operand Data Areas

Wd: Complement word	
IR, SR, AR, DM, HR, LR	

Description When the execution condition is OFF, COM(29) is not executed. When the execution condition is ON, COM(29) clears all ON bits and sets all OFF bits in Wd.

Example



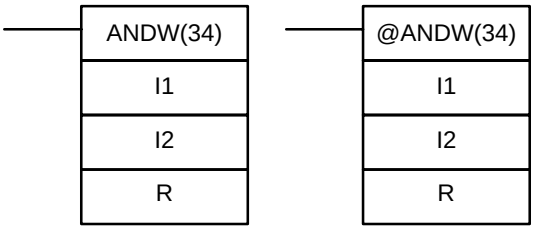
Flags

ER: Indirectly addressed DM word is non-existent. (Content of ⁎DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON when the result is 0.

5-22-2 LOGICAL AND – ANDW(34)

Ladder Symbols



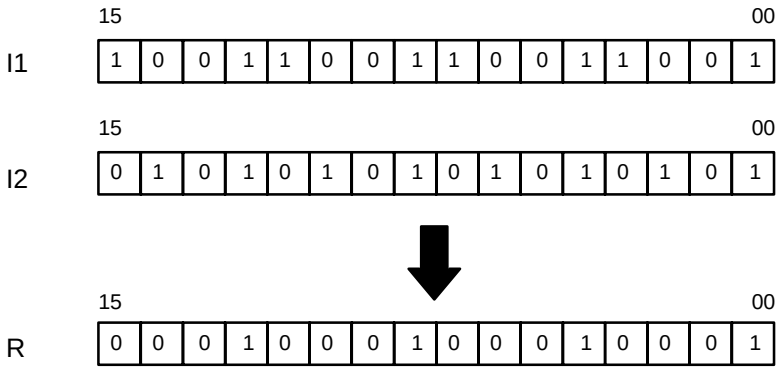
Operand Data Areas

I1: Input 1	IR, SR, AR, DM, HR, TC, LR, #
I2: Input 2	IR, SR, AR, DM, HR, TC, LR, #
R: Result word	IR, SR, AR, DM, HR, LR

Description

When the execution condition is OFF, ANDW(34) is not executed. When the execution condition is ON, ANDW(34) logically AND's the contents of I1 and I2 bit-by-bit and places the result in R.

Example



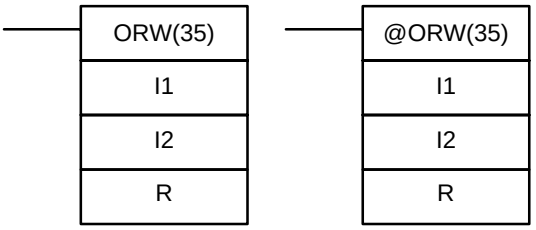
Flags

ER: Indirectly addressed DM word is non-existent. (Content of ⁎DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON when the result is 0.

5-22-3 LOGICAL OR – ORW(35)

Ladder Symbols



Operand Data Areas

I1: Input 1
IR, SR, AR, DM, HR, TC, LR, #

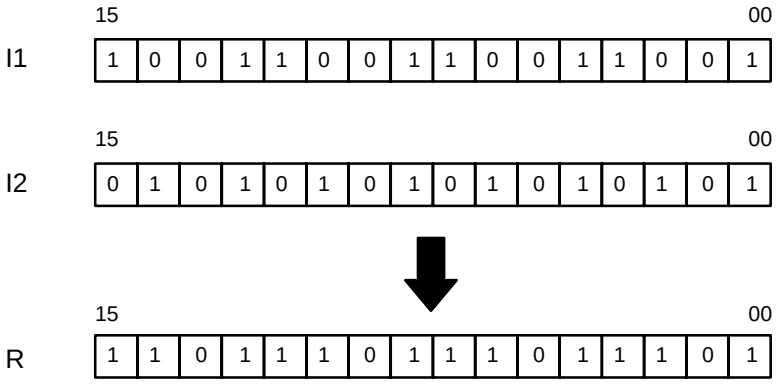
I2: Input 2
IR, SR, AR, DM, HR, TC, LR, #

R: Result word
IR, SR, AR, DM, HR, LR

Description

When the execution condition is OFF, ORW(35) is not executed. When the execution condition is ON, ORW(35) logically OR's the contents of I1 and I2 bit-by-bit and places the result in R.

Example



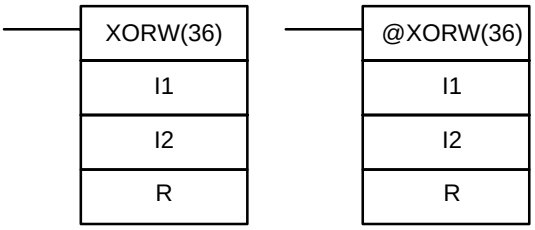
Flags

ER: Indirectly addressed DM word is non-existent. (Content of ✳DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON when the result is 0.

5-22-4 EXCLUSIVE OR – XORW(36)

Ladder Symbols

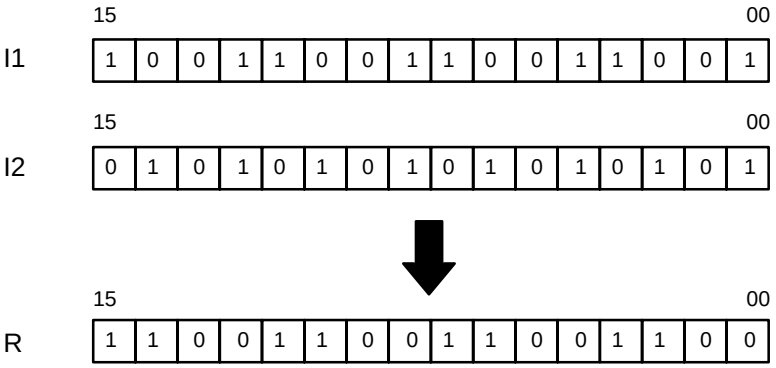


Operand Data Areas

I1: Input 1	
IR, SR, AR, DM, HR, TC, LR, #	
I2: Input 2	
IR, SR, AR, DM, HR, TC, LR, #	
R: Result word	
IR, SR, AR, DM, HR, LR	

Description When the execution condition is OFF, XORW(36) is not executed. When the execution condition is ON, XORW(36) exclusively OR's the contents of I1 and I2 bit-by-bit and places the result in R.

Example



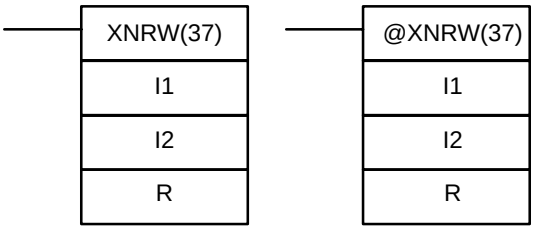
Flags

ER: Indirectly addressed DM word is non-existent. (Content of ⁎DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON when the result is 0.

5-22-5 EXCLUSIVE NOR – XNRW(37)

Ladder Symbols

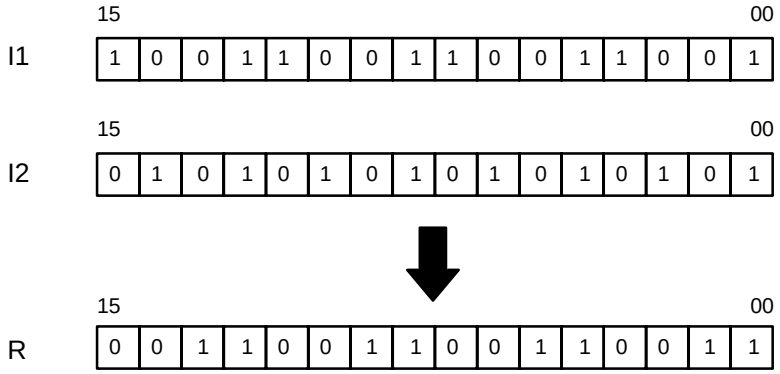


Operand Data Areas

I1: Input 1	IR, SR, AR, DM, HR, TC, LR, #
I2: Input 2	IR, SR, AR, DM, HR, TC, LR, #
R: Result word	IR, SR, AR, DM, HR, LR

Description

When the execution condition is OFF, XNRW(37) is not executed. When the execution condition is ON, XNRW(37) exclusively NOR's the contents of I1 and I2 bit-by-bit and places the result in R.



Flags

ER: Indirectly addressed DM word is non-existent. (Content of ✳DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON when the result is 0.

5-23 Subroutines and Interrupt Control

5-23-1 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).

Subroutines may also be activated by interrupts or the MCRO(99) instruction.

Interrupts

Like subroutine calls, 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 subroutine. An interrupt is caused either by an external source, such as an input signal from an Interrupt Input Unit, or a scheduled interrupt. In the case of the scheduled interrupt, the interrupt signal is repeated at regular intervals.

Whereas subroutine calls are controlled from within the main program, subroutines activated by interrupts are triggered when the interrupt signal is received.

In the case of the scheduled interrupt, the time interval between interrupts is set by the user and is unrelated to the cycle timing of the PC. This capability is useful for periodic supervisory or executive program execution.

INT(89) is used to control the interrupt signals received from the Interrupt Input Unit, and also to control the scheduling of the scheduled interrupt. INT(89) provides such functions as masking of interrupts (so that they are recorded but ignored) and clearing of interrupts.

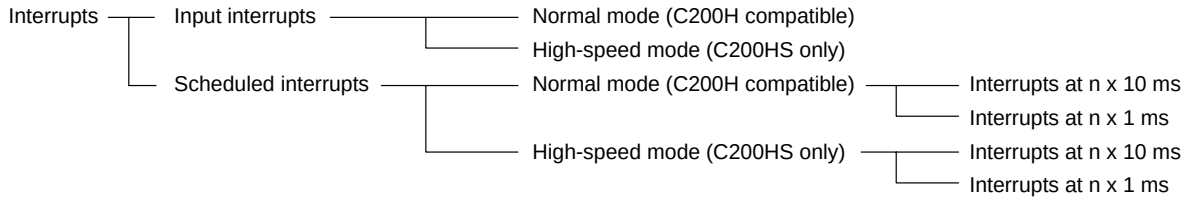
Refer to *5-23-2 Interrupts* for more details on interrupts.

MCRO(99)

The MACRO instruction allows a single subroutine (programming pattern) to replace several subroutines that have identical structure but different operands. Since a number of similar program sections can be managed with just one subroutine, the number of program steps can be greatly reduced. Refer to *5-23-5 MACRO – MCRO(99)* for more details on this instruction.

5-23-2 Interrupts

The C200HS supports both input interrupts and scheduled interrupts. The operating modes for these interrupts are shown in the following illustration.



Input Interrupts

Input interrupts are executed when external inputs are received via a C200HS-INT01 Interrupt Input Unit. The Interrupt Input Unit provides at total of 8 inputs numbered IN 0 through IN7 that can be used to generate interrupts #00 to #07. Generally speaking, subroutines #00 to #07 are executed when interrupts #00 to #07 are generated.

Only one Interrupt Input Unit can be mounted to each C200HS PC. When any of the 8 inputs on the Interrupt Input Unit goes ON, an interrupt is generated and the corresponding bit in the word allocated to the slot to which the Unit is mounted is turned ON.

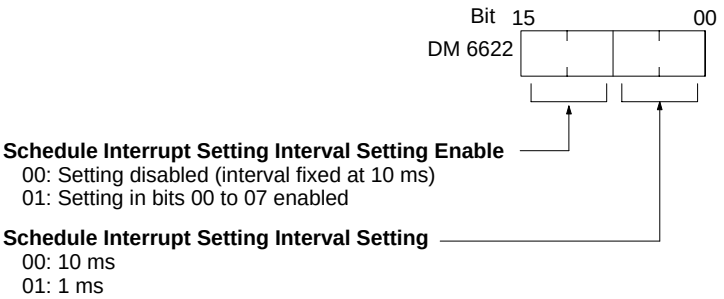
Note

1. Subroutines #00 to #07 can be used as normal subroutines when they are not used for input interrupts.
2. Inputs on the Interrupt Input Unit not used for interrupts can be used for normal inputs.

Scheduled Interrupts

Scheduled interrupts can be executed at intervals set either in increments of 10 ms or in increments or 1 ms. Interrupt #99 is used and subroutine #99 is executed.

The unit used to set the scheduled interrupt interval is set in the PC Setup at DM 6622.



Note Subroutine #99 can be used as a normal subroutine when it is not used for a scheduled interrupt.

Normal Interrupt Mode (C200H Compatible)

The following setting is used for normal interrupt mode.

DM 6620	0	0	0	0

In normal interrupt mode, the following processing will be completed once started even if an interrupt occurs The interrupt will be processed as soon as the current process is completed.

- Host Link servicing
- Remote I/O servicing
- Special I/O Unit servicing
- Individual instruction execution
- SYSMAC LINK/SYSMAC NET servicing (CPU31-E/33-E only)

Use this mode whenever using C200H interrupt subroutines without modification or whenever possible considering the response time required for interrupts.

Note The C200HS is set to normal interrupt mode by default.

High-speed Interrupt Mode (C200HS)

The following setting is used for high-speed interrupt mode.

DM 6620	1	–	–	–

Interrupt mode
(1 = high-speed)

In high-speed interrupt mode, the following processing will be interrupted and the interrupt subroutine executed as soon as an interrupt is generated.

- Host Link servicing
- Remote I/O servicing
- Special I/O Unit servicing
- Individual instruction execution

Use this mode whenever interrupt response time must be accurate to 1.0 ms.

Note

1. With the CPU31-E/33-E, SYSMAC LINK/SYSMAC NET servicing will be completed before processing interrupts, meaning up to 10 ms may be required to respond to and interrupt even in the High-speed mode.
2. Data will not necessarily be concurrent if the high-speed interrupt mode is used because Host Link servicing, remote I/O servicing, Special I/O Unit servicing, and individual instruction execution will not necessarily be completed when started. The program must be designed to allow for this when required by the application. (See the section on data concurrence for further details.)

Interrupt Priority

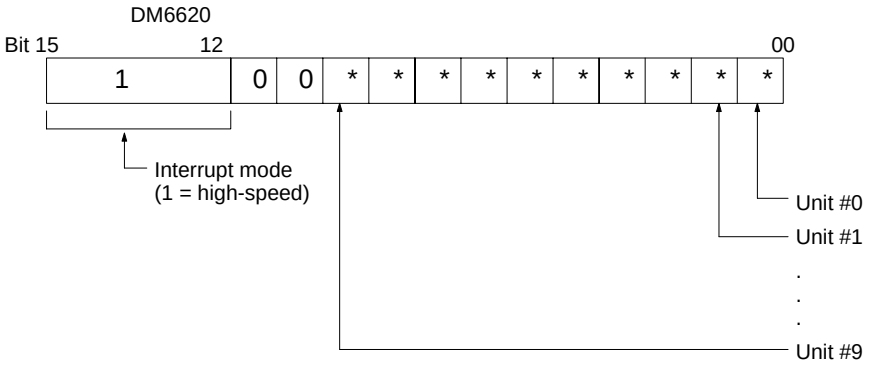
The specified subroutine will be executed when an interrupt is generated. If further interrupts are generated during execution of an interrupt subroutine, they will not be processed until execution of the current interrupt subroutine has been completed. If more than one interrupt is generated or is awaiting execution at the same time, the corresponding subroutines will be executed in the following order of priority.

Input interrupt 1 > input interrupt 2 > ... > input interrupt 7 > scheduled interrupt

Special I/O in Interrupt Subroutines

I/O for Special I/O Units can be refreshed from within interrupt subroutines by using the I/O REFRESH (IORF) instruction. If the high-speed interrupt mode is used, refreshing in the normal cycle (END refreshing and IORF refreshing in the main program) must be disabled for the Special I/O Unit that is to be refreshed in the interrupt subroutine. An interrupt programming error (system FAL error 8B) will occur if the same special I/O is refreshed both within an interrupt program and within the normal cycle, and the special I/O will not be refreshed within the interrupt subroutine.

The PC Setup for the C200HS contains settings in DM 6620 that disable refreshing in the normal cycle for specific Special I/O Units. This settings are as shown below.



Note Disabling special I/O refreshing in the normal cycle to refresh special I/O in an interrupt subroutine is necessary only in the high-speed mode. Disabling normal cycle refreshing of special I/O during normal interrupt mode will be ignored and the special I/O will be refreshed both in the normal cycle and in the interrupt subroutine.

The execution time of interrupt subroutines must be kept to less than10 ms if the high-speed interrupt mode is used and Special I/O Units, Host Link Units, or Remote I/O Units are programmed. An interrupt programming error (system FAL error 8B) will occur if the execution time is 10 ms or greater.

The execution time of interrupt subroutine with the longest execution time is output to SR 262 and the number of the subroutine with the longest execution time is output to SR 263.

Example: 12.3 ms for subroutine #80

SR 262	0 1 2 3	Maximum interrupt subroutine execution time (in 0.1 ms)
SR 263	8 0 * *	No. of interrupt subroutine with maximum execution time

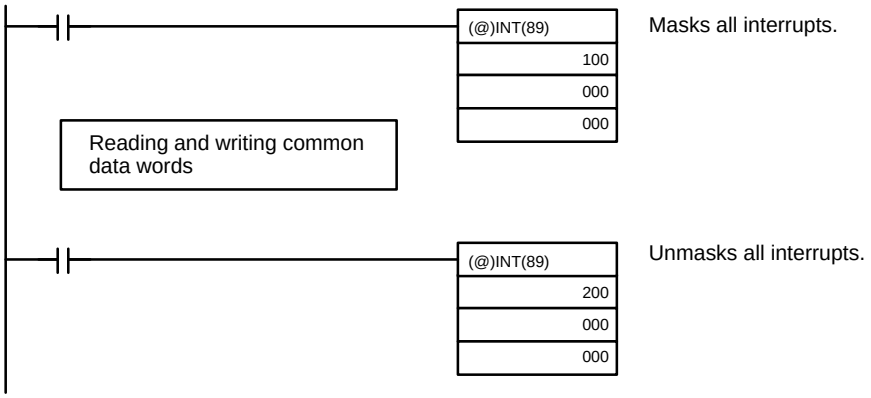
Note The above 10-ms limit does not apply when the normal interrupt mode is used or when the above Units are not mounted.

Data Concurrence

Although data concurrence is not a problem for execution of normal arithmetic instructions or comparison instructions, it can be a problem when executing longer instructions that handle multiple words, such as block transfer instructions, when the high-speed interrupt mode is used and the same data is handled both in the main program and in an interrupt subroutine.

Data may not be concurrent in two different situations: 1) if a data write operation in the main program is interrupted and the same data is read in an interrupt subroutine and 2) if a data read operation in the main program is interrupted and the same data is written in an interrupt subroutine.

If you must handle the same data both in the main program and in an interrupt subroutine, use programming such as that shown below to be sure that data concurrence is preserved, i.e., mask interrupts while read/writing data that is also handled in an interrupt subroutine.



Data concurrence can also be a problem if interrupts occur during data transfers occurring in servicing for Special I/O Units, remote I/O, or Host Link Systems. For any of these, data can be non-concurrent down to byte units.

Use one of the following methods to preserve data concurrence in the above situations. The second methods applies to Special I/O Units only.

- Mask interrupts in the main program while moving data transferred to/from Units to different words and use these alternate words in the interrupt subroutine.
- Use the I/O REFRESH instruction in interrupt subroutines to refresh required I/O from Special I/O Units and mask interrupts in the main program while reading/writing Special I/O Unit words.

5-23-3 SUBROUTINE ENTER – SBS(91)

Ladder Symbol

Definer Data Areas

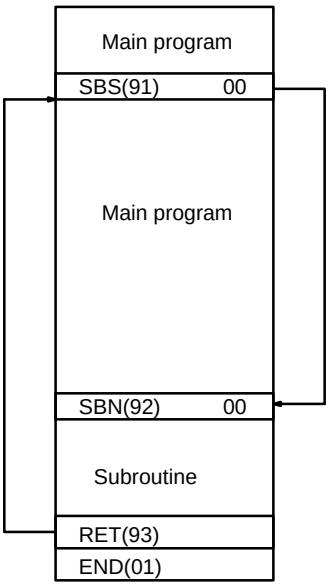
N: Subroutine number
00 to 99

Limitations

Subroutine numbers 00 through 07 are used with input interrupts and subroutine number 99 is used for the scheduled interrupt.

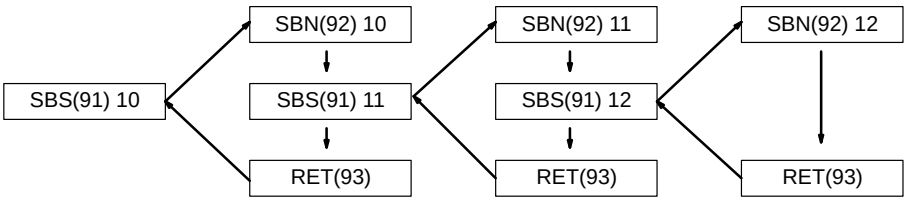
Description

A subroutine can be executed by placing SBS(91) in the main program at the point where the subroutine is desired. The subroutine number used in SBS(91) indicates the desired subroutine. When SBS(91) is executed (i.e., when the execution condition for it is ON), the instructions between the SBN(92) with the same subroutine number and the first RET(93) after it are executed before execution returns to the instruction following the SBS(91) that made the call.

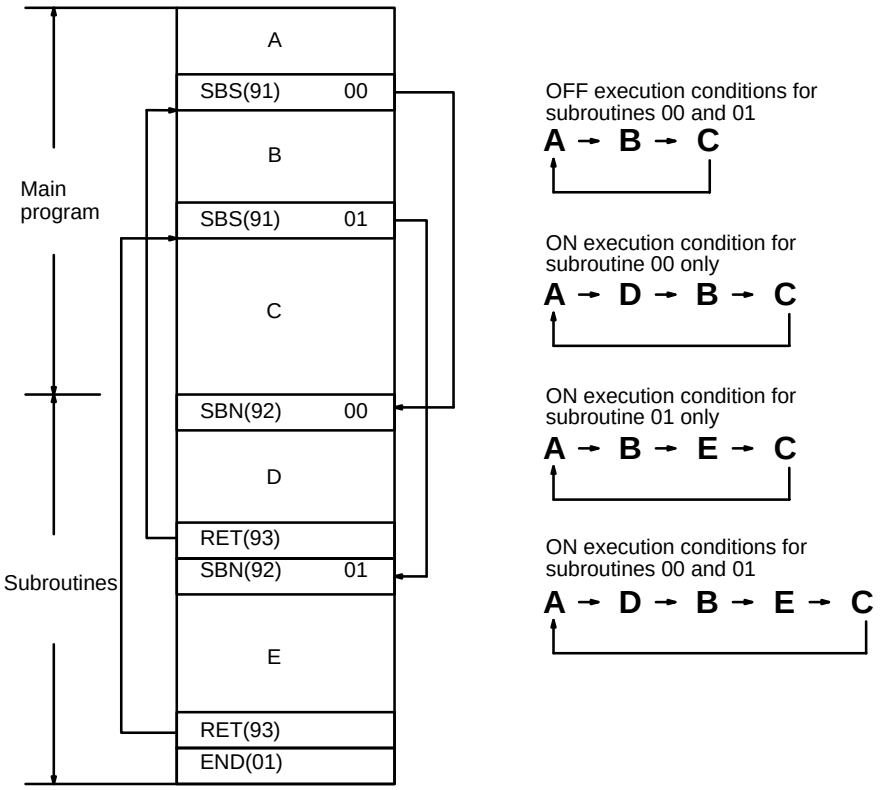


SBS(91) may be used as many times as desired in the program, i.e., the same subroutine may be called from different places in the program).

SBS(91) 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(93) has been reached), program execution returns to the original subroutine which is then completed before returning to the main program. Nesting is possible to up to sixteen levels. A subroutine cannot call itself (e.g., SBS(91) 00 cannot be programmed within the subroutine defined with SBN(92) 00). The following diagram illustrates two levels of nesting.



The following diagram illustrates program execution flow for various execution conditions for two SBS(91).



Note A non-fatal error (error code 8B) will be generated if a subroutine's execution time exceeds 10 ms.

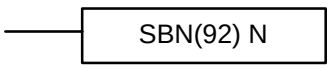
Flags

ER: A subroutine does not exist for the specified subroutine number.
A subroutine has called itself.
An active subroutine has been called.

Caution SBS(91) will not be executed and the subroutine will not be called when ER is ON.

5-23-4 SUBROUTINE DEFINE and RETURN – SBN(92)/RET(93)

Ladder Symbols



Definer Data Areas

N: Subroutine number
00 to 99

Limitations Each subroutine number can be used in SBN(92) once only.

Description SBN(92) is used to mark the beginning of a subroutine program; RET(93) is used to mark the end. Each subroutine is identified with a subroutine number, N, that is programmed as a definer for SBN(92). This same subroutine number is used in any SBS(91) that calls the subroutine (see 5-23-3 SUBROUTINE ENTER – SBS(91)). No subroutine number is required with RET(93).

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(92) before returning to address 00000 for the next cycle. Subroutines will not be executed unless called by SBS(91).

END(01) must be placed at the end of the last subroutine program, i.e., after the last RET(93). It is not required at any other point in the program.

Precautions

If SBN(92) 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(92) is encountered.

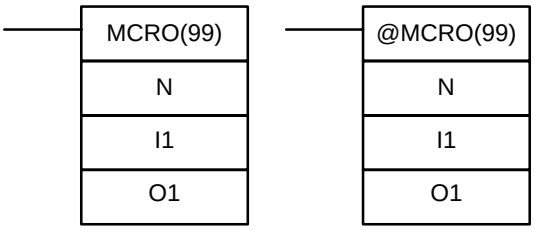
If either DIFU(13) or DIFU(14) is placed within a subroutine, the operand bit will not be turned OFF until the next time the subroutine is executed, i.e., the operand bit may stay ON longer than one cycle.

Flags

There are no flags directly affected by these instructions.

5-23-5 MACRO – MCRO(99)

Ladder Symbols



Operand Data Areas

N: Subroutine number
00 to 99

I1: First input word
IR, SR, AR, DM, HR, TC, LR

O1: First output word
IR, SR, AR, DM, HR, LR

Limitations

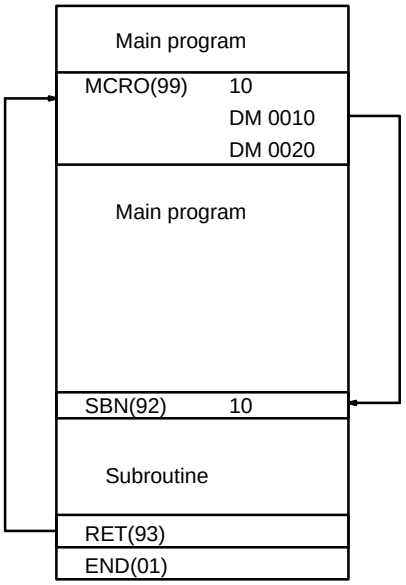
I1 and I1+3 must be in the same data area, as must O1 and O1+3.

Description

The MACRO instruction allows a single subroutine to replace several subroutines that have identical structure but different operands. There are 4 input words, SR 290 to SR 293, and 4 output words, SR 294 to SR 297, allocated to MCRO(99). These 8 words are used in the subroutine and take their contents from I1 to I1+3 and O1 to O1+3 when the subroutine is executed.

When the execution condition is OFF, MCRO(99) is not executed. When the execution condition is ON, MCRO(99) copies the contents of I1 to I1+3 to SR 290 to SR 293, copies the contents of O1 to O1+3 to SR 294 to SR 297, and then calls and executes the subroutine specified in N. When the subroutine is completed, the contents of SR 294 through SR 297 are then transferred back to O1 to O1+3 before MCRO(99) is completed.

In the following example, the contents of DM 0010 through DM 0013 are copied to SR 290 through SR 293, the contents of DM 0020 through DM 0023 are copied to SR 294 through SR 297, and subroutine 10 is called and executed. When the subroutine is completed, the contents of SR 294 through SR 297 are copied back to DM 0020 to DM 0023.



Note

1. Subroutines for macros are programmed just like other subroutines, except that SR 290 to SR 297 contents are transferred in from the designated input and output words.
2. Not only external I/O words, but internal I/O words can be used for I1 and O1.
3. SR 290 to SR 297 can be used as work bits when not used for macro programs.

Precautions

MCRO(99) can be used only for program sections that can be written using four or fewer consecutive input words and/or four or fewer consecutive output words. It is thus generally necessary to consider system and program design together to take full advantage of macro programming.

Be careful that the input and output words properly correspond to the macro input and output words.

Flags

ER: A subroutine does not exist for the specified subroutine number.

An operand has exceeded a data area boundary.

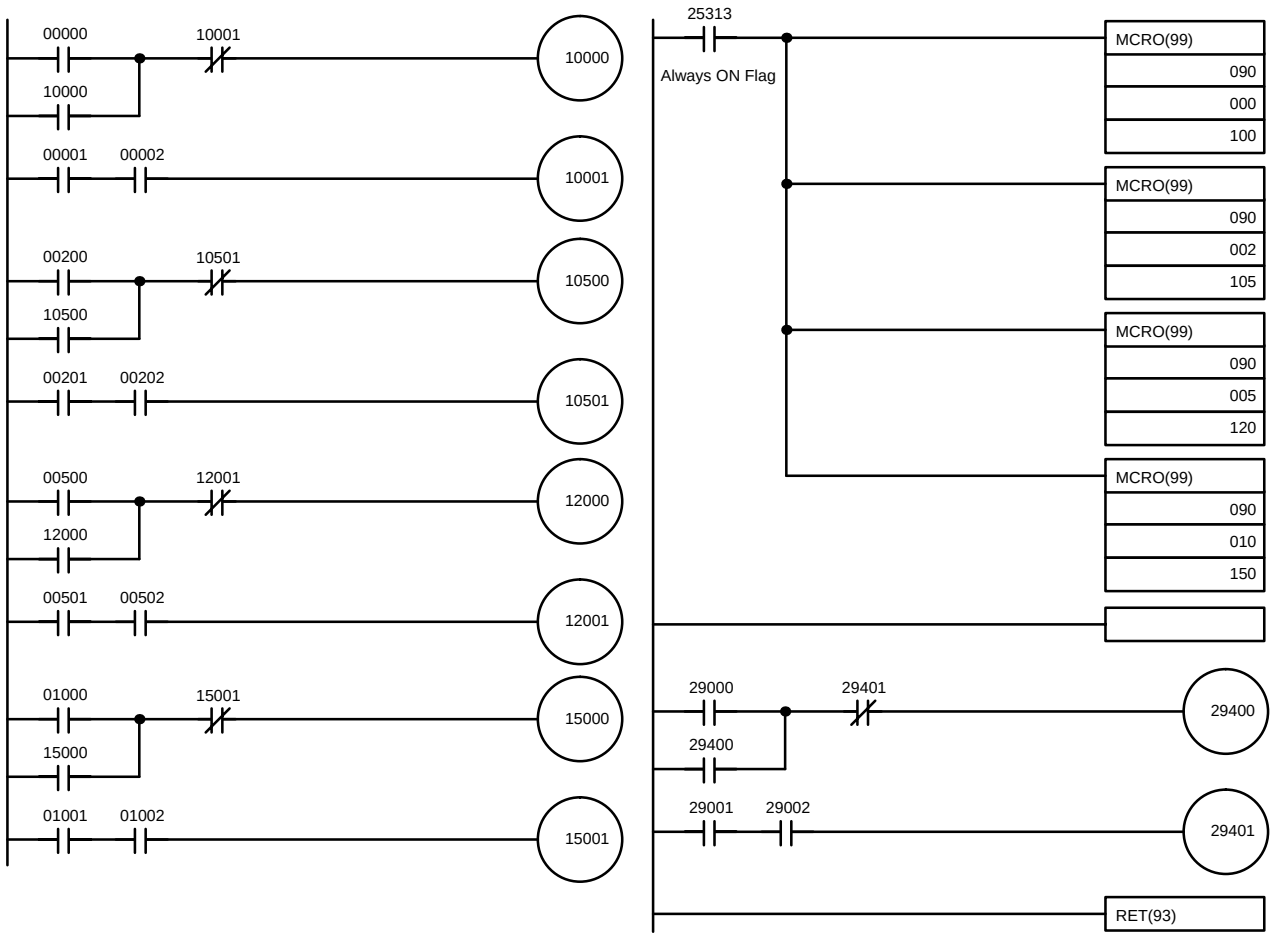
Indirectly addressed DM word is non-existent. (Content of ⁎DM word is not BCD, or the DM area boundary has been exceeded.)

A subroutine has called itself.

An active subroutine has been called.

Example

The following examples shows the use of four MCRO(99) instructions that access the same subroutine. The program section on the left shows the same program without the use of MCRO(99).



5-23-6 INTERRUPT CONTROL – INT(89)

Ladder Symbols

INT(89)
C
N
D

@INT(89)
C
N
D

Operand Data Areas

C: Control code
(000, 001, 002, 100, or 200)

N: Interrupt type
(000 or 004)

D: Control data
IR, AR, DM, HR, TC, LR, #

Limitations

D must be between 0000 and 00FF when N=000 and C=000 or 001.

D must be BCD between 0001 and 9999 when N=004 and C=000 or 001.

Description

INT(89) is used to control interrupts and performs one of 8 functions depending on the values of C and N. As shown in the following tables, three of the functions act on input interrupts, three act on the scheduled interrupt, and the other two mask or unmask all interrupts.

Interrupt	Value of C	INT(89) Function	Comments
Input (N=000)	000	Mask/unmask input interrupts	Bits 00 to 07 of D indicate inputs 00 to 07.
	001	Clear input interrupts	
	002	Read current mask status	Status written to D.
Scheduled (N=004)	000	Set interrupt interval	
	001	Set time to first interrupt	
	002	Read interrupt interval	

The following 2 functions depend on the value of C only.

Value of C	INT(89) Function
100	Mask all interrupts
200	Unmask all interrupts

Mask/unmask Input Interrupts (N=000, C=000)

This function is used to mask and unmask input interrupts 00 to 07. Masked inputs are recorded, but ignored. When an input is masked, the interrupt program for it will be run as soon as the bit is unmasked (unless it is cleared beforehand by executing INT(89) with C=001 and N=000).

Set the corresponding bit in D to 0 to unmask or to 1 to mask an I/O interrupt input. Bits 00 to 07 correspond to 00 to 07.

Clear Input Interrupts (N=000, C=001)

This function is used to clear I/O interrupt inputs 00 to 07. Since interrupt inputs are recorded, masked interrupts will be serviced after the mask is removed unless they are cleared first.

Set the corresponding bit in D to 1 to clear an interrupt input. Bits 00 to 07 correspond to 00 to 07.

Read Current Mask Status (N=000, C=002)

This function is used to write the current mask status for input interrupts 00 to 07 to word D. The corresponding bit will be ON if the input is masked. (Bits 00 to 07 correspond to 00 to 07.)

Set Interrupt Interval (N=004, C=000)

This function is used to set the interval between scheduled interrupts. The content of D (BCD: 0001 to 9999) is multiplied by the scheduled interrupt time unit (1 ms or 10 ms) to obtain the scheduled interrupt interval.

The scheduled interrupt time unit is set in DM 6622 of the PC Setup. Refer to *3-6-4 PC Setup* for details on setting this time unit.

Set Time to First Interrupt (N=004, C=001)

This function is used to set the time to the first scheduled interrupt. The content of D (BCD: 0001 to 9999) is multiplied by the scheduled interrupt time unit (1 ms or 10 ms) to obtain the time to the first scheduled interrupt. The scheduled interrupt time unit is set in DM 6622 of the PC Setup. Refer to *3-6-4 PC Setup* for details on setting this time unit.

Be sure to set the time to the first interrupt. If this setting is not made, the interval to the first interrupt (set with N=004, C=000) will be uncertain.

Use the First Cycle Flag (SR 25315) for the execution condition to INT(89) when setting the time to the first interrupt (C=001). The scheduled interrupt might never occur if the C=001 setting is made continuously.

Read Interrupt Interval (N=004, C=002)

This function is used to write the current setting for the scheduled interrupt interval to word D.

Mask/Unmasking All Interrupts (C=100/200)

This function is used to mask or unmask all interrupt processing. Masked inputs are recorded, but ignored. The masked inputs will be serviced as soon as they are unmasked. This function masks or unmask all interrupts at the same time and is independent of the masks created with other functions.

The control data, D, is not used for this function. Set D to #0000.

Flags

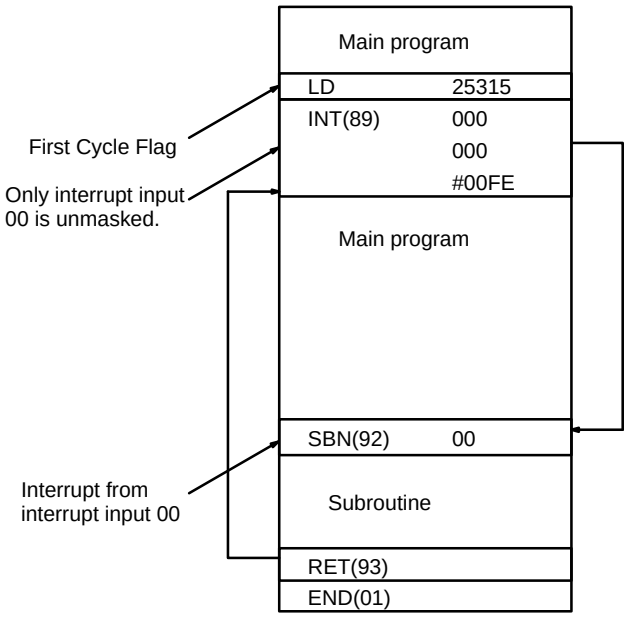
ER: Indirectly addressed DM word is non-existent. (Content of ✳DM word is not BCD, or the DM area boundary has been exceeded.)

C, and/or N are not within specified values.

Example 1: Input Interrupt

This example shows how to unmask a particular interrupt input. Input interrupt subroutines will be executed when the CPU receives the corresponding interrupt input, regardless of the location in the CPU's cycle. These interrupts are useful when using program sections of uncertain length, such as event programs.

All input interrupts are masked at the start of operation, and the desired interrupt input is unmasked using INT(89) with N=000 and C=000. As shown in the following diagram, the subroutine would be executed if there were an input from input interrupt 00 when that interrupt input was unmasked.

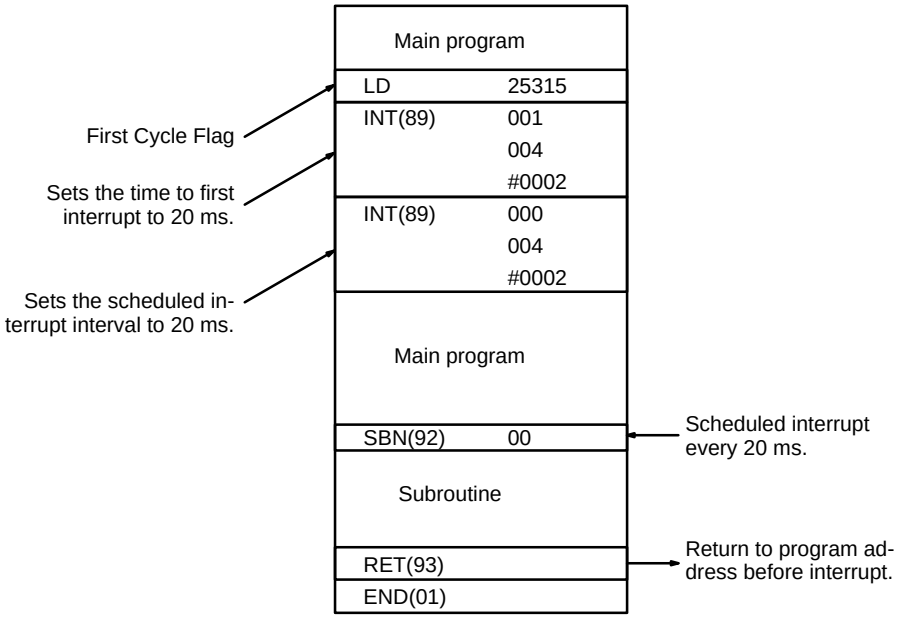


Note Depending on the setting of DM 6621 in the PC Setup, Host Link servicing, Remote I/O servicing, Special I/O Unit servicing, and individual instruction execution will be completed before the subroutine is executed. Refer to page 255 for details.

Example 2: Scheduled Interrupt

This example shows how to set the interval between scheduled interrupts. Scheduled interrupt subroutines will be executed at fixed intervals, regardless of the location in the CPU's cycle. This interrupt is useful for program sections such as regular monitoring programs.

The scheduled interrupt is disabled at the start of operation (the scheduled interrupt interval is 0), so the time to the first interrupt and scheduled interrupt interval must be set using INT(89) with N=004 and C=001/000. In the following diagram, the subroutine would be executed every 20 ms if the scheduled interrupt time unit is set to 10 ms in DM 6622 of the PC Setup.



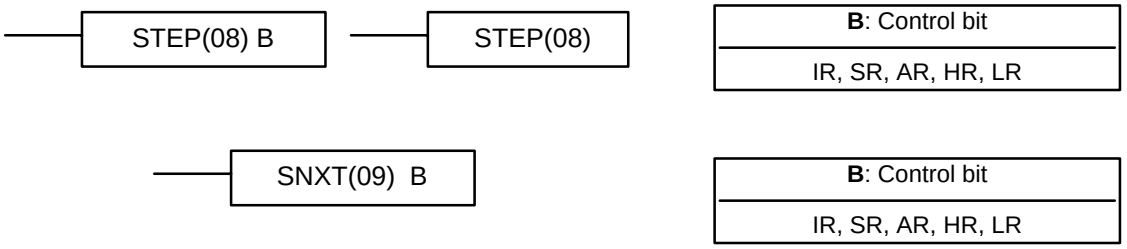
Note Depending on the setting of DM 6621 in the PC Setup, Host Link servicing, Remote I/O servicing, Special I/O Unit servicing, and individual instruction execution will be completed before the subroutine is executed. Refer to page 255 for details.

5-24 Step Instructions

The step instructions STEP(08) and SNXT(09) are used in conjunction to set up breakpoints 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 like a normal programming code, except that certain instructions (e.g., IL(02)/ILC(03), JMP(04)/JME(05)) may not be included.

5-24-1 STEP DEFINE and STEP START–STEP(08)/SNXT(09)

Ladder Symbols **Definer Data Areas**



Limitations

All control bits must be in the same word and must be consecutive.

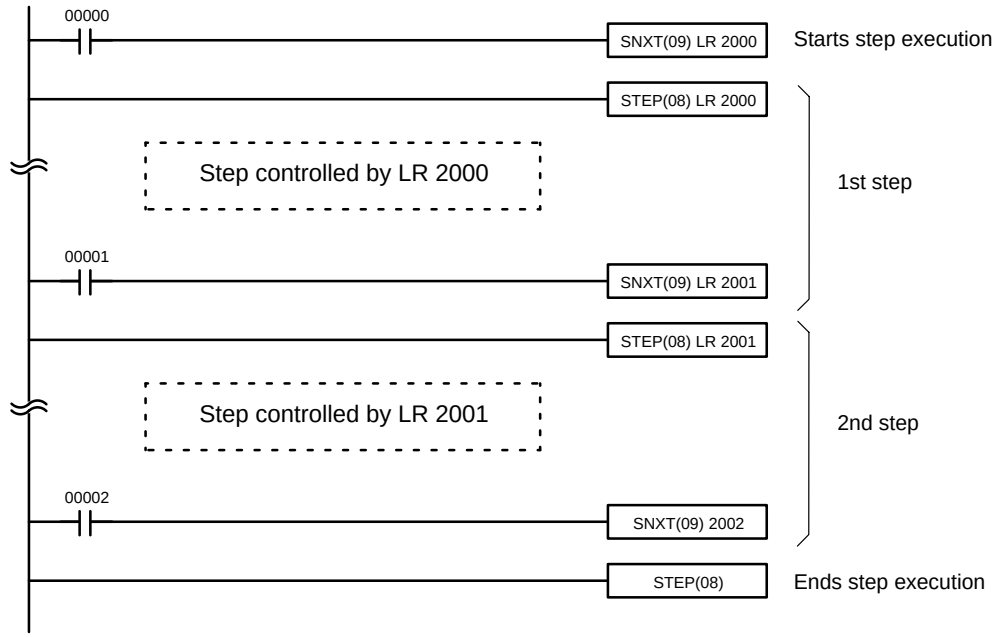
IR 29800 to IR 29915 cannot be used for B.

Description

STEP(08) uses a control bit in the IR or HR areas to define the beginning of a section of the program called a step. STEP(08) does not require an execution condition, i.e., its execution is controlled through the control bit. To start execution of the step, SNXT(09) is used with the same control bit as used for STEP(08). If SNXT(09) 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. The SNXT(09) instruction must be written into the program so that it is executed 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(09) will not be executed.

Once SNXT(09) is used in the program, step execution will continue until STEP(08) is executed without a control bit. STEP(08) without a control bit must be preceded by SNXT(09) with a dummy control bit. The dummy control bit may be any unused IR or HR bit. It cannot be a control bit used in a STEP(08).

Execution of a step is completed either by execution of the next SNXT(09) or by turning OFF the control bit for the step (see example 3 below). When the step is completed, all of the IR and HR bits in the step are turned OFF. All timers in the step except TTIM(—) are reset to their SVs. TTIM(—), counters, shift registers, bits set or reset with SET or RSET, and bits used in KEEP(11) maintain status. Two simple steps are shown below.



Address	Instruction	Operands
00000	LD	00000
00001	SNXT(09)	LR 2000
00002	STEP(08)	LR 2000
Step controlled by 20200.		
00100	LD	00001
00101	SNXT(09)	LR 2001

Address	Instruction	Operands
00102	STEP(08)	LR 2001
Step controlled by 20201.		
00200	LD	00002
00201	SNXT(09)	LR 2002
00202	STEP(08)	---

Steps can be programmed in consecutively. Each step must start with STEP(08) and generally ends with SNXT(09) (see example 3, below, for an exception). 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(09) determine how the steps are executed. The three examples given below demonstrate these three types of step execution.

Precautions

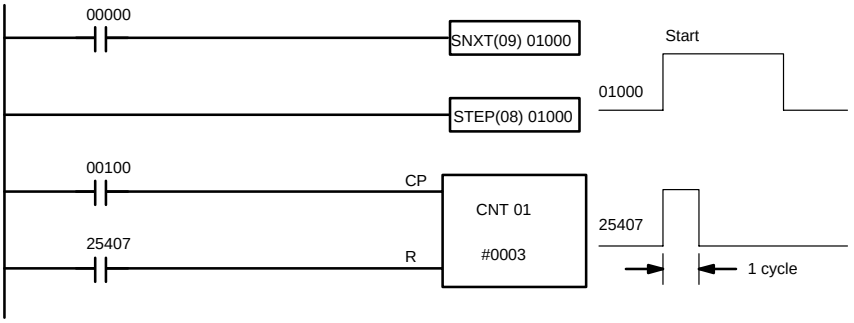
Interlocks, jumps, SBN(92), and END(01) cannot be used within step programs.

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.

If IR or LR bits are used for control bits, their status will be lost during any power interruption. If it is necessary to maintain status to resume execution at the same step, HR bits must be used.

Flags

25407: Step Start Flag; turns ON for one cycle when STEP(08) is executed and can be used to reset counters in steps as shown below if necessary.



Address	Instruction	Operands
00000	LD	00000
00001	SNXT(09)	01000
00002	STEP(08)	01000
00003	LD	00100

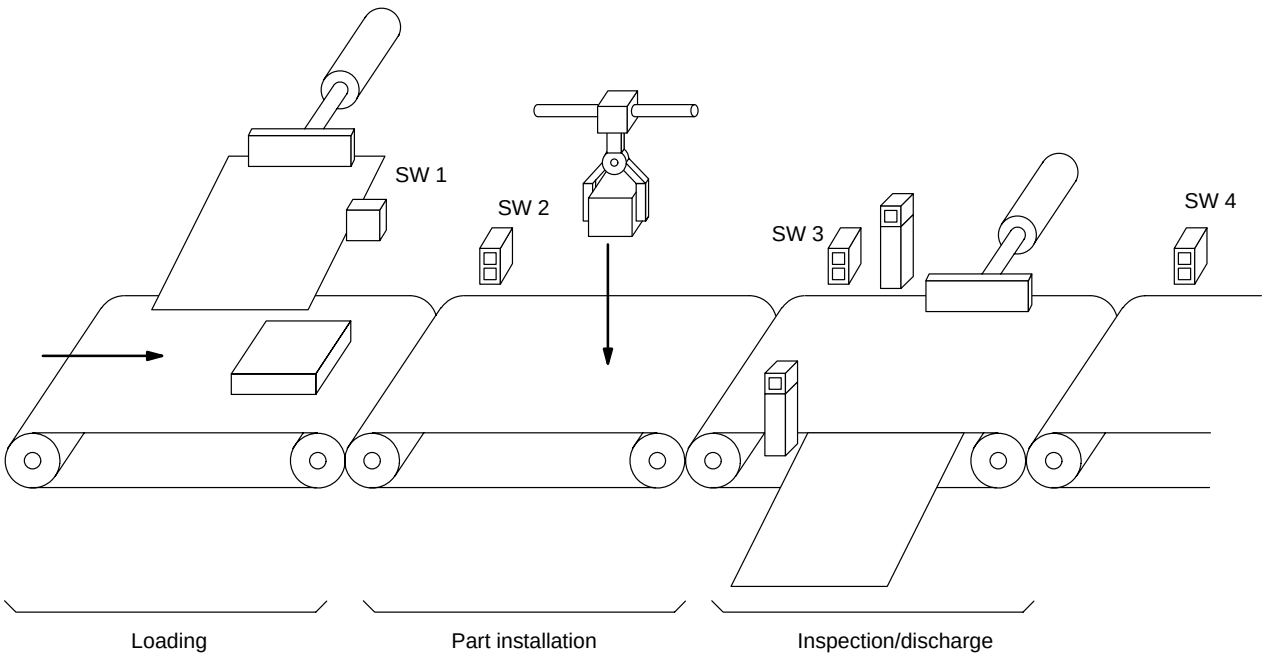
Address	Instruction	Operands
00004	LD	25407
00005	CNT	01
		# 0003

Examples

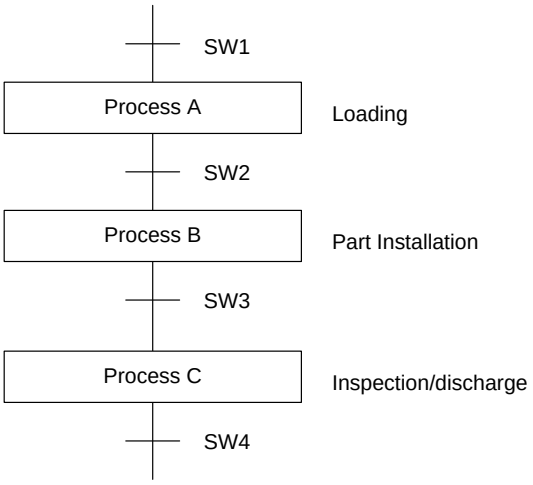
The following three examples demonstrate the three types of execution control possible with step programming. Example 1 demonstrates sequential execution; example 2, branching execution; and example 3, parallel execution.

Example 1: Sequential Execution

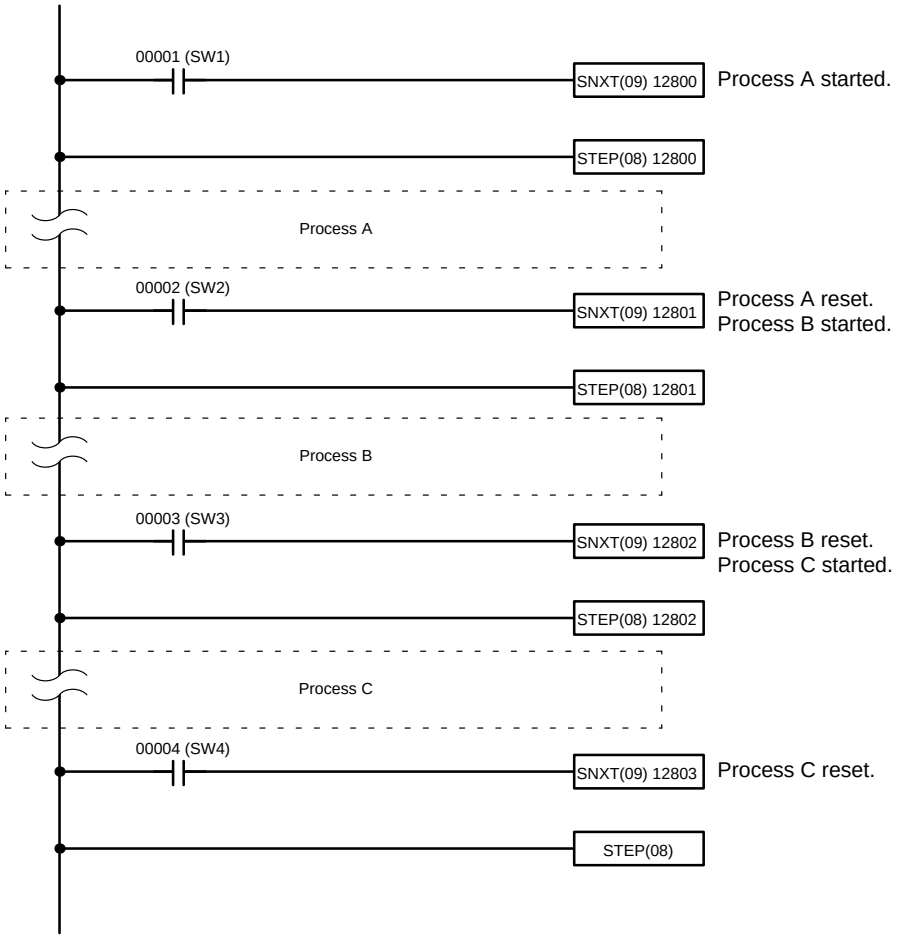
The following process requires that three processes, loading, part installation, and inspection/discharge, be executed in sequence with each process being reset before continuing on the the next process. Various sensors (SW1, SW2, SW3, and SW4) are positioned to signal when processes are to start and 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(09) that starts the next step. Each step starts when the switch that indicates the previous step has been completed turns ON.

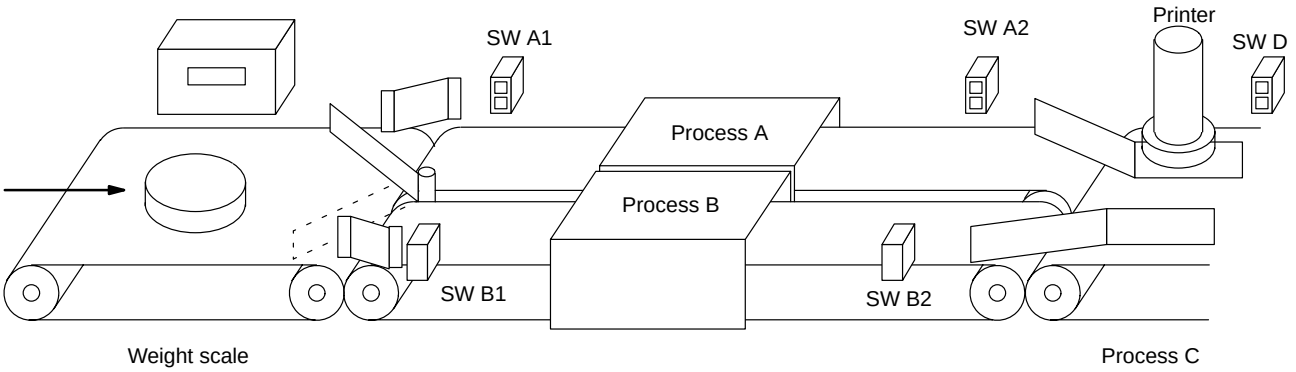


Address	Instruction	Operands
00000	LD	00001
00001	SNXT(09)	12800
00002	STEP(08)	12800
	Process A	
00100	LD	00002
00101	SNXT(09)	12801
00102	STEP(08)	12801

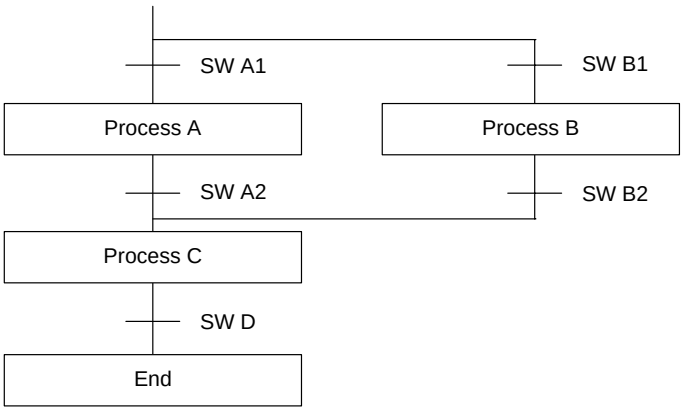
Address	Instruction	Operands
	Process B	
00100	LD	00003
00101	SNXT(09)	12802
00102	STEP(08)	12802
	Process C	
00200	LD	00004
00201	SNXT(09)	12803
00202	STEP(08)	---

Example 2: Branching Execution

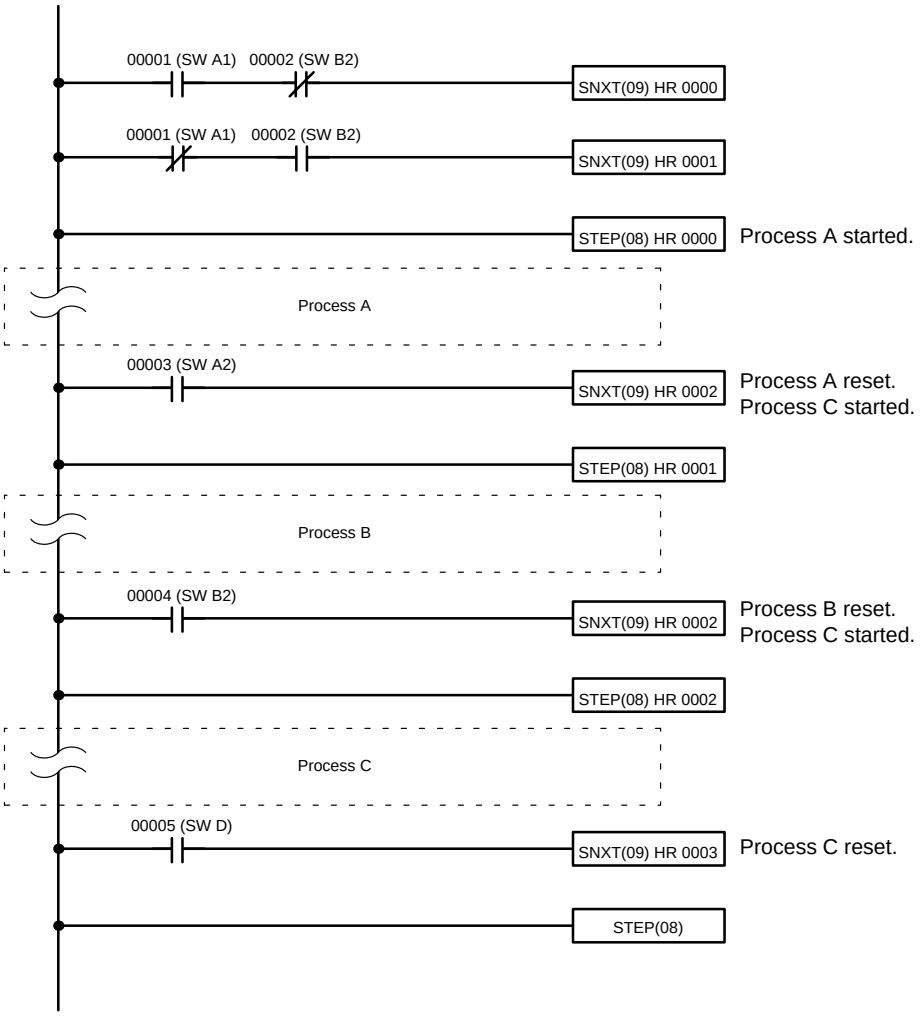
The following process requires that a product is processed in one of two ways, depending on its weight, before it is printed. The printing process is the same regardless of which of the first processes is used. Various sensors are positioned to signal when processes are to start and end.



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(09) instructions that start processes A and B. Because of the way 00001 (SW A1) and 00002 (SB B1) are programmed, only one of these will be executed to start either process A or process B. Both of the steps for these processes end with a SNXT(09) that starts the step for process C.

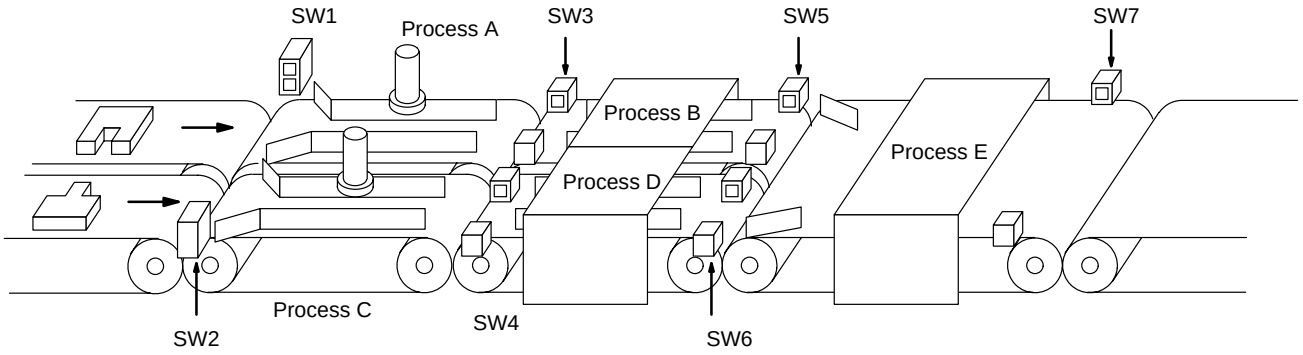


Address	Instruction	Operands	
00000	LD		00001
00001	AND NOT		00002
00002	SNXT(09)	HR	0000
00003	LD NOT		00001
00004	AND		00002
00005	SNXT(09)	HR	0001
00006	STEP(08)	HR	0000
	Process A		
00100	LD		00003
00101	SNXT(09)	HR	0002
00102	STEP(08)	HR	0001

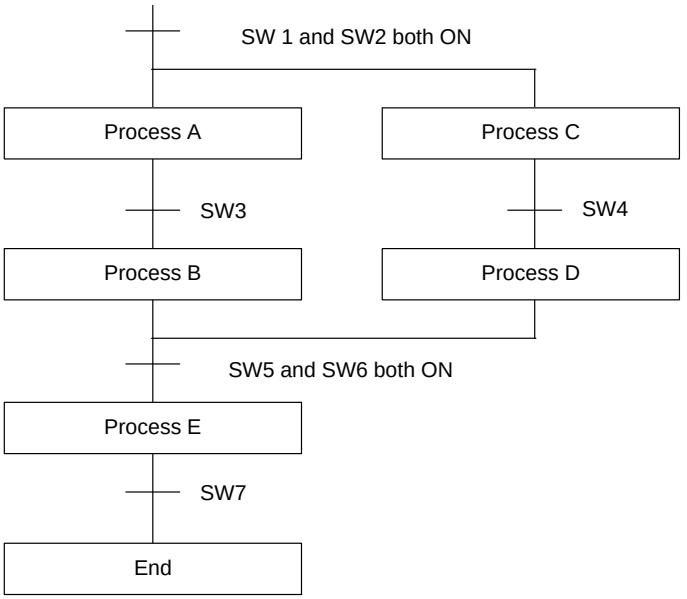
Address	Instruction	Operands	
	Process B		
00100	LD		00004
00101	SNXT(09)	HR	0002
00102	STEP(08)	HR	0002
	Process C		
00200	LD		00005
00201	SNXT(09)	HR	0003
00202	STEP(08)	---	

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.

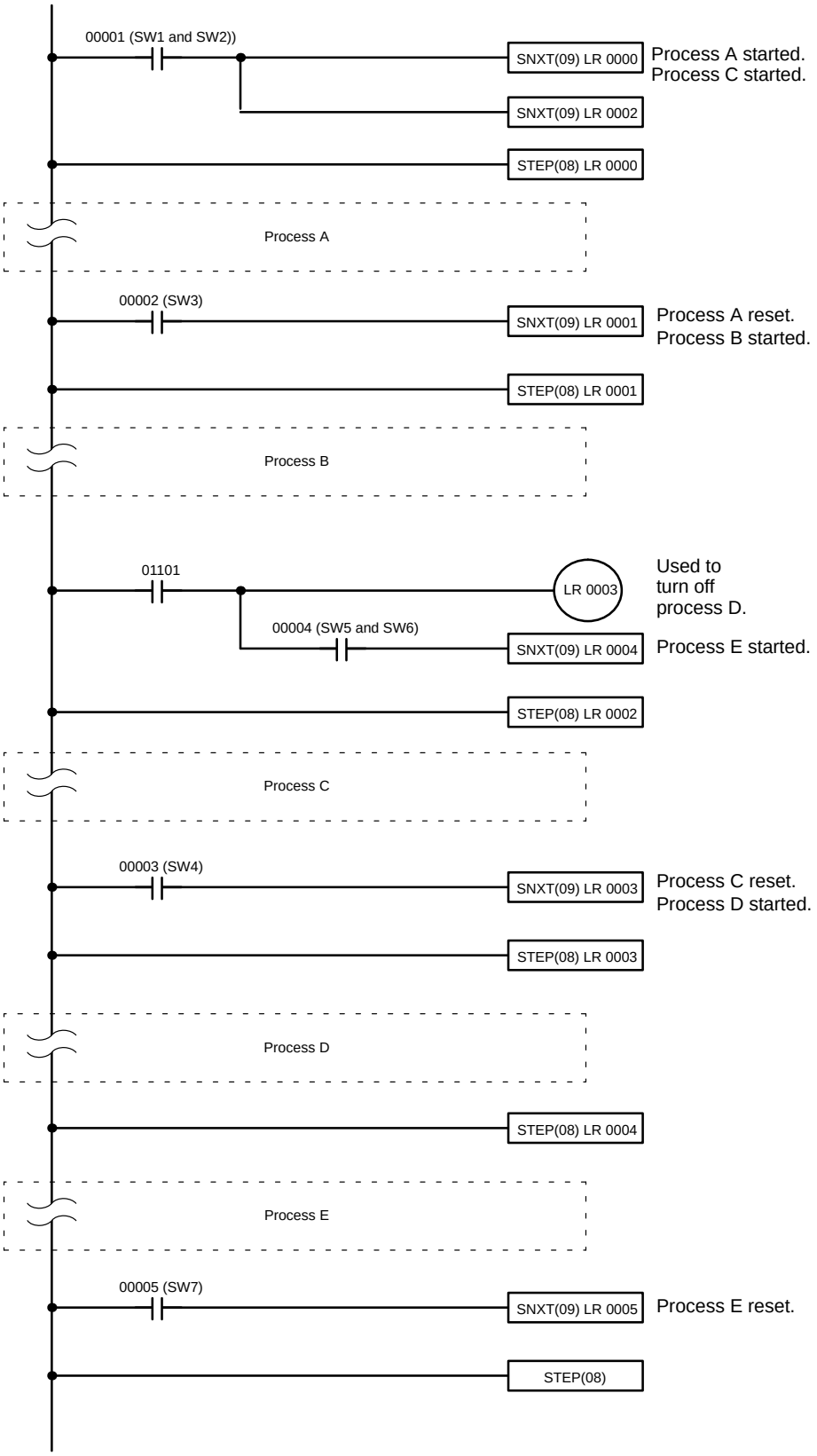


The following diagram demonstrates the flow of processing and the switches that are used for execution control. Here, process A and process C are started together. When process A finishes, process B starts; when process C finishes, process D starts. When both processes B and D have finished, process E starts.



The program for this operation, shown below, starts with two SNXT(09) 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 the status for both of them is “ON”, but SW5 and SW6 have turned ON), processes B and D are reset together by the SNXT(09) at the end of the programming for process B. Although there is no SNXT(09) at the end of process D, the control bit for it is turned OFF by executing SNXT(09) LR 0004. This is because the OUT for LR 0003 is in the step reset by SNXT(09) LR 0004, i.e., LR 003 is turned OFF when SNXT(09) LR 0004 is executed Process B is thus reset directly and process D is reset indirectly before executing the step for process E.


00001 (SW1 and SW2))
SNXT(09) LR 0000
Process A started.
Process C started.
SNXT(09) LR 0002
STEP(08) LR 0000
Process A
00002 (SW3)
SNXT(09) LR 0001
Process A reset.
Process B started.
STEP(08) LR 0001
Process B
01101
LR 0003
Used to
turn off
process D.
00004 (SW5 and SW6)
SNXT(09) LR 0004
Process E started.
STEP(08) LR 0002
Process C
00003 (SW4)
SNXT(09) LR 0003
Process C reset.
Process D started.
STEP(08) LR 0003
Process D
STEP(08) LR 0004
Process E
00005 (SW7)
SNXT(09) LR 0005
Process E reset.
STEP(08)

Address	Instruction	Operands
00000	LD	00001
00001	SNXT(09)	LR 0000
00002	SNXT(09)	LR 0002
00003	STEP(08)	LR 0000
	Process A	
00100	LD	00002
00101	SNXT(09)	LR 0001
00102	STEP(08)	LR 0001
	Process B	
00100	LD	01101
00101	OUT	LR
0003		
00101	AND	00004
00101	SNXT(09)	LR 0004

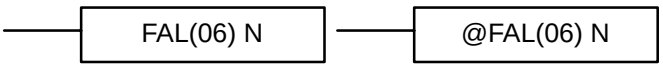
Address	Instruction	Operands
00102	STEP(08)	LR 0002
	Process C	
00200	LD	00003
00201	SNXT(09)	LR 0003
00202	STEP(08)	LR 0003
	Process D	
00300	STEP(08)	LR 0004
	Process E	
00400	LD	00005
00401	SNXT(09)	LR 0005
00402	STEP(08)	---

5-25 Special Instructions

The instructions in this section are used for various operations, including programming user error codes and messages, counting ON bits, setting the watchdog timer, and refreshing I/O during program execution.

5-25-1 FAILURE ALARM – FAL(06) and SEVERE FAILURE ALARM – FALS(07)

Ladder Symbols



Definer Data Areas

N: FAL number
(00 to 99)

N: FAL number
(01 to 99)

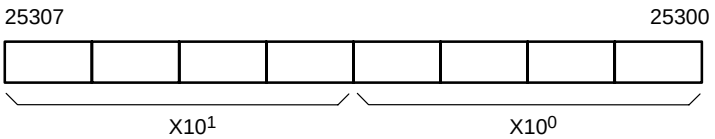
Limitations

FAL(06) and FALS(07) share the same FAL numbers. Be sure to use a number in either FAL(06) or FALS(07), not both.

Description

FAL(06) and FALS(07) are provided so that the programmer can output error numbers for use in operation, maintenance, and debugging. When executed with an ON execution condition, either of these instructions will output a FAL number to bits 00 to 07 of SR 253. The FAL number that is output can be between 01 and 99 and is input as the definer for FAL(06) or FALS(07). FAL(06) with a definer of 00 is used to reset this area (see below).

FAL Area



FAL(06) produces a non-fatal error and FAL(07) produces a fatal error. When FAL(06) is executed with an ON execution condition, the ALARM/ERROR indicator on the front of the CPU will flash, but PC operation will continue. When FALS(07) is executed with an ON execution condition, the ALARM/ERROR indicator will light and PC operation will stop.

The system also generates error codes to the FAL area.

Resetting Errors

All FAL error codes will be retained in memory, although only one of these is available in the FAL area. To access the other FAL codes, reset the FAL area by executing FAL(06) 00. Each time FAL(06) 00 is executed, another FAL error will be moved to the FAL area, clearing the one that is already there. FAL error codes are recorded and will be recalled in the following order: First code generated, 9A, 9B, 9D, 8A, 8B, and then 01 to 99.

Clearing Messages

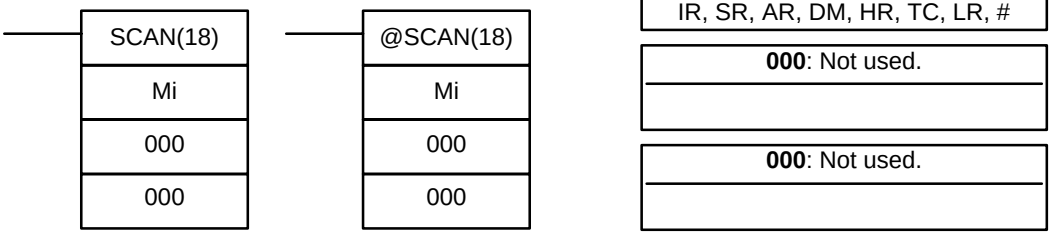
FAL(06) 00 is also used to clear message programmed with the instruction, MSG(46).

If the FAL area cannot be cleared, as is generally the case when FALS(07) is executed, first remove the cause of the error and then clear the FAL area through the Programming Console (see *4-6-5 Clearing Error Messages*).

5-25-2 CYCLE TIME – SCAN(18)

Operand Data Areas

Ladder Symbols



Limitations

Mi must be BCD. Only the rightmost three digits of Mi are used.

Description

SCAN(18) is used to set a minimum cycle time. Mi is the minimum cycle time that will be set in tenths of milliseconds, e.g., if Mi is 1200, the minimum cycle time will be 120.0 ms. The possible setting range is from 000.0 to 999.9 seconds.

If the actual cycle time is less than the cycle time set with SCAN(18) the CPU will wait until the designated time has elapsed before starting the next cycle. If the actual cycle time is greater than the set time, the set time will be ignored and the program will be executed to completion.

Flags

ER: Indirectly addressed DM word is non-existent. (Content of ⁎DM word is not BCD, or the DM area boundary has been exceeded.)

Mi is not BCD.

5-25-3 TRACE MEMORY SAMPLING – TRSM(45)

Data tracing can be used to facilitate debugging programs. To set up and use data tracing it is necessary to have a host computer running LSS; no data tracing is possible from a Programming Console. Data tracing is described in detail in the *LSS Operation Manual*. This section shows the ladder symbol for TRSM(45) and gives an example program.

Ladder Symbol

Description

TRSM(45) 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 (refer to the *LSS Operation Manual* for details).

TRSM(45) is not controlled by an execution condition, but rather by two bits in the AR area: AR 2515 and AR 2514. AR 2515 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 the peripheral device. AR 2514 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 the Programming Device. A positive or negative delay can also be set to alter the actual point from which tracing will begin.

Data can be recorded in any of three ways. TRSM(45) can be placed at one or more locations in the program to indicate where the specified data is to be traced. If TRSM(45) is not used, the specified data will be traced when END(01) is executed. The third method involves setting a timer interval from the peripheral devices so that the specified data will be tracing at a regular interval independent of the cycle time (refer to the *LSS Operation Manual*).

TRSM(45) can be incorporated anywhere in a program, any number of times. The data in the trace memory can then be monitored via a Programming Console, host computer, etc.

AR 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.

Flag	Function
AR 2515	Sampling Start Bit
AR 2514	Trace Start Bit
AR 2513	Tracing Flag
AR 2512	Trace Completed Flag

Precautions

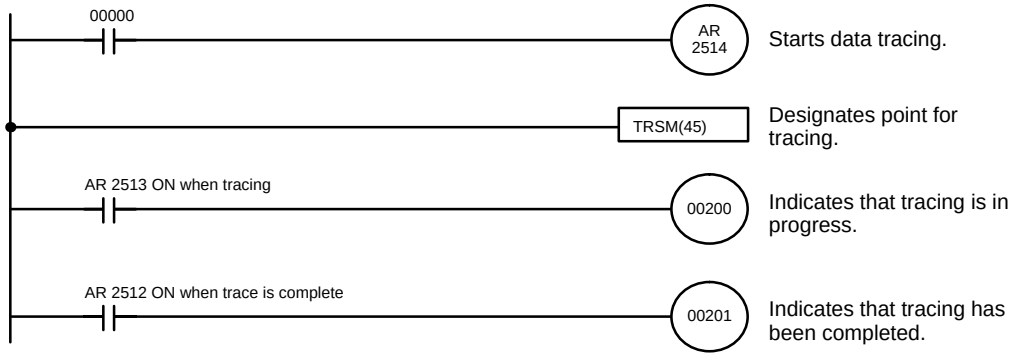
If TRSM(45) occurs TRSM(45) will not be executed within a JMP(08) – JME(09) block when the jump condition is OFF.

Example

The following example shows the basic program and operation for data tracing. Force set the Sampling Start Bit (AR 2515) to begin sampling. The Sampling Start Bit must not be turned ON from the program. The data is read and stored into trace memory.

When IR 00000 is ON, the Trace Start Bit (AR 2514) is also turned ON, and 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.

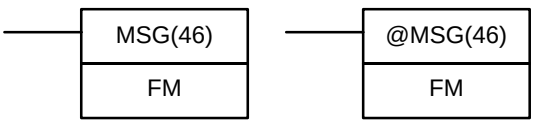


Address	Instruction	Operands	
00000	LD		0000
00001	OUT	AR	2514
00002	TRSM(45)		
00003	LD	AR	2513

Address	Instruction	Operands	
00004	OUT		00200
00005	LD	AR	2512
00006	OUT		00201

5-25-4 MESSAGE DISPLAY – MSG(46)

Ladder Symbols



Operand Data Areas

FM: First message word
IR, AR, DM, HR, LR

Limitations

FM and FM+7 must be in the same data area.

Description

When executed with an ON execution condition, MSG(46) reads eight words of extended ASCII code from FM to FM+7 and displays the message on the Programming Console. The displayed message can be up to 16 characters long, i.e., each ASCII character code requires eight bits (two digits). Refer to *Appendix I* for the extended ASCII codes. Japanese katakana characters are included in this code.

If not all eight 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.

Message Buffering and Priority

Up to three messages can be buffered in memory. Once stored in the buffer, they are displayed on a first in, first out basis. Since it is possible that more than three MSG(46)s may be executed within a single cycle, there is a priority scheme, based on the area where the messages are stored, for the selection of those messages to be buffered.

The priority of the data areas is as follows for message display:

LR > IR (I/O) > IR (not I/O) > HR > AR > TC > DM

In handling messages from the same area, those with the lowest address values have higher priority.

In handling indirectly addressed messages (i.e. ⁎DM), those with the lowest DM address values have higher priority.

Clearing Messages

To clear a message, execute FAL(06) 00 or clear it via a Programming Console using the procedure in *4-6-5 Clearing Error Messages*.

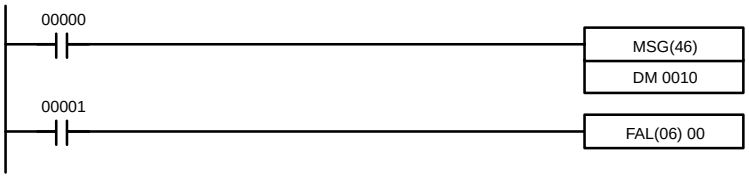
If the message data changes while the message is being displayed, the display will also change.

Flags

ER: Indirectly addressed DM word is non-existent. (Content of ⁎DM word is not BCD, or the DM area boundary has been exceeded.)

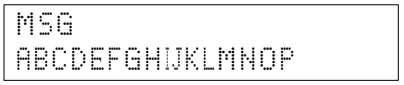
Example

The following example shows the display that would be produced for the instruction and data given when 00000 was ON. If 00001 goes ON, a message will be cleared.



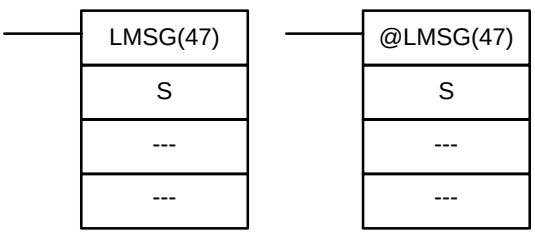
Address	Instruction	Operands
00000	LD	00000
00001	MSG(46)	
		DM 0010
00002	LD	00001
00003	FAL(06)	00

DM contents					ASCII equivalent	
DM 0010	4	1	4	2	A	B
DM 0011	4	3	4	4	C	D
DM 0012	4	5	4	6	E	F
DM 0013	4	7	4	8	G	H
DM 0014	4	9	4	A	I	J
DM 0015	4	B	4	C	K	L
DM 0016	4	D	4	E	M	N
DM 0017	4	F	5	0	O	P



5-25-5 LONG MESSAGE – LMSG(47)

Ladder Symbols



Operand Data Areas

S: First source word (ASCII)
IR, SR, AR, DM, HR, TC, LR
---: Not used.
Set to 000
---: Not used.
Set to 000

Limitations

S through S+15 must be in the same data area and must be in ASCII. The message will be truncated if a null character (00) is contained between S and S+15.

IR 298 and IR 299 cannot be used for S.

Description

LMSG(47) is used to output a 32-character message to a Programming Console. The message to be output must be in ASCII beginning in word S and ending in S+15, unless a shorter message is desired. A shorter message can be produced by placing a null character (00) into the string; no characters from the null character on will be output.

To output to the Programming Console, it must be set in TERMINAL mode. Although LMSG(47) will be executed as normal, the message will not appear correctly on the Programming Console unless TERMINAL mode is set. Refer to *5-25-6 TERMINAL MODE – TERM(48)* for details on switching to TERMINAL mode.

When pin 6 of the CPU's DIP switch is OFF, the Programming Console can be switched to TERMINAL mode by pressing the CHG Key or by executing TERM(48) in the program. When pin 6 of the CPU's DIP switch is ON, the Programming Console can be switched to Expansion TERMINAL mode by turning on bit AR 0709.

Flags

ER: S and S+15 are not in the same data area.

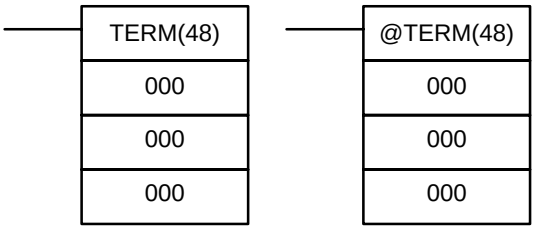
Indirectly addressed DM word is non-existent. (Content of *DM word is not BCD, or the DM area boundary has been exceeded.)

Example

Although the display is longer and there is a choice of output devices, the coding for LMSG(47) is the same as that for MSG(46). Refer to *Example* under the previous section for an example using MSG(46).

5-25-6 TERMINAL MODE – TERM(48)

Ladder Symbols



Description

When the execution condition is OFF, TERM(48) is not executed. When the execution condition is ON, TERM(48) switches the Programming Console to TERMINAL mode. (Instructions MSG(46), LMSG(47), and the keyboard mapping function are executed in TERMINAL mode.)

The Programming Console will return to CONSOLE mode when the CHG key is pressed again. There is no instruction that returns the Programming Console to CONSOLE mode from the program.

The Programming Console can also be switched to TERMINAL mode by pressing the CHG key on the Programming Console before inputting the password or when the mode is being displayed provided that pin 6 of the CPU's DIP switch is OFF. The Programming Console will return to CONSOLE mode when the CHG key is pressed again.

When pin 6 of the CPU's DIP switch is ON, the Programming Console can be switched to Expansion TERMINAL mode by turning on bit AR 0709.

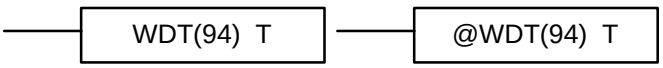
Example

In the following example, TERM(48) is used to switch the Programming Console to TERMINAL mode when 00000 is ON. Be sure that pin 6 of the CPU's DIP switch is OFF.

Address	Instruction	Operands
00000	LD	00000
00001	TERM(48)	
		000
		000
		000

5-25-7 WATCHDOG TIMER REFRESH – WDT(94)

Ladder Symbols



Definer Data Areas

T: Watchdog timer value
(00 to 63)

Description

When the execution condition is OFF, WDT(94) is not executed. When the execution condition is ON, WDT(94) extends the setting of the watchdog timer (normally set by the system to 130 ms) by 100 ms times T.

Timer extension = 100 ms x T.

Precautions

If the cycle time is longer than the time set for the watchdog timer, 9F will be output to the FAL area and the CPU will stop.

If the cycle time exceeds 6,500 ms, a FALS 9F will be generated and the system will stop.

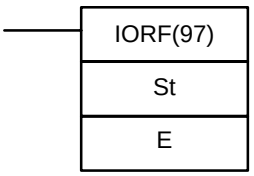
Timers might not function properly when the cycle time exceeds 100 ms. When using WDT(94), the same timer should be repeated in the program at intervals that are less than 100 ms apart. TIMH(15) should be used only in a scheduled interrupt routine executed at intervals of 10 ms or less.

Flags

There are no flags affected by this instruction.

5-25-8 I/O REFRESH – IORF(97)

Ladder Symbol



Operand Data Areas

St: Starting word
IR 000 to IR 049

E: End word
IR 000 to IR 049

Limitations

IORF(97) can be used to refresh I/O words allocated to only I/O Units (IR 000 to IR 039) and Special I/O Units (IR 100 to IR 199) mounted to the CPU or Expansion I/O Racks. It cannot be used for other I/O words, such as I/O Units on Slaves Racks or Group-2 High-density I/O Units.

St must be less than or equal to E.

Description

To refresh I/O words allocated to CPU or Expansion I/O Racks (IR 000 to IR 030), simply indicate the first (St) and last (E) I/O words to be refreshed. When the execution condition for IORF(97) 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 cycle.

To refresh I/O words allocated to Special I/O Units (IR 100 to IR 199), indicate the unit numbers of the Units by designating IR 040 to IR 049 (see note). IR 040 to IR 049 correspond to Special I/O Units 0 to 9. For example, set St to IR 043 and E to IR 045 to refresh the I/O words allocated to Special I/O Units 3, 4, and 5. The I/O words allocated to those Units (IR 130 to IR 159) will be refreshed when IORF(97) is executed.This will be in addition to the normal I/O refresh performed during the CPU's cycle.

Note I/O will not be refreshed if IORF(97) is executed for any of the following Units:

- Group-2 High-density I/O Units
- I/O Units or Special I/O Units mounted to Slave Racks (in Remote I/O Systems)
- Optical I/O Units (in Remote I/O Systems)

Refer to *5-25-9 GROUP-2 HIGH-DENSITY I/O REFRESH – MPRF(61)* for details on refreshing words allocated to Group-2 High-density I/O Units.

Execution Time

When Basic I/O Units are specified, the execution time for IORF(97) is computed as follows:

T_{IORF} = instruction execution time + Input Unit I/O refresh time + Output Unit I/O refresh time

= 0.13 ms + 0.02 ms × (no. of 8-pt Units + no. of 16-pt Units × 2) + 0.02 ms × (no. of 5- and 8-pt Units + no. of 12-/16-pt Units × 2)

When Special I/O Units are specified, the execution time for IORF(97) is computed as follows:

T_{IORF} = instruction execution time + ∑(Special I/O Unit I/O refresh times)

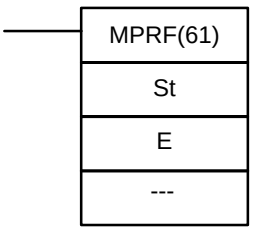
The instruction execution time is 0.1 ms. Refer to *6-1 Cycle Time* for I/O refresh times for Special I/O Units.

Flags

There are no flags affected by this instruction.

5-25-9 GROUP-2 HIGH-DENSITY I/O REFRESH – MPRF(61)

Ladder Symbol



Operand Data Areas

St: Starting Unit
#0000 to #0009

E: End Unit
#0000 to #0009

Limitations

MPRF(61) can be used to refresh I/O words allocated to Group-2 High-density I/O Units (IR 030 to IR 049) only. It cannot be used for other I/O words.

St and E must be between #0000 and #0009. St must be less than or equal to E.

Description

When the execution condition is OFF, MPRF(61) is not executed. When the execution condition is ON, the I/O words allocated to Group-2 High-density I/O Units with I/O numbers St through E will be refreshed.This will be in addition to the normal I/O refresh performed during the CPU's cycle.

It is not possible to specify the I/O words by address, only by the I/O number of the Unit to which they are allocated.

Execution Time

The execution time for MPRF(61) is computed as follows:

T_{MPRF} = Instruction execution time + ∑(Group-2 High-density I/O Unit I/O refresh times)

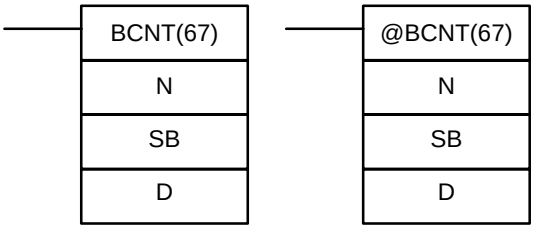
Refer to *6-1 Cycle Time* for a table showing I/O refresh times for Group-2 High-density I/O Units.

Flags

ER: St or E is not BCD between #0000 and #0009.
St is greater than E.

5-25-10 BIT COUNTER – BCNT(67)

Ladder Symbols



Operand Data Areas

N: Number of words (BCD)
IR, SR, AR, DM, HR, TC, LR, #
SB: Source beginning word
IR, SR, AR, DM, HR, TC, LR
D: Destination word
IR, SR, AR, DM, HR, TC, LR

Limitations

N must be BCD between 0000 and 6656.

Description

When the execution condition is OFF, BCNT(67) is not executed. When the execution condition is ON, BCNT(67) counts the total number of bits that are ON in all words between SB and SB+(N–1) and places the result in D.

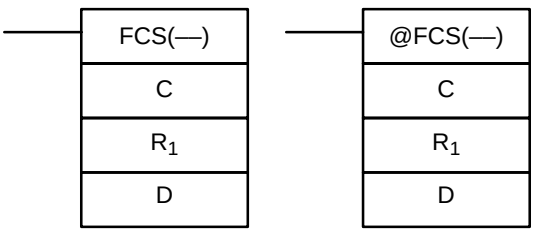
Flags

ER: N is not BCD, or N is 0; SB and SB+(N–1) are not in the same area.
The resulting count value exceeds 9999.
Indirectly addressed DM word is non-existent. (Content of ✳DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON when the result is 0.

5-25-11 FRAME CHECKSUM – FCS(––)

Ladder Symbols



Operand Data Areas

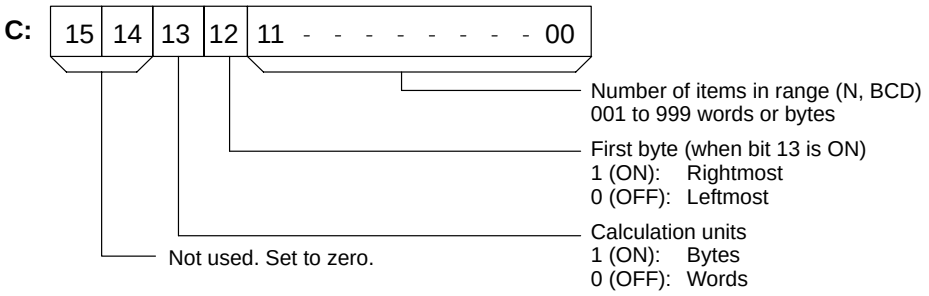
C: Control data
IR, SR, AR, DM, HR, LR
R_1: First word in range
IR, SR, AR, DM, HR, TC, LR
D: First destination word
IR, SR, AR, DM, HR, LR

Description

FCS(—) can be used to check for errors when transferring data through communications ports.

When the execution condition is OFF, FCS(—) is not executed. When the execution condition is ON, FCS(—) calculates the frame checksum of the specified range by exclusively ORing either the contents of words R_1 to R_1+N–1 or the bytes in words R_1 to R_1+N–1. The frame checksum value (hexadecimal) is then converted to ASCII and output to the destination words (D and D+1).

The function of bits in C are shown in the following diagram and explained in more detail below.



Number of Items in Range

The number of items within the range (N) is contained in the 3 rightmost digits of C, which must be BCD between 001 and 999.

Calculation Units

The frame checksum of words will be calculated if bit 13 is OFF and the frame checksum of bytes will be calculated if bit 13 is ON.

If bytes are specified, the range can begin with the leftmost or rightmost byte of R_1. The leftmost byte of R_1 will not be included if bit 12 is ON.

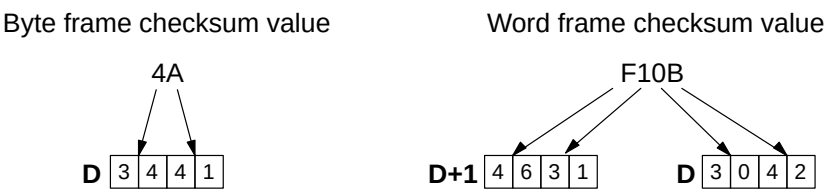
	MSB	LSB
R_1	1	2
R_1+1	3	4
R_1+2	5	6
R_1+3	7	8
⋮	⋮	⋮

When bit 12 is OFF the bytes will be ORed in this order: 1, 2, 3, 4, ...

When bit 12 is ON the bytes will be ORed in this order: 2, 3, 4, 5, ...

Conversion to ASCII

The byte frame checksum calculation yields a 2-digit hexadecimal value which is converted to its 4-digit ASCII equivalent. The word frame checksum calculation yields a 4-digit hexadecimal value which is converted to its 8-digit ASCII equivalent, as shown below.



Flags

ER: Indirectly addressed DM word is non-existent. (Content of ⁜DM word is not BCD, or the DM area boundary has been exceeded.)

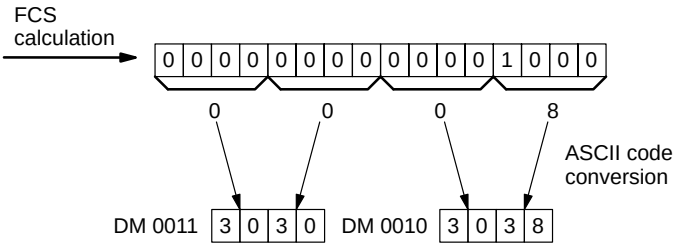
The number of items is not 001 to 999 BCD.

Example

When IR 00000 is ON in the following example, the frame checksum (0008) is calculated for the 8 words from DM 0000 to DM 0007 and the ASCII equivalent (30 30 30 38) is written to DM 0011 and DM 0010.

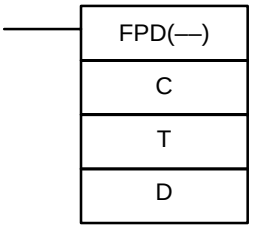
Address	Instruction	Operands
00000	LD	00000
00001	@FCS(—)	
		# 0008
		DM 0000
		DM 0010

DM 0000	0001
DM 0001	0002
DM 0002	0003
DM 0003	0004
DM 0004	0005
DM 0005	0006
DM 0006	0007
DM 0007	0008



5-25-12 FAILURE POINT DETECTION – FPD(––)

Ladder Symbols



Operand Data Areas

C: Control data
#

T: Monitoring time (BCD)
IR, SR, AR, DM, HR, TC. LR, #

D: First register word
IR, AR, DM, HR, LR

Limitations

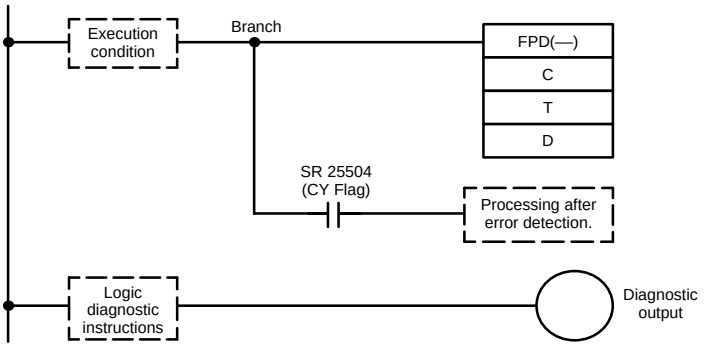
D and D+8 must be in the same data area when bit 15 of C is ON.

C must be input as a constant.

Description

FPD(—) can be used in the program as many times as desired, but each must use a different D. It is used to monitor the time between the execution of FPD(—) and the execution of a diagnostic output. If the time exceeds T, an FAL(06) non-fatal error will be generated with the FAL number specified in C.

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.



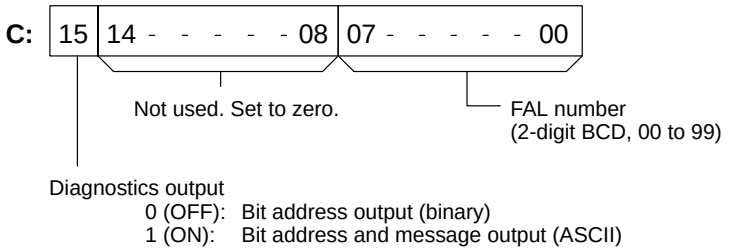
When the execution condition is OFF, FPD(—) is not executed. When the execution condition is ON, FPD(—) monitors the time until the logic diagnostics condition goes ON, turning ON the diagnostic output. If this time exceeds T, the following will occur:

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 logic diagnostic instructions are searched for the first OFF input condition and this condition's bit address is output to the destination words beginning at D.
3. The CY Flag (SR 25504) is turned ON. An error processing program section can be executed using the CY Flag if desired.
4. 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.

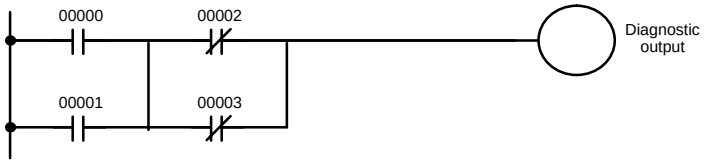
Control Data

The function of the control data bits in C are shown in the following diagram.



Logic Diagnostic Instructions

If the time until the logic diagnostics condition goes ON exceeds T, the logic diagnostic instructions are searched for the OFF input condition. 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, the normally closed condition IR 00002 would be found as the cause of the diagnostic output not turning ON.

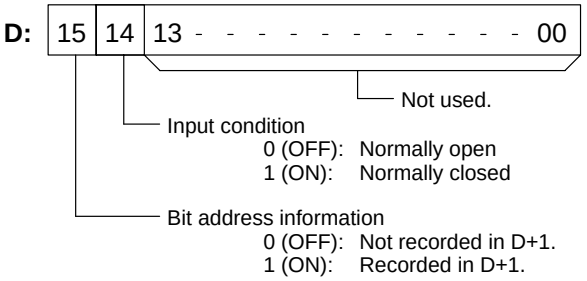
Diagnostics Output

There are two ways to output the bit address of the OFF condition detected in the logic diagnostics condition.

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 bit address code of the input condition, as shown below. The word addresses, bit numbers, and TC numbers are in binary.

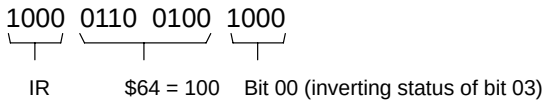
Data Area	D+1 bit status															
	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
IR, SR (see note c)	1	0	0	0	Word address								Bit number			
	1	0	1	0	Word address								Bit number			
HR	1	0	0	1	1	Word address							Bit number			
LR	1	0	0	1	0	0	Word address						Bit number			
TC*	1	0	0	1	0	1	*	Timer or counter number								

Note a) *For the TC area, bit 09 of D+1 indicates whether the number is a timer or counter. A 0 indicates a timer, and a 1 indicates a counter.

b) The status of the leftmost bit of the bit number (bit 03) is reversed.

c) Although the same word address designations are used for both ranges, bit 13 is turned OFF to indicate IR 00000 through SR 25515 and turned ON to indicate SR 25600 through IR 51115

Example: If D + 1 contains 1000 0110 0100 1000, IR 10000 would be indicated as follows:



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+5 to D+8 contain information in ASCII that are displayed on a Peripheral Device along with the bit address when FPD(—) 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 = “–”	“0”=normally open, “1”=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.

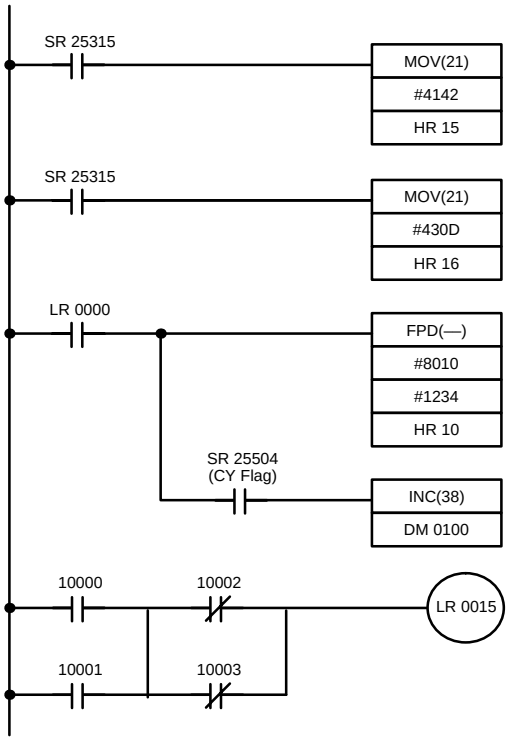
Determining Monitoring Time The procedure below can be used to automatically set the monitoring time, T, under actual operating conditions when specifying a word operand for T. This operation cannot be used if a constant is set for T.

1, 2, 3...

1. Switch the C200HS to MONITOR Mode operation.
2. Connect a Peripheral Device, such as a Programming Console.
3. Use the Peripheral Device to turn ON control bit AR 2508.
4. Execute the program with AR 2508 turned ON. If the monitoring time currently in T is exceeded, 1.5 times the actual monitoring time will be stored in T. FAL(06) errors will not occur while AR 2508 is ON.
5. Turn OFF AR 2508 when an acceptable value has been stored in T.

Example

In the following example, the FPD(—) is set to display the bit address and message (“ABC”) when a monitoring time of 123.4 s is exceeded.



Address	Instruction	Operands	
00000	LD		25315
00001	MOV(21)		
		#	4142
		HR	15
00002	LD		25315
00003	MOV(21)		
		#	430D
		HR	16
00004	LD	LR	0000
00005	FPD(—)		
		#	8010
		#	1234
		HR	10
00006	AND		25504
00007	INC(38)		
		DM	0100
00008	LD		10000
00009	OR		10001
00010	LD NOT		10002
00011	OR NOT		10003
00012	AND LD		
00013	OUT	LR	0015

FPD(—) is executed and begins monitoring when LR 0000 goes ON. If LR 0015 does not turn ON within 123.4 s and IR 10000 through IR 10003 are all ON, IR 10002 will be selected as the cause of the error, an FAL(06) error will be generated with an FAL number of 10, and the bit address and preset message (“10002–1ABC”) will be displayed on the Peripheral Device.

HR 10	0000
HR 11	0000
HR 12	0000
HR 13	0000
HR 14	0000
HR 15	4142
HR 16	430D
HR 17	0000
HR 18	0000

→

HR 10	C000	Indicates information, normally closed condition
HR 11	2031	“1”
HR 12	3030	“00”
HR 13	3032	“02”
HR 14	2D31	“–1”
HR 15	4142	“AB”
HR 16	430D	“C”, and CR code
HR 17	0000	The last two words are ignored.
HR 18	0000	(Displayed as spaces.)

Flags

ER: T is not BCD.

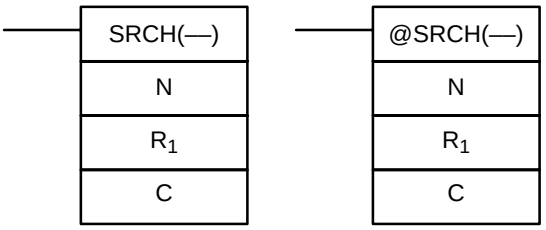
C is not a constant or the rightmost two digits of C are not BCD 00 to 99.

Indirectly addressed DM word is non-existent. (Content of ⁜DM word is not BCD, or the DM area boundary has been exceeded.)

CY: ON when the time between the execution of FPD(—) and the execution of a diagnostic output exceeds T.

5-25-13 DATA SEARCH – SRCH(–)

Ladder Symbols



Operand Data Areas

N: Number of words
IR, SR, AR, DM, HR, TC, LR, #
R_1: First word in range
IR, SR, AR, DM, HR, TC, LR
C: Comparison data, result word
IR, SR, AR, DM, HR, LR

Limitations

N must be BCD between 0001 to 6656.

R_1 and R_1+N–1 must be in the same data area.

Description

When the execution condition is OFF, SRCH(—) is not executed. When the execution condition is ON, SRCH(—) searches the range of memory from R_1 to R_1+N–1 for addresses that contain the comparison data in C. If one or more addresses contain the comparison data, the EQ Flag (SR 25506) is turned ON and the lowest address containing the comparison data is identified in C+1. The address is identified differently for the DM area:

1, 2, 3...

1. For an address in the DM area, the word address is written to C+1. For example, if the lowest address containing the comparison data is DM 0114, then #0114 is written in C+1.
2. For an address in another data area, the number of addresses from the beginning of the search is written to C+1. For example, if the lowest address containing the comparison data is IR 114 and the first word in the search range is IR 014, then #0100 is written in C+1.

If none of addresses in the range contain the comparison data, the EQ Flag (SR 25506) is turned OFF and C+1 is left unchanged.

Flags

ER: Indirectly addressed DM word is non-existent. (Content of ✳DM word is not BCD, or the DM area boundary has been exceeded.)

N is not BCD between 0001 and 6655.

EQ: ON when the comparison data has been matched in the search range.

Example

In the following example, the 10 word range from DM 0010 to DM 0019 is searched for addresses that contain the same data as DM 0000 (#FFFF). Since DM 0012 contains the same data, the EQ Flag (SR 25506) is turned ON and #0012 is written to DM 0001.

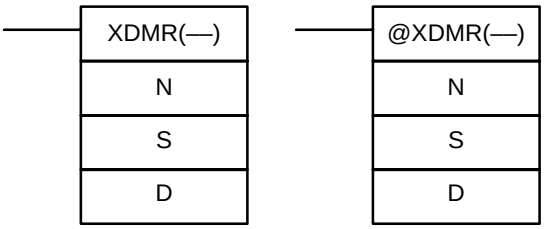
Address	Instruction	Operands
00000	LD	00001
00001	@SRCH(—)	
		# 0010
		DM 0010
		DM 0000

DM 0010	0000
DM 0011	9898
DM 0012	FFFF
DM 0013	9797
DM 0014	AAAA
DM 0015	9595
DM 0016	1414
DM 0017	0000
DM 0018	0000
DM 0019	FFFF

DM 0000	FFFF
DM 0001	0012

5-25-14 EXPANSION DM READ – XDMR(––)

Ladder Symbols



Operand Data Areas

N: Number of words
IR, SR, AR, DM, HR, TC, LR, #
S: First expansion DM word
IR, SR, AR, DM, HR, TC, LR, #
D: First destination word
IR, SR, AR, DM, HR, LR

Limitations

N must be BCD between 0001 and 3000.

S must be BCD between 7000 and 9999.

S and S+N–1 must be in the same data area, as must D and D+N–1.

Description

When the execution condition is OFF, XDMR(—) is not executed. When the execution condition is ON, XDMR(—) copies the contents of expansion DM words S through S+N–1 to the destination words D through D+N–1.

Precautions

The Expansion DM area must be set in the PC Setup before it can be used in programming. Do not exceed the set range of the Expansion DM area.

Execution of XDMR(—) is given priority whenever a power interruption occurs.

Flags

ER: The specified expansion DM words are non-existent. Make sure that the specified words have been allocated to expansion DM. Refer to *7-1-15 UM Area Allocation* for details.

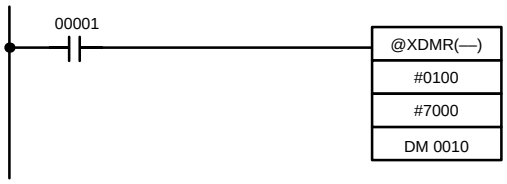
Indirectly addressed DM word is non-existent. (Content of *DM word is not BCD, or the DM area boundary has been exceeded.)

N is not BCD between 0001 and 3000.

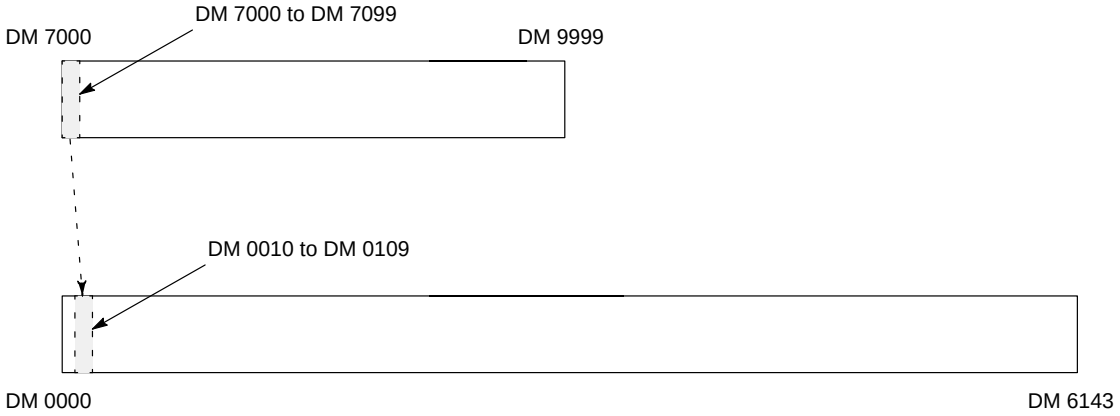
S is not BCD between 7000 and 9999.

Example

In the following example, the 100 word range from DM 7000 through DM 7099 is copied to DM 0010 through DM 0109 when IR 00001 is ON.



Address	Instruction	Operands
00000	LD	00001
00001	@XDMR(—)	
		# 0100
		# 7000
		DM 0010

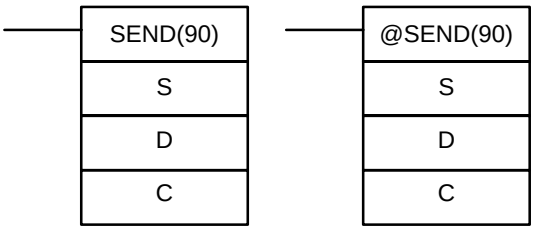


5-26 Network Instructions

The SYSMAC NET Link/SYSMAC LINK instructions are used for communicating with other PCs linked through the SYSMAC NET Link System or SYSMAC LINK System. These instructions are applicable to the C200H-CPU31-E and CPU33-E only.

5-26-1 NETWORK SEND – SEND(90)

Ladder Symbols



Operand Data Areas

S: Source beginning word
IR, SR, AR, DM, HR, TC, LR
D: Destination beginning word
IR, AR, DM, HR, TC, LR
C: First control data word
IR, AR, DM, HR, TC, LR

Limitations

Can be performed with the CPU31-E/33-E only. C through C+2 must be within the same data area and must be within the values specified below. To be able to use SEND(90), the system must have a SYSMAC NET Link or SYSMAC LINK Unit mounted.

Description

When the execution condition is OFF, SEND(90) is not executed. When the execution condition is ON, SEND(90) transfers data beginning at word S, to addresses specified by D in the designated node on the SYSMAC NET Link/SYSMAC LINK System. The control words, beginning with C, specify the number of words to be sent, the destination node, and other parameters. The contents of the control data depends on whether a transmission is being sent in a SYSMAC NET Link System or a SYSMAC LINK System.

The status of bit 15 of C+1 determines whether the instruction is for a SYSMAC NET Link System or a SYSMAC LINK System.

Control Data

SYSMAC NET Link Systems

The destination port number is always set to 0. Set the destination node number to 0 to send the data to all nodes. Set the network number to 0 to send data to a node on the same Subsystem (i.e., network). Refer to the *SYSMAC NET Link System Manual* for details.

Word	Bits 00 to 07	Bits 08 to 15
C	Number of words (0 to 1000 in 4-digit hexadecimal, i.e., 0000_{hex} to $03E8_{hex}$)	
C+1	Network number (0 to 127 in 2-digit hexadecimal, i.e., 00_{hex} to $7F_{hex}$)	Bit 14 ON: Operating level 0 OFF: Operating level 1 Bits 08 to 13 and 15: Set to 0.
C+2	Destination node (0 to 126 in 2-digit hexadecimal, i.e., 00_{hex} to $7E_{hex}$)*	Destination port NSB: 00 NSU: 01/02

*The node number of the PC executing the send may be set.

SYSMAC LINK Systems

Set the destination node number to 0 to send the data to all nodes. Refer to the *SYSMAC LINK System Manual* for details.

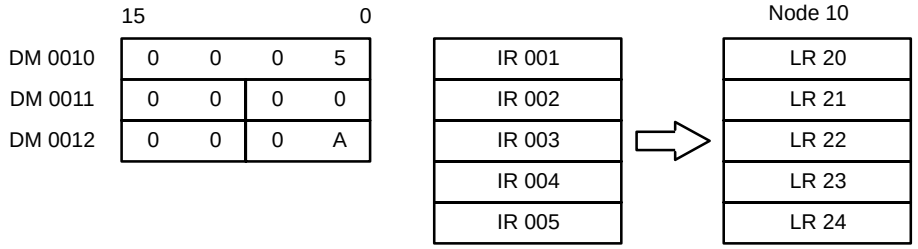
Word	Bits 00 to 07	Bits 08 to 15
C	Number of words (0 to 256 in 4-digit hexadecimal, i.e., 0000_{hex} to 0100_{hex})	
C+1	Response time limit (0.1 and 25.4 seconds in 2-digit hexadecimal without decimal point, i.e., 00_{hex} to FF_{hex}) **Note:** The response time will be 2 seconds if the limit is set to 0_{hex}. There will be no time limit if the time limit is set to FF_{hex}.	Bits 08 to 11: No. of retries (0 to 15 in hexadecimal, i.e., 0_{hex} to F_{hex}) Bit 12: Set to 0. Bit 13 ON: Response not returned. OFF: Response returned. Bit 14 ON: Operating level 0 OFF: Operating level 1 Bit 15: Set to 1.
C+2	Destination node (0 to 62 in 2-digit hexadecimal, i.e., 00_{hex} to $3E_{hex}$)*	Set to 0.

*The node number of the PC executing the send cannot be set.

Examples

This example is for a SYSMAC NET Link System. When 00000 is ON, the following program transfers the content of IR 001 through IR 005 to LR 20 through LR 24 on node 10.

Address	Instruction	Operands
00000	LD	00000
00001	SEND(90)	
		001
		LR 20
		DM 0010



Flags

ER: The specified node number is greater than 126 in a SYSMAC NET Link System or greater than 62 in a SYSMAC LINK System.

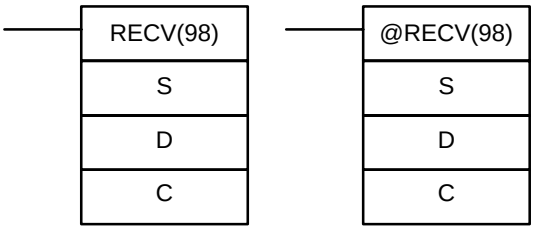
The sent data overruns the data area boundaries.

Indirectly addressed DM word is non-existent. (Content of ⁎DM word is not BCD, or the DM area boundary has been exceeded.)

There is no SYSMAC NET Link/SYSMAC LINK Unit.

5-26-2 NETWORK RECEIVE – RECV(98)

Ladder Symbols



Operand Data Areas

S: Source beginning word
IR, SR, AR, DM, HR, TC, LR
D: Destination beginning word
IR, AR, DM, HR, TC, LR
C: First control data word
IR, AR, DM, HR, TC, LR

Limitations

Can be performed with the CPU31-E/33-E only. C through C+2 must be within the same data area and must be within the values specified below. To be able to use RECV(98), the system must have a SYSMAC NET Link or SYSMAC LINK Unit mounted.

Description

When the execution condition is OFF, RECV(98) is not executed. When the execution condition is ON, RECV(98) transfers data beginning at S from a node on the SYSMAC NET Link/SYSMAC LINK System to words beginning at D. The control words, beginning with C, provide the number of words to be received, the source node, and other transfer parameters.

The status of bit 15 of C+1 determines whether the instruction is for a SYSMAC NET Link System or a SYSMAC LINK System.

Control Data

SYSMAC NET Link Systems

The source port number is always set to 0. Set the network number to 0 to receive data to a node on the same Subsystem (i.e., network). Refer to the *SYSMAC NET Link System Manual* for details.

Word	Bits 00 to 07	Bits 08 to 15
C	Number of words (0 to 1000 in 4-digit hexadecimal, i.e., 0000_{hex} to $03E8_{hex}$)	
C+1	Network number (0 to 127 in 2-digit hexadecimal, i.e., 00_{hex} to $7F_{hex}$)	Bit 14 ON: Operating level 0 OFF: Operating level 1 Bits 08 to 13 and 15: Set to 0.
C+2	Source node (1 to 126 in 2-digit hexadecimal, i.e., 01_{hex} to $7E_{hex}$)	Source port NSB: 00 NSU: 01/02

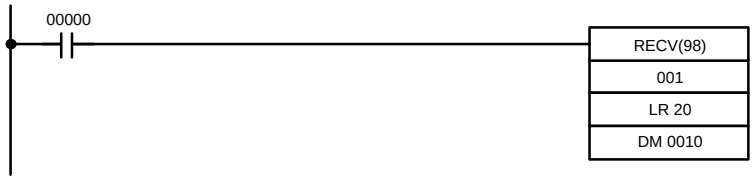
SYSMAC LINK Systems

Refer to the *SYSMAC LINK System Manual* for details.

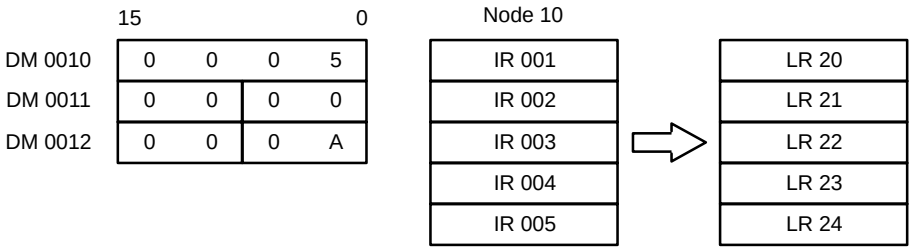
Word	Bits 00 to 07	Bits 08 to 15
C	Number of words (0 to 256 in 4-digit hexadecimal, i.e., 0000_{hex} to 0100_{hex})	
C+1	Response time limit (0.1 and 25.4 seconds in 2-digit hexadecimal without decimal point, i.e., 00_{hex} to FF_{hex}) **Note:** The response time will be 2 seconds if the limit is set to 0_{hex}. There will be no time limit if the time limit is set to FF_{hex}.	Bits 08 to 11: No. of retries (0 to 15 in hexadecimal, i.e., 0_{hex} to F_{hex}) Bit 12: Set to 0. Bit 13: Set to 0. Bit 14 ON: Operating level 0 OFF: Operating level 1 Bit 15: Set to 1.
C+2	Source node (0 to 62 in 2-digit hexadecimal, i.e., 00_{hex} to $3E_{hex}$)	Set to 0.

Examples

This example is for a SYSMAC NET Link System. When 00000 is ON, the following program transfers the content of IR 001 through IR 005 to LR 20 through LR 24 on node 10.



Address	Instruction	Operands
00000	LD	00000
00001	RECV(98)	
		001
		LR 20
		DM 0010



Flags

ER: The specified node number is greater than 126 in a SYSMAC NET Link System or greater than 62 in a SYSMAC LINK System.

The received data overflows the data area boundaries.

Indirectly addressed DM word is non-existent. (Content of ⁜DM word is not BCD, or the DM area boundary has been exceeded.)

There is no SYSMAC NET Link/SYSMAC LINK Unit.

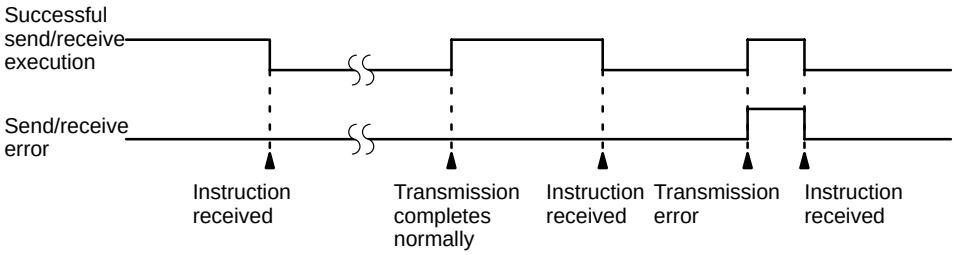
5-26-3 About Network Communications

SEND(90) and RECV(98) 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. Note that the SEND(90)/RECV(98) Enable Flag is not turned ON until the first END(01) after the transmission is completed. Refer to the *SYSMAC NET Link System Manual* or *SYSMAC LINK System Manual* for details about command/response operations.

If multiple SEND(90)/RECV(98) operations are used, the following flags must be used to ensure that any previous operation has completed before attempting further send/receive SEND(90)/RECV(98) operations

SR Flag	Functions
SEND(90)/RECV(98) Enable Flags (SR 25201, SR 25204)	OFF during SEND(90)/RECV(98) execution (including command response processing). Do not start a SEND(90)/RECV(98) operation unless this flag is ON.
SEND(90)/RECV(98) Error Flags (SR 25200, SR 25203)	OFF following normal completion of SEND/RECV (i.e., after reception of response signal) ON after an unsuccessful SEND(90)/RECV(98) attempt. Error status is maintained until the next SEND(90)/RECV(98) operation. Error types: Time-out error (command/response time greater than 1 second) Transmission data errors

Timing

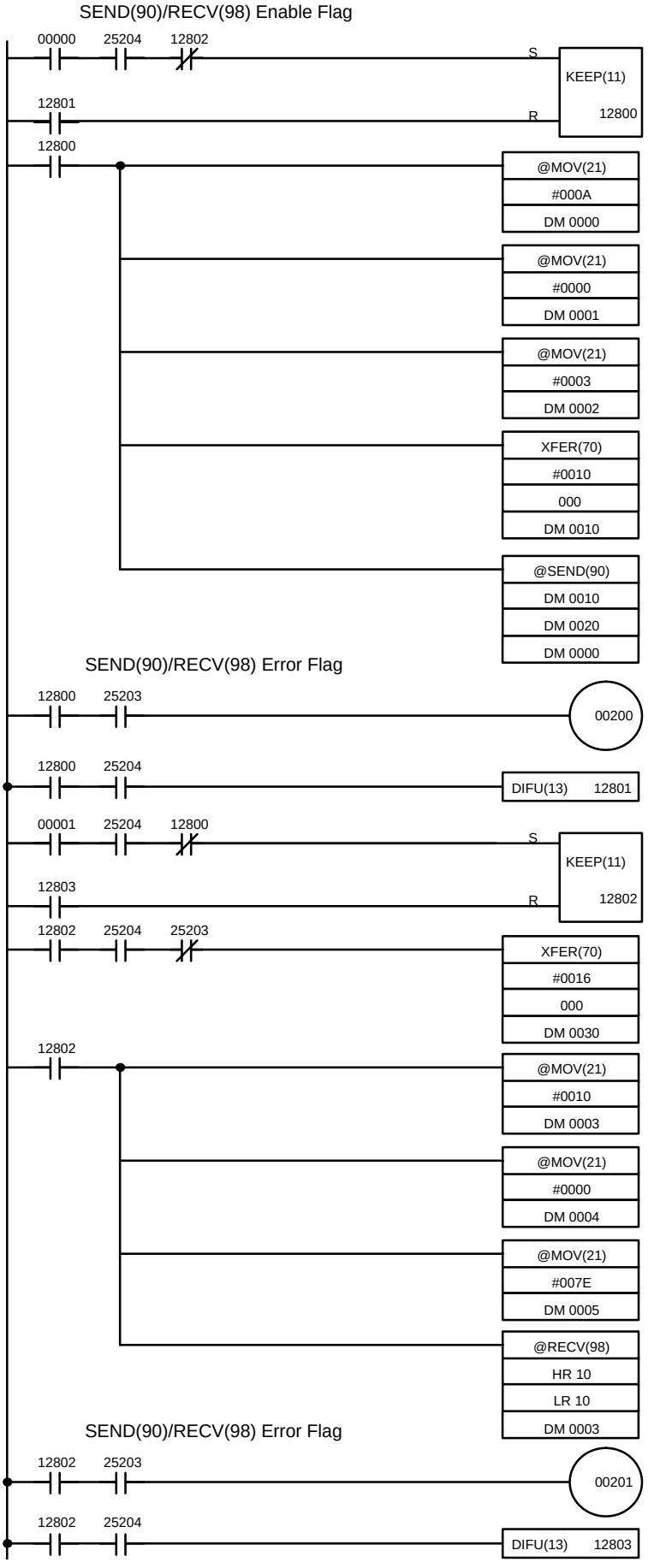


Data Processing for SEND(90)/RECV(98)

Data is transmitted for SEND(90) and RECV(98) for all PCs when SEND(90)/RECV(98) is executed. Final processing for transmissions/receptions is performed during servicing of peripheral devices and Link Units.

Programming Example: Multiple SEND(90)/RECV(98)

To ensure successful SEND(90)/RECV(98) operations, your program must use the SEND(90)/RECV(98) Enable Flags and SEND(90)/RECV(98) Error Flags to confirm that execution is possible. The following program shows one example of how to do this for a SYSMAC NET Link System.



12800 prevents execution of SEND(90) until RECV(98) (below) has completed. IR 00000 is turned ON to start transmission.

Data is placed into control data words to specify the 10 words to be transmitted to node 3 in operating level 1 of network 00 (NSB).

Turns ON to indicate transmission error.

Resets 12800, above.

12802 prevents execution of RECV(98) when SEND(90) above has not completed. IR 00001 is turned ON to start transmission.

Transmitted data moved into words beginning at DM 0030 for storage.

Data moved into control data words to specify the 16 words to be transmitted from node 126 in operating level 1 of network 00 (NSB).

Turns ON to indicate reception error.

Resets 12802, above.

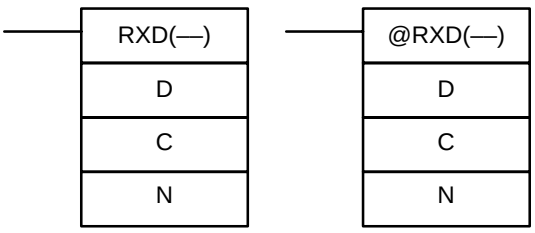
Address	Instruction	Operands	
00000	LD		00000
00001	AND		25204
00002	AND NOT		12802
00003	LD		12801
00004	KEEP(11)		12800
00005	LD		12800
00006	@MOV(21)		
		#	000A
		DM	0000
00007	@MOV(21)		
		#	0000
		DM	0001
00008	@MOV(21)		
		#	0003
		DM	00002
00009	@XFER(70)		
		#	0010
			000
		DM	0002
00010	@SEND(90)		
		DM	0010
		DM	0020
		DM	0000
00011	LD		12800
00012	AND		25203
00013	OUT		00200
00014	LD		12800
00015	AND		25204
00016	DIFU(13)		12801
00017	LD		00001
00018	AND		25204

Address	Instruction	Operands	
00019	AND NOT		12800
00020	LD		12803
00021	KEEP(11)		12802
00022	LD		12802
00023	AND		25204
00024	AND NOT		25203
00025	XFER(70)		
		#	0016
			000
		DM	0030
00026	LD		12802
00027	@MOV(21)		
		#	0010
		DM	0003
00028	@MOV(21)		
		#	0000
		DM	0004
00029	@MOV(21)		
		#	007E
		DM	0005
00030	@RECV(98)		
		HR	10
		LR	10
		DM	0003
00031	LD		12802
00032	AND		25203
00033	OUT		00201
00034	LD		12802
00035	AND		25204
00036	DIFU(13)		12803

5-27 Serial Communications Instructions

5-27-1 RECEIVE – RXD(––)

Ladder Symbols



Operand Data Areas

D: First destination word
IR, SR, AR, DM, HR, TC, LR

C: Control word
IR, SR, AR, DM, HR, TC, LR, #

N: Number of bytes
IR, SR, AR, DM, HR, TC, LR, #

Limitations

D and D+(N÷2)–1 must be in the same data area.

N must be BCD from #0000 to #0256. (#0000 to #0061 in host link mode)

Description

When the execution condition is OFF, RXD(—) is not executed. When the execution condition is ON, RXD(—) reads N bytes of data received at the peripheral port, and then writes that data in words D to D+(N÷2)–1. Up to 256 bytes of data can be read at one time.

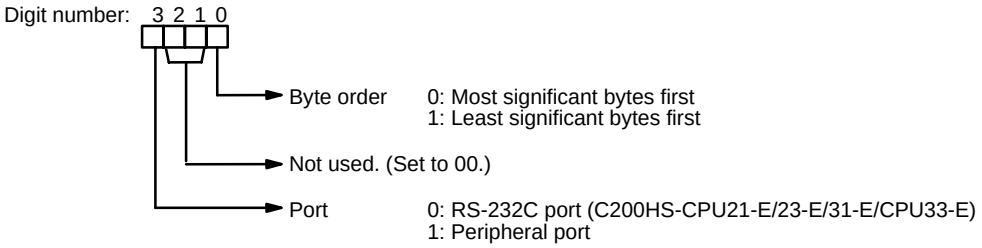
If fewer than N bytes are received, the amount received will be read.

Note RXD(—) is required to receive data via the peripheral port or RS-232C port only. Transmission sent from a host computer to a Host Link Unit are processed automatically and do not need to be programmed.

⚠ Caution The PC will be incapable of receiving more data once 256 bytes have been received if received data is not read using RXD(—). Read data as soon as possible after the Reception Completed Flag is turned ON (SR 26414 for peripheral port).

Control Word

The value of the control word determines the port from which data will be read and the order in which data will be written to memory.



The order in which data is written to memory depends on the value of digit 0 of C. Eight bytes of data 12345678... will be written in the following manner:

Digit 0 = 0

	MSB	LSB
D	1	2
D+1	3	4
D+2	5	6
D+3	7	8
⋮	⋮	⋮

Digit 0 = 1

	MSB	LSB
D	2	1
D+1	4	3
D+2	6	5
D+3	8	7
⋮	⋮	⋮

Flags

ER: The CPU is not equipped with an RS-232C port.

Another device is not connected to the specified port.

There is an error in the communications settings (PC Setup) or the operand settings.

Indirectly addressed DM word is non-existent. (Content of *DM word is not BCD, or the DM area boundary has been exceeded.)

The destination words (D to D+(N÷2)–1) exceed the data area.

Peripheral Port

26414: SR 26414 will be turned ON when data has been received normally at the peripheral port and will be reset when the data is read using RXD(—) is executed.

266: SR 266 contains the number of bytes received at the peripheral port and is reset to 0000 when RXD(—) is executed.

RS-232C Port

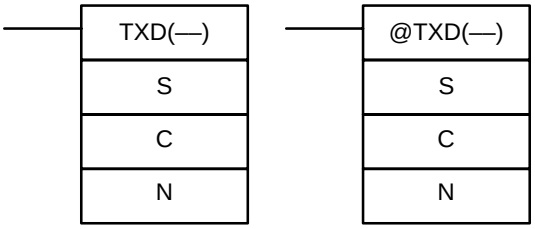
26406: SR 26406 will be turned ON when data has been received normally at the peripheral port and will be reset when the data is read using RXD(—) is executed.

265: SR 265 contains the number of bytes received at the RS-232C port and is reset to 0000 when RXD(—) is executed.

Note Communications flags and counters can be cleared either by specifying 0000 for N or using the Port Reset Bit (SR 25208 for peripheral port and SR 25209 for RS-232C port).

5-27-2 TRANSMIT – TXD(––)

Ladder Symbols



Operand Data Areas

S: First source word
IR, SR, AR, DM, HR, TC, LR

C: Control word
IR, SR, AR, DM, HR, TC, LR, #

N: Number of bytes
IR, SR, AR, DM, HR, TC, LR, #

Limitations

S and S+(N÷2)–1 must be in the same data area.

N must be BCD from #0000 to #0256. (#0000 to #0061 in host link mode)

Description

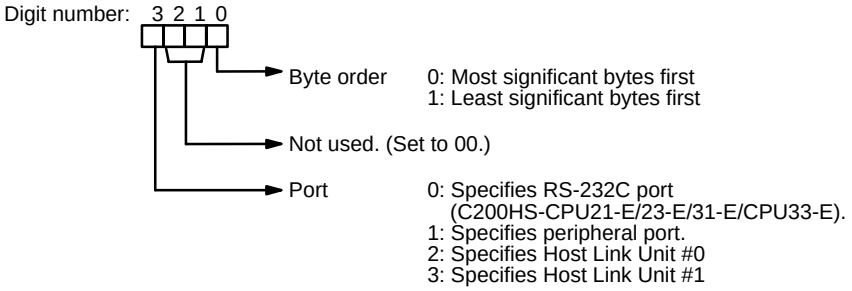
When the execution condition is OFF, TXD(—) is not executed. When the execution condition is ON, TXD(—) reads N bytes of data from words S to S+(N÷2)–1, converts it to ASCII, and outputs the data from the specified port. TXD(—) operates differently in host link mode and RS-232C mode, so these modes are described separately.

Note The following flags will be ON to indicate that communications are possible through the various ports. Be sure the corresponding flag is ON before executing TXD(—).

SR 26405: RS-232C port
SR 26413: Peripheral port
SR 26705: Host Link Unit #0
SR 26713: Host Link Unit #1

Host Link Mode

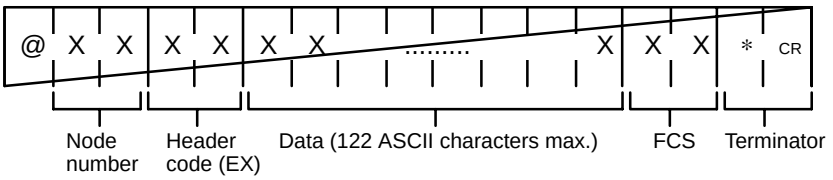
N must be BCD from #0000 to #0061 (i.e., up to 122 bytes of ASCII). The value of the control word determines the port from which data will be output, as shown below.



The specified number of bytes will be read from S through S+(N/2)–1, converted to ASCII, and transmitted through the specified port. The bytes of source data shown below will be transmitted in this order: 12345678...

	MSB	LSB
S	1	2
S+1	3	4
S+2	5	6
S+3	7	8
⋮	⋮	⋮

The following diagram shows the format for host link command (TXD) sent from the C200HS. The C200HS automatically attaches the prefixes and suffixes, such as the node number, header, and FCS.

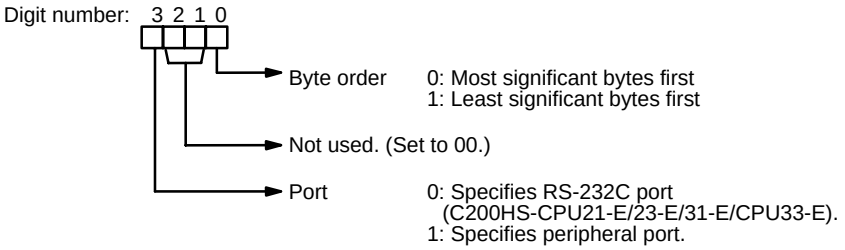


RS-232C Mode

N must be BCD from #0000 to #0256. The value of the control word determines the port from which data will be output and the order in which data will be written to memory.

Control Word

The value of the control word determines the port from which data will be read and the order in which data will be written to memory.



The specified number of bytes will be read from S through S+(NP2)–1 and transmitted through the specified port.

	MSB	LSB
S	1	2
S+1	3	4
S+2	5	6
S+3	7	8
⋮	⋮	⋮

When digit 0 of C is 0, the bytes of source data shown above will be transmitted in this order: 12345678...

When digit 0 of C is 1, the bytes of source data shown above will be transmitted in this order: 21436587...

Note When start and end codes are specified the total data length should be 256 bytes max., including the start and end codes.

Flags

ER: Another device is not connected to the peripheral port.

There is an error in the communications settings (PC Setup) or the operand settings.

Indirectly addressed DM word is non-existent. (Content of *DM word is not BCD, or the DM area boundary has been exceeded.)

The source words (S to S+(N÷2)–1) exceed the data area.

26405: RS-232C Port Communications Enabled Flag

26413: Peripheral Port Communications Enabled Flag

26705: Host Link Unit #0 Communications Enabled Flag

26713: Host Link Unit #1 Communications Enabled Flag

5-28 Advanced I/O Instructions

Advanced I/O instructions enable control, with a single instruction, of previously complex operations involving external I/O devices (digital switches, 7-segment displays, etc.).

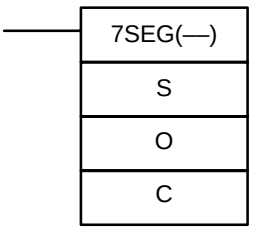
There are five advanced I/O instructions, as shown in the following table. All of these are expansion instructions and must be assigned to function codes before they can be used.

Name	Mnemonic	Function
7-SEGMENT DISPLAY OUTPUT	7SEG(—)	BCD output to 7-segment display
DIGITAL SWITCH INPUT	DSW(—)	Data input from a digital switch
HEXADECIMAL KEY INPUT	HKY(—)	Hexadecimal input from 16-key keypad
TEN-KEY INPUT	TKY(—)	BCD input from 10-key keypad
MATRIX INPUT	MTR(—)	Data input from an 8 x 8 matrix

Although TKY(—) is used only to simplify programming, the other advanced I/O instructions can be used to shorten cycle time, reduce the need for Special I/O Units, and reduce system cost. With the exception of TKY(—), however, the advanced I/O instructions can only be used once each in the program and cannot be used for I/O Units mounted to Slave Racks, where Special I/O Units must be used.

5-28-1 7-SEGMENT DISPLAY OUTPUT – 7SEG(––)

Ladder Symbols



Operand Data Areas

S: First source word	
IR, SR, AR, DM, HR, TC, LR	
O: Output word	
IR, SR, AR, HR, LR	
C: Control data	
000 to 007	

Limitations

S and S+1 must be in the same data area.

Do not set C to values other than 000 to 007.

Overview

When the execution condition is OFF, 7SEG(—) is not executed. When the execution condition is ON, 7SEG(—) reads the source data (either 4 or 8-digit), converts it to 7-segment display data, and outputs that data to the 7-segment display connected to the output indicated by O.

The value of C indicates the number of digits of source data and the logic for the Input and Output Units, as shown in the following table.

Source data	Display's data input logic	Display's latch input logic	C
4 digits (S)	Same as Output Unit	Same as Output Unit	0000
		Different from Output Unit	0001
	Different from Output Unit	Same as Output Unit	0002
		Different from Output Unit	0003
8 digits (S, S+1)	Same as Output Unit	Same as Output Unit	0004
		Different from Output Unit	0005
	Different from Output Unit	Same as Output Unit	0006
		Different from Output Unit	0007

If there are 8 digits of source data, they are placed in S and S+1, with the most significant digits placed in S+1. If there are 4 digits of source data, they are placed in S.

7SEG(—) displays the 4 or 8-digit data in 12 cycles, and then starts over and continues displaying the data.

The 7-segment display must provide four data lines and one latch signal line for each display digit.

Note
1. Consider the cycle time and the characteristics of the 7-segment display when designing the system.
2. Output bits not used here can be used as ordinary output bits.

Precautions

I/O refreshing must be performed for all I/O points used by 7SEG(—) each time it is executed to ensure effective operation. The I/O REFRESH instruction must thus be used with 7SEG(—) whenever 7SEG(—) is used in a subroutine to ensure that the I/O points are refreshed each execution. Refer to page 315 for an example of this type of programming.

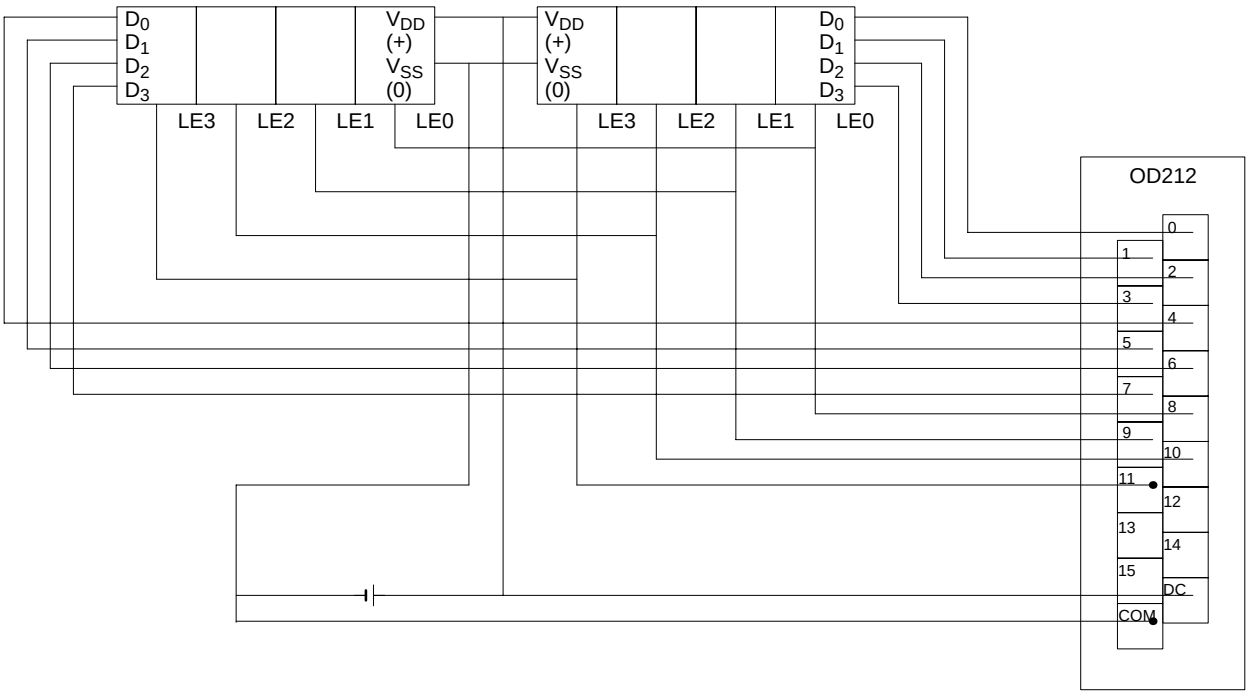
7SEG(—) will be executed from the first cycle whenever program execution is started, including restarts made after power interruptions.

Do not use 7SEG(—) more than once in the program.

7SEG(—) cannot be used for I/O Units mounted to Slave Racks.

Hardware

This instruction outputs word data to a 7-segment display. It utilizes either 8 output bits for 4 digits or 12 output bits for 8 digits. The 7-segment display is connected to an Output Unit as shown in the diagram below. For 4-digit display, the data outputs (D0 to D3) are connected to output points 0 through 3 (allocated word O), and latch outputs (CS0 to CS3) are connected to output points 4 through 7. Output point 12 (for 8-digit display) or output point 8 (for 4-digit display) will be turned ON when one round of data is displayed, but there is no need to connect them unless required by the application.



The outputs must be connected from an Output Unit with 8 or more output points for four digits or 16 or more output points for eight digits. Basic Output, Special I/O, or High-density Output Units can be used.

Note
1. Output Unit outputs normally employ negative logic. (Only the PNP output type employs positive logic.)

2. The 7-segment display may require either positive or negative logic, depending on the model.
3. The 7-segment display must have 4 data signal lines and 1 latch signal line for each digit.

Using the Instruction

If the first word holding the data to be displayed is specified at S, and the output word is specified at O, and the SV taken from the table below is specified at C, then operation will proceed as shown below when the program is executed. If only four digits are displayed, then only word S will be used.

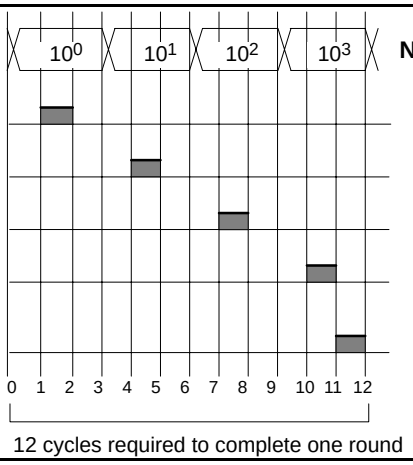
Data Storage Format

Leftmost 4 digits	Rightmost 4 digits
S+1	S

Timing

The timing of data output is shown in the following table. “O” is the first word holding display data and “C” is the output word.

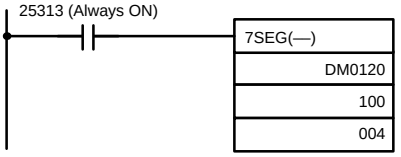
Function	Bit(s) in O		Output status (Data and latch logic depends on C)
	(4 digits, 1 block)	(4 digits, 2 blocks)	
Data output	00 to 03	00 to 03 04 to 07	
Latch output 0	04	08	
Latch output 1	05	09	
Latch output 2	06	10	
Latch output 3	07	11	
One Round Flag	08	12	



Note 0 to 3: Data output for word S
4 to 7: Data output for word S+1

Application Example

This example shows a program for displaying 8-digit BCD numbers at a 7-segment LED display. Assume that the 7-segment display is connected to output word IR 100. Also assume that the Output Unit is using negative logic, and that the 7-segment display logic is also negative for data signals and latch signals.



The 8-digit BCD data in DM 0120 (rightmost 4 digits) and DM 0121 (leftmost 4 digits) are always displayed by means of 7SEG(—). When the contents of DM 0120 and DM 0121 change, the display will also change.

Flags

ER: S and S+1 are not in the same data area. (When set to display 8-digit data.)

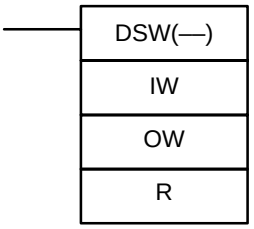
Indirectly addressed DM word is non-existent. (Content of *DM word is not BCD, or the DM area boundary has been exceeded.)

There is an error in operand settings.

25409: SR 25409 will be ON while 7SEG(—) is being executed.

5-28-2 DIGITAL SWITCH INPUT – DSW(––)

Ladder Symbols



Operand Data Areas

IW: Input word
IR, SR, AR, HR, LR
OW: Output word
IR, SR, AR, HR, LR
R: First result word
IR, SR, AR, DM, HR, LR

Overview

DSW(—) is used to read the value set on a digital switch connected to I/O Units. When the execution condition is OFF, DSW(—) is not executed. When the execution condition is ON, DSW(—) reads the 8-digit value set on the digital switch from IW and places the result in R.

The 8-digit value it is placed in R and R+1, with the most significant digits placed in R+1.

DSW(—) reads the 8-digit value in 20 executions, and then starts over and continues reading the data.

The digital switch must provide four data lines and one latch signal line and read signal line for each digit being input.

Precautions

I/O refreshing must be performed for all I/O points used by DSW(—) each time it is executed to ensure effective operation. The I/O REFRESH instruction must thus be used with DSW(—) whenever DSW(—) is used in a subroutine to ensure that the I/O points are refreshed each execution. Refer to page 315 for an example of this type of programming.

DSW(—) will be executed from the first cycle whenever program execution is started, including restarts made after power interruptions.

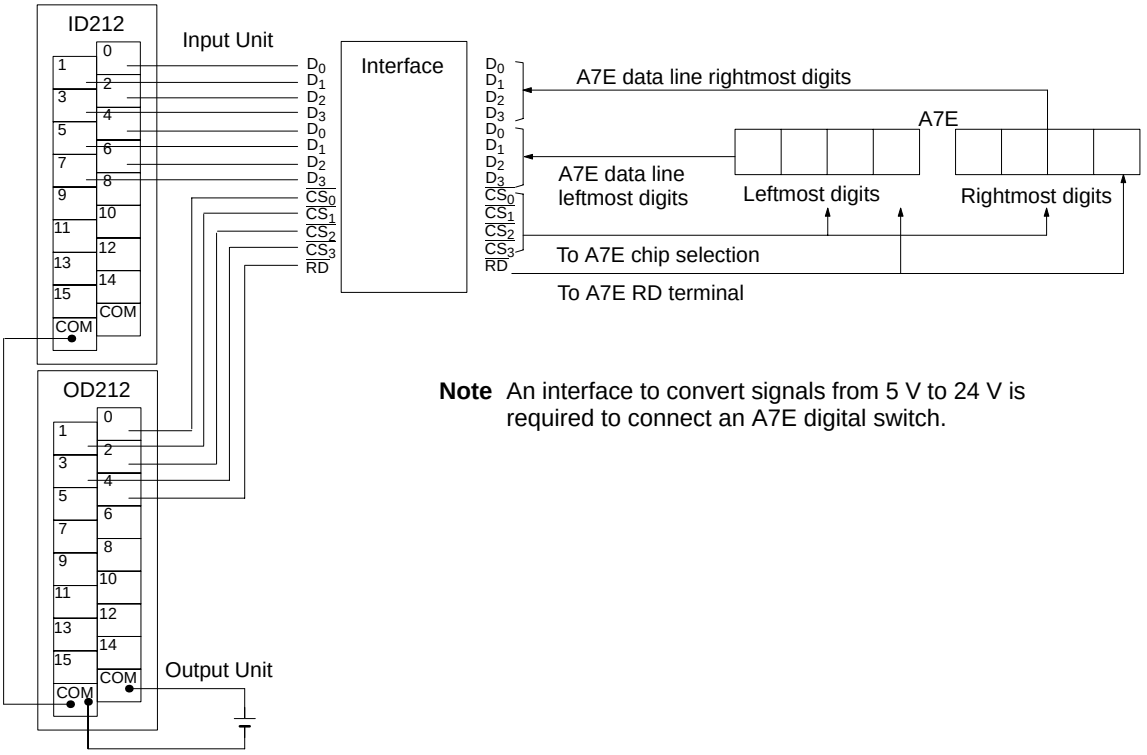
Do not use DSW(—) more than once in the program.

DSW(—) cannot be used for I/O Units mounted to Slave Racks.

Note Input and output bits not used here can be used as ordinary input and output bits.

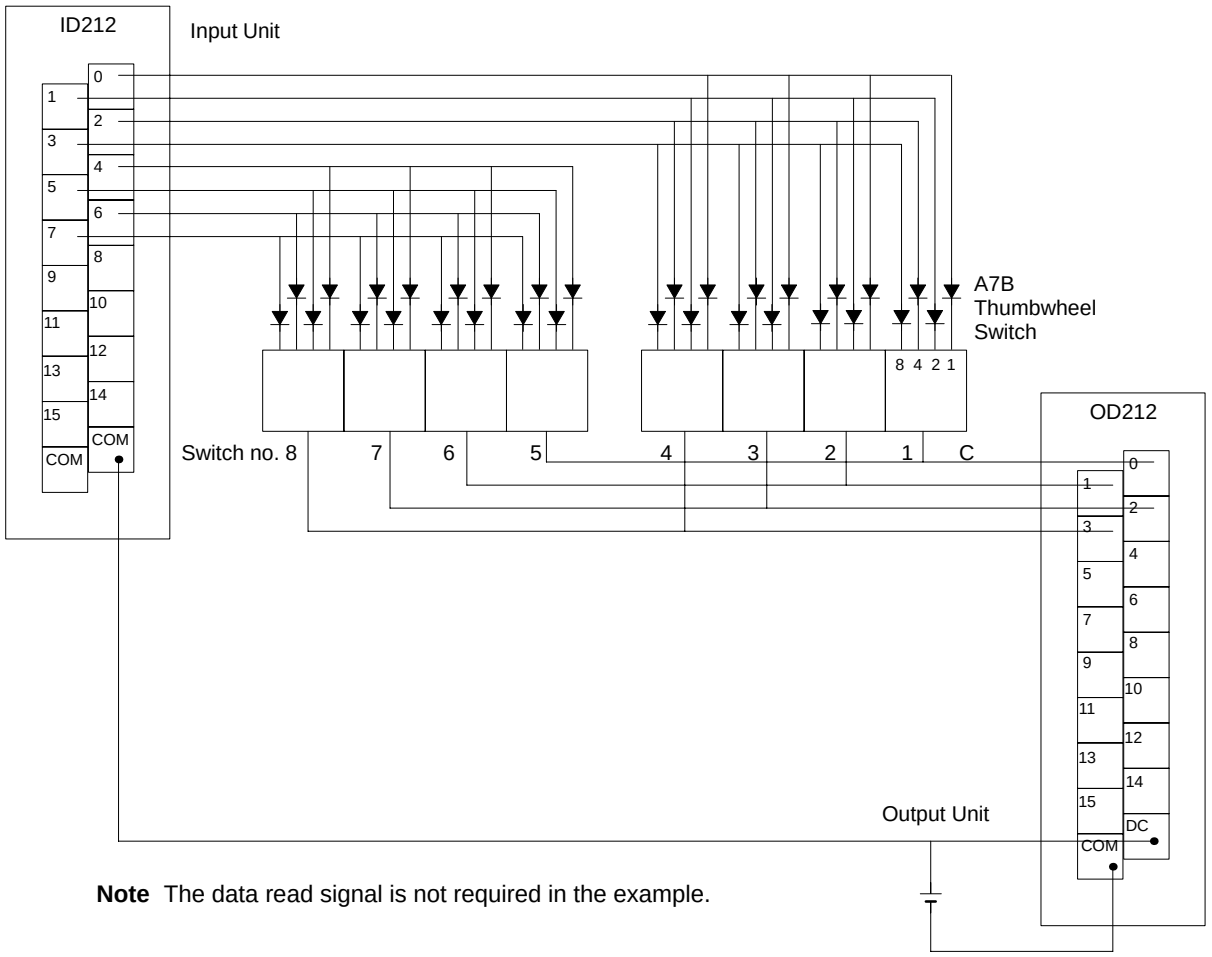
Hardware

With this instruction, 8-digit BCD set values are read from a digital switch. DSW(—) utilizes 5 output bits and 8 input bits. Connect the digital switch and the Input and Output Units as shown in the diagram below. Output point 5 will be turned ON when one round of data is read, but there is no need to connect output point 5 unless required for the application.



Note An interface to convert signals from 5 V to 24 V is required to connect an A7E digital switch.

The following example illustrates connections for an A7B Thumbwheel Switch.

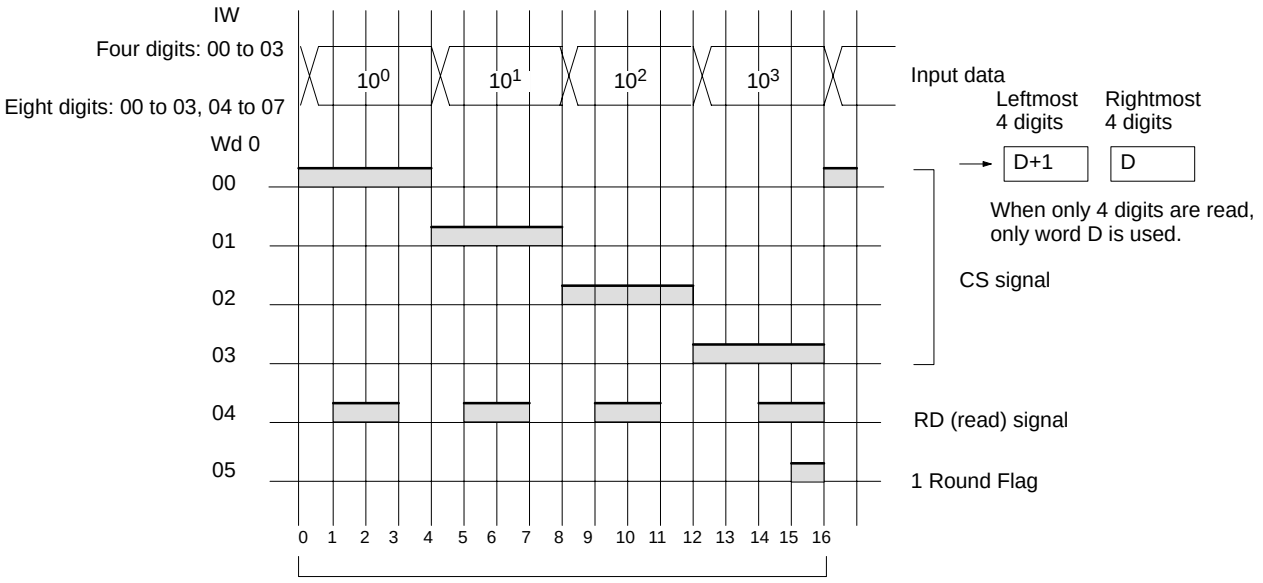


Note The data read signal is not required in the example.

The inputs must be connected to a DC Input Unit with 8 or more input points and the outputs must be connected from a Transistor Output Unit with 8 or more output points.

Using the Instruction

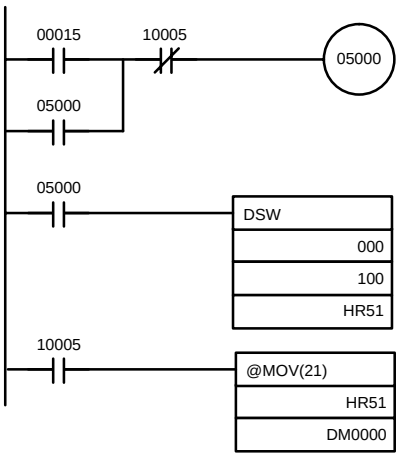
If the input word for connecting the digital switch is specified at for word A, and the output word is specified for word B, then operation will proceed as shown below when the program is executed.



16 cycles to complete one round of execution

Application Example

This example shows a program for reading 8 digits in BCD from the digital switch. Assume that the digital switch is connected to IR 000 (input) and IR 100 (output).



When IR 00015 turns ON, the IR 05000 will hold itself ON until the One Round Flag (IR 10005) turns ON upon completion of one round of reading by DSW(—).

The data set from the digital switch by DSW(—) is stored in HR 51.

When the One Round Flag (10005) turns ON after reading has been completed, the number stored in HR 51 is transferred to DM 0000.

Flags

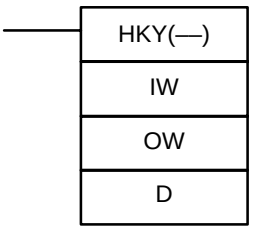
ER: Indirectly addressed DM word is non-existent. (Content of *DM word is not BCD, or the DM area boundary has been exceeded.)

R and R+1 are not in the same data area.

25410: ON while DSW(—) is being executed.

5-28-3 HEXADECIMAL KEY INPUT – HKY(––)

Ladder Symbols



Operand Data Areas

IW: Input word
IR, SR, AR, HR, LR
OW: Control signal output word
IR, SR, AR, HR, LR
D: First register word
IR, SR, AR, DM, HR, LR

Limitations

D and D+2 must be in the same data area.

Overview

When the execution condition is OFF, HKY(—) is not executed. When the execution condition is ON, HKY(—) inputs data from a hexadecimal keypad connected to the input indicated by IW. The data is input in two ways:

1, 2, 3...

1. An 8-digit shift register is created in D and D+1. When a key is pressed on the hexadecimal keypad, the corresponding hexadecimal digit is shifted into the least significant digit of D. The other digits of D, D+1 are shifted left and the most significant digit of D+1 is lost.
2. The bits of D+2 and bit 4 of OW indicate key input. When one of the keys on the keypad (0 to F) is being pressed, the corresponding bit in D+2 (00 to 15) and bit 4 of OW are turned ON.

Note

1. When one of the keypad keys is being pressed, input from the other keys is disabled.
2. Input and output bits not used here can be used as ordinary input and output bits.

With this instruction, one key input is read in 4 to 13 cycles. More than one cycle is required because the ON keys can only be determined as the outputs are turned ON to test them.

The hexadecimal key input device must be connectable in a 4 x 4 matrix.

Precautions

I/O refreshing must be performed for all I/O points used by HKY(—) each time it is executed to ensure effective operation. The I/O REFRESH instruction must thus be used with HKY(—) whenever HKY(—) is used in a subroutine to ensure that the I/O points are refreshed each execution. Refer to page 315 for an example of this type of programming.

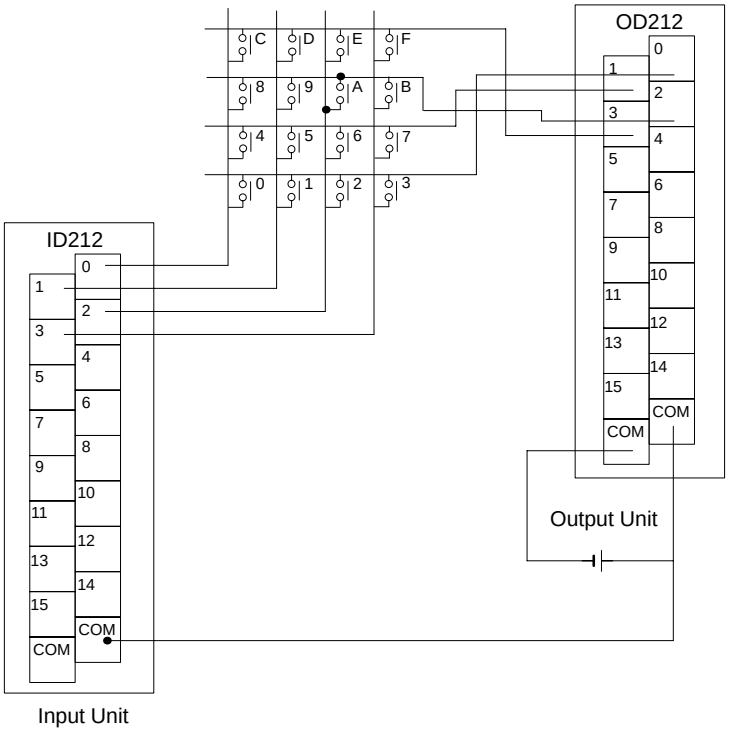
HKY(—) will be executed from the first cycle whenever program execution is started, including restarts made after power interruptions.

Do not use HKY(—) more than once in the program.

HKY(—) cannot be used for I/O Units mounted to Slave Racks.

Hardware

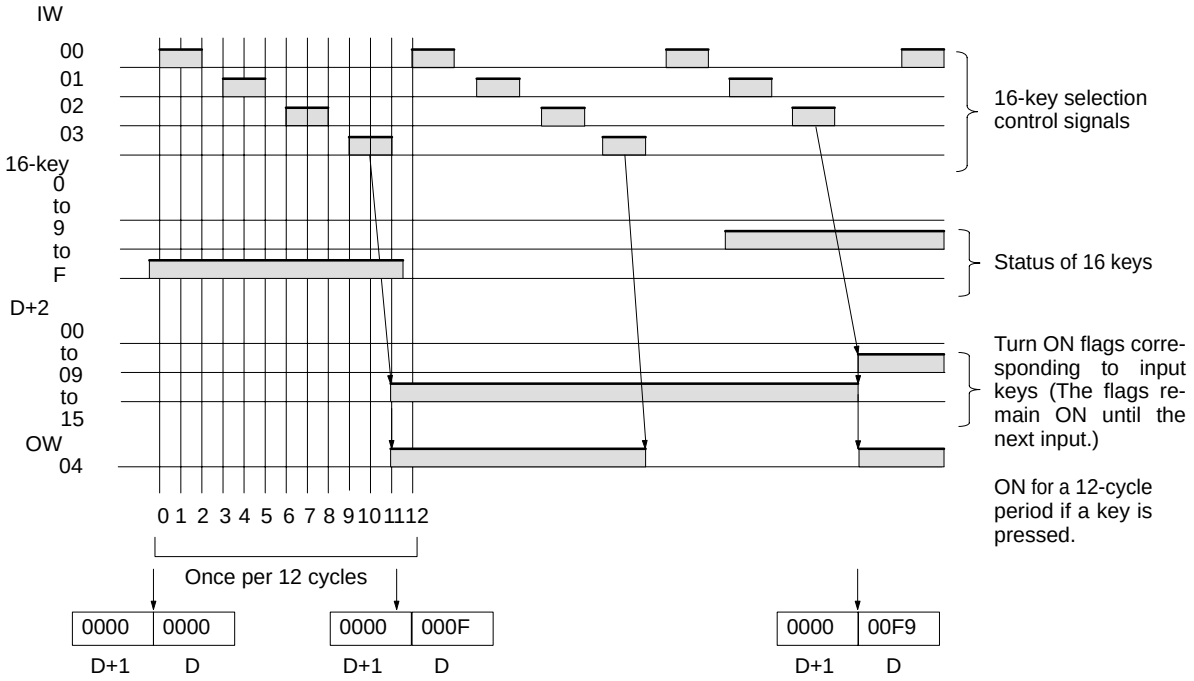
This instruction inputs 8 digits in hexadecimal from a hexadecimal keyboard. It utilizes 5 output bits and 4 input bits. Prepare the hexadecimal keyboard, and connect the 0 to F numeric key switches, as shown below, to input points 0 through 3 and output points 0 through 3. Output point 4 will be turned ON while any key is being pressed, but there is no need to connect it unless required by the application.



The inputs connected to the input terminals must be on a DC Input Unit with 8 or more input points and the outputs connected to the output terminals must be from a Transistor Output Unit with 8 points or more.

Using the Instruction

If the input word for connecting the hexadecimal keyboard is specified at word A, and the output word is specified at word B, then operation will proceed as shown below when the program is executed.



Flags

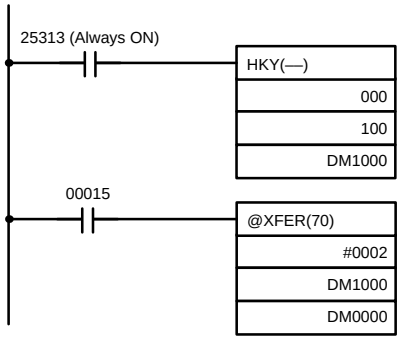
ER: Indirectly addressed DM word is non-existent. (Content of ⁎DM word is not BCD, or the DM area boundary has been exceeded.)

D and D+2 are not in the same data area.

SR 25408: ON while HKY(—) is being executed.

Example

This example shows a program for inputting numbers from a hexadecimal keyboard. Assume that the hexadecimal keyboard is connected to IR 000 (input) and IR 100 (output).

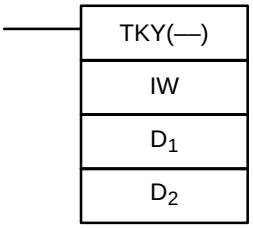


The hexadecimal key information that is input to IR 000 by HKY(—) is converted to hexadecimal and stored in words DM1000 and DM1001.

IR 00015 is used as an “ENTER key”, and when IR 00015 turns ON, the numbers stored in DM 1000 and DM 1001 are transferred to DM 0000 and DM 0001.

5-28-4 TEN KEY INPUT – TKY(––)

Ladder Symbols



Operand Data Areas

IW: Input word
IR, SR, AR, HR, LR
D_1: First register word
IR, SR, AR, DM, HR, LR
D_2: Key input word
IR, SR, AR, DM, HR, LR

Limitations

D_1 and D_1+1 must be in the same data area.

Overview

When the execution condition is OFF, TKY(—) is not executed. When the execution condition is ON, TKY(—) inputs data from a ten-key keypad connected to the input indicated by IW. The data is input in two ways:

TKY(—) can be used in several locations in the program by changing the input word, IW.

1, 2, 3...

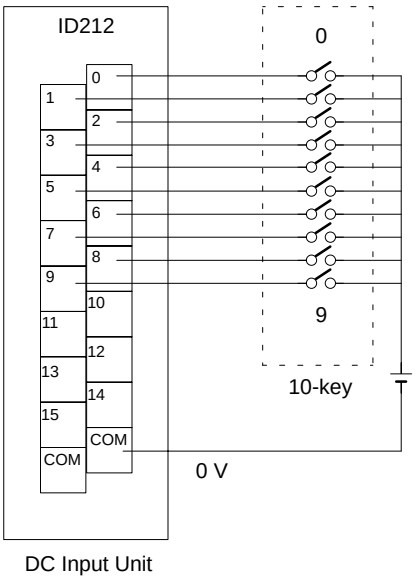
1. An 8-digit shift register is created in D_1 and D_1+1. When a key is pressed on the ten-key keypad, the corresponding BCD digit is shifted into the least significant digit of D_1. The other digits of D_1, D_1+1 are shifted left and the most significant digit of D_1+1 is lost.
2. The first ten bits of D_2 indicate key input. When one of the keys on the keypad (0 to 9) is being pressed, the corresponding bit of D_2 (00 to 09) is turned ON.

Note

1. While one key is being pressed, input from other keys will not be accepted.
2. If more than eight digits are input, digits will be deleted beginning with the leftmost digit.
3. Input bits not used here can be used as ordinary input bits.

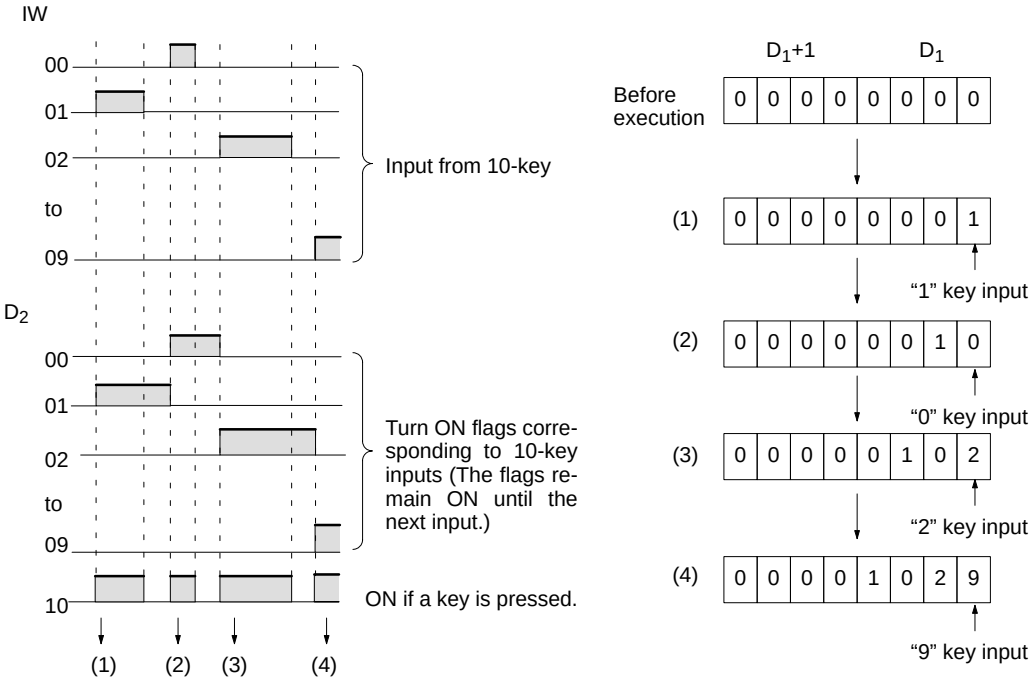
Hardware

This instruction inputs 8 digits in BCD from a 10-key keypad and uses 10 input points. Prepare a 10-key keypad, and connect it so that the switches for numeric keys 0 through 9 are input to points 0 through 9 as shown in the following diagram. The inputs on a DC Input Unit with 16 or more input points can be used.



Using the Instruction

If the input word for connecting the 10-key keypad is specified for IW, then operation will proceed as shown below when the program is executed.



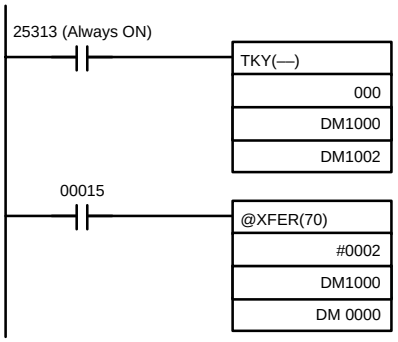
Flags

ER: Indirectly addressed DM word is non-existent. (Content of ✳DM word is not BCD, or the DM area boundary has been exceeded.)

D_1 and D_1+1 are not in the same data area.

Example

In this example, a program for inputting numbers from the 10-key is shown. Assume that the 10-key is connected to IR 000.

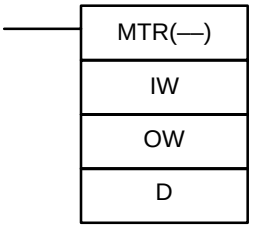


The 10-key information input to IR 000 using TKY(—) is converted to BCD and stored in DM 1000 and DM 1001. Key information is stored in DM 1002.

IR 00015 is used as an “ENTER key”, and when IR 00015 turns ON, the data stored in DM 1000 and DM 1001 will be transferred to DM 0000 and DM 0001.

5-28-5 MATRIX INPUT – MTR(––)

Ladder Symbols



Operand Data Areas

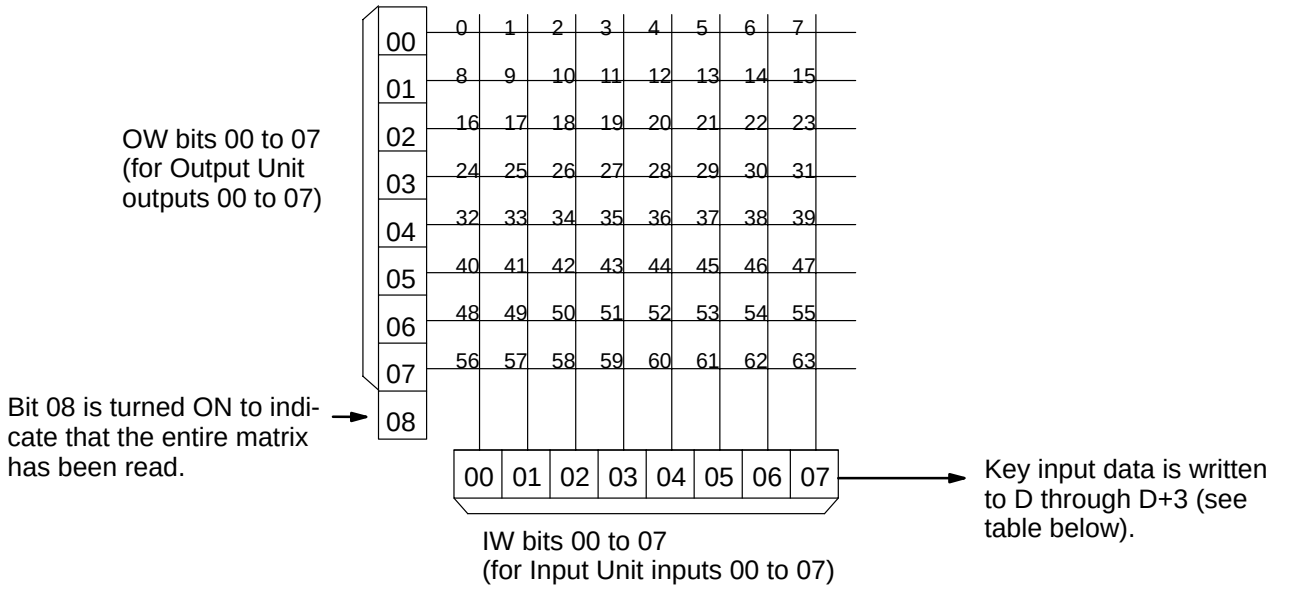
IW: Input word
IR, SR, AR, HR, LR
OW: Output word
IR, SR, AR, HR, LR
D: First destination word
IR, SR, AR, DM, HR, LR

Limitations

D and D+3 must be in the same data area.

Overview

When the execution condition is OFF, MTR(—) is not executed. When the execution condition is ON, MTR(—) inputs data from an 8 × 8 matrix and records that data in D to D+3. Data for all 64 points in the matrix will be recorded even when fewer than 64 keys are connected.



A selection signal is output to OW bits 00 to 07 consecutively for 3 cycles. Only one output bit will be turned on at a time. Bit 08 of OW is turned ON for 3 cycles after 07 to indicate when each round of reading the matrix has been completed.

When one of the 64 keys is pressed, an input will be received at one of the input bits. The key that was pressed is identified by comparing the output bit to which the signal was output and input bit at which it was received.

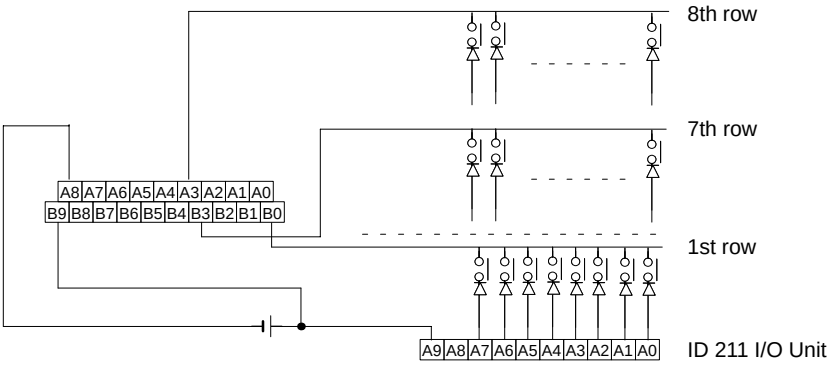
When an key input is detected, the corresponding bit in D through D+3 is turned ON. The following table shows the correspondence between keys and bits in D through D+3.

Word	Bits	Corresponding Keys
D	00 to 15	0 to 15
D+1	00 to15	16 to 31
D+2	00 to 15	32 to 47
D+3	00 to 15	48 to 63

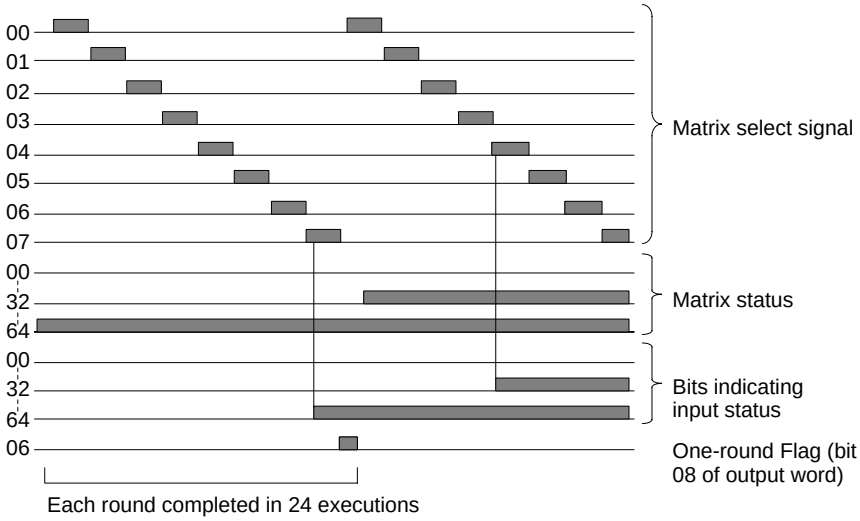
Hardware

This instruction inputs up to 64 signals from an 8 x 8 matrix using 8 input points and 8 output points. Any 8 x 8 matrix can be used. The inputs must be connected through a DC Input Unit with 8 or more points and the outputs must be connected through a Transistor Output Unit with 8 or more points. The basic wiring and timing diagrams for MTR(—) are shown below.

Wiring



Timing Diagram



Precautions

The 64 keys can be divided into 8 rows (including a row for OW bit 08) which are scanned consecutively. Since each row is scanned for 3 cycles, a delay of up to 25 cycles can occur before a given row of keys is scanned for inputs.

I/O refreshing must be performed for all I/O points used by MTR(—) each time it is executed to ensure effective operation. The I/O REFRESH instruction must thus be used with MTR(—) whenever MTR(—) is used in a subroutine to ensure that the I/O points are refreshed each execution.

MTR(—) will be executed from the first cycle whenever program execution is started, including restarts made after power interruptions.

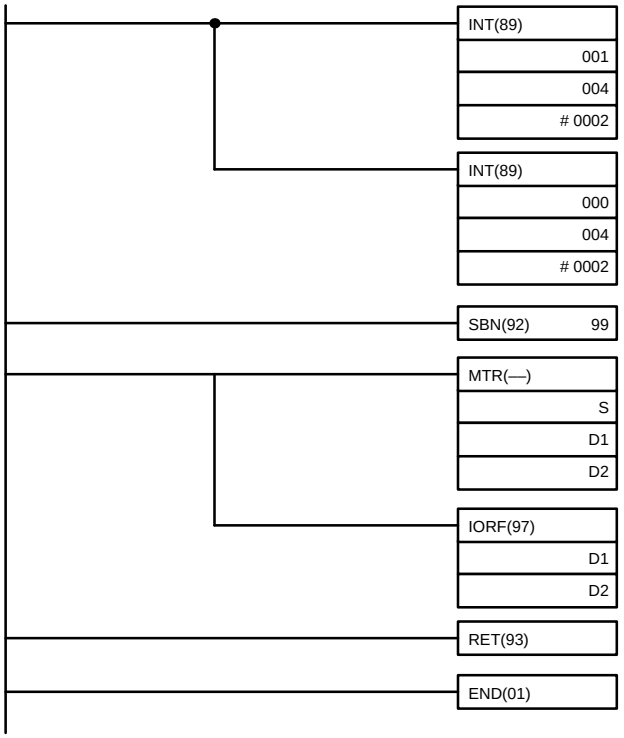
SR 25403, which is turned on while MTR(—) is being executed, is reset in an interlocked program section and MTR(—) is not executed in an interlocked program section.

Do not use MTR(—) more than once in the program.

MTR(—) cannot be used for I/O Units mounted to Slave Racks.

Example

The following examples shows programming MTR(—) in a scheduled subroutine, where IORF(97) is programmed to ensure that the I/O words used by MTR(—) are refreshed each time MTR(—) is executed.



Flags

ER: Indirectly addressed DM word is non-existent. (Content of ⁎DM word is not BCD, or the DM area boundary has been exceeded.)

25403: SR 25403 is ON while MTR(—) is being executed.

SECTION 6
Program Execution Timing

The timing of various operations must be considered both when writing and debugging a program. The time required to execute the program and perform other CPU operations is important, as is the timing of each signal coming into and leaving the PC in order to achieve the desired control action at the right time. This section explains the cycle and shows how to calculate the cycle time and I/O response times.

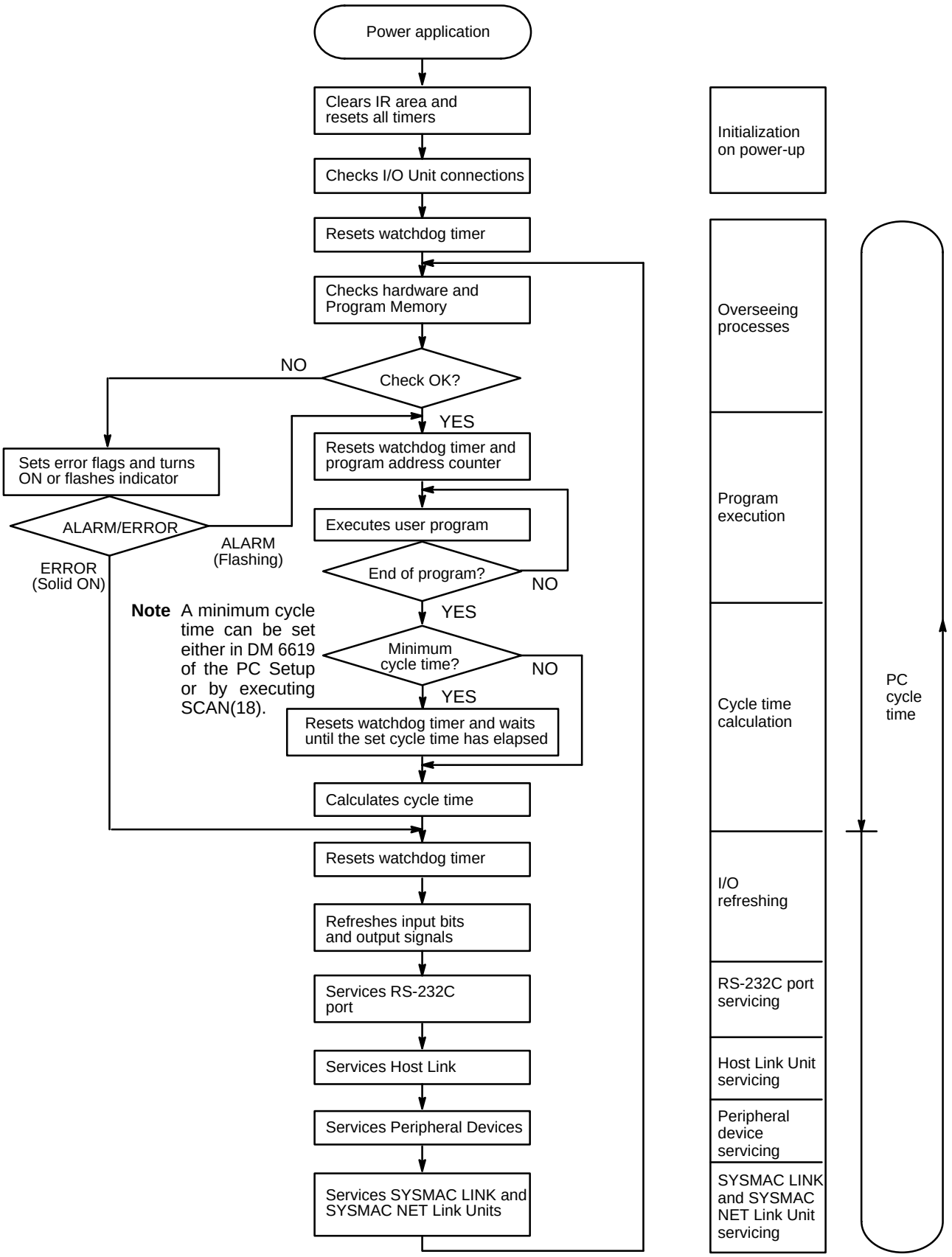
6-1 Cycle Time

To aid in PC operation, the average, maximum, and minimum cycle times can be displayed on the Programming Console or any other Programming Device and the maximum cycle time and current cycle time values are held in AR 26 and AR 27. Understanding the operations that occur during the cycle and the elements that affect cycle time is, however, 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 run through all required CPU operations is called a cycle; the time required for each cycle is called the cycle time.

The overall flow of the CPU operation is as shown in the following flowchart.

Flowchart of CPU Operation



The first three operations immediately after power application are performed only once each time the PC is turned on. The rest of the operations are performed in cyclic fashion. The cycle time is the time that is required for the CPU to complete one of these cycles. This cycle includes basically eight types of operation.

1, 2, 3...

1. Overseeing
2. Program execution
3. Cycle time calculation
4. I/O refreshing
5. Host Link Unit servicing
6. RS-232C port servicing
7. Peripheral Device servicing
8. SYSMAC NET/SYSMAC LINK servicing.

The cycle time is the total time required for the PC to perform all of the above operations. The time required for operation 3, cycle time calculation, is negligible and can be ignored in actual calculations

Operation	Time required	Function
1. Overseeing	0.7 ms	Watchdog timer set. I/O Bus, Program Memory checked. Clock refreshed.
2. Program execution	Total execution time for all instructions varies with program size, the instructions used, and execution conditions. Refer to *6-3 Instruction Execution Times* for details.	Program executed.
3. Cycle time calculation	Negligible, but a wait can be generated to bring the cycle time up to the minimum setting if one has been made.	Cycle time calculated. When the CYCLE TIME instruction (SCAN(18)) is executed, waits until the set time has elapsed and then resets the watchdog timer.
4. I/O refreshing	Total of following times: 20 µs per input byte (8 points). 20 µs per output byte. (12-point Output Units calculated as 16 points.) PC Link Unit I/O refresh time. Special I/O Unit refresh time. 1.1 ms per Remote I/O Master Unit + 0.17 ms per I/O word used on Slave Racks. Group-2 (High-density) I/O Unit refresh time. Refer to the tables below for details on PC Link, Special I/O Unit, and Group-2 High-density I/O Unit refresh times.	Input bits set according to status of input signals. Output signals sent according to status of output bits in memory. Inputs and Outputs in Remote I/O Systems refreshed. Special I/O Units serviced. Group-2 High-density I/O Units serviced.
5. Host Link Unit servicing	6 ms per Unit max.	Commands from computers connected through Rack-mounting Host Link Units processed.
6. RS-232C port servicing (except CPU01-E/03-E)	0 ms when no device is connected. T x 0.05, where T is the cycle time calculated in operation 3	Communications with devices connected to RS-232C port processed.
7. Peripheral device servicing	0 ms when no device is connected. 0.26 ms minimum or T x 0.05, where T is the cycle time calculated in operation 3	Commands from Programming Devices (computers, Programming Consoles, etc.) processed.
8. SYSMAC NET/ SYSMAC LINK servicing (CPU31-E/33-E only)	0 ms when no Communications Unit is mounted. 0.8 ms + 10 ms max. per Unit.	Commands from computers and other devices connected to SYSMAC NET/SYSMAC LINK Units processed.

PC Link Unit I/O Refresh

I/O pts to refresh	Time required (ms)
512	7.4
256	4.1
128	2.7
64	1.7

Special I/O Unit Refresh

Unit	Time required per Unit
C200H-ID501/215	0.6 ms
C200H-OD501/215	0.6 ms when set for 32 I/O pts.
C200H-MD501/215	1.6 ms when set for dynamic I/O
C200H-CT001-V1/CT002	2.6 ms
C200H-NC111/NC112	2.5 ms
C200H-NC211	5.0 ms
C200H-AD001	1.3 ms
C200H-DA001	1.0 ms
C200H-TS001/TS101	1.2 ms
C200H-ASC02	1.9 ms normally, 5.0 ms for @ format
C200H-IDS01-V1/IDS21	2.0 ms normally, 5.5 ms for command transfer
C200H-OV001	4.1 ms
C200H-FZ001	2.0 ms
C200H-TC001/002/003/ 101/102/103	3.0 ms
C200H-CP114	2.5 ms
C200H-AD002	1.4 ms
C200H-PID□□	3.0 ms

Group-2 High-density I/O Unit Refresh

Unit	Time required per Unit
C200H-ID216	0.18 ms
C200H-OD218	0.14 ms
C200H-ID217	0.31 ms
C200H-OD219	0.23 ms

NT Links

If the PC is connected to a Programmable Terminal (PT) via a C200H Interface Unit, the times shown in the following table will be required to refresh I/O for the PT.

Number of table entries for PT	I/O refresh time
Minimum setting: Character string table: 0 Numeral table: 0	2.5 ms
Maximum setting: Character string table: 32 Numeral table: 128	5.4 ms

Watchdog Timer and Long Cycle Times

Within the PC, the watchdog timer measures the cycle time and compares it to a set value. If the cycle time exceeds the set value of the watchdog timer, a FALS 9F error is generated and the CPU stops. WDT(94) can be used to extend the set value for the watchdog timer.

Even if the cycle time does not exceed the set value of the watchdog timer, a long cycle time can adversely affect the accuracy of system operations as shown in the following table.

Cycle time (ms)	Possible adverse affects
10 or greater	TIMH(15) inaccurate when TC 016 through TC 511 are used. (Accuracy when using TC 000 through TC 0015 not affected.)
20 or greater	0.02-second clock pulse (SR 25401) not accurately readable.
100 or greater	0.1-second clock pulse (SR 25500) not accurately readable and Cycle Time Error Flag (SR 25309) turns ON.
200 or greater	0.2-second clock pulse (SR 25501) not accurately readable.
6,500 or greater	FALS code 9F generated regardless of watchdog timer setting and the system stops.

Online Editing

When online editing is executed from a Programming Device, operation will be interrupted for a maximum of 260 ms and interrupts will be masked to rewrite the user program. No warnings will be given for long cycle times during this interval. Check the effects on I/O response time before editing the program online.

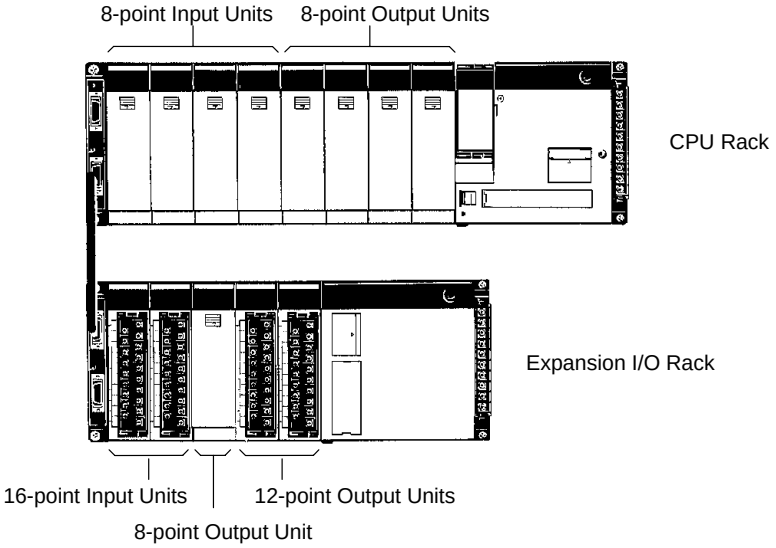
⚠ **Caution** Editing the program online can cause delays in I/O responses with no warnings being given from the system for the long cycle time produced by editing online. Before editing online, make sure that delays in I/O responses will not create a dangerous situation in the controlled system.

6-2 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 employed. This section shows some basic cycle time calculation examples. To simplify the examples, the instructions used in the programs have been assumed to be all either LD or OUT. The average execution time for the instructions is thus 0.47 μs. (Execution times are given in the table in *6-3 Instruction Execution Times.*)

6-2-1 PC with I/O Units Only

Here, we'll compute the cycle time for a simple PC. The CPU controls only I/O Units, eight on the CPU Rack and five on a 5-slot Expansion I/O Rack. The PC configuration for this would be as shown below. It is assumed that the program contains 5,000 instructions requiring an average of 0.47 μs each to execute.



Calculations

The equation for the cycle time from above is as follows:

Cycle time = Overseeing time + Program execution time + I/O refresh time + Peripheral device servicing time

Process	Calculation	With Peripheral Device	Without Peripheral Device
Overseeing	Fixed	0.7 ms	0.7 ms
Program execution	0.47 μs/instruction × 5,000 instructions	2.35 ms	2.35 ms
I/O refresh	See below.	0.34 ms	0.34 ms
Peripheral device servicing	Minimum time	0.26 ms	0.0 ms
Cycle time	**Total of above**	**3.65 ms**	**3.39 ms**

The I/O refresh time would be as follows for two16-point Input Units, four 8-point Input Units, two 12-point Output Units (12-point Units are treated as 16-point Units), and five 8-point Output Units controlled by the PC:

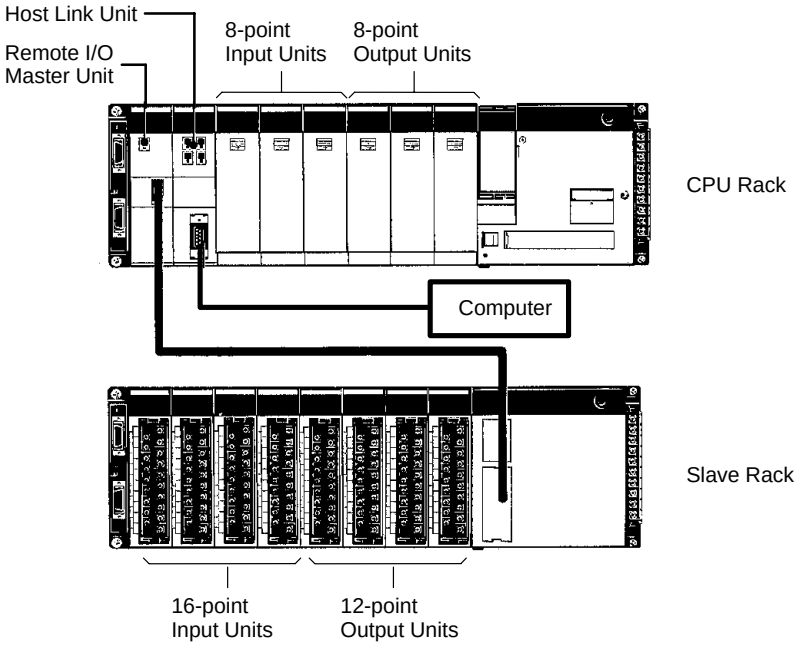
$$\frac{(16 \text{ pts} \times 2) + (8 \text{ pts} \times 4)}{8 \text{ pts}} \times 20\ \mu\text{s} + \frac{(16 \text{ pts} \times 2) + (8 \text{ pts} \times 5)}{8 \text{ pts}} \times 20\ \mu\text{s} = 0.34 \text{ ms}$$

6-2-2 PC with Link Units

Here, the cycle time is computed for a PC with a CPU21-E or CPU23-E CPU that has a Memory Unit with a clock function installed. The PC configuration for this could be as shown below.

The CPU controls three 8-point Input Units, three 8-point Output Units, a Host Link Unit, and a Remote I/O Master Unit connected to a Remote I/O Slave Rack containing four 16-point Input Units and four 12-point Output Units.

It is assumed that the program contains 5,000 instructions requiring an average of 0.47 μs each to execute, and that nothing is connected to the RS-232C port and no SYSMAC NET/SYSMAC LINK Unit is mounted.



Calculations

The equation for the cycle time is as follows:

Cycle time = Overseeing time + Program execution time
+ I/O refreshing time + Host Link Unit servicing time
+ Peripheral device servicing time

Process	Calculation	With Peripheral Device	Without Peripheral Device
Overseeing	Fixed	0.7 ms	0.7 ms
Program execution	0.47 μs/instruction × 5,000 instructions	2.35 ms	2.35 ms
I/O refresh	See below.	2.58 ms	2.58 ms
Host Link servicing	Fixed	6.0 ms	6.0 ms
Peripheral device servicing	0.7 + 2.35 + 2.58 + 6 = 11.63 11.63 x 0.05 = 0.58	0.58 ms	0.0 ms
Cycle time	**Total of above**	**12.21 ms**	**11.63 ms**

The I/O refreshing time would be as follows for three 8-point Input Units and three 8-point Output Units mounted in the CPU Rack, and eight Units mounted in a Slave Rack.

$$\frac{(8 \text{ pts} \times 3) + (8 \text{ pts} \times 3)}{8 \text{ pts}} \times 20\ \mu s + 1.1 \text{ ms} + 8 \text{ Units} \times 0.17 \text{ ms} = 2.58 \text{ ms}$$

6-3 Instruction Execution Times

The following table lists the execution times for all instructions that are available for the C200HS. 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(04) 00 and JME(05) 00 and the execution condition for JMP(04) 00 is OFF, or whether it is reset by an OFF execution condition. “R”, “IL”, and “JMP” are used to indicate these three times.

All execution times are given in microseconds unless otherwise noted.

Instruction	Conditions	ON execution time (μs)	OFF execution time (μs)
LD	For IR and SR 23600 to SR 25515	0.375	0.375
	For SR 25600 to SR 51115	0.75	0.375
LD NOT	For IR and SR 23600 to SR 25515	0.375	0.375
	For SR 25600 to SR 51115	0.75	0.375
AND	For IR and SR 23600 to SR 25515	0.375	0.375
	For SR 25600 to SR 51115	0.75	0.375
AND NOT	For IR and SR 23600 to SR 25515	0.375	0.375
	For SR 25600 to SR 51115	0.75	0.375
OR	For IR and SR 23600 to SR 25515	0.375	0.375
	For SR 25600 to SR 51115	0.75	0.375
OR NOT	For IR and SR 23600 to SR 25515	0.375	0.375
	For SR 25600 to SR 51115	0.75	0.375

Instruction	Conditions	ON execution time (μs)	OFF execution time (μs)
AND LD	---	0.375	0.375
OR LD	---	0.375	0.375
OUT	For IR and SR 23600 to SR 25515	0.563	0.563
	For SR 25600 to SR 51115	0.938	0.563
OUT NOT	For IR and SR 23600 to SR 25515	0.563	0.563
	For SR 25600 to SR 51115	0.938	0.563
TIM	Constant for SV	1.125	R: 1.125 IL: 1.125 JMP: 1.125
	⁎DM for SV		R: 39.0875 IL: 1.125 JMP: 1.125
	For designated words 256 to 511		R: 22.2875 IL: 1.125 JMP: 1.125
CNT	Constant for SV	1.125	R: 1.125 IL: 1.125 JMP: 1.125
	⁎DM for SV		R: 39.0875 IL: 1.125 JMP: 1.125
	For designated words 256 to 511		R: 22.0875 IL: 1.125 JMP: 1.125
SET	For IR and SR 23600 to SR 25515	0.563	0.563
	For SR 25600 to SR 51115	0.938	0.563
RSET	For IR and SR 23600 to SR 25515	0.563	0.563
	For SR 25600 to SR 51115	0.938	0.563
NOP(00)	---	0.375	---
END(01)	---	34.20	---
IL(02)	---	11.44	14.00
ILC(03)	---	12.60	12.64
JMP(04)	---	12.32	14.56
JME(05)	---	12.28	12.28
FAL(06) 01 to 99	---	93.76	0.375
FAL(06) 00	---	77.56	0.375
FALS(07)	---	4.28 ms	0.375
STEP(08)	---	50.60 (with operand)	2.25
		24.56 (without operand)	
SNXT(09)	---	13.96	2.25

Instruction	Conditions	ON execution time (μs)	OFF execution time (μs)
SFT(10)	With 1-word shift register	47.06	R: 35.80 IL: 15.70 JMP: 15.60
	With 100-word shift register	340.00	R: 256.80 IL: 15.72 JMP: 15.68
	With 250-word shift register	800.00	R: 590.80 IL: 15.60 JMP: 15.66
KEEP(11)	For IR and SR 23600 to SR 25515	0.563	0.563
	For SR 25600 to SR 51115	0.938	
CNTR(12)	Constant for SV	38.20	R: 27.90 IL: 20.30 JMP: 20.30
	*DM for SV	53.20	R: 28.00 IL: 20.30 JMP: 20.30
DIFU(13)	---	20.60	Normal: 20.60 IL: 20.40 JMP: 18.00
DIFD(14)	---	20.40	Normal: 20.40 IL: 20.20 JMP: 17.80
TIMH(15)	Interrupt Constant for SV	32.20	R: 40.40 IL: 39.20 JMP: 24.90
	Normal cycle	29.40	R: 36.30 IL: 35.20 JMP: 20.80
	Interrupt *DM for SV	29.80	R: 59.80 IL: 58.60 JMP: 24.90
	Normal cycle	27.40	R: 56.00 IL: 54.60 JMP: 20.80
WSFT(16)	When shifting 1 word	36.90	3
	When shifting 6,144 words using *DM	11.27 ms	
CMP(20)	When comparing a constant to a word	24.20	1.125
	When comparing two words	26.40	
	When comparing two *DM	61.80	
MOV(21)	When transferring a constant to a word	19.00	1.125
	When comparing two words	21.20	
	When transferring *DM to *DM	57.20	
MVN(22)	When transferring a constant to a word	20.20	1.125
	When comparing two words	22.40	
	When transferring *DM to *DM	57.20	

Instruction	Conditions	ON execution time (μs)	OFF execution time (μs)
BIN(23)	When converting a word to a word	40.40	1.125
	When converting ⁎DM to ⁎DM	74.80	
BCD(24)	When converting a word to a word	38.40	1.125
	When converting ⁎DM to ⁎DM	72.80	
ASL(25)	When shifting a word	21.20	0.75
	When shifting ⁎DM	38.20	
ASR(26)	When shifting a word	21.20	0.75
	When shifting ⁎DM	38.20	
ROL(27)	When rotating a word	21.80	0.75
	When rotating ⁎DM	39.00	
ROR(28)	When rotating a word	21.80	0.75
	When rotating ⁎DM	39.00	
COM(29)	When inverting a word	21.90	0.75
	When inverting ⁎DM	39.30	
ADD(30)	Constant + word → word	40.10	1.5
	Word + word→ word	42.50	
	⁎DM + ⁎DM → ⁎DM	94.10	
SUB(31)	Constant – word → word	40.10	1.5
	Word – word→ word	42.50	
	⁎DM – ⁎DM → ⁎DM	94.10	
MUL(32)	Constant x word → word	56.90	1.5
	Word x word→ word	59.30	
	⁎DM x ⁎DM → ⁎DM	110.90	
DIV(33)	Word ÷ constant → word	56.90	1.5
	Word ÷ word→ word	59.10	
	⁎DM ÷ ⁎DM → ⁎DM	110.70	
ANDW(34)	Constant AND word → word	34.10	1.5
	Word AND word→ word	37.10	
	⁎DM AND ⁎DM → ⁎DM	88.70	
ORW(35)	Constant OR word → word	34.10	1.5
	Word OR word→ word	36.70	
	⁎DM OR ⁎DM → ⁎DM	88.30	
XORW(36)	Constant XOR word → word	34.10	1.5
	Word XOR word→ word	36.70	
	⁎DM XOR ⁎DM → ⁎DM	88.30	
XNRW(37)	Constant XNOR word → word	34.30	1.5
	Word XNOR word→ word	36.90	
	⁎DM XNOR ⁎DM → ⁎DM	88.50	
INC(38)	When incrementing a word	23.70	0.75
	When incrementing ⁎DM	41.10	
DEC(39)	When decrementing a word	24.30	0.75
	When decrementing ⁎DM	41.70	
STC(40)	---	12.20	0.375
CLC(41)	---	12.30	0.375
TRSM(45)	---	27.70	27.70
MSG(46)	Designated as DM	23.20	0.75
	Designated as ⁎DM	40.80	

Instruction	Conditions	ON execution time (μs)	OFF execution time (μs)
ADB(50)	Constant + word → word	43.20	1.5
	Word + word → word	45.80	
	⁎DM + ⁎DM → ⁎DM	97.40	
SBB(51)	Constant – word → word	43.20	1.5
	Word – word → word	45.80	
	⁎DM – ⁎DM → ⁎DM	97.40	
MLB(52)	Constant x word → word	36.00	1.5
	Word x word → word	38.50	
	⁎DM x ⁎DM → ⁎DM	91.10	
DVB(53)	Word ÷ constant → word	36.70	1.5
	Word ÷ word → word	39.30	
	⁎DM ÷ ⁎DM → ⁎DM	90.80	
ADDL(54)	Word + word → word	45.50	1.5
	⁎DM + ⁎DM → ⁎DM	99.00	
SUBL(55)	Word – word → word	45.50	1.5
	⁎DM – ⁎DM → ⁎DM	99.00	
MULL(56)	Word x word → word	155.90	1.5
	⁎DM x ⁎DM → ⁎DM	209.50	
DIVL(57)	Word ÷ word → word	166.10	1.5
	⁎DM ÷ ⁎DM → ⁎DM	219.70	
BINL(58)	When converting words to words	58.70	1.125
	When converting ⁎DM to ⁎DM	93.50	
BCDL(59)	When converting words to words	47.30	1.125
	When converting ⁎DM to ⁎DM	82.20	
XFER(70)	When transferring 1 word	54.80	1.5
	When transferring 1,024 words using ⁎DM	2.40 ms	
	When transferring 6,143 words using ⁎DM	13.99 ms	
BSET(71)	When setting a constant to 1 word	37.70	1.5
	When setting ⁎DM ms to 1,024 words using ⁎DM	26.20	
	When setting ⁎DM ms to 6,144 words using ⁎DM	95.80	
ROOT(72)	When taking root of word and placing in a word	68.60	1.125
	When taking root of 99,999,999 in ⁎DM and placing in ⁎DM	137.80	
XCHG(73)	Between words	33.50	1.125
	Between ⁎DM	68.50	
SLD(74)	When shifting 1 word	34.00	1.125
	When shifting 1,024 DM words using ⁎DM	4.35 ms	
	When shifting 6,144 DM words using ⁎DM	25.93 ms	
SRD(75)	When shifting 1 word	34.00	1.125
	When shifting 1,024 DM words using ⁎DM	4.35 ms	
	When shifting 6,144 DM words using ⁎DM	26.01 ms	
MLPX(76)	When decoding word to word	86.80	1.5
	When decoding ⁎DM to ⁎DM	178.20	

Instruction	Conditions	ON execution time (μs)	OFF execution time (μs)
DMPX(77)	When encoding a word to a word	48.90	1.5
	When encoding ✳DM to ✳DM	185.90	
SDEC(78)	When decoding a word to a word	53.20	1.5
	When decoding 2 digits ✳DM to ✳DM	113.60	
	When decoding 4 digits ✳DM to ✳DM	126.00	
FDIV(79)	Word ÷ word → word (equals 0)	118.20	1.5
	Word ÷ word → word (doesn't equal 0)	357.20	
	✳DM ÷ ✳DM → ✳DM	409.20	
DIST(80)	Constant → (word + (word))	49.00	1.5
	✳DM → (✳DM + (✳DM))	106.70	
COLL(81)	(Word + (word)) → word	52.90	1.5
	(✳DM + (✳DM)) → ✳DM	113.50	
MOVB (82)	When transferring a constant to a word	38.60	1.5
	When transferring word to a word	45.30	
	When transferring ✳DM to ✳DM	97.60	
MOVD(83)	When transferring a constant to a word	34.00	1.5
	When transferring word to a word	41.00	
	When transferring ✳DM to ✳DM	97.60	
SFTR(84)	When shifting 1 word	48.40	1.5
	When shifting 1,024 ✳DM using ✳DM	1.92 ms	
	When shifting 6,144 ✳DM using ✳DM	11.8 ms	
TCMP(85)	Comparing to words in a designated table	69.10	1.5
	Comparing to words in a designated table	71.50	
	Comparing ✳DM → ✳DM-designated table	123.50	
ASC(86)	Between words	56.90	1.5
	Between ✳DM	133.50	
SEND(90)	1-word transmit	563	3.75
	1000-word transmit	752	
SBS(91)	---	37.6	2.25
SBN(92)	---	---	---
RET(93)	---	45.6	13.7
WDT(94)	---	17.60	0.75
IORF(97)	1-word refresh	130.70	1.125
	30-word refresh	2.27 ms	
RECV(98)	1-word refresh	559	3.75
	1000-word refresh	764	
MCRO(99)	Designating a word parameter	91.00	1.5
	Designating a ✳DM parameter	125.80	
ASFT(—)	When resetting 1 word	43.60	1.5
Default code: (17)	When shifting 1 word using ✳DM	50.30	
	When shifting 10 word using ✳DM	68.80	
SCAN(—)	Constant for SV	31.80	1.5
Default code: (18)	✳DM for SV	51.20	
MCMP(—)	Comparing 2 words, result word	104.30	1.5
Default code: (19)	Comparing 2 ✳DM, result ✳DM	159.30	
LMSG(—)	Word for SV	104.30	1.5
Default code: (47)	✳DM for SV	159.30	

Instruction	Conditions	ON execution time (μs)	OFF execution time (μs)
TERM(—) Default code: (48)	---	16.40	1.5
CMPL(—)	When comparing words to words	51.40	1.5
Default code: (60)	When comparing ✳DM to ✳DM	85.90	
MPRF(—)	1 Unit	33.70	1.5
Default code: (61)	10 Units	74.20	
XFRB(—)	Sending 1 bit from word to word	45.50	1.5
Default code: (62)	Sending FF bits from ✳DM to ✳DM	241.90	
LINE(—) Default code: (63)	When transferring from words to a constant	102.90	1.5
	When transferring from words to a word	106.40	
	When transferring ✳DM to ✳DM	293.80	
COLM(—) Default code: (64)	When transferring from a constant to words	115.20	1.5
	When transferring from a word to words	118.70	
	When transferring ✳DM to ✳DM	303.10	
SEC(—)	DM to DM	78.50	1.5
Default code: (65)	✳DM to ✳DM	112.90	
HMS(—)	DM to DM	80.00	1.5
Default code: (66)	✳DM to ✳DM	114.50	
BCNT(—)	Constant for SV	69.56	1.5
Default code: (67)	✳DM for SV	37.52 ms	
BCMP(—) Default code: (68)	Comparing constant to word-designated table	105.00	1.5
	To a word after comparing with a word	107.40	
	Comparing ✳DM → ✳DM-designated table	166.50	
APR(—)	SIN designation	43.90	1.5
Default code: (69)	✳DM for SV	740.70	
TTIM(—) Default code: (87)	Setting to a constant	43.50	Constant input OFF: 36.9 R: 38.4 IL: 36.8 JPM: 36.8
	Setting to ✳DM	81.70	Constant input OFF: 75.1 R: 76.6 IL: 75.0 JPM: 75.1
ZCP(—) Default code: (88)	Comparing a constant to a word	35.40	1.5
	Comparing a word to a word	38.00	
	Comparing ✳DM to a ✳DM	89.70	
INT(—)	Word for SV	21.70 to 47.40	1.5
Default code: (89)	✳DM for SV	21.70 to 64.60	
TKY(—)	Input to DM	63.10	1.5
	Input to ✳DM	97.90	
RXD(—)	When designating a word	76.50	1.5
	When designating ✳DM	128.50	
TXD(—)	When designating a word	65.30	1.5
	When designating ✳DM	125.30	
7SEG(—)	Word-designated 4 digits	49.10 to 55.40	21.70

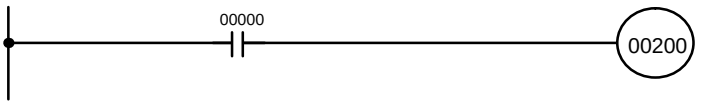
Instruction	Conditions	ON execution time (μs)	OFF execution time (μs)
	✳DM-designated 4 digits	66.60 to 72.80	
	Word-designated 8 digits	56.70 to 64.80	
	✳DM-designated 8 digits	74.20 to 82.30	
FPD(—)	Word designation, code output	121.00 to 147.50	17.30
	✳DM designation, message output	170.50 to 228.80	
SRCH(—)	Constant for SV	78.50	1.5
	✳DM for SV	2.11 ms	
	✳DM for SV	12.10 ms	
MAX(—)	DM search	57.20	1.5
	✳DM search	2.05 ms	
MIN(—)	DM search	57.20	1.5
	✳DM search	2.05 ms	
SUM(—)	DM add	60.10	1.5
	✳DM add	1.80 ms	
FCS(—)	Add a word → word	52.90	1.5
	Add 999 words → ✳DM	1.41 ms	
HEX(—)	DM conversion	66.70	1.5
	✳DM conversion	169.90	
AVG(—)	Average of an operation	61.40	19.7
	Average of 64 operations	223.70	
PID(—)	When designating a word	83.00	1.5
	When designating ✳DM	138.00	
XDMR(—)	Constant for SV	74.20	1.5
	Word for SV	2.32 ms	
	✳DM for SV	6.89 ms	
MTR(—)	Input to DM	54.60 to 63.60	21.7
	Input to ✳DM	72.00 to 81.20	
ADBL(—)	DM + DM → DM	72.20	1.5
	✳DM + ✳DM → ✳DM	123.30	
SBBL(—)	DM – DM → DM	71.70	1.5
	✳DM – ✳DM → ✳DM	123.30	
MBS(—)	Constant x word → word	50.20	1.5
	DM x DM → DM	52.80	
	✳DM x ✳DM → ✳DM	104.30	
DBS(—)	Constant ÷ word → word	51.20	1.5
	DM ÷ DM → DM	53.70	
	✳DM ÷ ✳DM → ✳DM	106.20	
MBSL(—)	DM x DM → DM	81.90	1.5
	✳DM x ✳DM → ✳DM	133.50	
DBSL(—)	DM ÷ DM → DM	90.70	1.5
	✳DM ÷ ✳DM → ✳DM	143.70	
CPS(—)	When comparing two constants	34.20	1.5
	When comparing DM to DM	29.70	
	When comparing DM to ✳DM	64.80	
CPSL(—)	When comparing two DM	50.90	1.5
	When comparing two ✳DM	86.10	

Instruction	Conditions	ON execution time (μs)	OFF execution time (μs)
NEG(—)	When converting a constant to a word	34.90	1.5
	When converting a word to a word	37.50	
	When converting ✳DM to ✳DM	72.10	
NEGL(—)	When converting a word to a word	47.00	1.5
	When converting ✳DM to ✳DM	81.90	
ZCPL(—)	When comparing two words	71.90	1.5
	When comparing two ✳DM	123.10	
SCL(—)	Word for SV	98.20	1.5
	✳DM for SV	150.00	
HKY(—)	When designating a word	55.7	21.7
	When designating ✳DM	72.9	
DSW(—)	DM CS output	60.20	21.1
	DM RD output	61.00	
	DM data retrieval	78.80	
	✳DM CS output	77.70	
	✳DM RD output	78.40	
	✳DM data retrieval	94.20	

6-4 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.

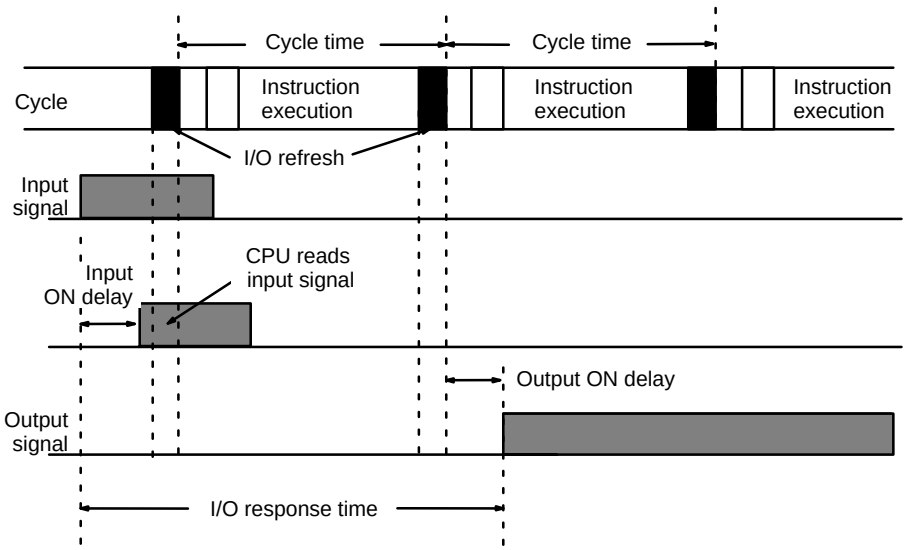
The minimum and maximum I/O response time calculations described below are for where IR 00000 is the input bit that receives the signal and IR 00200 is the output bit corresponding to the desired output point.



6-4-1 Basic Systems

Minimum I/O Response Time

The PC responds most quickly when it receives an input signal just prior to the I/O 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 I/O refresh operation would have to be repeated to refresh 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.

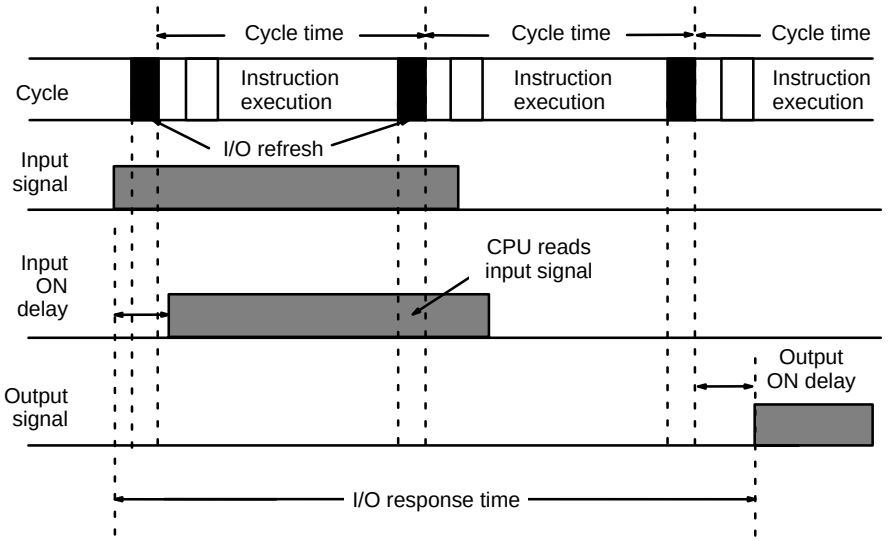


Minimum I/O response time =

Input ON delay + Cycle time + I/O refresh time + Output ON delay

Maximum I/O Response Time

The PC takes longest to respond when it receives the input signal just after the I/O 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, except that the I/O refresh time would not need to be added in because the input comes just after it rather than before it.



Maximum I/O response time =

Input ON delay + (Cycle time x 2) + Output ON delay

Calculation Example

The data in the following table would produce the minimum and maximum cycle times shown calculated below.

Item	Time
Input ON-delay	1.5 ms
Output ON-delay	15 ms
Cycle time	20 ms

Minimum I/O response time = 1.5 + 20 + 15 = 36.5 ms

Maximum I/O response time = 1.5 + (20 x 2) +15 = 56.5 ms

Note In this example the I/O refresh time is negligible has not been included in the minimum I/O response time.

6-4-2 Remote I/O Systems

With C200HS Remote I/O Systems, only the cycle time of the PC needs to be considered in computing the I/O response times as long as the remote I/O transmission time is negligible and smaller than the cycle time. The cycle time, however, is increased by the presence of the Remote I/O System.

The processing that determines and the methods for calculating maximum and minimum response times from input to output are provided in this section. Calculations assume that both the input and the output are located on Slave Racks in a Remote I/O System, but the calculations are the same for I/O points on Optical I/O Units, I/O Link Units, I/O Terminals, etc.

Although more precise equations are possible if required, equations used for the following calculations do not consider fractions of a scan.

In looking at the following timing charts, it is important to remember the sequence in which processing occurs during the PC scan, particular that inputs will not produce programmed actions until the program has been executed.

When calculating the response times involving inputs and outputs from another CPU connected by an I/O Link Unit, the cycle time of the controlling CPU and the cycle time of the PC to which the I/O Link Unit is mounted must both be considered.

⚠ Caution Noise may increase I/O delays.

Remote I/O Transmission Times

The remote I/O transmission time is computed as follows:

T_{RM} = Total Slave transmission time for one Master

= $\Sigma T_{RT} + T_{TT}$

T_{RT} = Transmission time for each Slave

= 1.4 ms + (0.2 ms x n)

Where n = number of I/O words on the Slave Rack

T_{TT} = Optical I/O Unit/I/O Terminal transmission time

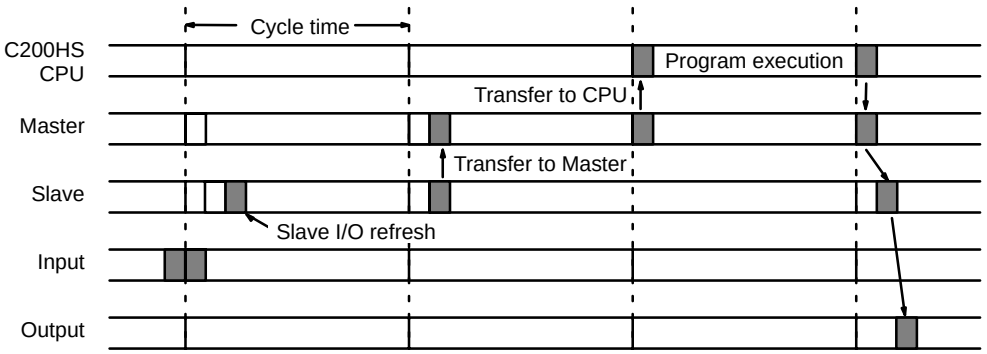
= 2 ms x m

Where m = number of Optical I/O Units/I/O Terminals

Minimum I/O Response Time

The minimum response time occurs when all signals are processed as soon as they are received. Here, three scans are required so that the program is executed, as shown in the following diagram.

Time = Input ON delay + cycle time x 3 + output ON delay

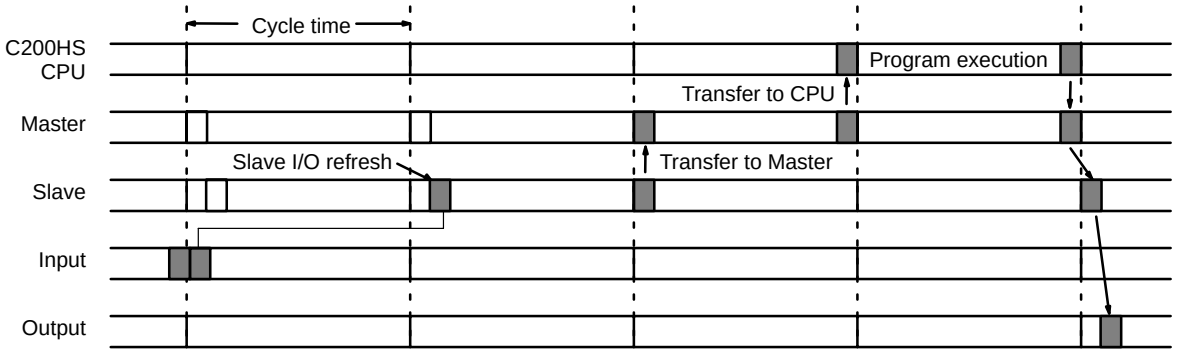


Maximum I/O Response Time

The maximum response time occurs when the input just misses the program execution portion of the scan, meaning that processing must wait for the next transmission and then the next (i.e., the fourth) scan.

Time = Input ON delay + cycle time x 4 + output ON delay

Note Use the maximum cycle time output to AR 26 in computing the maximum I/O response time.



Example Calculations

Calculations would be as shown below for an input ON delay of 1.5 ms, an output ON delay of 15 ms, and a cycle time of 20 ms.

Minimum I/O Response Time

Time = 1.5 ms + (20 ms x 3) + 15 ms = 76.5 ms

Maximum I/O Response Time

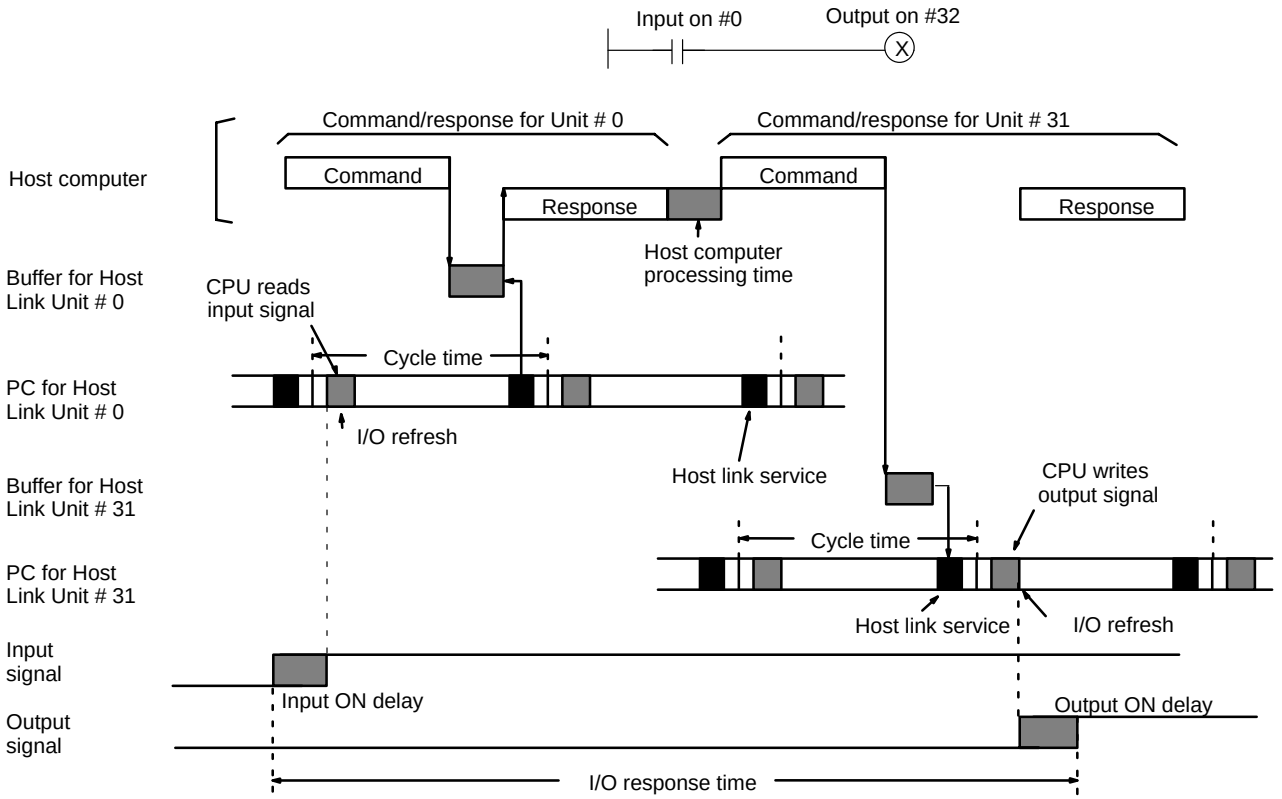
Time = 1.5 ms + (20 ms x 4) + 15 ms = 96.5 ms

Note

1. The cycle time may be less than or equal to the remote I/O transmission time when there are Special I/O Units on Slave Racks. If this is the case, there may be cycles when I/O is not refreshed between the Master and the C200HS CPU.
2. Refreshing is performed for Masters only once per cycle, and then only after confirming completion of the remote cycle.
3. The short duration of ON/OFF status produced by differentiated instructions can cause inaccurate signals when dealing with Remote I/O Systems unless appropriate programming steps are taken.

6-4-3 Host Link Systems

The following diagram illustrates the processing that takes place when an input on one PC is transferred through the Host Link System to turn ON an output on another PC. Refer to Host Link System documentation for further details.



The equations used to calculate the minimum and maximum cycle times are given below. The number of cycles required for each PC depends on the amount of data being read/written.

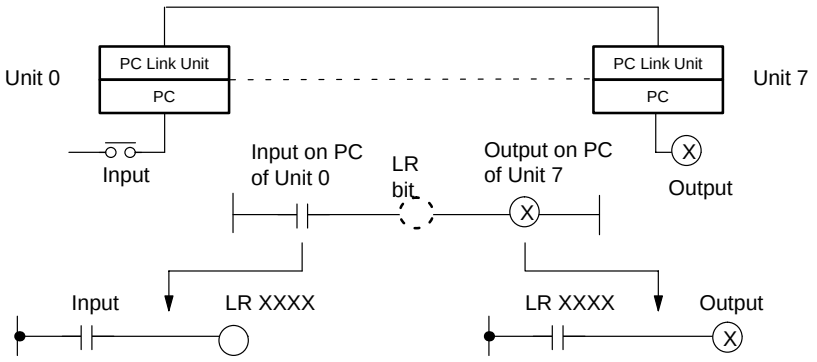
Minimum response time = Input ON delay + Command transmission time + (Cycle time of PC for Unit #0 x 3) + Response transmission time + Host computer processing time + Command transmission time + (Cycle time of PC for Unit #31 x 3) + Output ON delay

Maximum response time = Input ON delay + Command transmission time + (Cycle time of PC for Unit #0 x 10) + Response transmission time + Host computer processing time + Command transmission time + (Cycle time of PC for Unit #31 x 10) + Output ON delay

6-4-4 PC Link Systems

The processing that determines and the methods for calculating maximum and minimum response times from input to output are provided in this subsection. The following System and I/O program steps will be used in all examples below. This System contains eight PC Link Units.

In looking at the following timing charts, it is important to remember the sequence processing occurs during the PC scan, particular that inputs will not produce programmed-actions until the program has been execution.



⚠ Caution Noise may increase I/O delays.

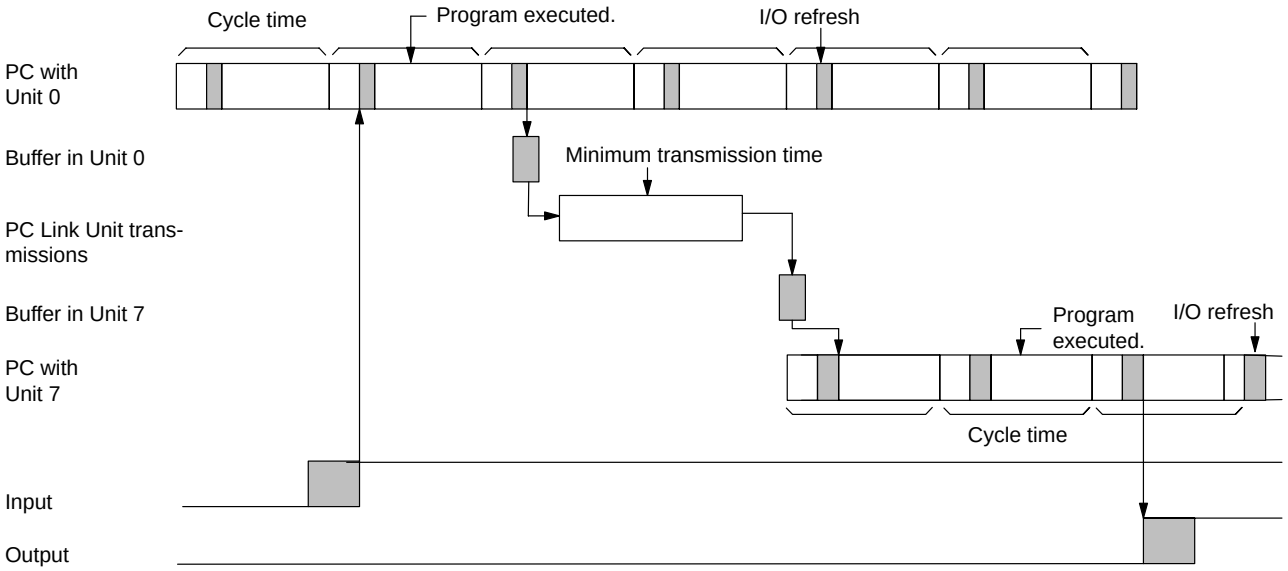
PC Link Conditions

The PC Link System used in this example consists of the following:

- No. of PCs linked: 8
- No. of LR points linked: 128 per PC
- Maximum PC: 8
- LR points used: 1,024

Minimum Response Time

The following illustrates the data flow that will produce the minimum response time, i.e., the time that results when all signals and data transmissions are processed as soon as they occur.



The equation for minimum I/O response time is thus as follows:

Response time = Input ON delay + Cycle time of PC of Unit 0 + Minimum transmission time + (Cycle time of PC of Unit 7 x 2) + Output ON delay

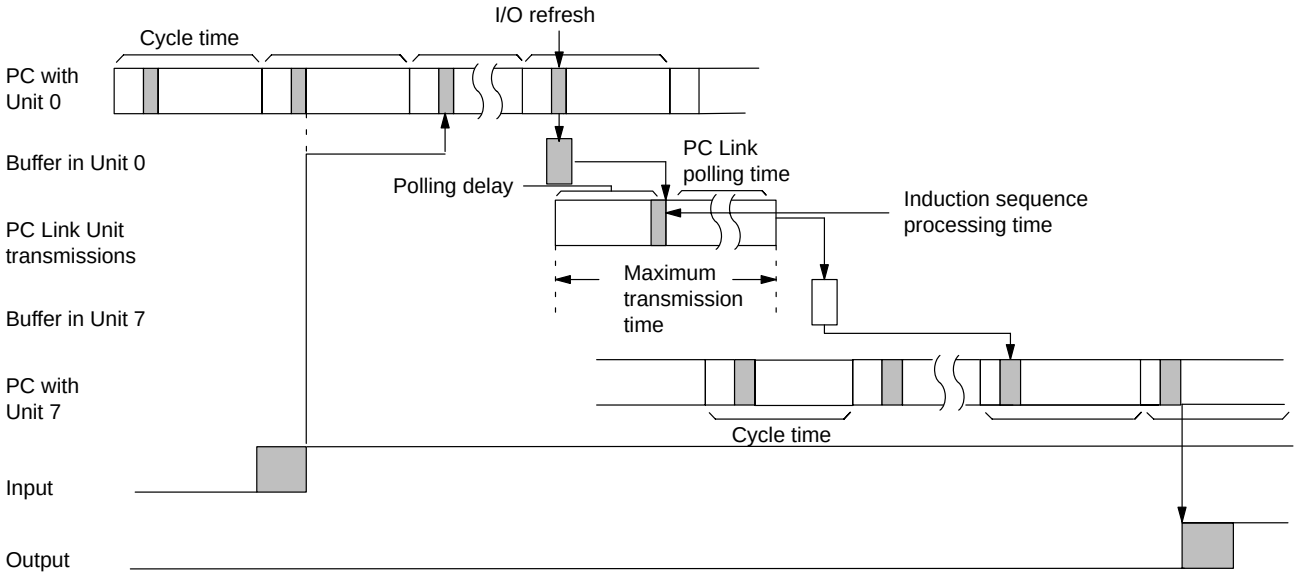
Inserting the following values into this equation produces a minimum I/O response time of 149.3 ms.

Input ON delay:	1.5 ms
Output ON delay:	15 ms
Cycle time for PC of Unit 0:	20 ms
Cycle time for PC of Unit 7:	50 ms

Maximum Response Time

The following diagram illustrates the data flow that will produce the maximum response time. Delays occur because signals or data is received just after they would be processed or because data is sent during processing. In either case, processing must wait until the next scan/polling cycle.

First output to the buffer in the polling unit is delayed by the setting of the number of LR bits to be refreshed each scan. A similar delay is present when the LR data reaches Unit 7. The polling delay is the result of the LR data in its PC being updated immediately after the previous was sent to the buffer in the PC Link Unit, cause a delay until the next polling cycle. One more polling cycle is then required before the data reaches the buffer in PC Link Unit 7.



The equation for maximum I/O response time is thus as follows:

Response time = Input ON delay + [Cycle time of PC of Unit 0 x (Number of LR transfer bits ÷ I/O refresh bits)] + α + (PC Link polling time + Induction sequence processing time) + {Cycle time of PC of Unit 7 x [(Number of LR transfer bits ÷ I/O refresh bits) x 2 + 1]} + β + Output ON delay

If cycle time of PC of Unit 0 > PC Link polling time, α = cycle time of PC of Unit 0. If cycle time of PC of Unit 0 < PC Link polling time, α = PC Link polling time.

If cycle time of PC of Unit 7 > PC Link polling time, β = cycle time of PC of Unit 7. If cycle time of PC of Unit 7 < PC Link polling time, β = PC Link polling time.

Inserting the following values into this equation produces a maximum I/O response time of 661.3 ms.

Input ON delay:	1.5 ms
Output ON delay:	15 ms
Cycle time for PC of Unit 0:	20 ms
Cycle time for PC of Unit 7:	50 ms
PC Link polling time:	2.8 ms x 8 PCs + 10 ms = 32.4 ms

Induction sequence processing: 15 ms x (8 PCs – 8 PCs) = 0 ms

I/O refresh bits for Unit 0 256

I/O refresh bits for Unit 7 256

Reducing Response Time

IORF(97) can be used in programming to shorten the I/O response time greater than is possible by setting a high number of refresh bits. (Remember, increasing the number of refresh bits set on the back-panel LED shortens response time, but increases the cycle time of the PC.)

The following calculations for the maximum cycle time use the same example System configuration as that used in 5–2 Response Times. In programming the PCs for PC Link Units #0 and #7, IORF(97) is executed during every PC scan for the PC Link Units. The basic equation for the maximum I/O response time is as follows:

Response time = Input ON delay + [Cycle time of PC of Unit 0 x (Number of LR transfer bits ÷ Number of I/O refresh bits ÷ 2)] + α + PC Link polling time + Induction sequence processing time + {Cycle time of PC of Unit 7 x [(Number of LR transfer bits ÷ Number of I/O refresh bits ÷ 2) x 2 + 1]} + β + Output ON delay

If cycle time of PC of Unit 0 > PC Link polling time, α = cycle time of PC of Unit 0. If cycle time of PC of Unit 0 < PC Link polling time, α = PC Link polling time.

If cycle time of PC of Unit 7 > PC Link polling time, β = cycle time of PC of Unit 7. If cycle time of PC of Unit 7 < PC Link polling time, β = PC Link polling time.

The required data from the example System configuration is as follows:

Input ON delay	1.5 ms
Output ON delay	15 ms
Cycle time of PC of Unit 0	20 ms + 5.7 ms = 25.7 (5.7 ms required for IORF execution)
Cycle time of PC of Unit 7	50 ms + 5.7 ms = 55.7 (5.7 ms required for IORF execution)
Number of PC Link Units	8
Number of LR bits	1,024
Number of refresh bits for Unit 0	256
Number of refresh bits for Unit 7	256
PC Link polling time	2.8 ms x 8 PCs + 10 ms = 32.4 ms
Induction sequence processing time	15 ms x (8 PCs – 8 PCs) = 0 ms

Placing these values into the equation produces a maximum I/O response time of 466.9 ms, approximately 200 ms shorter than when IORF is not used.

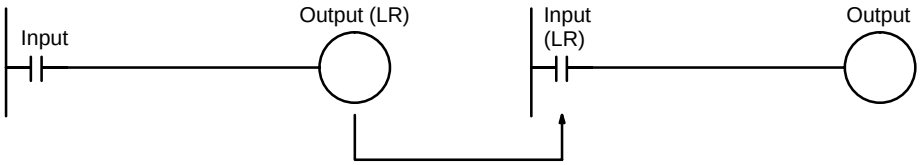
6-4-5 One-to-one Link I/O Response Time

When two C200HSs are linked one-to-one, the I/O response time is the time required for an input executed at one of the C200HSs to be output to the other C200HS by means of one-to-one link communications.

One-to-one link communications are carried out reciprocally between the master and the slave. The respective transmission times are as shown below, depending on the number of LR words used.

Number of words used	Transmission time
64 words (LR 00 to LR 63)	39 ms
32 words (LR 00 to LR 31)	20 ms
16 words (LR 00 to LR 15)	10 ms

The minimum and maximum I/O response times are shown here, using as an example the following instructions executed at the master and the slave. In this example, communications proceed from the master to the slave.



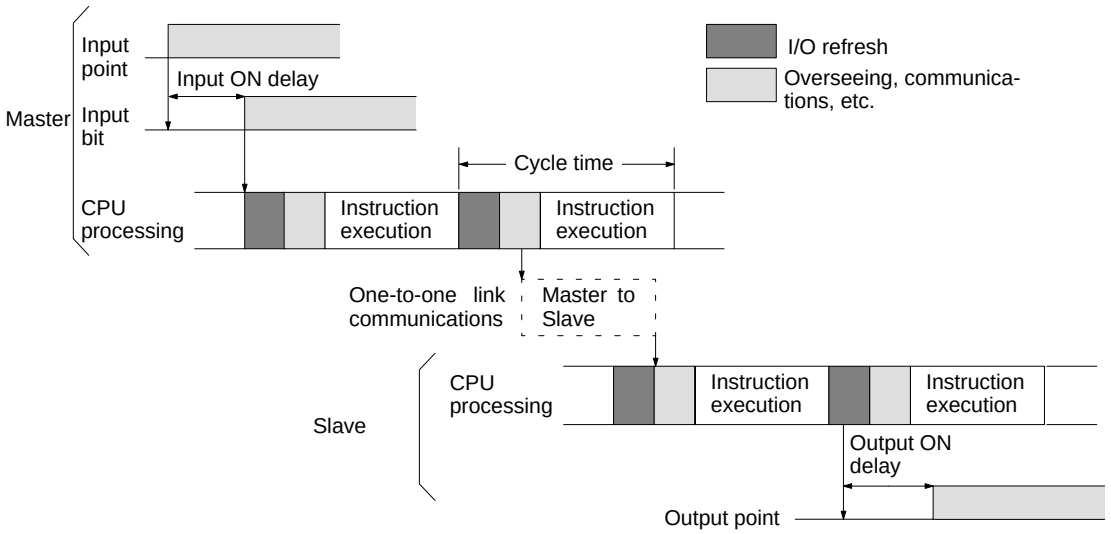
The following conditions are taken as examples for calculating the I/O response times.

Input ON delay:	8 ms
Master cycle time:	10 ms
Slave cycle time:	14 ms
Output ON delay:	10 ms
Number of LR words:	64 words

Minimum I/O Response Time The C200HS responds most quickly under the following circumstances:

1, 2, 3...

1. The C200HS receives an input signal just prior to the input refresh phase of the cycle.
2. The master to slave transmission begins immediately.
3. The slave executes communications servicing immediately after completion of communications.



The minimum I/O response time is as follows:

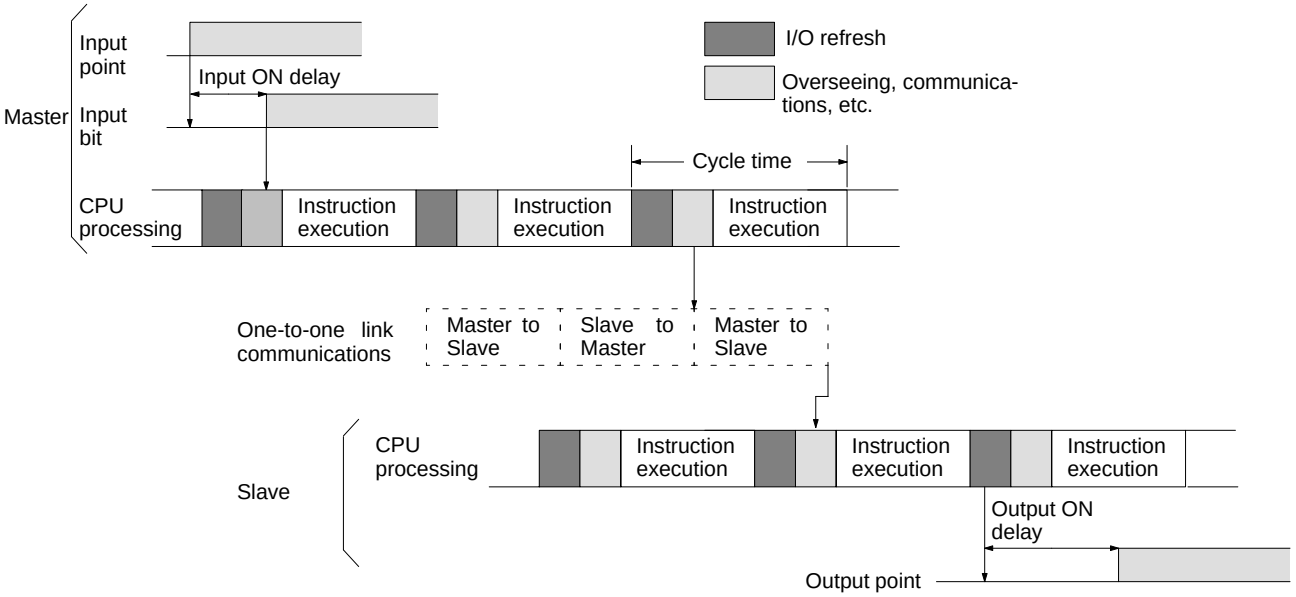
	Input ON delay:	8 ms
	Master cycle time:	10 ms
	Transmission time:	39 ms
	Slave cycle time:	15 ms
+	Output ON delay:	10 ms
	Minimum I/O response time:	82 ms

Maximum I/O Response Time The C200HS takes the longest to respond under the following circumstances:

1, 2, 3...

1. The C200HS receives an input signal just after the input refresh phase of the cycle.
2. The master to slave transmission does not begin immediately.

3. Communications are completed just after the slave executes communications servicing.



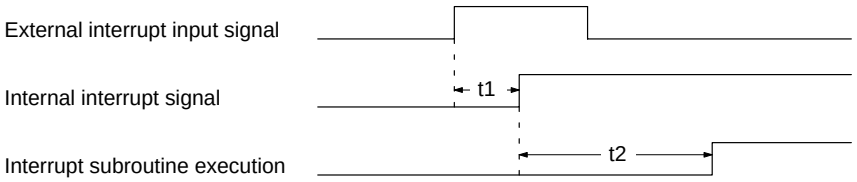
The maximum I/O response time is as follows:

	Input ON delay:	8 ms
	Master cycle time:	10 ms × 2
	Transmission time:	39 ms × 3
	Slave cycle time:	15 ms × 2
+	Output ON delay:	10 ms
	Maximum I/O response time:	185 ms

6-4-6 Interrupt Response Times

The response time from the time an interrupt input is received until the interrupt subroutine execution has been completed is described next.

Input Interrupts



t1 = ON delay of Interrupt Input Unit
t2 = Software interrupt response time

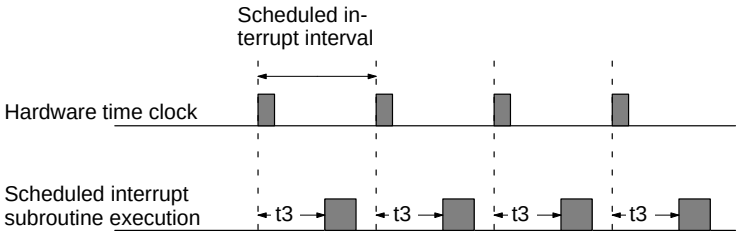
Total interrupt response time = t1 + t2

The ON delay of Interrupt Input Unit is 0.2 ms or less.

The software interrupt response time depends on the interrupt response parameter setting in DM 6620 of the PC Setup. If the DM 6620 is set for the C200H-compatible mode (0000), the software interrupt response time is less than 10 ms. If the DM 6620 is set for the C200HS mode (1xxx), the software interrupt response time is less than 1 ms. The total interrupt response time is thus as shown in the following table.

Interrupt response setting	Total interrupt response time
C200H-compatible mode	0.2 ms + (Total of following: Special I/O processing time, Remote I/O processing time, Host Link servicing time, instruction execution)
C200HS mode	1.2 ms or less

Scheduled Interrupts



t3 = Software interrupt response time

Total interrupt response time = t3 (software interrupt response time)

The software interrupt response time depends on the interrupt response parameter setting in DM 6620 of the PC Setup. If the DM 6620 is set for the C200H-compatible mode (0000), the software interrupt response time is less than 10 ms. If the DM 6620 is set for the C200HS mode (1xxx), the software interrupt response time is less than 1 ms. The total interrupt response time is thus as shown in the following table.

Interrupt response setting	Total interrupt response time
C200H-compatible mode	Total of following: Special I/O processing time, Remote I/O processing time, Host Link servicing time, instruction execution
C200HS mode	1.0 ms or less

Note

1. If there is any instruction in the program that requires longer than 10 ms to execute when using the C200H-compatible mode, the total interrupt response time will be equal to the execution time of the instruction requiring longer than 10 ms.
2. The above calculations assume that only one interrupt requires executed at any one time. If multiple interrupts are generated at the same time, execution of all but the first interrupt will go on standby, increasing the response times given above.
3. If SYSMAC NET/SYSMAC LINK Units are being serviced when an interrupt occurs, the interrupt will not be processed until SYSMAC NET/SYSMAC LINK Unit servicing has been completed. The response times in this case will be as shown in the following table and will not be affected by the interrupt response setting.

Interrupt	Total interrupt response time
Input interrupt	10.2 ms max.
Scheduled interrupt	10 ms max.

Interrupt Processing Time

The processing time from receiving an interrupt input, through program execution, and until a return is made to the original program location is described next.

The limit of the count frequency resulting from using the scheduled interrupt period or input interrupts as the count input is determined by the interrupt processing time.

Interrupt processing time =
Total interrupt response time + Interrupt program execution time + Interrupt return time

The interrupt program execution time is determined by the content of the interrupt subroutine. This time is negligible f only SBN(92) and RET(93) are executed.

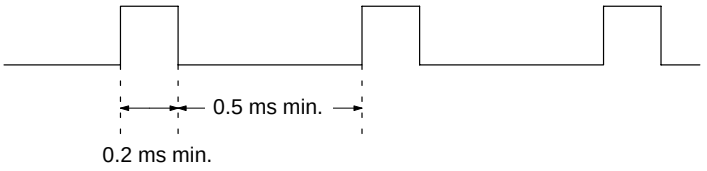
The interrupt return time is 0.04 ms.

Note

1. If there are several elements that can cause interrupts or if the interrupt period is shorted than the average interrupt processing time, the interrupt subroutine will be executed and the main program will not be executed. This will cause the cycle monitoring time to be exceeded and an FALS 9F error will be generated, stopping PC operation.
2. The maximum interrupt program execution time is contained in SR 262 and SR 263.

Interrupt Input Pulse Width

The pulse width input to Interrupt Input Units must be set to within the conditions shown in the following diagram.



ON time: 0.2 ms min.
OFF time: 0.5 ms min.

SECTION 7
Program Monitoring and Execution

This section provides the procedures for monitoring and controlling the PC through a Programming Console. Refer to the *LSS Operation Manual* for LSS procedures if you are using a computer running LSS.

7-1 Monitoring Operation and Modifying Data

The simplest form of operation monitoring is to display the address whose operand bit status is to be monitored using the Program Read or one of the search operations. As long as the operation is performed in RUN or MONITOR mode, the status of any bit displayed will be indicated.

This section provides other procedures for monitoring data as well as procedures for modifying data that already exists in a data area. Data that can be modified includes the PV (present value) and SV (set value) for any timer or counter.

All monitor operations in this section can be performed in RUN, MONITOR, or PROGRAM mode and can be cancelled by pressing CLR.

All data modification operations except for timer/counter SV changes are performed after first performing one of the monitor operations. Data modification is possible in either MONITOR or PROGRAM mode, but cannot be performed in RUN mode.

7-1-1 Bit/Word Monitor

The status of any bit or word in any data area can be monitored using the following operation. Although the operation is possible in any mode, ON/OFF status displays will be provided for bits in MONITOR or RUN mode only.

The Bit/Digit Monitor operation can be entered either from a cleared display by designating the first bit or word to be monitored or it can be entered from any address in the program by displaying the bit or word address whose status is to be monitored and pressing MONTR.

When a bit is monitored, it's ON/OFF status will be displayed (in MONITOR or RUN mode); when a word address is designated other than a timer or counter, the digit contents of the word will be displayed; and when a timer or counter number is designated, the PV of the timer will be displayed and a small box will appear if the completion flag of a timer or counter is ON. When multiple words are monitored, a caret will appear under the leftmost digit of the address designation to help distinguish between different addresses. The status of TR bits and SR flags (e.g., the arithmetic flags), cleared when END(01) is executed, cannot be monitored.

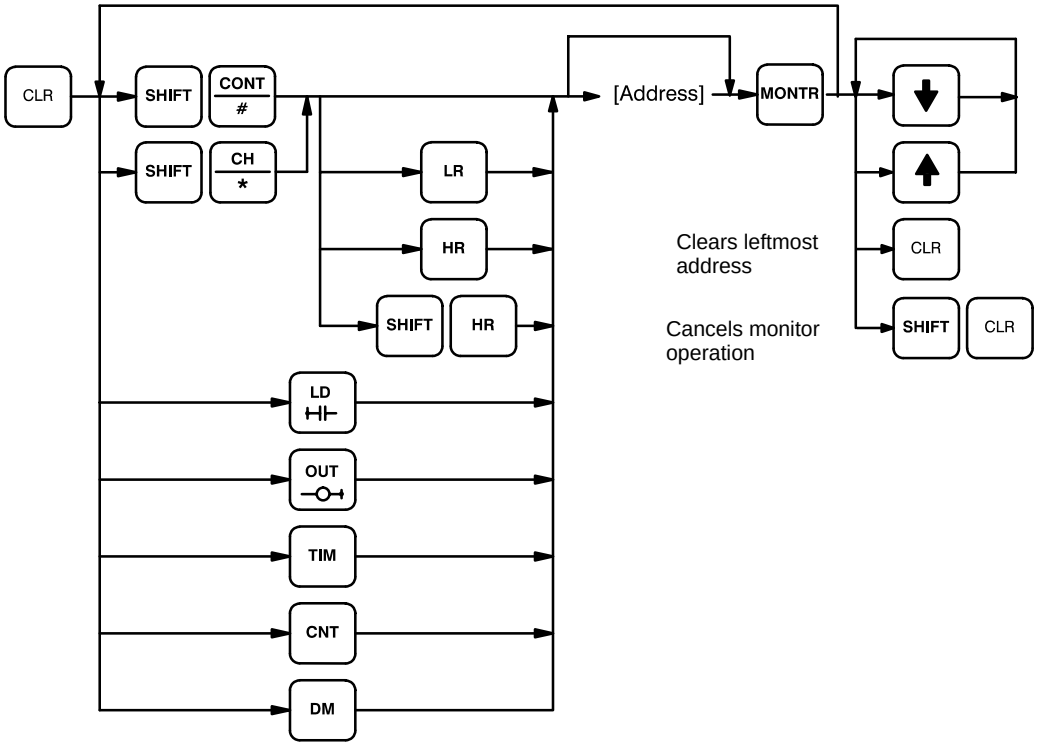
Up to six memory addresses, either bits, words, or a combination of both, can be monitored at once, although only three of these are displayed at any one time. To monitor more than one address, return to the start of the procedure and continue designating addresses. Monitoring of all designated addresses will be maintained unless more than six addresses are designated. If more than six addresses are designated, the leftmost address of those being monitored will be cancelled.

To display addresses that are being monitored but are not presently on the Programming Console display, press MONTR without designating another address. The addresses being monitored will be shifted to the right. As MONTR is pressed, the addresses being monitored will continue shifting to the right until the rightmost address is shifted back onto the display from the left.

During a monitor operation the up and down keys can be pressed to increment and decrement the leftmost address on the display and CLR can be pressed to cancel monitoring the leftmost address on the display. If the last address is cancelled, the monitor operation will be cancelled. The monitor operation can also be cancelled regardless of the number of addresses being monitored by pressing SHIFT and then CLR.

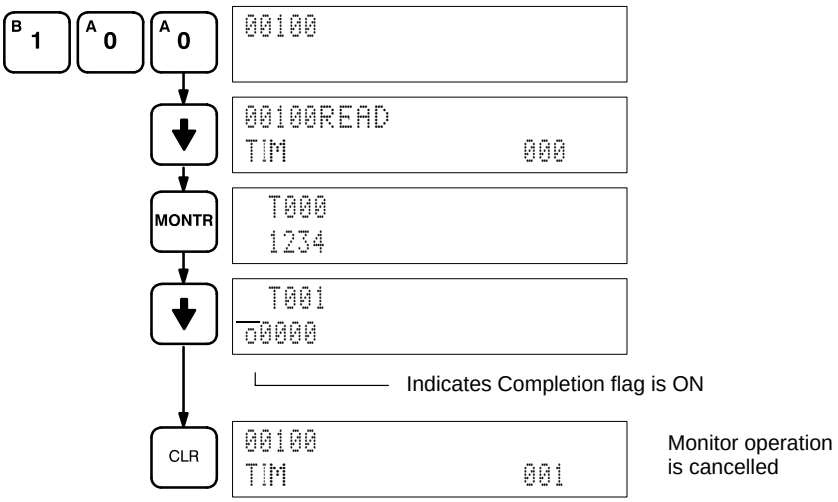
LD and OUT can be used only to designate the first address to be displayed; they cannot be used when an address is already being monitored.

Key Sequence

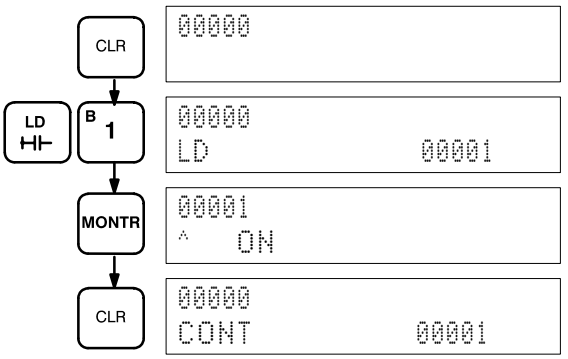


Examples The following examples show various applications of this monitor operation.

Program Read then Monitor

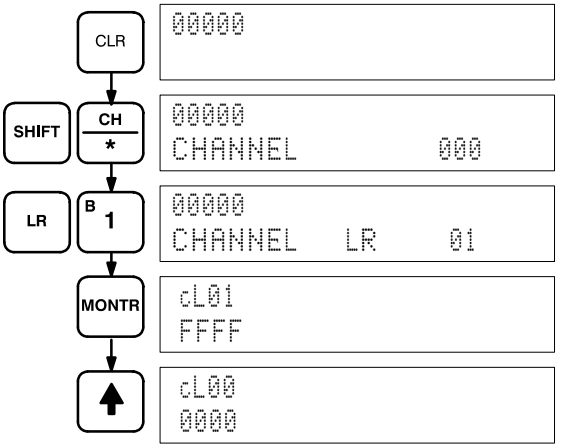


Bit Monitor

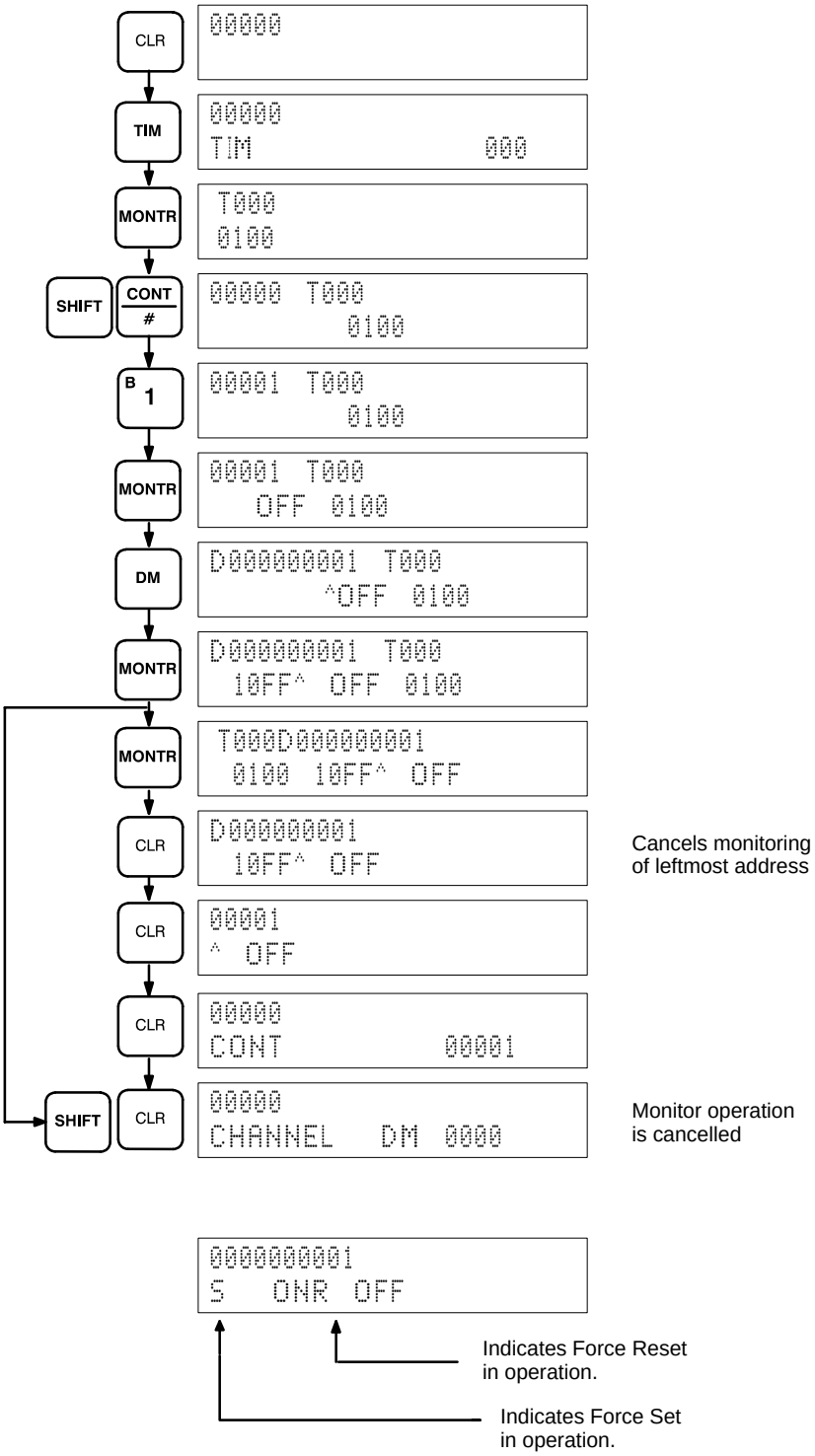


Note The status of TR bits SR flags SR 25503 to 25507 (e.g., the arithmetic flags), cleared when END(01) is executed, cannot be monitored.

Word Monitor



+Multiple Address Monitoring



7-1-2 Forced Set/Reset

When the Bit/Digit Monitor operation is being performed and a bit, timer, or counter address is leftmost on the display, PLAY/SET can be pressed to turn ON the bit, start the timer, or increment the counter and REC/RESET can be pressed to turn OFF the bit or reset the timer or counter. Timers will not operate in PROGRAM mode. SR bits cannot be turned ON and OFF with this operation.

Bit status will remain ON or OFF only as long as the key is held down; the original status will return as soon as the key is released. If a timer is started, the completion flag for it will be turned ON when SV has been reached.

SHIFT and PLAY/SET or SHIFT and REC/RESET can be pressed to maintain the status of the bit after the key is released. The bit will not return to its original status until the NOT key is pressed, or one of the following conditions is met.

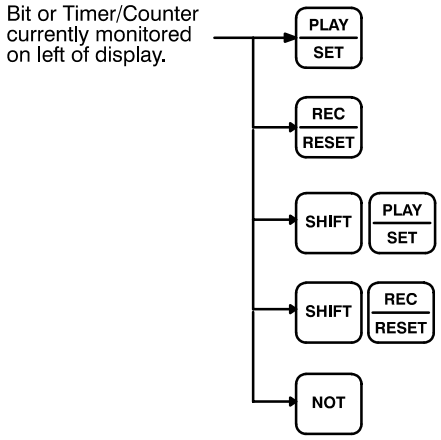
1, 2, 3...

1. The Force Status Clear operation is performed.
2. The PC mode is changed. (See note.)
3. Operation stops due to a fatal error or power interruption.
4. The I/O Table Registration operation is performed.

This operation can be used in MONITOR mode to check wiring of outputs from the PC prior to actual program execution. This operation cannot be used in RUN mode.

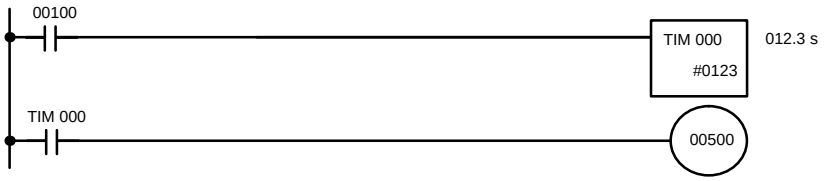
Note The forced set/reset bit status will be maintained when switching from PROGRAM to MONITOR mode if the Force Status Hold Bit is ON and DM 6601 of the PC Setup has been set maintain the bit's status. Refer to *3-6-4 PC Setup* for details.

Key Sequence



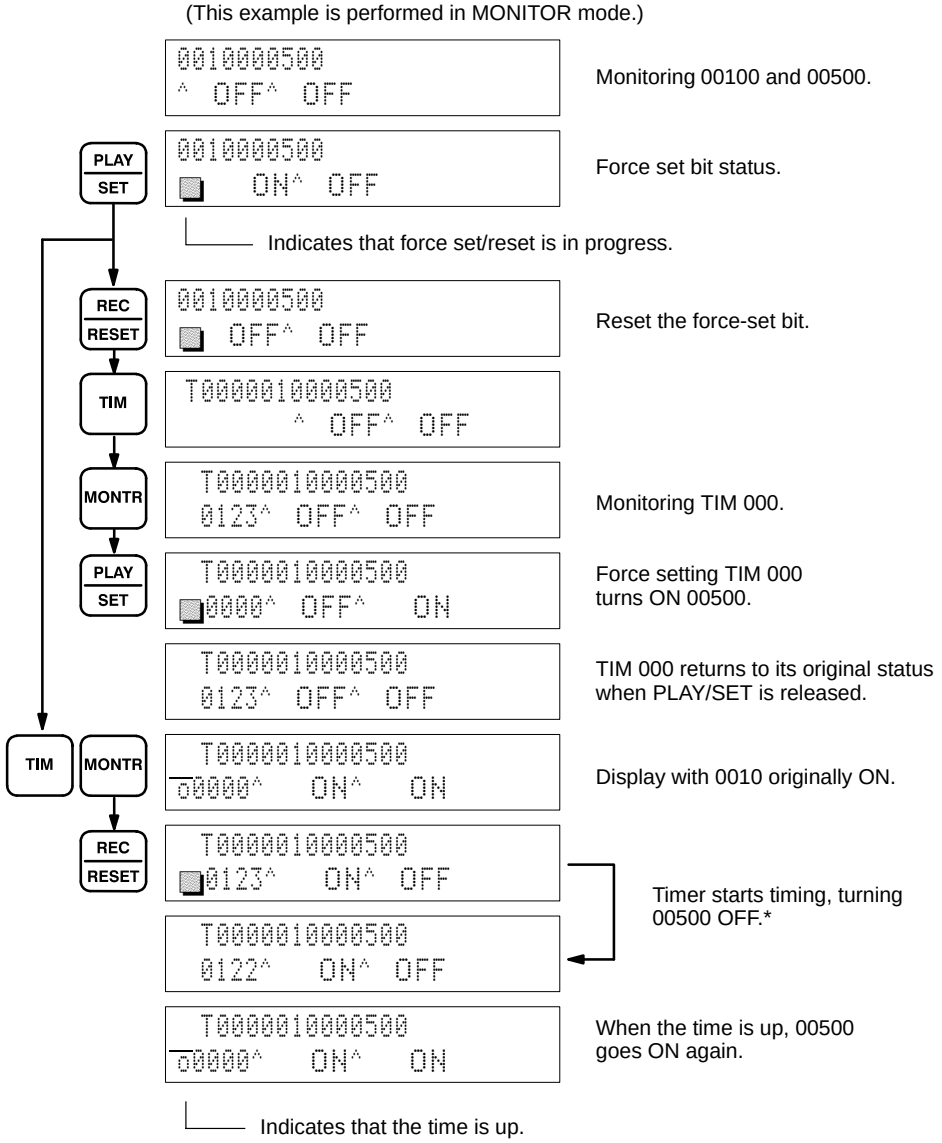
Example

The following example shows how either bits or timers can be controlled with the Force Set/Reset operation. The displays shown below are for the following program section.



Address	Instruction	Data	
00200	LD		00100
00201	TIM		000
		#	0123
00202	LD	TIM	000
00203	OUT		00500

The following displays show what happens when TIM 000 is set with 00100 OFF (i.e., 00500 is turned ON) and what happens when TIM 000 is reset with 00100 ON (i.e., timer starts operation, turning OFF 00500, which is turned back ON when the timer has finished counting down the SV).

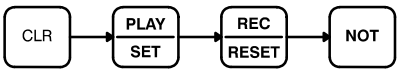


*Timing not done in PROGRAM mode.

7-1-3 Forced Set/Reset Cancel

This operation restores the status of all bits in the I/O, IR, TIM, CNT, HR, AR, or LR areas which have been force set or reset. It can be performed in PROGRAM or MONITOR mode.

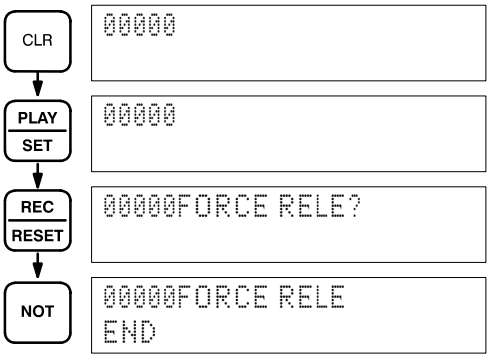
Key Sequence



When the PLAY/SET and REC/RESET keys are pressed, a beeper will sound. If you mistakenly press the wrong key, then press CLR and start again from the beginning.

Example

The following example shows the displays that appear when Restore Status is carried out normally.



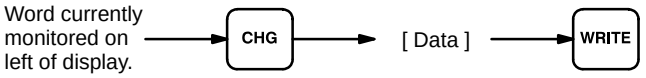
7-1-4 Hexadecimal/BCD Data Modification

When the Bit/Digit Monitor operation is being performed and a BCD or hexadecimal value is leftmost on the display, CHG can be input to change the value. SR words cannot be changed.

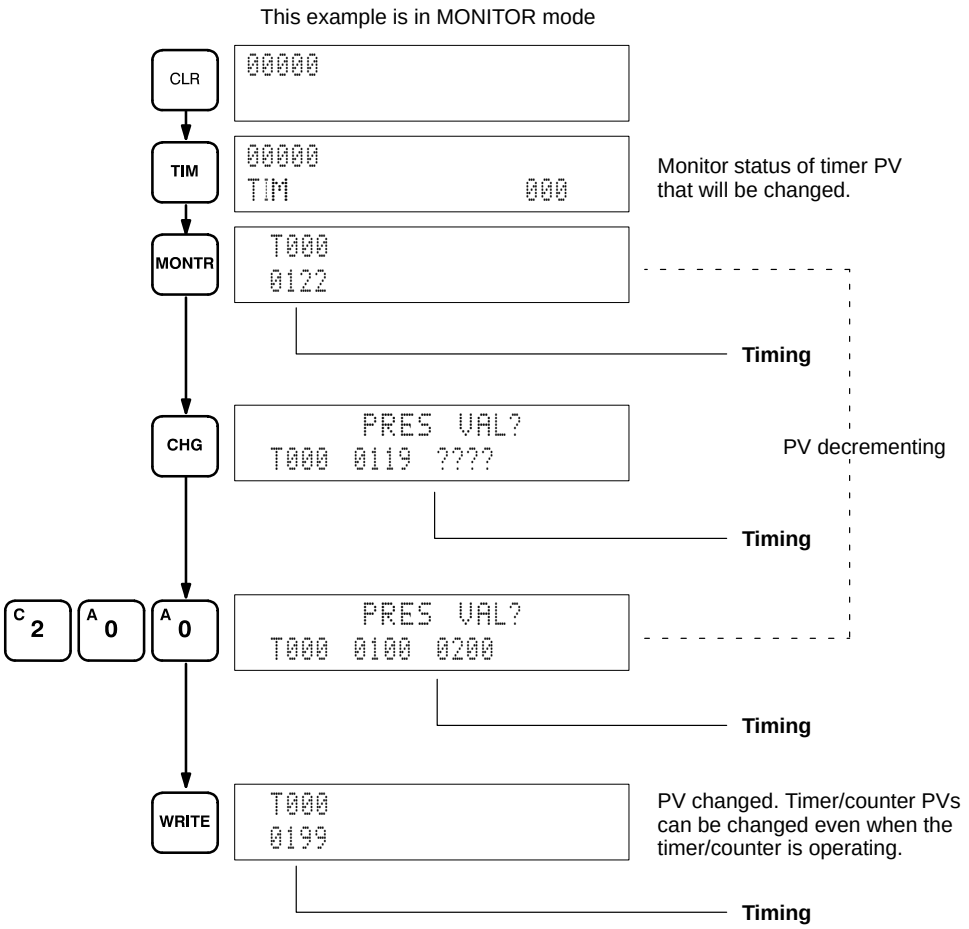
If a timer or counter is leftmost on the display, the PV will be displayed and will be the value changed. See *7-1-13 Changing Timer/Counter SV* for the procedure to change SV. PV can be changed in MONITOR mode only when the timer or counter is operating.

To change contents of the leftmost word address, press CHG, input the desired value, and press WRITE

Key Sequence



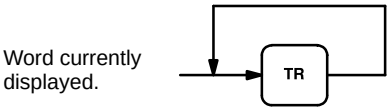
Example The following example shows the effects of changing the PV of a timer.



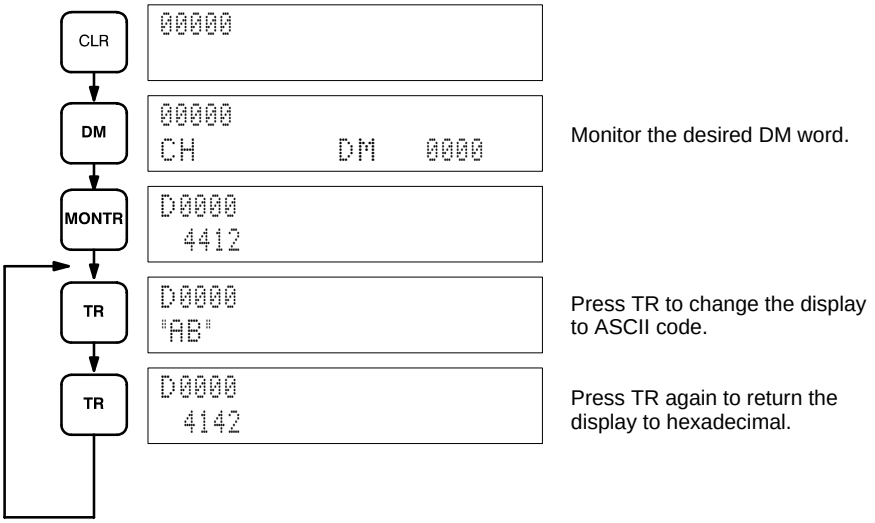
7-1-5 Hex/ASCII Display Change

This operation converts DM data displays from 4-digit hexadecimal data to ASCII and vice versa.

Key Sequence



Example

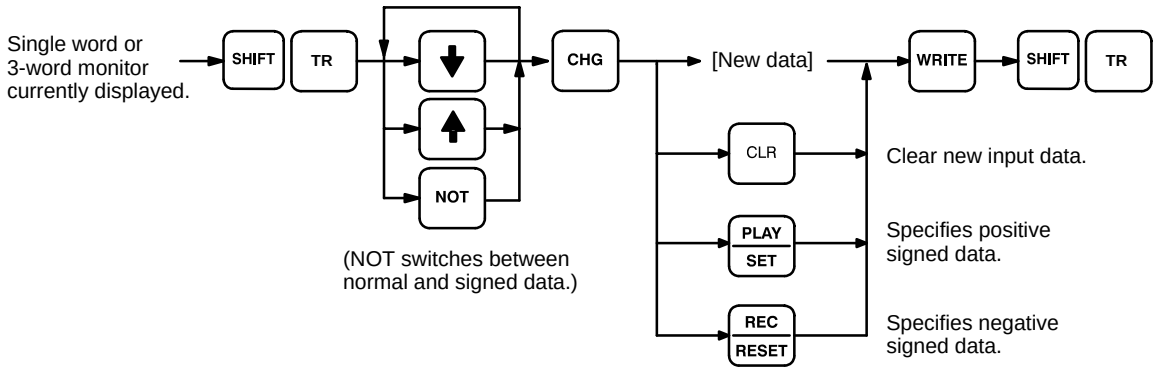


7-1-6 4-digit Hex/Decimal Display Change

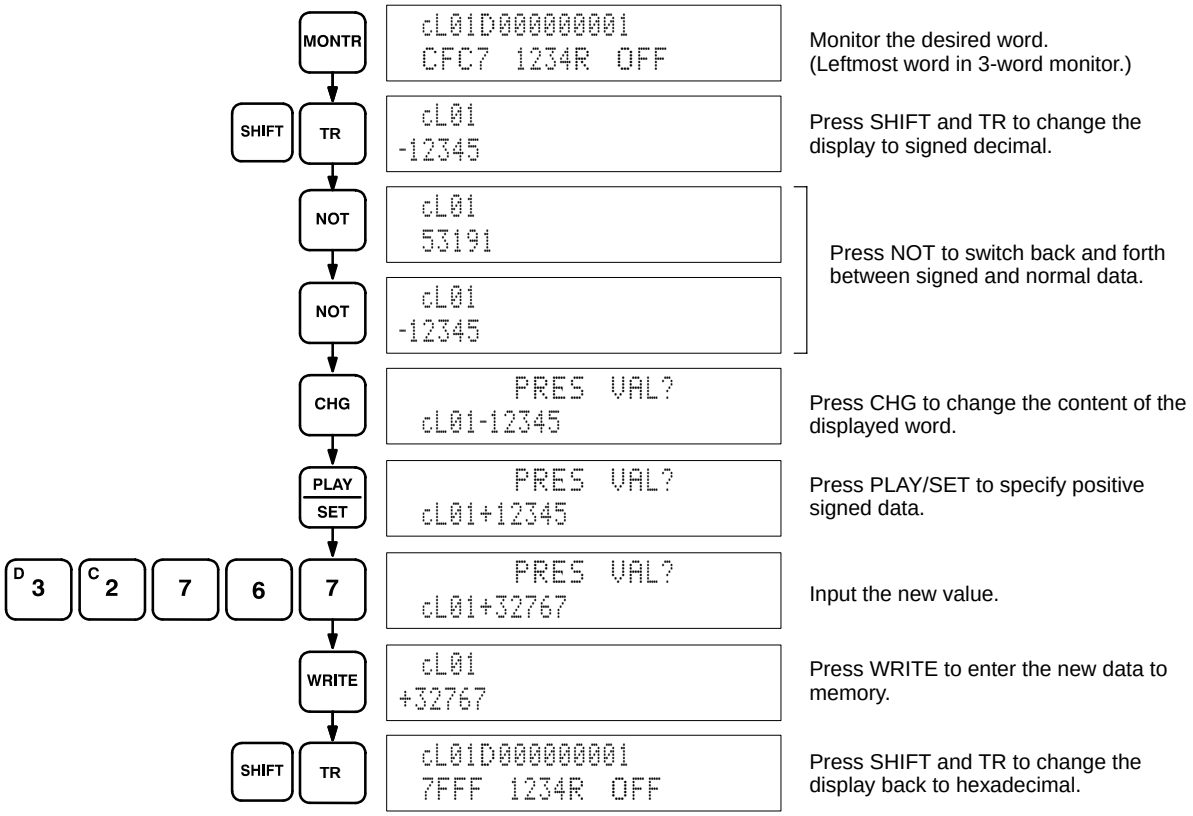
This operation converts data displays from normal or signed 4-digit hexadecimal data to decimal and vice versa.

Decimal values from 0 to 65,535 are valid when inputting normal 4-digit hexadecimal data, and decimal values from –32,768 to +32,767 are valid when inputting signed 4-digit hexadecimal data.

Key Sequence



Example

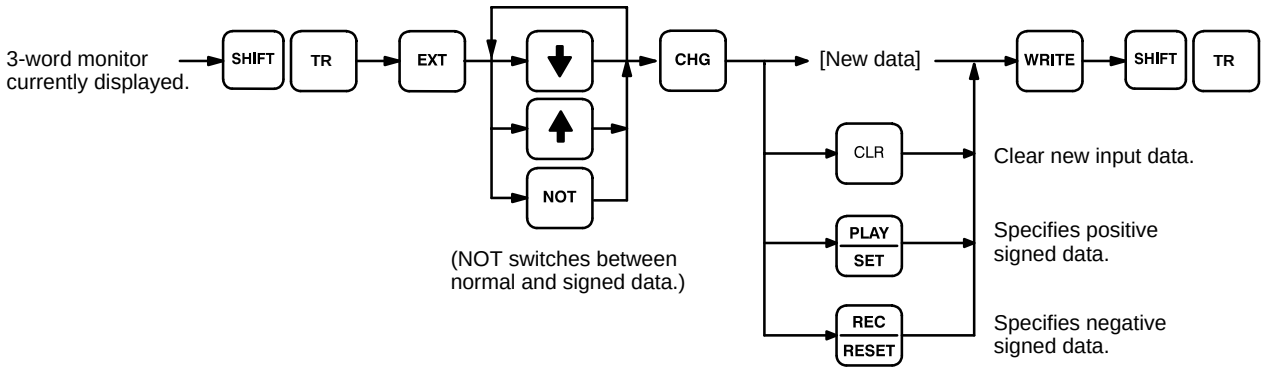


7-1-7 8-digit Hex/Decimal Display Change

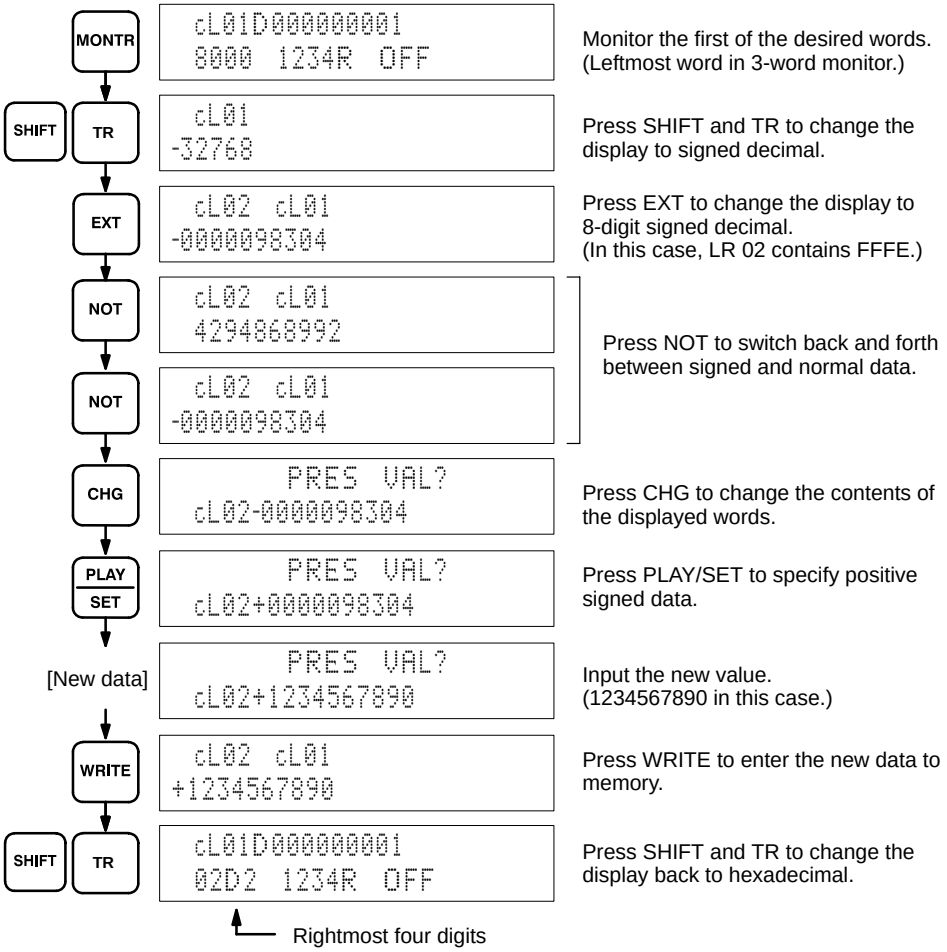
This operation converts data displays from normal or signed, 4 or 8-digit hexadecimal data to decimal and vice versa.

Decimal values from 0 to 4,294,967,295 are valid when inputting normal 8-digit hexadecimal data, and decimal values from –2,147,483,648 to +2,147,483,647 are valid when inputting signed 8-digit hexadecimal data.

Key Sequence



Example

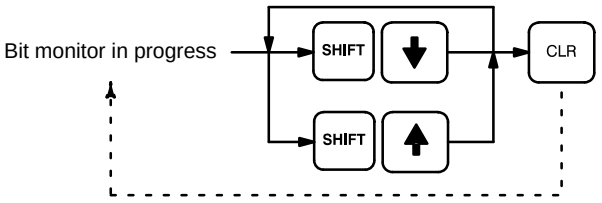


7-1-8 Differentiation Monitor

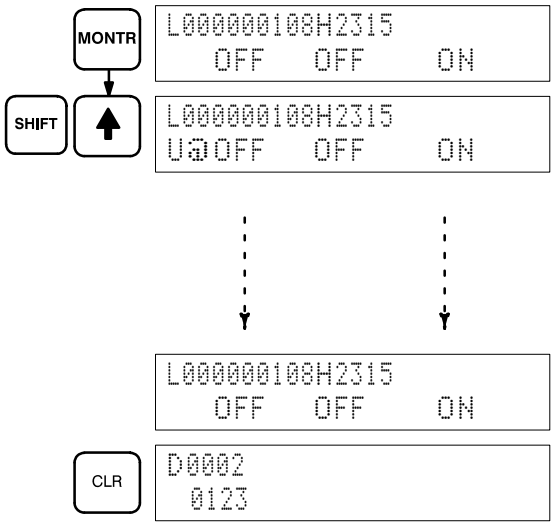
This operation can be used to monitor the up or down differentiation status of bits in the IR, SR, AR, LR, HR, and TC areas. To monitor up or down differentiation status, display the desired bit leftmost on the bit monitor display, and then press SHIFT and the Up or Down Arrow Key.

A CLR entry changes the Differentiation Monitor operation back to a normal bit monitor display.

Key Sequence



Example



Monitor the desired bit so that it is leftmost on the screen.

Press SHIFT and Up Arrow to specify up differentiation (U@).

(Press SHIFT and Down Arrow to specify down differentiation (D@).

The buzzer will sound when up (U@) or down (D@) differentiation is detected.

The original bit monitor display will return when differentiation monitoring is completed.

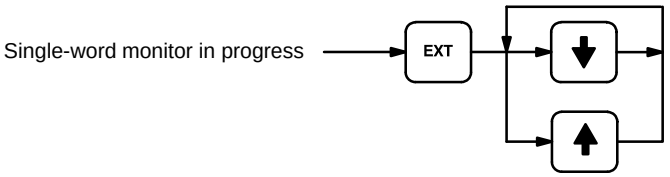
Press CLR to cancel differentiation monitoring and return to the original bit monitor display.

7-1-9 3-word Monitor

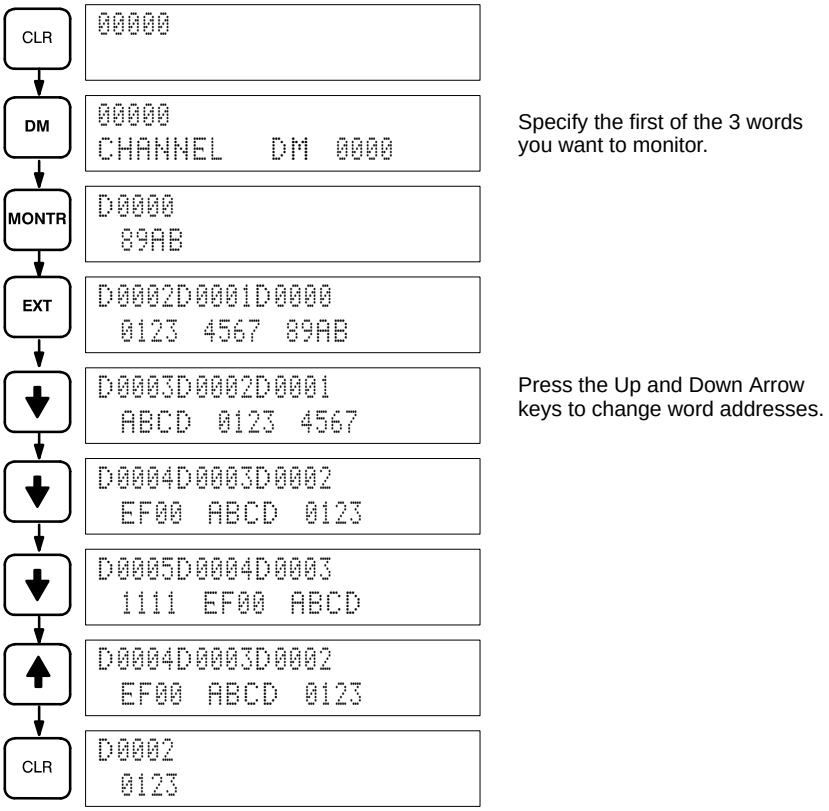
To monitor three consecutive words together, specify the lowest numbered word, press MONTR, and then press EXT to display the data contents of the specified word and the two words that follow it.

A CLR entry changes the Three-word Monitor operation to a single-word display.

Key Sequence



Example

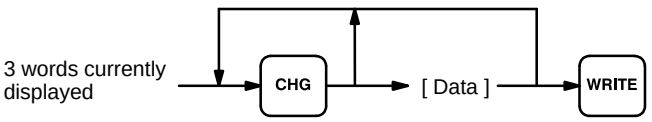


7-1-10 3-word Data Modification

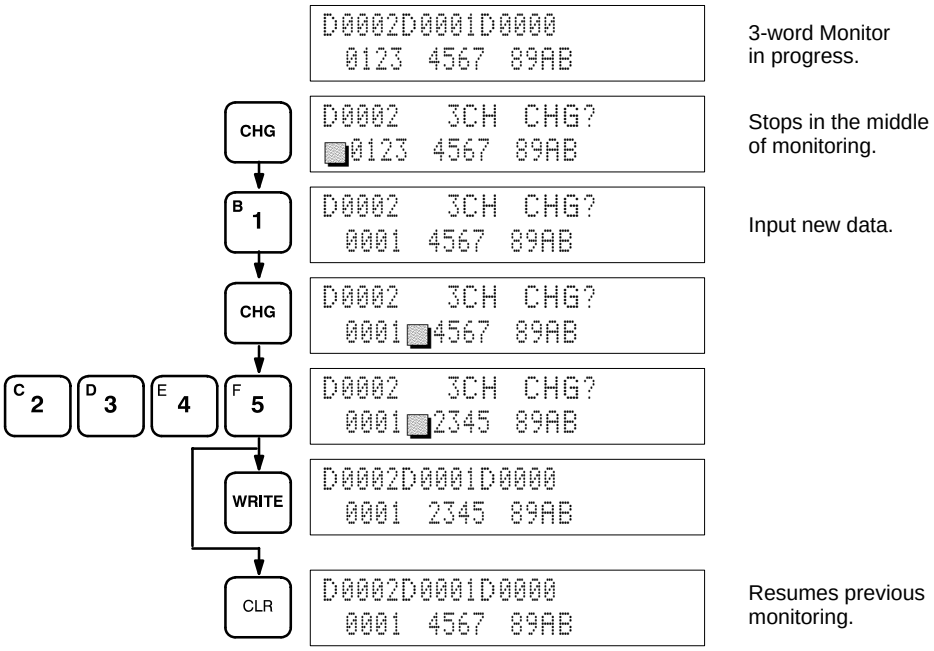
This operation changes the contents of a word during the 3-Word Monitor operation. The blinking square indicates where the data can be changed. After the new data value is keyed in, pressing WRITE causes the original data to be overwritten with the new data. If CLR is pressed before WRITE, the change operation will be cancelled and the previous 3-word Monitor operation will resume.

This operation cannot be used to change SR 253 through SR 255. Only those words displayed on the 3-word Monitor display can be changed.

Key Sequence



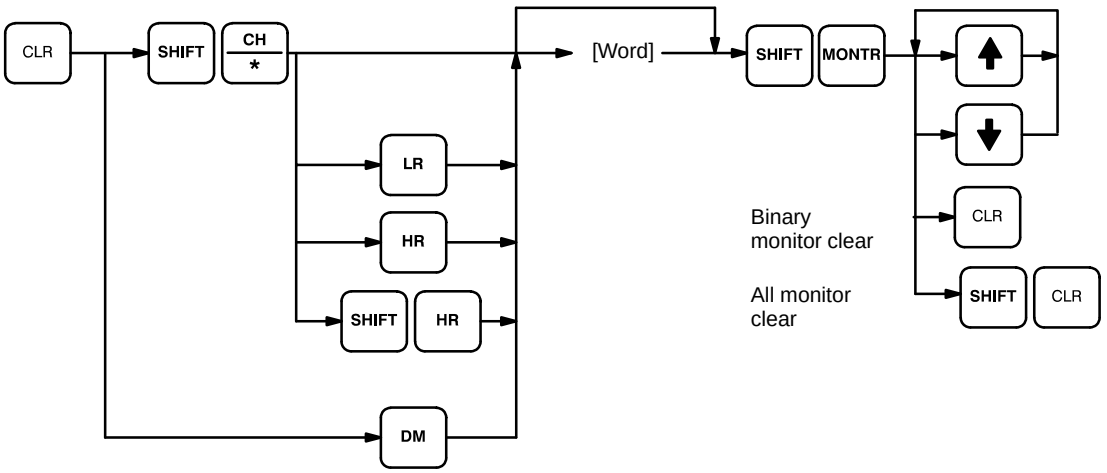
Example



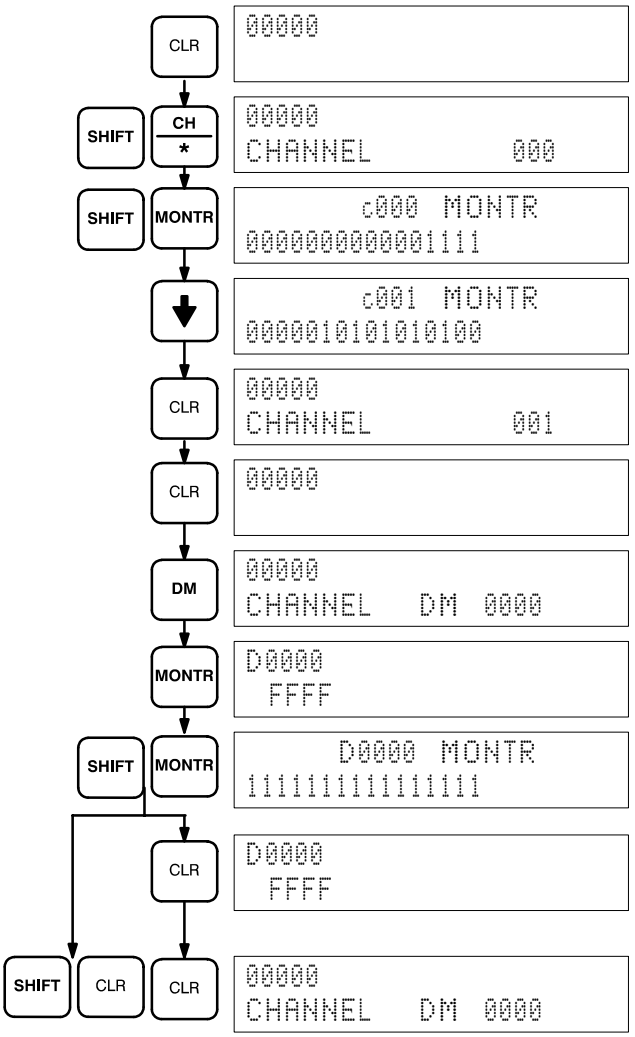
7-1-11 Binary Monitor

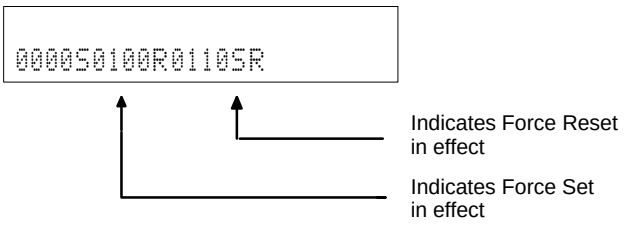
You can specify that the contents of a monitored word be displayed in binary by pressing SHIFT and MONTR after the word address has been input. Words can be successively monitored by using the up and down keys to increment and decrement the displayed word address. To clear the binary display, press CLR.

Key Sequence



Example


CLR
00000
SHIFT
CH
*
00000
CHANNEL 000
SHIFT
MONTR
c000 MONTR
0000000000001111
c001 MONTR
0000010101010100
CLR
00000
CHANNEL 001
CLR
00000
DM
00000
CHANNEL DM 0000
MONTR
D0000
FFFF
SHIFT
MONTR
D0000 MONTR
1111111111111111
CLR
D0000
FFFF
SHIFT
CLR
CLR
00000
CHANNEL DM 0000

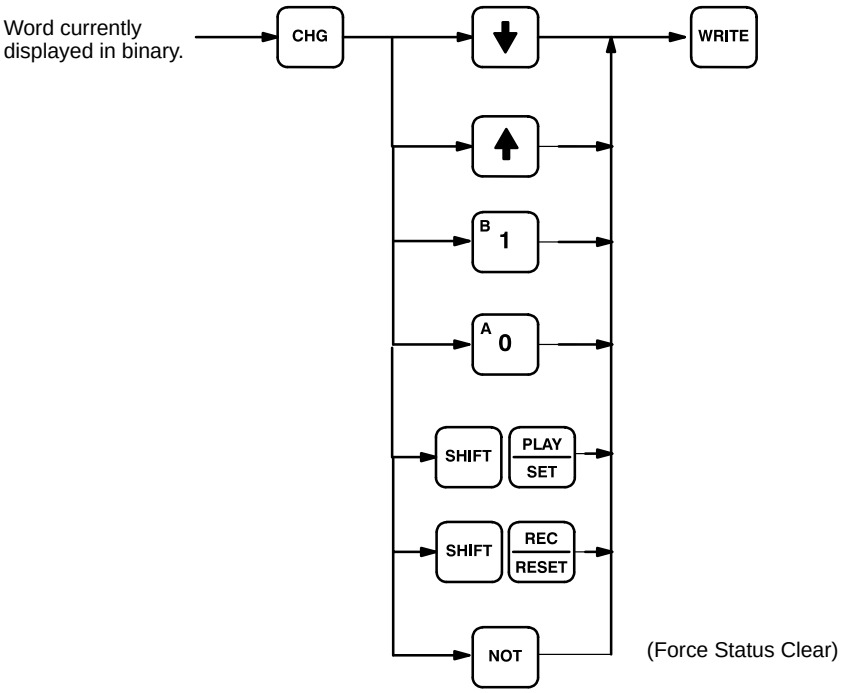

0000S0100R0110SR
Indicates Force Reset in effect
Indicates Force Set in effect

7-1-12 Binary Data Modification

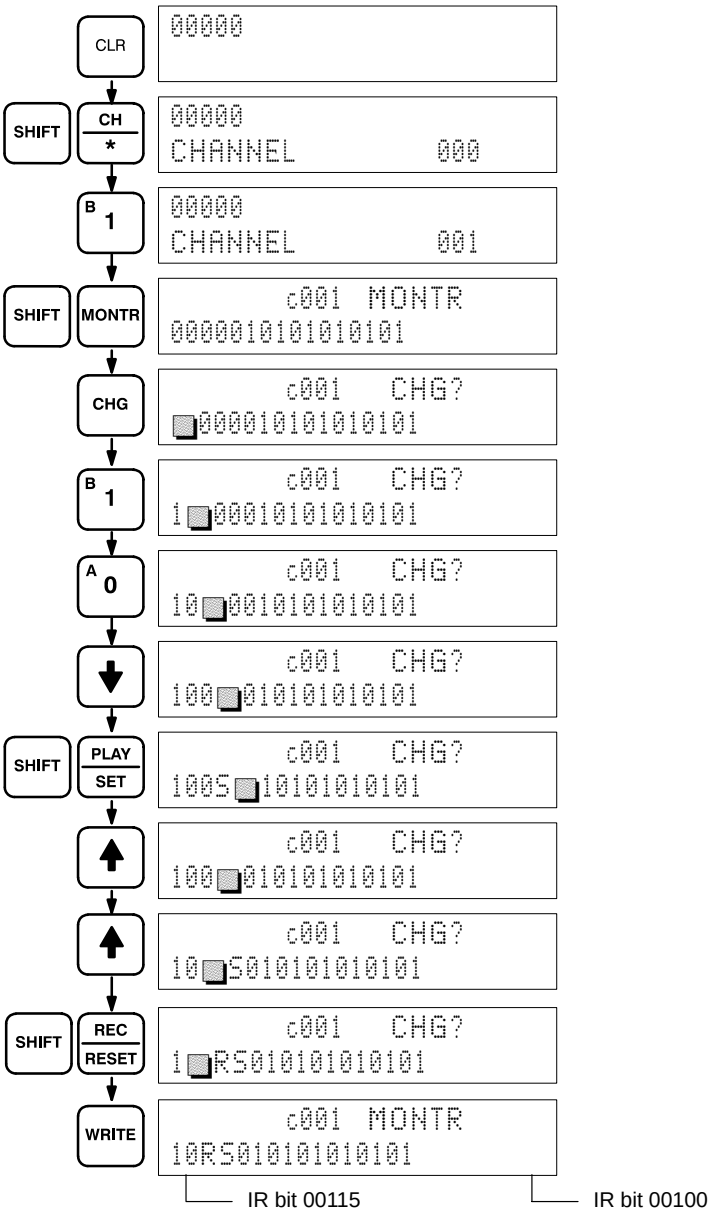
This operation assigns a new 16-digit binary value to an IR, HR, AR, LR, or DM word.

The cursor, which can be shifted to the left with the up key and to the right with the down key, indicates the position of the bit that can be changed. After positioning to the desired bit, a 0 or a 1 can then be entered as the new bit value. The bit can also be Force Set or Force Reset by pressing SHIFT and either PLAY/SET or REC/RESET. An S or R will then appear at that bit position. Pressing the NOT key will clear the force status, S will change to 1, and R to o. After a bit value has been changed, the blinking square will appear at the next position to the right of the changed bit.

Key Sequence



Example



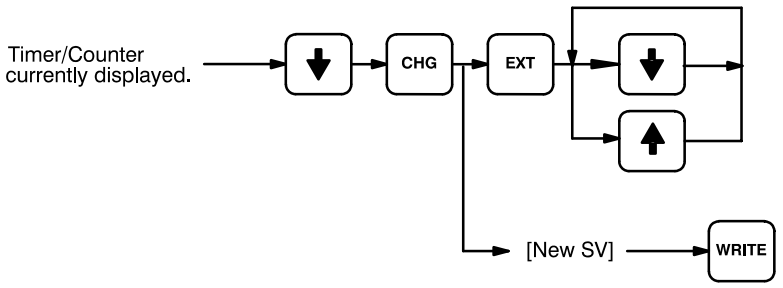
7-1-13 Changing Timer/Counter SV

There are two ways to change the SV of a timer or counter. It can be done either by inputting a new value; or by incrementing or decrementing the current SV. Either method can be used only in MONITOR or PROGRAM mode. In MONITOR mode, the SV can be changed while the program is being executed. Incrementing and decrementing the SV is possible only when the SV has been entered as a constant.

To use either method, first display the address of the timer or counter whose SV is to be changed, presses the down key, and then press CHG. The new value can then be input numerically and WRITE pressed to change the SV or EXT can be pressed followed by the up and down keys to increment and decrement the current SV. When the SV is incremented and/or decremented, CLR can be pressed once to change the SV to the incremented or decremented value but remaining in the display that appeared when EXT was pressed or CLR can be pressed twice to return to the original display with the new SV.

This operation can be used to change a SV from designation as a constant to a word address designation and visa verse.

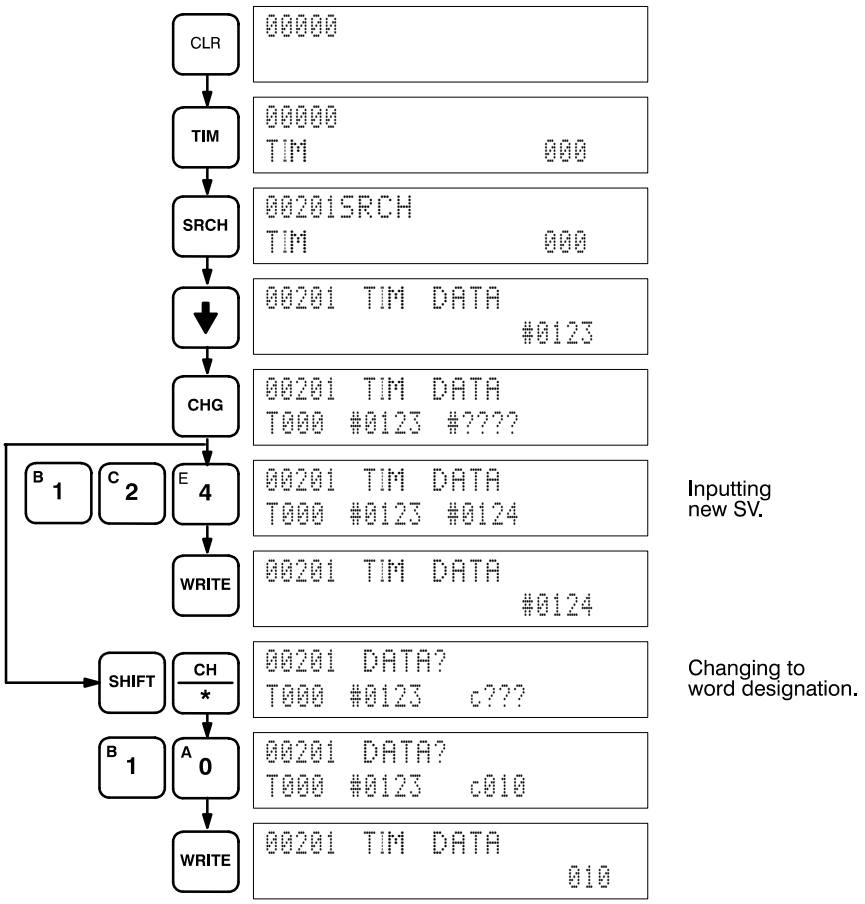
Key Sequence



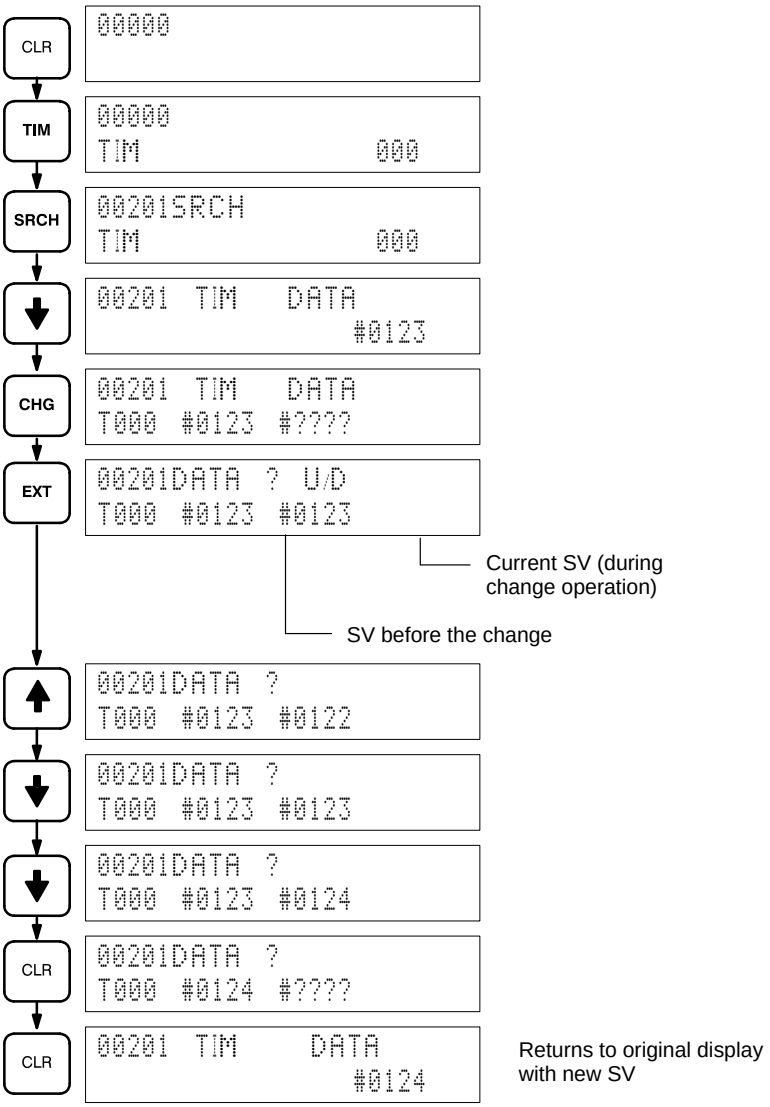
Example

The following examples show inputting a new constant, changing from a constant to an address, and incrementing to a new constant.

Inputting New SV and Changing to Word Designation



Incrementing and Decrementing

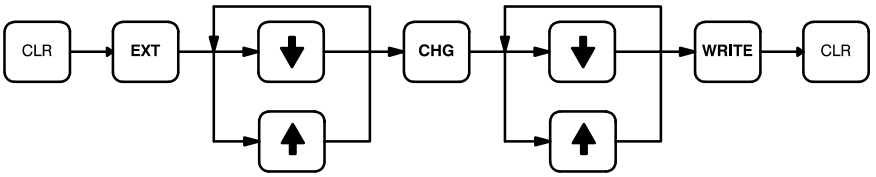


7-1-14 Expansion Instruction Function Code Assignments

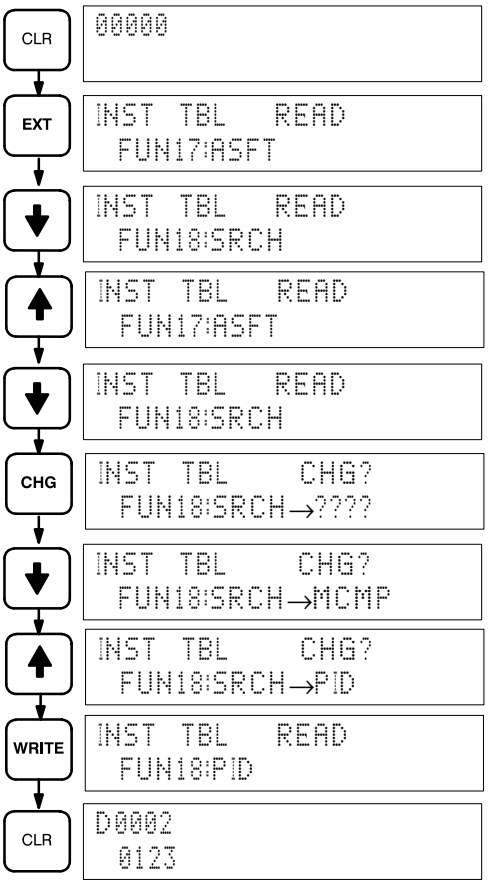
This operation is used to read or change the function codes assigned to expansion instructions. There are 18 function codes that can be assigned to expansion instructions: 17, 18, 19, 47, 48, 60 to 69, and 87 to 89. More than one function code can be assigned to an expansion instruction.

Note Function Code Assignments can be read in any mode, but can be changed in PROGRAM mode only.

Key Sequence



Example



Press CLR to bring up the initial display.

Press EXT to begin displaying function code assignments.

Press the Up and Down Arrow keys to scroll through the function code assignments.

The Up Arrow key displays function codes in ascending order:
17, 18, ... , 89, 17, 18, ...

The Down Arrow key displays function codes in descending order:
17, 89, 88, ... 17, 89, ...

Press CHG to change the displayed function code assignment.

Press the Up and Down Arrow keys to scroll through the instructions.

Press WRITE to enter the change into memory.

Press CLR to return to the initial display.

7-1-15 UM Area Allocation

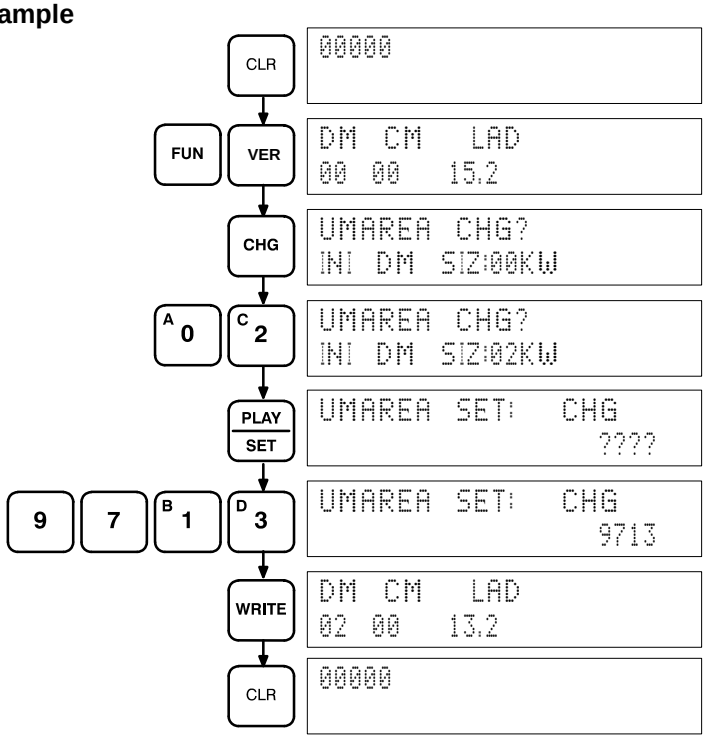
This operation is used to allocate part of the UM Area for use as expansion DM. It can be performed in PROGRAM mode only. Memory allocated to expansion DM is deducted from the ladder program area.

The amount of memory available for the ladder program depends on the amount of RAM in the CPU. About 15.2 KW of memory is available with the16-KW RAM and about 31.2 KW is available with the 32-KW RAM.

This operation cannot be used to allocate UM to the I/O comment area. UM can be allocated to the I/O comment area only with a host computer equipped with LSS (V6 or higher).

Key Sequence

Example



Clear memory completely if the UM Area allocation will be changed.

The current UM Area allocation will be displayed. "??" will be displayed if the allocation information has been lost.

Press CHG to change the UM Area allocation.

Expansion DM can be set to 00, 01, 02, or 03 KW.

Enter the password by pressing PLAY/SET and 9713.

The new UM Area allocation will be displayed. UM allocated to expansion DM is deducted from the ladder program.

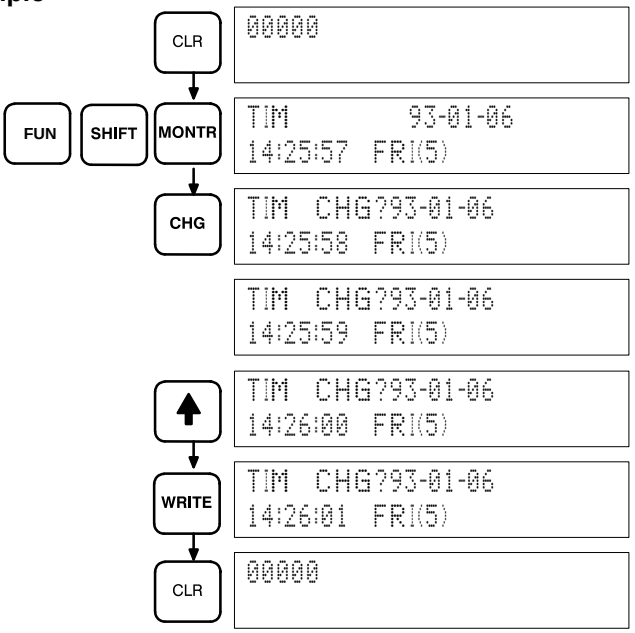
Press CLR to return to the initial display.

7-1-16 Reading and Setting the Clock

This operation is used to read or set the CPU's clock. The clock can be read in any mode, but it can be set in MONITOR or PROGRAM mode only.

The CPU will reject entries outside of the acceptable range, i.e., 01 to 12 for the month, 01 to 31 for the day of the month, 00 to 06 for the day of the week, or 00 to 60 for the seconds, but it will not recognize non-existent dates, such as 2/31.

Example



Press CLR to bring up the initial display.

The display will monitor the current date and time.

Press CHG to change the date and/or time. The "9" in "93" will blink, indicating that it can be changed.

Press the Up and Down Arrow keys to move the cursor through the data and time settings. Input new values to change settings if necessary.

In this case, a "0" was input to replace the "8".

Press CLR to return to the initial display.

7-1-17 Expansion Keyboard Mapping

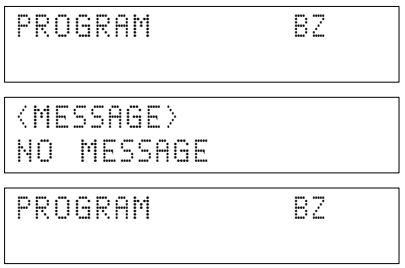
This operation is used to control the ON/OFF status of bits SR 27700 through SR 27909 by pressing keys on the Programming Console's keyboard. The C200HS also supports the C200H's Keyboard Mapping operation, which controls the status of bits in AR 22. These operations can be performed in any PC mode, but the Programming Console must be in TERMINAL or expansion TERMINAL mode.

To enable expansion keyboard mapping, pin 6 of the CPU's DIP switch and AR 0709 must be ON and AR 0708 must be OFF.

Bits turned ON with this operation can be turned OFF by toggling AR 0708. Turn AR 0709 OFF to stop expansion keyboard mapping and switch the Programming Console from Expansion TERMINAL mode to CONSOLE mode.

TERMINAL Mode

The Programming Console can be put into TERMINAL mode by pressing CHG or executing TERM(48) in the program. Pin 6 of the CPU's DIP switch must be OFF.

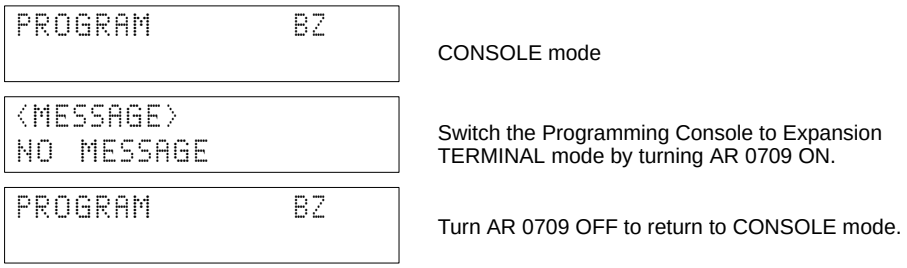


CONSOLE mode

Switch the Programming Console to TERMINAL mode by pressing CHG or executing TERM(48).

Press CHG again to return to CONSOLE mode.

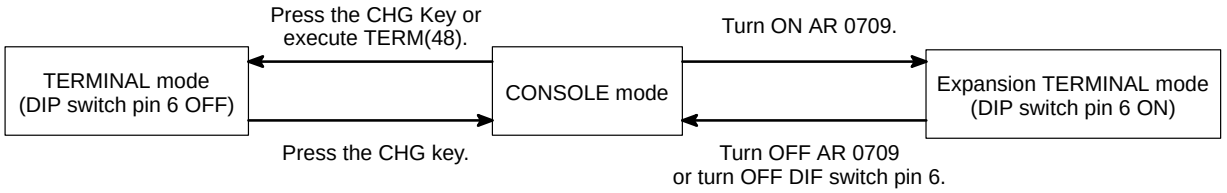
Expansion TERMINAL Mode The Programming Console can be put into Expansion TERMINAL mode by turning ON AR 0709. Pin 6 of the CPU's DIP switch must be ON.



7-1-18 Keyboard Mapping

The C200HS supports the expansion keyboard mapping as well as normal keyboard mapping. Expansion keyboard mapping controls the status of the 41 bits SR 27700 through SR 27909, while normal keyboard mapping controls only the 16 bits in AR 22. The status of these bits can be controlled by pressing the corresponding Programming Console keys when the Programming Console is in TERMINAL mode or expansion TERMINAL mode.

The following diagram shows how to switch the Programming Console between CONSOLE mode (normal operating mode) and TERMINAL or expansion TERMINAL mode.



TERMINAL Mode The Programming Console can be put into TERMINAL mode by pressing CHG or executing TERM(48) in the program. Pin 6 of the CPU's DIP switch must be OFF.

Press the CHG key again to return to CONSOLE mode.

When the Programming Console is in TERMINAL mode it can perform normal keyboard mapping and display messages output by MSG(46) or LMSG(47). With keyboard mapping, bits 00 to 15 of AR 22 will be turned ON when keys 0 to F are pressed on the Programming Console's keyboard. A bit will remain ON after the Programming Console's key is released.

All bits in AR 22 will be turned OFF when AR 0708 is turned ON. Keyboard mapping inputs are disabled when AR 0708 is ON.

In addition to the keyboard mapping function, TERMINAL mode allows messages output by MSG(46) and LMSG(47) to be displayed on the Programming Console. These messages will be erased if the Programming Console is switch back to CONSOLE mode.

Expansion TERMINAL Mode The Programming Console can be put into Expansion TERMINAL mode by turning ON AR 0709. Pin 6 of the CPU's DIP switch must be ON.

Turn off AR 0709 or pin 6 of the CPU's DIP switch to return to CONSOLE mode.

When the Programming Console is in TERMINAL mode it can perform expansion keyboard mapping and display messages output by MSG(46) or LMSG(47). With expansion keyboard mapping, bits SR 27700 through SR 27909 will be turned ON when the corresponding key is pressed on the Programming Console's keyboard. A bit will remain ON after the Programming Console's key is released.

All bits from SR 27700 through SR 27909 will be turned OFF when AR 0708 is turned ON. Expansion keyboard mapping inputs are disabled when AR 0708 is ON.

In addition to the keyboard mapping function, expansion TERMINAL mode allows messages output by MSG(46) and LMSG(47) to be displayed on the Programming Console. These messages will be erased if the Programming Console is switch back to CONSOLE mode.

The following diagram shows the correspondence between the position of Programming Console keys and bits in the SR Area. Each key corresponds 1 to 1 with a bit. Shifted inputs are not recognized. Keys 0 to 15 correspond to bits SR 27700 to SR 27715, keys 16 to 31 correspond to bits SR 27800 to SR 27815, and keys 32 to 41 correspond to bits SR 27900 to SR 27909.

FUN key →

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17
18	19	20	21	22	23
24	25	26	27	28	29
30	31	32	33	34	35
36	37	38	39	40	41

The following table shows the correspondence between the actual Programming Console keys and bits SR 27700 to SR 27909.

SR word	Bit	Corresponding key(s)
277	00	FUN
	01	SET
	02	NOT
	03	*1
	04	*2
	05	SHIFT
	06	AND
	07	OR
	08	CNT
	09	TR
	10	LR
	11	HR

SR word	Bit	Corresponding key(s)
277	12	LD
	13	OUT
	14	TIM
	15	DM
278	00	CH / *
	01	CONT / #
	02	7
	03	8
	04	9
	05	EXT
	06	CHG
	07	SRCH
	08	E 4
	09	F 5
	10	6
	11	PLAY / SET
	12	DEL
	13	MONTR
	14	B 1
	15	C 2
279	00	D 3
	01	REC / RESET
	02	INS

SR word	Bit	Corresponding key(s)
279	03	↑
	04	A 0
	05	*3
	06	CLR
	07	VER
	08	WRITE
	09	↓

SECTION 8
Communications

This section provides an overview of the communications features provided by the C200HS.

8-1 Introduction

The C200HS supports the following types of communications.

- Communications with Programming Devices (e.g., Programming Console, LSS, or SSS.)
- Host Link communications with personal computers and other external devices.
- RS-232C (no-protocol) communications with personal computers and other external devices.
- One-to-one link communications with another C200HS CPU or a CQM1 CPU.
- NT link communications with Programmable Terminals (PTs) equipped with an NT link interface.

This section describes the connection methods and application of all of these different types of communications except for communications with Programming Devices, which is described elsewhere in this manual.

Note
1. One-to-one link communications are possible only with CPUs that have an RS-232C port. They are not possible with the C200HS-CPU01-E/03-E
2. Refer to documentation on the NT-series Interface Units for details on NT link communications.
3. One-to-one link communications and NT link communications are not possible through the peripheral port.

8-2 Parameters for Host Link and RS-232C Communications

The parameters in the PC Setup described in this section are used both by Host Link communications and RS-232C (no-protocol) communications. These parameters must be set in advance to enable communications.

Note
1. PC Setup parameters in DM 6645 to DM 6654 can be set under the PC Setup item on the Utility Menu.
2. The parameters set in the PC Setup will be ignored and the following communications settings will be used if pin 5 on the C200HS CPU's DIP switch is turned ON.

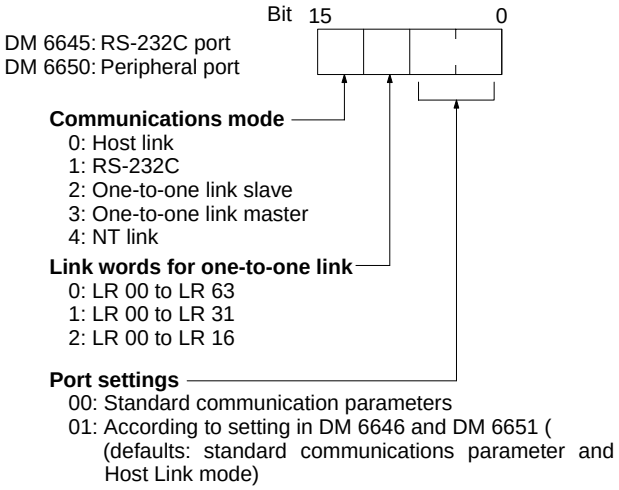
Item	Setting
Communications mode	Host Link
Unit no.	00
Start bits	1
Data length	7
Stop bits	2
Parity	Even
Baud rate	9,600 bps
Transmission delay	None

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.

8-2-1 Standard Communications Parameters

The settings in DM 6645 and DM 6650 determine the main communications parameters, as shown in the following diagram.

The settings in bits 00 through 07 and bits 12 through 15 are valid only when pin 5 on the CPU's DIP switch is OFF. Bits 08 though 11 are valid only in a PC set as the master for a 1:1 link.



The standard communications parameters are as shown in the following table.

Item	Setting
Start bits	1
Data length	7
Stop bits	2
Parity	Even
Baud rate	9,600 bps

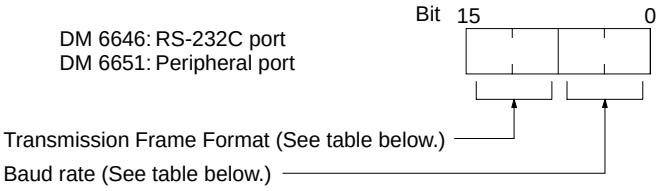
Be sure to set the proper communications mode.

If the above parameters are acceptable, set the rightmost two digits (port settings) to 00. To change the parameters, use the setting described next.

8-2-2 Specific Communications Parameters

The following settings are valid only when pin 5 on the CPU's DIP switch is turned OFF and DM 6645 and DM 6655 are set to specify using the settings in words DM 6646 and DM 6656.

Be sure to set the communications parameters to the same settings for both ends of the communications.



Transmission Frame Format

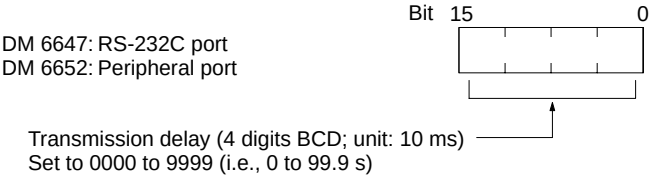
Setting	Stop bits	Data length	Stop bits	Parity
00	1	7	1	Even
01	1	7	1	Odd
02	1	7	1	None
03	1	7	2	Even
04	1	7	2	Odd
05	1	7	2	None
06	1	8	1	Even
07	1	8	1	Odd
08	1	8	1	None
09	1	8	2	Even
10	1	8	2	Odd
11	1	8	2	None

Baud Rate

Setting	Baud rate
00	1,200 bps
01	2,400 bps
02	4,800 bps
03	9,600 bps
04	19,200 bps

Transmission Delay Time

Depending on the devices connected to the RS-232C port, it may be necessary to allow time for transmission. When that is the case, set the transmission delay to regulate the amount of time allowed.

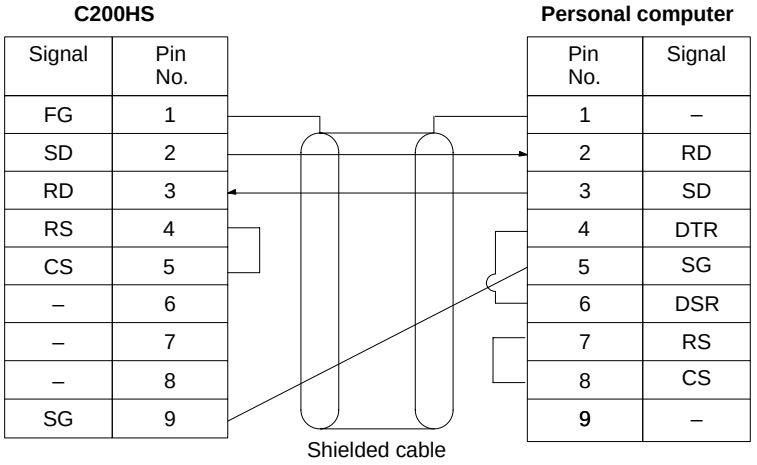


Default: No delay

8-2-3 Wiring Ports

Use the wiring diagram shown below as a guide in wiring the port to the external device. Refer to documentation provided with the computer or other external device for wire details for it.

The connections between the C200HS and a personal computer are illustrated below as an example.



Applicable Connectors

The following connectors are applicable. One plug and one hood are included with the CPU.

Plug: XM2A-0901 (OMRON) or equivalent
Hood: XM2S-0901 (OMRON) or equivalent

Note Ground the FG terminal on the C200HS and at the computer to 100 Ω or less. Refer to the *C200HS Installation Manual* and to the documentation for your computer for details.

8-2-4 Host Link Communications

This section describes the PC Setup parameters and communications procedure for the Host Link communications mode.

Host link communications were developed by OMRON to connect PCs and one or more host computers by RS-232C cable, and to control PCs through communications from the host computer. Normally the host computer issues a command to a PC, and the PC automatically sends back a response. Thus the communications are carried out without the PCs being actively involved. The PCs also have the ability to initiate data transmissions when direct involvement is necessary.

In general, there are two means for implementing host link communications. One is based on C-mode commands, and the other on FINS (CV-mode) commands. The C200HS supports C-mode commands only. For details on host link communications, refer to *Section 11 Host Link Commands.*

The C200HS supports Host Link communications either through the peripheral or RS-232C port, or though Host Link Units (#0 and #1) mounted to the PC. The Host Link Units include the C200H-LD201-V1, C200H-LD202-V1, and C200H-LK101-PV1.

Note The leftmost two digits of DM 6645 and/or DM 6650 must be set to 00 to enable the Host Link Mode. Also, set the rightmost two digits of DM 6645 and DM 6655 and all digits of DM 6646 and DM 6656 to the required communications parameters before attempting to use Host Link communications.

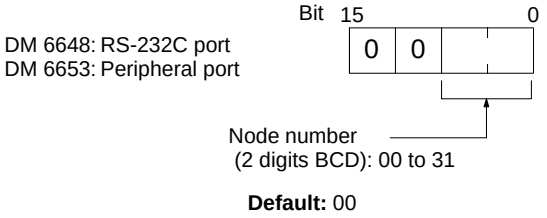
PC Setup

The following parameter in the PC Setup is used only when the Host Link communications mode is being used.

Host Link Node Number

A node number must be set for host link communications to differentiate between nodes when multiple nodes are participating in communications.

Set the node number to 00 unless multiple nodes are connected in a network.

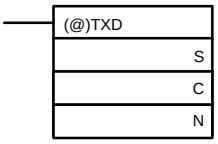


Communications Procedure

This section explains how to use the host link to execute data transmissions from the C200HS. Using this method enables automatic data transmission from the C200HS when data is changed, and thus simplifies the communications process by eliminating the need for constant monitoring by the computer.

1, 2, 3...

1. Check to see that SR 26405 (RS-232C Port Transmit Ready Flag) or SR 26413 (Peripheral Port Transmit Ready Flag) is ON. If using Host Link Units, check SR 26705 for Unit #0 and SR 26713 for Unit #1.
2. Use the TXD(—) instruction to transmit the data.



S: Address of first word of transmission data

C: Control data

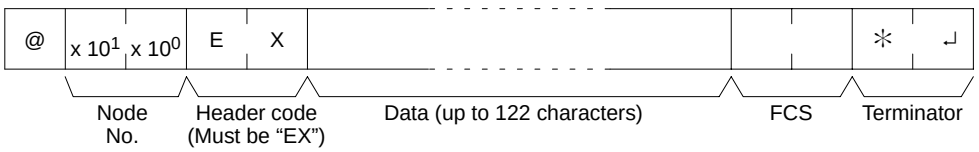
0000:	RS-232C port
1000:	Peripheral port
2000:	Host Link Unit #0
3000:	Host Link Unit #1

N: Number of bytes of data to be sent (4 digits BCD)
0000 to 0061

3. From the time this instruction is executed until the data transmission is complete, SR 26405, SR 26413, SR 26705, or SR 26713 will remain OFF. The bits for the RS-232C or peripheral port will turn ON again upon completion of the data transmission. The bits for the Host Link Units will turn ON again when the data transmission has been passed to the Host Link Unit.
4. The TXD(—) instruction does not provide for a response, so in order to receive confirmation that the computer has received the data, the computer's program must be written so that it gives notification when data is written from the C200HS.

Note

1. The transmission data frame is as shown below for data transmitted in the Host Link mode by means of the TXD(—) instruction.



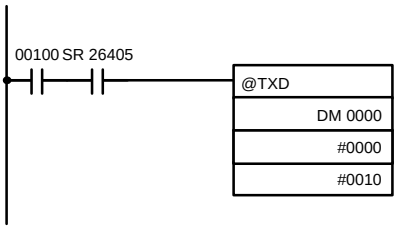
2. To reset the RS-232C port (i.e., to restore the initial status), turn ON SR 25209. To reset the peripheral port, turn ON SR 25208. These bits will turn OFF automatically after the reset. Host Link Unit #0 is reset using SR 25213 and Host Link Unit #1 is reset using SR 25207.
3. If the TXD(—) instruction is executed while the C200HS is in the middle of responding to a command from the computer, the response transmission will first be completed before the transmission is executed according to the

TXD(—) instruction. In all other cases, data transmission based on a TXD(—) instruction will be given first priority.

Application Example

This example shows a program for using the RS-232C port in the Host Link mode to transmit 10 bytes of data (DM 0000 to DM 0004) to a computer. From DM 0000 to DM 0004, “1234” is stored in every word.

The default values are assumed for all of the PC Setup (i.e., the RS-232C port is used in Host Link mode, the node number is 00, and the standard communications parameters are used.)



If SR 26405 (the Transmit Ready Flag) is ON when IR 00100 turns ON, the ten bytes of data (DM 0000 to DM 0004) will be transmitted.

The following type of program must be prepared in the host computer to receive the data. This program allows the computer to read and display the data received from the PC while a host link read command is being executed to read data from the PC.

```
10 'C200HS SAMPLE PROGRAM FOR EXCEPTION
20 CLOSE 1
30 CLS
40 OPEN "COM:E73" AS #1
50 *KEYIN
60 INPUT "DATA  --------",S$
70 IF S$=" " THEN GOTO 190
80 PRINT "SEND DATA = ";S$
90 ST$=S$
100 INPUT "SEND OK? Y or N?=",B$
110 IF B$="Y" THEN GOTO 130 ELSE GOTO *KEYIN
120 S$=ST$
130 PRINT #1,S$                          'Sends command to PC
140 INPUT #1,R$                          'Receives response from PC
150 PRINT "RECV DATA = ";R$
160 IF MID$(R$,4,2)="EX" THEN GOTO 210 'Identifies command from PC
170 IF RIGHT$(R$,1)<>"*" THEN S$=" ":GOTO 130
180 GOTO *KEYIN
190 CLOSE 1
200 END
210 PRINT "EXCEPTION!! DATA"
220 GOTO 140
```

The data received by the computer will be as shown below. The FCS is “59.”

“@00EX12341234123412341234123459*CR”

8-2-5 RS-232C Communications

This section explains RS-232C communications. By using RS-232C communications, the data can be printed out by a printer or read by a bar code reader. Handshaking is not supported for RS-232C communications.

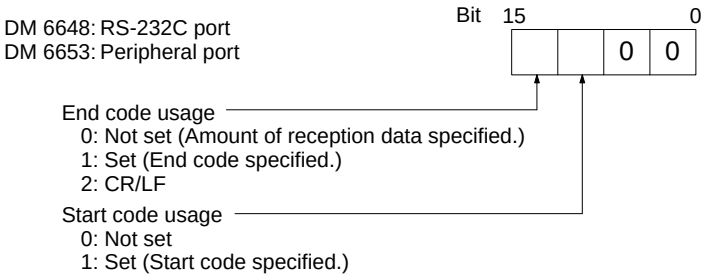
Note The leftmost two digits of DM 6645 and/or DM 6650 must be set to 10 to enable the RS-232C communications. Also, set the rightmost two digits of DM 6645 and DM 6655 and all digits of DM 6646 and DM 6656 to the required communications parameters before attempting to use Host Link communications.

PC Setup

Start and end codes or the amount of data to be received can be set as shown in the following diagrams if required for RS-232C communications. This setting is required only for RS-232C communications.

The following settings are valid only with pin 5 on the CPU's DIP switch is turned OFF.

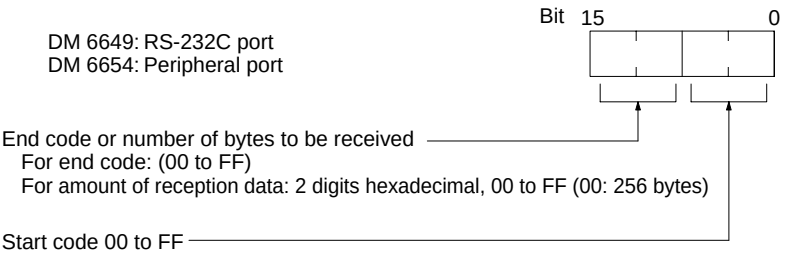
Enabling Start and End Codes



Defaults: No start code; data reception complete at 256 bytes.

Specify whether or not a start code is to be set at the beginning of the data, and whether or not an end code is to be set at the end. Instead of setting the end code, it is possible to specify the number of bytes to be received before the reception operation is completed. Both the codes and the number of bytes of data to be received are set in DM 6649 or DM 6654.

Setting the Start Code, End Code, and Amount of Reception Data



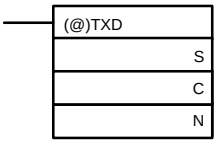
Defaults: No start code; data reception complete at 256 bytes.

Communications Procedure

Transmissions

1, 2, 3...

1. Check to see that SR 26413 (Peripheral Port Transmit Ready Flag) or SR 26405 (RS-232C Port Transmit Ready Flag) is ON.
2. Use the TXD(—) instruction to transmit the data.



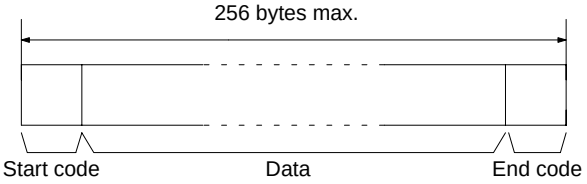
S: Address of first word of data to be transmitted

C: Control data
- Bits 00 to 03
  - 0: Leftmost bytes first
  - 1: Rightmost bytes first
- Bits 12 to 15
  - 0: RS-232C port
  - 1: Peripheral port

N: Number of bytes to be transmitted (4 digits BCD), 0000 to 0256

3. From the time this instruction is executed until the data transmission is complete, SR 26405 (or SR 25413 for the peripheral port) will remain OFF. It will turn ON again upon completion of the data transmission.

Start and end codes are not included when the number of bytes to be transmitted is specified. The largest transmission that can be sent with or without start and end codes in 256 bytes, i.e., N will be between 254 and 256 depending on the designations for start and end codes. If the number of bytes to be sent is set to 0000, only the start and end codes will be sent.

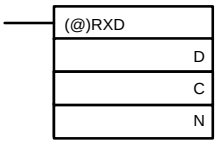


To reset the RS-232C port (i.e., to restore the initial status), turn on SR 25209. To reset the peripheral port, turn on SR 25208. These bits will turn OFF automatically after the reset.

Receptions

1, 2, 3... 1. Check to see that SR 26414 (Peripheral Port Reception Ready Flag) or SR 26406 (RS-232C Port Reception Ready Flag) is ON.

2. Use the RXD(—) instruction to receive the data.



D: Leading word no. for storing reception data

C: Control data
- Bits 00 to 03
  - 0: Leftmost bytes first
  - 1: Rightmost bytes first
- Bits 12 to 15
  - 0: RS-232C port
  - 1: Peripheral port

N: Number of bytes stored (4 digits BCD), 0000 to 0256 (including start and end bits)

3. The status resulting from reading the data received will be stored in the SR Area. Check to see that the operation was successfully completed. The contents of these bits will be reset each time RXD(—) is executed.

RS-232C	Peripheral	Error
SR 26400 to SR 26403	SR 26408 to SR 2641	Communications port error code (1 digit BCD) 0: Normal completion 1: Parity error 2: Framing error 3: Overrun error
SR 26404	SR 26412	Communications error
SR 26407	SR 26415	Reception Overrun Flag (After reception was completed, the subsequent data was received before the data was read by means of the RXD instruction.)
SR 265	SR 266	Number of bytes received (not including start and end bits)

Note The contents of SR 265 and SR 266 may not be zero when power is turned on or when the communications cable is attached. Execute a dummy RXD(—) or reset the communications port to reset these words to zero.

To reset the RS-232C port (i.e., to restore the initial status), turn ON SR 25209. To reset the peripheral port, turn ON SR 25208. These bits will turn OFF automatically after the reset.

Note Always program the Transmit Ready Flag in a NO condition on the instruction line to TXD(—) to ensure that this Flag is ON before the transmission can be executed.

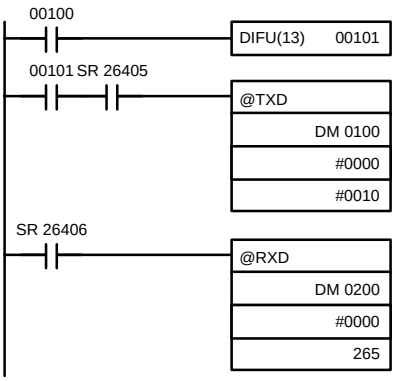
Application Example

This example shows a program for using the RS-232C port in the RS-232C mode to transmit 10 bytes of data (DM 0100 to DM 0104) to the computer, and to store the data received from the computer in the DM area beginning with DM 0200. Before executing the program, the following PC Setup setting must be made.

DM 6645: 1000 (RS-232C port in RS-232C mode; standard communications conditions)

DM 6648: 2000 (No start code; end code CR/LF)

The default values are assumed for all other PC Setup settings. From DM 0100 to DM 0104, 3132 is stored in every word. From the computer, execute a program to receive C200HS data with the standard communications conditions.



If SR 26405 (the Transmit Ready Flag) is ON when IR 00100 turns ON, the ten bytes of data (DM 0100 to DM 0104) will be transmitted, leftmost bytes first.

When SR 26406 (Reception Completed Flag) goes ON, the number of bytes of data specified in SR 265 will be read from the C200HS's reception buffer and stored in memory starting at DM 0200, leftmost bytes first.

The data will be as follows:

"313231323132313231323132CR LF"

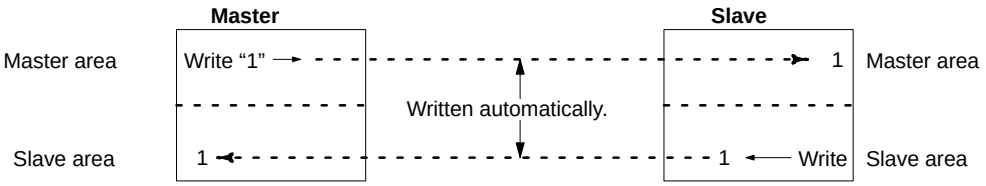
8-2-6 One-to-one Link Communications

If two C200HSs or one C200HS and one CQM1 are linked one-to-one by connecting them together through their RS-232C ports, they can share common LR areas. When two PCs are linked one-to-one, one of them will serve as the master and the other as the slave.

Note The peripheral port cannot be used for 1:1 links.

One-to-one Links

A one-to-one link allows two C200HSs (or a CQM1) to share common data in their LR areas. As shown in the diagram below, when data is written into a word the LR area of one of the linked Units, it will automatically be written identically into the same word of the other Unit. Each PC has specified words to which it can write and specified words that are written to by the other PC. Each can read, but cannot write, the words written by the other PC.



The word used by each PC will be as shown in the following table, according to the settings for the master, slave, and link words.

DM 6645 setting	LR 00 to LR 63	LR 00 to LR 31	LR 00 to LR 15
Master words	LR00 to LR31	LR00 to LR15	LR00 to LR07
Slave words	LR32 to LR63	LR16 to LR31	LR08 to LR15

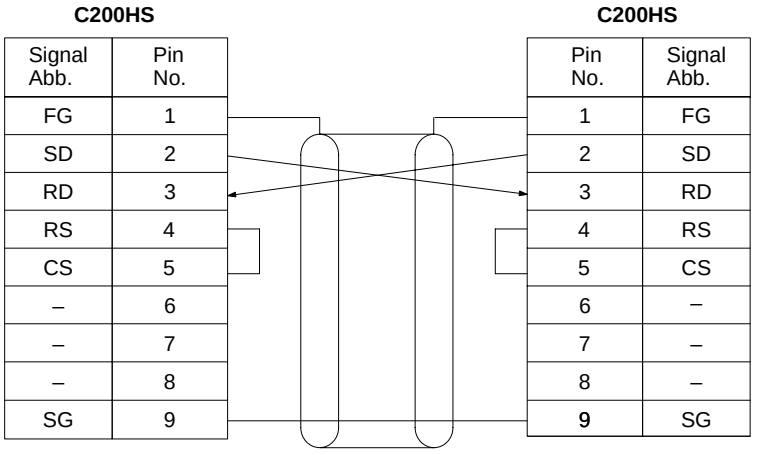
Wiring

Wire the cable as shown in the diagram below using the connector listed.

Applicable Connectors

The following connectors are applicable. One plug and one hood are included with the CPU.

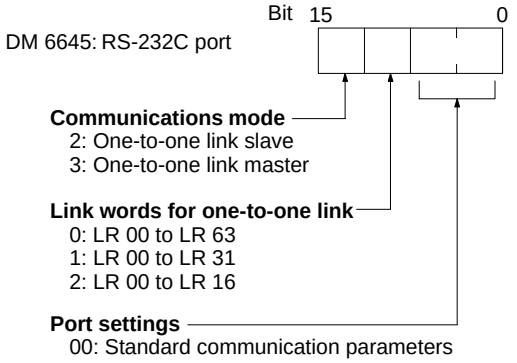
Plug: XM2A-0901 (OMRON) or equivalent
Hood: XM2S-0901 (OMRON) or equivalent



Note Ground the FG terminals the C200HS to a resistance of 100 Ω or less.

PC Setup

To use a 1:1 link, the only settings necessary are the communications mode and the link words. Set the communications mode for one of the PCs to the 1:1 master and the other to the 1:1 slave, and then set the link words in the PC designated as the master. Bits 08 to 11 are valid only for the master for link one-to-one.

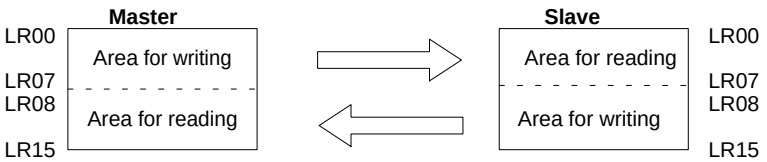


Communications Procedure

If the settings for the master and the slave are made correctly, then the one-to-one link will be automatically started up simply by turning on the power supply to both the C200HSs and operation will be independent of the C200HS operating modes.

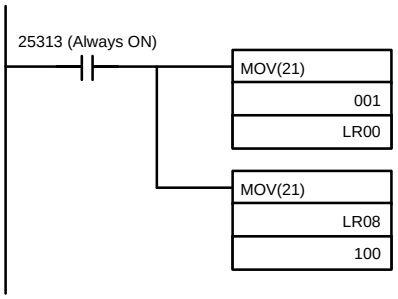
Application Example

This example shows a program for verifying the conditions for executing a one-to-one link using the RS-232C ports. Before executing the program, set the following PC Setup parameters.
Master: DM 6645: 3200 (one-to-one link master; Area used: LR 00 to LR 15)
Slave: DM 6645: 2000 (one-to-one link slave)
The defaults are assumed for all other PC Setup parameters. The words used for the one-to-one link are as shown below.

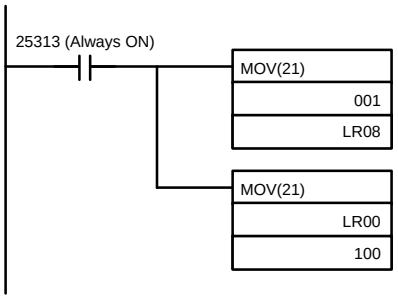


When the program is executed at both the master and the slave, the status of IR 001 of each Unit will be reflected in IR 100 of the other Unit. IR 001 is an input word and IR 100 is an output word.

In the Master



In the Slave



8-2-7 NT Links

NT links can be established by connecting the RS-232C port of the C200HS to the NT Link Interface Unit of a Programmable Terminal.

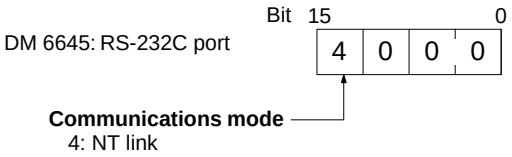
Note The peripheral port cannot be used for NT links.

NT Links

NT link allow high-speed communications with a Programmable Terminal through direct access of PC memory.

PC Setup

Only the following setting is necessary.



Application

Refer to documentation provided for the NT Link Interface Unit for details on actual application of an NT link.

SECTION 9
Memory Cassette Operations

This section describes how to manage both UM Area and IOM data via Memory Cassettes. mounted in the CPU.

9-1 Memory Cassettes

The C200HS comes equipped with a built-in RAM for the user's program so programs can be created even without installing a Memory Cassette. An optional Memory Cassette, however, can provide flexibility in handling program data, PC Setup data, DM data, I/O comment data, and other IOM Area data. Memory Cassettes can be used for the following purposes.

- Saving, retrieving, and comparing data in the UM Area. This UM Area data can include the user program, fixed DM data such as the PC Setup, expansion DM data, I/O comment data, I/O table data, and UM Area allocation data.
- Automatically loading Memory Cassette data on PC startup.
- Saving and retrieving IOM Area data. IOM data includes IR 000 to IR 231, the two work areas, the two SR Areas, the LR Area, the HR Area, the AR Area, the TC Area, and DM 0000 to DM 6143.

There are two types of Memory Cassette available, each with a capacity of 16K words. For instructions on installing Memory Cassettes, refer to the *C200HS Installation Guide*.

The following table shows the Memory Cassettes used with C200HS PCs.

Memory	Capacity	Model number	Features
EEPROM	16K words	C200HS-ME16K	Can be written to while mounted in the C200HS. Can be used for both UM Area data and IOM Area data. A memory backup battery is not required.
EPROM	16K words	C200HS-MP16K	Can be written with a commercially available PROM writer. Can be used for only UM Area data.

Note
1. Memory Cassettes for the C200HS cannot be used with the C200H, and Memory Units for the C200H cannot be used with the C200HS.
2. EEPROM can be written up to 50,000 times. Data may be corrupted if this limit is exceeded.
3. EPROM chips are sold separately.

⚠ **Caution** The C200HS will not operate properly unless the data for it is created on LSS version 3 or later. The C200HS is not supported by earlier versions of the LSS.

⚠ **Caution** Do not mount or remove a Memory Cassette without turning off the power supply to the C200HS CPU.

9-2 Memory Cassette Settings and Flags

The following settings and flags are available for Memory Cassettes in the SR Area. Bits/words used in the procedures to manipulate Memory Cassettes are described in the remainder of this section. Refer to page 46 for other details on SR Area operation.

Word	Bit(s)	Function
SR 269	00 to 07	Memory Cassette Contents 00: Nothing; 01: UM; 02: IOM (03: HIS)
	08 to 10	Memory Cassette Capacity 0: 0 KW (no board); 3: 16 KW
	11 to 13	Reserved by system (not accessible by user)
	14	EEPROM Memory Cassette Protected or EPROM Memory Cassette Mounted Flag
	15	Memory Cassette Flag

Word	Bit(s)	Function	
SR 270	00	Save UM to Cassette Bit	Data transferred to Memory Cassette when Bit is turned ON in PROGRAM mode. Bit will automatically turn OFF. An error will be produced if turned ON in any other mode.
	01	Load UM from Cassette Bit	
	02	Collation Execution Flag	
	03	Collation NG Flag	
	04 to 11	Reserved by system (not accessible by user)	
	12	Transfer Error Flag: Not PROGRAM mode	Data will not be transferred from UM to the Memory Cassette if an error occurs (except for Board Checksum Error). Detailed information on checksum errors occurring in the Memory Cassette will not be output to SR 272 because the information is not needed. Repeat the transmission if SR 27015 is ON.
	13	Transfer Error Flag: Read Only	
	14	Transfer Error Flag: Insufficient Capacity or No UM	
	15	Transfer Error Flag: Board Checksum Error	
SR 271	00 to 07	Ladder program size stored in Memory Cassette Ladder-only File: 04: 4 KW; 08: 8 KW; 12: 12 KW; ... (64: 64 KW) 00: No Ladder program or no file Data updated at data transfer from CPU at startup. The file must begin in segment 0.	
	08 to 15	Ladder program size and type in CPU (Specifications are the same as for bits 00 to 07.)	
SR 272	00 to 10	Reserved by system (not accessible by user)	
	11	Memory Error Flag: PC Setup Checksum Error	
	12	Memory Error Flag: Ladder Checksum Error	
	13	Memory Error Flag: Instruction Change Vector Area Checksum Error	
	14	Memory Error Flag: Memory Cassette Online Disconnection	
	15	Memory Error Flag: Autoboot Error	
SR273	00	Save IOM to Cassette Bit	Data transferred to Memory Cassette when Bit is turned ON in PROGRAM mode. Bit will automatically turn OFF. An error will be produced if turned ON in any other mode.
	01	Load IOM from Cassette Bit	
	02 to 11	Reserved by system (not accessible by user)	
	12	Transfer Error Flag: Not PROGRAM mode	Data will not be transferred from IOM to the Memory Cassette if an error occurs (except for Read Only Error).
	13	Transfer Error Flag: Read Only	
	14	Transfer Error Flag: Insufficient Capacity or No IOM	

9-3 UM Area Data

The following procedures can be used to read UM Area data from, write UM Area data to, and compare UM Area data on an EEPROM Memory Cassette mounted to a C200HS CPU. This UM Area data can include the user program, fixed DM data such as the PC Setup, expansion DM data, I/O comment data, I/O table data, and UM Area allocation data. The procedures cannot be used to write to EPROM Memory Cassettes, which require a PROM writer. Refer to the LSS operation manuals for PROM writer procedures.

Note The data inside the Memory Cassette should be protected by turning on the write-protect switch whenever you are not planning to write to the Cassette.

Writing Data

The following procedure is used to write UM Area data from the C200HS CPU to a Memory Cassette mounted in the CPU.

1, 2, 3...

1. Turn off the write-protect switch on the Memory Cassette to write-enable it.
2. Make sure that power to the C200HS CPU is turned OFF.
3. Mount the Memory Cassette to the CPU.

4. Turn on the CPU.
5. If the desired program or UM Area data is not already in the CPU, write the data or transfer it to the CPU.
6. Switch the C200HS to PROGRAM mode.
7. Turn ON SR 27000 from the LSS or a Programming Console. The UM Area data will be written to the Memory Card and SR 27000 will be turned OFF automatically.

Reading Data

The following procedures are used to read UM Area data from a Memory Cassette mounted in the C200HS CPU to the CPU. This can be achieved either automatically when the C200HS is turned on or manually from a Programming Device.

Automatic Reading at Startup

1, 2, 3...

1. Turn ON pin 2 on the CPU's DIP switch to enable reading at startup.
2. Make sure that power to the C200HS CPU is turned OFF.
3. Mount the Memory Cassette containing the data to be read to the CPU.
4. Turn on the CPU. The data in the Memory Cassette will be read to the CPU.

Manual Reading

1, 2, 3...

1. Make sure that power to the C200HS CPU is turned OFF.
2. Mount the Memory Cassette containing the data to be read to the CPU.
3. Turn on the CPU.
4. Switch the C200HS to PROGRAM mode.
5. Turn ON SR 27001 from the LSS or a Programming Console. The data in the Memory Cassette will be read to the CPU and SR 27001 will be turned OFF automatically.

Comparing Data

The following procedure is used to compare the UM Area data in a Memory Cassette to the UM Area data in the CPU.

1, 2, 3...

1. Make sure that power to the C200HS CPU is turned OFF.
2. Mount the Memory Cassette containing the data to be compared.
3. Turn on the CPU.
4. Switch the C200HS to PROGRAM mode.
5. Turn ON SR 27002 from the LSS or a Programming Console. The data in the Memory Cassette will be compared to the data in CPU and SR 27002 will be turned OFF automatically.
6. Read the status of SR 27003 to determine the results of the comparison. If SR 27003 is OFF, the data in the Memory Cassette is the same as that in the CPU. If SR 27003 is ON, either the data is different or the C200HS was not in PROGRAM mode when the operation was performed.

Note

1. SR 27003 is used only to show the results of comparisons and cannot be manipulated by the user.
2. If the comparison operation is executed in any but PROGRAM mode, SR 27002 will turn ON, as will SR 27003. In this case, the status of SR 27003 will be meaningless.
3. SR 273003 will turn ON if the comparison operation is used without a Memory Cassette mounted to the CPU.

9-4 IOM Area Data

The following procedures can be used to read IOM data from and write IOM data to an EEPROM Memory Cassette mounted to a C200HS CPU. IOM data includes IR 000 to IR 231, the two work areas, the two SR Areas, the LR Area, the HR Area, the AR Area, the TC Area, and DM 0000 to DM 6143. This procedure cannot be used to write to EPROM Memory Cassettes, which require a PROM writer. Refer to the LSS operation manuals for PROM writer procedures.

Note The data inside the Memory Cassette should be protected by turning on the write-protect switch whenever you are not planning to write to the Cassette.

Writing Data

The following procedure is used to write IOM data from the C200HS CPU to a Memory Cassette mounted in the CPU.

1, 2, 3...

1. Turn off the write-protect switch on the Memory Cassette to write-enable it.
2. Make sure that power to the C200HS CPU is turned OFF.
3. Mount the Memory Cassette to the CPU.
4. Turn on the CPU.
5. Switch the C200HS to PROGRAM mode.
6. Turn ON SR 27300 from the LSS or a Programming Console. The IOM data will be written to the Memory Card and SR 27300 will be turned OFF automatically.

Reading Data

The following procedures are used to read IOM data from a Memory Cassette mounted in the C200HS CPU to the CPU. This data cannot be read out automatically at startup and must be read manually. Also, the error log in DM 6000 to DM 6030 cannot be read with this procedure.

1, 2, 3...

1. Make sure that power to the C200HS CPU is turned OFF.
2. Mount a Memory Cassette to the CPU.
3. Turn on the CPU.
4. Switch the C200HS to PROGRAM mode.
5. Turn ON SR 27301 from the LSS or a Programming Console. The IOM data in the Memory Cassette will be read to the CPU and SR 27301 will be turned OFF automatically.

SECTION 10
Troubleshooting

The C200HS provides self-diagnostic functions to identify many types of abnormal system conditions. These functions minimize downtime and enable quick, smooth error correction.

This section provides information on hardware and software errors that occur during PC operation. Program input errors are described in *4-7 Inputting, Modifying, and Checking the Program.* Although described in *Section 3 Memory Areas,* flags and other error information provided in SR and AR areas are listed in *10-5 Error Flags*.

10-1 Alarm Indicators

The ALM/ERR indicator on the front of the CPU provides visual indication of an abnormality in the PC. When the indicator is ON (ERROR), a fatal error (i.e., ones that will stop PC operation) has occurred; when the indicator is flashing (ALARM), a nonfatal error has occurred. This indicator is shown in *2-1-1 CPU Indicators*.

⚠ **Caution** The PC will turn ON the ALM/ERR indicator, stop program execution, and turn OFF all outputs from the PC for most hardware errors, for certain fatal software errors, or when FALS(07) 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.

10-2 Programmed Alarms and Error Messages

FAL(06), FALS(07), and MSG(46) can be used in the program to provide user-programmed information on error conditions. With these three instructions, the user can tailor error diagnosis to aid in troubleshooting.

FAL(06) is used with a FAL number other than 00, which is output to the SR area when FAL(06) is executed. Executing FAL(06) will not stop PC operation or directly affect any outputs from the PC.

FALS(07) is also used with a FAL number, which is output to the same location in the SR area when FALS(07) is executed. Executing FALS(07) will stop PC operation and will cause all outputs from the PC to be turned OFF.

When FAL(06) is executed with a function number of 00, the current FAL number contained in the SR area is cleared and replaced by another, if more have been stored in memory by the system.

When MSG(46) is used a message containing specified data area words is displayed onto the Programming Console or another Programming Device.

The use of these instructions is described in detail in *Section 5 Instruction Set*.

10-3 Reading and Clearing Errors and Messages

System error messages can be displayed onto the Programming Console or other Programming Device.

On the Programming Console, press the CLR, FUN, and MONTR keys. If there are multiple error messages stored by the system, the MONTR key can be pressed again to access the next message. If the system is in PROGRAM mode, pressing the MONTR key will clear the error message, so be sure to write down all message errors as you read them. (It is not possible to clear an error or a message while in RUN or MONITOR mode; the PC must be in PROGRAM mode.) When all messages have been cleared, "ERR CHK OK" will be displayed.

Details on accessing error messages from the Programming Console are provided in *7-1 Monitoring Operation and Modifying Data*. Procedures for the LSS are provided in the *LSS Operation Manual*.

10-4 Error Messages

There are basically three types of errors for which messages are displayed: initialization errors, non-fatal operating errors, and fatal operating errors. Most of these are also indicated by FAL number being transferred to the FAL area of the SR area.

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.

After eliminating the cause of an error, clear the error message from memory before resuming operation.

Asterisks in the error messages in the following tables indicate variable numeric data. An actual number would appear on the display.

Initialization Errors

The following error messages appear before program execution has been started. The POWER indicator will be lit and the RUN indicator will not be lit for either of these. The RUN output will be OFF for each of these errors.

Error and message	FAL no.	Probable cause	Possible correction
Waiting for Special I/O or Interrupt Input Units CPU WAIT G	None	A Special I/O Unit or Interrupt Input Unit has not initialized.	Perform the I/O Table Read operation to check unit numbers. Replace Unit if it is indicated by "$" only in the I/O table. (High-density I/O Units will not appear on I/O Table Read display for all peripheral devices.)
Waiting for Remote I/O CPU WAIT G	None	Power to Remote I/O Unit is off or terminator cannot be found.	Check power supply to Remote I/O Units, connections between Remote I/O Units, and terminator setting.

Non-fatal Operating Errors

The following error messages appear for errors that 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 and RUN indicators will be lit and the ALM/ERR indicator will be flashing. The RUN output will be ON.

Error and message	FAL no.	Probable cause	Possible correction
FAL error SYS FAIL FAL**	01 to 99	FAL(06) has been executed in program. Check the FAL number to determine conditions that would cause execution (set by user).	Correct according to cause indicated by FAL number (set by user).
Interrupt Input Unit error SYS FAIL FAL8A	8A	An error occurred in data transfer between the Interrupt Input Unit and the CPU.	Replace the Interrupt Input Unit.
Interrupt subroutine error SYS FAIL FAL8B	8B	Interrupt subroutine containing Special I/O Unit refresh was too long. Cyclic Special I/O Unit refreshing was not disabled for interrupt subroutine refresh.	Check the interrupt subroutine and PC Setup.

Error and message	FAL no.	Probable cause	Possible correction
High-density I/O Unit error SYS FAIL FAL9A	9A	An error occurred in data transfer between a High-density I/O Unit and the CPU.	Check AR 0205 to AR 0214 to identify the Unit with a problem, replace the Unit, and restart the PC.
PC Setup error SYS FAIL FAL9B	9B	An error has been detected in the PC Setup. This error will be generated when the setting is read or used for the first time.	Check an correct PC Setup settings.
Memory Cassette Transfer error SYS FAIL FAL9D	9D	An error has occurred during data transmission between UM and a Memory Cassette because: Not in PROGRAM Mode. UM or Memory Cassette is read-only. Insufficient capacity in UM or Memory Cassette. A checksum error occurred in the Memory Cassette	Make sure that the PC is in PROGRAM mode. Make sure that the Memory Cassette is not write-protected. Make sure that the UM and Memory Cassette capacity is sufficient. Make sure that SYSMAC NET data links are not active during the transfer. Transfer the data again.
Cycle time overrun CYCLE TIME OVER	F8	Watchdog timer has exceeded 100 ms.	Program cycle time is longer than recommended. Reduce cycle time if possible.
I/O table verification error I/O VER ERR	E7	Unit has been removed or replaced by a different kind of Unit, making I/O table incorrect.	Use I/O Table Verify Operation to check I/O table and either connect dummy Units or register the I/O table again.
Remote I/O error REMOTE ERR * Remote I/O Master Unit number	B0 or B1	Error occurred in transmissions between Remote I/O Units.	Check transmission line between PC and Master and between Remote I/O Units.
Special I/O Unit error SIOU ERR	D0	Error has occurred in PC Link Unit, Remote I/O Master Unit, between a Host Link, SYSMAC LINK, or SYSMAC NET Link Unit and the CPU, or in refresh between Special I/O Unit and the CPU.	Determine the unit number of the Unit which caused the error (AR 00), correct the error, and toggle the appropriate Restart Bit in AR 01, SR 250, or SR 252. If the Unit does not restart, replace it.
Group-2 High-density I/O error SYS FAIL FAL 9A	9A	An error has occurred during data transmission between the CPU and a Group-2 High-density I/O Unit.	Determine the unit number of the Unit which caused the error (AR 02), replace the Unit, and try to power-up again.
Battery error BATT LOW	F7	Backup battery is missing or its voltage has dropped.	Check battery, and replace if necessary.

Fatal Operating Errors

The following error messages appear for errors that 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. No CPU indicators will be lit for the power interruption error. For all other fatal operating errors, the POWER and ALM/ERR indicators will be lit. The RUN output will be OFF.

Error and message	FAL no.	Probable cause	Possible correction
Power interruption No message.	None	Power has been interrupted for at least 10 ms.	Check power supply voltage and power lines. Try to power-up again.
CPU error No message.	None	Watchdog timer has exceeded maximum setting (default setting: 130 ms).	Restart system in PROGRAM mode and check program. Reduce cycle time or reset watchdog timer if longer time required. (Consider effects of longer cycle time before resetting.)
Memory error MEMORY ERR	F1	SR 27211 ON: A checksum error has occurred in the PC Setup (DM 6600 to DM 6655).	Check the PC Setup.
		SR 27212 ON: A checksum error has occurred in the program, indicating an incorrect instruction.	Check the program.
		SR 27213 ON A checksum error has occurred in an expansion instruction change.	
		SR 27214 ON: Memory Cassette was installed or removed with the power on.	Install the Memory Cassette correctly.
		SR 27215 ON: Autoboot error.	Check whether the CPU memory is protected or a checksum error occurred in the Memory Cassette.
No END(01) instruction NO END INST	F0	END(01) is not written anywhere in program.	Write END(01) at the final address of the program.
I/O bus error I/O BUS ERR * Rack no.	C0 to C2	Error has occurred in the bus line between the CPU and I/O Units.	The rightmost digit of the FAL number will indicate the number of the Rack where the error was detected. Check cable connections between the Racks.

Error and message	FAL no.	Probable cause	Possible correction
Too many Units I/O UNIT OVER	E1	Two or more Special I/O Units are set to the same unit number Two or more Group-2 High-density I/O Units are set to the same I/O number or I/O word. The I/O number of a 64-pt Group-2 High-density I/O Unit is set to 9. Two SYSMAC NET Link or SYSMAC LINK Units share the same operating level. Two or more Interrupt Input Units are mounted.	Perform the I/O Table Read operation to check unit numbers, and eliminate duplications. (High-density I/O Units other than Group-2 are Special I/O Units, too.) Set unit numbers of 64-pt Group-2 High-density I/O Units to numbers other than 9. Check the SYSMAC NET Link and SYSMAC LINK Unit operating levels and eliminate duplications. Mount only one Interrupt Input Unit.
Input-output I/O table error I/O SET ERROR	E0	Input and output word designations registered in I/O table do no agree with input/output words required by Units actually mounted.	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.
FALS error SYS FAIL FAL**	01 to 99 or 9F	FALS has been executed by the program. Check the FAL number to determine conditions that would cause execution (Set by user or by system).	Correct according to cause indicated by FAL number. If FAL number is 9F, check watchdog timer and cycle time, which may be to long. 9F will be output when FALS(07) is executed and the cycle time is between 120 and 130 ms.

Communications Errors

If errors occur in communications, the indicator for the peripheral port (COMM/COMM1 when using RS-232C Programming Console cable) or the RS-232C port (COMM2) will not light. Check the connection, programming on both ends (C200HS and peripheral), and then reset the port using the reset bits (peripheral port: SR 25208; RS-232C port: SR 25209).

Other Error Messages

A number of other error messages are detailed within this manual. Errors in program input and debugging can be examined in *Section 4 Writing and Inputting the Program.*

10-5 Error Flags

The following table lists the flags and other information provided in the SR and AR areas that can be used in troubleshooting. Details are provided in *3-4 SR Area* and *3-5 AR Area*.

SR Area

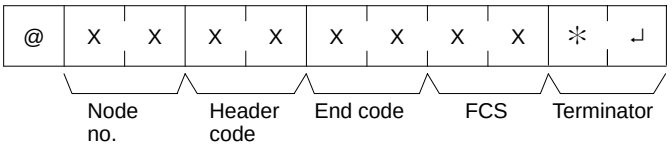
Address(es)	Function
23600 to 23615	Node loop status for SYSMAC NET Link system
23700 to 23715	Completion/error code output area for SEND(90)/RECV(98) in SYSMAC LINK/SYSMAC NET Link System
24700 to 25015	PC Link Unit Run and Error Flags
25100 to 25115	Remote I/O Error Flags
25200	SYSMAC LINK/SYSMAC NET Link Level 0 SEND(90)/RECV(98) Error Flag
25203	SYSMAC LINK/SYSMAC NET Link Level 1 SEND(90)/RECV(98) Error Flag
25206	Rack-mounting Host Link Unit Level 1 Error Flag
25300 to 25307	FAL number output area.
25308	Low Battery Flag
25309	Cycle Time Error Flag
25310	I/O Verification Error Flag
25311	Rack-mounting Host Link Unit Level 0 Error Flag
25312	Remote I/O Error Flag
25411	Interrupt Input Unit Error Flag
25413	Interrupt Programming Error Flag
25414	Group-2 High-density I/O Unit Error Flag
25415	Special Unit Error Flag (Special I/O, PC Link, Host Link, Remote I/O Master, SYSMAC NET Link, or SYSMAC Link Unit Error Flag)
25503	Instruction Execution Error (ER) Flag
26400 to 26403	RS-232C Port Error Code
26404	RS-232C Port Communications Error
26408 to 26411	Peripheral Port Error Code (except Peripheral Mode)
26412	Peripheral Port Communications Error Flag (except Peripheral Mode)
27011	UM Transfer Error Flag: SYSMAC NET data link active during data link table transfer.
27012	UM Transfer Error Flag: Not PROGRAM mode
27013	UM Transfer Error Flag: Read Only
27014	UM Transfer Error Flag: Insufficient capacity or no UM
27015	UM Transfer Error Flag: Board Checksum Error
27211	Memory Error Flag: PC Setup Checksum Error
27212	Memory Error Flag: UM or Ladder Checksum Error
27213	Memory Error Flag: Expansion Instruction Code Change Area Checksum Error
27214	Memory Error Flag: Memory Cassette Online Disconnection
27215	Memory Error Flag: Autoboot Error
27312	IOM Transfer Error Flag: Not PROGRAM mode
27313	IOM Transfer Error Flag: Read Only
27314	IOM Transfer Error Flag: Insufficient capacity
27315	IOM Transfer Error Flag: Checksum Error
27500	PC Setup Error (DM 6600 to DM 6614)
27501	PC Setup Error (DM 6615 to DM 6644)
27502	PC Setup Error (DM 6645 to DM 6655)

AR Area

Address(es)	Function
0000 to 0009	Special I/O or PC Link Unit Error Flags
0010	SYSMAC LINK/SYSMAC NET Link Level 1 System Error Flags
0011	SYSMAC LINK/SYSMAC NET Link Level 0 System Error Flags
0012	Rack-mounting Host Link Unit Level 1 Error Flag
0013	Rack-mounting Host Link Unit Level 0 Error Flag
0014	Remote I/O Master Unit 1 Error Flag
0015	Remote I/O Master Unit 0 Error Flag
0200 to 0204	Error Flags for Slave Racks 0 to 4
0205 to 0215	Group-2 High-density I/O Unit Error Flags (AR 0205 to AR 0214 correspond to I/O numbers 0 to 9.)
0215	Group-2 High-density I/O Unit was not recognized.
0300 to 0315	Optical I/O Units (0 to 7) Error Flags
0400 to 0415	Optical I/O Units (8 to 15) Error Flags
0500 to 0515	Optical I/O Units (16 to 23) Error Flags
0600 to 0615	Optical I/O Units (24 to 31) Error Flags
0710 to 0712	Error Flags for Slave Racks 5 to 7
0713 to 0715	Error History Bits
1114	Communications Controller Error Flag Level 0
1115	EEPROM Error Flag for operating level 0
1514	Communications Controller Error Flag Level 1
1515	EEPROM Error Flag for operating level 1

10-6 Host Link Errors

These error codes are received as the response code (end code) when a command received by the C200HS from a host computer cannot be processed. The error code format is as shown below.



The header code will vary according to the command and can contain a subcode (for composite commands).

End code	Contents	Probable cause	Corrective measures
00	Normal completion	---	---
01	Not executable in RUN mode	The command that was sent cannot be executed when the PC is in RUN mode.	Check the relation between the command and the PC mode.
02	Not executable in MONITOR mode	The command that was sent cannot be executed when the PC is in MONITOR mode.	
0B	Not executable in PROGRAM mode	The command that was sent cannot be executed when the PC is in PROGRAM mode.	This code is not presently being used.
13	FCS error	The FCS is wrong. Either the FCS calculation is mistaken or there is adverse influence from noise.	Check the FCS calculation method. If there was influence from noise, transfer the command again.
14	Format error	The command format is wrong.	Check the format and transfer the command again.
15	Entry number data error	The areas for reading and writing are wrong.	Correct the areas and transfer the command again.
16	Command not supported	The specified command does not exist.	Check the command code.
18	Frame length error	The maximum frame length was exceeded.	Divide the command into multiple frames.
19	Not executable	Items to read not registered for composite command (QQ).	Execute QQ to register items to read before attempting batch read.
23	User memory write-protected	Pin 1 on C200HS DIP switch is ON.	Turn OFF pin 1 to execute.
A3	Aborted due to FCS error in transmit data	The error was generated while a command extending over more than one frame was being executed. **Note:** The data up to that point has already been written to the appropriate area of the CPU.	Check the command data and try the transfer again.
A4	Aborted due to format error in transmit data		
A5	Aborted due to entry number data error in transmit data		
A8	Aborted due to frame length error in transmit data		
Other	---	Influence from noise was received.	Transfer the command again.

Power Interruptions

The following responses may be received from the C200HS if a power interruption occurs, including momentary interruptions. If any of these responses are received during or after a power interruption, repeat the command.

Undefined Command Response

@00IC4A*↵

No Response

If no response is received, abort the last command and resend.

SECTION 11
Host Link Commands

This section explains the methods and procedures for using host link commands, which can be used for host link communications via the C200HS ports.

11-1 Communications Procedure

Command Chart

The commands listed in the chart below can be used for host link communications with the C200HS. These commands are all sent from the host computer to the PC.

Header code	PC mode			Name	Page
	RUN	MON	PRG		
RR	Valid	Valid	Valid	IR/SR AREA READ	407
RL	Valid	Valid	Valid	LR AREA READ	407
RH	Valid	Valid	Valid	HR AREA READ	408
RC	Valid	Valid	Valid	PV READ	408
RG	Valid	Valid	Valid	TC STATUS READ	409
RD	Valid	Valid	Valid	DM AREA READ	409
RJ	Valid	Valid	Valid	AR AREA READ	410
WR	Not valid	Valid	Valid	IR/SR AREA WRITE	410
WL	Not valid	Valid	Valid	LR AREA WRITE	411
WH	Not valid	Valid	Valid	HR AREA WRITE	411
WC	Not valid	Valid	Valid	PV WRITE	412
WG	Not valid	Valid	Valid	TC STATUS WRITE	412
WD	Not valid	Valid	Valid	DM AREA WRITE	413
WJ	Not valid	Valid	Valid	AR AREA WRITE	413
R#	Valid	Valid	Valid	SV READ 1	414
R$	Valid	Valid	Valid	SV READ 2	415
R%	Valid	Valid	Valid	SV READ 3	416
W#	Not valid	Valid	Valid	SV CHANGE 1	417
W$	Not valid	Valid	Valid	SV CHANGE 2	417
W%	Not valid	Valid	Valid	SV CHANGE 3	418
MS	Valid	Valid	Valid	STATUS READ	419
SC	Valid	Valid	Valid	STATUS WRITE	420
MF	Valid	Valid	Valid	ERROR READ	421
KS	Not valid	Valid	Valid	FORCED SET	422
KR	Not valid	Valid	Valid	FORCED RESET	423
FK	Not valid	Valid	Valid	MULTIPLE FORCED SET/RESET	424
KC	Not valid	Valid	Valid	FORCED SET/RESET CANCEL	425
MM	Valid	Valid	Valid	PC MODEL READ	425
TS	Valid	Valid	Valid	TEST	426
RP	Valid	Valid	Valid	PROGRAM READ	426
WP	Not valid	Not valid	Valid	PROGRAM WRITE	427
MI	Not valid	Not valid	Valid	I/O TABLE GENERATE	427
QQ	Valid	Valid	Valid	COMPOUND COMMAND	427
XZ	Valid	Valid	Valid	ABORT (command only)	429
⁎⁎	Valid	Valid	Valid	INITIALIZE (command only)	430
IC	---	---	---	Undefined command (response only)	430

Host link communications are executed by means an exchange of commands and responses between the host computer and the PC. With the C200HS, there are two communications methods that can be used. One is the normal method, in which commands are issued from the host computer to the PC. The other method allows commands to be issued from the PC to the host computer.

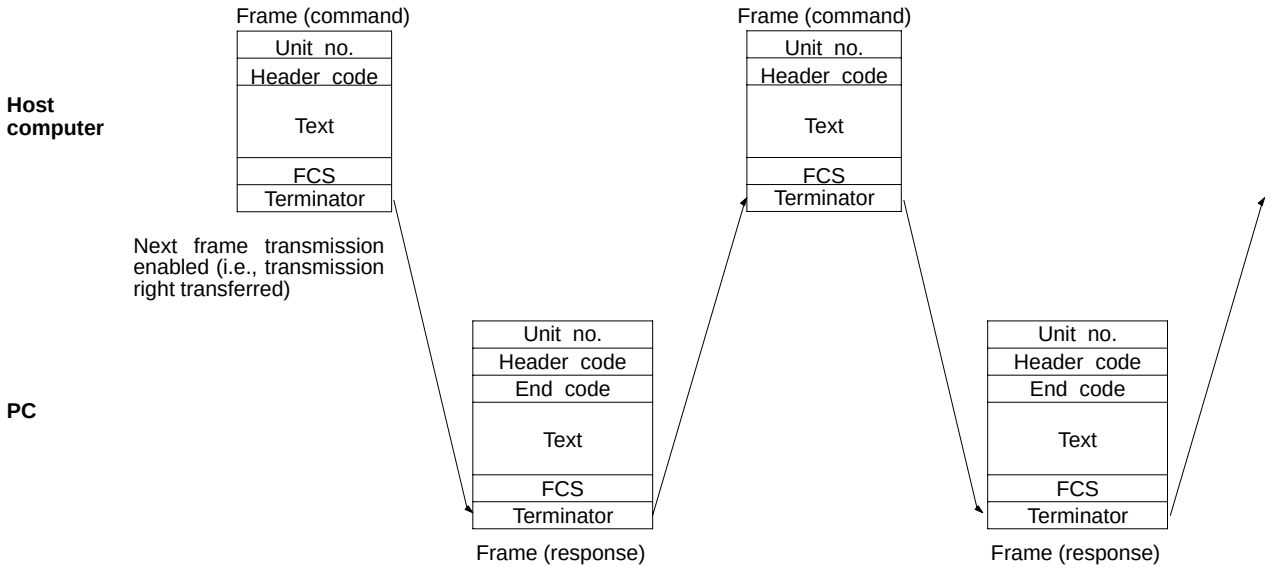
Frame Transmission and Reception

Commands and responses are exchanged in the order shown in the illustration below. The block of data transferred in a single transmission is called a “frame”. A single frame is configured of a maximum of 131 characters of data.

The right to send a frame is called the “transmission right”. The Unit that has the transmission right is the one that can send a frame at any given time. The transmission right is traded back and forth between the host computer and the PC each time a frame is transmitted. The transmission right is passed from the transmitting Unit to the receiving Unit when either a terminator (the code that marks the end of a command or response) or a delimiter (the code that sets frames apart) is received.

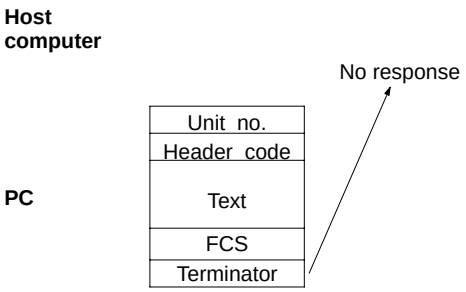
Commands from Host

In host link communications, the host computer ordinarily has the transmission right first and initiates the communications. The PC then automatically sends a response.



Commands from PC

It is also possible in host link communications for the PC to send commands to the host computer. In this case it is the PC that has the transmission right and initiates the communications.



When commands are issued to the host computer, the data is transmitted in one direction from the PC to the host computer. If a response to a command is required use a host link communications command to write the response from the host computer to the PC.

11-2 Command and Response Formats

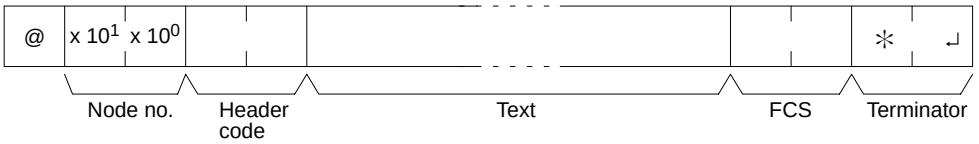
This section explains the formats for the commands and responses that are exchanged in host link communications.

11-2-1 Commands from the Host Computer

When a command is issued from the host computer, the command and response formats are as shown below.

Command Format

When transmitting a command from the host computer, prepare the command data in the format shown below.



@
An "@" symbol must be placed at the beginning.

Node No.
Identifies the PC communicating with the host computer.
Specify the node number set for the PC in the PC Setup (DM 6648, DM 6653).

Header Code
Set the 2-character command code.

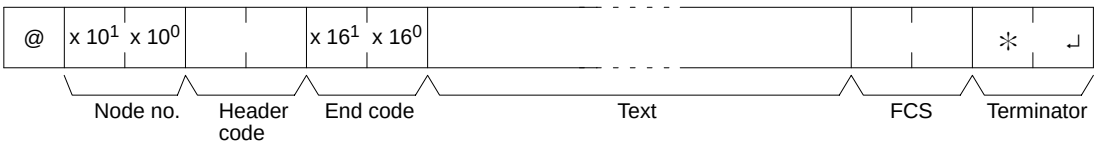
Text
Set the command parameters.

FCS
Set a 2-character Frame Check Sequence code. See page 405.

Terminator
Set two characters, "*" and the carriage return (CHR$(13)) to indicate the end of the command.

Response Format

The response from the PC is returned in the format shown below. Prepare a program so that the response data can be interpreted and processed.



@, Node No., Header Code
Contents identical to those of the command are returned.

End Code
The completion status of the command (e.g., whether or not an error has occurred) is returned.

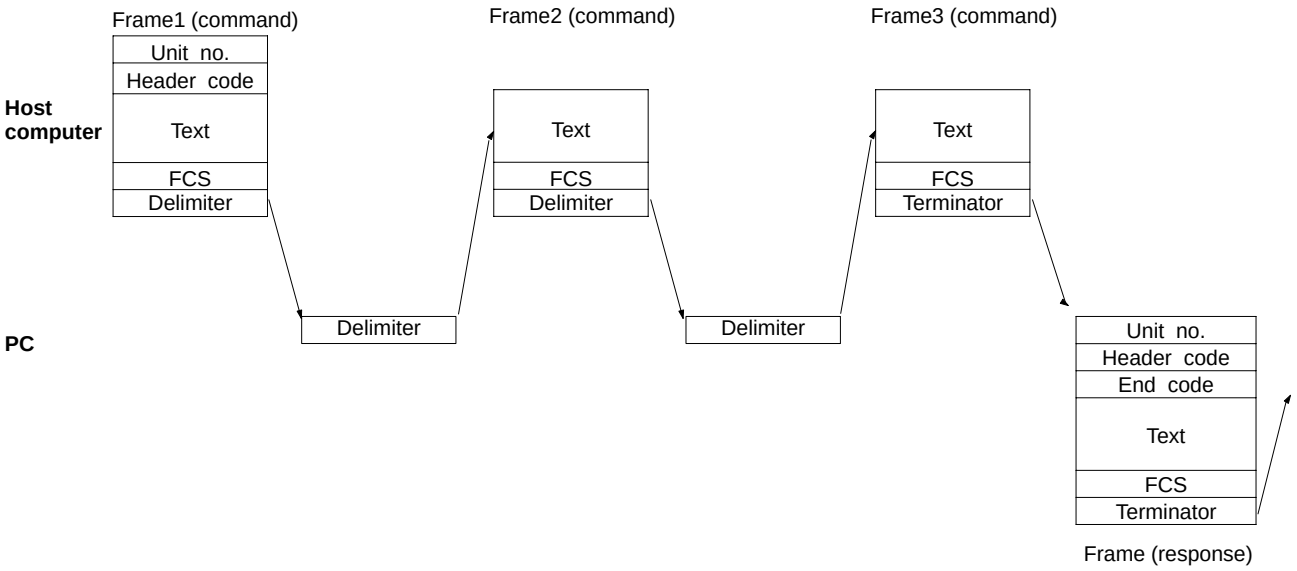
Text
Text is returned only when there is data such as read data.

FCS, Terminator
Refer to the corresponding explanations under "Command Format".

Long Transmissions

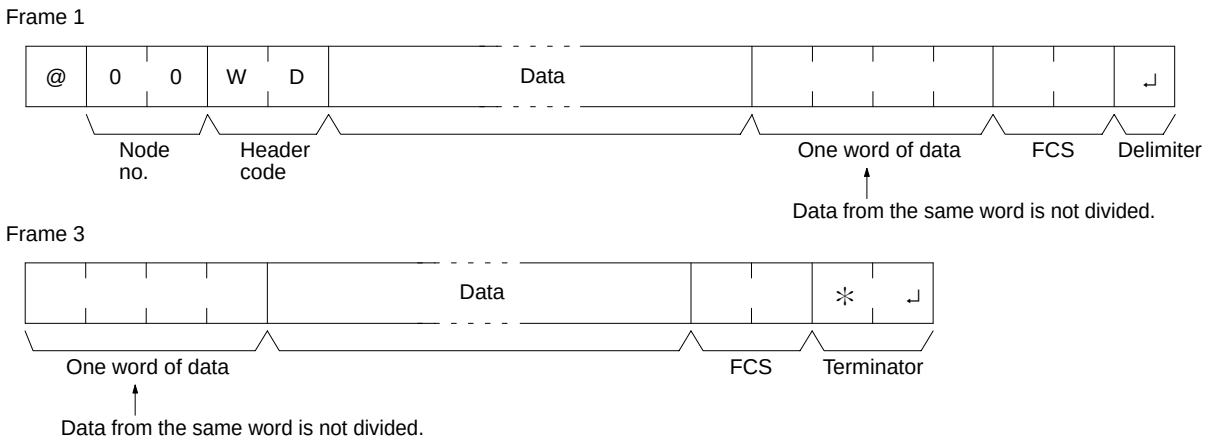
The largest block of data that can be transmitted as a single frame is 131 characters. A command or response of 132 characters or more must therefore be divided into more than one frame before transmission. When a transmission is split, the ends of the first and intermediate frames are marked by a delimiter instead of a terminator.

As each frame is transmitted, the receiving node waits for the delimiter to be transmitted. After the delimiter has been transmitted, the next frame will then be sent. This procedure is repeated until the entire command or response has been transmitted. The following diagram shows an example of host link communications addressed to a PC.



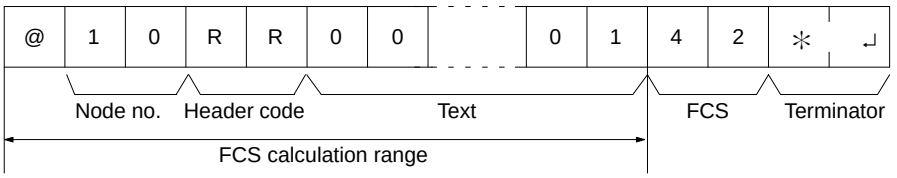
Precautions for Long Transmissions

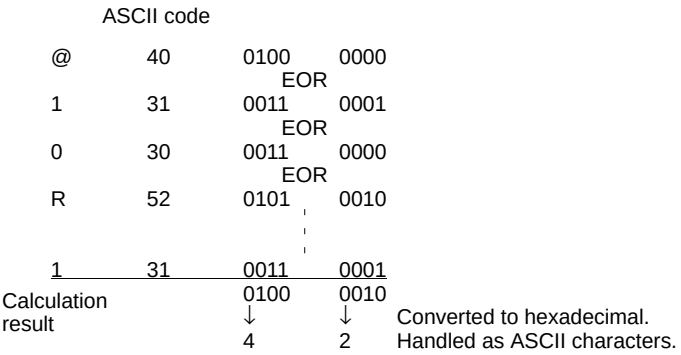
When dividing commands such as WR, WL, WC, or WD that execute write operations, be careful not to divide into separate frames data that is to be written into a single word. As shown in the illustration below, be sure to divide frames so that they coincide with the divisions between words.



FCS (Frame Check Sequence) When a frame is transmitted, an FCS is placed just before the delimiter or terminator in order to check whether any data error has been generated. The FCS is 8-bit data converted into two ASCII characters. The 8-bit data is the result of an EXCLUSIVE OR performed on the data from the beginning of the frame until the end of the text in that frame (i.e., just before the FCS). Calculating the FCS each

time a frame is received and checking the result against the FCS that is included in the frame makes it possible to check for data errors in the frame.





Example Program for FCS

This example shows a BASIC subroutine program for executing an FCS check on a frame received by the host computer.

```
400 *FCSCHECK
410 L=LEN(RESPONSE$) ' ..... Data transmitted and received
420 Q=0:FCSCK$=" "
430 A$=RIGHT$(RESPONSE$,1)
440 PRINT RESPONSE$,AS,L
450 IF A$="*" THEN LENGS=LEN(RESPONSE$)-3
              ELSE LENGS=LEN(RESPONSE$)-2
460 FCSP$=MID$(RESPONSE$,LENGS+1,2) ' ... FCS data received
470 FOR I=1 TO LENGS ' ..... Number of characters in FCS
480 Q=ASC(MID$(RESPONSE$,I,1)) XOR Q
490 NEXT I
500 FCSD$=HEX$(Q)
510 IF LEN(FCSD$)=1 THEN FCSD$="0"+FCSD$ 'FCS result
520 IF FCSD$<>FCSP$ THEN FCSCK$="ERR"
530 PRINT"FCSD$=";FCSD$,"FCSP$=";FCSP$,"FCSCK$=";FCSCK$
540 RETURN
```

Note

1. Normal reception data includes the FCS, delimiter or terminator, and so on. When an error occurs in transmission, however the FCS or some other data may not be included. Be sure to program the system to cover this possibility.
2. In this program example, the CR code (CHR$(13)) is not entered for RESPONSE$. When including the CR code, make the changes in lines 430 and 450.

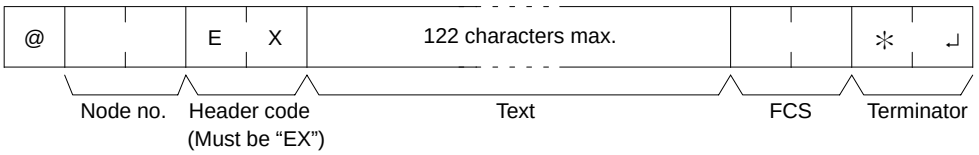
11-2-2 Commands from the PC

In host link communications, commands are ordinarily sent from the host computer to the PC, but it is also possible for commands to be sent from the PC to the host computer. In Host Link Mode, any data can be transmitted from the PC to the host computer. To send a command to the host computer, use the TRANSMIT instruction (TXD(--)) in the PC program in Host Link Mode.

TXD(—) outputs data from the specified port (the RS-232C port or the peripheral port). Refer to page 299 for details on using TXD(—).

Reception Format

When TXD(—) is executed, the data stored in the words beginning with the first send word is converted to ASCII and output to the host computer as a host link command in the format shown below. The "@" symbol, node number, header code, FCS, and delimiter are all added automatically when the transmission is sent. At the host computer, it is necessary to prepare in advance a program for interpreting and processing this format.



One byte of data (2 digits hexadecimal) is converted to two characters in ASCII for transmission, the amount of data in the transmission is twice the amount of words specified for TXD(—). The maximum number of characters for transmission is 122 and the maximum number of bytes that can be designated for TXD(—) is one half of that, or 61.

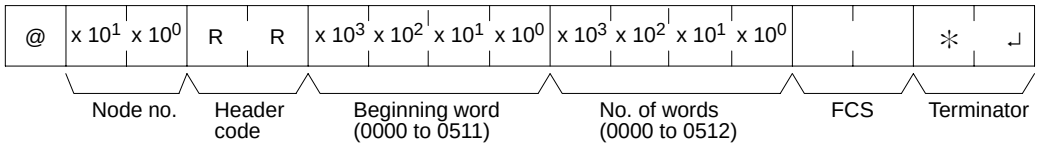
11-3 Host Link Commands

This section explains the commands that can be issued from the host computer to the PC.

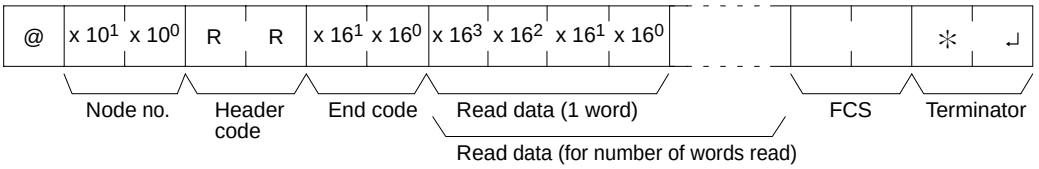
11-3-1 IR/SR AREA READ –– RR

Reads the contents of the specified number of IR and SR words, starting from the specified word.

Command Format



Response Format



Parameters

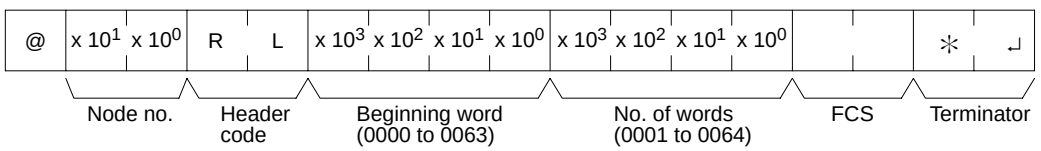
Read Data (Response)

The contents of the number of words specified by the command are returned in hexadecimal as a response. The words are returned in order, starting with the specified beginning word.

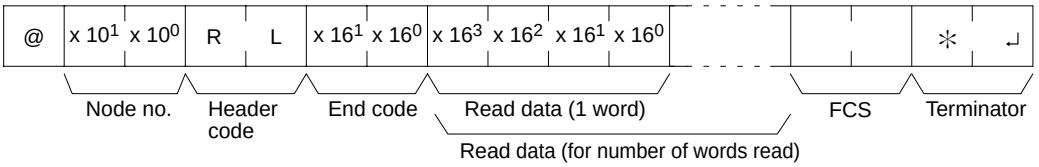
11-3-2 LR AREA READ –– RL

Reads the contents of the specified number of LR words, starting from the specified word.

Command Format



Response Format



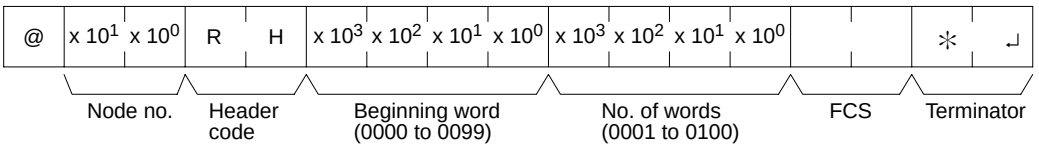
Parameters

Read Data (Response)
The contents of the number of words specified by the command are returned in hexadecimal as a response. The words are returned in order, starting with the specified beginning word.

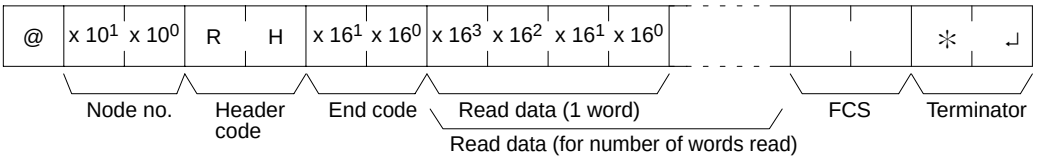
11-3-3 HR AREA READ –– RH

Reads the contents of the specified number of HR words, starting from the specified word.

Command Format



Response Format



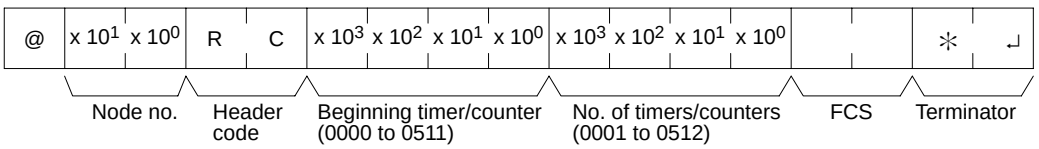
Parameters

Read Data (Response)
The contents of the number of words specified by the command are returned in hexadecimal as a response. The words are returned in order, starting with the specified beginning word.

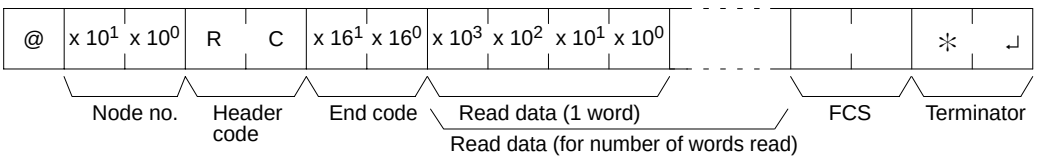
11-3-4 PV READ –– RC

Reads the contents of the specified number of timer/counter PVs (present values), starting from the specified timer/counter.

Command Format



Response Format



Parameters

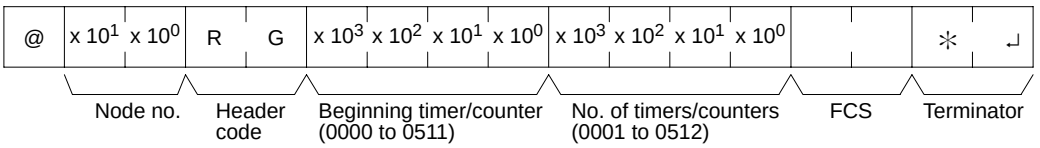
Read Data (Response)
The number of present values specified by the command is returned in hexade-

cimal as a response. The PVs are returned in order, starting with the specified beginning timer/counter.

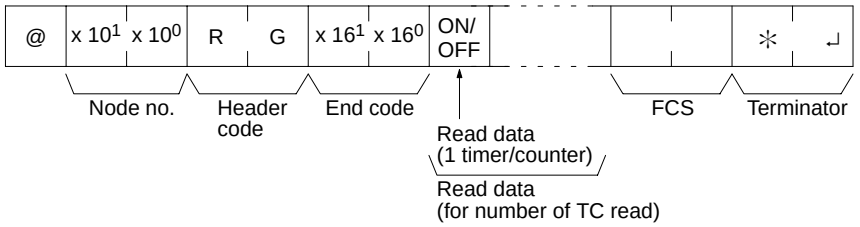
11-3-5 TC STATUS READ –– RG

Reads the status of the Completion Flags of the specified number of timers/counters, starting from the specified timer/counter.

Command Format



Response Format



Parameters

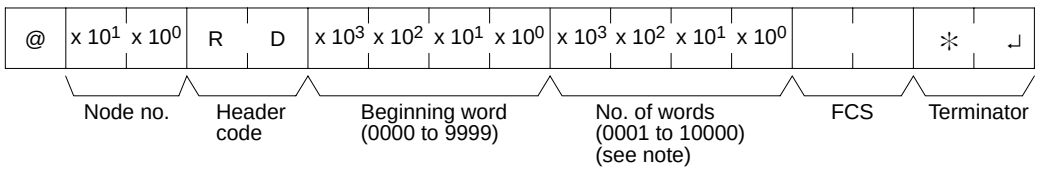
Read Data (Response)

The status of the number of Completion Flags specified by the command is returned as a response. “1” indicates that the Completion Flag is ON.

11-3-6 DM AREA READ –– RD

Reads the contents of the specified number of DM words, starting from the specified word.

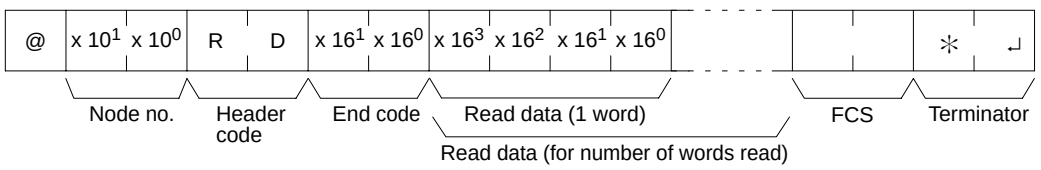
Command Format



Note

1. If 10,000 words have to be read, specify the number of words to be read as 0000.
2. DM 6656 to DM 6999 do not exist. An error will not, however, result if you try to read these words. Instead, “0000” will be returned as a response.

Response Format



Parameters

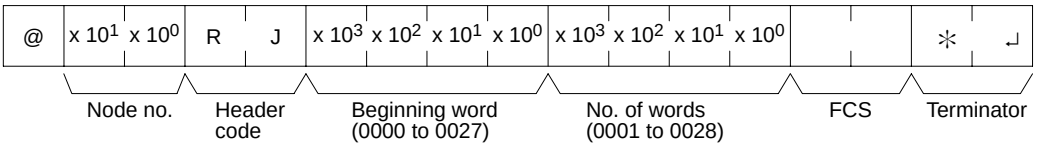
Read Data (Response)

The contents of the number of words specified by the command are returned in hexadecimal as a response. The words are returned in order, starting with the specified beginning word.

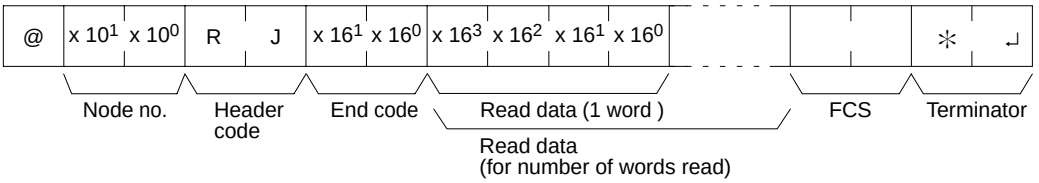
11-3-7 AR AREA READ –– RJ

Reads the contents of the specified number of AR words, starting from the specified word.

Command Format



Response Format



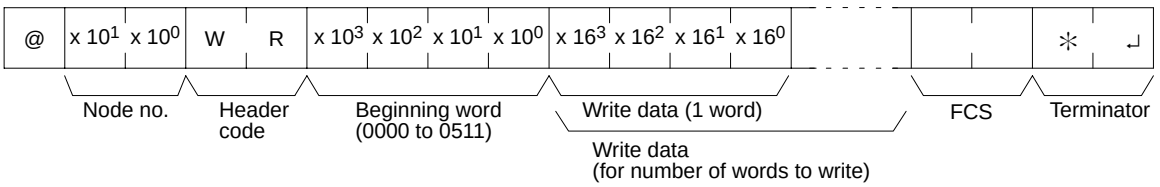
Parameters

Read Data (Response)
The contents of the number of words specified by the command are returned in hexadecimal as a response. The words are returned in order, starting with the specified beginning word.

11-3-8 IR/SR AREA WRITE –– WR

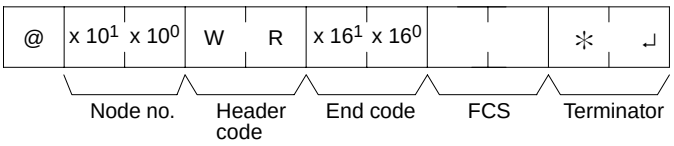
Writes data to the IR and SR areas, starting from the specified word. Writing is done word by word.

Command Format



Note Data cannot be written to words 253 to 255. If there is an attempt to write to these words, no error will result, but nothing will be written to these words.

Response Format



Parameters

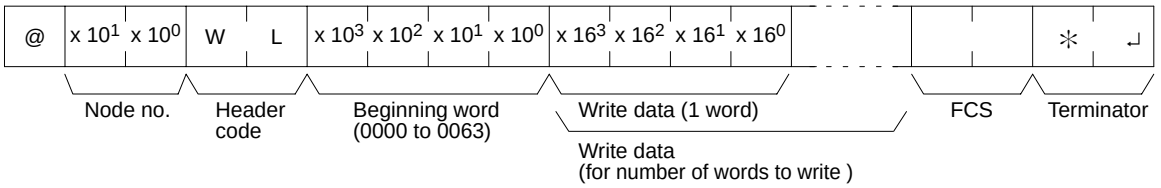
Write Data (Command)
Specify in order the contents of the number of words to be written to the IR or SR area in hexadecimal, starting with the specified beginning word.

Note If data is specified for writing which exceeds the allowable range, an error will be generated and the writing operation will not be executed. If, for example, 511 is specified as the beginning word for writing,and two words of data are specified, then 0512 will become the last word for writing data, and the command will not be executed because SR 512 is beyond the writeable range.

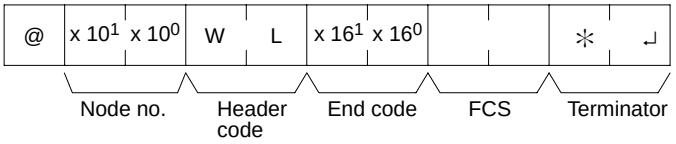
11-3-9 LR AREA WRITE –– WL

Writes data to the LR area, starting from the specified word. Writing is done word by word.

Command Format



Response Format



Parameters

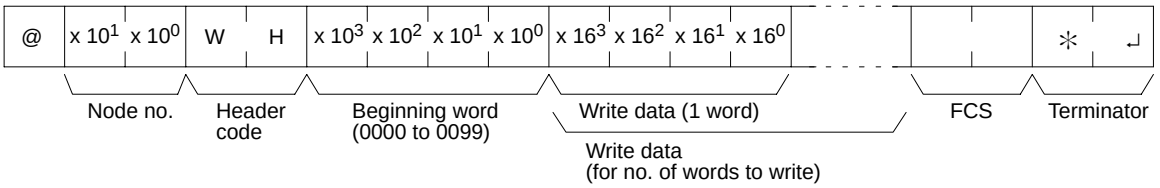
Write Data (Command)
Specify in order the contents of the number of words to be written to the LR area in hexadecimal, starting with the specified beginning word.

Note If data is specified for writing which exceeds the allowable range, an error will be generated and the writing operation will not be executed. If, for example, 60 is specified as the beginning word for writing and five words of data are specified, then 64 will become the last word for writing data, and the command will not be executed because LR 64 is beyond area boundary.

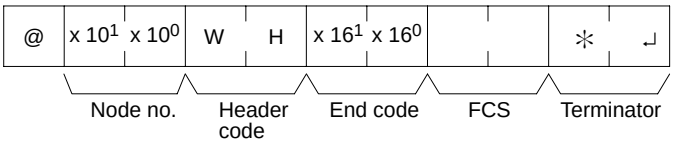
11-3-10 HR AREA WRITE –– WH

Writes data to the HR area, starting from the specified word. Writing is done word by word.

Command Format



Response Format



Parameters

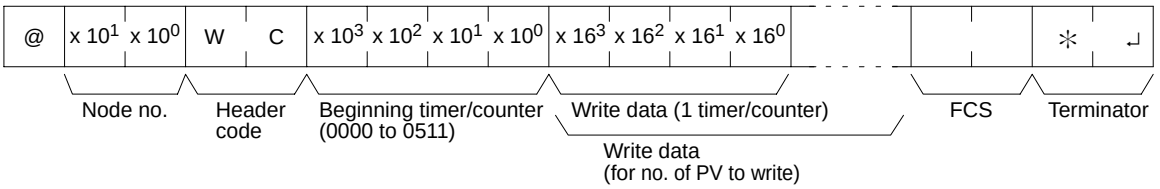
Write Data (Command)
Specify in order the contents of the number of words to be written to the HR area in hexadecimal, starting with the specified beginning word.

Note If data is specified for writing which exceeds the allowable range, an error will be generated and the writing operation will not be executed. If, for example, 98 is specified as the beginning word for writing, and three words of data are specified, then 100 will become the last word for writing data, and the command will not be executed because HR 100 is beyond area boundary.

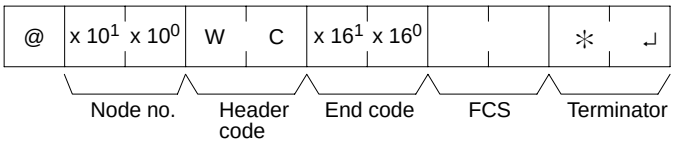
11-3-11 PV WRITE –– WC

Writes the PVs (present values) of timers/counters starting from the specified timer/counter.

Command Format



Response Format



Parameters

Write Data (Command)

Specify in decimal numbers (BCD) the present values for the number of timers/counters that are to be written, starting from the beginning timer/counter.

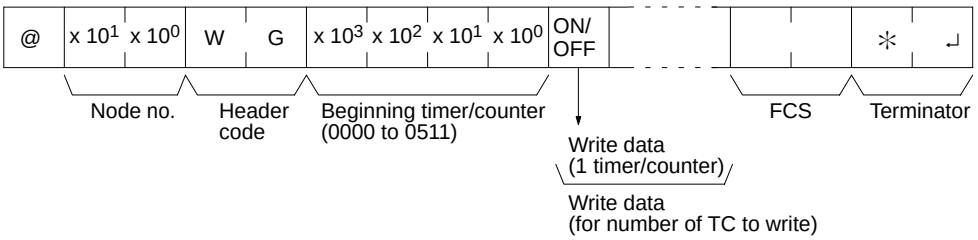
Note

1. When this command is used to write data to the PV area, the Completion Flags for the timers/counters that are written will be turned OFF.
2. If data is specified for writing which exceeds the allowable range, an error will be generated and the writing operation will not be executed. If, for example, 510 is specified as the beginning word for writing, and three words of data are specified, then 512 will become the last word for writing data, and the command will not be executed because TC 512 is beyond area boundary.

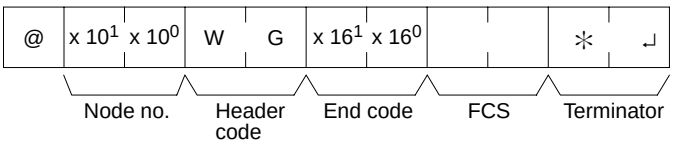
11-3-12 TC STATUS WRITE –– WG

Writes the status of the Completion Flags for timers and counters in the TC area, starting from the specified timer/counter (number). Writing is done number by number.

Command Format



Response Format



Parameters

Write Data (Command)

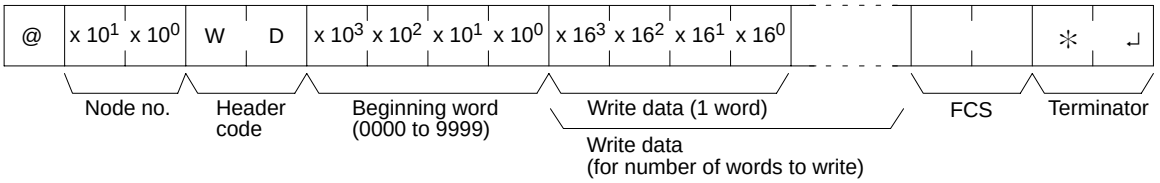
Specify the status of the Completion Flags, for the number of timers/counters to be written, in order (from the beginning word) as ON (i.e., “1”) or OFF (i.e., “0”). When a Completion Flag is ON, it indicates that the time or count is up.

Note If data is specified for writing which exceeds the allowable range, an error will be generated and the writing operation will not be executed. If, for example, 510 is specified as the beginning word for writing, and three words of data are specified, then 512 will become the last word for writing data, and the command will not be executed because TC 512 is beyond area boundary.

11-3-13 DM AREA WRITE –– WD

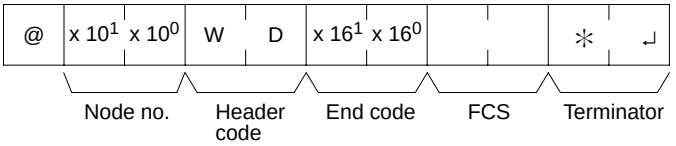
Writes data to the DM area, starting from the specified word. Writing is done word by word.

Command Format



Note DM 6656 to DM 6999 do not exist. An error will not, however, result if you try to write to these words.

Response Format



Parameters

Write Data (Command)

Specify in order the contents of the number of words to be written to the DM area in hexadecimal, starting with the specified beginning word.

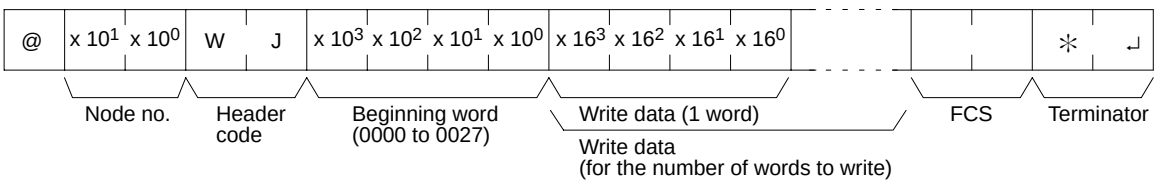
Note

1. If data is specified for writing which exceeds the allowable range, an error will be generated and the writing operation will not be executed. If, for example, 9998 is specified as the beginning word for writing, and three words of data are specified, then 10000 will become the last word for writing data, and the command will not be executed because DM 10000 is beyond the writeable range.
2. Be careful about the configuration of the DM area, as it varies depending on the CPU model.

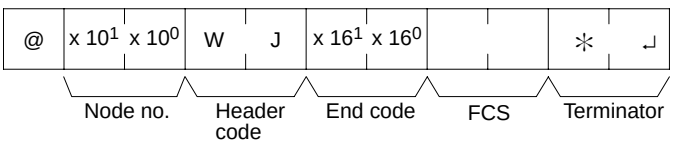
11-3-14 AR AREA WRITE –– WJ

Writes data to the AR area, starting from the specified word. Writing is done word by word.

Command Format



Response Format



Parameters

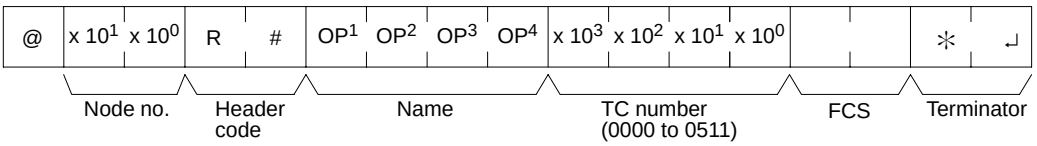
Write Data (Command)
Specify in order the contents of the number of words to be written to the AR area in hexadecimal, starting with the specified beginning word.

Note If data is specified for writing which exceeds the allowable range, an error will be generated and the writing operation will not be executed. If, for example, 26 is specified as the beginning word for writing, and three words of data are specified, then 28 will become the last word for writing data, and the command will not be executed because AR 28 is beyond the writeable range.

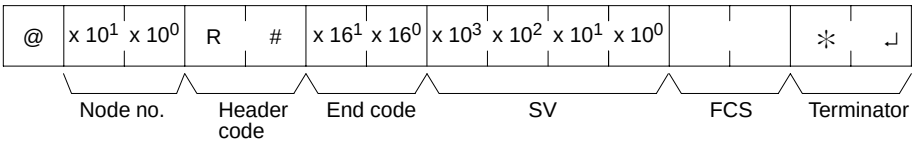
11-3-15 SV READ 1 –– R#

Searches for the first instance of a TIM, TIMH(15), CNT, CNTR(12), or TTIM(87) instruction with the specified TC number in the user's program and reads the PV, which assumed to be set as a constant. The SV that is read is a 4-digit decimal number (BCD). The program is searched from the beginning, and it may therefore take approximately 10 seconds to produce a response.

Command Format



Response Format



Parameters

Name, TC Number (Command)
Specify the instruction for reading the SV in "Name". Make this setting in 4 characters. In "TC number", specify the timer/counter number used for the instruction.

Instruction name				Classification	TC number range
OP1	OP2	OP3	OP4		
T	I	M	(S)	TIMER	0000 to 0511
T	I	M	H	HIGH-SPEED TIMER	
C	N	T	(S)	COUNTER	
C	N	T	R	REVERSIBLE COUNTER	
T	T	I	M	TOTALIZING TIMER	

(S): Space

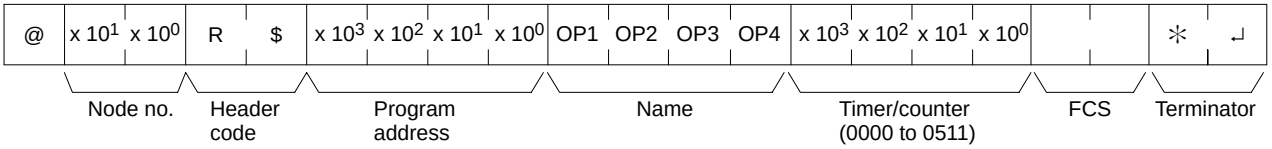
SV (Response)
The constant SV is returned.

Note
1. The instruction specified under "Name" must be in four characters. Fill any gaps with spaces to make a total of four characters.
2. If the same instruction is used more than once in a program, only the first one will be read.
3. Use this command only when it is definite that a constant SV has been set.

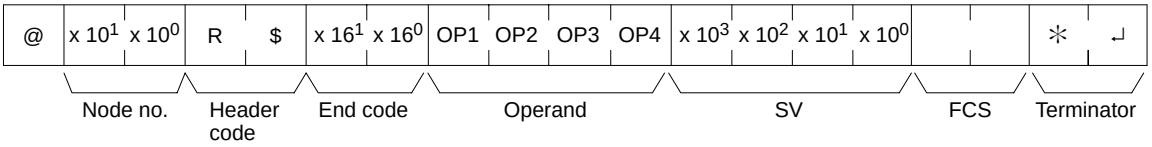
11-3-16 SV READ 2 –– R$

Reads the constant SV or the word address where the SV is stored. The SV that is read is a 4-digit decimal number (BCD) written as the second operand for the TIM, TIMH(15), CNT, CNTR(12), or TTIM(87) instruction at the specified program address in the user's program. This can only be done with a program of less than 10K.

Command Format



Response Format



Parameters

Name, TC Number (Command)
Specify the name of the instruction for reading the SV in "Name". Make this setting in 4 characters. In "TC number", specify the timer/counter number used by the instruction.

Instruction name				Classification	TC number range
OP1	OP2	OP3	OP4		
T	I	M	(S)	TIMER	0000 to 0511
T	I	M	H	HIGH-SPEED TIMER	
C	N	T	(S)	COUNTER	
C	N	T	R	REVERSIBLE COUNTER	
T	T	I	M	TOTALIZING TIMER	

(S): Space

Operand, SV (Response)
The name that indicates the SV classification is returned to "Operand", and either the word address where the SV is stored or the constant SV is returned to "SV".

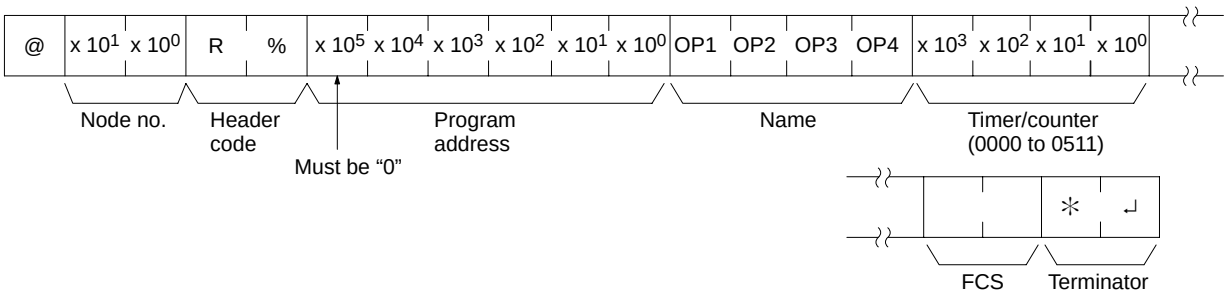
Operand				Classification	Constant or word address
OP1	OP2	OP3	OP4		
C	I	O	(S)	IR or SR	0000 to 0511
L	R	(S)	(S)	LR	0000 to 0063
H	R	(S)	(S)	HR	0000 to 0099
A	R	(S)	(S)	AR	0000 to 0027
D	M	(S)	(S)	DM	0000 to 6655
D	M	*	(S)	DM (indirect)	0000 to 6655
C	O	N	(S)	Constant	0000 to 9999

Note The instruction name specified under "Name" must be in four characters. Fill any gaps with spaces to make a total of four characters.

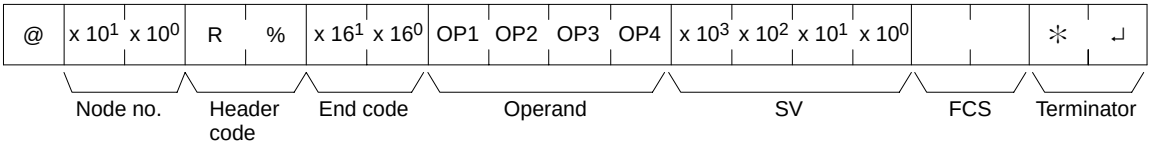
11-3-17 SV READ 3 –– R%

Reads the constant SV or the word address where the SV is stored. The SV that is read is a 4-digit decimal number (BCD) written in the second word of the TIM, TIMH(15), CNT, CNTR(12), or TTIM(87) instruction at the specified program address in the user's program. With this command, program addresses can be specified for a program of 10K or more.

Command Format



Response Format



Parameters

Name, TC Number (Command)

Specify the name of the instruction for reading the SV in "Name". Make this setting in 4 characters. In "TC number", specify the timer/counter number used by the instruction.

Instruction name				Classification	TC number range
OP1	OP2	OP3	OP4		
T	I	M	(S)	TIMER	0000 to 0511
T	I	M	H	HIGH-SPEED TIMER	
C	N	T	(S)	COUNTER	
C	N	T	R	REVERSIBLE COUNTER	
T	T	I	M	TOTALIZING TIMER	

(S): Space

Operand, SV (Response)

The name that indicates the SV classification is returned to "Operand", and either the word address where the SV is stored or the constant SV is returned to "SV".

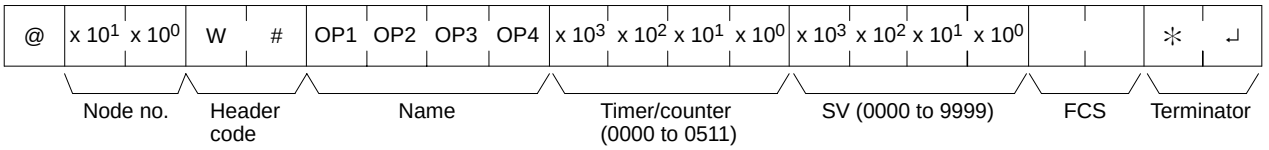
Operand				Classification	Constant or word address
OP1	OP2	OP3	OP4		
C	I	O	(S)	IR or SR	0000 to 0511
L	R	(S)	(S)	LR	0000 to 0063
H	R	(S)	(S)	HR	0000 to 0099
A	R	(S)	(S)	AR	0000 to 0027
D	M	(S)	(S)	DM	0000 to 6655
D	M	*	(S)	DM (indirect)	0000 to 6655
C	O	N	(S)	Constant	0000 to 9999

Note The instruction name specified under "Name" must be in four characters. Fill any gaps with spaces to make a total of four characters.

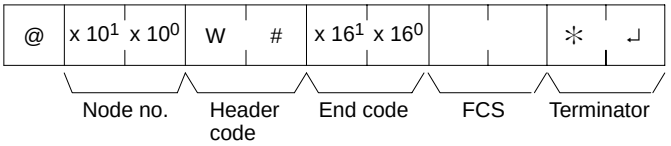
11-3-18 SV CHANGE 1 –– W#

Searches for the first instance of the specified TIM, TIMH(15), CNT, CNTR(12), or TTIM(87) instruction in the user's program and changes the SV to new constant SV specified in the second word of the instruction. The program is searched from the beginning, and it may therefore take approximately 10 seconds to produce a response.

Command Format



Response Format



Parameters

Name, TC Number (Command)

In "Name", specify the name of the instruction, in four characters, for changing the SV. In "TC number", specify the timer/counter number used for the instruction.

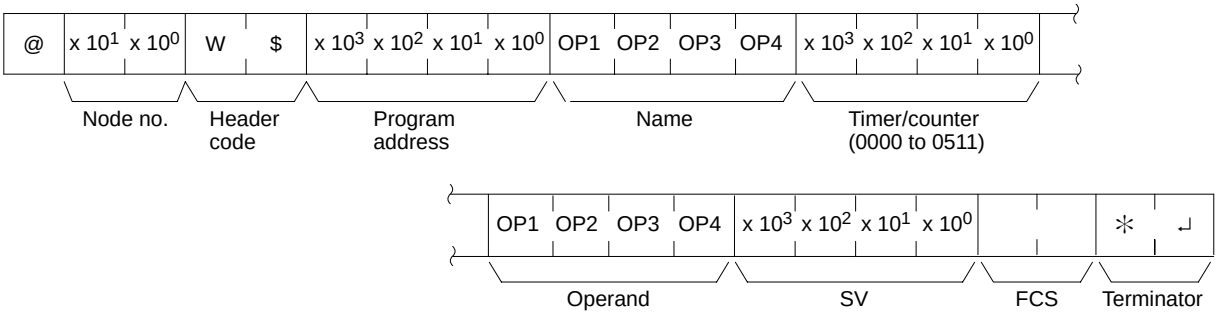
Instruction name				Classification	TC number range
OP1	OP2	OP3	OP4		
T	I	M	(S)	TIMER	0000 to 0511
T	I	M	H	HIGH-SPEED TIMER	
C	N	T	(S)	COUNTER	
C	N	T	R	REVERSIBLE COUNTER	
T	T	I	M	TOTALIZING TIMER	

(S): Space

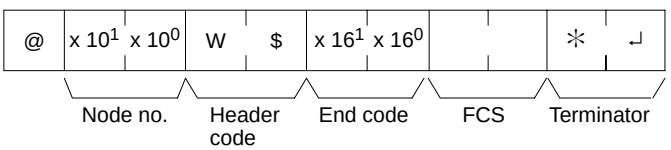
11-3-19 SV CHANGE 2 –– W$

Changes the contents of the second word of the TIM, TIMH(15), CNT, CNTR(12), or TTIM(87) at the specified program address in the user's program. This can only be done with a program of less than 10K.

Command Format



Response Format



Parameters

Name, TC Number (Command)

In “Name”, specify the name of the instruction, in four characters, for changing the SV. In “TC number”, specify the timer/counter number used for the instruction.

Instruction name				Classification	TC number range
OP1	OP2	OP3	OP4		
T	I	M	(S)	TIMER	0000 to 0511
T	I	M	H	HIGH-SPEED TIMER	
C	N	T	(S)	COUNTER	
C	N	T	R	REVERSIBLE COUNTER	
T	T	I	M	TOTALIZING TIMER	

(S): Space

Operand, SV (Response)

In “Operand”, specify the name that indicates the SV classification. Specify the name in four characters. In “SV”, specify either the word address where the SV is stored or the constant SV.

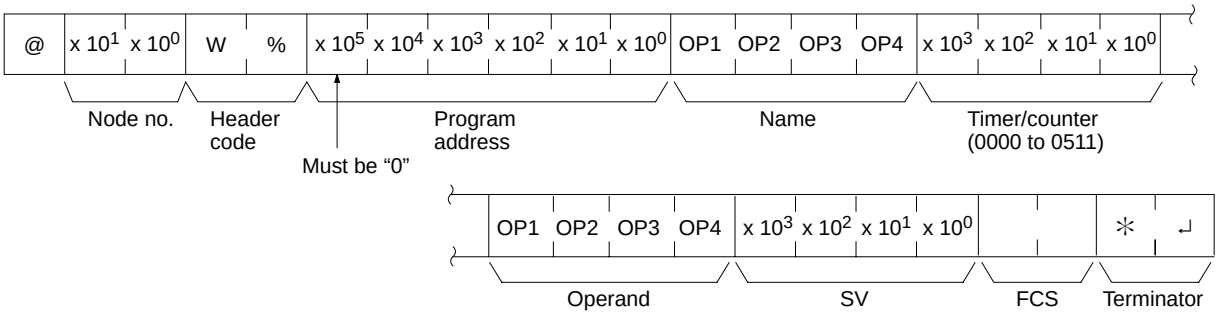
Operand				Classification	Constant or word address
OP1	OP2	OP3	OP4		
C	I	O	(S)	IR or SR	0000 to 0511
L	R	(S)	(S)	LR	0000 to 0063
H	R	(S)	(S)	HR	0000 to 0099
A	R	(S)	(S)	AR	0000 to 0027
D	M	(S)	(S)	DM	0000 to 6655
D	M	*	(S)	DM (indirect)	0000 to 6655
C	O	N	(S)	Constant	0000 to 9999

(S): Space

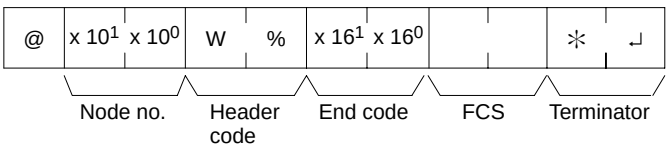
11-3-20 SV CHANGE 3 –– W%

Changes the contents of the second word of the TIM, TIMH(15), CNT, CNTR(12), or TTIM(87) at the specified program address in the user’s program. With this command, program address can be specified for a program of more than 10K.

Command Format



Response Format



Parameters

Name, TC Number (Command)
In “Name”, specify the name of the instruction, in four characters, for changing the SV. In “TC number”, specify the timer/counter number used for the instruction.

Instruction name				Classification	TC number range
OP1	OP2	OP3	OP4		
T	I	M	(S)	TIMER	0000 to 0511
T	I	M	H	HIGH-SPEED TIMER	
C	N	T	(S)	COUNTER	
C	N	T	R	REVERSIBLE COUNTER	
T	T	I	M	TOTALIZING TIMER	

(S): Space

Operand, New SV (Response)
In “Operand”, specify the name that indicates the SV classification. Specify the name in four characters. In “SV”, specify either the word address where the SV is stored or the constant SV.

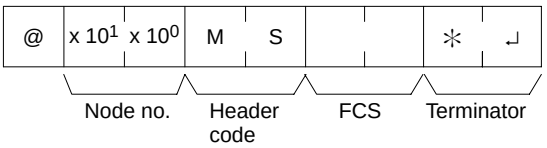
Operand				Classification	Constant or word address
OP1	OP2	OP3	OP4		
C	I	O	(S)	IR or SR	0000 to 0511
L	R	(S)	(S)	LR	0000 to 0063
H	R	(S)	(S)	HR	0000 to 0099
A	R	(S)	(S)	AR	0000 to 0027
D	M	(S)	(S)	DM	0000 to 6655
D	M	*	(S)	DM (indirect)	0000 to 6655
C	O	N	(S)	Constant	0000 to 9999

(S): Space

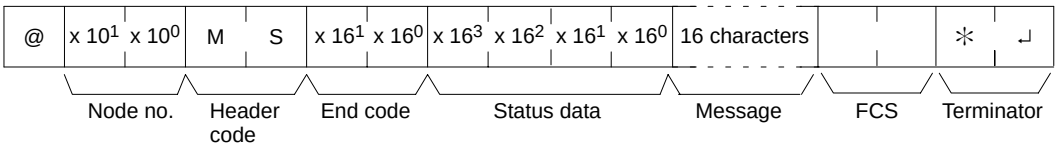
11-3-21 STATUS READ –– MS

Reads the PC operating conditions.

Command Format



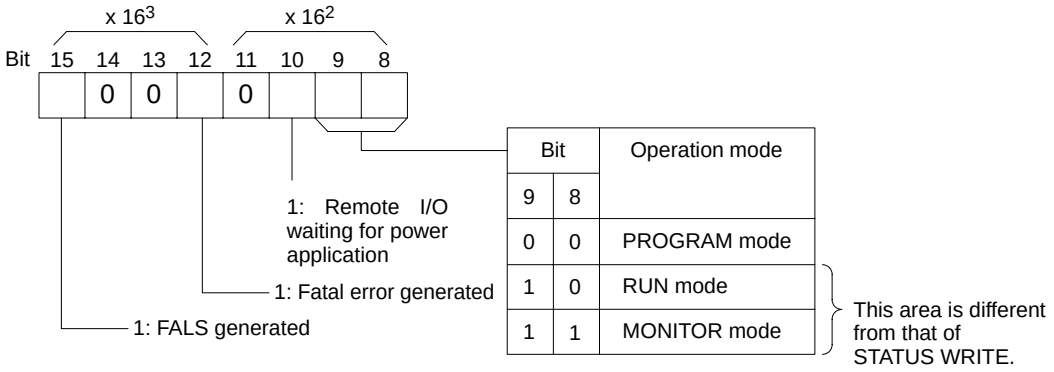
Response Format

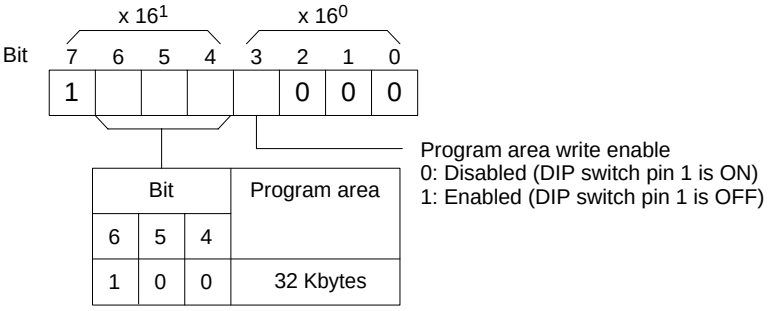


Parameters

Status Data, Message (Response)

"Status data" consists of four digits (two bytes) hexadecimal. The leftmost byte indicates CPU operation mode, and the rightmost byte indicates the size of the program area.



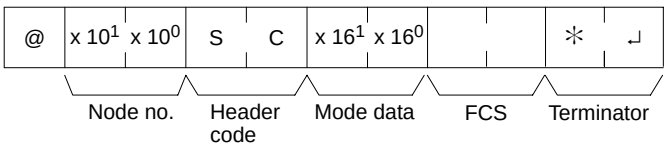


"Message" indicates the FAL/FALS number generated at the point when the command is executed. When there is no message, this is omitted.

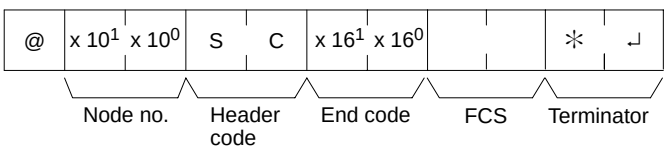
11-3-22 STATUS WRITE –– SC

Changes the PC operating mode.

Command Format



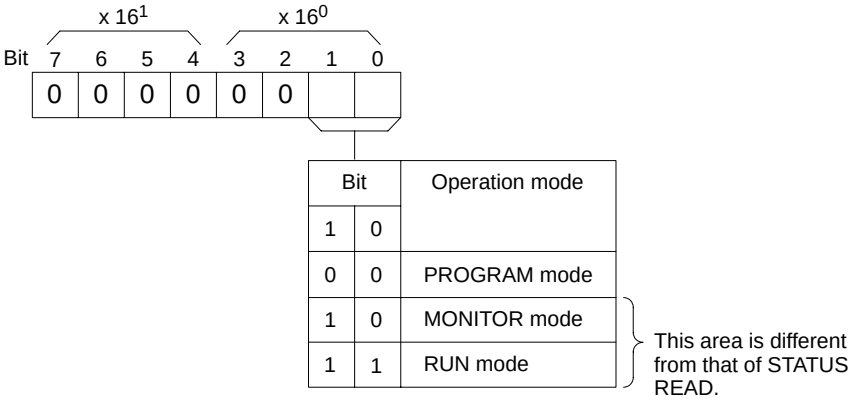
Response Format



Parameters

Mode Data (Command)

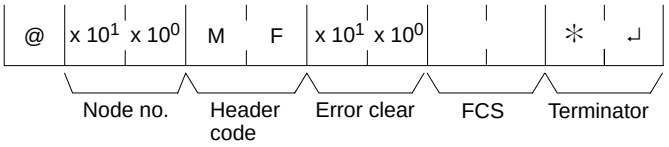
"Mode data" consists of two digits (one byte) hexadecimal. With the leftmost two bits, specify the PC operating mode. Set all of the remaining bits to "0".



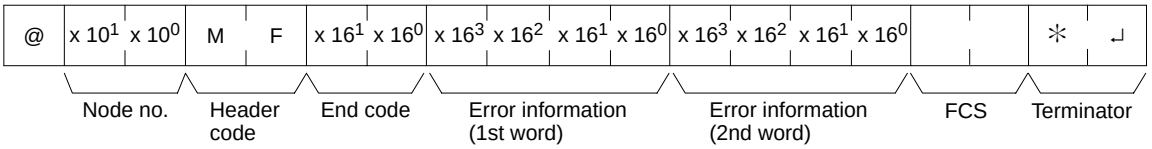
11-3-23 ERROR READ –– MF

Reads and clears errors in the PC. Also checks whether previous errors have been cleared.

Command Format



Response Format



Parameters

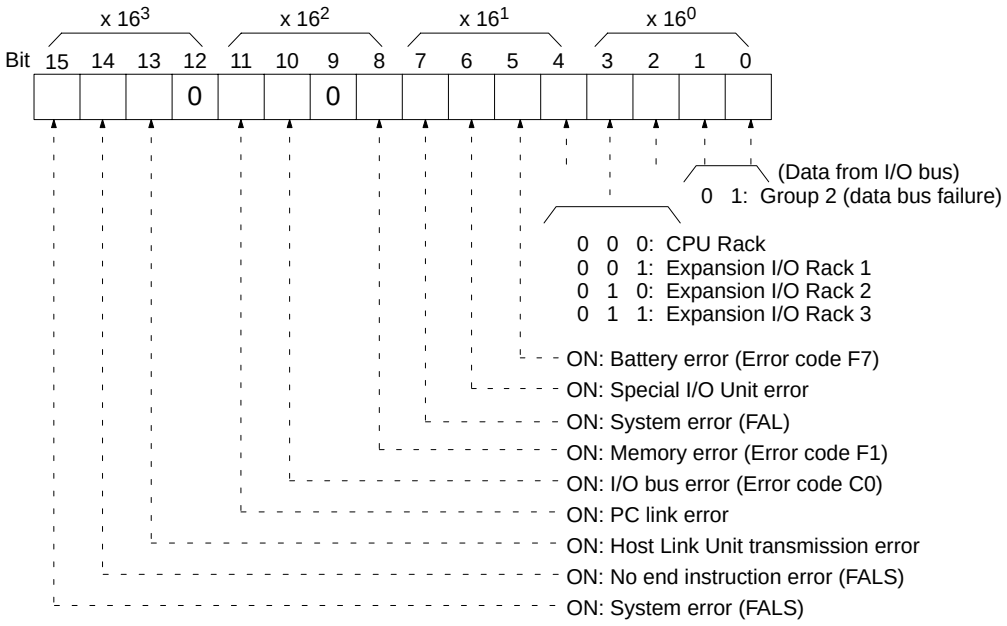
Error Clear (Command)

Specify 01 to clear errors and 00 to not clear errors (BCD). Fatal errors can be cleared only when the PC is in PROGRAM mode.

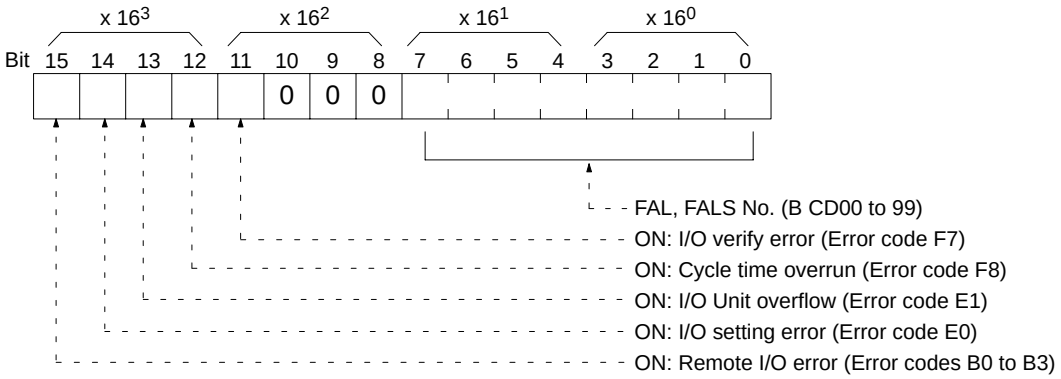
Error Information (Response)

The error information comes in two words.

1st word



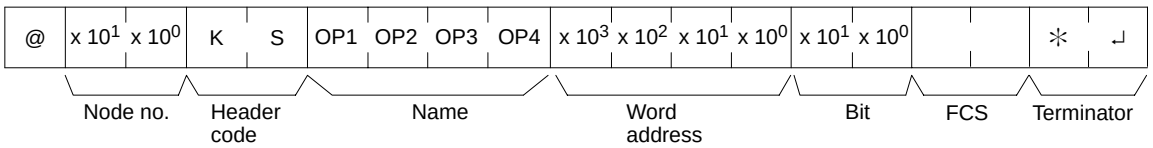
2nd word



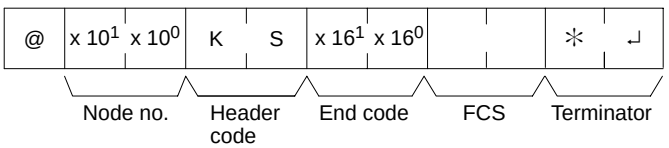
11-3-24 FORCED SET –– KS

Force sets a bit in the IR, SR, LR, HR, AR, or TC area. Once a bit has been forced set or reset, that status will be retained until FORCED SET/RESET CANCEL (KC) is transmitted.

Command Format



Response Format



Parameters

Name, Word address, Bit (Command)

In “Name”, specify the area (i.e., IR, SR, LR, HR, AR, or TC) that is to be forced set. Specify the name in four characters. In “Word address”, specify the address of the word, and in “Bit” the number of the bit that is to be forced set.

Name				Classification	Word address setting range	Bit
OP1	OP2	OP3	OP4			
C	I	O	(S)	IR or SR	0000 to 0511	00 to 15 (decimal)
L	R	(S)	(S)	LR	0000 to 0063	
H	R	(S)	(S)	HR	0000 to 0099	
A	R	(S)	(S)	AR	0000 to 0027	
T	I	M	(S)	Completion Flag (timer)	0000 to 0511	Always 00.
T	I	M	H	Completion Flag (high-speed timer)		
C	N	T	(S)	Completion Flag (counter)		
C	N	T	R	Completion Flag (reversible counter)		
T	T	I	M	Completion Flag (totalizing timer)		

(S): Space

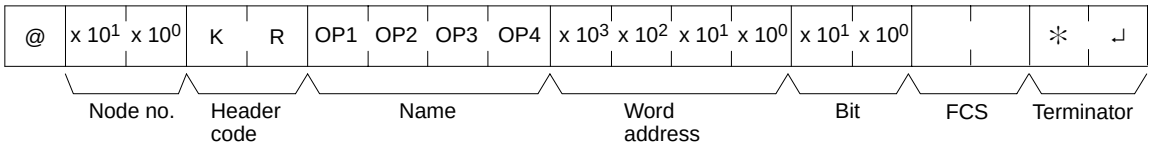
Note

1. The area specified under “Name” must be in four characters. Fill any gaps with spaces to make a total of four characters.
2. Words 253 to 255 cannot be set when the CIO Area is specified.

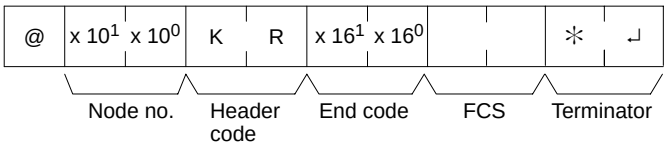
11-3-25 FORCED RESET –– KR

Force resets a bit in an IR, SR, LR, HR, AR, or TC area. Once a bit has been forced set or reset, that status will be retained until FORCED SET/RESET CANCEL (KC) is transmitted.

Command Format



Response Format



Parameters

Name, Word address, Bit (Command)

In “Name”, specify the area (i.e., IR, SR, LR, HR, AR, or TC) that is to be forced reset. Specify the name in four characters. In “Word address”, specify the address of the word, and in “Bit” the number of the bit that is to be forced reset.

Name				Classification	Word address setting range	Bit
OP1	OP2	OP3	OP4			
C	I	O	(S)	IR or SR	0000 to 0511	00 to 15 (decimal)
L	R	(S)	(S)	LR	0000 to 0063	
H	R	(S)	(S)	HR	0000 to 0099	
A	R	(S)	(S)	AR	0000 to 0027	
T	I	M	(S)	Completion Flag (timer)	0000 to 0511	Always 00.
T	I	M	H	Completion Flag (high-speed timer)		
C	N	T	(S)	Completion Flag (counter)		
C	N	T	R	Completion Flag (reversible counter)		
T	T	I	M	Completion Flag (totalizing timer)		

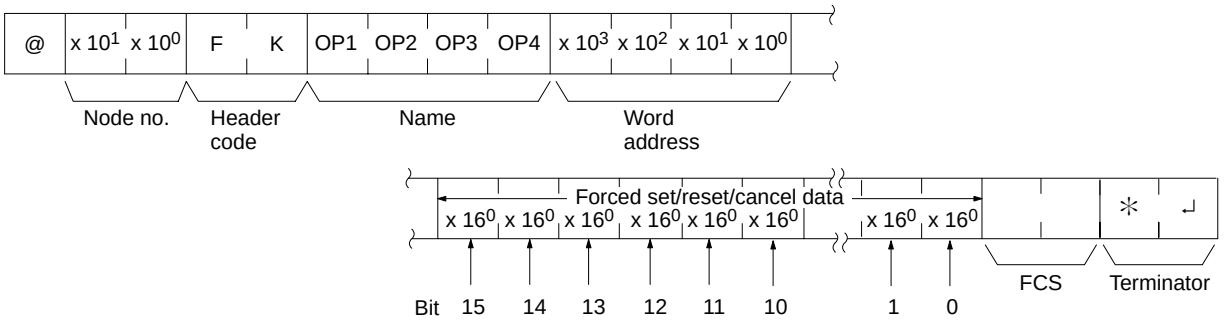
(S): Space

Note
1. The area specified under “Name” must be in four characters. Fill any gaps with spaces to make a total of four characters.
2. Words 253 to 255 cannot be set when the CIO Area is specified.

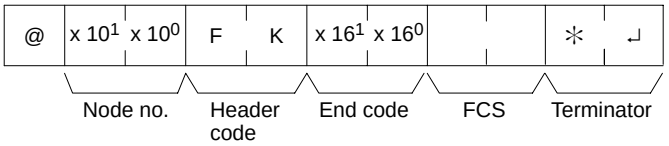
11-3-26 MULTIPLE FORCED SET/RESET –– FK

Force sets, force resets, or cancels the status of the bits in one word in the IR, SR, LR, HR, AR, or TC area.

Command Format



Response Format



Parameters

Name, Word address (Command)
In “Name”, specify the area (i.e., IR, SR, LR, HR, AR, or TC) that is to be forced set or reset. Specify the name in four characters. In “Word address”, specify the address of the word that is to be forced set or reset.

Name				Classification	Word address setting range	Bit
OP1	OP2	OP3	OP4			
C	I	O	(S)	IR or SR	0000 to 0511	00 to 15 (decimal)
L	R	(S)	(S)	LR	0000 to 0063	
H	R	(S)	(S)	HR	0000 to 0099	
A	R	(S)	(S)	AR	0000 to 0027	
T	I	M	(S)	Completion Flag (timer)	0000 to 0511	Always 00.
T	I	M	H	Completion Flag (high-speed timer)		
C	N	T	(S)	Completion Flag (counter)		
C	N	T	R	Completion Flag (reversible counter)		
T	T	I	M	Completion Flag (totalizing timer)		

(S): Space

Note Words 253 to 255 cannot be set when the CIO Area is specified.

Forced set/reset/cancel Data (Command)

A separate hexadecimal digit is used to specify the desired process for each bit in the specified word, bits 00 to bit 5. The bits that are merely set or reset may change status the next time the program is executed, but bits that are force-set or force-reset will maintain the forced status until it is cleared. If the item specified under “Name” is a timer or counter, the status of the Completion Flag can be force-set or force-reset using bit 15, and all other bits will be ignored. Only force-setting and force-resetting are possible for timers/counters.

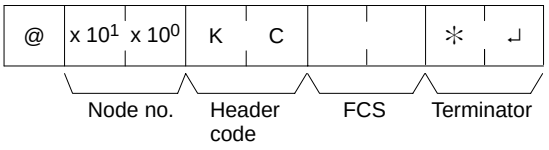
Bits 00 to 15	Setting
00	Bit status not changed
02	Reset
03	Set
04	Forced-reset
05	Forced-set
08	Forced set/reset status cancel

Note The item specified under “Name” must be in four characters. Fill any gaps with spaces to make a total of four characters.

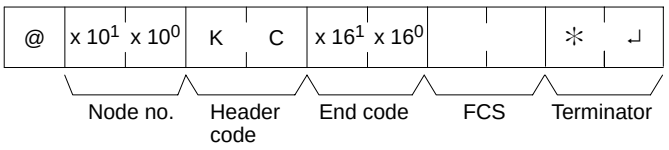
11-3-27 FORCED SET/RESET CANCEL –– KC

Cancels all forced set and forced reset bits (including those set by FORCED SET, FORCED RESET, and MULTIPLE FORCED SET/RESET). If multiple bits are set, the forced status will be cancelled for all of them. It is not possible to cancel bits one by one using KC.

Command Format



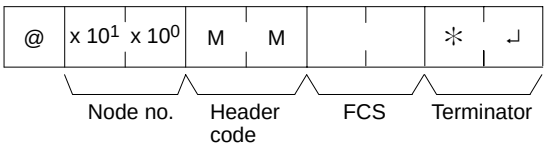
Response Format



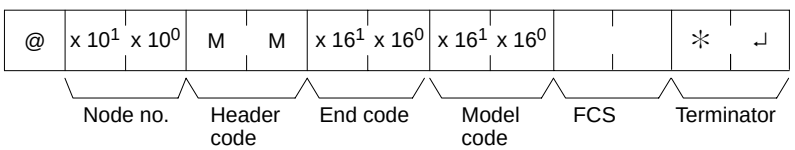
11-3-28 PC MODEL READ –– MM

Reads the model type of the PC.

Command Format



Response Format



Parameters

Model Code

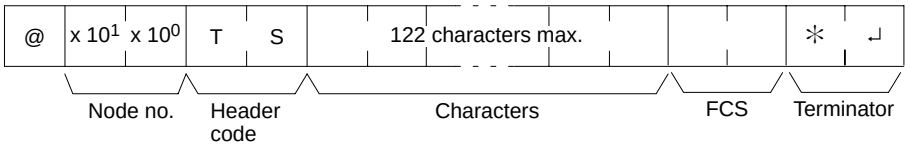
"Model code" indicates the PC model in two digits hexadecimal.

Model code	Model
01	C250
02	C500
03	C120
0E	C2000
10	C1000H
11	C2000H/CQM1
12	C20H/C28H/C40H/C200H/C200HS
20	CV500
21	CV1000
22	CV2000
40	CVM1-CPU01-E
41	CVM1-CPU11-E

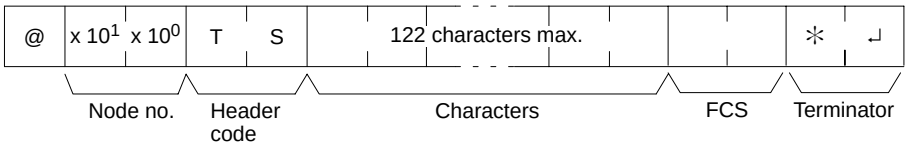
11-3-29 TEST–– TS

Returns, unaltered, one block of data transmitted from the host computer.

Command Format



Response Format



Parameters

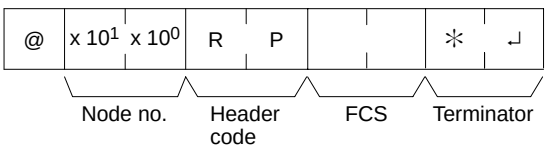
Characters (Command, Response)

For the command, this setting specifies any characters other than the carriage return (CHR$(13)). For the response, the same characters as specified by the command will be returned unaltered if the test is successful.

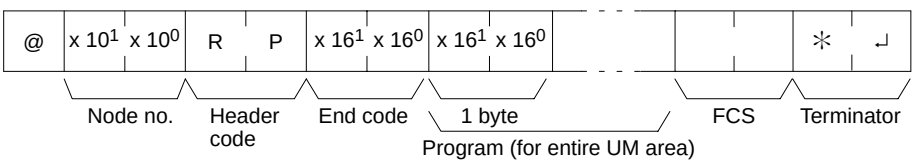
11-3-30 PROGRAM READ –– RP

Reads the contents of the PC user's program area in machine language (object code). The contents are read as a block, from the beginning to the end.

Command Format



Response Format



Parameters

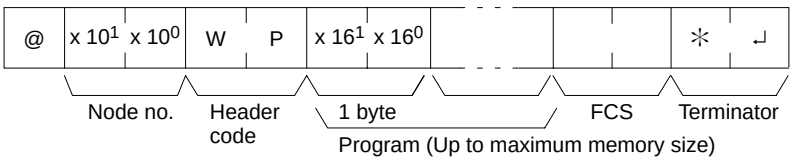
Program (Response)
The program is read from the entire program area.

Note To stop this operation in progress, execute the ABORT (XZ) command.

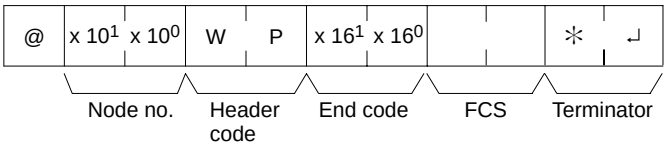
11-3-31 PROGRAM WRITE –– WP

Writes to the PC user's program area the machine language (object code) program transmitted from the host computer. The contents are written as a block, from the beginning.

Command Format



Response Format



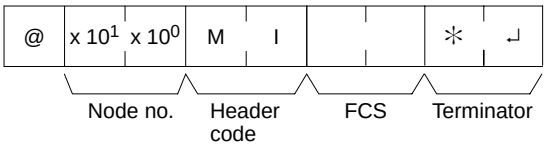
Parameters

Program ((Command)
Program data up to the maximum memory size.

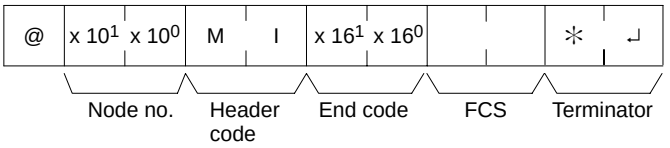
11-3-32 I/O TABLE GENERATE –– MI

Corrects the registered I/O table to match the actual I/O table.

Command Format



Response Format



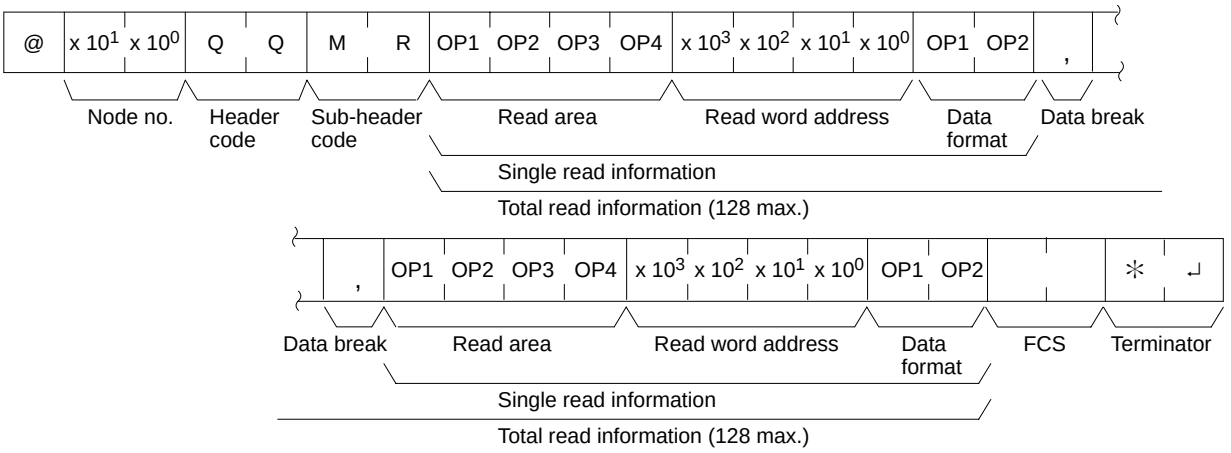
11-3-33 COMPOUND COMMAND –– QQ

Registers at the PC all of the bits, words, and timers/counters that are to be read, and reads the status of all of them as a batch.

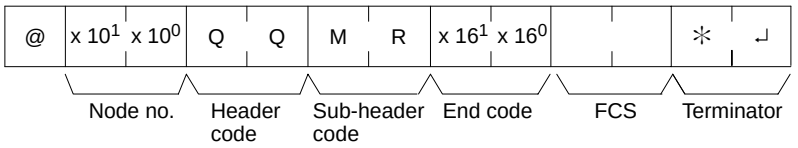
Registering Read Information

Register the information on all of the bits, words, and timers/counters that are to be read.

Command Format



Response Format



Parameters

Read Area (Command)

Specify in four-character code the area that is to be read. The codes that can be specified are listed in the following table.

Read Word address, Data Format (Command)

Depending on the area and type of data that are to be read, the information to be read is as shown in the following table. The “read data” is specified in four digits BCD, and the data format is specified in two digits BCD.

Area classification	Read data	Read area	Read word	Data format
IR or SR	Bit	C I O (S)	0000 to 0255	00 to 15 (decimal)
	Word			“CH”
LR	Bit	L R (S) (S)	0000 to 0063	00 to 15 (decimal)
	Word			“CH”
HR	Bit	H R (S) (S)	0000 to 0099	00 to 15 (decimal)
	Word			“CH”
AR	Bit	A R (S) (S)	0000 to 0027	00 to 15 (decimal)
	Bit			“CH”
Timer	Completion Flag	T I M (S)	0000 to 0511	2 characters other than “CH”
	PV			“CH”
High-speed timer	Completion Flag	T I M H	0000 to 0511	2 characters other than “CH”
	PV			“CH”
Counter	Completion Flag	C N T (S)	0000 to 0511	2 characters other than “CH”
	PV			“CH”
Reversible counter	Completion Flag	C N T R	0000 to 0511	2 characters other than “CH”
	PV			“CH”
Totalizing timer	Completion Flag	T T I M	0000 to 0511	2 characters other than “CH”
	PV			“CH”
DM	Word	D M (S) (S)	0000 to 6655	Any 2 characters

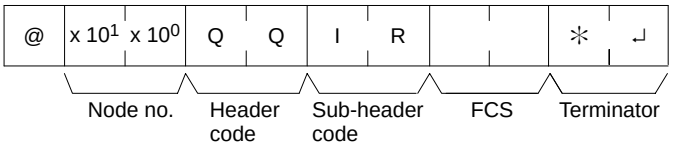
(S): Space

Data Break (Command)
The read information is specified one item at a time separated by a break code (,). The maximum number of items that can be specified is 128. (When the PV of a timer/counter is specified, however, the status of the Completion Flag is also returned, and must therefore be counted as two items.)

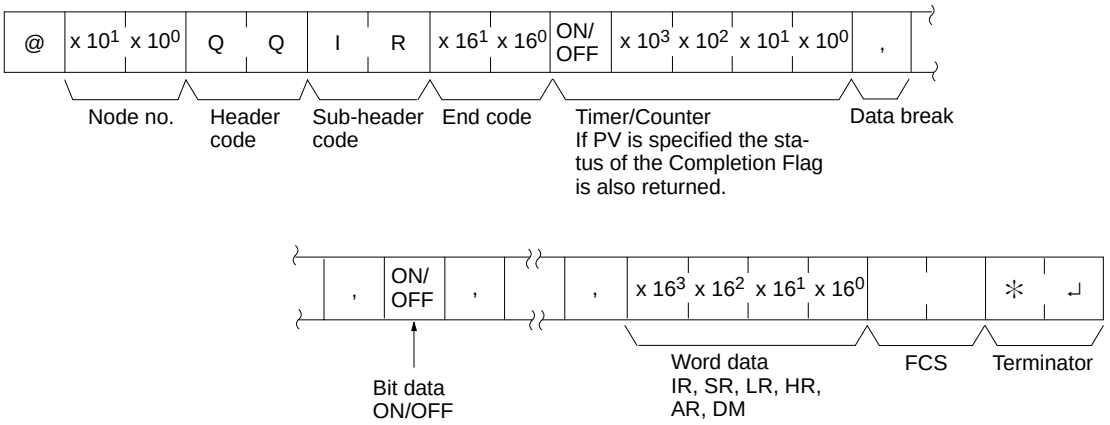
Batch Reading

The bit, word, and timer/counter status is read as a batch according to the read information that was registered with QQ.

Command Format



Response Format



Parameters

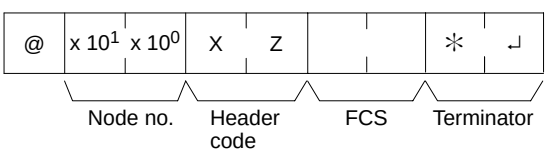
Read Data (Response)
Read data is returned according to the data format and the order in which read information was registered using QQ. If “Completion Flag” has been specified, then bit data (ON or OFF) is returned. If “Word” has been specified, then word data is returned. If “PV” has been specified for timers/counters, however, then the PV is returned following the Completion Flag.

Data Break (Response)
The break code (,) is returned between sections that are read.

11-3-34 ABORT –– XZ

Aborts the Host Link operation that is currently being processed, and then enables reception of the next command. The ABORT command does not receive a response.

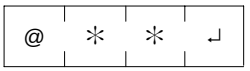
Command Format



11-3-35 INITIALIZE –– ⁎⁎

Initializes the transmission control procedure of all the PCs connected to the host computer. The INITIALIZE command does not use node numbers or FCS, and does not receive a response.

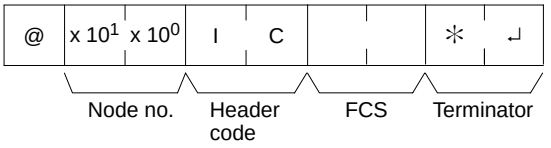
Command Format



11-3-36 Undefined Command –– IC

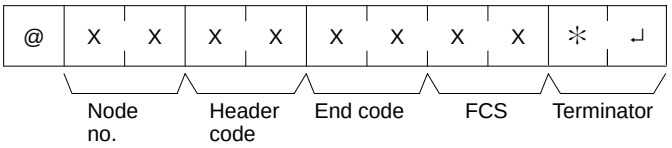
This response is returned if the header code of a command cannot be decoded. Check the header code.

Response Format



11-4 Host Link Errors

These error codes are received as the response code (end code) when a command received by the C200HS from a host computer cannot be processed. The error code format is as shown below.



The header code will vary according to the command and can contain a subcode (for composite commands).

End code	Contents	Probable cause	Corrective measures
00	Normal completion	---	---
01	Not executable in RUN mode	The command that was sent cannot be executed when the PC is in RUN mode.	Check the relation between the command and the PC mode.
02	Not executable in MONITOR mode	The command that was sent cannot be executed when the PC is in MONITOR mode.	
0B	Not executable in PROGRAM mode	The command that was sent cannot be executed when the PC is in PROGRAM mode.	This code is not presently being used.
13	FCS error	The FCS is wrong. Either the FCS calculation is mistaken or there is adverse influence from noise.	Check the FCS calculation method. If there was influence from noise, transfer the command again.
14	Format error	The command format is wrong.	Check the format and transfer the command again.
15	Entry number data error	The areas for reading and writing are wrong.	Correct the areas and transfer the command again.
16	Command not supported	The specified command does not exist.	Check the command code.
18	Frame length error	The maximum frame length was exceeded.	Divide the command into multiple frames.
19	Not executable	Items to read not registered for composite command (QQ).	Execute QQ to register items to read before attempting batch read.
23	User memory write-protected	Pin 1 on C200HS DIP switch is ON.	Turn OFF pin 1 to execute.
A3	Aborted due to FCS error in transmit data	The error was generated while a command extending over more than one frame was being executed. **Note:** The data up to that point has already been written to the appropriate area of the CPU.	Check the command data and try the transfer again.
A4	Aborted due to format error in transmit data		
A5	Aborted due to entry number data error in transmit data		
A8	Aborted due to frame length error in transmit data		
Other	---	Influence from noise was received.	Transfer the command again.

Power Interruptions

The following responses may be received from the C200HS if a power interruption occurs, including momentary interruptions. If any of these responses are received during or after a power interruption, repeat the command.

Undefined Command Response

@00IC4A*↵

No Response

If no response is received, abort the last command and resend.

Appendix A
Standard Models

C200HS Racks

Name		Specifications			Model number
Backplane (same for all Racks)		10 slots			C200H-BC101-V2
		8 slots			C200H-BC081-V2
		5 slots			C200H-BC051-V2
		3 slots			C200H-BC031-V2
CPU Rack	CPU	100 to 120/200 to 240 VAC w/built-in power supply	——		C200HS-CPU01-E
			Conforms to EC directives (see note)		C200HS-CPU01-EC
			RS-232C port		C200HS-CPU21-E
			RS-232C port and conforms to EC directives		C200HS-CPU21-EC
			RS-232C port and SYSMAC NET/ SYSMAC LINK supported		C200HS-CPU31-E
		24 VDC w/built-in power supply	Conforms to EC directives (see note)		C200HS-CPU03-E
			RS-232C port		C200HS-CPU23-E
			RS-232C port and SYSMAC NET/ SYSMAC LINK supported		C200HS-CPU33-E
	Memory Cassette	EPROM Chip; 27256; 150 ns			ROM-JD-B
		EPROM Chip; 27512; 150 ns			ROM-KD-B
		EEPROM; 16K words			C200HS-ME16K
		EPROM; 16K words			C200HS-MP16K
Expansion I/O Racks	I/O Power Supply Unit	100 to 120/200 to 240 VAC (switchable)			C200H-PS221
			Conforms to EC directives		C200H-PS221-C
		24 VDC	Conforms to EC directives (see note)		C200H-PS211
	I/O Connecting Cable (max. total length: 12 m)	30 cm			C200H-CN311
		70 cm			C200H-CN711
		2 m			C200H-CN221
		5 m			C200H-CN521
		10 m			C200H-CN131
Slave Racks	Remote I/O Slave Unit	100 to 120/200 to 240 VAC (switchable)		APF/ PCF	C200H-RT001-P
		24 VDC			C200H-RT002-P
		100 to 120/200 to 240 VAC (switchable)		Wired	C200H-RT201
			Conforms to EC directives		C200H-RT201-C
		24 VDC			C200H-RT202

Note: Units with lot numbers □□Z5 (Dec. 1995) or later.

C200H Standard I/O Units

Name		Specifications		Model number
Input Units	AC Input Unit	8 pts	100 to 120 VAC	C200H-IA121
		16 pts	100 to 120 VAC	C200H-IA122/122V
		8 pts	200 to 240 VAC	C200H-IA221
		16 pts	200 to 240 VAC	C200H-IA222/222V
	DC Input Unit	8 pts	No-voltage contact; NPN	C200H-ID001
		8 pts	No-voltage contact; PNP	C200H-ID002
		8 pts	12 to 24 VDC	C200H-ID211
		16 pts	24 VDC	C200H-ID212
	AC/DC Input Unit	8 pts	12 to 24 VAC/DC	C200H-IM211
		16 pts	24 VAC/DC	C200H-IM212
	Interrupt Input Unit[1]	8 pts	12 to 24 VDC	C200HS-INT01
Output Units	Relay Output Unit	8 pts	2 A, 250 VAC/24 VDC (For resistive loads)	C200H-OC221
		12 pts	2 A, 250 VAC/24 VDC (For resistive loads)	C200H-OC222
		16 pts	2 A, 250 VAC/24 VAC (For resistive loads)	C200H-OC225[2, 3]
		5 pts	2 A, 250 VAC/24 VDC (For resistive loads) Independent commons	C200H-OC223
		8 pts	2 A, 250 VAC/24 VDC (For resistive loads) Independent commons	C200H-OC224
		12 pts	2 A, 250 VAC/24 VDC (For resistive loads)	C200H-OC222V
		16 pts	2 A, 250 VAC/24 VDC (For resistive loads)	C200H-OC226
		8 pts	2 A, 250 VAC/24 VDC (For resistive loads) Independent commons	C200H-OC224V
	Triac Output Unit	8 pts	1 A, 120 VAC	C200H-OA121-E
		8 pts	1.2 A, 120 VAC	C200H-OA122-E
		8 pts	1 A, 200 VAC	C200H-OA221
		12 pts	0.3 A, 250 VAC	C200H-OA222V
		12 pts	0.5 A, 250 VAC	C200H-OA224
	Transistor Output Unit	8 pts	1 A, 12 to 48 VDC	C200H-OD411
		12 pts	0.3 A, 24 VDC	C200H-OD211
		16 pts	0.3 A, 24 VDC	C200H-OD212[2]
		8 pts	2.1 A, 24 VDC	C200H-OD213
		8 pts	0.8 A, 24 VDC; source type (PNP); with load short protection	C200H-OD214
		8 pts	5 to 24 VDC; source type (PNP)	C200H-OD216
		12 pts	5 to 24 VDC; source type (PNP)	C200H-OD217
		16 pts	1.0 A, 24 VDC; source type (PNP); with load short protection	C200H-OD21A
Analog Timer Unit		4 timers	0.1 to 1 s/1 to 10 s/10 to 60 s/1 min to 10 min (switchable)	C200H-TM001
	Variable Resistor Connector	Connector w/lead wire (2 m) for 1 external resistor		C4K-CN223
Standard B7A Interface Units		16 input pts	Connects to B7A Link Terminals.	C200H-B7AI1
		16 output pts		C200H-B7AO1

Note 1. If the Interrupt Input Unit is mounted on an Expansion I/O Rack, the interrupt function cannot be used and the Interrupt Input Unit will be treated as an ordinary 8-point Input Unit. Moreover, Interrupt Input Units cannot be used on Slave Racks. In addition, Interrupt Input Units require that a version 2 (i.e., model numbers with a "-V2" suffix) Backplane be used at the CPU Rack. If an earlier version Backplane is mounted, the interrupt function cannot be used.

2. Transistor Output Unit C200H-OD212 and Contact Output Unit C200H-OC225 must be mounted to either a C200H-BC031-V1/V2, C200H-BC051-V1/V2, C200H-BC081-V1/V2, or C200H-BC101-V1/V2 Backplane.

3. The C200H-OC225 might overheat if more than 8 outputs are turned ON simultaneously.

C200H Group-2 High-density I/O Units

Name	Specifications		Model number
DC Input Unit	32 pts.	24 VDC	C200H-ID216
			C200H-ID218
	64 pts.	24 VDC	C200H-ID217
			C200H-ID219
Transistor Output Unit	32 pts.	16 mA 4.5 VDC to 100 mA 26.4 VDC	C200H-OD218
		0.5 A (5A/Unit) 24 VDC	C200H-OD21B
	64 pts.	16 mA 4.5 VDC to 100 mA 26.4 VDC	C200H-OD219

C200H Group-2 B7A Interface Units

Name	Specifications		Model number
Group-2 B7A Interface Units	32 input pts	Connects to B7A Link Terminals.	C200H-B7A12
	32 output pts		C200H-B7A02
	16 input pts and 16 output points		C200H-B7A21
	32 input pts and 32 output points		C200H-B7A22

C200H Special I/O Units

All of the following are classified as Special I/O Units except for the ASCII Unit, which is an Intelligent I/O Unit.

Name		Specifications		Model number
High-density I/O Units	DC Input Unit	32 pts	5 VDC (TTL inputs); with high-speed input function	C200H-ID501
		32 pts	24 VDC; with high-speed inputs	C200H-ID215
	Transistor Output Unit	32 pts	0.1 A, 24 VDC (usable as 128-point dynamic output unit)	C200H-OD215
		32 pts	35 mA, 5 VDC (TTL outputs) (usable as 128-point dynamic output unit)	C200H-OD501
	DC Input/ Transistor Output Unit	16 input/ 16 output pts	12-VDC inputs; with high-speed input function 0.1 A , 12-VDC outputs (usable as 128-point dynamic input unit)	C200H-MD115
		16 input/ 16 output pts	24-VDC inputs; with high-speed input function 0.1 A , 24-VDC outputs (usable as 128-point dynamic input unit)	C200H-MD215
		16 input/ 16 output pts	5 VDC (TTL inputs); with high speed input function 35 mA, 5 VDC Output (TTL outputs) (usable as 128-point dynamic input unit)	C200H-MD501
Analog I/O Units	Analog Input Unit	4 to 20 mA, 1 to 5/0 to 10 V (switchable); 4 inputs		C200H-AD001
		4 to 20 mA, 1 to 5/0 to 10/–10 to 10 V (switchable); 8 inputs		C200H-AD002
	Analog Output Unit	4 to 20 mA, 1 to 5/0 to 10 V (switchable); 2 outputs		C200H-DA001
Temperature Sensor Unit		Thermocouple (K(CA) or J(IC)) (switchable); 4 inputs		C200H-TS001
		Thermocouple (K(CA) or L(Fe-CuNi)) (switchable); 4 inputs		C200H-TS002
		Platinum resistance thermometer (JPt) (switchable), DIN standards; 4 inputs		C200H-TS101
		Platinum resistance thermometer (Pt) (switchable); 4 inputs		C200H-TS102
Temperature Control Unit		Thermocouple	Transistor output	C200H-TC001
			Voltage output	C200H-TC002
			Current output	C200H-TC003
		Pt resistance thermometer	Transistor output	C200H-TC101
			Voltage output	C200H-TC102
			Current output	C200H-TC103

Name		Specifications		Model number
Heat/Cool Temperature Control Unit		Thermocouple	Transistor output	C200H-TV001
			Voltage output	C200H-TV002
			Current output	C200H-TV003
		Pt resistance thermometer	Transistor output	C200H-TV101
			Voltage output	C200H-TV102
			Current output	C200H-TV103
PID Control Unit		Transistor output		C200H-PID01
		Voltage output		C200H-PID02
		Current output		C200H-PID03
Position Control Unit		1 axis	Pulse output; speeds: 1 to 99,990 pps	C200H-NC111
		1 axis	Directly connectable to servomotor driver; compatible with line driver; speeds: 1 to 250,000 pps	C200H-NC112
		2 axis	1 to 250000. pps. 53 pts per axis	C200H-NC211
Cam Positioner Unit		Detects angles of rotation by means of a resolver and provides ON and OFF outputs at specified angles. A maximum of 48 cam outputs (16 external outputs and 32 internal outputs) maximum are available.		C200H-CP114
High-speed Counter Unit		1 axis	Pulse input; counting speed: 50 kcps; 5 VDC/12 VDC/24 VDC	C200H-CT001-V1
		1 axis	Pulse input; counting speed: 75 kcps; RS-422 line driver	C200H-CT002
ASCII Unit		EEPROM		C200H-ASC02
I/D Sensor Unit		Local application, electromagnetic coupling		C200H-IDS01-V1
		Remote application, microwave transmissions		C200H-IDS21
	Read/Write Head	Electromagnetic type		V600-H series
		Microwave type		V620-H series
	Data Carrier (see note)	SRAM type for V600-H series.		V600-D□□R□□
		EEPROM type for V600-H series.		V600-D□□P□□
Voice Unit		60 messages max.; message length: 32, 48, or 64 s (switchable)		C200H-OV001
	Connecting Cable	RS-232C		C200H-CN224
Fuzzy Logic Unit		Up to 8 inputs and 4 outputs. (I/O to and from specified data area words)		C200H-FZ001
	Fuzzy Support Software	Available on either 3.5” or 5.25” floppy disks.		C500-SU981-E

Note For Read/Write Head and Data Carrier combinations, refer to the *V600 FA ID System R/W Heads and EEPROM Data Carriers Operation Manual and Supplement* or *V600 FA ID System R/W Heads and SRAM Data Carriers Operation Manual and Supplement*.

C200H Link Units

Name	Specifications			Model number
Host Link Unit	Rack-mounting	C200H only	APF/PCF	C200H-LK101-PV1
			RS-422	C200H-LK202-V1
			RS-232C	C200H-LK201-V1
PC Link Unit	Multilevel		RS-485	C200H-LK401
Remote I/O Master Unit	Up to two per PC; connectable to up to 5 Slaves per PC total		APF/PCF	C200H-RM001-PV1
			Wired	C200H-RM201

SYSMAC LINK Unit/SYSMAC NET Link Unit

The SYSMAC LINK Units and SYSMAC NET Link Unit can only be used with the C200HS-CPU31-E and C200HS-CPU33-E CPUs.

Name		Specifications			Model number
SYSMAC LINK Unit		Wired via coaxial cable. Bus Connection Unit required separately. One C1000H-CE001 F Adapter included.			C200HS-SLK22
		Wired via optical fiber cable. Bus Connection Unit required separately. An Optical Fiber Cable Bracket must be used to support an optical cable connected to the SYSMAC LINK Unit.			C200HS-SLK12
	Terminator	One required for each node at ends of System			C1000H-TER01
	Attachment Stirrup	Provided with SYSMAC LINK Unit			C200H-TL001
	F Adapter	---			C1000H-CE001
	F Adapter Cover	---			C1000H-COV01
	Communications Cable	Coaxial cables	Manufactured by Hitachi		ECXF5C-2V
			Manufactured by Fujigura		5C-2V
	Auxiliary Power Supply Unit	Supplies backup power to either one or two SYSMAC LINK Units. One C200H-CN111 Power Connecting Cable included.			C200H-APS03
SYSMAC NET Link Unit		Bus Connection Unit required separately. An Optical Fiber Cable Bracket must be used to support an optical cable connected to the SYSMAC NET Link Unit.			C200HS-SNT32
	Power Supply Adapter	Required when supplying power from Central Power Supply		For 1 Unit	C200H-APS01
				For 2 Units	C200H-APS02
	Power Cable	Connects Power Supply Adapter and SYSMAC NET Link Unit		For 1 Unit	C200H-CN111
				For 2 Units	C200H-CN211
Bus Connection Unit		Connects SYSMAC LINK Unit or SYSMAC NET Link Unit to C200HS-CPU31-E/CPU33-E		For 1 Unit	C200H-CE001
				For 2 Units	C200H-CE002

Optional Products

Name	Specifications	Model number
I/O Unit Cover	Cover for 10-pin terminal block	C200H-COV11
Terminal Block Cover	Short protection for 10-pin terminal block	C200H-COV02
	Short protection for 19-pin terminal block	C200H-COV03
Connector Cover	Protective cover for unused I/O Connecting Cable connectors	C500-COV02
Space Unit	Used for vacant slots	C200H-SP001
Battery Set	For C200H RAM Memory Unit only	C200H-BAT09
Relay	24 VDC	G6B-1174P-FD-US DC24
Backplane Insulation Plate	For 10-slot Backplane	C200H-ATTA1
	For 8-slot Backplane	C200H-ATT81
	For 5-slot Backplane	C200H-ATT51
	For 3-slot Backplane	C200H-ATT31
I/O Bracket	For 10-slot Backplane	C200H-ATTA3
	For 8-slot Backplane	C200H-ATT83
	For 5-slot Backplane	C200H-ATT53
	For 3-slot Backplane	C200H-ATT33

Note 1. When ordering, specify the model name (any component of which is not sold separately).
2. Order the press-fit tool from the manufacturer.

Mounting Rails and Accessories

Name	Specifications	Model number
DIN Track Mounting Bracket	1 set (2 included)	C200H-DIN01
DIN Track	Length: 50 cm; height: 7.3 mm	PFP-50N
	Length: 1 m; height: 7.3 mm	PFP-100N
	Length: 1 m; height: 16 mm	PFP-100N2
End Plate	---	PFP-M
Spacer	---	PFP-S

Link Adapters

Name	Specifications	Model number
Link Adapter	3 RS-422 connectors	3G2A9-AL001
	3 optical connectors (APF/PCF)	3G2A9-AL002-PE
	3 optical connectors (PCF)	3G2A9-AL002-E
	1 connector each for APF/PCF, RS-422, and RS-232C	3G2A9-AL004-PE
	1 connector each for PCF, RS-422, and RS-232C	3G2A9-AL004-E
	1 connector each for APF/PCF and APF	3G2A9-AL005-PE
	1 connector each for PCF and AGF	3G2A9-AL005-E
	1 connector for APF/PCF; 2 for AGF	3G2A9-AL006-PE
	1 connector for PCF; 2 for AGF	3G2A9-AL006-E
	O/E converter; 1 connector for RS-485, 1 connector each for APF/PCF	B500-AL007-PE
	Used for on-line removal of SYSMAC NET Link Units from the SYSMAC NET Link System, SYSMAC NET Optical Link Adapter 3 connectors for APF/PCF.	B700-AL001

SYSMAC BUS/SYSMAC WAY Optical Fiber Products

Plastic Clad Optical Fiber Cable/All Plastic Optical Fiber Cable

Name	Specifications		Model number
Plastic Clad Optical Fiber Cable (indoor)	0.1 m, w/connectors	Ambient temp: –10° to 70°C	3G5A2-OF011
	1 m, w/connectors		3G5A2-OF101
	2 m, w/connectors		3G5A2-OF201
	3 m, w/connectors		3G5A2-OF301
	5 m, w/connectors		3G5A2-OF501
	10 m, w/connectors		3G5A2-OF111
	20 m, w/connectors		3G5A2-OF211
	30 m, w/connectors		3G5A2-OF311
	40 m, w/connectors		3G5A2-OF411
	50 m, w/connectors		3G5A2-OF511
	Cable only; order desired length between 1 and 200 m in increments of 1 m.		B500-OF002
All Plastic Optical Fiber Cable	Cable only; order desired length in 5 m increments between 5 and 100 m, or in increments of 200 m or 500 m.		B500-PF002
Optical Connectors A	Two optical connectors (brown) for APF (10 m max.)		3G5A2-CO001
Optical Connectors B	Two optical connectors (black) for APF (8 to 20 m)		3G5A2-CO002

Name	Specifications	Model number
All Plastic Optical Fiber Cable Set	1-m cable with an Optical Connector A connected to each end	3G5A2-PF101
Optical Fiber Processing Kit	Accessory: 125-mm nipper (Muromoto Tekko's 550M) for APF	3G2A9-TL101

H-PCF

Name	Specifications		Model number
Optical Fiber Cable SYSMAC BUS, SYSMAC WAY	10 m, black	Two-core cable	S3200-HCCB101
	50 m, black		S3200-HCCB501
	100 m, black		S3200-HCCB102
	500 m, black		S3200-HCCB502
	1000 m, black		S3200-HCCB103
	10 m, orange		S3200-HCCO101
	50 m, orange		S3200-HCCO501
	100 m, orange		S3200-HCCO102
	500 m, orange		S3200-HCCO502
	1000 m, orange		S3200-HCCO103
	10 m, black	Two-core cable	S3200-HBCB101
	50 m, black		S3200-HBCB501
	100 m, black		S3200-HBCB102
	500 m, black		S3200-HBCB502
	1000 m, black		S3200-HBCB103
Optical Fiber Cable Connector	SYSMAC BUS: C200H-RM001-PV1 C200H-RT001/RT002-P C500-RM001-(P)V1 C500-RT001/RT002-(P)V1 B500-□□□(-P)	Half-lock connector for Remote I/O Master, Remote I/O Slave, Host Link Unit, and Link Adapter	S3200-COCH82

Note 1. Optical fiber cables must be prepared and connected by specialists.

2. If the user prepares and connects optical fiber cables, the user must take a seminar held under the auspices of Sumitomo Electric Industries, Ltd. and obtain a proper certificate.

3. The Optical Power Tester, Head Unit, Master Fiber Set, and Optical Fiber Assembling Tool are required to connect optical fiber cables.

Optical Fiber Assembling Tool

Name	Specifications	Model number
Optical Fiber Assembling Tool	Used to connect H-PCF and crimp-cut connectors for optical transmission systems such as the SYSMAC C- and CV-series SYSMAC BUS, SYSMAC LINK and SYSMAC NET.	S3200-CAK1062

Note 1. Optical fiber cables must be prepared and connected by specialists.

2. The Optical Power Tester, Head Unit, Master Fiber set, and Optical Fiber Assembling Tool are required to connect optical fiber cables.

Optical Power Tester

Name	Specifications	Head Unit	Model number
Optical Power Tester (see note) (provided with a connector adapter, light source unit, small single-head plug, hard case, and AC adapter)	SYSMAC BUS: C200H-RM001-PV1 C200H-RT001/RT002-P C500-RM001-(P)V1 C500-RT001/RT002-(P)V1	S3200-CAT2822 (provided with the Tester)	S3200-CAT2820

Note: There is no difference between the light source unit and connector adapter for the Head Unit and those for the Optical Power Tester.

Head Unit

Name	Specifications	Model number
Head Unit (a set consisting of light source unit and connector adapter) (see note)	SYSMAC BUS: C200H-RM001-PV1 C200H-RT001/RT002-P	S3200-CAT2822
	SYSMAC NET: S3200-LSU03-V1/LSU03-01E C500-SNT31-V4 3G8FX-TM111 3G8SX-TM111	S3200-CAT3202

Note: Use a proper Head Unit model for the optical module to be used. If two types of optical modules (unit type and board type) are used, order an Optical Power Tester plus a proper Head Unit model.

Master Fiber Set

Name	Specifications	Model number
Master Fiber Set (1 m)	S3200-CAT3202 (SYSMAC NET, NSB, NSU, Bridge)	S3200-CAT3201
	S3200-CAT2002/CAT2702 (SYSMAC NET, SYSMAC LINK)	S3200-CAT2001H
	S3200-CAT2822 (SYSMAC BUS)	S3200-CAT2821

Note 1. The Master Fiber Set is used in combination with the Optical Power Tester to check the optical levels of optical fiber cables connected to optical fiber cable connectors.
2. Optical fiber cables must be prepared and connected by specialists.
3. The Optical Power Tester, Head Unit, Master Fiber set, and Optical Fiber Assembling Tool are required to connect optical fiber cables.

SYSMAC LINK/SYSMAC NET Link Optical Fiber Products

Optical Fiber Cables for SYSMAC LINK and SYSMAC NET Link Systems

Use hard-plastic-clad quartz optical fiber (H-PCF) cables for SYSMAC LINK and SYSMAC NET Link Systems. H-PCF cables are available with connectors already attached, or cables and connectors can be purchased separately and assembled by the user. Refer to the *System Manual* for the SYSMAC LINK or SYSMAC NET Link Systems for assembly procedures. Models numbers for H-PCF cables with connecters attached are provided in the following illustration.

Model Numbers for H-PCF Cables with Connectors

S3200-CN□□□-□□-□□

Cable length

201:	2 m
501:	5 m
102:	10 m
152:	15 m
202:	20 m
Blank:	Over 20 m (Specify.)

Connectors

20-25:	Full-lock connector on one end, half-lock connector on other end.
25-25:	Half-lock connectors on both ends.

An Optical Fiber Cable Bracket must be used to support an optical fiber cable connected to the C200HS-SNT32 SYSMAC NET Link Unit or C200HS-SLK12 SYSMAC LINK Unit.

User optical fiber cables with both tension members and power supply lines.

The following half-lock connector is used and connects to the C200HS SYSMAC LINK and SYSMAC NET Link Units: S3200-COCF2511.

The following full-lock connector is used and connects to the CV500 SYSMAC LINK and SYSMAC NET Link Units and the C1000H SYSMAC LINK Unit: S3200-COCF2011. This full-lock connector cannot be connected to the C200HS SYSMAC LINK and SYSMAC NET Link Units.

The above connectors cannot be used for the C500 SYSMAC NET Link Unit, cable relays, and the SYSMAC NET Link Network Service Board. Refer to the *SYSMAC NET Link System Manual* for further information.

Programming Devices

Name	Specifications		Model number
Programming Console	Hand-held, w/backlight; requires the C200H-CN222 or C200H-CN422, see below		C200H-PR027-E
	2-m Connecting Cable attached.		CQM1-PRO01-E
Programming Console Mounting Bracket	Used to attach Hand-held Programming Console to a panel.		C200H-ATT01
Programming Console Connecting Cables	For Hand-held Programming Console	2 m	C200H-CN222
		4 m	C200H-CN422
Data Setting Console	Used for data input and process value display for the C200H-TC□□□, C200H-TV□□□, C200H-CP114, and C200H-PID0□.		C200H-DSC01
Data Setting Console Connecting Cables	For C200H-DSC01	2 m	C200H-CN225
		4 m	C200H-CN425
Connecting Cable	Used to connect an IBM PC/AT or compatible to the C200HS.	3.3 m	CQM1-CIF02

Ladder Support Software (LSS)

Name	Specifications	Model number
Ladder Support Software (for C20, C□□P, C□□K, C120, C□□H, C200H, C200HS, C500, C1000H, C2000H, and CQM1)	5.25”, 2D for IBM PC/AT compatible	C500-SF711-EV3
	3.5”, 2HD for IBM PC/AT compatible	C500-SF312-EV3

SYSMAC Support Software (SSS)

Product	Description	Model no.
SYSMAC Support Software	3.5”, 2HD for IBM PC/AT compatible	C500-ZL3AT1-E

Training Materials

Name	Specifications	Model number
SYSMAC Training System	Includes text book, cassette tape, and input switch board.	C200H-ETL01-E
Fuzzy Training System	Includes a Fuzzy Training System Manual, a Main Unit, a C200H-MR831 Memory Unit, a C200H-PRO27-E Programming Console, a C200H-CN222 Cable for the Programming Console, C500-SU381-E Fuzzy Training Software, an RS-232C Cable, and a carrying belt.	C200H-ETL13-E

Appendix B
Programming Instructions

A PC instruction is input either by pressing the corresponding Programming Console key(s) (e.g., LD, AND, OR, NOT) or by using function codes. To input an instruction with its function code, press FUN, the function code, and then WRITE. Refer to the pages listed programming and instruction details.

Code	Mnemonic	Name	Function	Page
—	AND	AND	Logically ANDs status of designated bit with execution condition.	129
—	AND LD	AND LOAD	Logically ANDs results of preceding blocks.	130
—	AND NOT	AND NOT	Logically ANDs inverse of designated bit with execution condition.	129
—	CNT	COUNTER	A decrementing counter.	145
—	LD	LOAD	Used to start instruction line with the status of the designated bit or to define a logic block for use with AND LD and OR LD.	129
—	LD NOT	LOAD NOT	Used to start instruction line with inverse of designated bit.	129
—	OR	OR	Logically ORs status of designated bit with execution condition.	129
—	OR LD	OR LOAD	Logically ORs results of preceding blocks.	130
—	OR NOT	OR NOT	Logically ORs inverse of designated bit with execution condition.	129
—	OUT	OUTPUT	Turns ON operand bit for ON execution condition; turns OFF operand bit for OFF execution condition.	130
—	OUT NOT	OUTPUT NOT	Turns operand bit OFF for ON execution condition; turns operand bit ON for OFF execution condition (i.e., inverts operation).	130
—	RSET	RESET	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.	133
—	SET	SET	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.	133
—	TIM	TIMER	ON-delay (decrementing) timer operation.	139
00	NOP	NO OPERATION	Nothing is executed and program moves to next instruction.	138
01	END	END	Required at the end of the program.	138
02	IL	INTERLOCK	If interlock condition is OFF, all outputs are turned OFF and all timer PVs reset between this IL(02) and the next ILC(03). Other instructions are treated as NOP; counter PVs are maintained.	135
03	ILC	INTERLOCK CLEAR		135
04	JMP	JUMP	If jump condition is OFF, all instructions between JMP(04) and the corresponding JME(05) are ignored.	137
05	JME	JUMP END		137
(@)06	FAL	FAILURE ALARM AND RESET	Generates a non-fatal error and outputs the designated FAL number to the Programming Console.	275
07	FALS	SEVERE FAILURE ALARM	Generates a fatal error and outputs the designated FALS number to the Programming Console.	275
08	STEP	STEP DEFINE	When used with a control bit, defines the start of a new step and resets the previous step. When used without N, defines the end of step execution.	266
09	SNXT	STEP START	Used with a control bit to indicate the end of the step, reset the step, and start the next step.	266
10	SFT	SHIFT REGISTER	Creates a bit shift register.	150
11	KEEP	KEEP	Defines a bit as a latch controlled by set and reset inputs.	133
12	CNTR	REVERSIBLE COUNTER	Increases or decreases PV by one whenever the increment input or decrement input signals, respectively, go from OFF to ON.	148

Code	Mnemonic	Name	Function	Page
13	DIFU	DIFFERENTIATE UP	Turns ON the designated bit for one cycle on the rising edge of the input signal.	131
14	DIFD	DIFFERENTIATE DOWN	Turns ON the bit for one cycle on the trailing edge.	131
15	TIMH	HIGH-SPEED TIMER	A high-speed, ON-delay (decrementing) timer.	143
(@)16	WSFT	WORD SHIFT	Shifts data between starting and ending words in word units, writing zeros into starting word.	157
17 to 19	For expansion instructions.			
20	CMP	COMPARE	Compares the contents of two words and outputs result to GR, EQ, and LE Flags.	170
(@)21	MOV	MOVE	Copies source data (word or constant) to destination word.	159
(@)22	MVN	MOVE NOT	Inverts source data (word or constant) and then copies it to destination word.	159
(@)23	BIN	BCD TO BINARY	Converts four-digit, BCD data in source word into 16-bit binary data, and outputs converted data to result word.	180
(@)24	BCD	BINARY TO BCD	Converts binary data in source word into BCD, and outputs converted data to result word.	181
(@)25	ASL	ARITHMETIC SHIFT LEFT	Shifts each bit in single word of data one bit to left, with CY.	154
(@)26	ASR	ARITHMETIC SHIFT RIGHT	Shifts each bit in single word of data one bit to right, with CY.	154
(@)27	ROL	ROTATE LEFT	Rotates bits in single word of data one bit to left, with CY.	155
(@)28	ROR	ROTATE RIGHT	Rotates bits in single word of data one bit to right, with CY.	155
(@)29	COM	COMPLEMENT	Inverts bit status of one word of data.	249
(@)30	ADD	BCD ADD	Adds two four-digit BCD values and content of CY, and outputs result to specified result word.	205
(@)31	SUB	BCD SUBTRACT	Subtracts a four-digit BCD value and CY from another four-digit BCD value and outputs result to the result word.	207
(@)32	MUL	BCD MULTIPLY	Multiplies two four-digit BCD values and outputs result to specified result words.	211
(@)33	DIV	BCD DIVIDE	Divides four-digit BCD dividend by four-digit BCD divisor and outputs result to specified result words.	212
(@)34	ANDW	LOGICAL AND	Logically ANDs two 16-bit input words and sets corresponding bit in result word if corresponding bits in input words are both ON.	250
(@)35	ORW	LOGICAL OR	Logically ORs two 16-bit input words and sets corresponding bit in result word if one or both of corresponding bits in input data are ON.	251
(@)36	XORW	EXCLUSIVE OR	Exclusively ORs two 16-bit input words and sets bit in result word when corresponding bits in input words differ in status.	252
(@)37	XNRW	EXCLUSIVE NOR	Exclusively NORs two 16-bit input words and sets bit in result word when corresponding bits in input words are same in status.	253
(@)38	INC	BCD INCREMENT	Increments four-digit BCD word by one.	204
(@)39	DEC	BCD DECREMENT	Decrements four-digit BCD word by one.	204
(@)40	STC	SET CARRY	Sets carry flag (i.e., turns CY ON).	205
(@)41	CLC	CLEAR CARRY	Clears carry flag (i.e, turns CY OFF).	205
45	TRSM	TRACE MEMORY SAMPLE	Initiates data tracing.	277
(@)46	MSG	MESSAGE	Displays a 16-character message on the Programming Console display.	278
47 & 48	For expansion instructions.			
(@)50	ADB	BINARY ADD	Adds two four-digit hexadecimal values and content of CY, and outputs result to specified result word.	219
(@)51	SBB	BINARY SUBTRACT	Subtracts a four-digit hexadecimal value and CY from another four-digit hexadecimal value and outputs result to the result word.	221

Code	Mnemonic	Name	Function	Page
(@)52	MLB	BINARY MULTIPLY	Multiplies two four-digit hexadecimal values and outputs result to specified result words.	224
(@)53	DVB	BINARY DIVIDE	Divides four-digit hexadecimal dividend by four-digit hexadecimal divisor and outputs result to specified result words.	224
(@)54	ADDL	DOUBLE BCD ADD	Adds two eight-digit values (2 words each) and content of CY, and outputs result to specified result words.	206
(@)55	SUBL	DOUBLE BCD SUBTRACT	Subtracts an eight-digit BCD value and CY from another eight-digit BCD value and outputs result to the result words.	209
(@)56	MULL	DOUBLE BCD MULTIPLY	Multiplies two eight-digit BCD values and outputs result to specified result words.	212
(@)57	DIVL	DOUBLE BCD DIVIDE	Divides eight-digit BCD dividend by eight-digit BCD divisor and outputs result to specified result words.	213
(@)58	BINL	DOUBLE BCD TO DOUBLE BINARY	Converts BCD value in two consecutive source words into binary and outputs converted data to two consecutive result words.	181
(@)59	BCDL	DOUBLE BINARY TO DOUBLE BCD	Converts binary value in two consecutive source words into BCD and outputs converted data to two consecutive result words.	182
60 to 69	For expansion instructions.			
(@)70	XFER	BLOCK TRANSFER	Moves content of several consecutive source words to consecutive destination words.	161
(@)71	BSET	BLOCK SET	Copies content of one word or constant to several consecutive words.	160
(@)72	ROOT	SQUARE ROOT	Computes square root of eight-digit BCD value and outputs truncated four-digit integer result to specified result word.	217
(@)73	XCHG	DATA EXCHANGE	Exchanges contents of two different words.	162
(@)74	SLD	ONE DIGIT SHIFT LEFT	Left shifts data between starting and ending words by one digit (four bits).	156
(@)75	SRD	ONE DIGIT SHIFT RIGHT	Right shifts data between starting and ending words by one digit (four bits).	156
(@)76	MLPX	8-TO-256 DECODER	Converts up to four hexadecimal digits in source word into decimal values from 0 to 15 and turns ON, in result word(s), bit(s) whose position corresponds to converted value. Can also convert up to eight hexadecimal digits and turn ON corresponding bits in result words R to R+15.	185
(@)77	DMPX	256-TO-8 ENCODER	Determines position of highest ON bit in source word(s) and writes the ON bit's position (0 to F) to digit(s) in R. Can also determine the position of the highest ON bit in one or two groups of 16 words (S to S+15, S+16 to S+31) and writes the ON bit's position (00 to FF) to byte(s) in R.	188
(@)78	SDEC	7-SEGMENT DECODER	Converts hexadecimal values from source word to data for seven-segment display.	191
(@)79	FDIV	FLOATING POINT DIVIDE	Divides one floating point value (Dd+1, Dd) by another (Dr+1, Dr) and outputs the result to R+1 and R.	214
(@)80	DIST	SINGLE WORD DISTRIBUTE	Moves one word of source data to destination word whose address is given by destination base word plus offset.	162
(@)81	COLL	DATA COLLECT	Extracts data from source word and writes it to destination word.	164
(@)82	MOVB	MOVE BIT	Transfers designated bit of source word or constant to designated bit of destination word.	166
(@)83	MOVD	MOVE DIGIT	Moves hexadecimal content of specified four-bit source digit(s) to specified destination digit(s) for up to four digits.	167
(@)84	SFTR	REVERSIBLE SHIFT REGISTER	Shifts data in specified word or series of words to either left or right.	152
(@)85	TCMP	TABLE COMPARE	Compares four-digit hexadecimal value with values in table consisting of 16 words.	175

Code	Mnemonic	Name	Function	Page
(@)86	ASC	ASCII CONVERT	Converts hexadecimal values from the source word to eight-bit ASCII code starting at leftmost or rightmost half of starting destination word.	194
87 to 89	For expansion instructions.			
(@)90	SEND	NETWORK SEND	Used for communications with other PCs linked through the SYSMAC NET Link System or SYSMAC LINK System. (CPU31-E/33-E only)	291
(@)91	SBS	SUBROUTINE ENTRY	Calls and executes subroutine N.	257
92	SBN	SUBROUTINE DEFINE	Marks start of subroutine N.	259
93	RET	RETURN	Marks the end of a subroutine and returns control to main program.	259
(@)94	WDT	WATCHDOG TIMER REFRESH	Increases the watchdog timer PV by 0 to 6300 ms.	281
(@)97	IORF	I/O REFRESH	Refreshes all I/O words between the start and end words.	281
(@)98	RECV	NETWORK RECEIVE	Used for communications with other PCs linked through the SYSMAC NET Link System or SYSMAC LINK System. (CPU31-E/33-E only)	293
(@)99	MCRO	MACRO	Calls and executes a subroutine replacing I/O words.	260

Expansion Instructions

The following table shows the instructions that can be treated as expansion instructions. The default function codes are given for instructions that have codes assigned by default.

Code	Mnemonic	Name	Function	Page
17	(@)ASFT	ASYNCHRONOUS SHIFT REGISTER	Creates a shift register that exchanges the contents of adjacent words when one of the words is zero and the other is not.	157
18	(@)SCAN	CYCLE TIME	Sets the minimum cycle time (0 to 999.0 s).	276
19	(@)MCMP	MULTI-WORD COMPARE	Compares a block of 16 consecutive words to another block of 16 consecutive words.	169
47	(@)LMSG	32-CHARACTER MESSAGE	Outputs a 32-character message to the Programming Console.	279
48	(@)TERM	TERMINAL MODE	Switches the Programming Console to TERMINAL mode for the normal keyboard mapping operation.	280
60	CMPL	DOUBLE COMPARE	Compares two eight-digit hexadecimal values.	172
61	(@)MPRF	GROUP-2 HIGH-DENSITY I/O REFRESH	Refreshes I/O words allocated to Group-2 High-density I/O Units.	282
62	(@)XFRB	TRANSFER BITS	Copies the status of up to 255 specified source bits to the specified destination bits.	168
63	(@)LINE	COLUMN TO LINE	Copies a bit column from 16 consecutive words to the specified word.	200
64	(@)COLM	LINE TO COLUMN	Copies the 16 bits from the specified word to a bit column of 16 consecutive words.	201
65	(@)SEC	HOURS TO SECONDS	Converts hour and minute data to second data.	183
66	(@)HMS	SECONDS TO HOURS	Converts second data to hour and minute data.	184
67	(@)BCNT	BIT COUNTER	Counts the total number of bits that are ON in the specified block of words.	283
68	(@)BCMP	BLOCK COMPARE	Judges whether the value of a word is within 16 ranges (defined by lower and upper limits).	174
69	(@)APR	ARITHMETIC PROCESS	Performs sine, cosine, or linear approximation calculations.	239
87	TTIM	TOTALIZING TIMER	Creates a totalizing timer.	144
88	ZCP	AREA RANGE COMPARE	Compares a word to a range defined by lower and upper limits and outputs the result to the GR, EQ, and LE flags.	176

Code	Mnemonic	Name	Function	Page
89	(@)INT	INTERRUPT CONTROL	Performs interrupt control, such as masking and unmasking the interrupt bits for I/O interrupts.	262
---	7SEG	7-SEGMENT DISPLAY OUTPUT	Converts 4- or 8-digit BCD data to 7-segment display format and then outputs the converted data.	301
---	(@)ADBL	DOUBLE BINARY ADD	Adds two 8-digit binary values (normal or signed data) and outputs the result to R and R+1.	225
---	AVG	AVERAGE VALUE	Adds the specified number of hexadecimal words and computes the mean value. Rounds off to 4 digits past the decimal point.	235
---	CPS	SIGNED BINARY COMPARE	Compares two 16-bit (4-digit) signed binary values and outputs the result to the GR, EQ, and LE flags.	178
---	CPSL	DOUBLE SIGNED BINARY COMPARE	Compares two 32-bit (8-digit) signed binary values and outputs the result to the GR, EQ, and LE flags.	179
---	(@)DBS	SIGNED BINARY DIVIDE	Divides one 16-bit signed binary value by another and outputs the 32-bit signed binary result to R+1 and R.	231
---	(@)DBSL	DOUBLE SIGNED BINARY DIVIDE	Divides one 32-bit signed binary value by another and outputs the 64-bit signed binary result to R+3 to R.	232
---	DSW	DIGITAL SWITCH INPUT	Inputs 4- or 8-digit BCD data from a digital switch.	304
---	(@)FCS	FCS CALCULATE	Checks for errors in data transmitted by a Host Link command.	283
---	FPD	FAILURE POINT DETECT	Finds errors within an instruction block.	285
---	(@)HEX	ASCII-TO-HEXADECIMAL	Converts ASCII data to hexadecimal data.	195
---	(@)MAX	FIND MAXIMUM	Finds the maximum value in specified data area and outputs that value to another word.	233
---	(@)MBS	SIGNED BINARY MULTIPLY	Multiplies the signed binary content of two words and outputs the 8-digit signed binary result to R+1 and R.	229
---	(@)MBSL	DOUBLE SIGNED BINARY MULTIPLY	Multiplies two 32-bit (8-digit) signed binary values and outputs the 16-digit signed binary result to R+3 through R.	230
---	(@)MIN	FIND MINIMUM	Finds the minimum value in specified data area and outputs that value to another word.	234
---	MTR	MATRIX INPUT	Inputs data from an 8 input point × 8 output point matrix and records that data in D to D+3.	313
---	(@)NEG	2'S COMPLEMENT	Converts the four-digit hexadecimal content of the source word to its 2's complement and outputs the result to R.	202
---	(@)NEGL	DOUBLE 2'S COMPLEMENT	Converts the eight-digit hexadecimal content of the source words to its 2's complement and outputs the result to R and R+1.	203
---	(@)PID	PID CONTROL	PID control is performed according to the operand and PID parameters that are preset.	242
---	(@)RXD	RECEIVE	Receives data via a communications port.	297
---	(@)SBBL	DOUBLE BINARY SUBTRACT	Subtracts an 8-digit binary value (normal or signed data) from another and outputs the result to R and R+1.	227
---	(@)SCL	SCALING	Performs a scaling conversion on the calculated value.	198
---	(@)SRCH	DATA SEARCH	Searches the specified range of memory for the specified data. Outputs the word address(es) of words in the range that contain the data.	289
---	(@)SUM	SUM CALCULATE	Computes the sum of the contents of the words in the specified range of memory.	237
---	(@)TKY	TEN KEY INPUT	Inputs 8 digits of BCD data from a 10-key keypad.	311
---	(@)TXD	TRANSMIT	Sends data via a communications port.	299

Code	Mnemonic	Name	Function	Page
---	(@)XDMR	EXPANSION DM READ	The contents of the designated number of words of the fixed expansion DM data are read and output to the destination word on the PC side.	290
---	ZCPL	DOUBLE AREA RANGE COMPARE	Compares an 8-digit value to a range defined by lower and upper limits and outputs the result to the GR, EQ, and LE flags.	177

Appendix C
Error and Arithmetic Flag Operation

The following table shows the instructions that affect the ER, CY, GR, LE and EQ flags. In general, ER indicates that operand data is not within requirements. CY indicates arithmetic or data shift results. GT indicates that a compared value is larger than some standard, LT that it is smaller, and EQ, that it is the same. EQ also indicates a result of zero for arithmetic operations. Refer to *Section 5 Instruction Set* for details.

Vertical arrows in the table indicate the flags that are turned ON and OFF according to the result of the instruction.

Although ladder diagram instructions,TIM, and CNT are executed when ER is ON, other instructions with a vertical arrow under the ER 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, differentiated instructions affect flags in exactly the same way.

The ER, CY, GR, LE and EQ Flags are turned OFF when END(01) is executed, so their status cannot be monitored with a Programming Console.

Instructions	25503 (ER)	25504 (CY)	25505 (GR)	25506 (EQ)	25507 (LE)	Page
TIM	↕	Unaffected	Unaffected	Unaffected	Unaffected	139
CNT						145
END(01)[1]	OFF	OFF	OFF	OFF	OFF	138
STEP(08)	Unaffected	Unaffected	Unaffected	Unaffected	Unaffected	266
SNXT(09)						266
CNTR(12)	↕	Unaffected	Unaffected	Unaffected	Unaffected	148
TIMH(15)						143
WSFT(16)						157
CMP(20)	↕	Unaffected	↕	↕	↕	170
MOV(21)	↕	Unaffected	Unaffected	↕	Unaffected	159
MVN(22)						159
BIN(23)						180
BCD(24)						181
ASL(25)	↕	↕	Unaffected	↕	Unaffected	154
ASR(26)						154
ROL(27)						155
ROR(28)						155
COM(29)	↕	Unaffected	Unaffected	↕	Unaffected	249
ADD(30)	↕	↕	Unaffected	↕	Unaffected	205
SUB(31)						207
MUL(32)	↕	Unaffected	Unaffected	↕	Unaffected	211
DIV(33)						212
ANDW(34)						250
ORW(35)						251
XORW(36)						252
XNRW(37)						253
INC(38)						204
DEC(39)						204
STC(40)	Unaffected	ON	Unaffected	Unaffected	Unaffected	205
CLC(41)	Unaffected	OFF	Unaffected	Unaffected	Unaffected	205
MSG(46)	↕	Unaffected	Unaffected	Unaffected	Unaffected	278
ADB(50)[1]	↕	↕	Unaffected	↕	Unaffected	219
SBB(51)[1]						221

Instructions	25503 (ER)	25504 (CY)	25505 (GR)	25506 (EQ)	25507 (LE)	Page
MLB(52)	Unaffected	↕	Unaffected	Unaffected	↕	224
DVB(53)	↕	Unaffected	Unaffected	↕	Unaffected	224
ADDL(54)	↕	↕	Unaffected	↕	Unaffected	206
SUBL(55)						209
MULL(56)	↕	Unaffected	Unaffected	↕	Unaffected	212
DIVL(57)						213
BINL(58)						181
BCDL(59)						182
XFER(70)	↕	Unaffected	Unaffected	Unaffected	Unaffected	161
BSET(71)						160
ROOT(72)	↕	Unaffected	Unaffected	↕	Unaffected	217
XCHG(73)	↕	Unaffected	Unaffected	Unaffected	Unaffected	162
SLD(74)						156
SRD(75)						156
MLPX(76)						185
DMPX(77)						188
SDEC(78)						191
FDIV(79)	↕	Unaffected	Unaffected	↕	Unaffected	214
DIST(80)						162
COLL(81)						164
MOVB(82)	↕	Unaffected	Unaffected	Unaffected	Unaffected	166
MOVD(83)						167
SFTR(84)	↕	↕	Unaffected	Unaffected	Unaffected	152
TCMP(85)	↕	Unaffected	Unaffected	↕	Unaffected	175
ASC(86)	↕	Unaffected	Unaffected	Unaffected	Unaffected	194
SEND(90)						291
SBS(91)						257
RECV(98)						293
MCRO(99)						260

Note END(01), ADB(50), and SBB(51) also affect the signed binary arithmetic flags. Refer to page 451 for details.

Expansion Instructions

Instructions	25503 (ER)	25504 (CY)	25505 (GR)	25506 (EQ)	25507 (LE)	Page
7SEG(—)[2]	↕	Unaffected	Unaffected	Unaffected	Unaffected	301
ADBL(—)[2]	↕	↕	Unaffected	Unaffected	Unaffected	225
APR(69)	↕	Unaffected	Unaffected	↕	Unaffected	239
ASFT(17)	↕	Unaffected	Unaffected	Unaffected	Unaffected	157
AVG(—)						235
BCMP(68)	↕	Unaffected	Unaffected	Unaffected	Unaffected	174
BCNT(67)	↕	Unaffected	Unaffected	↕	Unaffected	283
CMPL(60)	↕	Unaffected	↕	↕	↕	172
COLM(64)	↕	Unaffected	Unaffected	↕	Unaffected	201
CPS(—)	↕	Unaffected	↕	↕	↕	178
CPSL(—)						179
DBS(—)	↕	Unaffected	Unaffected	↕	Unaffected	231
DBSL(—)						232
DSW(—)[3]	↕	Unaffected	Unaffected	Unaffected	Unaffected	304
FCS(—)						283

Instructions	25503 (ER)	25504 (CY)	25505 (GR)	25506 (EQ)	25507 (LE)	Page
FPD(—)	↕	↕	Unaffected	Unaffected	Unaffected	285
HEX(—)	↕	Unaffected	Unaffected	Unaffected	Unaffected	195
HMS(66)	↕	Unaffected	Unaffected	↕	Unaffected	184
INT(89)	↕	Unaffected	Unaffected	Unaffected	Unaffected	262
LINE(63)	↕	Unaffected	Unaffected	↕	Unaffected	200
LMSG(47)	↕	Unaffected	Unaffected	Unaffected	Unaffected	279
MAX(—)	↕	Unaffected	Unaffected	↕	Unaffected	233
MBS(—)						229
MBSL(—)						230
MCMP(19)						169
MIN(—)						234
MTR(—)[4]	↕	Unaffected	Unaffected	Unaffected	Unaffected	313
NEG(—)[2]	↕	Unaffected	Unaffected	↕	Unaffected	202
NEGL(—)[2]						203
PID(—)						242
RXD(—)	↕	Unaffected	Unaffected	Unaffected	Unaffected	297
SBBL(—)[2]	↕	↕	Unaffected	Unaffected	Unaffected	227
SCAN(18)	↕	Unaffected	Unaffected	Unaffected	Unaffected	276
SCL(—)	↕	Unaffected	Unaffected	↕	Unaffected	198
SEC(65)						183
SRCH(—)						289
SUM(—)						237
TKY(—)	↕	Unaffected	Unaffected	Unaffected	Unaffected	311
TTIM(87)						144
TXD(—)						299
XFRB(62)						168
XDMR(—)						290
ZCP(88)	↕	Unaffected	↕	↕	↕	176
ZCPL(—)						177

Note 1. SR 25409 will be ON when 7SEG(—) is being executed.

2. ADBL(—), NEG(—), NEGL(—), and SBBL(—) also affect the signed binary arithmetic flags. Refer to page 451 for details.

3. SR 25410 will be ON when DSW(—) is being executed.

4. SR 25408 will be ON when MTR(—) is being executed.

Signed Binary Arithmetic Flags

The following table shows the instructions that affect the OF and UF flags. In general, OF indicates that the result of a 16-bit calculation is greater than 32,767 (7FFF) or the result of a 32-bit calculation is greater than 2,147,483,647 (7FFF FFFF). UF indicates that the result of a 16-bit calculation is less than –32,768 (8000) or the result of a 32-bit calculation is less than –2,147,483,648 (8000 0000). Refer to *Section 5 Instruction Set* for details.

Vertical arrows in the table indicate the flags that are turned ON and OFF according to the result of the instruction. Instructions not shown do not affect any of the flags in the table.

The OF and UF Flags are turned OFF when END(01) is executed, so their status cannot be monitored with a Programming Console.

Instructions	SR 25404 (OF)	SR 25405 (UF)	Page
END(01)	OFF	OFF	138
ADB(50)	↕	↕	219
SBB(51)			221
ADBL(—)			225
SBBL(—)			227
NEG(—)	Unaffected	↕	202
NEGL(—)			203

These instructions also affect the ER, CY, and EQ Flags. Refer to the previous tables in this appendix for details.

Appendix D
Memory Areas

Overview

The following table shows the data areas in PC memory.

Area	Size	Range	Comments
I/O Area	480 bits	IR 000 to IR 029	
Group-2 High-density I/O Unit Area	320 bits	IR 030 to IR 049	Can be used as ordinary I/O or, if not used for real I/O, can be used as work bits.
SYSMAC BUS Area	800 bits	IR 050 to IR 099	Can be used as work bits if not used for real I/O.
Special I/O Unit Area	1,600 bits	IR 100 to IR 199	”
Optical I/O Unit Area	512 bits	IR 200 to IR 231	”
Work Area 1	64 bits	IR 232 to IR 235	
Special Relay Area 1	320 bits	SR 236 to SR 255	
Special Relay Area 2	704 bits	SR 256 to SR 299	
Macro Area	64 bits	SR 290 to SR 293	Inputs
	64 bits	SR 294 to SR 297	Outputs
Work Area 2	3,424 bits	IR 298 to IR 511	
Temporary Relay Area	8 bits	TR 00 to TR 07	
Holding Relay Area	1,600 bits	HR 00 to HR 99	
Auxiliary Relay Area	448 buts	AR 00 to AR 27	
Link Relay Area	1,024 bits	LR 00 to LR 63	
Timer/Counter Area	512 counters/ timers	TC 000 to TC 511	TIM 000 through TIM 015 can be refreshed via interrupt processing as high-speed timers.
Data Memory Area	6,144 words	DM 0000 to DM 6143	Read/Write
	1,000 words	DM 0000 to DM 1999	Special I/O Unit Area (see note)
	31 words	DM 6000 to DM 6030	History Log
	(44 words)	DM 6100 to DM 6143	Link test area (reserved)
	512 words	DM 6144 to DM 6655	Fixed DM Area (read only)
	456 words	DM 6144 to DM 6599	(SYSMAC NET Area)
	56 words	DM 6600 to DM 6655	PC Setup
Expansion DM Area	3,000 words max.	DM 7000 to DM 9999	Read only
I/O Comment Area	All UM Area max.	---	Stores I/O comments in the user program. Must be set using LSS.
Ladder Program Area	All UM Area max.	---	Stores the user program executed by the CPU. The program is retained even after the power supply is turned off.

Note The PC Setup can be set to use DM 7000 through DM 7999 as the Special I/O Area.

SR Area

Word(s)	Bit(s)	Function
236	00 to 07	Node loop status output area for operating level 0 of SYSMAC NET Link System
	08 to 15	Node loop status output area for operating level 1 of SYSMAC NET Link System
237	00 to 07	Completion code output area for operating level 0 following execution of SEND(90)/RECV(98) SYSMAC LINK/SYSMAC NET Link System
	08 to 15	Completion code output area for operating level 1 following execution of SEND(90)/RECV(98) SYSMAC LINK/SYSMAC NET Link System
238 and 241	00 to 15	Data link status output area for operating level 0 of SYSMAC LINK or SYSMAC NET Link System
242 and 245	00 to 15	Data link status output area for operating level 1 of SYSMAC LINK or SYSMAC NET Link System
246	00 to 15	Reserved by system
247 and 248	00 to 07	PC Link Unit Run Flags for Units 16 through 31 or data link status for operating level 1
	08 to 15	PC Link Unit Error Flags for Units 16 through 31 or data link status for operating level 1
249 and 250	00 to 07	PC Link Unit Run Flags for Units 00 through 15 or data link status for operating level 0
	08 to 15	PC Link Unit Error Flags for Units 00 through 15 or data link status for operating level 0
251 Writeable	00	Remote I/O Error Read Bit
	01 and 02	Not used
	03	Remote I/O Error Flag
	04 to 06	Unit number of Remote I/O Unit or Optical I/O Unit with error
	07	Not used
	08 to 15	Word allocated to Remote I/O Unit or Optical I/O Unit with error
252	00	SEND(90)/RECV(98) Error Flag for operating level 0 of SYSMAC LINK or SYSMAC NET Link System
	01	SEND(90)/RECV(98) Enable Flag for operating level 0 of SYSMAC LINK or SYSMAC NET Link System
	02	Operating Level 0 Data Link Operating Flag
	03	SEND(90)/RECV(98) Error Flag for operating level 1 of SYSMAC LINK or SYSMAC NET Link System
	04	SEND(90)/RECV(98) Enable Flag for operating level 1 of SYSMAC LINK or SYSMAC NET Link System
	05	Operating Level 1 Data Link Operating Flag
	06	Rack-mounting Host Link Unit Level 1 Communications Error Flag
	07	Rack-mounting Host Link Unit Level 1 Restart Bit
	08	Peripheral Port Restart Bit
	09	RS-232C Port Restart Bit
	10	PC Setup Clear Bit
	11	Forced Status Hold Bit
	12	Data Retention Control Bit
	13	Rack-mounting Host Link Unit Level 0 Restart Bit
	14	Not used.
	15	Output OFF Bit
253	00 to 07	FAL number output area (see error information provided elsewhere)
	08	Low Battery Flag
	09	Cycle Time Error Flag
	10	I/O Verification Error Flag
	11	Rack-mounting Host Link Unit Level 0 Communications Error Flag
	12	Remote I/O Error Flag
	13	Always ON Flag
	14	Always OFF Flag
	15	First Cycle Flag

Word(s)	Bit(s)	Function
254	00	1-minute clock pulse bit
	01	0.02-second clock pulse bit
	02 and 03	Reserved for function expansion. Do not use.
	04	Overflow Flag (for signed binary calculations)
	05	Underflow Flag (for signed binary calculations)
	06	Differential Monitor End Flag
	07	Step Flag
	08	MTR Execution Flag
	09	7SEG Execution Flag
	10	DSW Execution Flag
	11	Interrupt Input Unit Error Flag
	12	Reserved by system
	13	Interrupt Programming Error Flag
	14	Group-2 High-density I/O Unit Error Flag
	15	Special Unit Error Flag (Special I/O, PC Link, Host Link, Remote I/O Master)
255	00	0.1-second clock pulse bit
	01	0.2-second clock pulse bit
	02	1.0-second clock pulse bit
	03	Instruction Execution Error (ER) Flag
	04	Carry (CY) Flag
	05	Greater Than (GR) Flag
	06	Equals (EQ) Flag
	07	Less Than (LE) Flag
	08 to 15	Reserved by system (used for TR bits)
256 to 261	00 to 15	Reserved by system
262	00 to 15	Longest interrupt subroutine (action) execution time (0.1 ms)
263	00 to 15	Number of interrupt subroutine (action) with longest execution time. (8000 to 8512) 8000 to 8007, 8099 Bit 15: Interrupt Flag
264	00 to 03	RS-232C Port Error Code 0: No error 1: Parity error 2: Framing error 3: Overrun error
	04	RS-232C Port Communications Error
	05	RS-232C Port Send Ready Flag
	06	RS-232C Port Reception Completed Flag
	07	RS-232C Port Reception Overflow Flag
	08 to 11	Peripheral Port Error Code in General I/O Mode 0: No error 1: Parity error 2: Framing error 3: Overrun error F: When in Peripheral Bus Mode
	12	Peripheral Port Communications Error in General I/O Mode
	13	Peripheral Port Send Ready Flag in General I/O Mode
	14	Peripheral Port Reception Completed Flag in General I/O Mode
	15	Peripheral Port Reception Overflow Flag in General I/O Mode
265	00 to 15	RS-232C Port Reception Counter in General I/O Mode
266	00 to 15	Peripheral Reception Counter in General I/O Mode (BCD)

Word(s)	Bit(s)	Function	
267	00 to 04	Reserved by system (not accessible by user)	
	05	Host Link Level 0 Send Ready Flag	
	06 to 12	Reserved by system (not accessible by user)	
	13	Host Link Level 1 Send Ready Flag	
	14 and 15	Reserved by system (not accessible by user)	
268	00 to 15	Reserved by system (not accessible by user)	
269	00 to 07	Memory Cassette Contents 00: Nothing; 01: UM; 02: IOM (03: HIS)	
	08 to 10	Memory Cassette Capacity 0: 0 KW (no cassette); 3: 16 KW	
	11 to 13	Reserved by system (not accessible by user)	
	14	EEPROM Memory Cassette Protected or EPROM Memory Cassette Mounted Flag	
	15	Memory Cassette Flag	
270	00	Save UM to Cassette Bit	Data transferred to Memory Cassette when Bit is turned ON in PROGRAM mode. Bit will automatically turn OFF. An error will be produced if turned ON in any other mode.
	01	Load UM from Cassette Bit	
	02	Compare UM to Cassette Bit	
	03	Comparison Results 0: Contents identical; 1: Contents differ or comparison not possible	
	04 to 10	Reserved by system (not accessible by user)	
	11	Transfer Error Flag: Transferring SYSMAC NET data link table on UM during active data link.	Data will not be transferred from UM to the Memory Cassette if an error occurs (except for Board Checksum Error). Detailed information on checksum errors occurring in the Memory Cassette will not be output to SR 272 because the information is not needed. Repeat the transmission if SR 27015 is ON.
	12	Transfer Error Flag: Not PROGRAM mode	
	13	Transfer Error Flag: Read Only	
	14	Transfer Error Flag: Insufficient Capacity or No UM	
	15	Transfer Error Flag: Board Checksum Error	
271	00 to 07	Ladder program size stored in Memory Cassette Ladder-only File: 04: 4 KW; 08: 8 KW; 12: 12 KW; ... (64: 64 KW) 00: No ladder program or no file Data updated at data transfer from CPU at startup. The file must begin in segment 0.	
	08 to 15	Ladder program size and type in CPU (Specifications are the same as for bits 00 to 07.) Data updated when indexes generated. Default value (after clearing memory) is 16.	
272	00 to 10	Reserved by system (not accessible by user)	
	11	Memory Error Flag: PC Setup Checksum Error	
	12	Memory Error Flag: Ladder Checksum Error	
	13	Memory Error Flag: Instruction Change Vector Area Checksum Error	
	14	Memory Error Flag: Memory Cassette Online Disconnection	
	15	Memory Error Flag: Autoboot Error	

Word(s)	Bit(s)	Function	
273	00	Save IOM to Cassette Bit	Data transferred to Memory Cassette when Bit is turned ON in PROGRAM mode. Bit will automatically turn OFF. An error will be produced if turned ON in any other mode.
	01	Load IOM from Cassette Bit	
	02 to 11	Reserved by system (not accessible by user)	
	12	Transfer Error Flag: Not PROGRAM mode	Data will not be transferred from IOM to the Memory Cassette if an error occurs (except for Read Only Error).
	13	Transfer Error Flag: Read Only	
	14	Transfer Error Flag: Insufficient Capacity or No IOM	
	15	Transfer Error Flag: Checksum Error	
274	00	Special I/O Unit #0 Restart Flag	These flags will turn ON during restart processing. These flags will not turn ON for Units on Slave Racks.
	01	Special I/O Unit #1 Restart Flag	
	02	Special I/O Unit #2 Restart Flag	
	03	Special I/O Unit #3 Restart Flag	
	04	Special I/O Unit #4 Restart Flag	
	05	Special I/O Unit #5 Restart Flag	
	06	Special I/O Unit #6 Restart Flag	
	07	Special I/O Unit #7 Restart Flag	
	08	Special I/O Unit #8 Restart Flag	
	09	Special I/O Unit #9 Restart Flag	
	10 to 15	Reserved by system (not accessible by user)	
275	00	PC Setup Startup Error (DM 6600 to DM 6614)	
	01	PC Setup RUN Error (DM 6615 to DM 6644)	
	02	PC Setup Communications/Error Setting/Misc. Error (DM 6645 to DM 6655)	
	03 to 15	Reserved by system (not accessible by user)	
276	00 to 07	Minutes (00 to 59)	Used for time increments.
	08 to 15	Hours (00 to 23)	
277 to 279	00 to 15	Used for keyboard mapping. See page 368.	
280 to 289	00 to 15	Reserved by system (not accessible by user)	
290 to 293	00 to 15	Macro Area inputs.	
294 to 297	00 to 15	Macro Area outputs.	
298 to 299	00 to 15	Reserved by system (not accessible by user)	

AR Area

Word(s)	Bit(s)	Function
00	00 to 09	Error Flags for Special I/O Units 0 to 9 (also function as Error Flags for PC Link Units)
	10	Error Flag for operating level 1 of SYSMAC LINK or SYSMAC NET Link System
	11	Error Flag for operating level 0 of SYSMAC LINK or SYSMAC NET Link System
	12	Host Computer to Rack-mounting Host Link Unit Level 1 Error Flag
	13	Host Computer to Rack-mounting Host Link Unit Level 0 Error Flag
	14	Remote I/O Master Unit 1 Error Flag
	15	Remote I/O Master Unit 0 Error Flag
01	00 to 09	Restart Bits for Special I/O Units 0 to 9 (also function as Restart Bits for PC Link Units)
	10	Restart Bit for operating level 1 of SYSMAC LINK or SYSMAC NET Link System
	11	Restart Bit for operating level 0 of SYSMAC LINK or SYSMAC NET Link System
	12, 13	Not used.
	14	Remote I/O Master Unit 1 Restart Flag.
	15	Remote I/O Master Unit 0 Restart Flag.
02	00 to 04	Slave Rack Error Flags (#0 to #4)
	05 to 14	Group-2 High-density I/O Unit Error Flags
	15	Group-2 High-density I/O Unit Error Flag
03	00 to 15	Error Flags for Optical I/O Units 0 to 7
04	00 to 15	Error Flags for Optical I/O Units 8 to 15
05	00 to 15	Error Flags for Optical I/O Units 16 to 23
06	00 to 15	Error Flags for Optical I/O Units 24 to 31
07	00 to 03	Data Link setting for operating level 0 of SYSMAC LINK System
	04 to 07	Data Link setting for operating level 1 of SYSMAC LINK System
	08	Normal TERMINAL Mode/Expansion TERMINAL Mode Input Cancel Bit
	09	Expansion TERMINAL Mode Changeover Flag
	10 and 11	Reserved by system.
	12	Terminal Mode Flag ON: Expansion; OFF: Normal (Same as status of pin 6 on CPU's DIP switch)
	13	Error History Overwrite Bit
	14	Error History Reset Bit
	15	Error History Enable Bit
08 to 11	00 to 15	Active Node Flags for SYSMAC LINK System nodes of operating level 0
12 to 15	00 to 15	Active Node Flags for SYSMAC LINK System nodes of operating level 1
16	00 to 15	SYSMAC LINK/SYSMAC NET Link System operating level 0 service time per cycle
17	00 to 15	SYSMAC LINK/SYSMAC NET Link System operating level 1 service time per cycle
18 Writeable	00 to 07	Seconds: 00 to 99
	08 to 15	Minutes: 00 to 59
19 Writeable	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)
20 Writeable	00 to 07	Month: 1 to 12
	08 to 15	Year: 00 to 99 (Rightmost two digits of year)
21 Writeable	00 to 07	Day of Week: 00 to 06 (00: Sunday; 01: Monday; 02: Tuesday; 03: Wednesday; 04: Thursday; 05: Friday; 06: Saturday)
	08 to 12	Not used.
	13	30-second Compensation Bit
	14	Clock Stop Bit
	15	Clock Set Bit
22	00 to 15	Keyboard Mapping

Word(s)	Bit(s)	Function
23	00 to 15	Power Off Counter (BCD)
24	00 to 04	Reserved by system.
	05	Cycle Time Flag
	06	SYSMAC LINK System Network Parameter Flag for operating level 1
	07	SYSMAC LINK System Network Parameter Flag for operating level 0
	08	SYSMAC/SYSMAC NET Link Unit Level 1 Mounted Flag
	09	SYSMAC/SYSMAC NET Link Unit Level 0 Mounted Flag
	10	Reserved by system.
	11 and 12	PC Link Level
	13	Rack-mounting Host Link Unit Level 1 Mounted Flag
	14	Rack-mounting Host Link Unit Level 0 Mounted Flag
	15	CPU-mounting Device Mounted Flag
25	00 to 11	Reserved by system.
	12	Trace End Flag
	13	Tracing Flag
	14	Trace Trigger Bit (writeable)
	15	Trace Start Bit (writeable)
26	00 to 15	Maximum Cycle Time (0.1 ms)
27	00 to 15	Present Cycle Time (0.1 ms)

DM Area (Error Log)

Word	Function
DM 6000	Error Pointer (Incremented by 1 for each error (0000 to 000A)
DM 6001 to DM 6030	Error Records (3 words per error for 10 errors, 30 words total) See following table for structure of individual records.

Word	Bit	Content	
First	00 to 07	Error code (see error information provided elsewhere)	
	08 to 15	00 (non-fatal) or 80 (fatal)	
Second	00 to 07	Seconds	Clock data read from AR 18 and AR 19.
	08 to 15	Minutes	
Third	00 to 07	Hours	
	08 to 15	Day of month	

Appendix E
PC Setup

Word(s)	Bit(s)	Function	Default
Startup Processing (DM 6600 to DM 6614)			
The following settings are effective after transfer to the PC only after the PC is restarted.			
DM 6600	00 to 07	Startup mode (effective when bits 08 to 15 are set to 02). 00: PROGRAM; 01: MONITOR 02: RUN	PROGRAM
	08 to 15	Startup mode designation 00: Programming Console switch 01: Continue operating mode last used before power was turned off 02: Setting in 00 to 07	Programming Console switch
DM 6601	00 to 07	Reserved	---
	08 to 11	IOM Hold Bit (SR 25212) Status 0: Reset; 1: Maintain	Reset
	12 to 15	Forced Status Hold Bit (SR 25211) Status 0: Reset; 1: Maintain	
DM 6602	00 to 07	Reserved	---
	08 to 15	Special I/O Unit Area 00: Use DM 1000 to DM 1999 01: Transfer DM 7000 through DM 7999 to DM 1000 through DM 1999 at startup and use DM 1000 to DM 1999. 02: Use DM 7000 to DM 7999	DM 1000 to DM 1999
DM 6603 to DM 6614	00 to 15	Reserved	---
Cycle Time Settings (DM 6615 to DM 6619)			
The following settings are effective after transfer to the PC the next time operation is started.			
DM 6615	00 to 15	Reserved	---
DM 6616	00 to 07	(Servicing time for RS-232C port (effective when bits 08 to 15 are set to 01)) 00 to 99 (BCD): Percentage of cycle time used to service RS-232C port. Minimum: 0.256 ms; maximum 65.536 ms	No setting (00)
	08 to 15	(RS-232C port servicing setting enable) 00: Do not set service time 01: Use time in 00 to 07. Service time is 10 ms when operation is stopped.	
DM 6617	00 to 07	Servicing time for peripheral port (effective when bits 08 to 15 are set to 01) 00 to 99 (BCD): Percentage of cycle time used to service peripheral. Minimum: 0.256 ms; maximum 65.536 ms	No setting (00)
	08 to 15	Peripheral port servicing setting enable 00: Do not set service time 01: Use time in 00 to 07. Service time is 10 ms when operation is stopped.	
DM 6618	00 to 07	Cycle monitor time (effective when bits 08 to 15 are set to 01, 02, or 03) 00 to 99 (BCD): Setting (see 08 to 15)	00
	08 to 15	Cycle monitor enable (Setting in 00 to 07 x unit; 99 s max.) 00: 120 ms (setting in bits 00 to 07 disabled) 01: Setting unit: 10 ms 02: Setting unit: 100 ms 03: Setting unit: 1 s	00: 120 ms
DM 6619	00 to 15	Cycle time 0000: Variable (no minimum) 0001 to 9999 (BCD): Minimum time in ms	Variable

Word(s)	Bit(s)	Function	Default
Interrupt/Refresh Processing (DM 6620 to DM 6622)			
The following settings are effective after transfer to the PC the next time operation is started.			
DM 6620	00 to 09	Special I/O Unit cyclic refresh (Bit number corresponds to unit number, PC Link Units included) 0: Enable cyclic refresh and I/O REFRESH (IORF(97)) from main program 1: Disable (refresh only for I/O REFRESH from interrupt programs) Disable not valid for normal (C200H) interrupt response or on Slave Racks.	Enable
	10 to 11	Reserved	---
	12 to 15	Interrupt response 0: Normal (C200H compatible) 1: High-speed (C200HS)	Normal
DM 6621	00 to 07	Reserved	---
	08 to 15	Special I/O Unit refresh (PC Link Units included) 00: Enable refresh for all Special I/O Units 01: Disable refresh for all Special I/O Units (but, not valid on Slave Racks)	Enable
DM 6622	00 to 07	Scheduled interrupt time unit 00: 10 ms 01: 1 ms	10 ms
	08 to 15	Scheduled interrupt time unit enable 00: Disable (10 ms) 01: Enable	
DM 6623 to DM 6644	00 to 15	Reserved	---
RS-232C Port Settings (DM 6645 to DM 6649)			
The following settings are effective after transfer to the PC.			
DM 6645	00 to 07	Port settings 00: Standard (1 start bit, 7-bit data, even parity, 2 stop bits, 9,600 bps) 01: Settings in DM 6646	Standard
	08 to 11	Words linked for 1:1 link 0: LR 00 to LR 63; 1: LR 00 to LR 31; 2: LR 00 to LR 15	LR 00 to LR 63
	12 to 15	Communications mode 0: Host link; 1: RS-232C; 2: 1-to-1 link slave; 3: 1-to-1 link master; 4: NT (PT) link	Host Link
DM 6646	00 to 07	Baud rate 00: 1.2K, 01: 2.4K, 02: 4.8K, 03: 9.6K, 04: 19.2K	1.2 K
	08 to 15	Frame format Start / Length / Stop / Parity 00: 1 bit / 7 bits / 1 bit / Even 01: 1 bit / 7 bits / 1 bit / Odd 02: 1 bit / 7 bits / 1 bit / None 03: 1 bit / 7 bits / 2 bit / Even 04: 1 bit / 7 bits / 2 bit / Odd 05: 1 bit / 7 bits / 2 bit / None 06: 1 bit / 8 bits / 1 bit / Even 07: 1 bit / 8 bits / 1 bit / Odd 08: 1 bit / 8 bits / 1 bit / None 09: 1 bit / 8 bits / 2 bit / Even 10: 1 bit / 8 bits / 2 bit / Odd 11: 1 bit / 8 bits / 2 bit / None	1 start bit, 7-bit data, 1 stop bit, even parity
DM 6647	00 to 15	Transmission delay 0000 to 9999: BCD in ms.	0 ms

Word(s)	Bit(s)	Function	Default
DM 6648	00 to 07	Node number (Host link) 00 to 31 (BCD)	0
	08 to 11	Start code enable (RS-232C) 0: Disable; 1: Set	Disabled
	12 to 15	End code enable (RS-232C) 0: Disable (number of bytes received) 1: Set (specified end code) 2: CR, LF	Disabled
DM 6649	00 to 07	Start code (RS-232C) 00 to FF (binary)	Not used
	08 to 15	12 to 15 of DM 6648 set to 0: Number of bytes received 00: Default setting (256 bytes) 01 to FF: 1 to 255 bytes 12 to 15 of DM 6648 set to 1: End code (RS-232C) 00 to FF (binary)	
Peripheral Port Settings (DM 6650 to DM 6654) The following settings are effective after transfer to the PC.			
DM 6650	00 to 07	Port settings 00: Standard (1 start bit, 7-bit data, even parity, 2 stop bits, 9,600 bps) 01: Settings in DM 6651	Standard
	08 to 11	Reserved	---
	12 to 15	Communications mode 0: Host link; 1: RS-232C	Host Link
DM 6651	00 to 07	Baud rate 00: 1.2K, 01: 2.4K, 02: 4.8K, 03: 9.6K, 04: 19.2K	1.2 K
	08 to 15	Frame format Start / Length / Stop / Parity 00: 1 bit / 7 bits / 1 bit / Even 01: 1 bit / 7 bits / 1 bit / Odd 02: 1 bit / 7 bits / 1 bit / None 03: 1 bit / 7 bits / 2 bit / Even 04: 1 bit / 7 bits / 2 bit / Odd 05: 1 bit / 7 bits / 2 bit / None 06: 1 bit / 8 bits / 1 bit / Even 07: 1 bit / 8 bits / 1 bit / Odd 08: 1 bit / 8 bits / 1 bit / None 09: 1 bit / 8 bits / 2 bit / Even 10: 1 bit / 8 bits / 2 bit / Odd 11: 1 bit / 8 bits / 2 bit / None	1 start bit, 7-bit data, 1 stop bit, even parity
DM 6652	00 to 15	Transmission delay (Host Link) 0000 to 9999: In ms.	0 ms
DM 6653	00 to 07	Node number (Host link) 00 to 31 (BCD)	0
	08 to 11	Start code enable (RS-232C) 0: Disable; 1: Set	Disable
	12 to 15	End code enable (RS-232C) 0: Disable (number of bytes received) 1: Set (specified end code) 2: CR, LF	Disable

Word(s)	Bit(s)	Function	Default
DM 6654	00 to 07	Start code (RS-232C) 00 to FF (binary)	0000
	08 to 15	12 to 15 of DM 6653 set to 0: Number of bytes received 00: Default setting (256 bytes) 01 to FF: 1 to 255 bytes 12 to 15 of DM 6653 set to 1: End code (RS-232C) 00 to FF (binary)	
Error Settings (DM 6655) The following settings are effective after transfer to the PC.			
DM 6655	00 to 03	Interrupt programming error enable 0: Detect interrupt programming errors 1: Do not detect	Detect
	04 to 07	(Multiple action execution error enable 0: Detect multiple action execution 1: Do not detect)	Detect
	08 to 11	Cycle time monitor enable 0: Detect long cycles as non-fatal errors 1: Do not detect long cycles	Detect
	12 to 15	Low battery error enable 0: Detect low battery voltage as non-fatal error 1: Do not detect low batter voltage	Detect

Appendix F
Word Assignment Recording Sheets

This appendix contains sheets that can be copied by the programmer to record I/O bit allocations and terminal assignments, as well as details of work bits, data storage areas, timers, and counters.

I/O Bits

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		

Work Bits

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

Timers and Counters

Programmer: **Program:** **Date:** **Page:**

TC address	T or C	Set value	Notes

TC address	T or C	Set value	Notes

Appendix G
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.

Program Coding Sheet

Programmer: **Program:** **Date:** **Page:**

Address	Instruction	Operand(s)	Address	Instruction	Operand(s)	Address	Instruction	Operand(s)

Appendix H
Data Conversion Tables

Normal Data

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

Signed Binary Data

Decimal	16-bit Hex	32-bit Hex
2147483647	——	7FFFFFFF
2147483646	——	7FFFFFFE
.	.	.
.	.	.
.	.	.
32768	——	00008000
32767	7FFF	00007FFF
32766	7FFE	00007FFE
.	.	.
.	.	.
.	.	.
5	0005	00000005
4	0004	00000004
3	0003	00000003
2	0002	00000002
1	0001	00000001
0	0000	00000000
–1	FFFF	FFFFFFFF
–2	FFFE	FFFFFFFE
–3	FFFD	FFFFFFFD
–4	FFFC	FFFFFFFC
–5	FFFB	FFFFFFFB
.	.	.
.	.	.
.	.	.
–32767	8001	FFFF8001
–32768	8000	FFFF8000
–32769	——	FFFF7FFF
.	.	.
.	.	.
.	.	.
–2147483647	——	80000001
–2147483648	——	80000000

Appendix I
Extended ASCII

Programming Console Displays

Bits 0 to 3		Bits 4 to 7													
BIN		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

address The location in memory where data is stored. For data areas, an address consists of a two-letter data area designation and a number that designates the word and/or bit location. For the UM area, an address designates the instruction location (UM area). In the FM area, the address designates the block location, etc.

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.

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.

APF Acronym for all plastic fiber-optic cable.

AR area A PC data area allocated to flags, control bits, and work bits.

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.

ASCII Unit An Intelligent I/O Unit used to program in BASIC. When connected to an NSU on a Net Link System, commands can be sent to other nodes.

Backplane A base onto which Units are mounted to form a Rack. Backplanes provide a series of connectors for these Units along with wiring to connect them to the CPU. Backplanes also provide connectors used to connect them to other Backplanes. In some Systems, different Backplanes are used for different Racks; in other Systems, Racks differ only according to the Units mounted to them.

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 to the base 2, i.e., any number can be written using only 1's or 2's. Each group of four binary bits is equivalent to one hexadecimal digit.

binary calculation An arithmetic calculation that uses numbers expressed in binary.

binary-coded decimal A system used to represent numbers so that each group of four binary bits is numerically equivalent to one decimal digit.

bit A binary digit; hence a unit of data in binary notation. The smallest unit of information that can be electronically stored in a PC. The status of a bit is either ON or OFF. Different bits at particular addresses are allocated to special purposes, such as holding the status 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 must specify (sometimes by default) the data area and word that is being addressed, as well as the number of the bit.

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.

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 components.

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.

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.

clock pulse A pulse available at a certain bit in memory for use in timing operations. Various clock pulses are available with different pulse widths.

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.

common data Data that is stored in the LR Area of a PC and which is shared by other PCs in the same the same system. Each PC has a specified section of the LR Area allocated to it. This allocation is the same in each LR Area of each PC.

condition An message placed in an instruction line to direct the way in which the terminal instructions, on the right side, are to be executed. Each condition is assigned to a bit in memory that determines its status. The status of the bit assigned to each condition determines, in turn, the execution condition for each instruction up to a terminal instruction on the right side of the ladder diagram.

constant An operand for which the actual numeric value is specified by the user, and which is then stored in a particular address in the data memory.

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 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.

control signal A signal sent from the PC to effect the operation of the controlled system.

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	An acronym for central processing unit. In a PC System, the CPU executes the program, processes I/O signals, communicates with external devices, etc.
CPU Backplane	A Backplane which is used to create a CPU Rack.
CPU Rack	Part of a building-block PC, the CPU Rack contains the CPU, a power supply, and other Units. With most PCs, the CPU Rack is the only Rack that provides linkable slots.
CTS	An acronym for clear-to-send, a signal used in communications between electronic devices to indicate that the receiver is ready to accept incoming data.
cycle	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.
cycle time	The time required for a single cycle of the ladder-diagram program.
data area	An area in the PC's memory that is designed to hold a specific type of data, e.g., the LR area is designed to hold common data in a PC Link System. Memory areas that hold programs are not considered data areas.
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 sharing	An aspect of PC Link Systems and of Data Links in Net Link Systems in which common data areas or common data words are created between two or more PCs.
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.
decimal	A number system where all 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.
default	A value automatically set by the PC when the user omits to set a specific 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.
delay	In tracing, a value that specifies where tracing is to begin in relationship to the trigger. A delay can be either positive or negative, i.e., can designate an offset on either side of the trigger.
destination	The location where an instruction is to place 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. Nondifferentiated instructions are executed each cycle 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 cycle 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.

distributed control An 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 one of the fundamental concepts of PC Systems.

DM area A data area used to hold only word data. Words in the DM area cannot be accessed bit by bit.

download The process of transferring a program or data from a higher-level computer to a lower-level computer or PC.

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.

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.

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.

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.

exection 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 time The time required for the CPU to execute either an individual instruction or an entire program.

Expansion I/O Backplane A Backplane which is used to create an Expansion I/O Rack.

Expansion I/O Rack Part of a building-block PC, an Expansion I/O Rack is connected to either a CPU Rack or another Expansion I/O Rack to increase the number of slots available for mounting 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.

Factory Intelligent Terminal A programming device provided with advanced programming and debugging capabilities to facilitate PC operation. The Factory Intelligent Terminal also provides various interfaces for external devices, such as floppy disk drives.

fatal error An error that stops PC operation and requires correction before operation can continue.

FIT Abbreviation for Factory Intelligent Terminal.

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 between 0 and 1 (the mantissa) multiplied by a power of 10, e.g., 0.538×10^{-5}.

Floppy Disk Interface Unit A Unit used to interface a floppy disk drive to a PC so that programs and/or data can be stored on floppy disks.

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.

function code A two-digit number used to input an instruction into the PC.

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.

Host Link System A system with one or more host computers connected to one or more PCs via Host Link Units 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 PC to a host computer in a Host Link System.

host computer A computer that is used to transfer data or programs to from a PC in a Host Link System. The host computer is used for data management and overall system control. Host computers are generally personal or business computers.

HR area A data area used to store and manipulate data, and to preserve data when power to the PC is turned OFF.

increment Increasing a numeric value.

indirect address An address whose contents indicates another address. The contents of the second address will be used as the operand. Indirect addressing is possible in the DM area only.

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 IR area that is allocated to hold the status of an input.

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.

instruction A direction given in the program that tells the PC of an action to be carried out, and which data is 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. Although any logically related group of instructions could be called an instruction block, the term is generally used to refer to blocks of instructions called logic blocks that require logic block instructions to relate them to other instructions or logic blocks.

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 within 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.

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.

interrupt (signal) A signal that stops normal program execution and causes a subroutine to be run.

Interrupt Input Unit A Rack-mounting Unit used to input external interrupts into a PC System.

inverse 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.

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 in certain PCs to monitor and control I/O points on Expansion I/O Units.

I/O devices The devices to which terminals on I/O Units, Special I/O Units, or Intelligent I/O Units are connected. 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 I/O Rack in certain PCs to interface the Expansion I/O Rack to the CPU Rack.

I/O Link Created in an Optical Remote I/O System to enable input/output of one or two IR words directly between PCs. The words are input/output between the PC controlling the Master and a PC connected to the Remote I/O System through an I/O Link Unit or an I/O Link Rack.

I/O Link Unit A Unit used with certain PCs to create an I/O Link in an Optical Remote I/O System.

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 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 IR area 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 Unit The most basic type of Unit mounted to a backplane to create a Rack. 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 word A word in the IR area that is allocated to a Unit in the PC System.

IR area A data area whose principal function is to hold the status of inputs coming into the system and that of outputs that are to be set out of the system. Bits and words in the IR that are used this way are called I/O bits and I/O words. The remaining bits in the IR area are work bits.

JIS 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 inbetween. Jumps are usually conditional on an execution condition.

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 a ladder-diagram program.

ladder instruction An instruction that represents the ‘rung’ portion of a ladder-diagram program. The other instructions in a ladder diagram fall along the right side of the diagram and are called terminal instructions.

Ladder Support Software A software package that provides most of the functions of the Factory Intelligent Terminal on an IBM AT, IBM XT, or compatible computer.

LAN An acronym for local area network.

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 Adapter A Unit used to connect communications lines, either to branch the lines or to convert between different types of cable. There are two types of Link Adapter: Branching Link Adapters and Converting Link Adapters.

link A hardware or software connection formed between two Units. “Link” can refer either to a part of the physical connection between two Units (e.g., optical links in Wired Remote I/O Systems) or a software connection created to data existing at another location (Network Data Links).

linkable slot A slot on either a CPU or Expansion I/O Backplane to which a Link Unit can be mounted. Backplanes differ in the slots to which Link Units can be mounted.

Link System A system that includes one or more of the following systems: Remote I/O System, PC Link System, Host Link System, or Net Link System.

Link Unit Any of the Units used to connect a PC to a Link System. These are Remote I/O Units, I/O Link Units, PC Link Units, Host Link Units, and 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.

local area network A network consisting of nodes or positions in a loop arrangement. Each node can be any one of a number of devices, which can transfer data to and from each other.

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.

LR area A data area that is used in a PC Link System so that data can be transferred between two or more PCs. If a PC Link System is not used, the LR area is available for use as work bits.

LSS Abbreviation for Ladder Support Software.

main program All of a program except for the subroutines.

masking 'Covering' an interrupt signal so that the interrupt is not effective until the mask is removed.

Master Short for Remote I/O Master Unit.

memory area Any of the areas in the PC used to hold data or programs.

mnemonic code A form of a ladder-diagram program that consists of a sequential list of the instructions without using a ladder diagram. Mnemonic code is required to input a program into a PC when using a Programming Console.

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.

nest 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.

Net Link System An optical LAN formed from PCs connected through Net Link Units. A Net Link System also normally contains nodes interfacing computers and other peripheral devices. PCs in the Net Link System can pass data back and forth, receive commands from any interfaced computer, and share any interfaced peripheral device.

Net Link Unit The Unit used to connect PCs to a Net Link System. The full name is "SYSMAC Net Link Unit".

Network Service Board A device with an interface to connect devices other than PCs to a Net Link System.

Network Service Unit A Unit that provides two interfaces to connect peripheral devices to a Net Link System.

node One of the positions in a LAN. Each node incorporates a device that can communicate with the devices at all of the other nodes. The device at a node is identified by the node number. One loop of a Net Link System (OMRON's LAN) can consist of up to 126 nodes. Each node is occupied by a Net Link Unit mounted to a PC or a device providing an interface to a computer or other peripheral device.

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 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.

NSB An acronym for Network Service Board.

NSU An acronym for Network Service Unit.

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).

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 cycle.

on-line removal Removing a Rack-mounted Unit for replacement or maintenance during PC operation.

operand Bit(s) or word(s) 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.

Optical I/O Unit A Unit that is connected in an Optical Remote I/O System to provide 8 I/O points. Optical I/O Units are not mounted to a Rack.

Optical Slave Rack A Slave Rack connected through an Optical Remote I/O Slave Unit.

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 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.

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.

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.

PC An acronym for Programmable Controller.

PCB An acronym for printed circuit board.

PC configuration The arrangement and interconnections of the Units that are put together to form a functional PC.

PCF Acronym for plastic-clad optical fiber cable.

PC Link System A system in which PCs are connected through PC Link Units to enable them to share common data areas, i.e., each of the PCs writes to certain words in the LR area and receives the data of the words written by all other PC Link Units connected in series with it.

PC Link Unit The Unit used to connect PCs in a PC Link System.

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.

peripheral device Devices connected to a PC System to aid in system operation. Peripheral devices include printers, programming devices, external storage media, etc.

port A connector on a PC or computer that serves as a connection to an external device.

present value The current value registered in a device at any instant during its operation. Present value is abbreviated as PV.

printed circuit board A board onto which electrical circuits are printed for mounting into a computer or electrical device.

Printer Interface Unit A Unit used to interface a printer so that ladder diagrams and other data can be printed out.

program The list of instructions that tells the PC the sequence of control actions to be carried out.

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-component Programmable Controllers are available, building-block Programmable Controllers are constructed from separate components. Such building-block Programmable Controllers are formed only when enough of these separate components are assembled to form a functional assembly, i.e., no one individual Unit is 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.

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.

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.

PV Acronym for present value.

Rack An assembly of various Units on a Backplane that forms a functional unit in a building-block PC System. Racks include CPU Racks, Expansion I/O Racks, I/O Racks, and Slave Racks.

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.

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 Master Unit The Unit in a Remote I/O System through which signals are sent to all other Remote I/O Units. The Remote I/O Master Unit is mounted either to a CPU Rack or an Expansion I/O Rack connected to the CPU Rack. Remote I/O Master Unit is generally abbreviated to Master.

Remote I/O Slave Unit A Unit mounted to a Backplane to form a Slave Rack. Remote I/O Slave Unit is generally abbreviated to Slave.

Remote I/O System A system in which remote I/O points are controlled through a Master mounted to a CPU Rack or an Expansion I/O Rack connected to the CPU Rack.

Remote I/O Unit Any of the Units in a Remote I/O System. Remote I/O Units include Masters, Slaves, Optical I/O Units, I/O Link Units, and Remote Terminals.

remote I/O word An I/O word allocated to a Unit in a Remote I/O System.

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.

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 Another term for 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.

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.

RUN mode The operating mode used by the PC for normal control operations.

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 for a specified period of time.

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.

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.

Slave Short for Remote I/O Slave Unit.

Slave Rack A Rack containing a Remote I/O Slave Unit and controlled through a Remote I/O Master Unit. Slave Racks are generally located away from the CPU Rack.

slot A position on a Rack (Backplane) to which a Unit can be mounted.

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.

source 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 dedicated 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.

SR area A data area in a PC used mainly for flags, control bits, and other information provided about PC operation. The status of only certain SR bits may be controlled by the operator, i.e., most SR bits can only be read.

SSS Abbreviation for SYSMAC Support Software.

subroutine A group of instructions placed after the main program and executed only if 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.

switching capacity The maximum voltage/current that a relay can safely switch on and off.

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 reserved SR bits as a destination), and mistakes in actual application of instructions (e.g., a call to a subroutine that does not exist).

SYSMAC Support Software A software package installed on a IBM PC/AT or compatible computer to function as a Programming Device.

system configuration The arrangement in which Units in a system are connected.

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.

TC area A data area that can be used only for timers and counters. Each bit in the TC area serves as the access point for the SV, PV, and Completion flag for the timer or counter defined with that bit.

TC number A definer that corresponds to a bit in the TC area and used to define the bit as either a timer or a counter.

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, whether it is a single-frame or multi-frame block. Frames within a multi-frame block are separated by delimiters.

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.

TM area A memory area used to store the results of a trace.

transmission distance The distance that a signal can be transmitted.

TR area A data area used to store execution conditions so that they can be reloaded later for use with other instructions.

trace An operation whereby the program is executed and the resulting data is stored in TM memory to enable step-by-step analysis and debugging.

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.

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 number A number assigned to some Link Units and Special I/O Units to facilitate identification when assigning words or other operating parameters to it.

watchdog timer A timer within the system that ensures that the cycle 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.

Wired Slave Rack A Slave Rack connected through a Wired Remote I/O Slave Unit.

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 multiplier A value between 0 and 3 that is assigned to a Master in a Remote I/O System so that words can be allocated to non-Rack-mounting Units within the System. The word setting made on the Unit is added to 32 times the word multiplier to arrive at the actual word to be allocated.

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 re-

served for work words. Parts of other areas not required for special purposes may also be used as work words, e.g., LR words not used in a PC Link or Net Link System.

Index

E

F

G

H

I

J

K

L

M

N

O

P

R

S

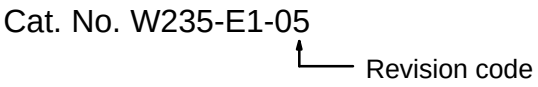
T

U

W

Revision History

A manual revision code appears as a suffix to the catalog number on the front cover of the manual.



The following table outlines the changes made to the manual during each revision. Page numbers refer to the previous version.

Revision code	Date	Revised content
1	January 1994	Original production
2	August 1994	Host Link Commands have been added as *Section 10*. **Page 5:** Available manuals table undated. **Page 16:** Paragraph and table added at top of page. **Page 17:** New CPU models and indicators graphics added. **Page 19:** New C200H CPU capabilities table added. **Page 20:** New C200HS CPU capabilities table added. **Pages 31, 381:** Functions redefined for SR 264 and SR 265. **Page 34:** New information added to SR Area table. **Page 48:** First table under *3-5 AR (Auxiliary Relay) Area* replaced. AR Area Flags and Control Bits table corrected. **Page 51:** *3-5-4 SYSMAC LINK System Data Link Settings* added. **Page 52:** *3-5-6 Active Node Flags* added. **Page 52:** *3-5-7 SYSMAC LINK/SYSMAC NET Link System Service Time (CPU31-E/33-E)* added. **Page 90:** *4-6-9 SYSMAC NET Link Table Transfer* added. **Page 125:** Commands SEND(90) and RECV(98) added to tables. **Page 291:** *5-26-1 Network Instructions* added. **Page 329:** SEND(90) and RECV(98) added to tables. **Page 399:** *9-6 Host Link Errors* added. **Page 359:** New CPU models added. **Page 363:** SYSMAC NET Link and SYSMAC LINK models added. **Page 366:** SYSMAC NET Link and SYSMAC LINK optical fiber product information added. **Appendix B :** Information added to Instructions table. **Appendix C :** Information added to Instructions table. **Appendix D :** Information added to SR and AR Memory tables.

Revision code	Date	Revised content
2A	April 1995	The following instructions have been corrected: ASFT(—) to ASFT(17), XFRB(—) to XFRB(62), MCMP(—) to MCMP(19), CMPL(—) to CMPL(60), BCMP(—) to BCMP(68), ZCP(—) to ZCP(88), SEC(—) to SEC(65), HMS(—) to HMS(66), LINE(—) to LINE(63), COLM(—) to COLM(64), APR(—) to APR(69), INT(—) to INT(89), SCAN(—) to SCAN(18), LMSG(—) to LMSG(47), TERM(—) to TERM(48), MPRF(—) to MPRF(61), and BCNT(—) to BCNT(67). **Page 7:** Internal auxiliary relays I and II areas renamed and corrected in *Increased SR Area*. **Page 17:** *CPU Indicators* descriptions corrected. **Page 21:** Note added after the table. **Page 23:** Functions for pin 5 corrected. **Page 34:** Function of bit 09 of word 252 corrected. **Pages 36, 37:** Function of bits 08 to 10 of word 269 corrected. Functions of words , 267, 270, 271, 274, and 277 to 279 corrected. Words 298 to 299 added. *SYSMAC NET/SYSMAC LINK System* added. **Page 43:** *RS-232C Port Communications Areas* added. **Page 44:** “UM” changed to “Ladder Diagram”. **Page 47:** *Restarting Special I/O Units* added. **Page 57:** Note added. **Page 83:** “PC Unit” corrected to “Main Rack”. **Page 139:** Last sentence in Limitations replaced. **Page 154:** Note removed. **Page 170:** “CB+32” corrected to “CB+31” in the description. The bits for the last five ranges in the description corrected. **Page 179:** Note removed. **Page 180:** Note removed. Example corrected. **Pages 196, 197:** Notes removed. **Page 240:** The operation amount when the SV and PV match corrected from 50% to 0%. **Page 235:** Note removed. **Page 241:** Second graph corrected. “Step response” corrected to “Ramp response” for the bottom graph. **Page 242:** “Step response” corrected to “Ramp response” for the top graph. Second graph corrected. **Page 249:** “DM 6618” corrected to “DM 6622” in the diagram for *Scheduled Interrupts*. **Page 250:** SYSMAC LINK/SYSMAC NET servicing added to *Normal Interrupt Mode (C200H Compatible)*. Interrupt response time and Note 1 added to *High-speed Interrupt Mode (C200HS)*. **Page 271:** *Resetting Errors* rewritten. **Page 275:** Third paragraph in the description for *5-25-6 TERMINAL MODE – TERM(—)* corrected. **Page 277:** Execution time for *5-25-9 GROUP-2 HIGH-DENSITY I/O REFRESH – MPRF(—)* corrected. **Page 282:** Note c) corrected. **Pages 293 to 295:** RS-232C port information added. Control word and flag information corrected. **Page 297:** *Hardware* rewritten. **Page 298:** Text added to *Timing*. Timing chart corrected. **Page 301:** Example corrected. **Pages 302, 304:** Word names corrected in *Using the Instruction*. **Pages 302, 304, 306:** Charts corrected in *Using the Instruction*. **Pages 312 to 317:** *6-1 Cycle Time* and *6-2 Calculating Cycle Time* have been rewritten. **Pages 326 and 327:** *6-4 I/O Response Time* rewritten. **Page 353:** AR area corrected to SR area. Table of corresponding keys added. **Page 355:** *Section 8 Communications* added. **Page 364:** Text added to possible correction for Memory Cassette Transfer Error. **Page 366:** Text added to possible cause and correction for Too many Units. *Communications Errors* added after the table. **Page 367:** Error flags added to the table. **Pages 403 to 411:** Standard models updated. **Page 423:** Data for Special Relay Areas 1 and 2 corrected. **Pages 424 to 427:** SR Area table updated to reflect SR Area table in *Section 3 Memory Areas*. **Page 429:** “DM 0000” corrected to “DM 6001” in *DM Area (Error Log)* table. **Page 432:** RS-232C Port Settings added.

Revision code	Date	Revised content
2B	July 1995	The following corrections and additions were made. **Page 6:** SYSMAC Support Software added and LSS removed. **Pages 23 and 374:** Default communications parameters changed. **Page 26:** I/O Terminals and B7A Interface Unit added and macro bits corrected. **Page 31:** List of Units that don't use slot words corrected. **Pages 32, 33, 34, 39, 43, 48, 50:** Group-2 B7A Interface Units and I/O Terminals added. **Page 39:** Type of data stored in SR 25108 to SR 25115 corrected to BCD. **Page 44:** Section added on Special Unit Error Flag **Pages 45, 46:** Mode for SR 26408 to SR 26415 changed. **Page 48:** Details added to first table. **Page 52:** Heading shortened. **Page 120:** Caution added. **Pages 215, 216:** Limits for operands clarified/corrected. **Page 243:** Caution added. **Pages 243, 244:** Variable name changed for Y and corrected variable description. **Page 245:** Bottom graphic and related description corrected. **Page 274:** Explanation in last paragraph clarified. **Page 283:** Execution time corrected for IORF(97). **Pages 299 to 301:** CPU models specified in control word contents corrected. **Page 314:** Graphic corrected. **Page 336:** Description of Host Link Systems expanded. **Page 382:** Last argument corrected in application example **Page 405:** Top graphic corrected and "frame 2" changed to "frame 3" in second graphic. **Page 461:** Default added for DM 6610 and DM 6618. **Page 462:** Settings for DM 6620 bits 00 to 09 and DM 6621 bit 08 to 15 corrected. **Page 463:** Default for DM 6654 bits 00 to 07 changed.
2C	August 1996	The *Precautions* section was added after *About this Manual*. **Page 88:** Special I/O Unit type "N" corrected from Host Link Unit to Position Control Unit. **Page 110:** Ladder diagram corrected for the self-maintaining bit example. **Page 158:** ASFT(17) example changed. **Page 191:** "Content of the source words is zero" corrected to "Content of a source word is zero" for the ER Flags. **Page 226:** DVB(53) example changed. **Page 280:** "SR" added to the first source word of LMSG(47). **Page 395:** "Possible correction" for "I/O bus error" corrected. **Page 430:** "Command Format" corrected to "Response Format" for *11-3-36 Undefined Command — IC*.
3	May 1999	**Page 245:** Proportional operation diagram corrected. **Page 283:** Note and execution time information in *5-25-8 I/O REFRESH – IORF(97)* corrected. **Page 321:** C200H-PID□□ added to *Special I/O Unit Refresh*.
4	April 2001	**Page xv:** Minor change made to precautionary information on mounting. **Page 23:** Information on pin number 5 changed. **Pages 140, 144, 145:** Information on SV settings added. **Pages 433, 434, 435:** Information added to tables in several places.
05	February 2002	**Page 177:** Ladder symbol removed.

OMRON

OMRON ELECTRONICS LLC

1 Commerce Drive
Schaumburg, IL 60173
847.843.7900
For US technical support or
other inquiries: 800.556.6766

OMRON CANADA, INC.

885 Milner Avenue
Toronto, Ontario M1B 5V8
416.286.6465

OMRON ON-LINE

Global - http://www.omron.com
USA - http://www.omron.com/oei
Canada - http://www.omron.ca

UNITED STATES

To locate a Regional Sales Office, local Distributor or to obtain product information, call: 847.843.7900

CANADA REGIONAL SALES OFFICES

Ontario	Toronto	416.286.6465
	Kitchener	519.896.1144
	Kingston	613.376.3968
Quebec	Montreal	514.636.6676
British Columbia	Vancouver	604.522.8855
Alberta	Edmonton	403.440.0818
	Calgary	403.257.3095

BRAZIL SALES OFFICE

Sao Paulo	55.11.5564.6488

ARGENTINA SALES OFFICE

Cono Sur	54.114.787.1129

MEXICO SALES OFFICES

Florida	954.227.2121	**Ciudad Juarez**	656.623.7083
Mexico, D.F.	555.534.1195	**Monterrey, N.L.**	818.377.4281

W235-E1-5 12/02

LifeElectronics
Живое партнерство

ООО «ЛайфЭлектроникс» **“LifeElectronics” LLC**

ИНН 7805602321 КПП 780501001 Р/С 40702810122510004610 ФАКБ "АБСОЛЮТ БАНК" (ЗАО) в г.Санкт-Петербурге К/С 30101810900000000703 БИК 044030703

Компания «Life Electronics» занимается поставками электронных компонентов импортного и отечественного производства от производителей и со складов крупных дистрибьюторов Европы, Америки и Азии.

С конца 2013 года компания активно расширяет линейку поставок компонентов по направлению коаксиальный кабель, кварцевые генераторы и конденсаторы (керамические, пленочные, электролитические), за счёт заключения дистрибьюторских договоров

Мы предлагаем:

- *Конкурентоспособные цены и скидки постоянным клиентам.*
- *Специальные условия для постоянных клиентов.*
- *Подбор аналогов.*
- *Поставку компонентов в любых объемах, удовлетворяющих вашим потребностям.*

- *Приемлемые сроки поставки, возможна ускоренная поставка.*
- *Доставку товара в любую точку России и стран СНГ.*
- *Комплексную поставку.*
- *Работу по проектам и поставку образцов.*
- *Формирование склада под заказчика.*

- *Сертификаты соответствия на поставляемую продукцию (по желанию клиента).*
- *Тестирование поставляемой продукции.*
- *Поставку компонентов, требующих военную и космическую приемку.*
- *Входной контроль качества.*
- *Наличие сертификата ISO.*

В составе нашей компании организован Конструкторский отдел, призванный помогать разработчикам, и инженерам.

Конструкторский отдел помогает осуществить:

- *Регистрацию проекта у производителя компонентов.*
- *Техническую поддержку проекта.*
- *Защиту от снятия компонента с производства.*
- *Оценку стоимости проекта по компонентам.*
- *Изготовление тестовой платы монтаж и пусконаладочные работы.*

Тел: +7 (812) 336 43 04 (многоканальный)
Email: org@lifeelectronics.ru

www.lifeelectronics.ru