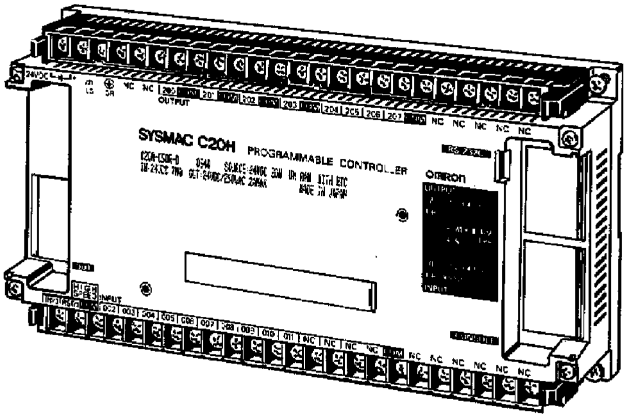
Mini H-type PCs:
C20H, C28H, C40H, and C60H
Programmable Controllers

Operation Manual

Revised July 1994



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 the product.

DANGER! Indicates information that, if not heeded, is likely to result in loss of life or serious injury.

WARNING Indicates information that, if not heeded, could possibly result in loss of life or serious injury.

Caution Indicates information that, if not heeded, could result in relative serious or minor injury, damage to the product, or faulty operation.

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.

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

TABLE OF CONTENTS

SECTION 1
Introduction

SECTION 2
Hardware Considerations

SECTION 3
Memory Areas

SECTION 4
Writing and Entering Programs

About this Manual:

The OMRON C20H, C28H, C40H, and C60H offer a simple but effective way to automate processing. Manufacturing, assembly, packaging, and many other processes can be automated to save time and money.

This manual describes the characteristics and abilities of the PCs, as well as programming operations and instructions and other aspects of operation and preparation. Before attempting to operate the PC, thoroughly familiarize yourself with the information contained herein. Hardware information is provided in detail in the *Mini H-type PCs: C20H/C28H/C40H/C60H Installation Guide*. A table of other manuals that can be used in conjunction with this manual is provided at the end of *Section 1 Introduction*.

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 Mini H-type 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 Mini H-type 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.

S*ection 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.

S*ection 8 RS-232C Interface* describes the modes, settings, and procedures essential for making use of the built-in RS-232C interface. It also list all of the commands that can be downloaded from a host computer connected to the RS-232C interface.

Finally, ***Section 9 Troubleshooting*** provides information on error indications and other means of reducing down-time. Information in this section is also useful when debugging programs.

The ***Appendices*** provide tables of standard OMRON products available for the Mini H-type 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.

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. Also provided are descriptions of peripheral devices used with the Mini H-type PCs, and a table of other manuals available to use with this manual for special PC applications.

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 an 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 Mini H-type PCs, like the other C-series PCs, use 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 Mini H-type PCs.

1-2 The Origins of PC Logic

PCs historically originate in relay-based control systems. 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.

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.

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 signal 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.

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 C20H-type PC. Assuming you have already purchased one or more of these PCs, you would be familiar with 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 Units will be required and whether your system configuration will require an Expansion I/O Units. Refer to the *Mini H-type PC Installation Guide*.
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.
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. (Refer to *Section 4 Writing and Entering Programs*, *Section 5 Instruction Set*, and *Section 6 Program Execution Timing*)
5. Input the program and all required operating parameters into the PC. (Refer to *Section 7 Program Debugging and Execution*)

6. Debug the program, first to eliminate any syntax errors, and then to find execution errors. (Refer to *Section 7 Program Debugging and Execution* and *Section 9 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 *C20H, C28H, C40H 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. (Refer to *Section 7 Program Debugging and Execution* and *Section 9 Troubleshooting*)

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.

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. It is connected directly to the CPU without requiring a separate interface. Programming Console operations are described later in this manual. The Programming Console cannot be used when a Peripheral Interface Unit or CPU-mounting Host Link Unit is connected.

Graphic Programming Console: GPC

The GPC 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 liquid crystal display, making confirmation and modification quick and easy. Syntax checks may also be performed on the programs before they are downloaded to the PC. Many other functions are available, depending on the Memory Pack used with the GPC.

The GPC also functions as an interface to copy programs directly to a standard cassette tape recorder. A PROM Writer, Floppy Disk Interface Unit, or Printer Interface Unit can be directly mounted to the GPC to output programs directly to an EPROM chip, floppy disk drive, or printing device, respectively.

The GPC is connected to a Mini H-type PC via a Peripheral Interface Unit.

Ladder Support Software: LSS

LSS is designed to run on IBM AT/XT compatibles to enable all of the operations available on the GPC.

The LSS is connected to a Mini H-type PC via a Peripheral Interface Unit or the RS-232C interface.

Factory Intelligent Terminal: FIT

The FIT is an OMRON computer with specially designed software that allows you to perform all of the operations that are available with the GPC or LSS. Programs can also be output directly to an EPROM chip, floppy disk drive, or printing device without any additional interface. The FIT has an EPROM writer and two 3.5” floppy disk drives built in.

The FIT is connected to a Mini H-type PC via a Peripheral Interface Unit or the RS-232C interface.

PROM Writer

Use only the GPC with the PROM Writer to write programs to EPROM chips.

Printer Interface Unit

Use only the GPC with the Printer Interface Unit connected to a printer or X–Y plotter to print out programs in either mnemonic or ladder-diagram form.

Data Access Console

The Data Access Console can be used to monitor or change data in the TC, IR and SR areas. It has two levels of operations, one of which protects certain areas from being changed. It also provides certain special functions, such as keyboard mapping to data areas. (The PC must be set to either Japanese or English to use the Data Access Console.)

1-7 Available Manuals

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

Name	Cat. No.	Contents
C20H, C28H, C40H, C60H PC Installation Guide	W175	Hardware specifications
Data Access Console Operation Guide	W173	Operating and installation procedures for the Data Access Console
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
LSS Operation Manual	W237	Programming procedures for using LSS (Ladder Support Software)
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 a PC to a floppy disk drive
Host Link Unit System Manual	W143	Procedures for creating Host Link Systems combining PCs and host computers.

SECTION 2
Hardware Considerations

This section provides information on hardware aspects of the Mini H-type PCs that are relevant to programming and software operation. These include indicators on the CPU and basic PC configuration. This information is covered in detail in the Mini H-type PCs: C20H/C28H/C40H/C60H *Installation Guide*.

2-1 Indicators

CPU indicators provide visual information on the general operation of the PC. Although not substitutes for proper error programming using the flags and other error indicators provided in the data areas of memory, these indicators provide ready confirmation of proper operation.

CPU Indicators

CPU indicators are described in the following table. Indicators are the same for all Mini H-type PCs.

Indicator	Function
POWER	Lights when power is supplied to the CPU.
RUN	Lights when the CPU is operating normally.
ALM/ERR	ALARM: Flashes when a non-fatal error is discovered in error diagnosis operations. PC operation will continue. ERROR: Lights when a fatal error is discovered in error diagnosis operations. When this indicator lights, the RUN indicator will go off, CPU operation will be stopped, and all outputs from the PC will be turned OFF.
OUTPUT	Lights when the Unit should turn on an output.
INPUT	Lights when the Unit has an input signal.

2-2 PC Configuration

With a Mini H-type PC System, you can set up a configuration utilizing anywhere from 20 to 240 I/O points. Depending on the number of I/O points you need, you can choose any of the four PCs (C20H, C28H, C40H, or C60H) and combine it with up to three Expansion I/O Units for a total of four Units in a PC System.

For example, if you only need a very simple system, a C20H PC alone will provide 12 input and 8 output points. For more sophisticated operations, a C60H PC can be connected with three C60H Expansion I/O Units to provide 128 input and 112 output points. You can select any other combination of Units within this range, which will provide exactly the capacity you require for your Control System.

For detailed explanations of possible system configurations, refer to the appropriate sections of the *Mini H-type PCs: C20H, C28H, C40H, C60H Installation Guide*.

RS-232C and Peripheral Device Interfaces

The Mini H-type PCs have a built-in RS-232C interface, which allows them to connect directly to an FA Computer, Display Terminal, commercially available printers, and other devices which employ an RS-232C interface. Mini H-type PCs equipped with an E-V1-version CPU can transfer data with similarly equipped Mini H-type PC's through the RS-232C interface.

In addition, the built-in Peripheral Interface Unit connector allows connection to an FA Computer via a CPU-mounting Host Link Unit, and the built-in Peripheral Device (Console) interface allows connection to a Programming Console or a Data Access Console.

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**, each with a different function. The areas generally accessible by the user in programming are classified as **data areas**. The other memory area is the Program Memory, where the user's program is actually stored. This section describes these areas individually and provides information that will be necessary to use them.

3-1 Introduction

Details, including the name, acronym, range, and function of each area are summarized in the following table. All but the last three of these areas are data areas. Data and memory areas are normally referred to by their acronyms.

Area	Acronym	Range	Function
Internal Relay	IR	Words: 000 to 246 Bits: 0000 to 24615	Used to control I/O points, other bits, timers, and counters, and to temporarily store data.
Special Relay	SR	Words: 247 to 255 Bits: 24700 to 25507	Contains system clocks, flags, control bits, and status information.
Auxiliary Relay	AR	Words: AR 00 to AR 27 Bits: AR 00 to AR 2715	Contains flags and bits for special functions. Retains status during power failure.
Data Memory	DM	Read/write: DM 0000 to DM 0999 Read only: DM 1000 to DM 1999	Used for internal data storage and manipulation.
Holding Relay	HR	Words: HR 00 to HR 99 Bits: HR 0000 to HR 9915	Used to store data and to retain the data values when the power to the PC is turned off.
Link Relay	LR	Words: LR 00 to LR 63 Bits: LR 0000 to 6315	Available for use as work bits.
Timer/Counter	TC	TC 000 to TC 511 (TC numbers used to access other information)	Used to define timers and counters, and to access completion flags, PV, and SV. In general, when used as a bit operand, a TC number accesses the completion flag for the timer or counter defined using the TC number. When used as a word operand, the TC number accesses the present value of the timer or counter.
Temporary Relay	TR	TR 00 to TR 07 (bits only)	Used to temporarily store and retrieve execution conditions. These bits can only be used in the Load and Output instructions. Storing and retrieving execution conditions is necessary when programming certain types of branching ladder diagrams.
Program Memory	UM	UM: Depends on Memory Unit used.	Contains the program executed by the CPU.

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 Entering Programs*.

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 all 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 (Timer/Counter) Area* for more details on TC numbers and to *5-10 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, and the next digit to the left must be 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 (Timer/Counter) 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 45	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 entering 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 entering 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 5757 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 *5-14 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.

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 C20H/C28H/C40H/C60H, the IR area is comprised of words 000 to 246.

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 are allocated to CPUs and Expansion I/O Units as follows. The CPU is allocated the first four words, beginning with 000, and each Expansion I/O Unit connected is allocated four words beginning with, in order, 010, 020, and 030. (Although each Unit is allocated four words, the number of bits actually used depends on the type of Unit, i.e., C20H, C28H, or C40H.) The first two words are allocated for inputs and the other two are allocated for outputs. The following table illustrates I/O allocation for CPUs and Expansion I/O Units.

Connected Unit	Designation	C20H or C28H	C40H or C60H
CPU	Input words	IR 000	IR 000 to IR 001
	Output words	IR 002	IR 002 to IR 003
1st Expansion I/O Unit	Input words	IR 010	IR 010 to IR 011
	Output words	IR 012	IR 012 to IR 013
2nd Expansion I/O Unit	Input words	IR 020	IR 020 to IR 021
	Output words	IR 022	IR 022 to IR 023
3rd Expansion I/O Unit	Input words	IR 030	IR 030 to IR 031
	Output words	IR 032	IR 032 to IR 033

The following tables show the I/O bits used by each of the CPUs and Expansion I/O Units. For the Expansion I/O Units, n = IR 010, IR 020, or IR 030, depending on the order of connection. Unused bits in input words cannot be used in the program. Unused bits in output words can be used as work bits, as shown below.

CPUs

Model	Input bits	Output bits	Work bits
C20H	IR 00000 to IR 00011	IR 00200 to IR 00207	IR 00208 to IR 00215
C28H	IR 00000 to IR 00015	IR 00200 to IR 00211	IR 00212 to IR 00215
C40H	IR 00000 to IR 00015 IR 00100 to IR 00107	IR 00200 to IR 00211 IR 00300 to IR 00303	IR 00212 to IR 00215 IR 00304 to IR 00315
C60H	IR 00000 to IR 00015 IR 00100 to IR 00115	IR 00200 to IR 00211 IR 00300 to IR 00315	IR 00212 to IR 00215

Expansion I/O Units

Model	Input bits	Output bits	Work bits
C20H	Bits 00 to 11 of IR n	Bits 00 to 07 of IR n+2	Bits 08 to 15 of IR n+2
C28H	Bits 00 to 15 of IR n	Bits 00 to 11 of IR n+2	Bits 12 to 15 of IR n+2
C40H	Bits 00 to 15 of IR n Bits 00 to 07 of IR n+1	Bits 00 to 11 of IR n+2 Bits 00 to 03 of IR n+3	Bits 12 to 15 of IR n+2 Bits 04 to 15 of IR n+3
C60H	Bits 00 to 15 of IR n Bits 00 to 15 of IR n+1	Bits 00 to 11 of IR n+2 Bits 00 to 15 of IR n+3	Bits 12 to 15 of IR n+2

High-speed Counter

When the High-speed Counter instruction is used, IR 00000 is used as a count input and IR 00001 is used as a hardware reset/disable bit.

I/O Words

If a Unit brings an input 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 is turned ON, the input bit assigned to it is turned ON also. These facts can be used in the program to access input status and control output status through I/O bits.

I/O bits that are not assigned to I/O points can be used as work 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 *5-11-1 SHIFT REGISTER – SFT(10)* for an example that uses an output bit in two ‘bit-control’ instructions.

IR 00200 to IR 00207 are used as interrupt outputs. For an explanation of these, refer to *3-8 TC (Timer/Counter) Area.*

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 247 through 255; bit addresses, from 24700 through 25515.

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 address except that Link System bits are grouped together first.

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.

SR 25209 through 25213 are all control bits. They can be turned ON and OFF from the program, i.e., they can be manipulated with the Output and Output NOT instructions. Any of these bits not assigned specific functions should be left OFF. Bits in words SR 247 through SR 250, only, can be used as work bits if the Systems for which these bits are dedicated are not used by the PC.

Word(s)	Bit(s)	Function
247 to 250	00 to 07	Reserved.
	08 to 15	Reserved.
251	00 to 15	Not used.
252	00 to 07	Not used.
	08	Dual-usage as: RS-232C Communications Error Flag and CPU-mounting Host Link Unit Communications Error Flag
	09	Dual-usage as: RS-232C Interface Restart Bit and CPU-mounting Host Link Unit Restart Bit

Word(s)	Bit(s)	Function
	10	Interrupt Output Enable Bit
	11	Forced Status Hold Bit
	12	I/O Status Hold Bit
	13	Reserved
	14 to 15	Not used.
253	00 to 07	FAL number output area.
	08	Battery Alarm Flag
	09	Cycle Time Error Flag
	10 to 12	Not used.
	13	Always ON Flag
	14	Always OFF Flag
	15	First Cycle
254	00	1-minute clock pulse bit
	01	0.02-second clock pulse bit
	02 to 06	Reserved
	07	Step Flag
	08 to 15	Reserved
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

3-4-1 RS-232C/CPU-mounting Host Link Unit Flags and Control Bits

The RS-232C Interface and CPU-mounting Host Link Unit share an Error Flag and a Restart Bit, as shown in the following table.

SR bit	Functions
25208	1. RS-232C Interface Error Flag 2. CPU-mounting Host Link Unit Communications Error Flag
25209	1. RS-232C Restart Bit 2. CPU-mounting Host Link Unit Restart Bit

The Error Flag is turned ON when an error has occurred either in communications via the RS-232 Interface or between the CPU and the CPU-mounting Host Link Unit. When the Restart Bit is turned ON and then OFF, both the RS-232C Interface and the CPU-mounting Host Link Unit will be restarted.

3-4-2 Interrupt Output Enable Bit

Turn ON SR 25210 to enable interrupt outputs used with the high-speed counter. SR 25210 is automatically turned OFF when power is turned ON, and must be turned ON from the program of from a Programming Device. SR 25210 can be turned ON during operation. No interrupt outputs will be made while SR 25210 is OFF.

3-4-3 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. Unless the system operation instruction (SYS(49)) is used to prevent it (see below), SR 25211 is turned off when power to the PC is turned on. If power is interrupted while force-setting/resetting bits, it will be necessary to repeat the operation.

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 FIT.

Maintaining Status during Startup

The status of SR 25211 and thus the status of force-set/force-reset bits can be maintained when power is turned off and on by inserting the System Operation instruction (SYS(49)) into the program with the proper operand. If SYS(49) is used in this way, 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/force-reset bits will also be preserved, as shown in the following table. The use of SYS(49) does not affect operation when switching to run mode, i.e., force-set/force-reset bits always return to default status when switching to RUN mode.

Status before shutdown		Status at next startup	
SR 25211	**SYS(49)**	**SR 25211**	**Force-set/reset bits**
ON	Executed	ON	Status maintained
	Not executed	OFF	Default status
OFF	Executed	OFF	Default status
	Not executed	OFF	Default status

Refer to *Section 5 Instruction Set* for details on SYS(49).

The System Operation instruction (SYS(49)) can be used to turn off the operation of the battery alarm if desired, e.g., when DM 1000 to DM 1999 is placed in ROM and a battery is not used in operation. Refer to *Section 5 Instruction Set* for details.

3-4-4 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. If SR 25212 is on, bit status will be maintained; if SR 25212 is off, all IR and LR area bits will be reset. Unless the system operation instruction (SYS(49)) is used to prevent it (see below), SR 25212 is turned off when power to the PC is turned on.

SR 25212 can be turned ON from the program using the Output instruction, or it can be turned ON from a Peripheral Device.

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 inserting the System Operation instruction (SYS(49)) into the program with the proper operand. If SYS(49) is used in this way, 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	**SYS(49)**	**SR 25212**	**IR and LR bits**
ON	Executed	ON	Status maintained
	Not executed	OFF	Reset
OFF	Executed	OFF	Reset
	Not executed	OFF	Reset

Refer to *Section 5 Instruction Set* for details on SYS(49).

3-4-5 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.

3-4-6 Battery Alarm Flag

SR bit 25308 is turned ON if the voltage of the CPU battery drops. The ALM 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 System Operation instruction (SYS(49)) can be used to turn off the operation of the battery alarm if desired, e.g., when DM 1000 to DM 1999 is placed in ROM and a battery is not used in operation. Refer to *Section 5 Instruction Set* for details.

3-4-7 Cycle Time Error Flag

SR bit 25309 is turned ON if the cycle time exceeds 100 ms. The ALM 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. Timers may become inaccurate after the cycle time exceeds 100 ms.

If the Cycle Time Set Enable Bit in the system parameter area is ON, the time used to determine a cycle time error, also in the system parameter area, can be set. Refer to *3-6 DM (Data Memory) Area* for details.

3-4-8 First Cycle Flag

SR bit 25315 is turned 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 *5-10 Timer and Counter Instructions*.

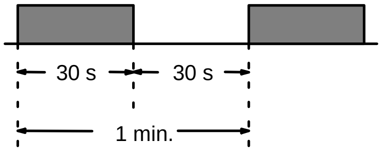
3-4-9 Clock Pulse Bits

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, i.e., each clock pulse has a duty factor of 50%.

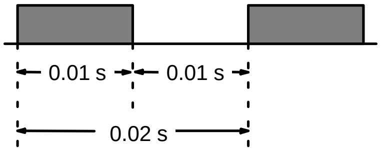
These clock pulse bits are often used with counter instructions to create timers. Refer to *5-10 Timer and Counter Instructions* for an example of this.

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

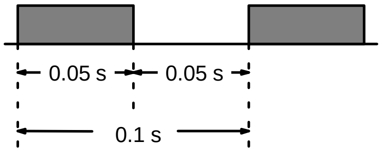
Bit 25400
1-min clock pulse



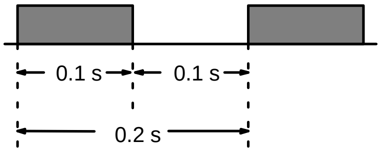
Bit 25401
0.02-s clock pulse



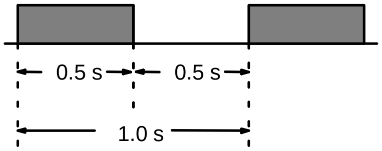
Bit 25500
0.1-s clock pulse



Bit 25501
0.2-s clock pulse



Bit 25502
1.0-s clock pulse



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.

3-4-10 Step Flag

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

3-4-11 Instruction Execution Error Flag, ER

SR bit 25503 is turned ON if an attempt is made to execute an instruction with incorrect operand data. Common causes of an instruction error are non-BCD operand data when BCD data is required, or an indirectly addressed DM word that is non-existent. **When the ER Flag is ON, the current instruction will not be executed.**

3-4-12 Arithmetic Flags

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

Caution These flags are all reset when the End instruction is executed, and therefore cannot be monitored from a programming device.

Refer to *5-11 Data Shifting*, *5-13 Data Comparison*, *5-15 BCD Calculations*, and *5-16 Binary Calculations* for details.

Carry Flag, CY

SR bit 25504 is turned ON when there is a carry in the result of an arithmetic operation or when a rotate or shift instruction moves a “1” into CY. The content of CY is also used in some arithmetic operations, e.g., it is added or subtracted along with other operands. This flag can be set and cleared from the program using the Set Carry and Clear Carry instructions.

Greater Than Flag, GR SR bit 25505 is turned 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 is turned ON when the result of a comparison shows two operands to be equal or when the result of an arithmetic operation is zero.

Less Than Flag, LE SR bit 25507 is turned 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.

3-5 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 cannot be used for any other purpose. The few bits that are not currently assigned cannot be accessed by the user for any purpose. An overview of the AR area is provided in the following table. Details are provided in order of address thereafter. Words/bits not listed in the following table are not used and cannot be accessed by the program.

The AR area retains status during power interruptions or when changing to PROGRAM mode.

AR Area Overview

Word(s)	Bit(s)	Function
02	00 to 10	Reversible Drum Counter (RDM(60)) Reset Bits
	11	High-speed Counter Reset Bit
	12	High-speed Counter Reset Flag
03	00 to 10	Reversible Drum Counter (RDM(60)) Direction Bits
	11	High-speed Counter Bank Bit
04	00 to 07	RS-232C Communications Error Code
	13	RS-232C Reception Impossible Flag
	14	RS-232C Reception Completed Flag
	15	RS-232C Transmission Possible Flag
05	00 to 07	RS-232C Reception Counter
	08 to 15	RS-232C Transmission Counter
06	00 to 15	RS-232C Bytes Received Area
07	08	TERMINAL Mode Input Cancel Bit
	13	Error History Overwrite Bit
	14	Error History Reset Bit
	15	Error History Enable Bit
08	00 to 15	RS-232C Bytes Input Area
12	00 to 15	System Parameter Warning Flags
13	00 to 13	System Parameter Warning Flags
	14	System Parameter Backup Flag
	15	System Parameter/Backup Area Checksum Flag
14	00 to 03	System Command Response Code
	04 to 06	Not used
	07	System Command Completion Flag
	08 to 11	System Command Command Code
	12 to 14	Not used
	15	System Command Execution Bit
15	00 to 07	Startup Operating Mode
17	00 to 15	Current Time Area

Word(s)	Bit(s)	Function
18 to 21	00 to 15	Calendar/clock Area (AR 2113: Seconds Round-off Bit; AR 2114: Stop Bit; AR 2115: Set Bit)
22	00 to 15	TERMINAL Mode Key Bits
23	00 to 15	Power-off Counter
24	05	SCAN(18) Cycle Time Flag
	15	Programming Console or Peripheral Interface Unit Mounted Flag
25	00 to 15	FALS-generating Address
26	00 to 15	Maximum Cycle Time Area
27	00 to 15	Current Cycle Time Area

3-5-1 Reversible Drum Counter Bits

AR 0200 to AR 0210 (Reversible Drum Counter (RDM(60)) Reset Bits) are turn ON by the user to reset reversible counters created using RDM(60). The number of the bit corresponds to the last two digits of the TC area bit used to define the counter, i.e., AR 0200 resets CNT 500, AR 0201 resets CNT 501, etc.

AR 0300 to AR 0310 (Reversible Drum Counter (RDM(60)) Direction Bits) are turn ON and OFF by the user to set the direction that the reversible counters operate. The number of the bit corresponds to the last two digits of the TC area bit used to define the counter, i.e., AR 0200 resets CNT 500, AR 0201 resets CNT 501, etc. If a bit is OFF, the corresponding counter will increment (count up); if a bit is ON, the corresponding counter will decrement. These bits are refreshed each cycle in MONITOR or RUN mode.

3-5-2 High-speed Counter Bits

AR 0211 (High-speed Counter Reset Bit) is used to reset the high-speed counter created using HDM(61) or using the interrupt drum output function. The number of the bit corresponds to the last two digits of the TC area bit used to define the counter, i.e., AR 0211 resets CNT 511.

AR 0212 (High-speed Counter Reset Flag) is turned ON for one cycle after the high-speed counter created with HDM(61) or with the interrupt drum output function is reset via the hardware reset input (i.e., not with AR 0211).

AR 0311 (High-speed Counter Bank Bit) is turned ON or OFF by the user to designate the counter table bank to be used for the high-speed counter. If AR 0311 is ON, bank 1 will be used; if AR 0311 is OFF, bank 0 will be used.

These bits are refreshed each cycle while in MONITOR or RUN mode. Refer to *Section 5 Instruction Set* for details on HDM(61) and *Section 3 Memory Areas* for details on the interrupt drum output function.

3-5-3 RS-232C Communications Error Code

When an error has occurred in RS-232C communications, the RS-232C Interface Communications Error Flag is turned ON, and a code that indicates the type of error is output to AR 0400 to AR 0407. These codes are in hexadecimal and are as follows:

00:	Parity error
01:	Framing error
02:	Overrun error
03:	FCS error

These bits are refreshed each cycle while using the RS-232C interface.

3-5-4 RS-232C Communications Flags

These 3 flags indicate the current status of RS-232C communications. These bits are refreshed each cycle while the RS-232C interface is being used in ASCII I/O Mode.

Reception Impossible Flag

The RS-232C Reception Impossible Flag (AR 0413) is turned ON when newly received data cannot be input. There are two possible reasons that the new data cannot be input:

1, 2, 3...

1. The previously received data has not yet been input by PIN(64), the RS-232C PORT INPUT instruction.
2. An error occurred during the previous reception.

Reception Completed Flag

The RS-232C Reception Completed Flag (AR 0414) is turned ON when an end code or 200 bytes of data have been received at the RS-232C interface.

Transmission Possible Flag

The RS-232C Transmission Possible Flag (AR 0415) is turned ON when data can be transmitted from the RS-232C interface (i.e., when data is not being transmitted).

3-5-5 RS-232C Communications Counters

When a transmission is received on the RS-232C interface, the number of characters is counted in hexadecimal and then output to AR 0500 to AR 0507 (RS-232C Reception Counter) to assist the user in debugging RS-232C interface communications. This counter is used for all RS-232C interface operating modes, as well as for error characters. The counter can be reset by turning SR 25209 ON and then OFF.

AR 0508 to AR 0515 (RS-232C Transmission Counter) operates the same as AR 0500 to AR 0507, except that it operates for transmisions sent from the PC through the RS-232C interface.

These counters are refreshed every cycle.

3-5-6 RS-232C Bytes Received Area

When a transmission is received in ASCII I/O Mode at the RS-232C interface, the number of bytes of data is counted in BCD and then output to AR 06. The start and end codes are not counted.

Counting stops when the RS-232C Reception Completed Flag (AR 0414) is turned ON. The value stored in AR 06 will not be updated after this flag is turned on even if new data is received at the RS-232C interface. The content of AR 06 is reset to 0 when PIN(64), the RS-232C PORT INPUT instruction, is executed.

AR 06 is refreshed every cycle while the RS-232C interface is being used in ASCII I/O Mode.

3-5-7 TERMINAL Mode Input Cancel Bit

AR 0708 is turned ON by the user to cancel a key equivalent during TERMINAL mode Programming Console operation. (Refer to *Section 7 Program Debugging and Execution* for details on terminal mode operation.)

This bit is refreshed each cycle.

3-5-8 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 (Data Memory) Area* for details on the Error History Area.

Error history bits are refreshed each cycle.

3-5-9 RS-232C Bytes Input Area

While AR 06 contains the number of bytes received in ASCII I/O Mode at the RS-232C interface, AR 08 contains the number of bytes of data (BCD) actually input to the PC by PIN(64), the RS-232C PORT INPUT instruction.

Normally, the bytes input will equal the bytes received, but in some cases the numbers will differ. The two most likely cases are described below.

1, 2, 3...

1. One operand of PIN(64) specifies the number of bytes to input. If the value of this operand is less than the content of AR 06, then the bytes input will not equal the bytes received.
2. PIN(64) might be executed before the RS-232C Reception Completed Flag (AR 0414) is turned ON. In this case, the number of bytes received would continue increasing as more data was received.

The content of AR 08 is updated each time that PIN(64) is executed.

3-5-10 System Parameter Flags

System Parameter Warning Flags

The system parameters are checked for errors when the system command is executed or the system parameters are reset at start-up. If a word in the system parameters has an unacceptable value, its corresponding System Parameter Warning Flag will be turned ON.

The 30 bits from AR 1200 through AR 1313 correspond to the 30 words of the Parameter Area (DM 0900 to DM 0929) as shown below.

AR 1200:	DM 0900	AR 1201:	DM 0901	AR 1202:	DM 0902
AR 1203:	DM 0903	AR 1204:	DM 0904	AR 1205:	DM 0905
AR 1206:	DM 0906	AR 1207:	DM 0907	AR 1208:	DM 0908
AR 1209:	DM 0909	AR 1210:	DM 0910	AR 1211:	DM 0911
AR 1212:	DM 0912	AR 1213:	DM 0913	AR 1214:	DM 0914
AR 1215:	DM 0115	AR 1300:	DM 0916	AR 1301:	DM 0917
AR 1302:	DM 0918	AR 1303:	DM 0919	AR 1304:	DM 0920
AR 1305:	DM 0921	AR 1306:	DM 0922	AR 1307:	DM 0923
AR 1308:	DM 0924	AR 1309:	DM 0925	AR 1310:	DM 0926
AR 1311:	DM 0927	AR 1312:	DM 0928	AR 1313:	DM 0929

If a Warning Flag is turned ON, the default value for the system parameter will be used. Bits assigned to unused DM words are always OFF.

System Parameter Backup Flag

AR 1314 is turned ON when the System Command Execute Bit has been force-set and remains ON until reset by the execution of a system command with a command code of 3.

Parameter/Backup Area Checksum Flag

AR 1315 is turned ON to indicate an error in the Parameter and Backup Areas checksum.

Refer to *3-5-11 System Command Bits* for details on system command bits and to *3-6 DM (Data Memory) Area* for details on the Parameter and Backup Areas.

These bits are refreshed when PC power is turned on and when the System Command Execute Bit is force-set.

3-5-11 System Command Bits

System Command Response Code

When the system command has been executed, a response code is placed in AR 1400 to AR 1403 to indicate the completion status of the command. The System Command Completion Flag (AR 1407) should be checked to confirm that system command execution has been completed before reading these codes. The response codes are as follows:

Response code	Name	Meaning
0	Normal completion	The system command has been successfully executed.
1	Undefined command error	The system command was executed with an undefined command code.
2	Write-enable error	Memory is either write-protected (i.e., the write enable switch is ON) or the memory is EPROM.
3	Sum check error	The checksum for the Parameter Area was not set when the system command was executed, i.e., a system command with a command code of 1 has not been executed.

System Command Completion Flag

AR 1407 is turned ON to indicate that the system command has completed execution. This Flag is turned ON even if the system command was not successfully executed.

System Command Command Code

A command code is set by the user in AR 1408 to AR 1411 to indicate how the system command is to be executed while the System Command Execution Bit is turned ON. The command code must be written into AR 1408 to AR 1411 using the Hex/BCD Data Change or the Binary Data Change from the Programming Console or other Peripheral Device. A system command will not be executed if the command code is set using any other operation or if set from the program.

The command codes are as follows:

Command code	Name	Meaning
1	Parameter set	The contents of the Parameter Area (DM 0900 to DM 0929) are set into the system, the value of each parameter is checked for validity, all invalid values are replaced with the default values, and a checksum is generated.
2	Parameter backup	The contents of the Parameter Area (DM 0900 to DM 0929) is transferred to the Parameter Backup Area (DM 1900 to 1929), a checksum is generated, the data in the Parameter Backup Area is enabled, and AR 1314 (System Parameter Backup Flag) is turned ON. Because the Parameter Backup Area is contained in the Memory Unit, parameter backup will not be possible if the Memory Unit is write-protected or EPROM.
3	Backup disable	The data contained in the Parameter Backup Area is disabled and AR 1314 (System Parameter Backup Flag) is turned OFF.

Command code	Name	Meaning
4	Parameter clear	All words in the Parameter Area (DM 0900 to DM 0929) are turned OFF (i.e., set to zero).
5	General parameter set	Works in the same way as 01, but only DM 0900 to DM 0905 are set.
6	High-speed counter parameter set	Works in the same way as 01, but only DM 0905 to DM 0919 are set.
7	RS-232C parameter set	Works in the same way as 01, but only DM 0920 to DM 0929 are set.

System Command Execute Bit

AR 1415 is turned ON by the execution of a system command. The system command is actually executed when the Programming Console or other Peripheral Device is used to change the contents of AR 1408 to AR 1414 (System Command Command Code).

To enable system command execution, AR 1415 must be force-set from a Peripheral Device (e.g., using the Force Set/Reset operation from the Programming Console). System command execution will not be enable is AR 1415 is set using any other operation or if set from the program. A system command will also not be executed if the command code is written by any method other than the Hex/BCD Data Change or the Binary Data Change operation from the Programming Console or equivalent operation from another Peripheral Device.

AR 1415 is automatically turned OFF by the system unless force-set from a programming device.

Refreshing

AR 14 is refreshed when a system command is executed.

Executing the System Command

The system command can be executed by manipulating the content of AR 14 with a Programming Device, or by executing SYS(49), the SET SYSTEM instruction.

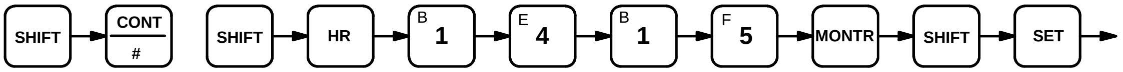
The following examples demonstrate how to execute a system command by manipulating the content of AR 14 with Programming Devices.

Example 1: Programming Console

1, 2, 3...

1. Set the PC to either PROGRAM or MONITOR mode. (Press CLR, MONTR, and turn the mode switch to PROGRAM or MONITOR.)
2. Use the Hexadecimal/BCD Data Modification operation to change any system parameters that need to be changed. (Refer to *7-2-4 Hexadecimal/BCD Data Modification* for details.)
3. Execute the system command to put the new system parameters into effect.

   a) Force-set AR 1415 (System Command Execute Bit).



   b) Write the desired command code into AR 1408 to AR 1411 using the Hex/BCD Data Change or the Binary Data Change operation.

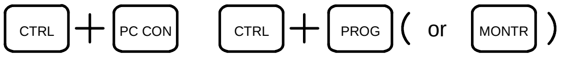
   c) Verify that the content of AR 14 is 8x80 (x is the command code entered in step 3b), indicating a successful execution.

   d) Cancel the force-set status of AR 1415.

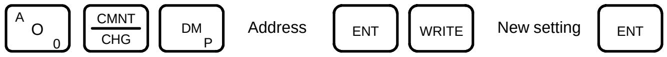
Example 2: GPC

1, 2, 3...

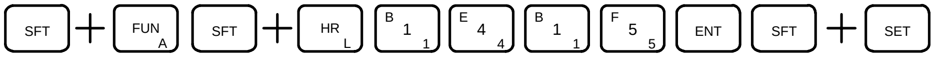
1. Connect the PC to the GPC.
2. Set the PC to either PROGRAM or MONITOR mode.



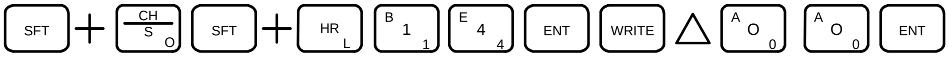
3. Switch to the monitor screen and use the Hexadecimal/BCD Data Modification operation to change any system parameters that need to be changed.



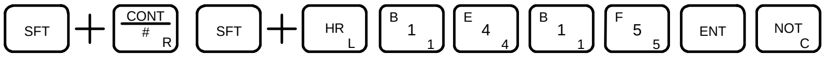
4. Execute the system command to put the new system parameters into effect.
   a) Force-set AR 1415 (System Command Execute Bit).



   b) Write the desired command code into AR 1408 to AR 1411 using the Hex/BCD Data Change or the Binary Data Change operation. Enter the command code (1 to 7) at the Δ in the key sequence below.



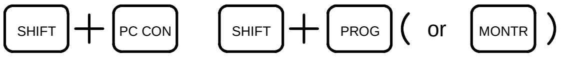
   c) Verify that the content of AR 14 is 8Δ80 (Δ is the command code entered in step 4b), indicating a successful execution.
   d) Cancel the force-set status of AR 1415.



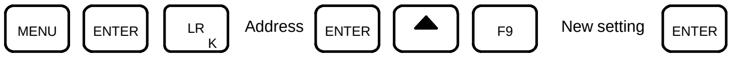
Example 3: FIT

1, 2, 3...

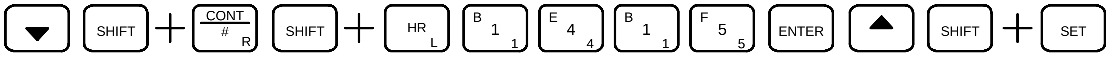
1. In the FIT System Setup, set the PC model to C200H and the PC interface to either Peripheral Interface or Host LInk depending on how the FIT is connected. Switch to the monitor screen. Refer to the FIT *Operation Manual* for details.
2. Connect the PC to the FIT and set the PC to either PROGRAM or MONITOR mode.



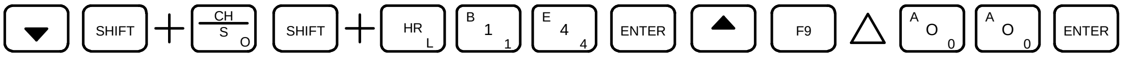
3. Select Data Monitor and use the data modification operation to change any system parameters that need to be changed.



4. Execute the system command to put the new system parameters into effect.
   a) Force-set AR 1415 (System Command Execute Bit).



   b) Write the desired command code into AR 1408 to AR 1411 using the data modification operation. Enter the command code (1 to 7) at the Δ in the key sequence below.



c) Verify that the content of AR 14 is 8∆80 (∆ is the command code entered in step 4b), indicating a successful execution.

d) Cancel the force-set status of AR 1415.

▲ F3

3-5-12 Startup Operating Mode

AR 1500 to AR 1507 contain a code in hexadecimal that indicate the operating mode that will be used when the PC is started if the Preserve Mode code (01) is set in bits 08 through 15 of DM 0900 (backup: DM 1900). These codes are as follows:

00:	PROGRAM mode
01:	MONITOR mode
02:	RUN mode

These bits are refreshed each cycle.

3-5-13 Current Time Area

AR 17 contains the current time in hours and minutes that is used when times are compared with BCMP(68). AR 1700 to AR 1703 contain the minutes; AR 1704 to AR 1707 contain the tens of minutes; AR 1708 to AR 1711 contain the hours; and AR 1712 to AR 1715 contain the tens of hours.

The current time area is available only on the following models:
C__H-C5__-_, C__H-C6__-_, C__H-C7__-_.
Refer to *Section 5 Instruction Set* for programming details.

The Current Time Area is refreshed each cycle while it is operational.

3-5-14 Calendar/Clock Area and Bits

Calendar/clock Area

If AR 2114 (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 (Seconds Round-off Bit) and AR 2115 (Set Bit).

AR word	AR bits	Contents	Possible values
18	00 to 07	Seconds	00 to 99
	08 to 15	Minutes	00 to 59
19	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	00 to 07	Month	1 to 12
	08 to 15	Year	00 to 99 (Rightmost two digits of year)
21	00 to 07	Day of week	00 to 06 (00: Sun.; 01: Mon.; 02: Tues.; 03: Wed.; 04: Thurs.; 05: Fri.; 06: Sat.)

Seconds Round-off 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.

Stop Bit

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

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 (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 (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.

3-5-15 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 either through Programming Console operations or by executing KEY(62).

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 5 Instruction Set* for details on KEY(62) and to *Section 7 Program Debugging and Execution* for details on the TERMINAL mode.

AR 22 is refreshed each cycle.

3-5-16 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 Hex/BCD Change operating from the Programming Console or other Peripheral Device. The Power-off Counter is refreshed every time the power is turned OFF.

3-5-17 SCAN(18) Cycle Time Flag

AR 2405 is turned 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.

3-5-18 Programming Console/Peripheral Interface Unit Mounted Flag

AR 2415 is turned ON to indicate that a Programming Console or Peripheral Interface Unit is mounted to the CPU.

AR 2415 is refreshed each cycle.

3-5-19 FALS-generating Address

AR 25 contains (in 4-digit BCD) the address generating a user-programmed FALS code or a system FALS code 9F (cycle time error). Refer to *5-20-1 FAILURE ALARM – FAL(06) and SEVERE FAILURE ALARM – FALS(07)* for details on FALS codes.

AR 25 is refreshed each cycle while an FALS code is being generated.

3-5-20 Cycle Time Indicators

AR 26 (Maximum Cycle Time Area) contains the maximum cycle time that has occurred since program execution was begun and AR 27 (Current Cycle Time Area) contains the present cycle time. Both of these times are to the tenths of a millisecond in 4-digit BCD.

AR 26 and AR 27 are refreshed each cycle.

3-6 DM (Data Memory) Area

The DM area is divided into various parts as described in the following table.

Addresses	User program read/write	Usage
DM 0000 to DM 0899	Read/write	General User Area
DM 0900 to DM 0929	Read/write	Parameter Area
DM 0930 to DM 0968	---	Not used.
DM 0969 to DM 0999	Read/write	Error History Area
DM 1000 to DM 1899	Read only	General User Area
DM 1900 to DM 1929	Read only	Parameter Backup Area
DM 1930 to DM 1989	---	Not used.
DM 1990 to DM 1999	Read only	User Program Header

Only the General User Areas can be used for general data storage and manipulation. Other DM words are used to control various aspects of PC operation and should never be used for any purposes other than their designed purposes. These words (DM 0900 to DM 0999 and DM 1900 to DM 1999) are known as System DM and are not cleared by the Data Clear operation.

Although composed of 16-bit words like any other data area, all data in any part of the DM area cannot be specified by bit for use in instructions with bit operands, such as LD, OUT, AND, and OR, nor can DM words be used with the Shift instruction.

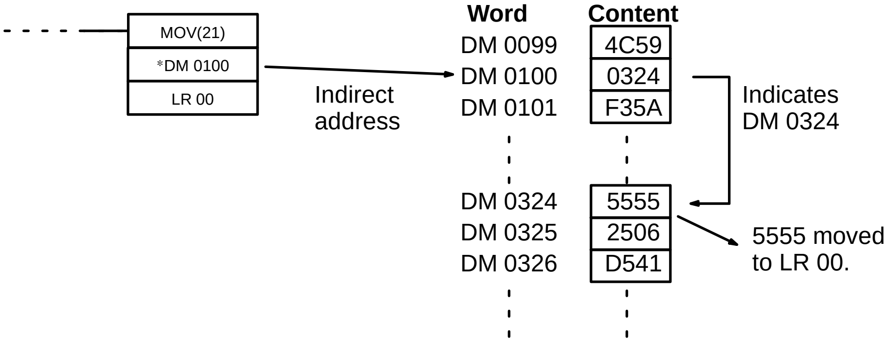
The DM area retains status during power interruptions.

Note When writing System DM to PROM via the GPC or to ROM via the FIT or LSS, use the DM Data Transfer operation to transfer System DM to the Peripheral Device before writing it to the chip.

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.

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, the content of the operand does not contain the actual data to be used. Instead, it contains 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 General User Areas

There are two sections of the DM area available for use in programming.

DM 0000 to DM 0899 and DM 1000 to DM 1899 can be used for manipulation and storage of data. DM 1000 to DM 1899 are in the read-only area, and cannot be written to from the program, i.e., they must be written to from a Programming device.

3-6-2 Parameter and Parameter Backup Areas

The Parameter Area (DM 0900 to DM 0929) and the Parameter Backup Area (DM 1900 to DM 1929) are used together to control various aspects of PC operation, enabling greater system flexibility.

The Parameter Backup Area is contained in the Memory Unit and can be written only by using the system command in the AR area. Also, to use the system command to write data into the Parameter Backup Area, the Memory Unit must be writeable, i.e., not write-protected and not EPROM.

System Command

The system command executes the 7 basic operations that control the Parameter Area. There are two ways to execute the system command:

1, 2, 3...

1. Manipulating the content of AR 14 with a Programming Device. Refer to *3-5-11 System Command Bits* for details.
2. Executing SYS(49), the SET SYSTEM instruction. Refer to *5-20-5 SET SYSTEM – SYS(49)* for details.

Basic Operation

The Parameter Backup Area is allocated exactly like the Parameter Area and is used to back up Parameter Area settings. The procedure for using these areas is as follows:

1, 2, 3...

1. Data is first set into the Parameter Area from a Peripheral Device or via programming instructions.
2. The system command is used to make these new settings effective (i.e., write them into the system) and generate a checksum for them. The checksum is used to check the accuracy of data in later operations. Each value is checked when it is set into the system and if an invalid value is discovered, the default value is used.

3. The system command is next used to transfer Parameter Area settings to the Parameter Backup Area and make it the valid area.
4. To change parameters once they have been set into the Parameter Backup Area, the Parameter Backup Area is disabled using the system command and then settings in the Parameter are changed and the operation is repeated.

The system command can also be used to clear the contents of the Parameter Area.

Caution Input data into the Parameter Area carefully. If parameter settings are incorrect, the PC may not operate correctly. If a setting is found to be invalid, the default setting is used.

Checksum

Whenever the system command is used to enable data in the Parameter Area or transfer data to the Parameter Backup Area, a checksum is generated. This checksum is a numeric value computed from the contents of the area and copied along with it. The copy (or the original file) can then be checked later to see if the same value (i.e., the checksum) results from the same computations. If the checksum disagrees with the previous one, it is assumed that data has changed and is no longer valid.

The checksums for the Parameter and Parameter Backup Areas are checked whenever new data is set, including data set at startup. If a checksum is found to be incorrect, the data is not used and alternate measures are taken (see next section for details).

Operation On Startup

When the PC is started, the parameters must be reset into the system. If the Parameter Backup Area is the valid area and it’s checksum is okay, its contents are transferred to the Parameter Area and from there it is set into the system.

If the Parameter Backup Area is not valid or if its checksum is incorrect, the data in the Parameter Area is set into the system, assuming its checksum is okay.

If the checksum for the Parameter Backup Area is incorrect, the default values for all settings in the Parameter Area are set into the system and an 9F FAL number is output and AR 1315 (Parameter/Backup Area Checksum Flag) is turned ON.

When parameters are finally set into the system, they are each checked for validity and default parameters are set for any invalid ones regardless of the method used to set parameters.

Parameter and Parameter Backup Area Allocations

The following table shows the allocations of words and bits in the Parameter and Parameter Backup Areas. The first DM address is the Parameter Area address. The contents of these words can be changed either via programming instructions or via a Peripheral Device (e.g., Programming Console). The second DM address (given in parenthesis) is the Parameter Backup Area address. The contents of these words can be updated by executing the system command with a command code to backup parameters (02).

The following table is designed to first provide the name and sometimes a basic description of the function of each, and then, if necessary, to detail the operation of each bit. The top line of the header thus applies to the first (and sometimes only) line of the table for each DM address. The next line applies to the remaining lines for each DM address. (Here, "address" applies to the 4-bit word address without the bit number, which is not considered part of a DM address.)

Address	Name/Description		Default
	Bit	Content/Meaning	
DM 0900 (DM 1900)	**Operating Mode on Startup** Bits 00 to 07 designate the PC mode on startup if bits 08 to 15 are set to 02. If bits 08 to 15 are set to 00, the PC startup mode will be designated by the key switch on the Programming Console. If bits 08 to 15 are set to 01, the PC startup mode will be the same mode as it was when the PC was last turned off.		Key switch
	00 to 07	00: PROGRAM 01: MONITOR 02: RUN	
	08 to 15	00: As set on Programming Console key switch 01: Mode when PC was last turned off (in AR 15) 02: Mode set in bits 00 to 07, above	
DM 0901 (DM 1901)	**Cycle Time Limit** A cycle time limit can be set in bits 00 to 07 that will be valid if bits 08 to 10 are set to 01. If bits 08 to 15 are set to 00, the cycle time limit will be 100 ms.		100 ms
	00 to 07	Cycle time limit in units of ten milliseconds. Setting is between 00 and 99 in BCD resulting in cycle time limits between 000 and 990 ms, respectively.	
	08 to 15	00: Bits 00 to 07 disabled (i.e., cycle time limit is 100 ms) 01: Bits 00 to 07 enabled	
DM 0902 (DM 1902)	**Peripheral Device Service Time** The percent of the cycle time allocated to servicing the Programming Console and other Peripheral Devices connected to the CPU can be set in bits 00 to 07. This value will be valid if bits 08 to 10 are set to 01. If bits 08 to 15 are set to 00, the service time will be 5%. The Device will be serviced for at least 1 ms regardless of the cycle time and this setting.		5%
	00 to 07	Percent of cycle time allocated to Device servicing between 00 and 99 in BCD.	
	08 to 15	00: Bits 00 to 07 disabled (i.e., servicing set to 5%) 01: Bits 00 to 07 enabled	
DM 0903 (DM 1903)	**RS-232C Interface Service Time** The percent of the cycle time allocated to servicing the RS-232C interface can be set in bits 00 to 07. This value will be valid if bits 08 to 10 are set to 01. If bits 08 to 15 are set to 00, the service time will be set to 5%. The RS-232C Interface will be serviced for at least 1 ms regardless of the cycle time and this setting.		5%
	00 to 07	Percent of cycle time allocated to RS-232C interface servicing between 00 and 99 in BCD.	
	08 to 15	00: Bits 00 to 07 disabled (i.e., servicing set to 5%) 01: Bits 00 to 07 enabled	
DM 0904 (DM 1904)	**Programming Console Message Language Bits** Bits 00 to 07 are not used. Bits 08 to 15 determine the language displayed on the Programming Console.		English
	08 to 15	00: English 01: Japanese 02: German 03: French 04: Italian 05: Spanish	

Address	Name/Description		Default
	Bit	Content/Meaning	
DM 0905 (DM 1905)	**General High-speed Counter Bits** Bits 00 to 07 are the Enable Bits for high-speed counter interrupt outputs. An interrupt output is enabled if it Enable Bit is ON. Bits 08 through 15 are various controls for the high-speed counter.		No counter
	00	IR 00200 Enable Bit	
	01	IR 00201 Enable Bit	
	02	IR 00202 Enable Bit	
	03	IR 00203 Enable Bit	
	04	IR 00204 Enable Bit	
	05	IR 00205 Enable Bit	
	06	IR 00206 Enable Bit	
	07	IR 00207 Enable Bit	
	08 to 10	Final step for bank 0 (0 to 7)	
	11 to 13	Final step for bank 1 (0 to 7)	
	14	Hard Reset Enable Bit (Turn ON to enable hard reset.)	
	15	High-speed Counter Enable Bit (Turn ON to enable counter.)	
DM 0906 (DM 1906)	**High-speed Counter Interrupt Output Table** The bits in this and the following seven DM words are used to set the pattern that will be output when the high-speed counter is set for operation according to this output table. Each set of eight bits determines the status that will be output to IR 00200 to IR 00207 for each step of interrupt outputs. Refer to *3-8 TC (Timer/Counter) Area* for details on the high-speed counter.		---
	00	Step 0 output status for IR 00200	
	01	Step 0 output status for IR 00201	
	02	Step 0 output status for IR 00202	
	03	Step 0 output status for IR 00203	
	04	Step 0 output status for IR 00204	
	05	Step 0 output status for IR 00205	
	06	Step 0 output status for IR 00206	
	07	Step 0 output status for IR 00207	
	08	Step 1 output status for IR 00200	
	09	Step 1 output status for IR 00201	
	10	Step 1 output status for IR 00202	
	11	Step 1 output status for IR 00203	
	12	Step 1 output status for IR 00204	
	13	Step 1 output status for IR 00205	
	14	Step 1 output status for IR 00206	
	15	Step 1 output status for IR 00207	

Address	Name/Description		Default
	Bit	Content/Meaning	
DM 0907 (DM 1907)	**High-speed Counter Interrupt Output Table (continued)** This word continues the table started in DM 0906.		---
	00	Step 2 output status for IR 00200	
	01	Step 2 output status for IR 00201	
	02	Step 2 output status for IR 00202	
	03	Step 2 output status for IR 00203	
	04	Step 2 output status for IR 00204	
	05	Step 2 output status for IR 00205	
	06	Step 2 output status for IR 00206	
	07	Step 2 output status for IR 00207	
	08	Step 3 output status for IR 00200	
	09	Step 3 output status for IR 00201	
	10	Step 3 output status for IR 00202	
	11	Step 3 output status for IR 00203	
	12	Step 3 output status for IR 00204	
	13	Step 3 output status for IR 00205	
	14	Step 3 output status for IR 00206	
	15	Step 3 output status for IR 00207	
DM 0908 (DM 1908)	**High-speed Counter Interrupt Output Table (continued)** This word continues the table started in DM 0906.		---
	00	Step 4 output status for IR 00200	
	01	Step 4 output status for IR 00201	
	02	Step 4 output status for IR 00202	
	03	Step 4 output status for IR 00203	
	04	Step 4 output status for IR 00204	
	05	Step 4 output status for IR 00205	
	06	Step 4 output status for IR 00206	
	07	Step 4 output status for IR 00207	
	08	Step 5 output status for IR 00200	
	09	Step 5 output status for IR 00201	
	10	Step 5 output status for IR 00202	
	11	Step 5 output status for IR 00203	
	12	Step 5 output status for IR 00204	
	13	Step 5 output status for IR 00205	
	14	Step 5 output status for IR 00206	
	15	Step 5 output status for IR 00207	

Address	Name/Description		Default
	Bit	Content/Meaning	
DM 0909 (DM 1909)3	**High-speed Counter Interrupt Output Table (continued)** This word continues the table started in DM 0906.		---
	00	Step 6 output status for IR 00200	
	01	Step 6 output status for IR 00201	
	02	Step 6 output status for IR 00202	
	03	Step 6 output status for IR 00203	
	04	Step 6 output status for IR 00204	
	05	Step 6 output status for IR 00205	
	06	Step 6 output status for IR 00206	
	07	Step 6 output status for IR 00207	
	08	Step 7 output status for IR 00200	
	09	Step 7 output status for IR 00201	
	10	Step 7 output status for IR 00202	
	11	Step 7 output status for IR 00203	
	12	Step 7 output status for IR 00204	
	13	Step 7 output status for IR 00205	
	14	Step 7 output status for IR 00206	
	15	Step 7 output status for IR 00207	
DM 0910 to DM 0917 (DM 1910 to DM 1917)	**High-speed Counter Step Table** These words contain the width of each step of the counter. The total width of the counter is the sum of all the widths set below. As shown below, the rightmost digit of the DM address is the same as the corresponding step. Step 0: DM 0910 Step 1: DM 0911 Step 2: DM 0912 Step 3: DM 0913 Step 4: DM 0914 Step 5: DM 0915 Step 6: DM 0916 Step 7: DM 0917 Set the rightmost two digits to 11 or higher (although 00 is O.K.) to prevent the poor responsiveness to high-frequency input signals. For example, set values of 0010 and 5001 would risk inaccurate counting, while set values of 2011 and 3000 would not.		0000 (10000) when the counter is enabled.
DM 0918 and DM 0919 (DM 1918 and DM 1919)	Not used.		---
DM 0920 to DM 0926 (DM 1920 to DM 1926)	**RS-232C Interface Settings** These words are used to set various aspects of communications through the RS-232C interface. DM 0920 bits 00 to 07 are used to select standard or custom communications settings. If these bits are set to 01, the settings in DM 0921 are used.		See below.
DM 0920 (DM 1920)	00 to 07	**Standard/Custom Communications Format Selection** 00: Standard (1 start bit, 7-bit data length, even parity, 2 stop bits, 9,600 baud) 01: Custom settings (i.e., according to contents of DM 0921)	Standard
	08 to 15	**RS-232C Mode** 00: Host link 01: Memory upload/download 02: ASCII I/O mode	Host link

Address	Name/Description		Default
	Bit	Content/Meaning	
DM 0921 (DM 1921)	00 to 07	**Baud Rate (if DM 0920 bits 00 to 07 are 01)** 00: 300 bps 01: 600 bps 02: 1,200 bps 03: 2,400 bps 04: 4,800 bps* 05: 9,600 bps*	9600 bps
	08 to 15	**Data Format (if DM 0920 bits 00 to 07 are 01)** 00: 1 start bit, 7-bit data, 2 stop bits, even parity 01: 1 start bit, 7-bit data, 2 stop bits, odd parity 02: 1 start bit, 8-bit data, 1 stop bits, no parity 03: 1 start bit, 8-bit data, 2 stop bits, no parity 04: 1 start bit, 8-bit data, 1 stop bits, even parity 05: 1 start bit, 8-bit data, 1 stop bits, odd parity	1 start bit, 7-bit data, 2 stop bits, even parity
DM 0922 (DM 1922)	00 to 07	**Transmission Delay** In tenths of milliseconds between 00 and 99 (BCD, correspond to 000 and 990 ms delays, respectively)	0 ms
	08 to 15	**RTS/CTS Control** 00: Without RTS/CTS 01: With RTS/CTS	Without RTS/CTS
DM 0923 (DM 1923)	00 to 07	Not used.	---
	08 to 15	Unit number for host link mode between 00 and 31 in BCD	#0
DM 0924 (DM 1924)	00 to 07	Not used.	---
	08 to 15	**Transmission Format for Memory Upload/Download** 00: Intel HEX 01: Motorola S-Record	Intel HEX
DM 0925 (DM 1925)	00 to 07	**Starting Code** This byte contains the starting code used in POUT(63) and PIN(64) transmissions when bits 08 to 15 contain 01.	No starting code
	08 to 15	**Starting Code/No Starting Code Selection** 00: No starting code 01: Starting code set in bits 00 to 07	
DM 0926 (DM 1926)	00 to 07	**End Code** This byte contains the end code used in POUT(63) and PIN(64) transmissions when bits 08 to 15 contain 01.	No end code
	08 to 15	**End Code/No End Code Selection** 00: No end code 01: End code set in bits 00 to 07	
DM 0927 to DM 0929 (DM 1927 to DM 1929)	**Not used.**		---

*Higher baud rates may produce errors in RS-232C communications if both RS-232C interface and Peripheral Interface Unit are used.

3-6-3 Error History Area

DM 0969 to DM 0999 are used to store up to 10 records that show the nature, time, and date of errors that have occurred in the PC. The time and date entries in these records are only recorded in PCs that are equipped with the calendar/clock function.

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 9 Troubleshooting* for details on error codes.

Area Structure

Error records occupy three words each stored between DM 0970 and DM 0999. The last record that was stored can be obtained via the content of DM 0969 (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 0970 to DM 0972	0001
2	DM 0973 to DM 0975	0002
3	DM 0976 to DM 0978	0003
4	DM 0979 to DM 0981	0004
5	DM 0982 to DM 0984	0005
6	DM 0985 to DM 0987	0006
7	DM 0988 to DM 0990	0007
8	DM 0991 to DM 0993	0008
9	DM 0994 to DM 0996	0009
10	DM 0997 to DM 0999	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 08 (fatal)
Second	00 to 07	Seconds
	08 to 15	Minutes
Third	00 to 07	Hours
	08 to 15	Day of month

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 User Program Header Area

DM 1990 to DM 1999 can be used to record the name, version, and creation/ alteration data of the user program to aid in program management. This information (except for the seconds) can also be displayed on the Programming Console by pressing CLR, SFT, and MONTR (see *Section 7 Program Debugging and Execution* for details).

The creation date will not be updated if programs are transferred into the PC from other than the C20H/C28H/C40H/C60H. It will be updated only if instructions are written (not transferred), deleted, or inserted from a Programming Device (e.g., Programming Console, GPC, or FIT). If the CPU is equipped with the Calendar/clock Area, the creation date will be updated whenever instructions are written, deleted, or inserted, or when timer/counter SVs are set.

If non-C20H/C28H/C40H/C60H programs are transferred into the PC, the Programming Console or other Peripheral Device must be used to artificially write data to the header area.

The User Program Header Area is structured as shown below.

Address*	Bits	Contents
DM 1990	00 to 07	Program version number in BCD without the decimal
	08 to 15	Name/version Enable Bit (Set to 5A to enable the name and version. These will not be displayed on the Programming Console unless enabled. Any other value will disable the name and version.)
DM 1991 to DM 1994	00 to 15	Program name in ASCII, eight characters. Characters are displayed on the Programming Console in order from DM 1991 to DM 1994, with the leftmost ASCII character in each word (bits 08 to 15) displayed to the left of the rightmost (bits 00 to 07).
DM 1995	00 to 07	Seconds of creation date
	08 to 15	Minutes of creation date
DM 1996	00 to 07	Hour of creation date
	08 to 15	Day of month of creation date
DM 1997	00 to 07	Month of creation date
	08 to 15	Year of creation date
DM 1995 to DM 1999	Not used.	

Note *Use the hexadecimal addresses shown below when using a commercially available PROM writer to read data from a User Program Header Area that has been stored in ROM.

DM 1990 (bits 08 to 15): 07BC
DM 1990 (bits 00 to 07): 07BD
. . .
. . .
. . .
DM 1997 (bits 08 to 15): 07CA
DM 1997 (bits 00 to 07): 07CB

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 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 Entering Programs* and *Section 5 Instruction Set*.

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(15), CNT, CNTR(12), RDM(60) or HDM(61). 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 *7-2 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 *5-7 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.

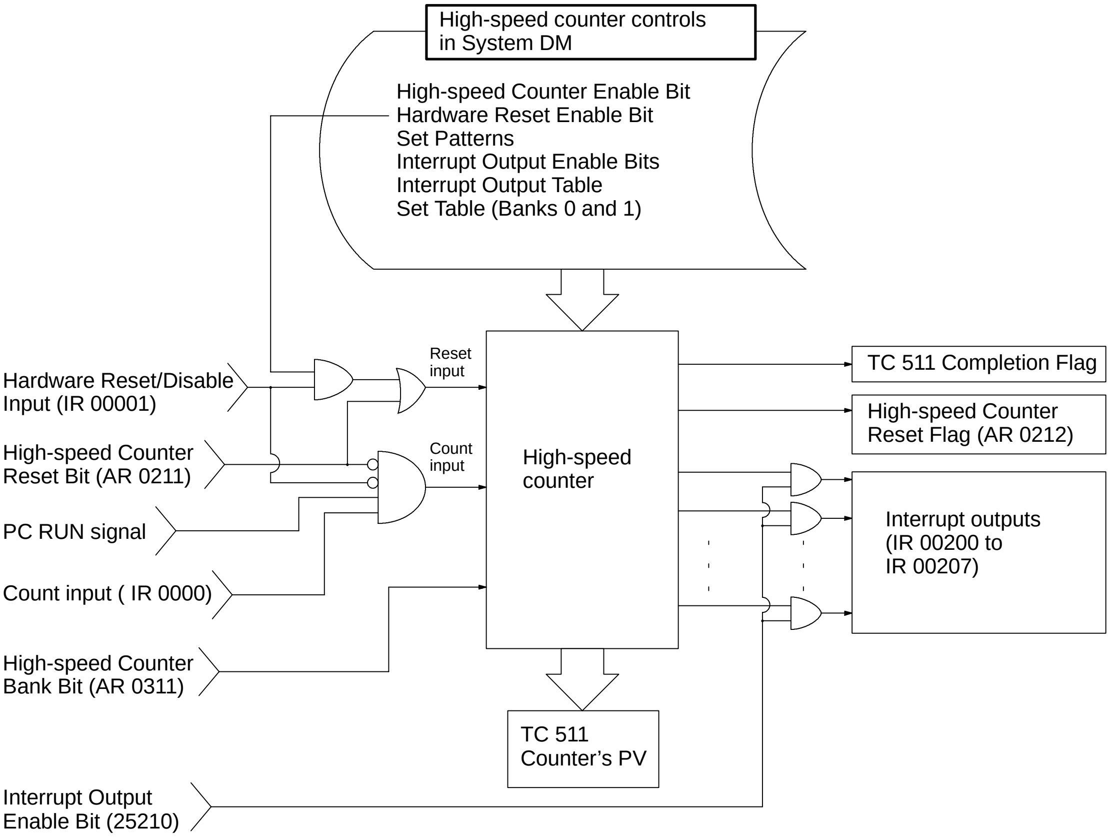
3-8-1 High-speed Counter

The C20H/C28H/C40H/C60H are equipped with a built-in high-speed counter that can be activated either through use of the interrupt drum output function or through the HIGH-SPEED DRUM COUNTER instruction (HDM(61)). The high-speed counter is allocated to TC 511. Even if the interrupt drum output function is used, TC 511 can be used like any other TC bit, i.e., it can be used to access the Completion Flag and PV for the counter. The PV for the high-speed counter, however, is always reset to 0000 when power to the PC is shut off.

The high-speed counter is an incremental ring counter that can be set for between 1 and 80,000 counts per revolution. All counter settings are in the Parameter and Parameter Backup Area of System DM. These settings include a High-speed Counter Enable Bit, a Hardware Reset Enable Bit, Interrupt Output Enable Bits, an Interrupt Output Table, and a Step Table. These are described below and also listed in tabular form in *3-6 DM (Data Memory) Area*.

There are also various controls for the high-speed counter in the AR area, including a High-speed Counter Reset Bit, a High-speed Counter Reset Flag, and a High-speed Counter Bank Bit.

The overall operation of the high-speed counter is as shown below.



High-speed counter controls in System DM
High-speed Counter Enable Bit
Hardware Reset Enable Bit
Set Patterns
Interrupt Output Enable Bits
Interrupt Output Table
Set Table (Banks 0 and 1)
Hardware Reset/Disable Input (IR 00001)
High-speed Counter Reset Bit (AR 0211)
PC RUN signal
Count input (IR 0000)
High-speed Counter Bank Bit (AR 0311)
Interrupt Output Enable Bit (25210)
Reset input
Count input
High-speed counter
TC 511 Completion Flag
High-speed Counter Reset Flag (AR 0212)
Interrupt outputs (IR 00200 to IR 00207)
TC 511 Counter's PV

Counter Controls and I/O

The high-speed counter is controlled and operates through the following IR and AR bits. The remaining controls for the high-speed counter are in the DM area and are described in *3-8-2 System DM High-speed Counter Parameters*.

Counter Input

The high-speed counter will be incremented on the rising edge of IR 00000 as long as the hardware reset/disable input (IR 00001) and the High-speed Counter Reset Bit (AR 0211) are both OFF and the PC is in either MONITOR or RUN mode. Inputs are not counted in PROGRAM mode.

The pulse width for the counter input must be at least 250 ms (2 kHz with a 1:1 duty rate).

Hardware Reset/Disable

IR 00001 (on the CPU) can be used either as a hardware reset and a counter disable, or it can be used as a counter disable only. When IR 0001 is used as a hard reset, AR 0212 will be turned ON for one cycle.

The Hardware Reset/Disable must remain ON for at least 250 ms. The reset is serviced immediately, i.e., without being delayed by the cycle time.

High-speed Counter Reset Bit

The high-speed counter can also be reset by turning on AR 0211. The high-speed counter will be reset when either this bit, IR 00001, or both are ON.

The High-speed Counter Reset Bit is not serviced until after the END(01) instruction is executed, i.e., servicing is delayed by the cycle time.

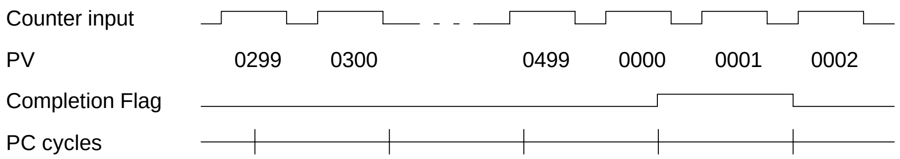
High-speed Counter Reset Flag

AR 0212 will turn ON for one cycle when IR 00001 has come ON to reset the high-speed counter.

CNT 511 Completion Flag

The completion flag of CNT will turn ON for one cycle each time a set value for a step has been reached. This flag can be used to activate a carry or as a normal counter completion output.

The following figure shows the relationship between the counter input, the present value, the completion flag, and the PC cycle. This figure assumes that only step 0 has been set and that it is set to 500.



High-speed Counter Bank Bit

AR 0311 is used with the interrupt output function. Eight steps each can be recorded into two banks and then the status of AR 0311 can be used to control which bank is currently being used.

If AR 0311 is ON, bank 1 is used; if it is OFF, bank 0 is used. If the bank is changed during counter operation, the present value and the set value will be reset. When switching from bank 1 to bank 0, counting will start from zero in step 0. When switching from bank 0 to bank 1, counting will start from zero in the step designated by the Beginning Step for Bank 1 setting.

Interrupt Outputs

IR 00200 through 00207 can be used with the high-speed counter as interrupt drum outputs. Output tables can be set for these bits by step. These outputs are refreshed on an interrupt basis, i.e., they are not affected by the cycle time. Refer to *3-8-3 Interrupt Drum Outputs* for details. SR 25210 (Interrupt Output Enable Bit) must be ON to use these outputs.

3-8-2 System DM High-speed Counter Parameters

System DM contain several parameters that can be used to control high-speed counter operations. Specific bit allocations are provided in *3-6 DM (Data Memory) Area*. This section explains operational aspects that are affected by these parameters.

Interrupt Output Enable Bits

Each of bits 00 to 07 of DM 0905 (DM 1905) correspond to one of the interrupt drum outputs for the high-speed counter. If the bit corresponding to an interrupt output is ON, that output will be refreshed in every step according to the output tables.

Bank Range Settings

Bits 08 to 10 of DM 0905 (DM 1905) are used to specify the final step for bank 0 between 0 and 7. Bank 0 will thus extend from step 0 to the step set here. Bits 11 to 13 are the beginning step for bank 1 and can be set between 0 and 7. Bank 1 will thus range from the step set here to set 7.

The steps for bank 0 and bank 1 may overlap, e.g., bank 0 may range from step 0 through step 5 at the same time that bank 1 ranges from step 2 to step 7.

Hard Reset Enable Bit

Bit 14 of DM 0905 (DM 1905) controls the operation of the hardware reset/disable (IR 00001). If bit 14 is turned ON, IR 00001 will function to both disable and reset the high-speed counter. If bit 14 is OFF, IR 00001 will function only to disable the high-speed counter; it will not reset it.

High-speed Counter Enable Bit

Bit 15 of DM 0905 (DM 1905) is turned ON to enable the high-speed counter and turned OFF to disable it.

Interrupt Output Table

The bits in DM 0906 through DM 0909 (DM 1906 through DM 1909) are used to set the patterns that will be output when the interrupt drum outputs are used. Each set of eight bits determines the status that will be output to IR 00200 to IR 00207 for each step of interrupt outputs. Refer to *3-8-3 Interrupt Drum Outputs* for details.

High-speed Counter Step Table

DM 0910 to DM 0917 (DM 1910 to DM 1917) contain the width settings (counts per revolution) for each step of the counter. There are a total of eight steps, each of which can be set to between 1 and 10,000 counts per revolution (a setting of 0000 is equivalent to 10,000). Data must be in BCD. The total width of the counter is the sum of all the step widths being used in the active bank. As shown below, the rightmost digit of the DM address is the same as the corresponding step.

Step 0 width:	DM 0910
Step 1 width:	DM 0911
Step 2 width:	DM 0912
Step 3 width:	DM 0913
Step 4 width:	DM 0914
Step 5 width:	DM 0915
Step 6 width:	DM 0916
Step 7 width:	DM 0917

The current step will be reset when PC power it turned ON, when the high-speed counter is reset, and when the bank designation is changed. In bank 0, the current step is always reset to step 0; in bank 1, to the step designated in the Beginning Step for Bank 1.

When the interrupt outputs are not used, the counter width can still be set to a value greater than 10,000 by setting widths for multiple step. The total width will be the sum of all width settings, producing a maximum of 80,000 counts per revolution.

Note The rightmost two digits of the width settings must be 11 or higher (although 00 is O.K.) to prevent the poor responsiveness to high-frequency input signals. For example, set values of 0010 and 5001 would risk inaccurate counting, while set values of 2011 and 3000 would not.

3-8-3 Interrupt Drum Outputs

The high-speed counter can be used to activate interrupt outputs (IR 00200 through IR 00207) according to patterns set in output tables for each of up to eight steps. These outputs are refreshed on an interrupt basis, i.e., they are not delayed by the cycle time. SR 25210 (Interrupt Output Enable Bit) must be ON to use the interrupt outputs.

The interrupt drum outputs can be used to control cam switches and similar devices according to inputs received from incremental rotary recorders.

Interrupt Output Response Time

Although the interrupt outputs are not delayed by the cycle time, there is delay of up to 1.5 ms from the time that the counter reaches its set value. There is an additional delay of up to 2 ms when a Peripheral Interface Unit is being used.

Interrupt Output Bits

IR 00200 through IR 00207 are used as the interrupt outputs. These bits are available for use as normal outputs when the high-speed counter is not being used. Note: these bits cannot be output directly from the program when they are enabled as the interrupt outputs.

When not using IR 00200 through IR 00207 as interrupt outputs, be sure to disable them by turning off the corresponding Interrupt Output Enable Bits.

Output Steps and Banks

There are eight steps available that can be set up in two banks to control the interrupt outputs. Each bank can be from 1 to 8 steps in length and can control anywhere from 1 to 8 of the interrupt outputs. The settings to control these parameters are described in *3-8-2 System DM High-speed Counter Parameters*.

When counter operation begins, all enabled interrupt outputs are refreshed according to the pattern set in the output table for the first step (step 0 if bank 0 is being used; the designated beginning step if bank 1 is being used).

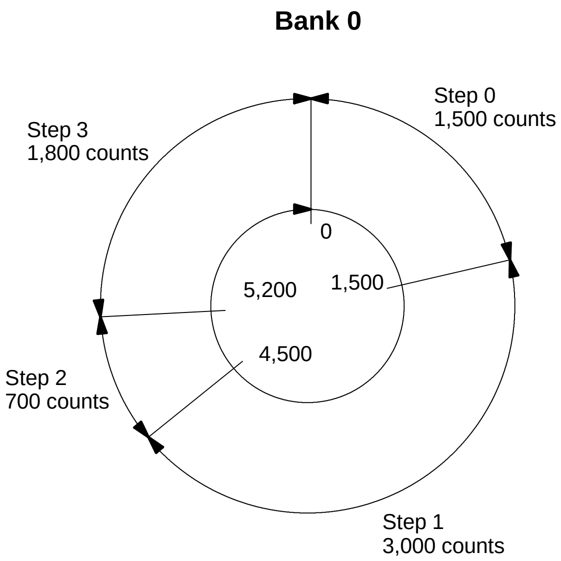
When the counter reaches the value set for the first step, the enabled interrupt outputs are again refreshed, this time according to the output table patterns set for the second step. This operation continues until the count for the last step has been reached (the designated last step for bank 0 or step 7 for bank 1), at which time the count is reset and operation repeats from the first step.

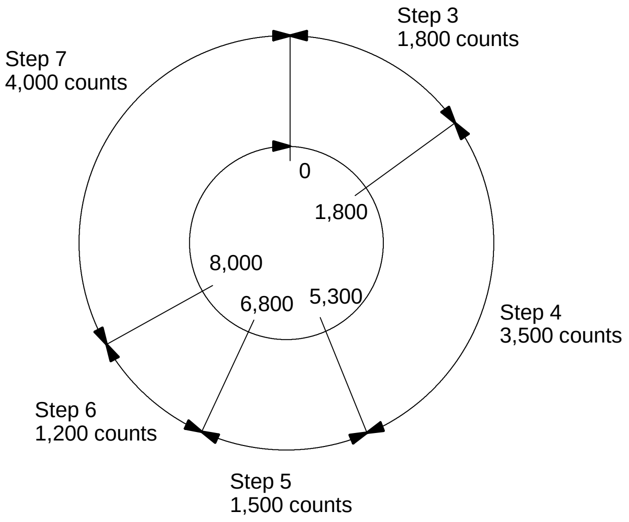
Example

This example describes operation when the End Step for Bank 0 is set to 3, the Beginning Step for Bank 1 is set to 4, all interrupt outputs are enabled, and the step widths and output table are set as shown below.

Bank	Step	Width	Output patterns							
			00200	00201	00202	00203	00204	00205	00206	00207
0	0	1,500	1	0	0	0	0	0	0	0
0	1	3,000	0	1	0	0	0	0	0	0
0	2	700	1	0	1	0	0	0	0	0
0 and 1	3	1,800	0	0	0	1	1	0	0	0
1	4	3,500	0	0	0	0	1	0	0	1
1	5	1,500	0	0	0	0	0	1	0	0
1	6	1,200	0	0	0	0	0	1	1	0
1	7	4,000	0	0	0	0	0	0	1	1

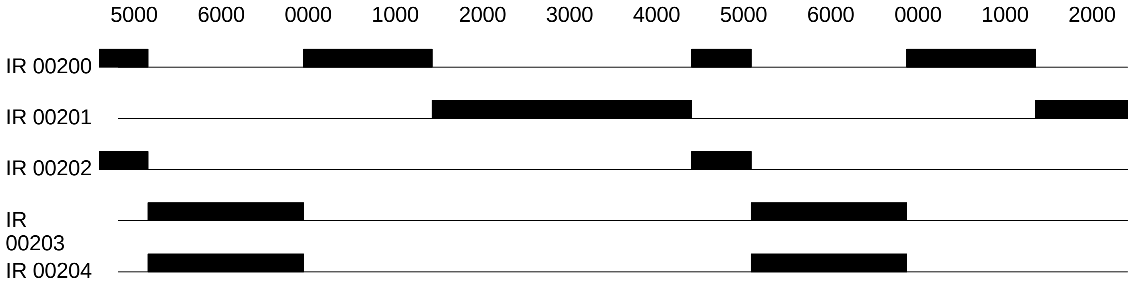
The above table shows that bank 0 runs from step 0 through step 3 and has a total width of 7,000 and that bank 1 runs from step 3 through step 7 and has a total width of 12,000. In drum form, the operation of bank 0 and bank 1 appears as follows:



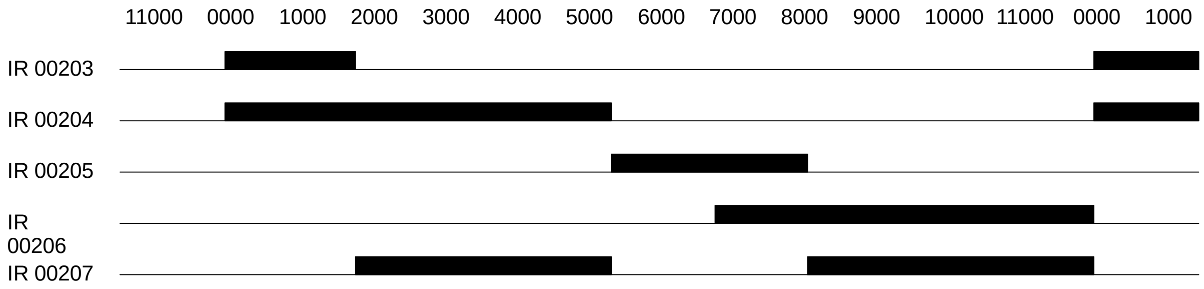


In terms of output timing, operation would appear as shown below. Only the outputs that are set to turn ON during operation of each bank are shown.

Bank 0



Bank 1



3-9 LR (Link Relay) Area

The LR area is used as a common data area by PCs that support 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 PC Link 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. In PCs that do not support a PC Link System, the LR area is available for 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 *5-7 INTERLOCK and INTERLOCK CLEAR – IL(02) and ILC(03)* for details on interlocks.

3-10 Program Memory

Program Memory is where the user program is stored. Memory Units come in different types, i.e., RAM, EEPROM, and ROM Units. There are different sizes for ROM. (Refer to the *Installation Guide* for details.)

The amount of Program Memory available is 2878 words, regardless of the type of chip used for ROM. The RAM and EEPROM chips are already installed and ready for use; the ROM chip must be purchased separately.

For every memory chip, 1218 words are used for the DM area and other purposes. The remainder of the chip’s capacity is available for the Program Memory. If a 27128 or 27256 EPROM chip is installed, only the first 4K of memory is usable. Refer to the memory chip table in *Appendix A Standard Models* for details.

To store instructions in Program Memory, input the instructions through the Programming Console, or download programming data from a FIT, floppy disk, cassette tape, or host computer. Refer to the end of *Appendix A Standard Products* for information on FIT and other special products. Programming Console operations, including those for program input, are described in *Section 7 Program Debugging and Execution*.

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 Entering Programs*.

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 TR bit is not used twice in the same instruction block.

SECTION 4
Writing and Entering Programs

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.
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 *5-10 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. Back up the program.

The basics of ladder-diagram programming and conversion to mnemonic code are described in *4-3 Basic Ladder Diagrams*. Preparing for and inputting the program via the Programming Console are described in *4-4 The Programming Console* through *4-6 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 Debugging and Execution*. *Section 9 Troubleshooting* also provides information required for debugging.

4-2 Instruction Terminology

There are basically two types of instructions used in ladder-diagram programming:

1, 2, 3...

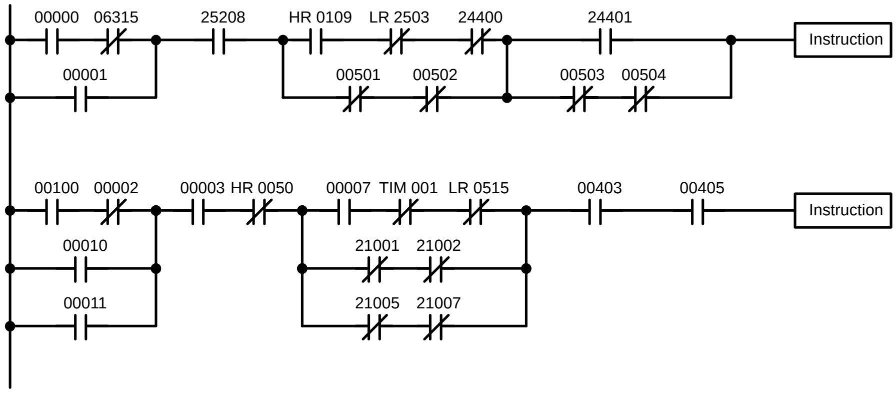
1. Instructions that correspond to the conditions on the ladder diagram and are used in instruction form only when converting a program to mnemonic code.
2. Instructions that are used on the right side of the ladder diagram and are executed according to the conditions on the instruction lines leading 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 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.



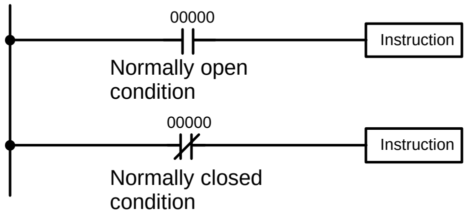
As shown in the diagram above, instruction lines can branch apart and they can join back together. The vertical pairs of lines are called conditions. Conditions without diagonal lines through them are called normally open conditions and correspond to a LOAD, AND, or OR instruction. The conditions with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

4-3-1 Basic Terms

Normally Open and Normally Closed Conditions

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



Instruction is executed when IR 00000 is ON.

Instruction is executed when IR 00000 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 *4-6-7 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.

4-3-2 Mnemonic Code

The ladder diagram cannot be directly input into the PC via a Programming Console; a GPC, a FIT, or 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 input 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 a GPC, a FIT, or LSS (although you can use it with these devices 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 additional data lines, the left two columns are left blank. If the instruction requires no definer or bit operand, the operand column is left blank for first line. It is a good idea to cross through any blank data column spaces (for all instruction words that do not require data) so that the data column can be quickly scanned to see if any addresses have been left out.

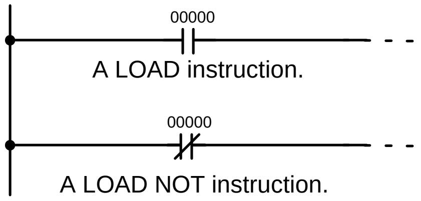
When programming, addresses are automatically displayed and do not have to be input unless for some reason a different location is desired for the instruction. When converting to mnemonic code, it is best to start at Program Memory address 00000 unless there is a specific reason for starting elsewhere.

4-3-3 Ladder Instructions

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

LOAD and LOAD NOT

3.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 instruction described later in this manual.

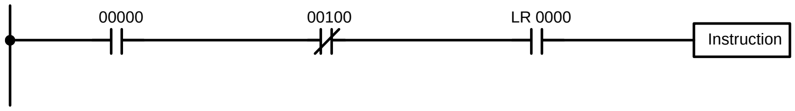


Address	Instruction	Operands
00000	LD	00000
00001	Instruction	
00002	LD NOT	00000
00003	Instruction	

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 would be ON when IR 00000 was ON; for the LOAD NOT instruction (i.e., a normally closed condition), it would be ON when IR 00000 was OFF.

AND and AND NOT

4.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, 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	

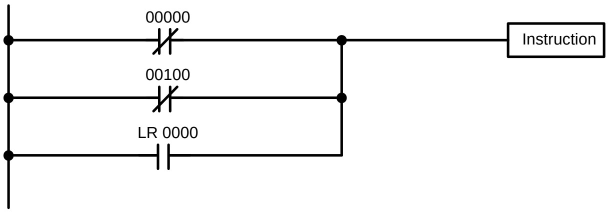
The instruction would have an ON execution condition only when all three conditions are ON, i.e., when IR 00000 was ON, IR 00100 was OFF, and LR 0000 was ON.

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 a series would take the logical AND between its execution condition and the inverse of its operand bit.

OR and OR NOT

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



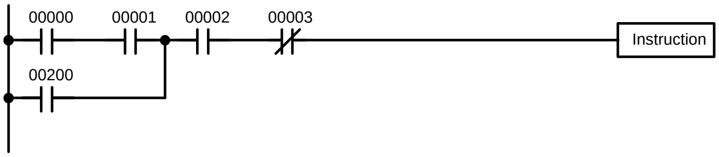
Address	Instruction	Operands
00000	LD	00000
00001	OR NOT	00100
00002	OR	LR 0000
00003	Instruction	

The instruction would have an ON execution condition when any one of the three conditions was ON, i.e., when IR 00000 was OFF, when IR 00100 was OFF, or when LR 0000 was 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 would be produced for the next instruction.

Combining AND and OR Instructions

When AND and OR instructions are combined in more complicated diagrams, they can sometimes be considered individually, with each instruction performing a logic operation on the 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.



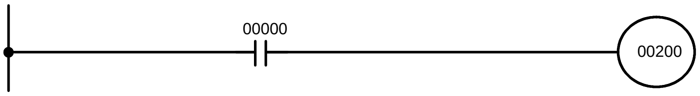
Address	Instruction	Operands
00000	LD	00000
00001	AND	00001
00002	OR	00200
00003	AND	00002
00004	AND NOT	00003
00005	Instruction	

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.

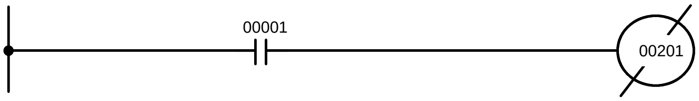
In more complicated diagrams, however, it is necessary to consider logic blocks before an execution condition can be determined for the final instruction, and that's where AND LOAD and OR LOAD instructions are used. Before we consider more complicated diagrams, however, we'll look at the instructions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

4-3-4 OUTPUT and OUTPUT NOT

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



Address	Instruction	Operands
00000	LD	00000
00001	OUT	00200



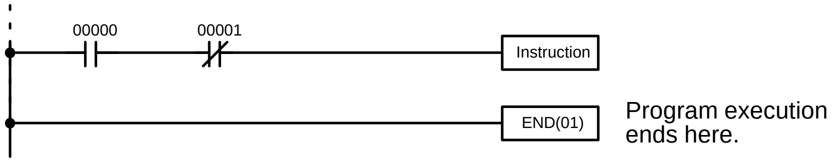
Address	Instruction	Operands
00000	LD	00001
00001	OUT NOT	00201

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 would 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 *5-10-1 TIMER – TIM* for details.

4-3-5 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 instructions up to the first END instruction before returning to the beginning of the program and beginning execution again. Although an END instruction can be placed at any point in a program, which is sometimes done when debugging, no instructions past the first END instruction will be executed 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.



Program execution ends here.

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.

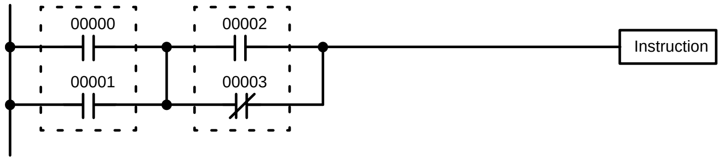
Now you have all of the instructions required to write simple input-output programs. Before we finish with ladder diagram basics and go onto inputting the program into the PC, let's look at logic block instruction (AND LOAD and OR LOAD), which are sometimes necessary even with simple programs.

4-3-6 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

6.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).

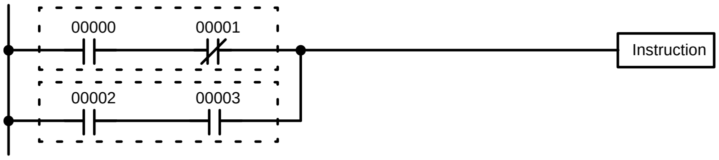
The above ladder diagram cannot, however, be converted to mnemonic code using AND and OR instructions alone. If an AND between IR 00002 and the results of an OR between IR 00000 and IR 00001 is attempted, the OR NOT between IR 00002 and IR 00003 is lost and the OR NOT ends up being an OR NOT between 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 special buffers and the logic process is begun over. To combine the results of the current execution condition with that of a previous “unused” execution condition, an AND LOAD or an OR LOAD instruction is used. Here “LOAD” refers to loading the last unused execution condition. An unused execution condition is produced by using the LOAD or LOAD NOT instruction for any but the first condition on an instruction line.

Analyzing the above ladder diagram in terms of mnemonic instructions, the condition for 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 would 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

7.The following diagram requires an OR LOAD instruction between the top logic block and the bottom logic block. An ON execution condition would be produced for the instruction at the right either when 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.



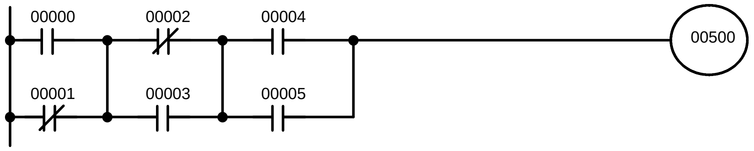
Address	Instruction	Operands
00000	LD	00000
00001	AND NOT	00001
00002	LD	00002
00003	AND	00003
00004	OR LD	---

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 means of coding the programs are also shown.

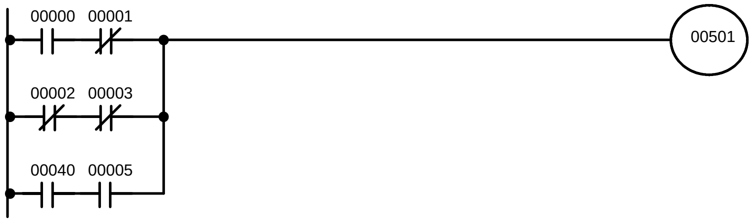


Address	Instruction	Operands
00000	LD	00000
00001	OR NOT	00001
00002	LD NOT	00002
00003	OR	00003
00004	AND LD	—
00005	LD	00004
00006	OR	00005
00007	AND LD	—
00008	OUT	00500

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 conditions in series 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 code for both methods is 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

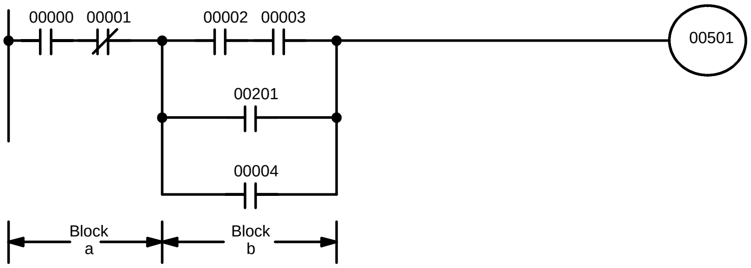
Address	Instruction	Operands
00000	LD	00000
00001	AND NOT	00001
00002	LD NOT	00002
00003	AND NOT	00003
00004	LD	00004
00005	AND	00005
00006	OR LD	—
00007	OR LD	—
00008	OUT	00501

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.

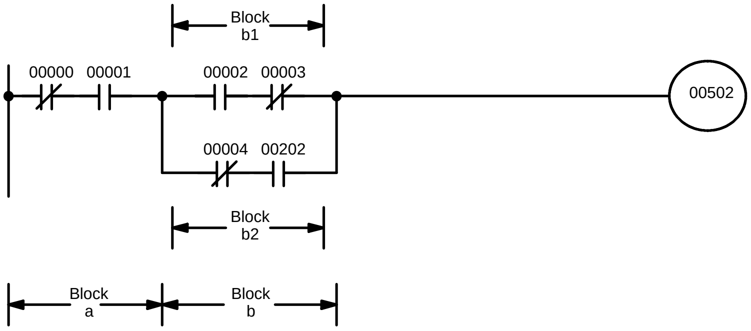
The following diagram contains only two logic blocks as shown. It is not necessary to further separate block b components, because it can coded directly using only AND and OR.



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.



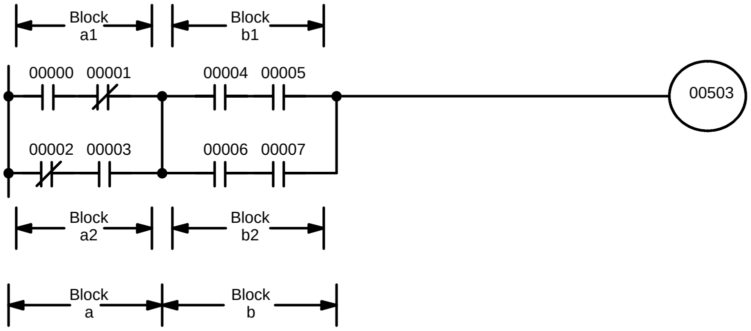
Address	Instruction	Operands
00000	LD NOT	00000
00001	AND	00001
00002	LD	00002
00003	AND NOT	00003
00004	LD NOT	00004
00005	AND	00202
00006	OR LD	—
00007	AND LD	—
00008	OUT	00502

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.

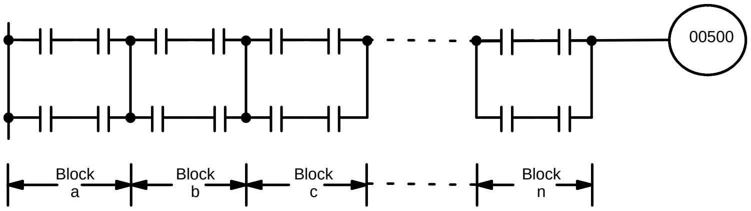
When working with complicated diagrams, blocks will ultimately be coded starting at the top left and moving down before moving across. This will generally mean that, when there might be a choice, OR LOAD will be coded before AND LOAD.

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.

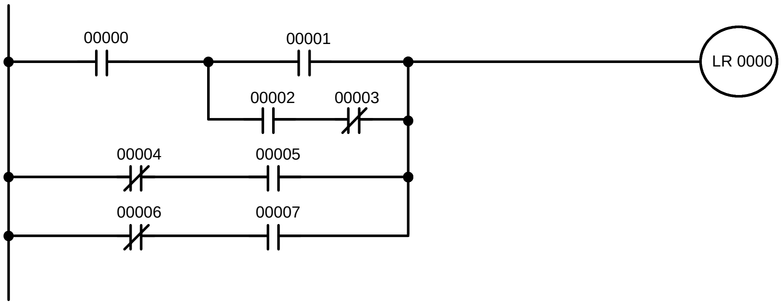


Address	Instruction	Operands	
00000	LD	00000	
00001	AND NOT	00001	
00002	LD NOT	00002	
00003	AND	00003	
00004	OR LD	—	Blocks a1 and a2
00005	LD	00004	
00006	AND	00005	
00007	LD	00006	
00008	AND	00007	
00009	OR LD	—	Blocks b1 and b2
00010	AND LD	—	Blocks a and b
00011	OUT	00503	

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.

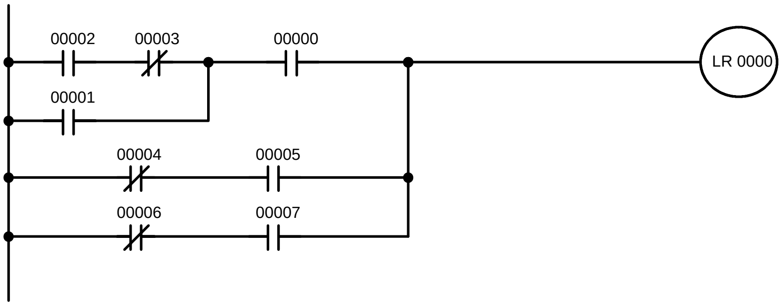


The following diagram requires an OR LOAD followed by an AND LOAD to code the top of the three blocks, and then two more OR LOADs to complete the mnemonic code.



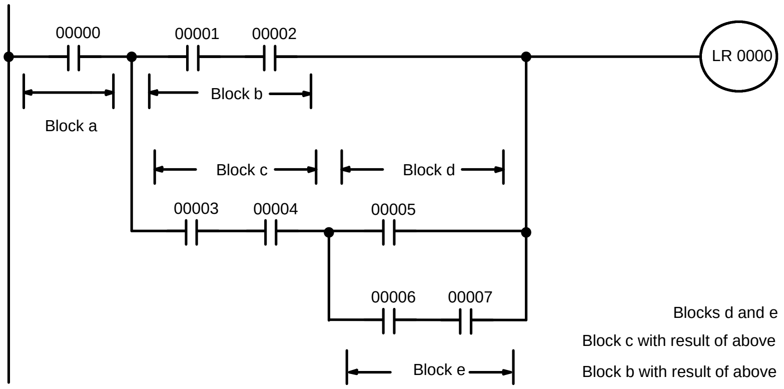
Address	Instruction	Operands
00000	LD	00000
00001	LD	00001
00002	LD	00002
00003	AND NOT	00003
00004	OR LD	––
00005	AND LD	––
00006	LD NOT	00004
00007	AND	00005
00008	OR LD	––
00009	LD NOT	00006
00010	AND	00007
00011	OR LD	––
00012	OUT	LR 0000

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.



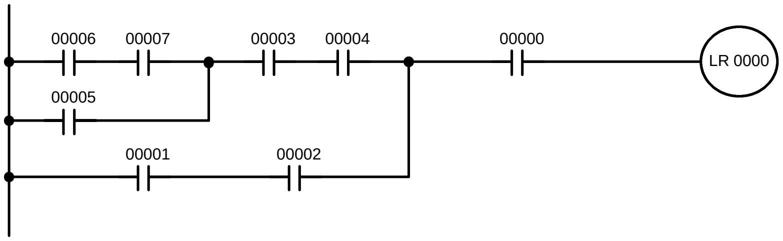
Address	Instruction	Operands
00000	LD	00002
00001	AND NOT	00003
00002	OR	00001
00003	AND	00000
00004	LD NOT	00004
00005	AND	00005
00006	OR LD	––
00007	LD NOT	00006
00008	AND	00007
00009	OR LD	––
00010	OUT	LR 0000

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.



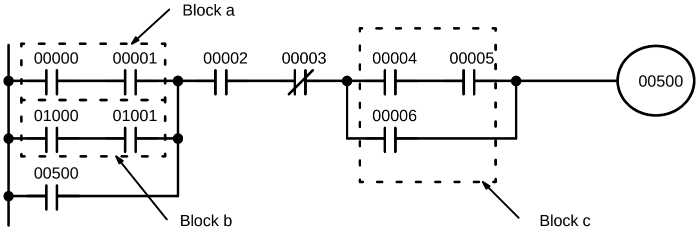
	Address	Instruction	Operands
	00000	LD	00000
	00001	LD	00001
	00002	AND	00002
	00003	LD	00003
	00004	AND	00004
	00005	LD	00005
	00006	LD	00006
	00007	AND	00007
Blocks d and e	00008	OR LD	––
Block c with result of above	00009	AND LD	––
Block b with result of above	00010	OR LD	––
Block a with result of above	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.

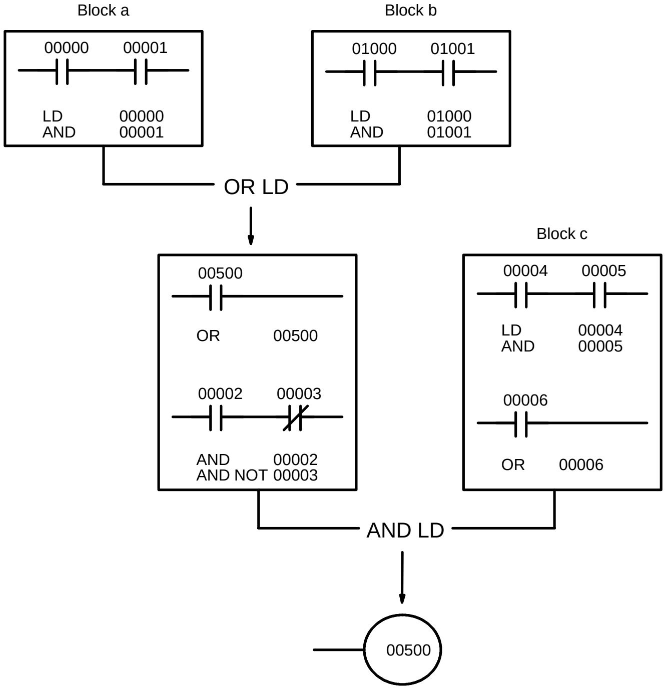


Address	Instruction	Operands
00000	LD	00006
00001	AND	00007
00002	OR	00005
00003	AND	00003
00004	AND	00004
00005	LD	00001
00006	AND	00002
00007	OR LD	––
00008	AND	00000
00009	OUT	LR 0000

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:



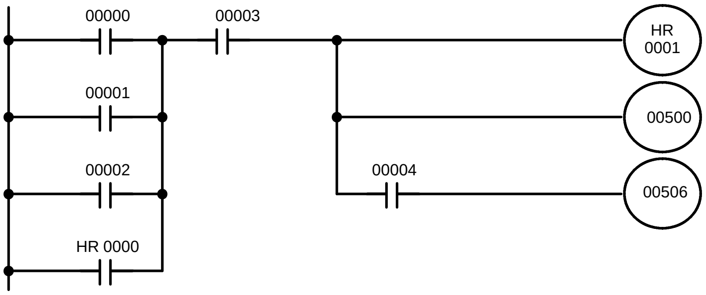
The first logic block instruction is used to combine the execution conditions resulting from blocks a and b, and the second one is to combine the execution condition of block c with the execution condition resulting from the normally closed condition assigned IR 00003. The rest of the diagram can be 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

4-3-7 Coding Multiple Instructions

If there is more than one 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

4-4 The Programming Console

Once a program has been written, it must be input into the PC. This can be done in graphic (ladder diagram) form using a GPC,a FIT, or LSS. The most common way of inputting a program, however, is through a Programming Console using mnemonic code. This section and *4-4-1 TERMINAL and CONSOLE Modes* describe the Programming Console and the operation necessary to prepare for program input. Refer to *4-6 Inputting, Modifying, and Checking the Program* for details on actual procedures for inputting the program into memory.

Depending on the model of Programming Console used, it is either connected to the CPU via a Programming Console Adapter and Connecting Cable or it is mounted directly to the CPU.

4-4-1 TERMINAL and CONSOLE Modes

When mounted to the C20H/C28H/C40H/C60H, the Programming Console can be set to operate as a display terminal in what is called TERMINAL mode. While the Programming Console is in TERMINAL mode, key inputs from the Programming Console can be used to turn ON/OFF certain memory bits. TERMINAL mode is a Programming Console mode; do not confuse it with PC modes (MONITOR, PROGRAM, and RUN mode). The normal Programming Console mode is CONSOLE mode.

When the Programming Console is in TERMINAL mode, all normal Programming Console operations and the mode switch on the Programming Console are disabled. You must return to CONSOLE mode to enable normal Programming Console functions.

Entering TERMINAL Mode

There are three ways to enter TERMINAL mode. One is directly through key operations on the Programming Console; the other is by executing KEY(62).

TERMINAL mode is entered from the Programming Console by pressing CHG while the password display or a mode display is on the Programming Console. If the Programming Console is not in one of these conditions, press CLR to reset the display before pressing CHG. When TERMINAL mode is entered, a message display will appear (see next section).

To return to CONSOLE mode, press CHG again. A mode display will appear to indicate the PC mode (either MONITOR, PROGRAM, or RUN mode).

TERMINAL mode can be enter from MONITOR, PROGRAM, or RUN mode. These modes will not change when TERMINAL mode is entered and will also be maintained when TERMINAL mode is left.

KEY(62) is used to execute the Programming Console key operations from the program. It can therefore be used to execute the key sequence required to enter and leave TERMINAL mode. The normal procedure is to execute KEY(62) with operand words that contain 4021. The first part, 40 is CLR and resets the Programming Console display and the second part, 21, is the CHG key code. The same code can then be used to return the Programming Console to CONSOLE mode.

KEY(62) will not be executed and TERMINAL mode will not be entered unless a Programming Console is mounted. Refer to *Section 5 Instruction Set* for details on KEY(62).

The last way to enter TERMINAL mode is to start up in TERMINAL mode by either designating it as the startup mode or by designating to restart in the same mode as when the PC was turned off and turning off the PC in TERMINAL mode. These designations are made in DM 0900. Refer to *4-4-3 PC Modes* for details.

TERMINAL Mode Displays

When the Programming Console is in TERMINAL mode, displays programmed with MSG(46) or LMSG(47)will be displayed. With these instructions, the programmer can input into memory either 16- or 32-character messages that are output when the instruction is executed.

If one of these instructions has been executed, the message will appear on the Programming Console as soon as TERMINAL mode is entered. If more than one of these instruction is executed, the last one will always be displayed. If none of these instructions has been executed, a display indicated that no message has yet been generated will appear.

The last message is preserved even after the PC mode is changed.

TERMINAL Mode Key Inputs

While in TERMINAL mode, the bits of AR 22 can be turned ON by pressing the numeric keys on the Programming Console. The keys and corresponding bits in AR 22 are as follows:

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

This operation is controlled by AR 0708 (TERMINAL Mode Input Cancel Bit). As long as AR 0708 is OFF, Programming Console keys will turn ON the corresponding bits when pressed. When AR 0708 is turned ON, all AR 22 bits will be turned OFF and key inputs will be disabled until AR 0708 is turned OFF again.

AR 22 status is maintained when the Programming Console enters and leaves TERMINAL mode and when PC power is turned off and on.

4-4-2 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.

4-4-3 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.

TERMINAL mode and CONSOLE mode are Programming Console modes and are independent of RUN, MONITOR, and PROGRAM modes, i.e., the System can simultaneously be in both TERMINAL and RUN mode. TERMINAL and CONSOLE modes are described in *Section 4-4-1 TERMINAL and CONSOLE Modes*.

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

Mode Changes

When the PC is turned on, the initial operating mode is affected by any Peripheral Device connected or mounted to the CPU as well as by designations made in DM 0900 as shown in the following table:

Setting in DM 0900, bits 08 to 15	CPU connector	Initial operating mode
Set to start in mode set on Programming Console mode switch	Nothing connected	RUN/CONSOLE mode
	Programming Console mounted	Mode set on mode switch and CONSOLE mode
	Peripheral Interface Unit mounted	PROGRAM/CONSOLE mode
Set to continue mode PC was in when last turned off	Nothing connected	Mode the PC was in when turned off.
	Programming Console mounted	
	Peripheral Interface Unit mounted	
Set to start in mode set in bits 00 to 07 of DM 0900	Nothing connected	Mode set in bits 00 to 07 of DM 0900
	Programming Console mounted	
	Peripheral Interface Unit mounted	

If the PC power supply is already turned on when a Peripheral Device is attached to the PC, the PC will stay in the same mode it was in before the Peripheral Device was attached. The mode can be changed with the mode switch on the Programming Console once the password has been entered.

The mode will not change when a peripheral device is removed from the PC after PC power is turned on.

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.

4-5 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.

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

1, 2, 3...

1. Confirm that all wiring for the PC has been installed and checked properly.
2. Confirm that a RAM Unit is mounted as the Memory Unit and that the write-protect switch is OFF.
3. Connect the Programming Console to the PC. Make sure that the Programming Console is securely connected or mounted to the CPU; improper connection may inhibit operation.
4. Set the mode switch to PROGRAM mode.
5. Turn on PC power.
6. Enter the password.
7. Clear memory.

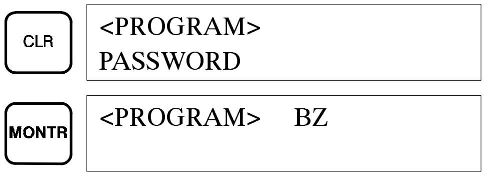
Each of these operations from entering the password on is described in detail in the following subsections. All operations should be done in PROGRAM mode unless otherwise noted.

4-5-1 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.



```
<PROGRAM>
PASSWORD
```

```
<PROGRAM>    BZ
```

Indicates the mode set by the mode selector switch.

Beeper

8.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 beeper that sounds when Programming Console keys are pressed. If BZ is displayed in the upper right corner, the beeper is operative. If BZ is not displayed, the beeper is not operative.

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

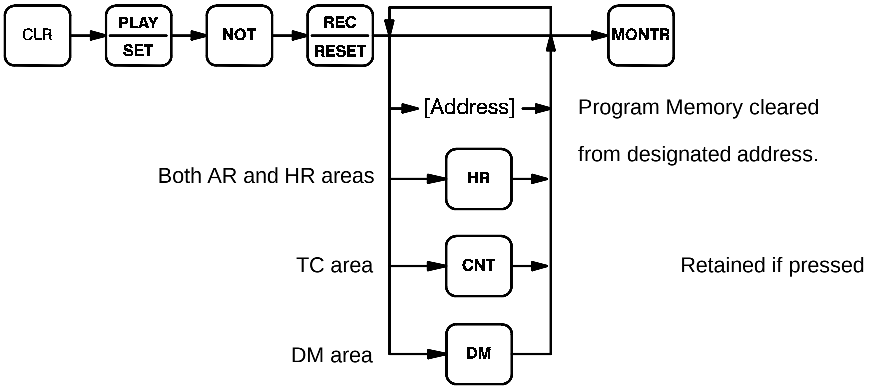
4-5-2 Clearing Memory

Using the Memory Clear operation it is possible to clear all or part of the Program Memory, and the IR, HR, AR, DM and TC areas. Unless otherwise specified, the clear operation will clear all of the above memory areas, provided that the Memory Unit attached to the PC is a RAM Unit or an EEPROM Unit and the write-enable switch is ON. If the write-enable switch is OFF or the Memory Unit is an EPROM Unit, Program Memory cannot be cleared. (The system DM areas, DM 0900 to DM 0999 and DM 1900 to DM 1999, will not be cleared in any circumstances.)

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 whether there is already a program loaded. If there is one, determine whether it is one that you need. If you do need the program, clear only the memory areas that you do not need. 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 Debugging 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.

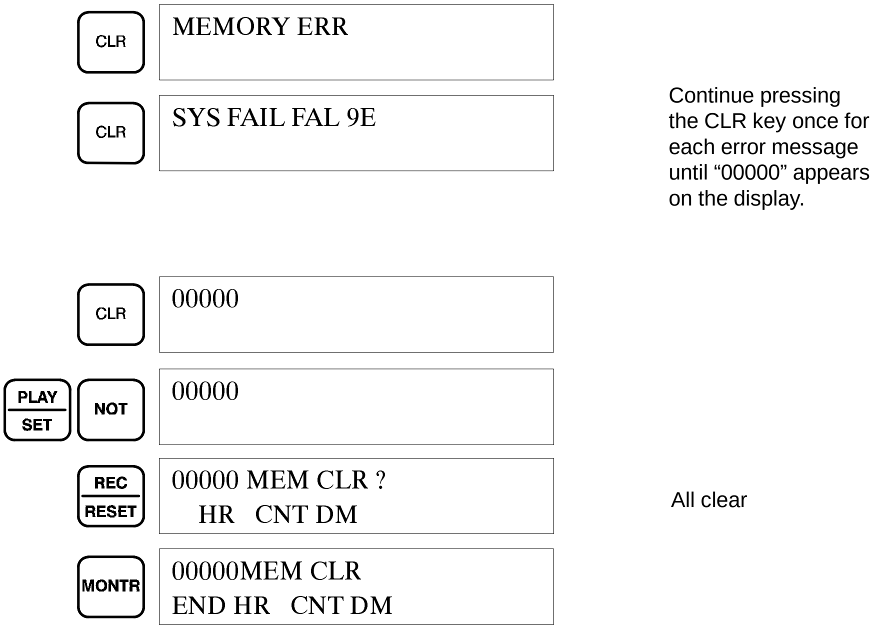
Memory can be cleared in PROGRAM mode only.

Key Sequence



All Clear

The following procedure is used to clear memory completely. Before executing these steps, the write enable switch on the CPU must be switched ON. If the switch is not ON, the Program Memory and DM 1000 to DM 1899 cannot be cleared. (The HR, AR, and TC areas, as well as DM 0000 to 0899, will be cleared even if the write enable switch is OFF.) After turning on the switch, follow the key sequence described below.



```
MEMORY ERR

SYS FAIL FAL 9E

00000

00000

00000 MEM CLR ?
   HR   CNT DM

00000MEM CLR
END HR   CNT DM
```

Continue pressing the CLR key once for each error message until “00000” appears on the display.

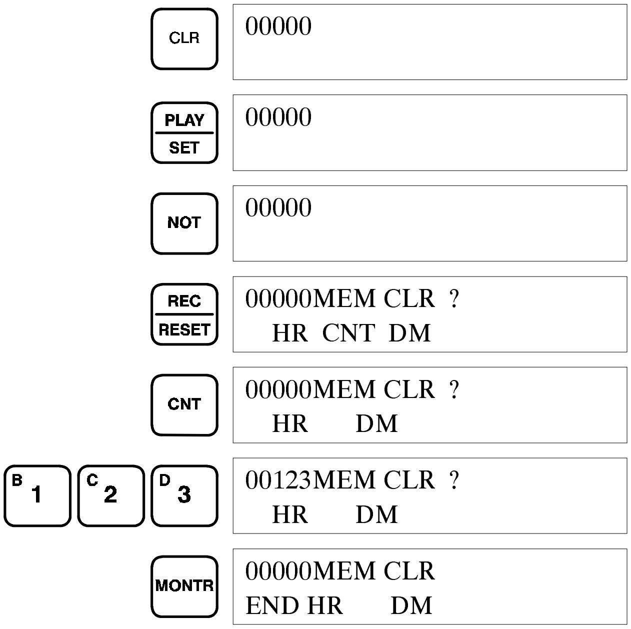
All clear

Partial Clear

It is possible to retain the data in specified areas or part of the Program Memory. 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 Program Memory from the beginning to a specified address. After designating the data areas to be retained, specify the first Program Memory 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.

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



```
00000

00000

00000

00000MEM CLR  ?
   HR  CNT  DM

00000MEM CLR  ?
   HR       DM

00123MEM CLR  ?
   HR       DM

00000MEM CLR
END HR      DM
```

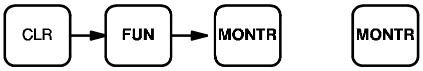
4-5-3 Clearing Error Messages

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 9 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 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 already 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 Debugging and Execution*.

4-6-1 Setting and Reading from Program Memory Address

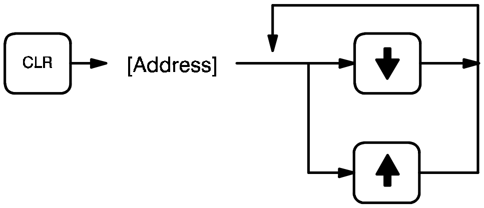
When inputting a program for the first time, it is generally written to Program Memory starting from address 00000. Because this address appears when the display is cleared, it is not necessary to specify it.

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.

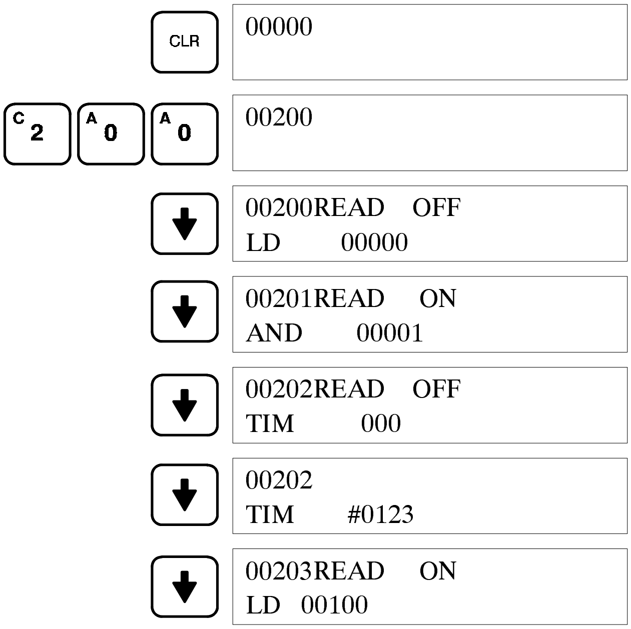
If Program Memory is read in RUN or MONITOR mode, the ON/OFF status of any displayed bit will also be shown.

Key Sequence



Example

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



```
00000

00200

00200READ    OFF
LD        00000

00201READ     ON
AND       00001

00202READ    OFF
TIM         000

00202
TIM       #0123

00203READ     ON
LD   00100
```

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

4-6-2 Entering or Editing Programs

Programs can be entered or edited only in PROGRAM mode. The write-enable switch on the CPU must also be set to ON.

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 input a program, just follow the mnemonic code that was produced from the ladder diagram, ensuring that the proper address is set before starting. Once the proper address is displayed, input the first instruction word, press WRITE. Next, input any operands required, and press WRITE after each, i.e., WRITE is pressed at the end of each line of the mnemonic code. When WRITE is pressed, the designated instruction will be 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

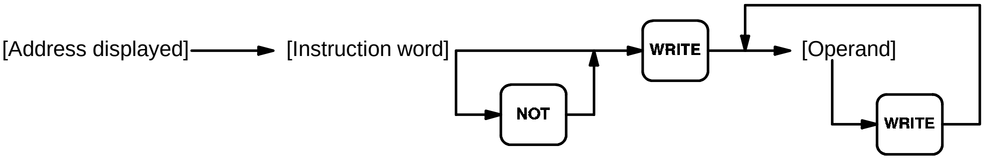
The most basic instructions are input using the Programming Console keys provided for them. All other instructions are entered using function codes. These function codes are always written after the instruction's mnemonic. If no function code is given, there should be a Programming Console key for that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

CLR	00000
C 2 A 0 A 0	00200
LD ┤├ C 2	00200 LD 00002
WRITE	00201READ NOP (00)
TIM	00201 TIM 000
WRITE	00201 TIM DATA #0000
B 1 C 2 D 3	00201 TIM #0123
WRITE	00202READ NOP (00)
FUN	00202 FUN (??)
B 1 F 5	00202 TIMH (15) 001
WRITE	00202 TIMH DATA #0000
F 5 A 0 A 0	00202 TIMH #0500
WRITE	00203READ NOP (00)

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 ROM or to write-protected RAM. Be sure a RAM Unit is mounted and that its 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 or use a larger Memory Unit.
****ADDR OVER	An address was set that is larger than the highest memory in Program Memory. 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-input 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.

4-6-3 Checking the Program

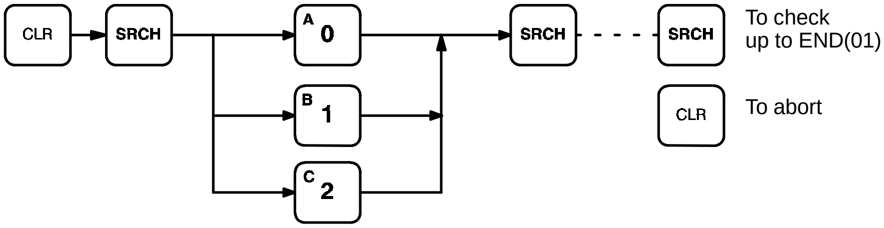
Once a program has been entered, it should be checked for syntax to be sure that no programming rules have been violated. This check should also be performed if the program has been changed in any way that might create a syntax error.

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.

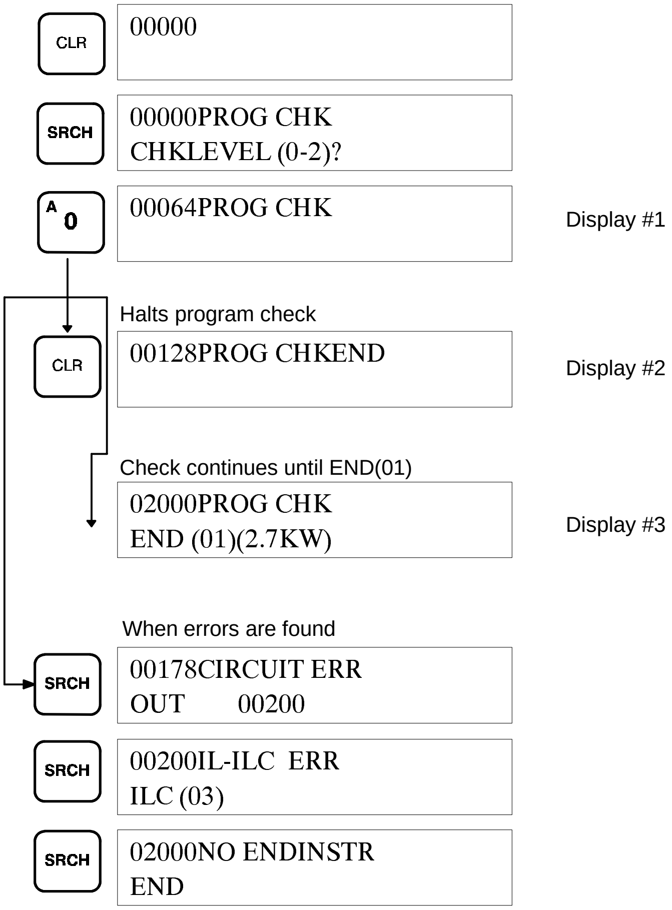
The address where the error was generated will also be displayed.

Many of the following errors are for instructions that have not yet been described. Refer to *4-7 Controlling Bit Status* or *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, block 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).
	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.

Note The Programming Console does not check whether output bits are controlled by more than one instruction. Check the program from the LSS, FIT, or GPC to check for duplicate output bits.

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



```
00000

00000PROG CHK
CHKLEVEL (0-2)?

00064PROG CHK

00128PROG CHKEND

02000PROG CHK
END (01)(2.7KW)

00178CIRCUIT ERR
OUT       00200

00200IL-ILC  ERR
ILC (03)

02000NO ENDINSTR
END
```

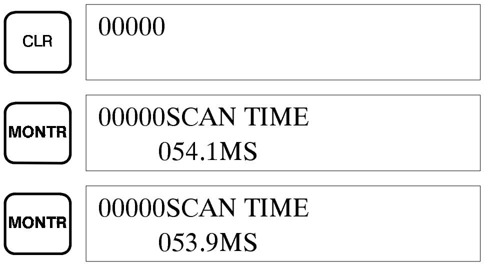
4-6-4 Displaying the Cycle Time

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.

Note "SCAN TIME" is displayed instead of cycle time.

Example



```
00000

00000SCAN TIME
      054.1MS

00000SCAN TIME
      053.9MS
```

4-6-5 Program Searches

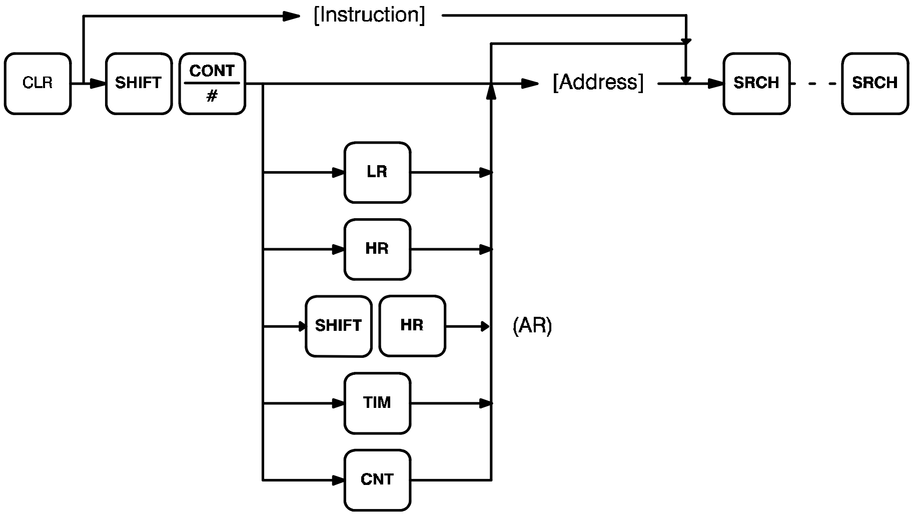
The program can be searched for occurrences of any designated instruction or data area address used in an instruction. Searches can be performed from any currently displayed address or from a cleared display.

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

CLR	00000
LD ⊣⊢	00000 LD 00000
SRCH	00200SRCH LD 00000
SRCH	00202 LD 00000
SRCH	02000SRCH END (01)(02.7KW)

CLR	00000
B 1 A 0 A 0	00100
TIM B 1	00100 TIM 001
SRCH	00203SRCH TIM 001
↓	00203 TIM DATA #0123

Example: Bit Search

CLR	00000
SHIFT CONT/# F 5	00000CONT SRCH CONT 00005
SRCH	00200CONT SRCH LD 00005
SRCH	00203CONT SRCH AND 00005
SRCH	02000 END (01)(02.7KW)

4-6-6 Inserting and Deleting Instructions

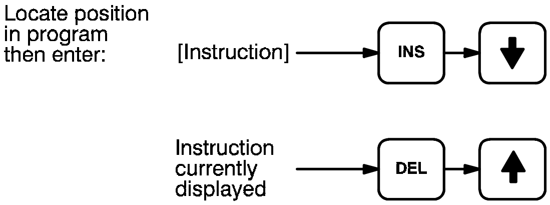
In PROGRAM mode, any instruction that is currently displayed can be deleted or another instruction can be inserted before it. These are not possible in RUN or MONITOR modes.

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

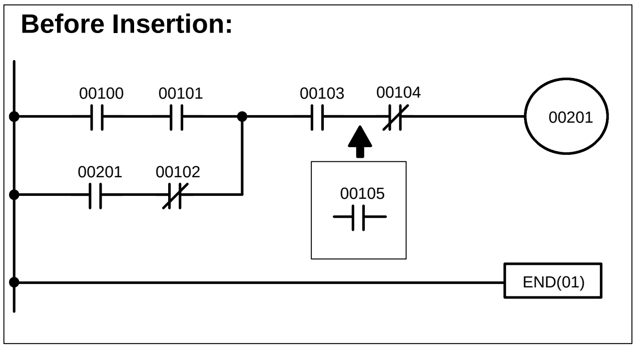


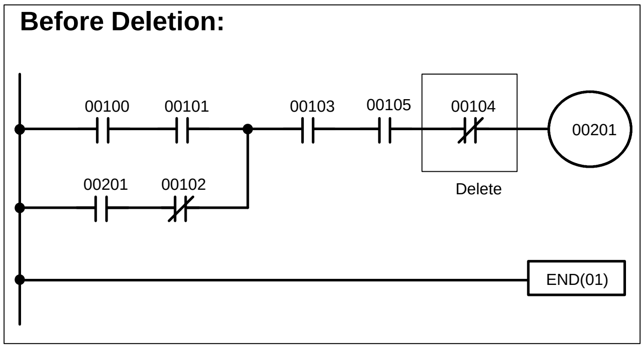
When an instruction is inserted or deleted, all addresses in Program Memory following the operation are adjusted automatically so that there are no blank addresses or no unaddressed instructions.

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

Original Program

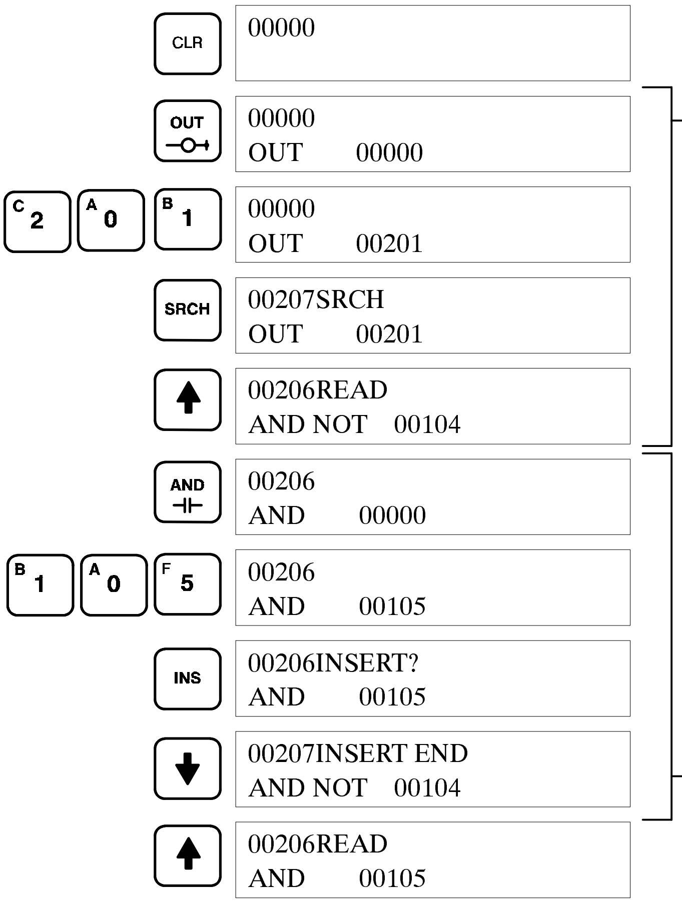
Address	Instruction	Operands
00000	LD	00100
00001	AND	00101
00002	LD	00201
00003	AND NOT	00102
00004	OR LD	–
00005	AND	00103
00006	AND NOT	00104
00007	OUT	00201
00008	END(01)	–


Before Insertion:


Before Deletion:

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

Inserting an Instruction



```
00000

00000
OUT       00000

00000
OUT       00201

00207SRCH
OUT       00201

00206READ
AND NOT   00104

00206
AND       00000

00206
AND       00105

00206INSERT?
AND       00105

00207INSERT END
AND NOT   00104

00206READ
AND       00105
```

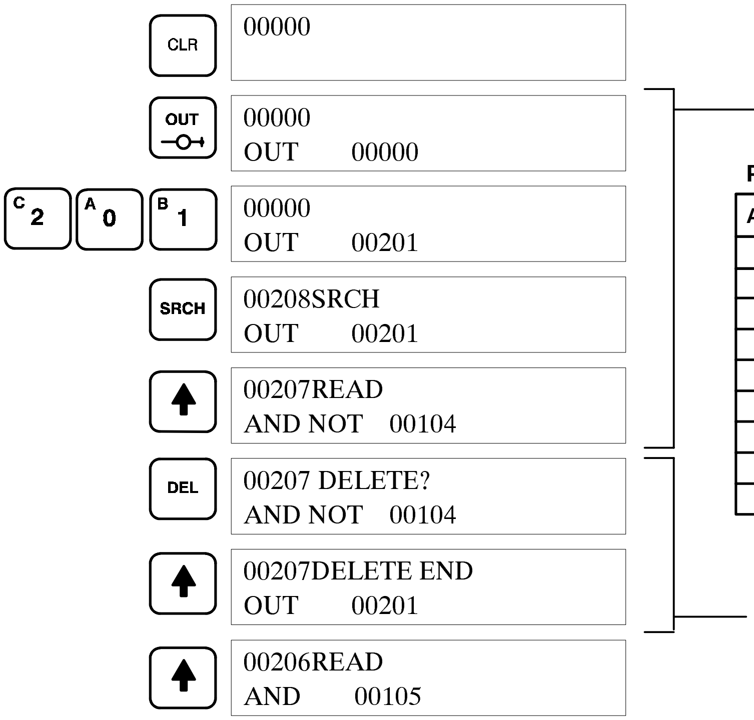
Find the address prior to the insertion point

Insert the 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



```
00000

00000
OUT       00000

00000
OUT       00201

00208SRCH
OUT       00201

00207READ
AND NOT   00104

00207 DELETE?
AND NOT   00104

00207DELETE END
OUT       00201

00206READ
AND       00105
```

Find the instruction that requires deletion.

Confirm that this is the instruction to be deleted.

Program After Deletion

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	OUT	00201
00008	END(01)	–

4-6-7 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 one a branch line. If a condition exists on any of the instruction lines after the branching point, the execution condition could change during this time making proper execution impossible. The following diagrams illustrate this. In both diagrams, instruction 1 is executed before returning to the branching point and moving on to the branch line leading to instruction 2.

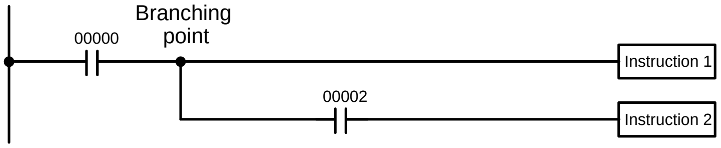


Diagram A: Correct Operation

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

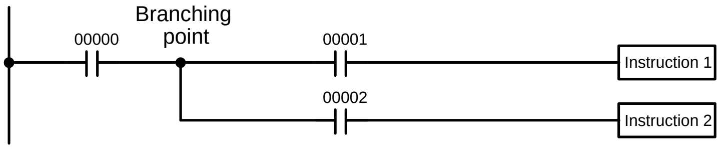


Diagram B: Incorrect Operation

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

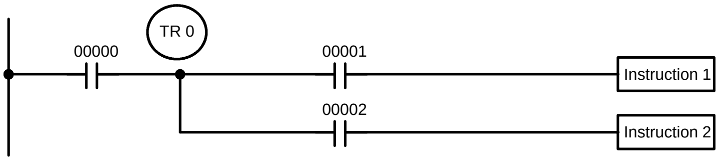
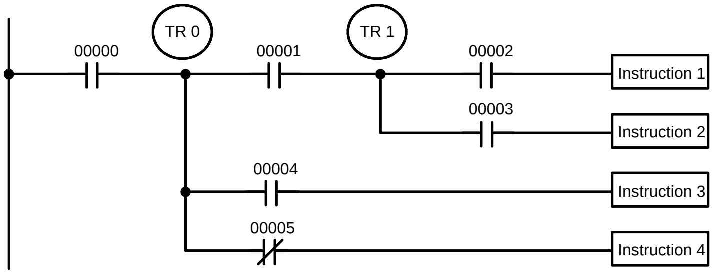


Diagram B: Corrected Using a TR bit

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.



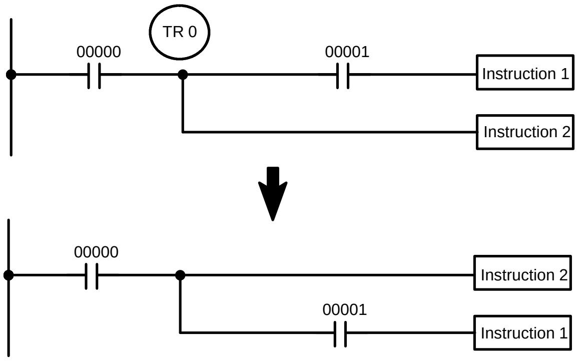
Address	Instruction	Operands
00000	LD	00000
00001	OUT	TR 0
00002	AND	00001
00003	OUT	TR 1
00004	AND	00002
00005	OUT	00500
00006	LD	TR 1
00007	AND	00003
00008	OUT	00501
00009	LD	TR 0
00010	AND	00004
00011	OUT	00502
00012	LD	TR 0
00013	AND NOT	00005
00014	OUT	00503

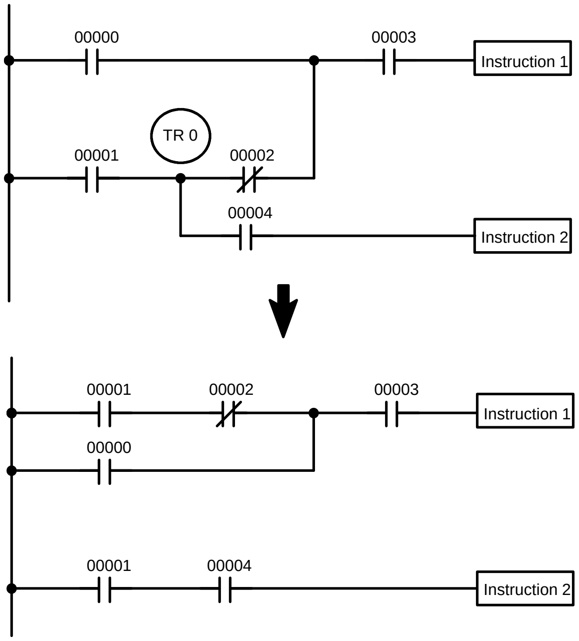
In this example, TR 0 and TR 1 are used to store the execution conditions at the branching points. After executing instruction 1, the execution condition stored in TR 1 is loaded for an AND with the status IR 00003. The execution condition stored in TR 0 is loaded twice, the first time for an AND with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





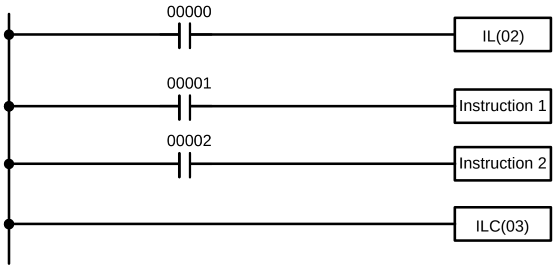
Note TR bits are only used when programming using mnemonic code. They are not necessary when inputting ladder diagrams directly, as is possible from a GPC. The above limitations on the number of branching points requiring TR bits, and considerations on methods to reduce the number of programming instructions, 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 *5-7 INTERLOCK and INTERLOCK CLEAR – IL(02) and ILC(03).*

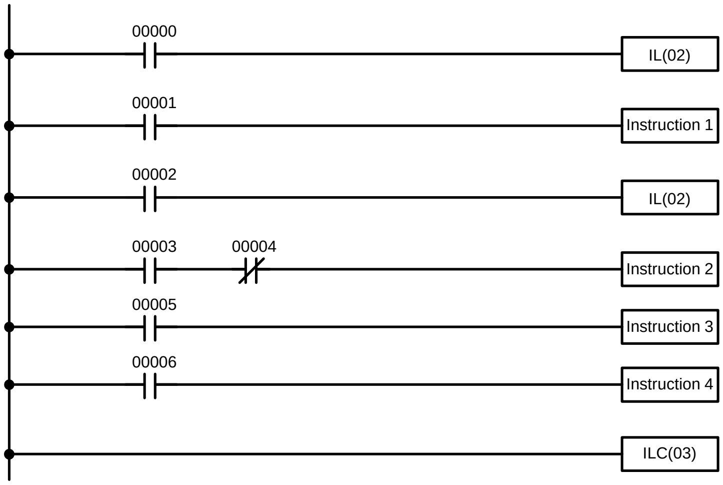
Diagram B on page 81 can also be corrected with an interlock. Here, the conditions leading up to the branching point are placed on an instruction line for the INTERLOCK instruction, all of lines leading from the branching point are written as separate instruction lines, and another instruction line is 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.

4-6-8 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 49. 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 49 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 49 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.

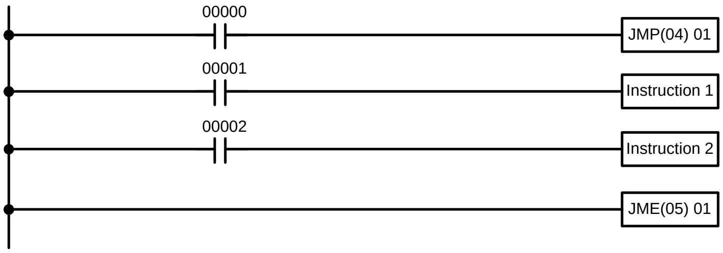


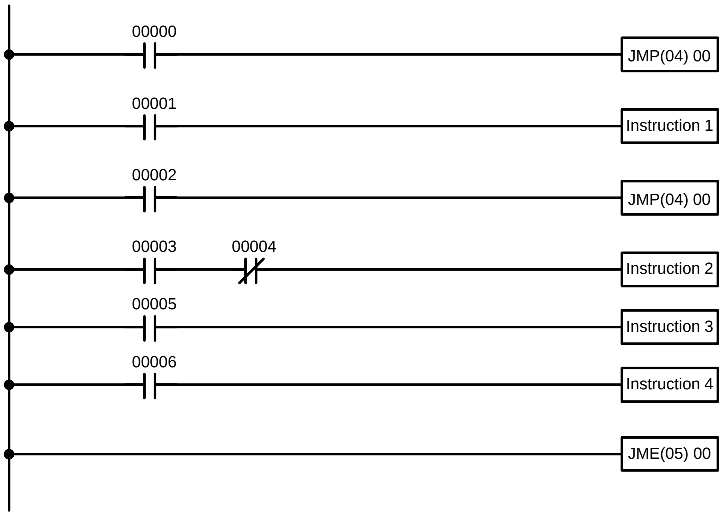
Diagram B: Corrected with a Jump

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)	01

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).



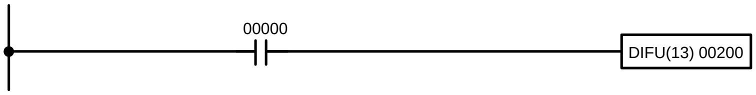
Address	Instruction	Operands
00000	LD	00000
00001	JMP(04)	00
00002	LD	00001
00003	Instruction 1	
00004	LD	00002
00005	JMP(04)	00
00006	LD	00003
00007	AND NOT	00004
00008	Instruction 2	
00009	LD	00005
00010	Instruction 3	
00011	LD	00006
00012	Instruction 4	
00013	JME(05)	00

4-7 Controlling Bit Status

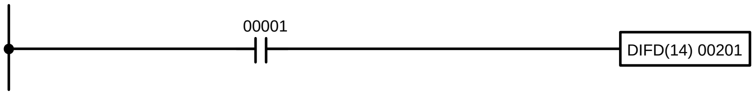
There are five instructions that can be used generally to control individual bit status. These are the OUTPUT, OUTPUT NOT, DIFFERENTIATE UP, DIFFERENTIATE DOWN, and KEEP instructions. All of these instructions appear as the last instruction in an instruction line and take a bit address for an operand. Although details are provided in *5-6 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.

4-7-1 DIFFERENTIATE UP and DIFFERENTIATE DOWN

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



Address	Instruction	Operands
00000	LD	00000
00001	DIFU(13)	00200



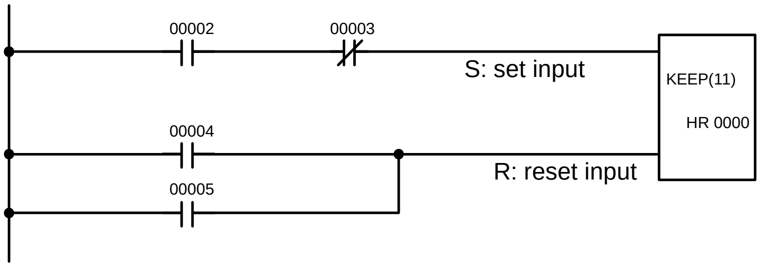
Address	Instruction	Operands
00000	LD	00001
00001	DIFD(14)	00201

Here, IR 00200 will be turned ON for one scan 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 scan 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.

4-7-2 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.



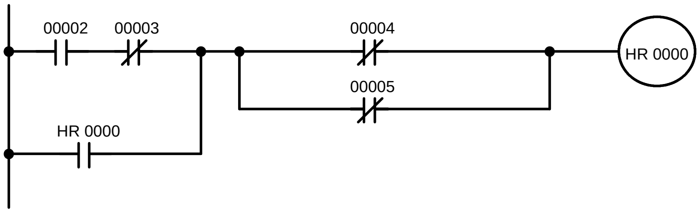
Address	Instruction	Operands
00000	LD	00002
00001	AND NOT	00003
00002	LD	00004
00003	OR	00005
00004	KEEP(11)	HR 0000

4-7-3 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	OR NOT	00005
00005	OUT	HR 0000

4-8 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

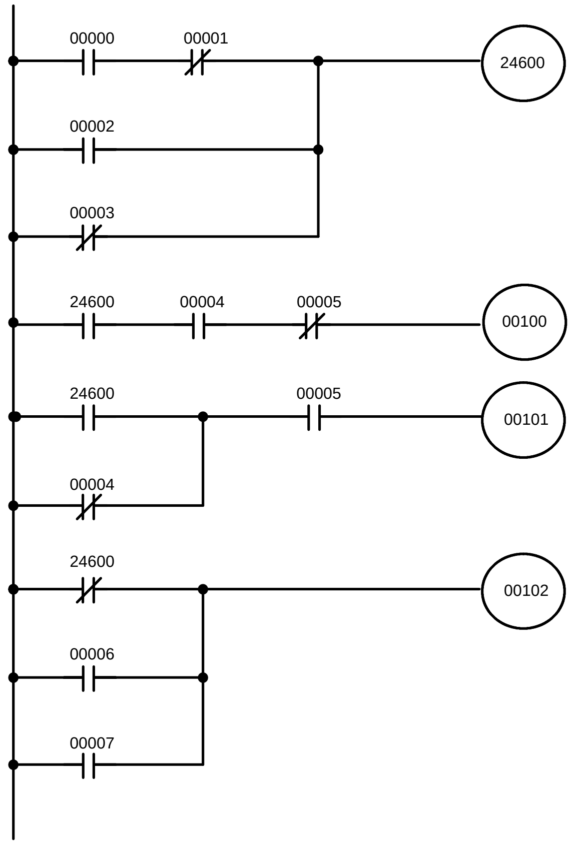
Examples given later in this subsection show two of the most common ways to employ work bits. These should act as a guide to the almost limitless number of ways in which the work bits can be used. Whenever difficulties arise in programming a control action, consideration should be given to work bits 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 *5-11-1 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.

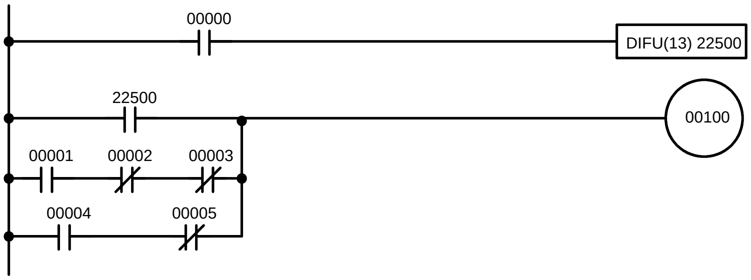


Address	Instruction	Operands
00000	LD	00000
00001	AND NOT	00001
00002	OR	00002
00003	OR NOT	00003
00004	OUT	24600
00005	LD	24600
00006	AND	00004
00007	AND NOT	00005
00008	OUT	00100
00009	LD	24600
00010	OR NOT	00004
00011	AND	00005
00012	OUT	00101
00013	LD NOT	24600
00014	OR	00006
00015	OR	00007
00016	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 scan 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 scan and then be turned OFF the next scan 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 scan 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

4-9 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.

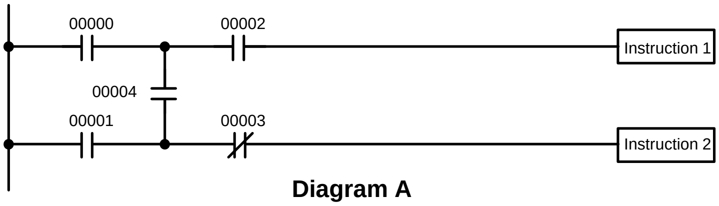


Diagram A

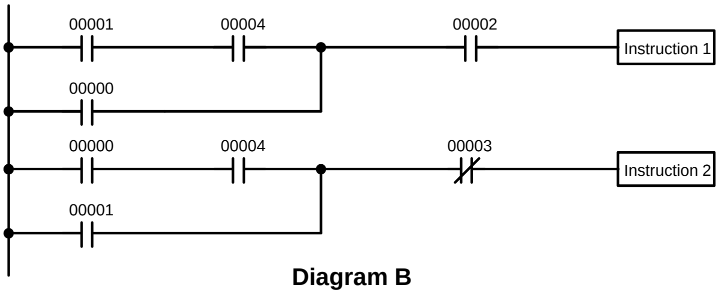


Diagram B

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.

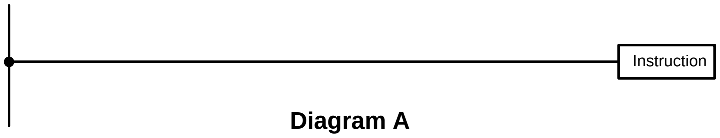


Diagram A

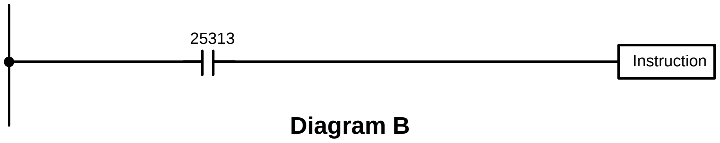
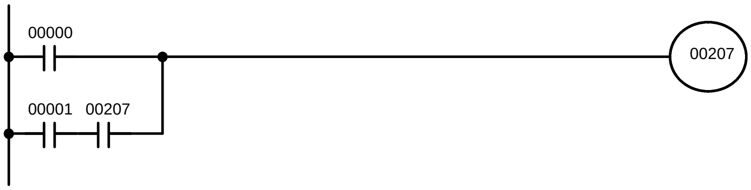


Diagram B

Address	Instruction	Operands
00000	LD	25313
00001	Instruction	

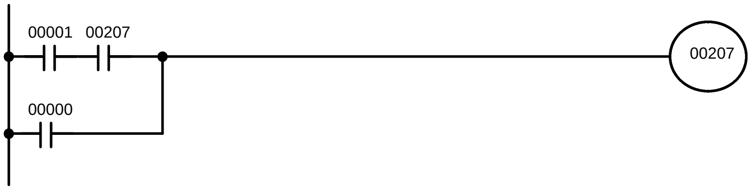
There are a few exceptions to this rule, including the INTERLOCK CLEAR, JUMP END, and step instructions. Each of these instructions is used as the second of a pair of instructions and is controlled by the execution condition of the first of the pair. Conditions should not be placed on the instruction 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 *5-5-2 AND LOAD and OR LOAD* for more details and *4-6 Inputting, Modifying and Checking the Program* for further examples.



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

Diagram A



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

Diagram B

4-10 Program Execution

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

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

SECTION 5
Instruction Set

This section explains each instruction in the large Mini H-type PC instruction sets and provides the ladder diagram symbol, data areas, and flags used with each.

The instructions are described in 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, Binary Calculation Instructions, BCD Calculation Instructions, Logic Instructions, Subroutines, and Special 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 *4-6 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 *5-4 Differentiated Instructions*.

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 required 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 used. 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 in the same data area. Unless a limit is specified, any bit/word in the area can be used. Any limitations are specified 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.

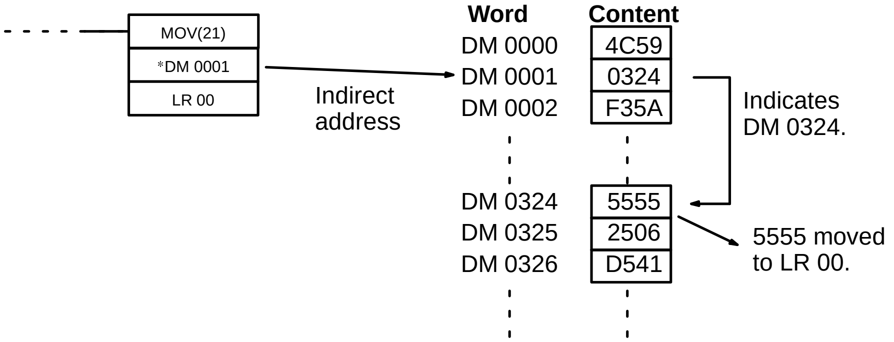
Abbreviation	Name	Bit
ER	Instruction Execution Error Flag	25503
CY	Carry Flag	25504
GR	Greater Than Flag	25505
EQ	Equals Flag	25506
LE	Less Than Flag	25507

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 D 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 0324, and DM 0324 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 0999).

Designating Constants

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

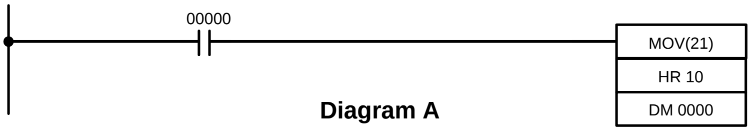


Diagram A

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

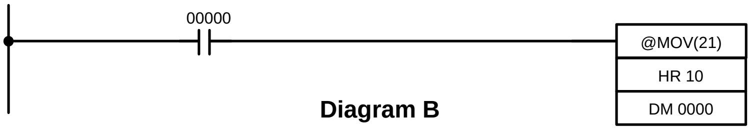


Diagram B

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 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 *5-7 INTERLOCK and INTERLOCK CLEAR – IL(02) and IL(03)* for the effects of interlocks on differentiated instructions.

The C20H/C28H/C40H/C60H also provide 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 *5-6-2 DIFFERENTIATE UP and DOWN – DIFU(13) and DIFD(14)* for details.

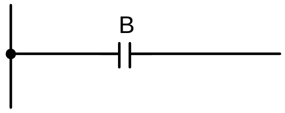
5-5 Ladder Diagram Instructions

Ladder Diagram instructions include Ladder instructions and Logic Block instructions. Ladder instructions correspond to the conditions on the ladder diagram. Logic block instructions are used to relate more complex parts of the diagram that cannot be programmed with Ladder instructions alone.

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

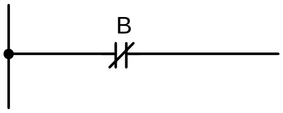
Ladder Symbols **Operand Data Areas**

LOAD – LD



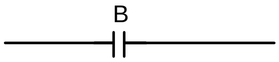
B: Bit
IR, SR, AR, HR, TC, LR, TR

LOAD NOT – LD NOT



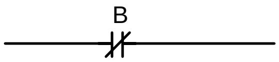
B: Bit
IR, SR, AR, HR, TC, LR

AND – AND



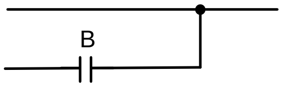
B: Bit
IR, SR, AR, HR, TC, LR

AND NOT – AND NOT



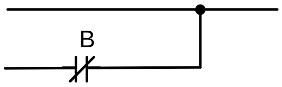
B: Bit
IR, SR, AR, HR, TC, LR

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 Entering Programs*, the status of the bits assigned to each instruction determines the execution conditions for all other instructions. Each of these instructions and each bit address can be used as many times as required. Each can be used in as many of these instructions as required.

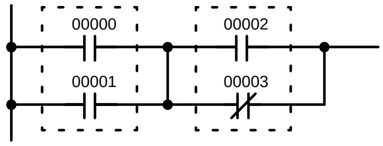
The status of the bit operand (B) assigned to LD or LD NOT determines the first execution condition. AND takes the logical AND between the execution condition and the status of its bit operand; AND NOT, the logical AND between the execution condition and the inverse of the status of its bit operand. 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. Refer to *4-3-3 Ladder Instructions* for details.

Flags There are no flags affected by these instructions.

5-5-2 AND LOAD and OR LOAD

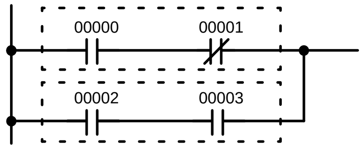
AND LOAD – AND LD

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 the GPC. 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 *4-6 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 *4-3-6 Logic Block Instructions*.

Flags There are no flags affected by these instructions.

5-5-3 Coding Conditions and Other 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.

Refer to the figures on page 102 for a sample ladder diagram and corresponding mnemonic code written on program coding sheet from *Appendix G*.

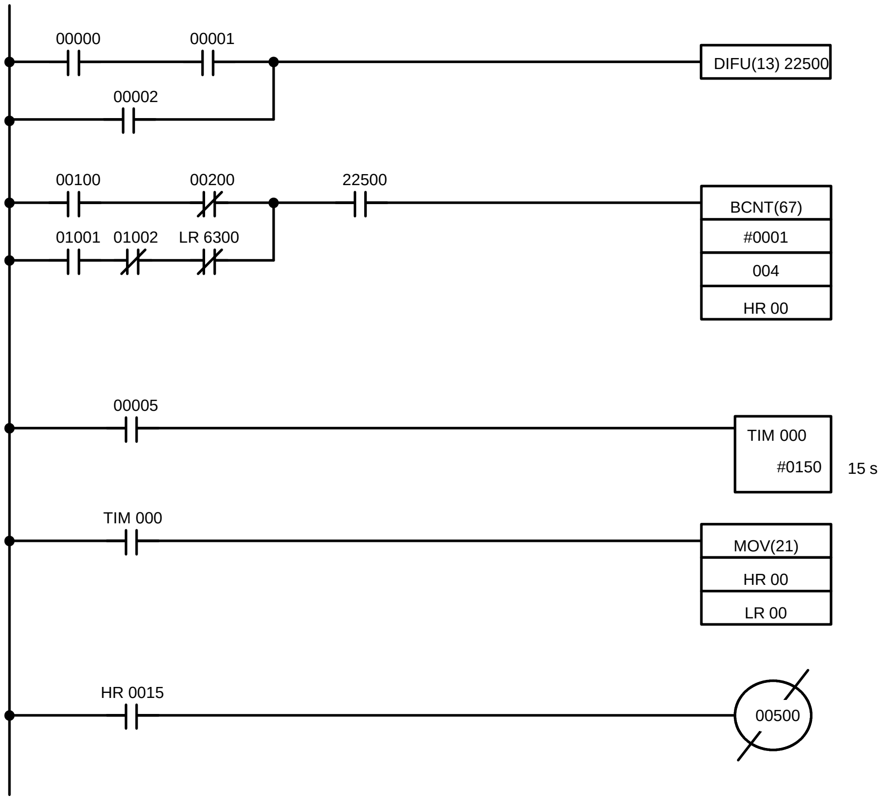
The first word of any instruction defines the instruction and provides any definers. If the instruction requires only a single bit operand with no definer, the bit operand is also placed on the same line as the mnemonic. All other operands are placed on lines after the instruction line, one operand per 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 cycled 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 is to be input, the number symbol (#) is placed on the left side of the data column and the number to be input is placed on the right side. Any numbers input as definers in the instruction word do not require the number symbol on the right side. 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.

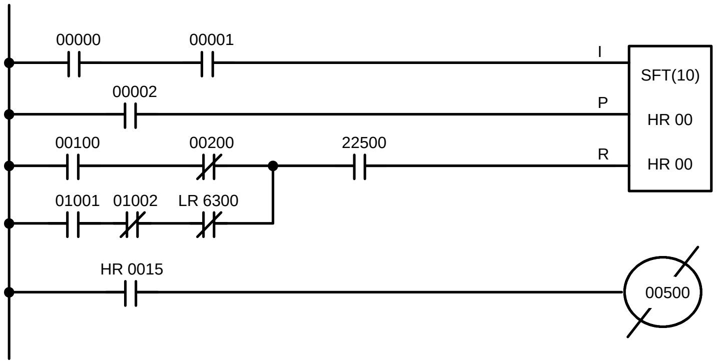
The following diagram and corresponding mnemonic code illustrate the points described above.



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 an 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 instruction. An example of this for SFT(10) is shown below.



Address	Instruction	Operands
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

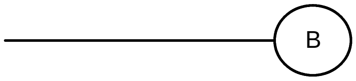
5-6 Bit Control Instructions

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.

5-6-1 OUTPUT and OUTPUT NOT – OUT and OUT NOT

OUTPUT – OUT

Ladder Symbol

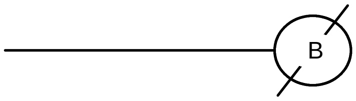


Operand Data Areas

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 *4-6-7 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.

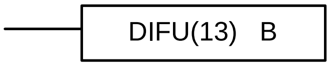
The length of time that a bit is ON or OFF can be controlled by combining the OUT or OUT NOT with TIM. Refer to Examples under *5-10-1 TIMER – TIM* for details.

Flags

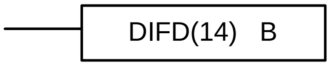
There are no flags affected by these instructions.

5-6-2 DIFFERENTIATE UP and DOWN – DIFU(13) and DIFD(14)

Ladder Symbols **Operand Data Areas**



B: Bit
IR, AR, HR, LR



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 the instruction 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 the instruction 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 *5-7 INTERLOCK and INTERLOCK CLEAR – IL(02) and ILC(03)*, *5-8 JUMP and JUMP END – JMP(04) and JME(05)*, and *5-18 Subroutines* for details.

Example 1: Use When There Are No Differentiated Instructions

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.

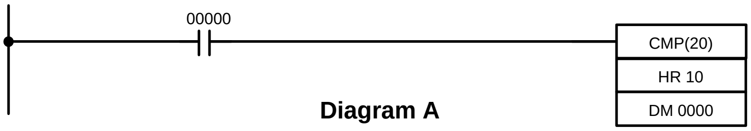


Diagram A

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

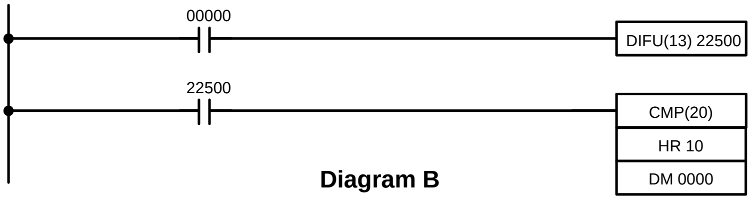
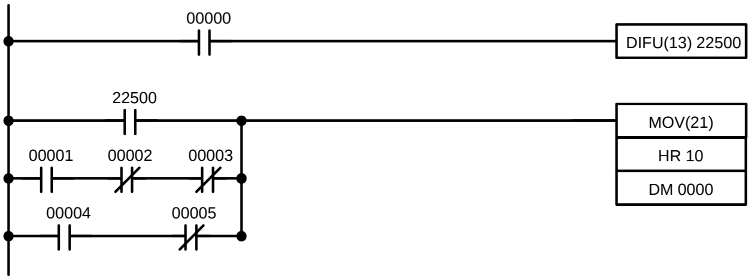


Diagram B

Address	Instruction	Operands
00000	LD	00000
00001	DIFU(13)	22500
00002	LD	22500
00003	CMP(20)	
		HR 10
		DM 000

Example 2: Use to Simplify Programming

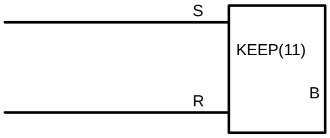
Although a differentiated form of MOV(21) is available, the following diagram would be very complicated to draw using it because only one of the conditions determining the execution condition for MOV(21) requires differentiated treatment.



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

5-6-3 KEEP – KEEP(11)

Ladder Symbol



Operand Data Areas

B: Bit
IR, AR, HR, LR

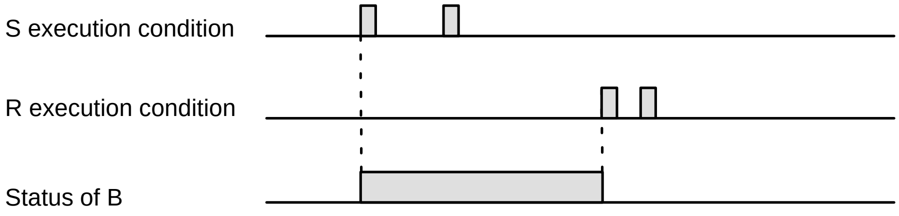
Limitations

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

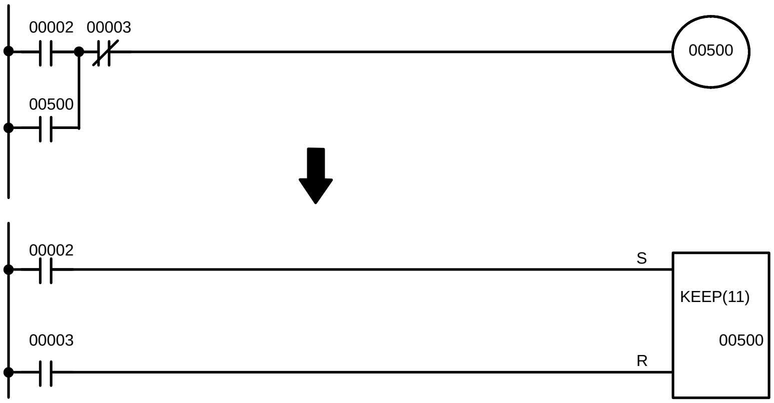
Description

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 set, regardless of whether R stays ON or goes OFF. The relationship between execution conditions and KEEP(11) bit status is shown below.



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

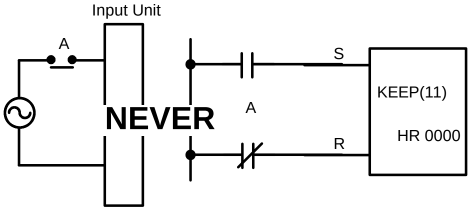
Address	Instruction	Operands
00000	LD	00002
00001	LD	00003
00002	KEEP(11)	00500

Flags

There are no flags affected by this instruction.

Precautions

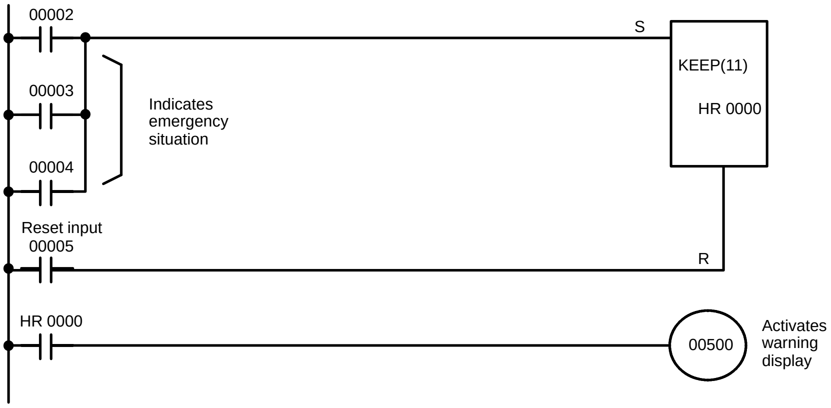
Never use an input bit in an inverse condition on the reset (R) for KEEP(11) when the input device uses an AC power supply. The delay in shutting down the PC's DC power supply (relative to the AC power supply to the input device) can cause the designated bit of KEEP(11) to be reset. This situation is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



Bits used in KEEP are not reset in interlocks. Refer to the *5-7 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.

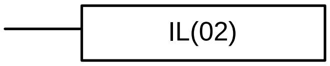


Address	Instruction	Operands	
00000	LD		00002
00001	OR		00003
00002	OR		00004
00003	LD		00005
00004	KEEP(11)	HR	0000
00005	LD	HR	0000
00006	OUT		00500

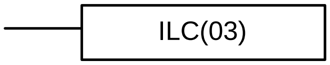
KEEP(11) can also be combined with TIM to produce delays in turning bits ON and OFF. Refer to *5-10-1 TIMER – TIM* for details.

5-7 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 *4-6-7 Branching Instruction Lines* for basic descriptions of both methods.

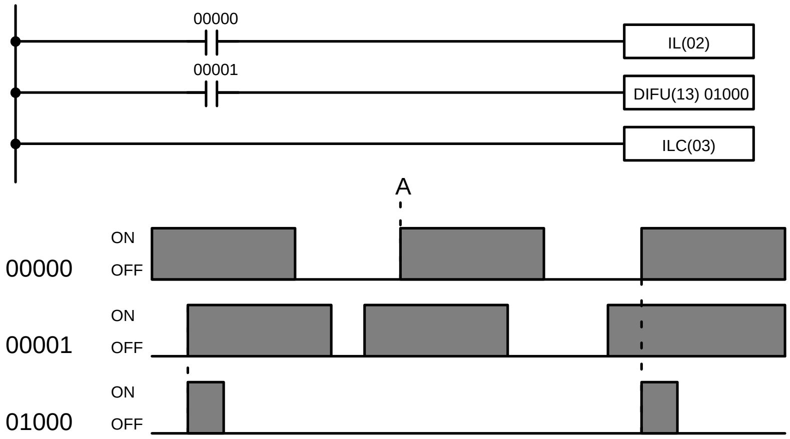
If the execution condition for IL(02) is OFF, the interlocked section between IL(02) and ILC(03) will be treated as shown in the following table:

Instruction	Treatment
OUT and OUT NOT	Designated bit turned OFF.
TIM and TIMH(15)	Reset.
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.

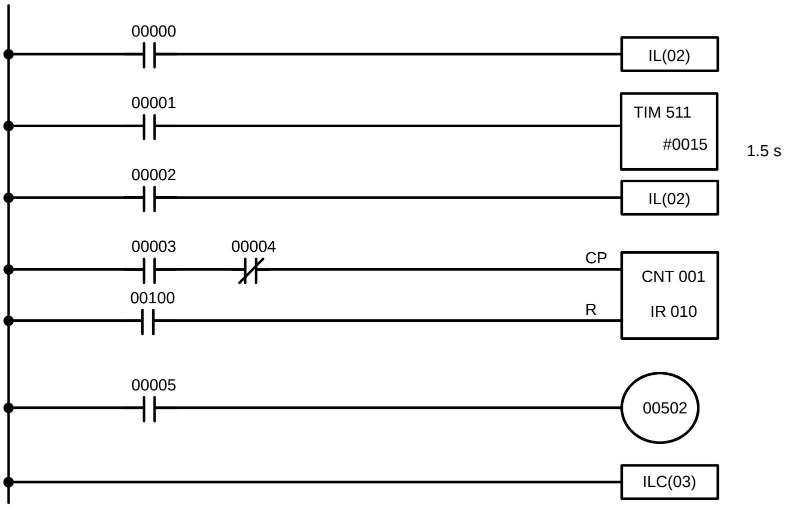
When more than one IL(02) is used with a single ILC(03), an error message will appear when the program check is performed, but execution will proceed normally.

Flags

There are no flags affected by these instructions.

Example

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



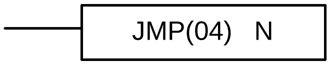
Address	Instruction	Operands
00000	LD	00000
00001	IL(02)	
00002	LD	00001
00003	TIM	511
		# 0015
00004	LD	00002
00005	IL(02)	
00006	LD	00003
00007	AND NOT	00004
00008	LD	00100
00009	CNT	001
		010
00010	LD	00005
00011	OUT	00502
00012	ILC(03)	

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.

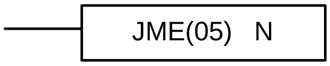
When the execution conditions for both the IL(02) are ON, the program will execute as written.

5-8 JUMP and JUMP END – JMP(04) and JME(05)

Ladder Symbols **Definer Values**



N: Jump number
(00 to 49)



N: Jump number
(00 to 49)

Limitations

Jump numbers 01 through 49 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 49, 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 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 49 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 *4-6-8 Jumps*.

5-9 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.

5-10 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.

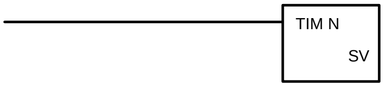
TC numbers can be designated as operands that require either bit or word data. When designated as an operand that requires bit data, the TC number accesses a bit that functions as a “completion flag” that indicates when the time/count has expired, i.e., the bit, which is normally OFF, will turn ON when 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.

5-10-1 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 000.1 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.

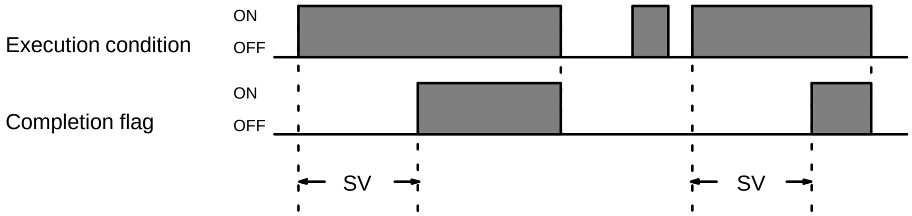
TC 000 through TC 003 should not be used in TIM if they are required for TIMH(15). Refer to *5-10-2 HIGH-SPEED TIMER – TIMH(15)* for details.

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. TIM accuracy is +0.0/–0.1 second.

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 *5-10-3 COUNTER – CNT* for details.

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

Flags

ER: SV is not in BCD.

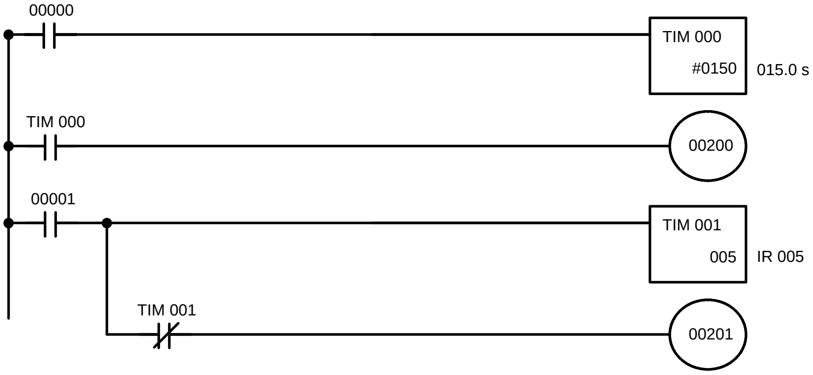
Indirectly addressed DM word is non-existent. (Content of *DM word is not BCD, or the DM area boundary has been exceeded.)

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

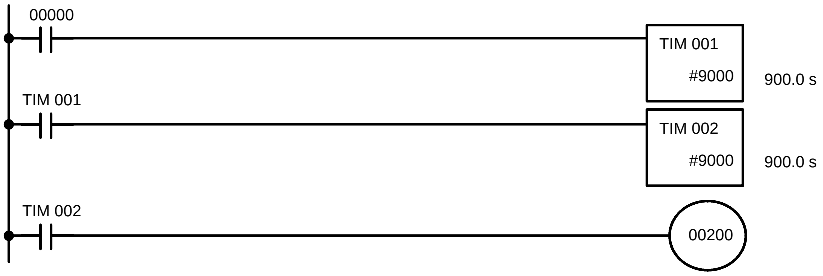
The following example shows two timers, one set with a constant and one set via input word 005. Here, 00200 will be turned ON after 00000 goes ON and stays ON for at least 15 seconds. When 00000 goes OFF, the timer will be reset and 00200 will be turned OFF. When 00001 goes ON, TIM 001 is started from 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		00201

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.

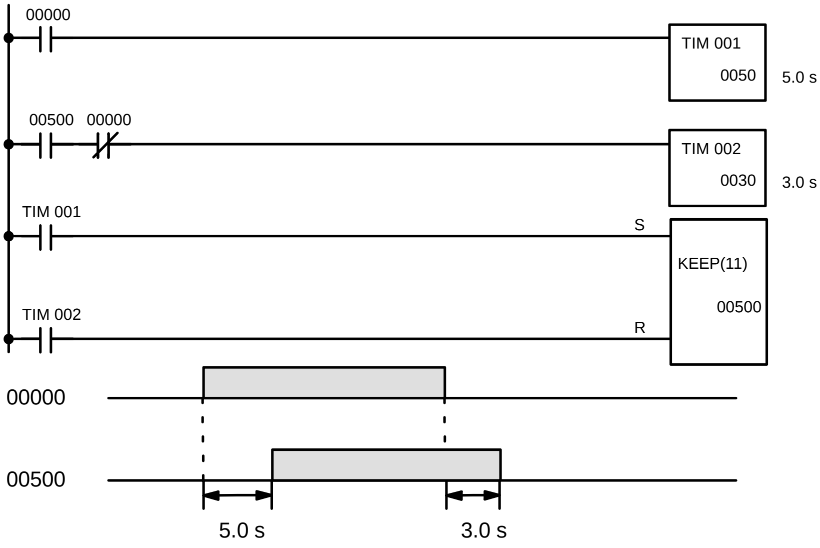
TIM can also be combined with CNT or CNT can be used to count SR area clock pulse bits to produce longer timers. An example is provided in *5-10-3 COUNTER – CNT*.

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 *5-6-3 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.

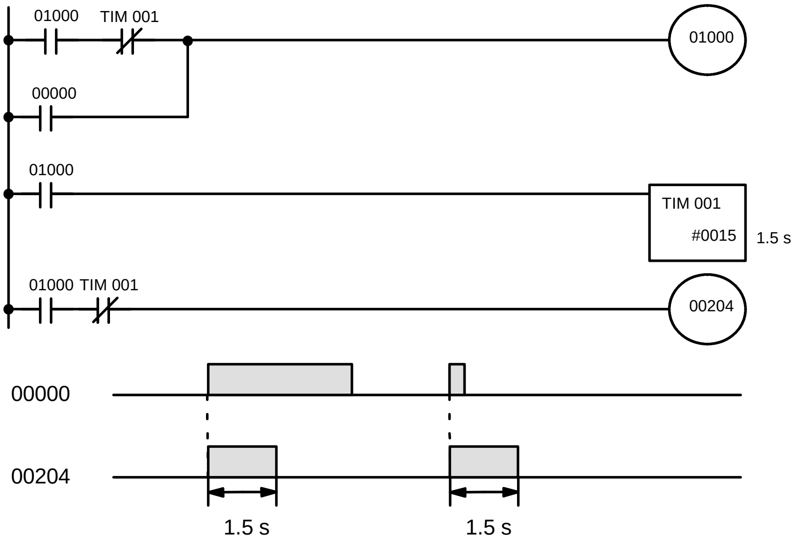
In the following example, 00500 would be turned ON 5.0 seconds after 00000 goes ON and then turned OFF 3.0 seconds after 00000 goes OFF. It is necessary to use both 00500 and 00000 to determine the execution condition for TIM 002; 00000 in an normally closed condition is necessary to reset TIM 002 when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

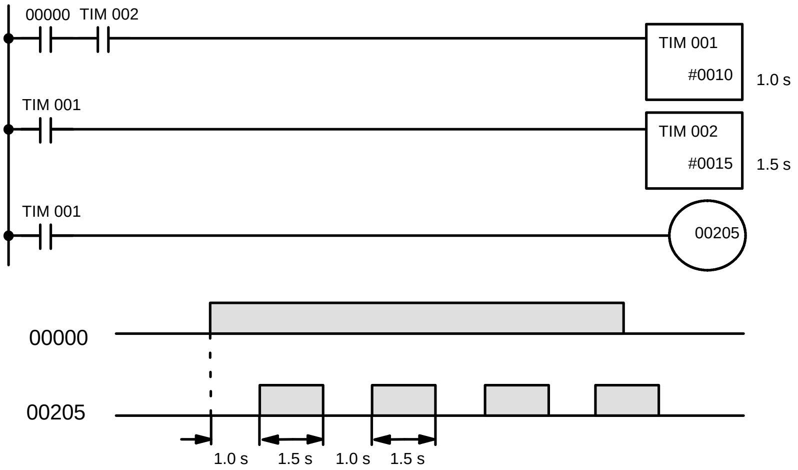
The length of time that a bit is kept ON or OFF can be controlled by combining TIM with OUT or OUT NO. The following diagram demonstrates how this is possible. In this example, 00204 would remain ON for 1.5 seconds after 00000 goes ON regardless of the time 00000 stays ON. This is achieved by using 01000 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

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.

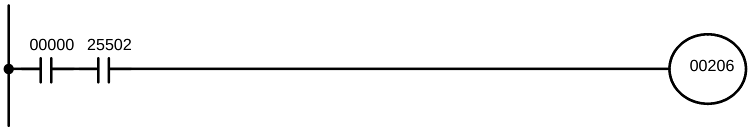


Address	Instruction	Operands	
00000	LD		00000
00001	AND	TIM	002
00002	TIM		001
		#	0010
00003	LD	TIM	001
00004	TIM		002
		#	0015
00005	LD	TIM	001
00006	OUT		00205

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

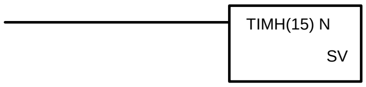
0.5 seconds and OFF for 0.5 seconds. Precise timing and the initial status of 00206 would depend on the status of the clock pulse when 00000 goes ON.



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

5-10-2 HIGH-SPEED TIMER – TIMH(15)

Ladder Symbol



Definer Values

N: TC number
(000 through 511, but 000 through 003 preferred)

Operand Data Areas

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

Limitations

SV is between 00.02 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.

TC 000 through TC 003 must be used to ensure adequate accuracy if the cycle time is greater than 10 ms.

Description

TIMH(15) operates in the same way as TIM except that TIMH measures in units of 0.01 second.

Refer to *5-10-1 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 *5-10-3 COUNTER – CNT* for details.

The cycle time affects TIMH(15) accuracy if TC 004 through TC 511 are used. If the cycle time is greater than 10 ms, use TC 000 through TC 003.

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

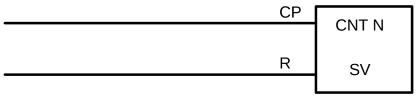
Flags

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.)

5-10-3 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

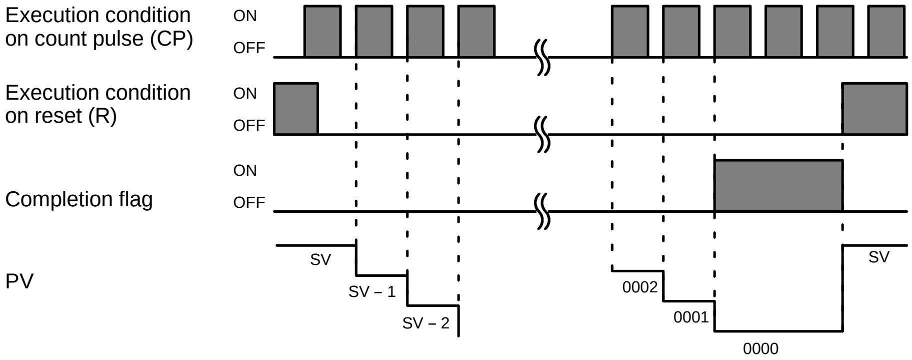
Each TC number can be used as the definer in only one TIMER or COUNTER instruction.

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.


Execution condition on count pulse (CP)
ON
OFF
Execution condition on reset (R)
ON
OFF
ON
Completion flag
OFF
PV
SV
SV – 1
SV – 2
0002
0001
0000
SV

Precautions

Program execution will continue if a non-BCD SV is used, but the SV might not be correct.

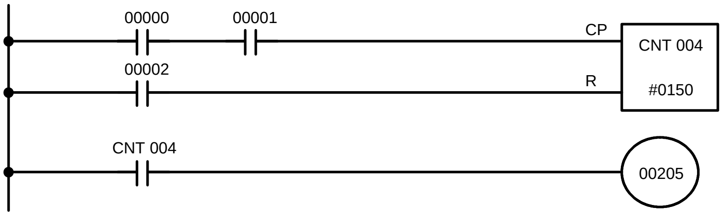
Flags

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

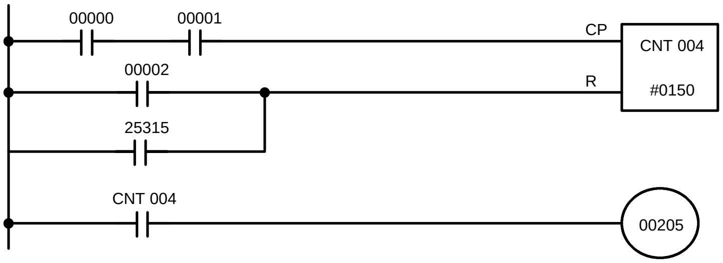
In the following example, the PV will be decremented whenever both 00000 and 00001 are ON provided that 00002 is OFF and either 00000 or 00001 was OFF the last time CNT 004 was executed. When 150 pulses have been counted down (i.e., when PV reaches zero), 00205 will be turned ON.



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

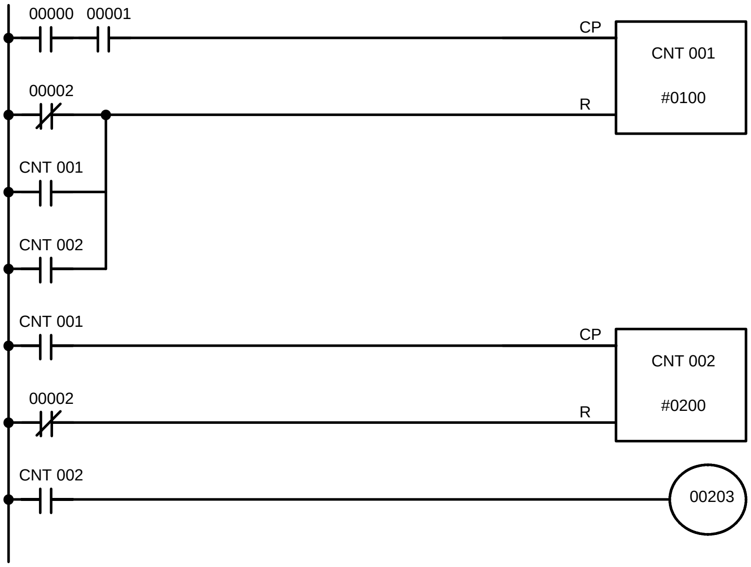
Example 2: Extended Counter

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

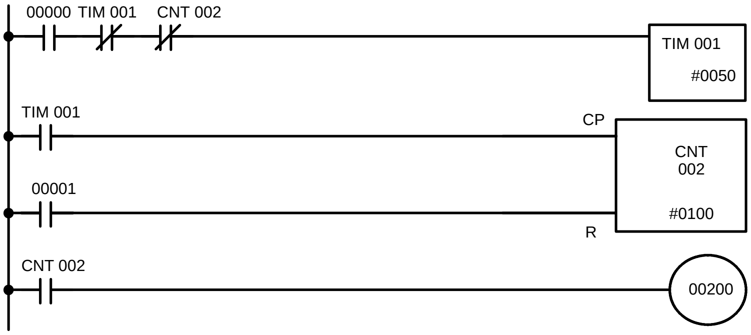
CNT can be used in sequence as many times as required to produce counters capable of counting any desired values.

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 is 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 between 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.

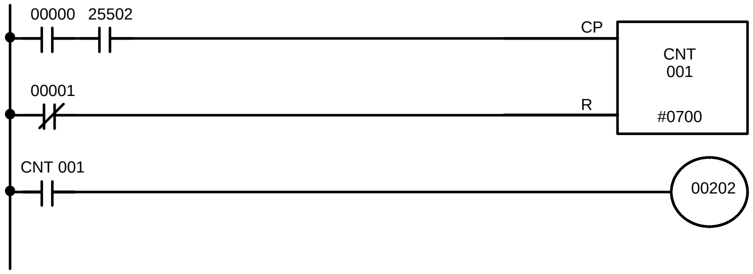
Because in this example the SV for TIM 001 is 5.0 seconds and the SV for CNT 002 is 100, the completion flag for CNT 002 turns ON when 5 seconds x 100 times, i.e., 500 seconds (or 8 minutes and 20 seconds) have expired. This would result in 00201 being turned ON.



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		00200

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.

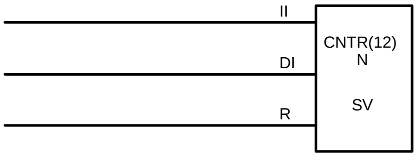


Address	Instruction	Operands
00000	LD	00000
00001	AND	25502
00002	LD NOT	00001
00003	CNT	001
		# 0700
00004	LD	CNT 001
00005	OUT	00202

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.

5-10-4 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).

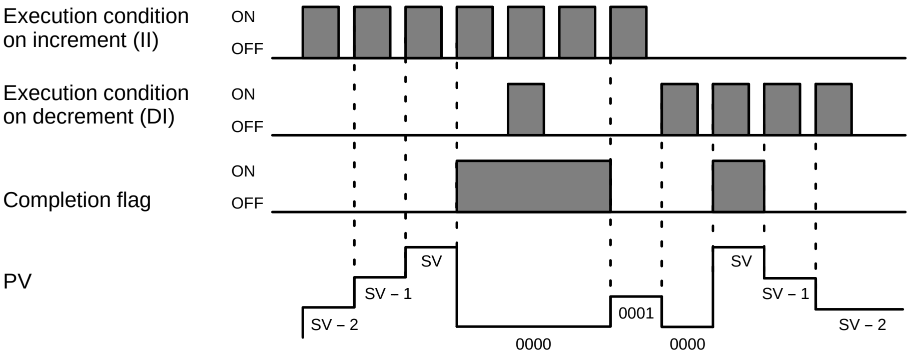
The present value (PV) will be incremented by one whenever CNTR(12) is executed with an ON execution condition for II and the last execution condition for II was OFF. The present value (PV) will be decremented by one whenever CNTR(12) is executed with an ON execution condition for DI and the last execution 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.


Execution condition on increment (II)
ON
OFF
Execution condition on decrement (DI)
ON
OFF
ON
Completion flag
OFF
PV
SV
SV – 1
SV – 2
0000
0001
0000
SV
SV – 1
SV – 2

Precautions

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

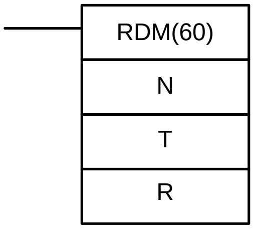
Flags

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.)

5-10-5 REVERSIBLE DRUM COUNTER – RDM(60)

Ladder Symbol



Operand Data Areas

N: TC number
TC (TC 500 to TC 511)

T: Beginning table word (BCD)
IR, AR, DM, HR, TC, LR

R: Beginning result word
IR, AR, DM, HR, TC, LR

Limitations

TC 500 through TC 511 must be used to create a reversible counter. The table starting in T must be within the same data area and all table words from T+1 on must be in BCD. Set this data carefully; the Error Flag (SR 25503) will not detect non-BCD data.

Description

RDM(60) is used to create a reversible ring counter that counts from 0 to 9999, compare the PV to a table of ranges, and turn ON corresponding bits in R whenever the PV is within a range in the table. The counter input is the execution condition of RDM(60). Inputs are counted on the rising edge.

The size of the table is determined by the value of n contained in bits 00 to 07 of T. n is one less than the number of ranges in the table and can be anywhere from 0 to 255, designating from 1 to 256 ranges. The actual table of ranges begin in T+1. Each range is determined by the BCD values in consecutive word pairs, with each pair beginning with T+1. Ranges should be set so that the first value is less than the second unless the range includes zero, e.g., if the range includes 9998, 9999, 0000, 0001, and 0002, the first value should be set to 9998; the second value to 0002.

When the counter value is found to lie within the first range, the first bit in R, bit 00, will be turned ON; if the counter value is within the second range, bit 01 will be turned ON. Results for more than the first 16 ranges are placed in consecutive result words from R+1 on. The correspondence between table words and result words is shown below.

First word	Second word	Result bit
T+1	T+2	00 of R
T+3	T+4	01 of R
T+5	T+6	02 of R
T+7	T+8	03 of R
T+9	T+10	04 of R
T+11	T+12	05 of R
T+13	T+14	06 of R
T+15	T+16	07 of R
T+17	T+18	08 of R
T+19	T+20	09 of R
T+21	T+22	10 of R
T+23	T+24	11 of R
T+25	T+26	12 of R
T+27	T+28	13 of R
T+29	T+30	14 of R
T+31	T+32	15 of R
T+33	T+34	00 of R+1
T+35	T+36	01 of R+1
T+37	T+38	02 of R+1
etc.	etc.	etc.
T+2n+1	T+2n+2	See below.

The last bit required for the result will be the truncated integer quotient of (n–1)/15, with one less than the remainder being the bit number of the last required bit.

Result words are updated according to the present value of the counter regardless of whether or not the Reset Bit (see below) is ON, i.e., then the Reset Bit is ON, the results words will be updated according to a present value of zero.

Table values can be changed during counter operation. Operation will continue according to the new values.

The direction of the counter and resetting the counter are designated via AR bits. Each of the eleven possible counters has a dedicated Reset Bit and Counter Direction Bit. The Reset Bits are contained in AR 0200 to AR 0210; the Counter Direction Bits, in AR 0300 to AR 0310. These correspond to the TC numbers as follows:

TC number	Reset Bit	Counter Direction Bit
CNT 500	AR 0200	AR 0300
CNT 501	AR 0201	AR 0301
CNT 502	AR 0202	AR 0302
CNT 503	AR 0203	AR 0303
CNT 504	AR 0204	AR 0304
CNT 505	AR 0205	AR 0305
CNT 506	AR 0206	AR 0306
CNT 507	AR 0207	AR 0307
CNT 508	AR 0208	AR 0308
CNT 509	AR 0209	AR 0309
CNT 510	AR 0210	AR 0310

When the corresponding Reset Bit is ON, the counter will be reset to zero and will not operate until the Reset Bit goes OFF. When the corresponding Counter Direction Bit is ON, the counter will count down; when the Counter Direction Bit is OFF, the counter will count up.

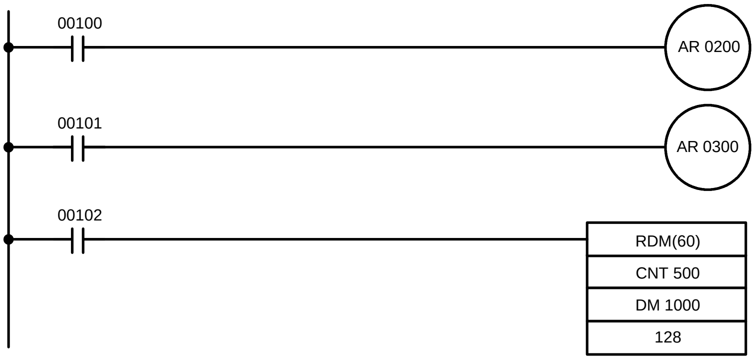
Flags

ER: T, T+1, and T+2 are not within the same data area.

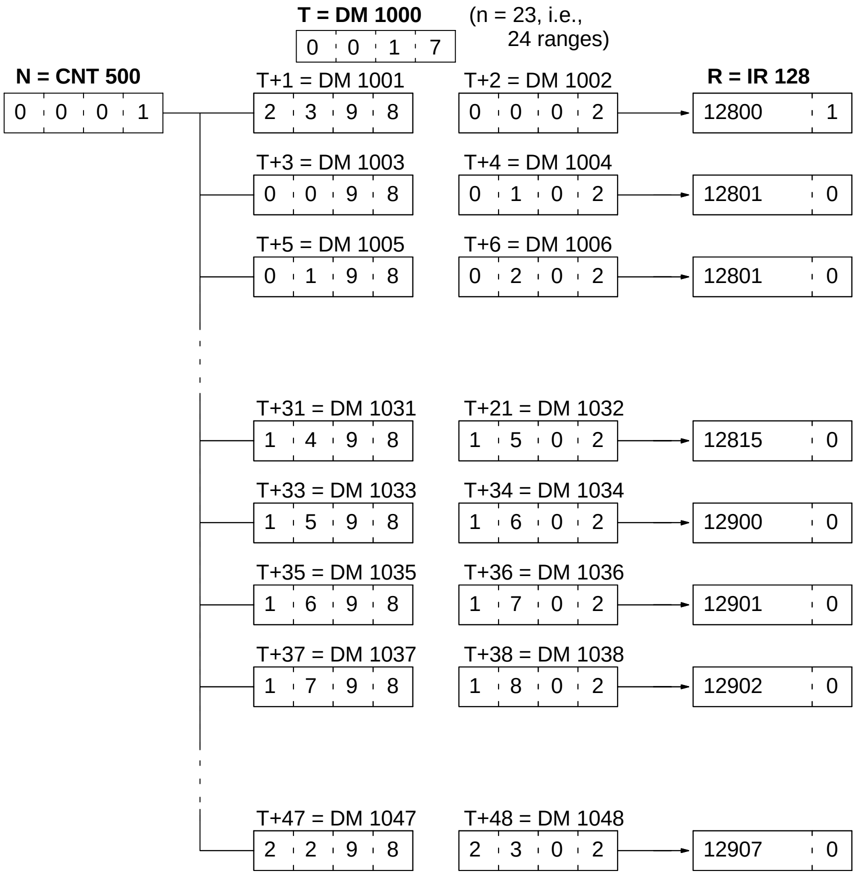
Indirectly addressed DM word is non-existent. (Content of *DM word is not BCD, or the DM area boundary has been exceeded.)

Example

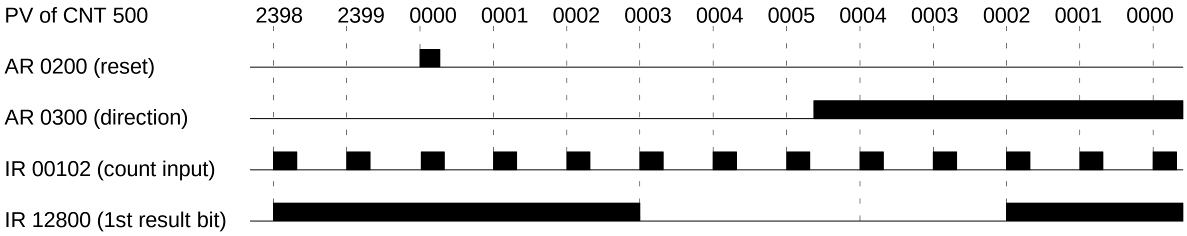
The following example shows the minimal programming required to use RDM(60). Both the Reset Bit and the Counter Direction Bit are control via input bits, i.e., IR 00100 and IR 00101. The table values and result word status for a PC of 0001 are shown. It is assumed that IR 00100 would be turned ON to reset the counter at 2,400 counts.



Address	Instruction	Operands
00000	LD	00100
00001	OUT	AR 0200
00002	LD	00101
00003	OUT	AR 0300
00004	LD	00102
00005	RDM(60)	---
		CNT 500
		DM 1000
		128

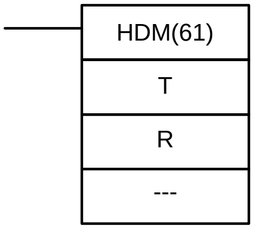


The following timing chart shows counter operation for a limited range of PVs. Although only IR 12800 is shown because it is the only result bit affected by the PVs shown, the other bits in result words would operate the same for PVs that lie within the corresponding table ranges.



5-10-6 HIGH-SPEED COUNTER – HDM(61)

Ladder Symbol



Operand Data Areas

T: Beginning table word (BCD)

IR, AR, DM, HR, TC, LR

R: Beginning result word

IR, AR, DM, HR, TC, LR

---: Not used.

Limitations

The table starting in T must be within the same data area and all table words from T+1 on must be in BCD. Set this data carefully; the Error Flag (SR

25503) will not detect non-BCD data. Ensure that the system parameters have been set to enable the High-speed Counter. See the table in *3-6 DM (Data Memory) Area*.

Description

HDM(61) is used to compare the PV of the high-speed counter (CNT 511) to a table of ranges, and turn ON corresponding bits in R whenever the PV is within a range in the table.

The size of the table is determined by the value of n contained in bits 00 to 07 of T. The value of n is one less than the number of ranges in the table and can be between 0 and 255, designating from 1 to 256 ranges. The actual table of ranges begin in T+1. Each range is determined by the BCD values in consecutive word pairs, with each pair beginning with T+1. Ranges should be set so that the first value is less than the second unless the range includes zero, e.g., if the range includes 9998, 9999, 0000, 0001, and 0002, the first value should be set to 9998; the second value to 0002.

When the counter value is found to lie within the first range, the first bit in R, bit 00, will be turned ON; if the counter value is within the second range, bit 01 will be turned ON; and so on. Results for more than the first 16 ranges are placed in consecutive result words from R+1 on. The correspondence between table words and result words is shown below.

First word	Second word	Result bit
T+1	T+2	00 of R
T+3	T+4	01 of R
T+5	T+6	02 of R
T+7	T+8	03 of R
T+9	T+10	04 of R
T+11	T+12	05 of R
T+13	T+14	06 of R
T+15	T+16	07 of R
T+17	T+18	08 of R
T+19	T+20	09 of R
T+21	T+22	10 of R
T+23	T+24	11 of R
T+25	T+26	12 of R
T+27	T+28	13 of R
T+29	T+30	14 of R
T+31	T+32	15 of R
T+33	T+34	00 of R+1
T+35	T+36	01 of R+1
T+37	T+38	02 of R+1
etc.	etc.	etc.
T+2n+1	T+2n+2	See below.

The last bit required for the result will be the truncated integer quotient of (n–1)/15, with one less than the remainder being the bit number of the last required bit.

Result words are updated according to the present value of the counter regardless of whether or not the High-speed Counter Reset/Disable Bit is ON, i.e., then the Reset/Disable Bit is ON, the result words will be updated according to a present value of zero. If the Reset/Disable Bit is ON, the high-speed counter is reset when END(01) is executed.

Although result words will be updated only when HDM(61) is executed with an ON execution condition, the execution condition for HDM(61) does not affect the operation of the high-speed counter.

Table values can be change during counter operation. Operation will continue according to the new values.

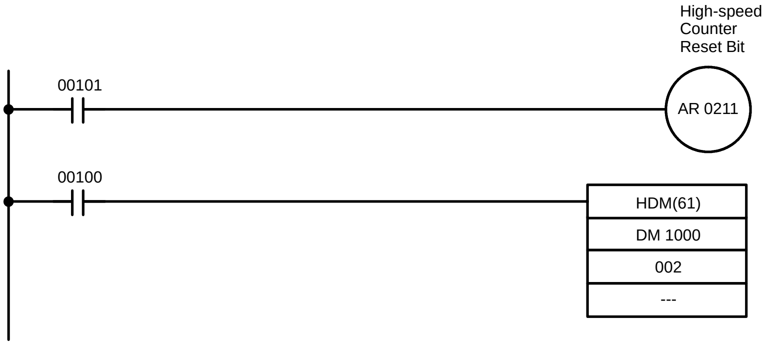
Flags

ER: T, T+1, and T+2 are not within the same data area.

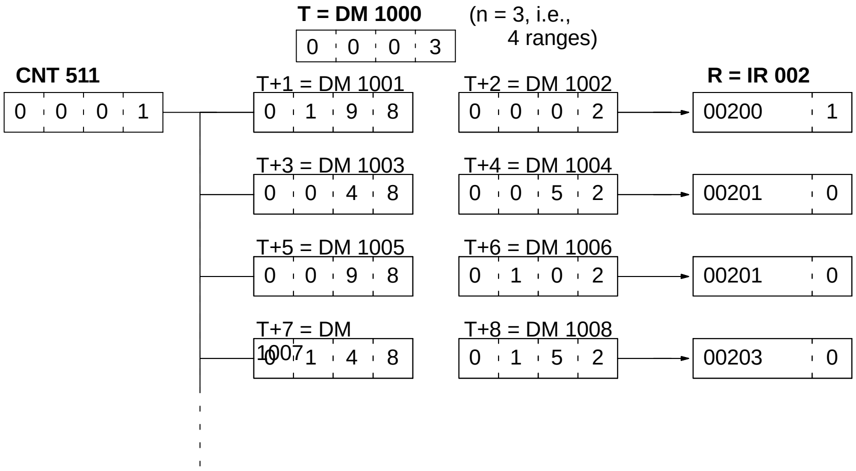
Indirectly addressed DM word is non-existent. (Content of *DM word is not BCD, or the DM area boundary has been exceeded.)

Example 1

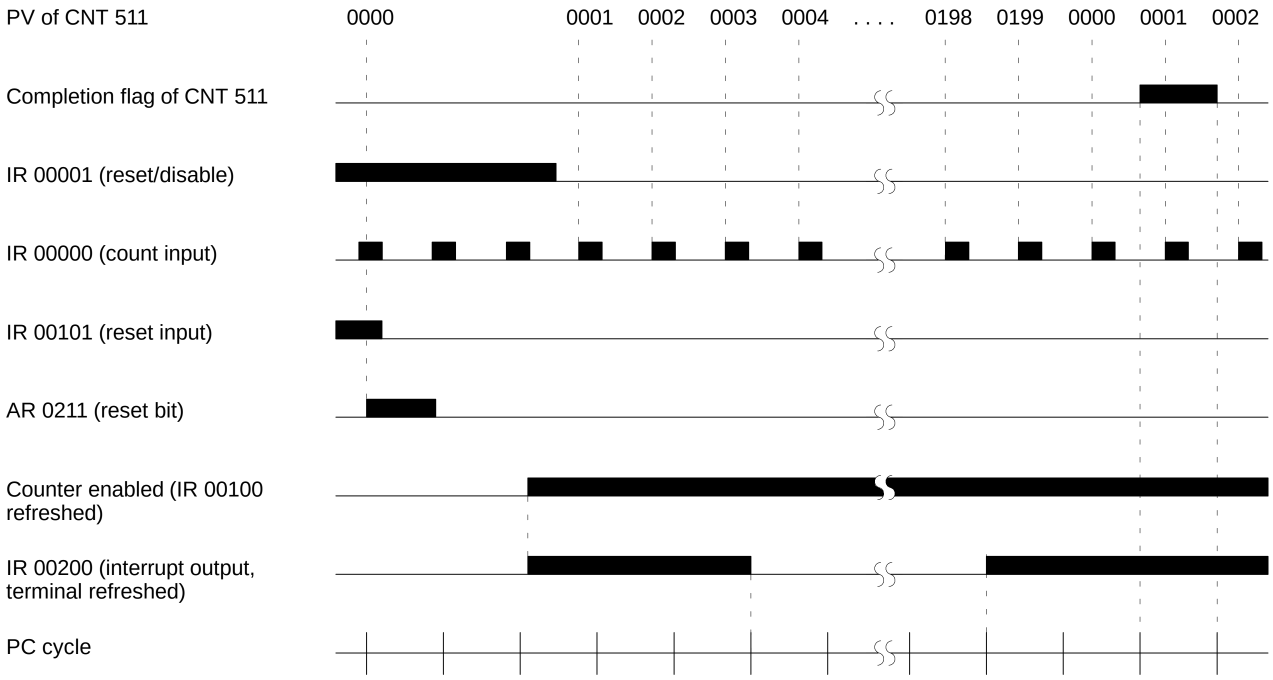
The following example shows the programming required to used HDM(61). Both the Reset Bit and the Counter Direction Bit are control via input bits, i.e., IR 00100 and IR 00101. The table values and result word status for a PC of 0001 are shown. It is assumed that IR 00100 would be turned ON to reset the counter at 2,400 counts.



Address	Instruction	Operands	
00000	LD		00101
00001	OUT	AR	0211
00002	LD		00100
00003	HDM(61)		---
		DM	1000
			002



The following timing chart shows counter operation for a limited range of PVs. Although only IR 12800 is shown because it is the only result bit affected by the PVs shown, the other bits in result words would operated the same for PVs that lie within the corresponding table ranges.



*The above timing chart does not take I/O circuit delays into consideration.

Example 2

The following example shows how to program a high-speed counter that counts past 9,999. In this particular example, a comparison table to 35,000 is created to detect a range from 23,000 to 25,000 counts.

DM Area Settings

The General High-speed Counter Bit settings in the DM area (DM0905) are as follows:

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 09	Bit 08	Bits 00 to 07
1	0 or 1	0	0	0	0	1	1	0 or 1

Turning ON bit 15 enables the counter, and the above settings for bits 08 to 10 set steps 0 to 3 (0 to 40,000) for bank 0.

Step Table Settings

The settings for the step table are as follows:

Word	Contents
DM 0910	0000
DM 0911	0000
DM 0912	0000
DM 0913	5000

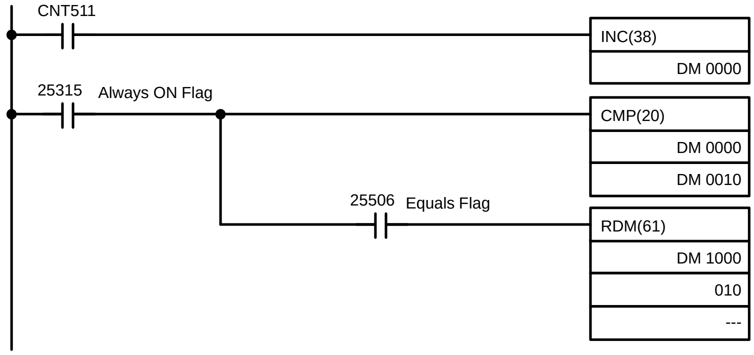
Data

The following data is used.

Word	Contents
DM 0010	0002
DM 1000	0000
DM 1001	3000
DM 1002	5000

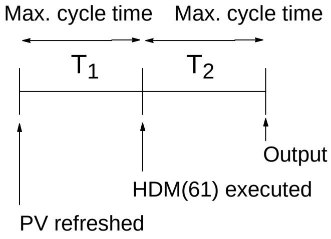
Programming

In the following program section, the INC(38) increments DM 0000 when each step is completed thus allowing CMP(20) to activate the high-speed counter for steps one and two. The high-speed counter is thus used to compare only the rightmost four digits and INC(38) and CMP(20) are combined to control execution of RDM(61) through the values of the leftmost digits.



Refreshing Result Words

Refreshing of results words (beginning with R) is influenced by the cycle time. The time required from refreshing of the PV of the high-speed counter and refreshing of a result word bit is as shown below. As shown, it is necessary to reduce the cycle time as much as possible when rapid response is desired in result words.



The second part of this delay time, T_2, can be eliminated by using IORF(97) to refresh the desired I/O terminals during program execution.

An interrupt drum output can also be used with HDM(61) to reduce the delay time (normally T_1 + T_2) to 1.5 ms max. Refer to *3-8-1 High-speed Counter* for details.

Output Delays

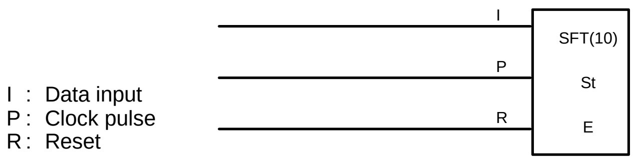
Relay outputs are standard on the C20H/C28H/C40H/C60H. When used with high-speed counters (e.g., in a simple positioning system), the 15 ms max. relay’s ON/OFF delay can interfere with operation. If this delay is a problem, the outputs can be easily replaced with transistor outputs, which have a delay of 1.5 ms. Refer to the *Installation Guide* for details.

5-11 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.

5-11-1 SHIFT REGISTER – SFT(10)

Ladder Symbol



Operand Data Areas

St: Starting word
IR, AR, HR, LR

E: End word
IR, AR, HR, LR

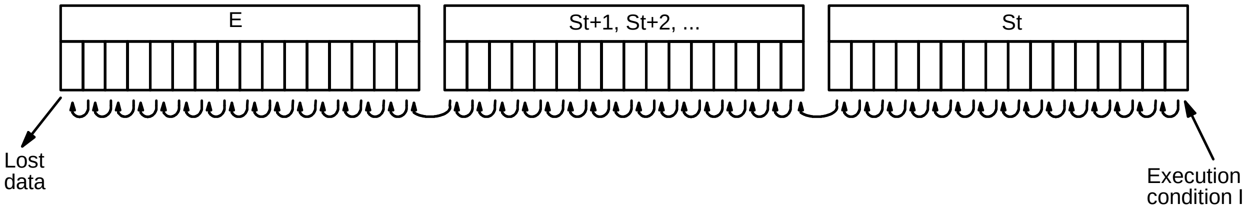
Limitations

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 such as KEEP (11) that controls individual bit status, an error (“COIL DUPL”) will be generated when program syntax is checked on a Programming Device other than the Programming Console. The program, however, will be executed as written. See *Example 2: Controlling Bits in Shift Registers* for a programming example that does this.

Description

If I is ON at the leading edge of P, then a logical 1 is placed in bit 00 of word St. If I is 0 at leading edge of P, the a logical 0 is put in bit 00 of word St. In either case, all bits are shifted one bit position to the left between St and E.



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.

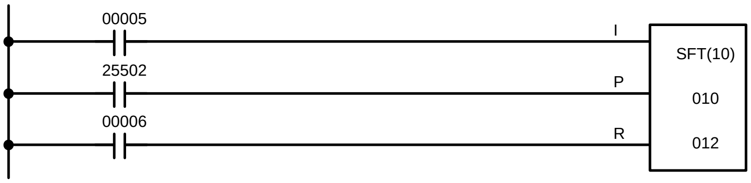
When execution condition R goes ON, all bits in the shift register will be turned OFF (i.e., set to 0) and the shift register will not operate until R goes OFF again.

Flags

There are no flags affected by SFT(10).

Example 1: Basic Application

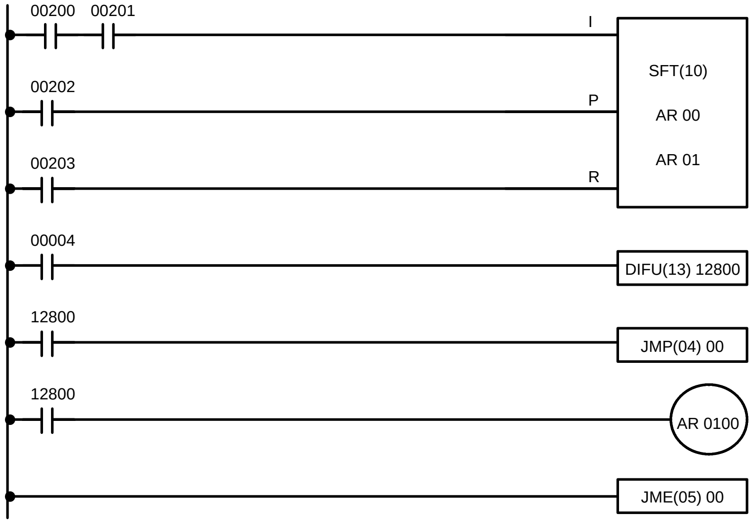
The following example uses the 1-second clock pulse bit (25502) so that the execution condition produced by 00005 is shifted into a 3-word register between IR 010 and IR 012 every second.



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.



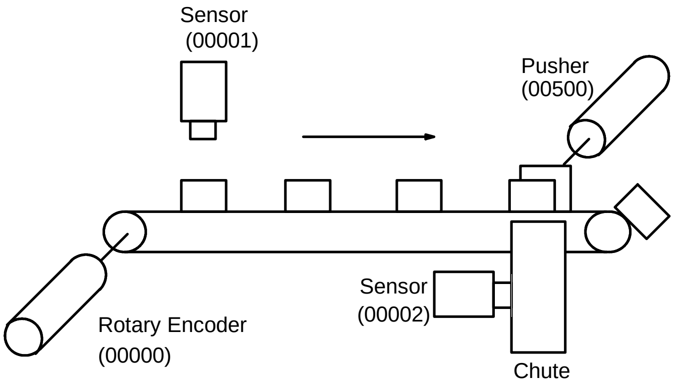
Address	Instruction	Operands	
00000	LD		00200
00001	AND		00201
00002	LD		00202
00003	LD		00203
00004	SFT(10)		
		AR	00
		AR	01
00005	LD		00004
00006	DIFU(13)		12800
00007	LD		12800
00008	JMP(04)		00
00009	LD		12800
00010	OUT	AR	0100
00011	JME(05)		00

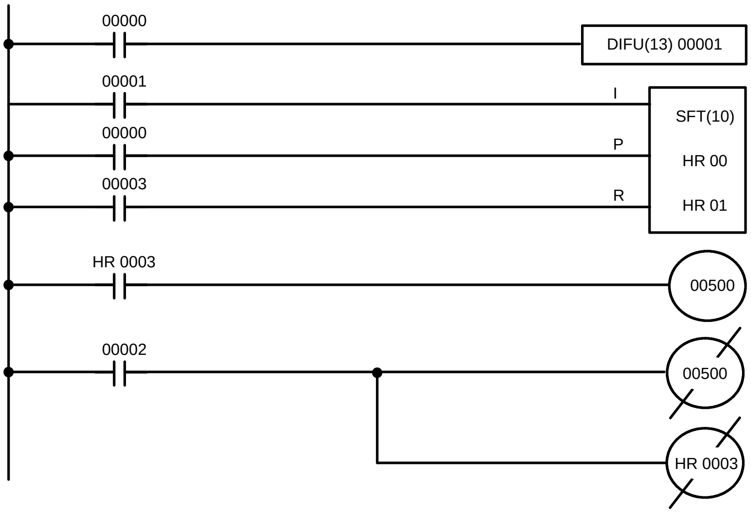
When a bit that is part of a shift register is used in OUT (or any other instruction that controls bit status), a syntax error will be generated during the program check, but the program will executed properly (i.e., as written).

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 chute. 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 chute so that the pusher output and HR 0003 of the shift register can be reset as required.

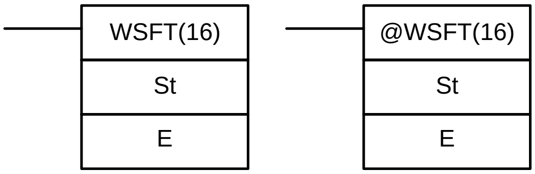




Address	Instruction	Operands	
00000	LD		00000
00001	DIFU(13)		00001
00002	LD		00001
00003	LD		00000
00004	LD		00003
00005	SFT(10)		
		HR	00
		HR	01
00006	LD	HR	0003
00007	OUT		00500
00008	LD		00002
00009	OUT NOT		00500
00010	OUT NOT	HR	0003

5-11-2 WORD SHIFT – WSFT(16)

Ladder Symbols



Operand Data Areas

St: Starting word
IR, AR, DM, HR, LR

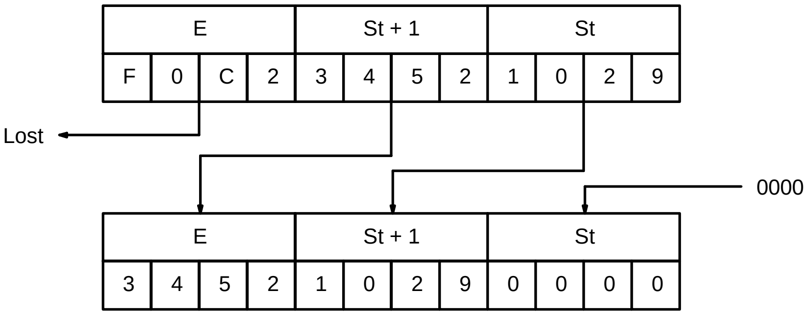
E: End word
IR, AR, DM, HR, LR

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, 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.



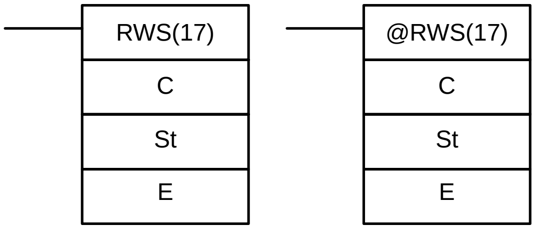
Flags

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.)

5-11-3 REVERSIBLE WORD SHIFT – RWS(17)

Ladder Symbols



Operand Data Areas

C: Control word

IR, AR, DM, HR, TC, LR, #

St: Start word

IR, AR, DM, HR, TC, LR

E: End word

IR, 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.

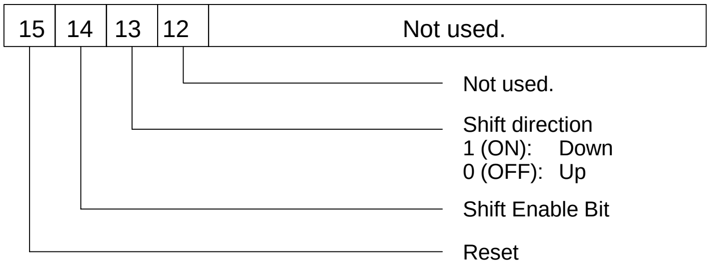
Description

When executed with an OFF execution condition, RWS(17) does nothing and the next instruction line is moved to. When executed with an ON execution condition, RWS(17) is used to create and control a reversible non-synchronous word shift register between St and E. This shift 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 the non-zero data 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 RWS(17) is executed with bit 15 ON. Turn bit 15 OFF for normal operation.



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 instruction is used to shift words in an 11-word shift register created between DM 0100 and DM 0110 assuming that HR 1215 (the Reset Bit in the control word) is OFF. If HR 1215 is ON, the entire register would be set to 0000. The data changes that would occur for the given register and control word contents are also shown.

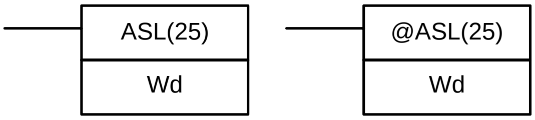
HR 1213: OFF (Shift upward)
HR 1214: ON (Shift enabled)
HR 1215: OFF (Reset OFF)

	Before execution	After execution
DM 0100	1234	0000
DM 0101	0000	1234
DM 0102	0000	0000
DM 0103	2345	2345
DM 0104	3456	0000
DM 0105	0000	3456
DM 0106	4567	4567
DM 0107	5678	5678
DM 0108	6789	0000
DM 0109	0000	6789
DM 0110	789A	789A

Shift direction ↓

5-11-4 ARITHMETIC SHIFT LEFT – ASL(25)

Ladder Symbols

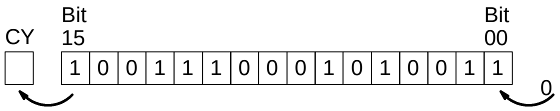


Operand Data Areas

Wd: Shift word
IR, 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

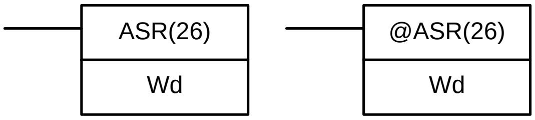
ER: Indirectly addressed DM word is non-existent. (Content of *DM word is not BCD, or the DM area boundary has been exceeded.)

CY: Receives the status of bit 15.

EQ: ON when the content of Wd is zero; otherwise OFF.

5-11-5 ARITHMETIC SHIFT RIGHT – ASR(26)

Ladder Symbols

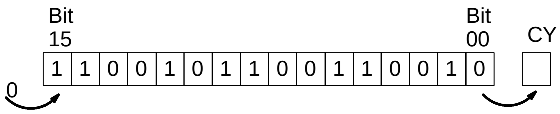


Operand Data Areas

Wd: Shift word
IR, 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

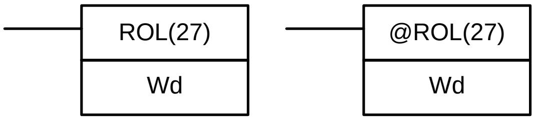
ER: Indirectly addressed DM word is non-existent. (Content of *DM word is not BCD, or the DM area boundary has been exceeded.)

CY: Receives the data of bit 00.

EQ: ON when the content of Wd is zero; otherwise OFF.

5-11-6 ROTATE LEFT – ROL(27)

Ladder Symbols

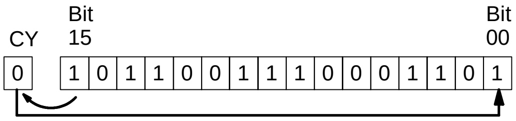


Operand Data Areas

Wd: Rotate word
IR, 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).

Flags

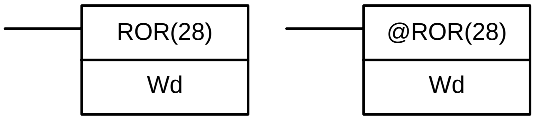
ER: Indirectly addressed DM word is non-existent. (Content of *DM word is not BCD, or the DM area boundary has been exceeded.)

CY: Receives the data of bit 15.

EQ: ON when the content of Wd is zero; otherwise OFF.

5-11-7 ROTATE RIGHT – ROR(28)

Ladder Symbols

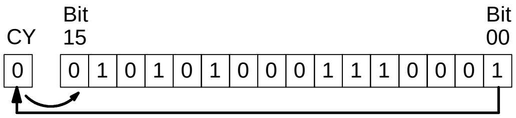


Operand Data Areas

Wd: Rotate word
IR, 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).

Flags

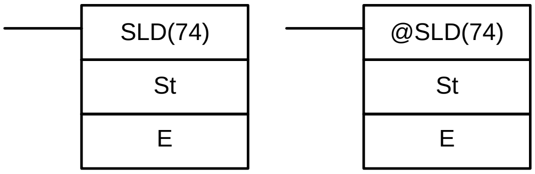
ER: Indirectly addressed DM word is non-existent. (Content of *DM word is not BCD, or the DM area boundary has been exceeded.)

CY: Receives the data of bit 15.

EQ: ON when the content of Wd is zero; otherwise OFF.

5-11-8 ONE DIGIT SHIFT LEFT – SLD(74)

Ladder Symbols



Operand Data Areas

St: Starting word
IR, AR, DM, HR, LR

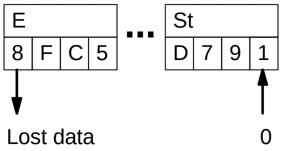
E: End word
IR, AR, DM, HR, LR

Limitations

St and E must be in the same data area, and E must be greater than or equal to St.

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

The shift operation might not be completed if a power failure occurs during shift operation.

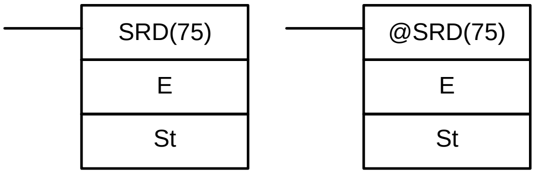
Flags

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.)

5-11-9 ONE DIGIT SHIFT RIGHT – SRD(75)

Ladder Symbols



Operand Data Areas

E: End word
IR, AR, DM, HR, LR

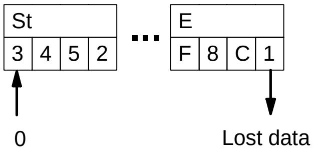
St: Starting word
IR, AR, DM, HR, LR

Limitations

St and E must be in the same data area, and E must be less than or equal to St.

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

The shift operation might not be completed if a power failure occurs during shift operation.

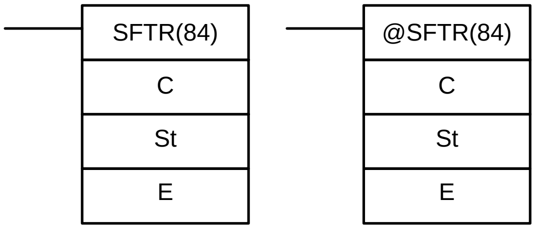
Flags

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.)

5-11-10 REVERSIBLE SHIFT REGISTER – SFTR(84)

Ladder Symbols



Operand Data Areas

C: Control word

IR, AR, DM, HR, LR

St: Starting word

IR, AR, DM, HR, LR

E: End word

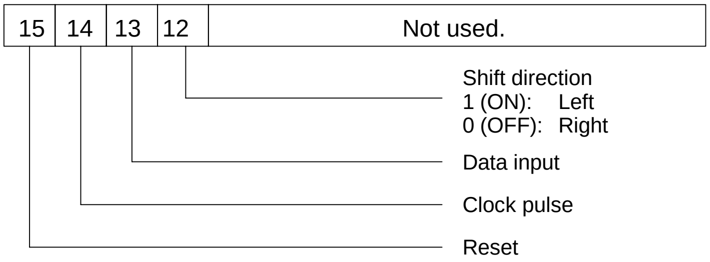
IR, AR, DM, HR, LR

Limitations

St must be less than or equal to E, and St and E must be in the same data area.

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:



Flags

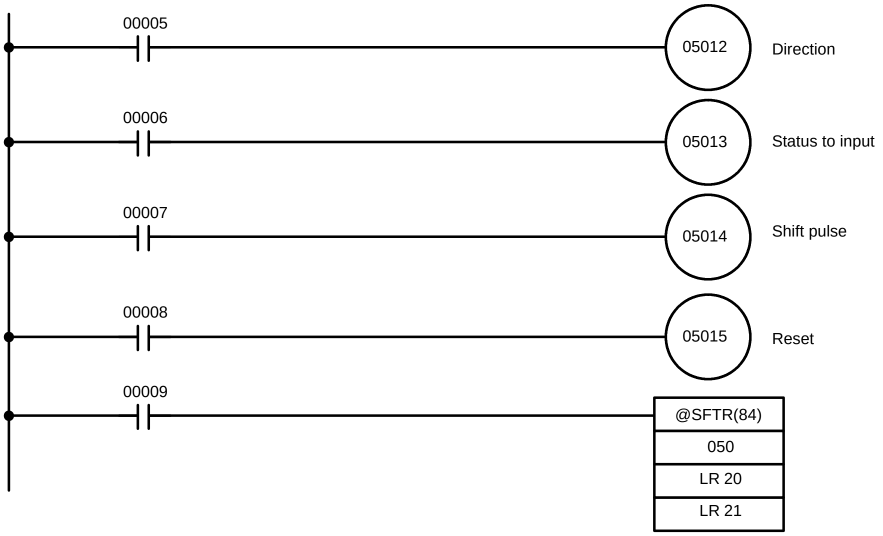
ER: St and E are not in the same data area or ST is greater than E.

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	05014
00006	LD	00008

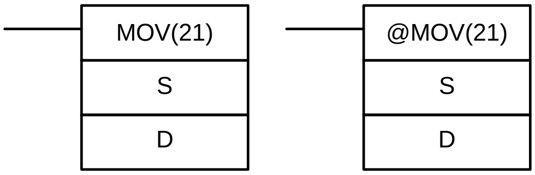
Address	Instruction	Operands
00007	OUT	05015
00008	LD	00009
00009	@SFTR(84)	
		050
		LR 20
		LR 21

5-12 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 require 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.

5-12-1 MOVE – MOV(21)

Ladder Symbols



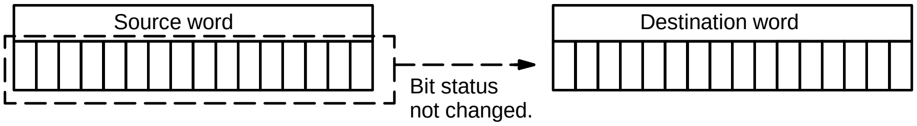
Operand Data Areas

S: Source word
IR, SR, AR, DM, HR, TC, LR, #

D: Destination word
IR, 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. However, these can be easily changed using BSET(71).

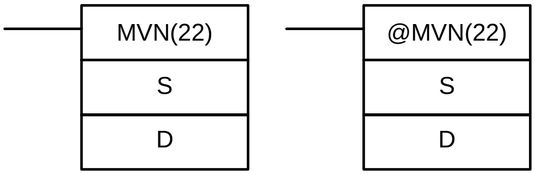
Flags

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.

5-12-2 MOVE NOT – MVN(22)

Ladder Symbols



Operand Data Areas

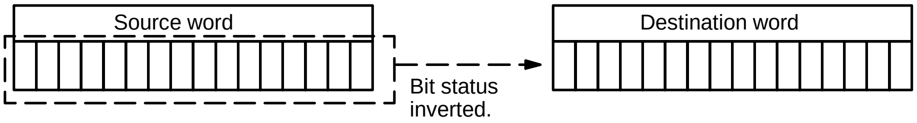
S: Source word
IR, SR, AR, DM, HR, TC, LR, #

D: Destination word
IR, 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 complement of the content of S (specified word or four-digit hexadecimal constant) to D, i.e., for each

ON bit in S, the corresponding bit in D is turned OFF, and for each OFF bit in S, the corresponding bit in D is turned ON.



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).

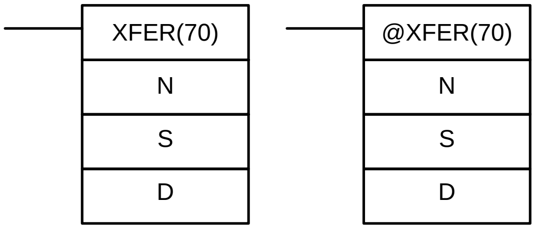
Flags

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.

5-12-3 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

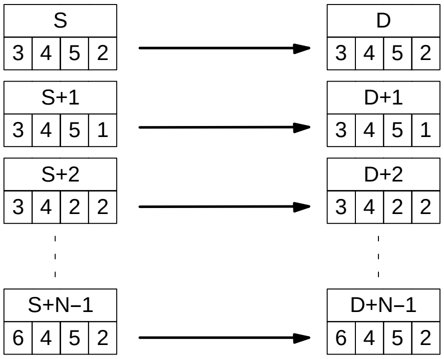
IR, AR, DM, HR, TC, LR

Limitations

Both S and D may be in the same data area, but their respective block areas must not overlap. S and S+N–1 must be in the same data area, as must D and D+N–1.

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–1 to D, D+1, ..., D+N–1.



Flags

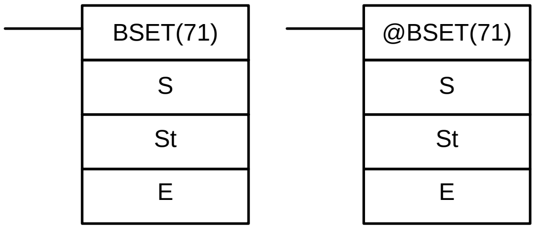
ER: N is not BCD

S and S+N–1 or D and D+N–1 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.)

5-12-4 BLOCK SET – BSET(71)

Ladder Symbols



Operand Data Areas

S: Source data

IR, SR, AR, DM, HR, TC, LR, #

St: Starting word

IR, AR, DM, HR, TC, LR

E: End Word

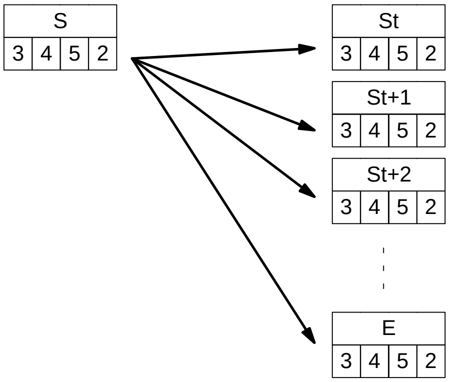
IR, AR, DM, HR, TC, LR

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.

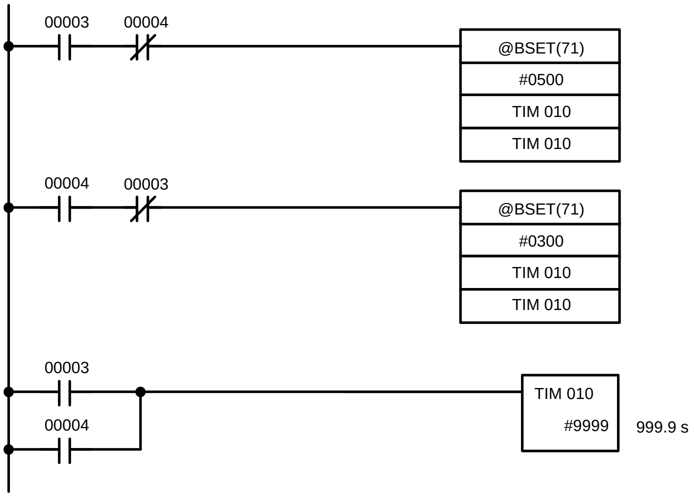
Flags

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

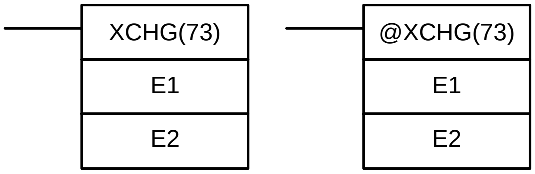
The following example shows how to use BSET(71) to change the PV of a timer depending on the status of IR 00003 and IR 00004. When IR 00003 is ON, TIM 010 will operate as a 50-second timer; when IR 00004 is ON, TIM 010 will operate as a 30-second timer.



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

5-12-5 DATA EXCHANGE – XCHG(73)

Ladder Symbols



Operand Data Areas

E1: Exchange word 1

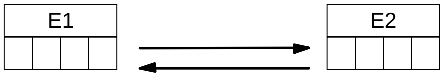
IR, AR, DM, HR, TC, LR

E2: Exchange word 2

IR, 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.



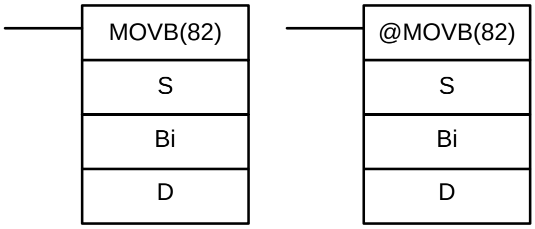
If you want to exchange content of blocks whose size is greater than 1 word, use work words as an intermediate buffer to hold one of the blocks using XFER(70) three times.

Flags

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

5-12-6 MOVE BIT – MOVB(82)

Ladder Symbols



Operand Data Areas

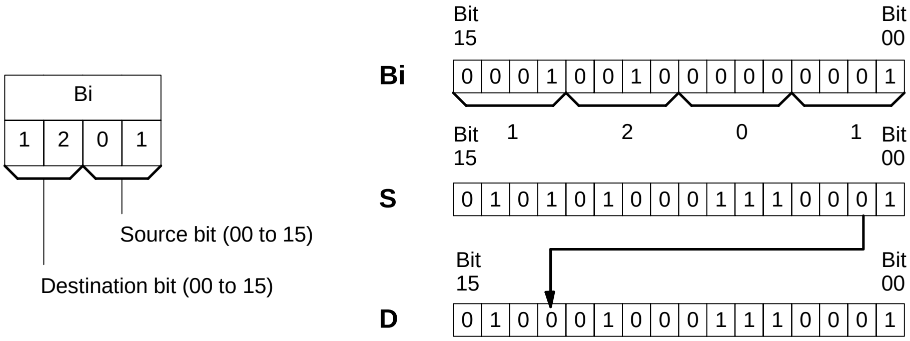
S: Source word
IR, SR, AR, DM, HR, LR, #

Bi: Bit designator (BCD)
IR, AR, DM, HR, TC, LR, #

D: Destination word
IR, 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.

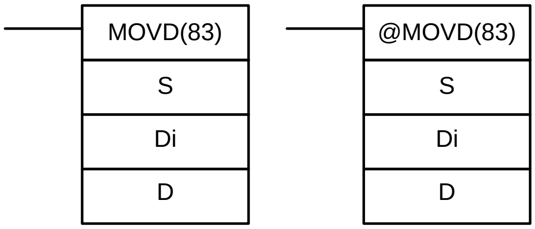


Flags **ER:** C is not BCD, or it is specifying a non-existent bit (i.e., bit specification must be between 00 and 15).

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

5-12-7 MOVE DIGIT – MOVD(83)

Ladder Symbols



Operand Data Areas

S: Source word
IR, SR, AR, DM, HR, TC, LR, #

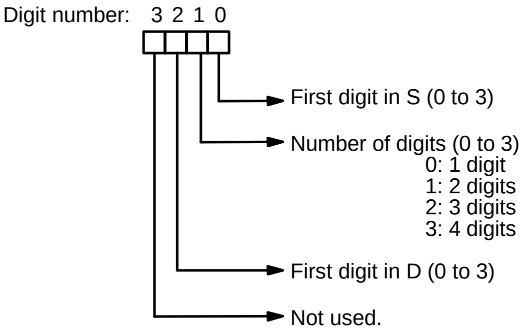
Di: Digit designator (BCD)
IR, AR, DM, HR, TC, LR, #

D: Destination word
IR, AR, DM, HR, TC, LR

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

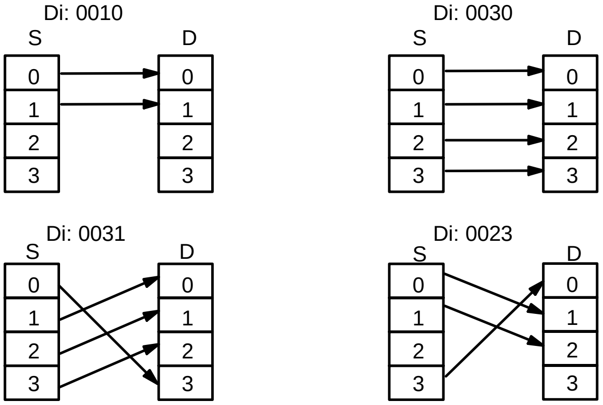
Description

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 number: 3 2 1 0
First digit in S (0 to 3)
Number of digits (0 to 3)
0: 1 digit
1: 2 digits
2: 3 digits
3: 4 digits
First digit in D (0 to 3)
Not used.

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.

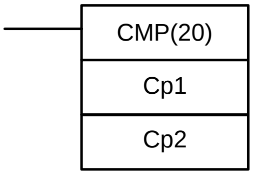
Indirectly addressed DM word is non-existent. (Content of *DM word is not BCD, or the DM area boundary has been exceeded.)

5-13 Data Comparison

This section describes the instructions used for comparing data. CMP(20) is used to compare the contents of two words; BCMP(68) is used to determine within which of several preset ranges the content of one word lies.

5-13-1 COMPARE – CMP(20)

Ladder Symbols



Operand Data Areas

Cp1: First compare word
IR, SR, AR, DM, HR, TC, TR, #

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.

Flags

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

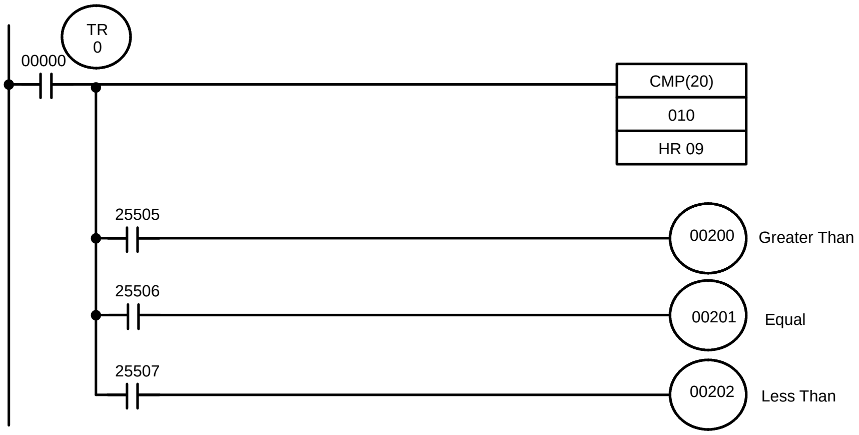
EQ: ON if Cp1 equals Cp2.

LE: ON if Cp1 is less than Cp2.

GR: ON if Cp1 is greater than Cp2.

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	AND	25505
00004	OUT	00200

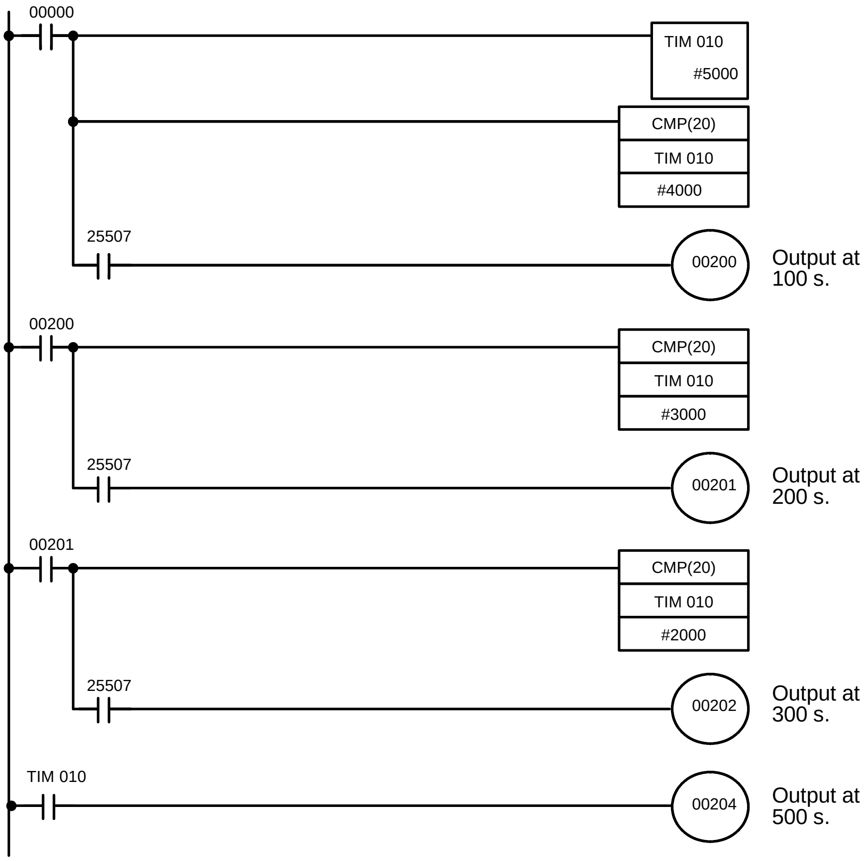
Address	Instruction	Operands
00005	LD	TR 0
00006	AND	25506
00007	OUT	00201
00008	LD	TR 0
0009	AND	25507
00010	OUT	00202

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.

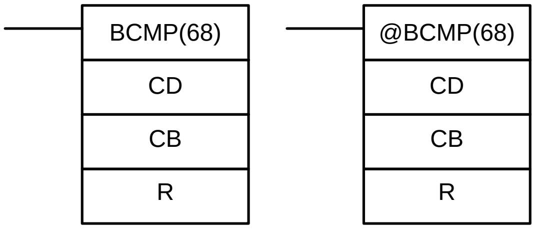
Because all of the comparisons here use to the timer’s PV as reference, the other operand for each CMP(20) must be in 4-digit BCD.



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
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

5-13-2 BLOCK COMPARE – BCMP(68)

Ladder Symbols



Operand Data Areas

CD: Compare data (BCD)

IR, SR, AR, DM, HR, TC, LR, #

CB: First comparison block word

IR, AR, DM, HR, TC, LR

R: First result word

IR, AR, DM, HR, TC, LR

Limitations

All data is compared in BCD form except for CB, which is in hexadecimal. All comparison block words must be within the same data area, as must all result words.

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 CB+1, CB+2, CB+3 ..., CB+2n+2. The size of the table is determined by n, which is given in bits 00 through 07 of CB. The value of n is in hexadecimal and can be between 0 and 255, i.e., comparisons can be made for between 1 and 256 ranges.

Each range is defined by two words, the first one providing one limit and the second word providing the other limit. If CD is found to be within any of these ranges (inclusive of the limits), the corresponding bit in R is set. If more than 16 ranges are compared, bits are turned ON in consecutive words following R, i.e., R+1, R+2, etc. 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 the results words will be turned OFF.

CB+1 ≤ CD ≤ CB+2		Bit 00 of R
CB+3 ≤ CD ≤ CB+4		Bit 01 of R
CB+5 ≤ CD ≤ CB+6		Bit 02 of R
CB+7 ≤ CD ≤ CB+8		Bit 03 of R
CB+9 ≤ CD ≤ CB+10		Bit 04 of R
CB+11 ≤ CD ≤ CB+12		Bit 05 of R
CB+13 ≤ CD ≤ CB+14		Bit 06 of R
CB+15 ≤ CD ≤ CB+16		Bit 07 of R
CB+17 ≤ CD ≤ CB+18		Bit 08 of R
CB+29 ≤ CD ≤ CB+20		Bit 09 of R
CB+21 ≤ CD ≤ CB+22		Bit 10 of R
CB+23 ≤ CD ≤ CB+24		Bit 11 of R
CB+25 ≤ CD ≤ CB+26		Bit 12 of R
CB+27 ≤ CD ≤ CB+28		Bit 13 of R
CB+39 ≤ CD ≤ CB+30		Bit 14 of R
CB+31 ≤ CD ≤ CB+32		Bit 15 of R
CB+33 ≤ CD ≤ CB+34		Bit 00 of R+1
CB+35 ≤ CD ≤ CB+36		Bit 01 of R+1
CB+37 ≤ CD ≤ CB+38		Bit 02 of R+1
etc.	etc.	
CB+2n+1 ≤ CD ≤ CB+2n+2		(See below.)

The last bit required for the result will be the truncated integer quotient of (n–1)/15, with one less than the remainder being the bit number of the last required bit.

Ring Tables

Because the values of word pairs determining any one range can be set with either the largest or the smallest value first, the table can be set up to operate in ring form.

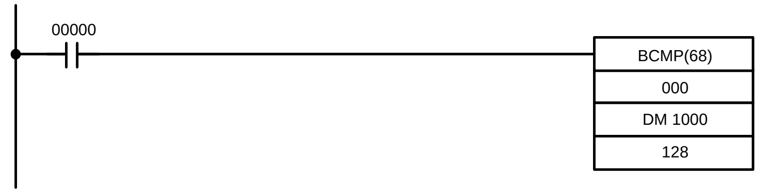
Flags

ER: CB through CB+2 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 1

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.



CD: 000

000 = 0175

Compare data in IR 000

(which contains 0175) with the given ranges.

CB: 1000

DM 1000 = 0017

Content of CB designates normal BCD comparison and 24 ranges, i.e., n is 23.

First limits	
DM 1001	0000
DM 1003	0080
DM 1005	0160
etc.	etc.
DM 1031	1200
DM 1033	1280
DM 1035	1360
DM 1037	1440
etc.	etc.
DM 1047	1840

Second limits	
DM 1002	0100
DM 1004	0180
DM 1006	0260
etc.	etc.
DM 1032	1300
DM 1034	1380
DM 1036	1460
DM 1038	1540
etc.	etc.
DM 1048	1940

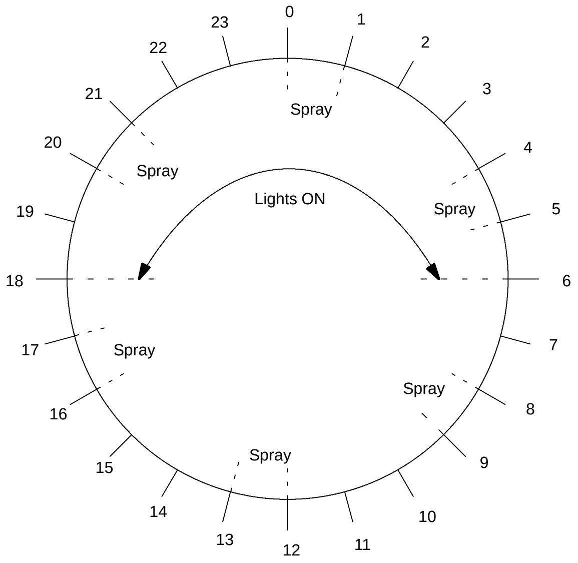
R: 128	
IR 12800	0
IR 12801	1
IR 12802	1
etc.	etc.
IR 12815	0
IR 12900	0
IR 12901	0
IR 12902	0
etc.	etc.
IR 12907	0

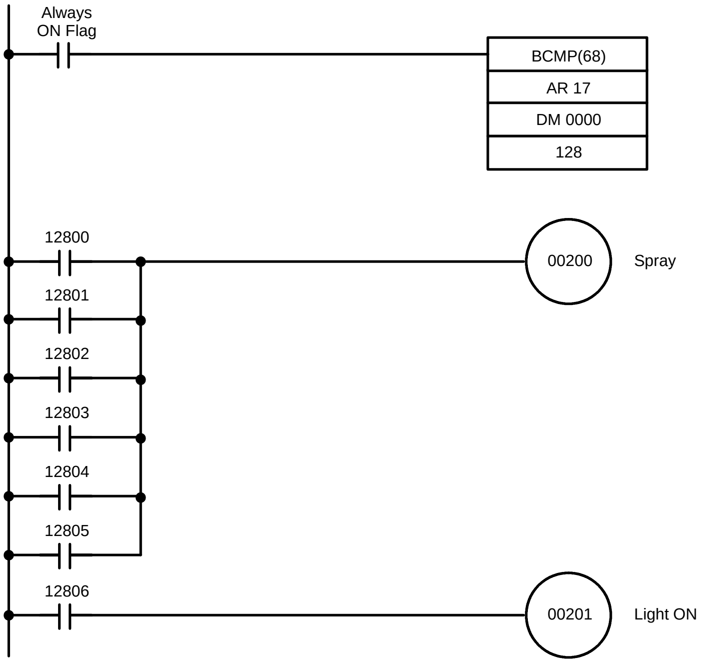
Flags

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

Example 2

The following example shows how to control the operation of water sprinklers and lights based on the current time in AR 17.





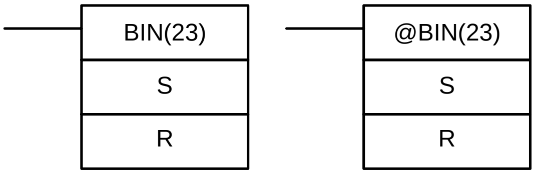
DM 0000	0006
DM 0001	0000
DM 0002	0100
DM 0003	0400
DM 0004	0500
DM 0005	0800
DM 0006	0900
DM 0007	1200
DM 0008	1300
DM 0009	1600
DM 0010	1700
DM 0011	2000
DM 0012	2100
DM 0013	1800
DM 0014	0600

5-14 Data Conversion

The conversion instructions convert word data from 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 and from 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.

5-14-1 BCD-TO-BINARY – BIN(23)

Ladder Symbols



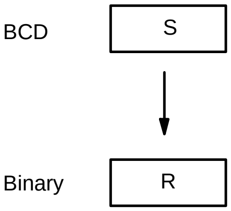
Operand Data Areas

S: Source word (BCD)
IR, SR, AR, DM, HR, TC, LR

R: Result word
IR, 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

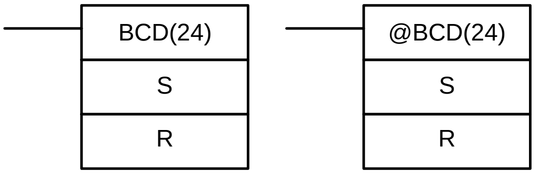
ER: The content of S is not BCD.

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 0 is placed in R.

5-14-2 BINARY-TO-BCD – BCD(24)

Ladder Symbols



Operand Data Areas

S: Source word (binary)
IR, SR, AR, DM, HR, LR

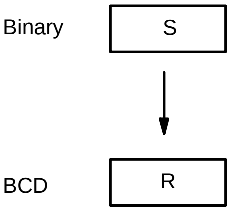
R: Result word
IR, AR, DM, HR, LR

Limitations

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.

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.

Flags

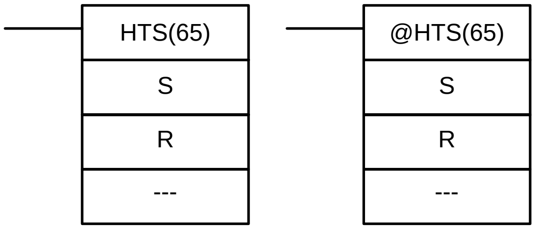
ER: S is greater than 270F.

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 content of R is zero.

5-14-3 HOURS-TO-SECONDS – HTS(65)

Ladder Symbols



Operand Data Areas

S: Beginning source word (BCD)

IR, SR, AR, DM, HR, TC, LR

R: Beginning result word (BCD)

IR, 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

HTS(65) is used to convert time notation in hours/minutes/seconds to an equivalent time in seconds only.

For the source data, the seconds are designated in bits 00 through 07 and the minutes are designated in bits 08 through 15 of S. The hours are designated in S+1. The maximum is thus 9,999 hours, 59 minutes, and 59 seconds.

The results are output to R and R+1. The maximum obtainable value is 35,999,999 seconds.

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.

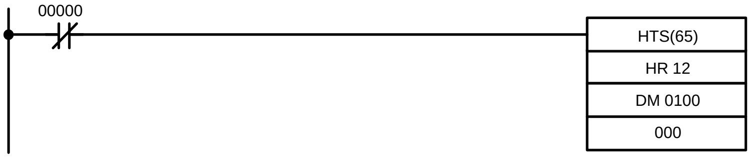
Number of seconds and/or minutes exceeds 59.

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., 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	HTS(65)		
		HR	12
		DM	0100
			000

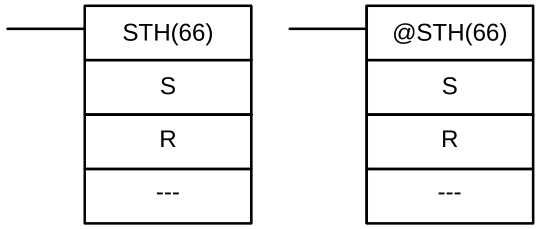
HR 12	3	2	0	7	2,815 hrs, 32 min, 07 s
HR 13	2	8	1	5	

↓

DM 0100	5	9	2	7	10,135,927 s
DM 0101	1	0	1	3	

5-14-4 SECONDS-TO-HOURS – STH(66)

Ladder Symbols



Operand Data Areas

S: Beginning source word (BCD)

IR, SR, AR, DM, HR, TC, LR

R: Beginning result word (BCD)

IR, 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. For any setting over 15,000,000 seconds, increase the watchdog timer to 180 ms.

Description

STH(66) is used to convert time notation in seconds to an equivalent time in hours/minutes/seconds.

The number of seconds designated in S and S+1 is converted to hours/minutes/seconds and placed in R and R+1.

For the results, the seconds are placed in bits 00 through 07 and the minutes are placed in bits 08 through 15 of R. The hours are placed in R+1. The maximum will be 9,999 hours, 59 minutes, and 59 seconds.

Flags

ER: S and S+1 or R and R+1 are not in the same data area.

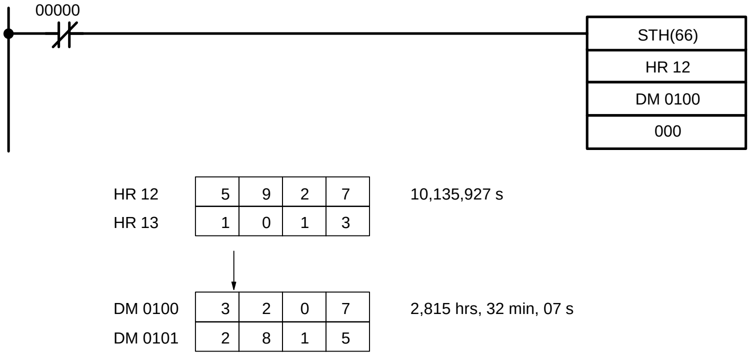
S and/or S+1 do not contain BCD or exceed 35,999,999 seconds.

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

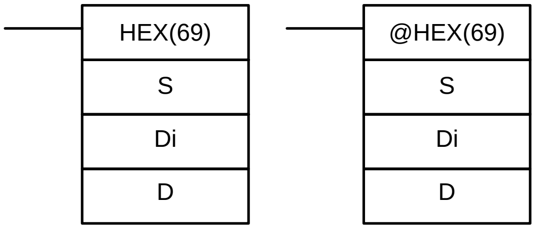
When 00000 is OFF (i.e., the execution condition is ON), the following instruction would convert the seconds given in HR 12 and HR 13 to hours, minutes, and seconds and store the results in DM 0100 and DM 0101 as shown.



Address	Instruction	Operands
00000	LD NOT	00000
00001	STH(66)	
		HR 12
		DM 0100
		000

5-14-5 HEXADECIMAL CONVERT – HEX(69)

Ladder Symbols



Operand Data Areas

S: Source word

IR, SR, AR, DM, HR, TC, LR

Di: Digit designator

IR, AR, DM, HR, TC, LR, #

D: First destination word

IR, AR, DM, HR, LR

Limitations

Di must be within the values given below.

All source words must be in the same data area.

The ASCII codes in the source bytes designated for conversion must be equivalent to hexadecimal numbers (between 0 and F).

Description

When the execution condition is OFF, HEX(69) is not executed. When the execution condition is ON, HEX(69) converts the designated 8-bit ASCII code(s) in S into it’s hexadecimal equivalent and places it into the destination word D beginning at the designated digit.

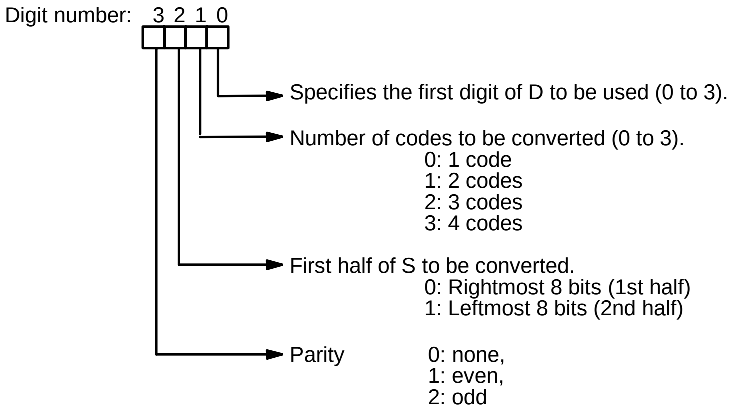
Up to four ASCII codes in S to S+2 may be converted in order from the designated first code. The first code (rightmost or leftmost 8 bits), the number of codes to be converted, and the digit of D to receive the first number are designated in Di. If multiple codes are designated, they will be placed in order starting from the designated digit of D. If more digits are designated than remain in D (counting from the first designated digit), further digits will be used starting back at the beginning of D.

ASC(86) can be used together with HEX(69) to convert data to and from ASCII characters for transmission through the RS-232C interface.

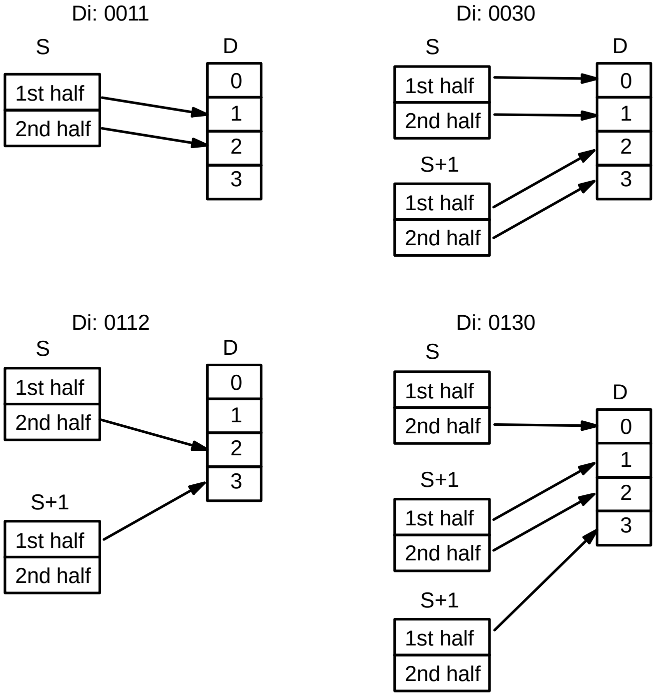
Refer to *Appendix I* for a table of extended ASCII characters.

Digit Designator

The digits of Di are set as shown below.


Digit number: 3 2 1 0
Specifies the first digit of D to be used (0 to 3).
Number of codes to be converted (0 to 3).
0: 1 code
1: 2 codes
2: 3 codes
3: 4 codes
First half of S to be converted.
0: Rightmost 8 bits (1st half)
1: Leftmost 8 bits (2nd half)
Parity
0: none,
1: even,
2: odd

Some examples of Di values and the 8-bit ASCII code to 4-bit binary conversions that they produce are shown below.



Parity

The leftmost bit of each 2-digit ASCII character is adjusted for either even, odd, or no parity. If no parity is designated, the leftmost bit must be zero.

When even parity is designated, the leftmost bit must 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 must be adjusted so that there is an odd number of ON bits.

If the parity in a source ASCII character does not agree with the parity specified in Di, the ER Flag will be turned ON and the instruction will not be executed.

Flags

ER: The first source word, S, is not in the same data area as S+1 or S+2.

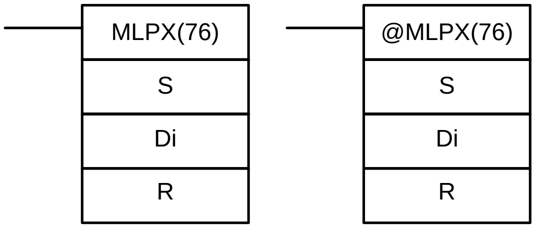
Indirectly addressed DM word is non-existent. (Content of *DM word is not BCD, or the DM area boundary has been exceeded.)

The parity in the source ASCII code does not agree with the parity specified in Di.

The source ASCII code is not in the range 0 to F and cannot be converted to hexadecimal.

5-14-6 4-TO-16 DECODER – MLPX(76)

Ladder Symbols



Operand Data Areas

S: Source word

IR, SR, AR, DM, HR, TC, LR

Di: Digit designator

IR, AR, DM, HR, TC, LR, #

R: First result word

IR, AR, DM, HR, LR

Limitations

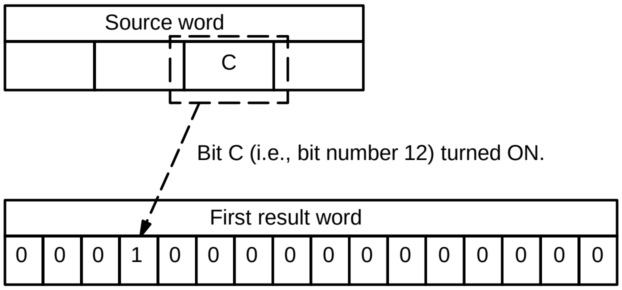
The rightmost two digits of Di must each be between 0 and 3.

All result words must be in the same data area.

Description

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.)

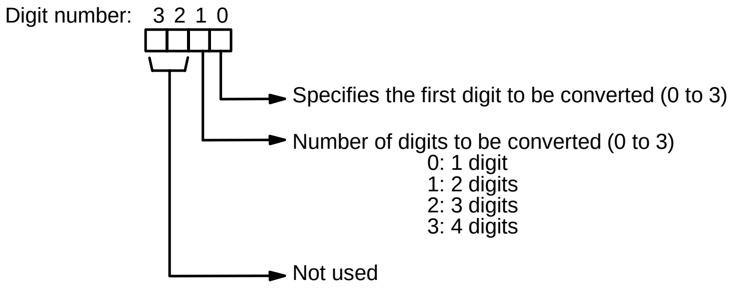
The following is an example of a one-digit decode operation from digit number 1 of S, i.e., here Di would be 0001.



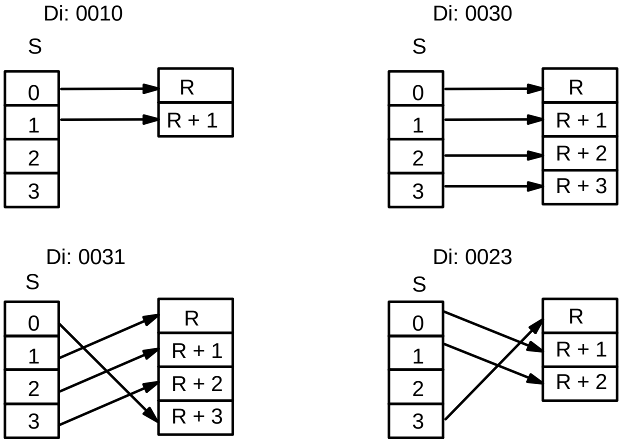
The first digit and the number of digits to be converted are designated in Di. If more digits are designated than remain in S (counting from the designated first digit), the remaining digits will be taken starting back at the beginning of S. The final word required to store the converted result (R plus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

Digit Designator

The digits of Di are set as shown below.


Digit number: 3 2 1 0
Specifies the first digit to be converted (0 to 3)
Number of digits to be converted (0 to 3)
0: 1 digit
1: 2 digits
2: 3 digits
3: 4 digits
Not used

Some example Di values and the digit-to-word conversions that they produce are shown below.



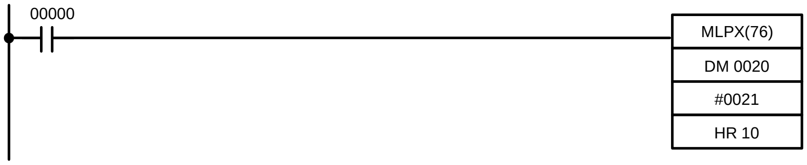
Flags

ER: Undefined digit designator, or R plus number of digits exceeds a data area.

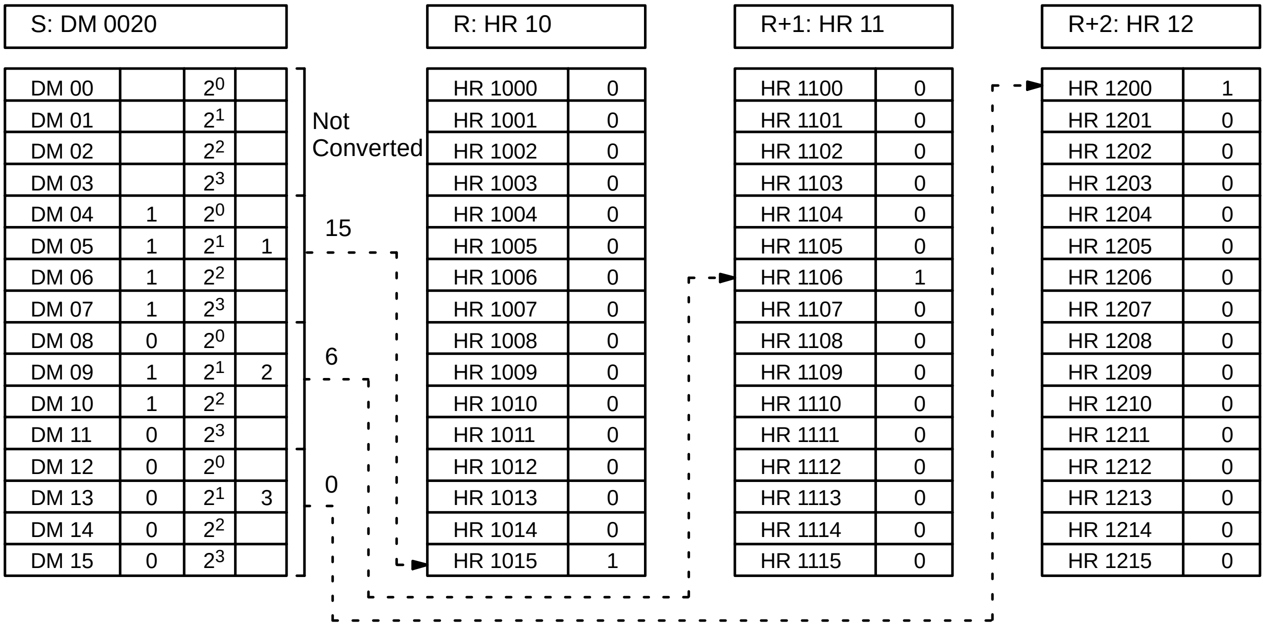
Indirectly addressed DM word is non-existent. (Content of *DM word is not BCD, or the DM area boundary has been exceeded.)

Example

The following program converts three digits of data from DM 0020 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)		
		DM	00200
		#	0021
		HR	10



S: DM 0020

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	

R: HR 10

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

R+1: HR 11

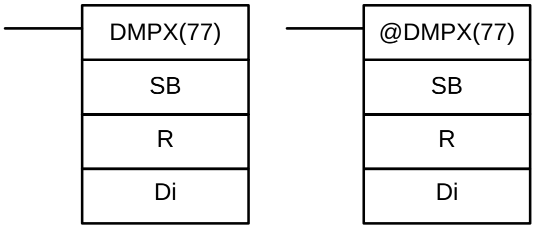
HR 1100	0
HR 1101	0
HR 1102	0
HR 1103	0
HR 1104	0
HR 1105	0
HR 1106	1
HR 1107	0
HR 1108	0
HR 1109	0
HR 1110	0
HR 1111	0
HR 1112	0
HR 1113	0
HR 1114	0
HR 1115	0

R+2: HR 12

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

5-14-7 16-TO-4 ENCODER – DMPX(77)

Ladder Symbols



Operand Data Areas

SB: First source word

IR, SR, AR, DM, HR, TC, LR

R: Result word

IR, AR, DM, HR, LR

Di: Digit designator

IR, AR, DM, HR, TC, LR, #

Limitations

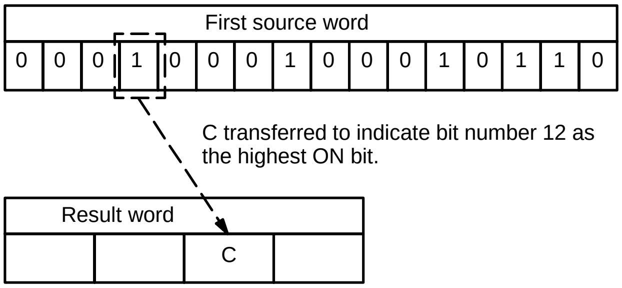
The rightmost two digits of Di must each be between 0 and 3.

All source words must be in the same data area.

Description

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 of the highest ON bit number, then transfers the hexadecimal value to the specified digit in R. The digits to receive the results are specified in Di, which also specifies the number of digits to be encoded.

The following is an example of a one-digit encode operation to digit number 1 of R, i.e., here Di would be 0001.

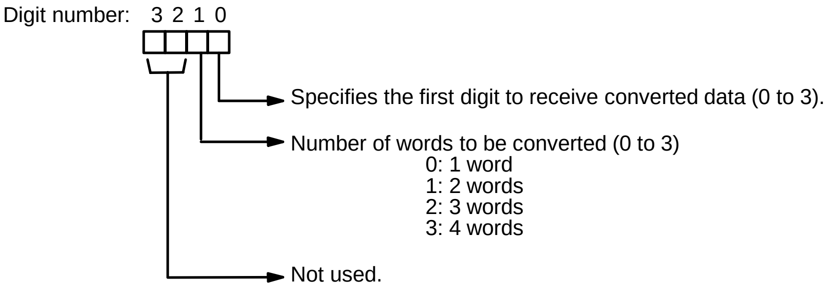

First source word
0 0 0 1 0 0 0 1 0 0 0 1 0 1 1 0
C transferred to indicate bit number 12 as the highest ON bit.
Result word
C

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.

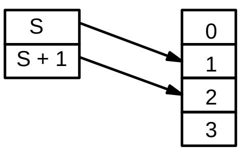
Digit Designator

The digits of Di are set as shown below.

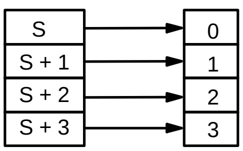

Digit number: 3 2 1 0
Specifies the first digit to receive converted data (0 to 3).
Number of words to be converted (0 to 3)
0: 1 word
1: 2 words
2: 3 words
3: 4 words
Not used.

Some example Di values and the word-to-digit conversions that they produce are shown below.

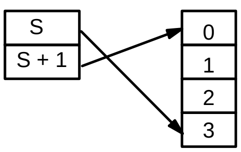
Di: 0011



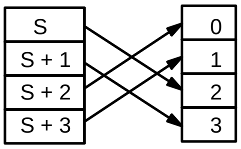
Di: 0030



Di: 0013



Di: 0032



Flags

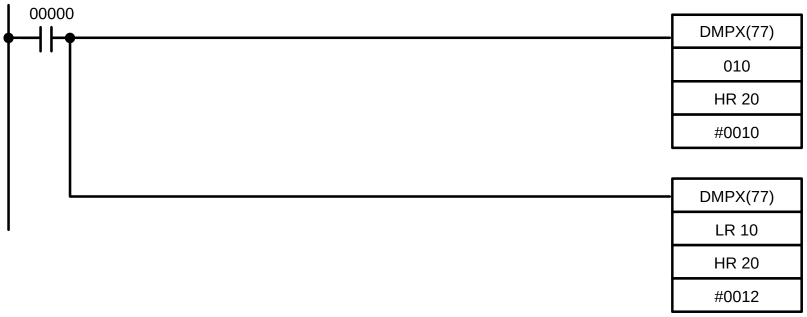
ER: Undefined digit designator, or S plus number of digits exceeds a data area.

Content of a source word is 0.

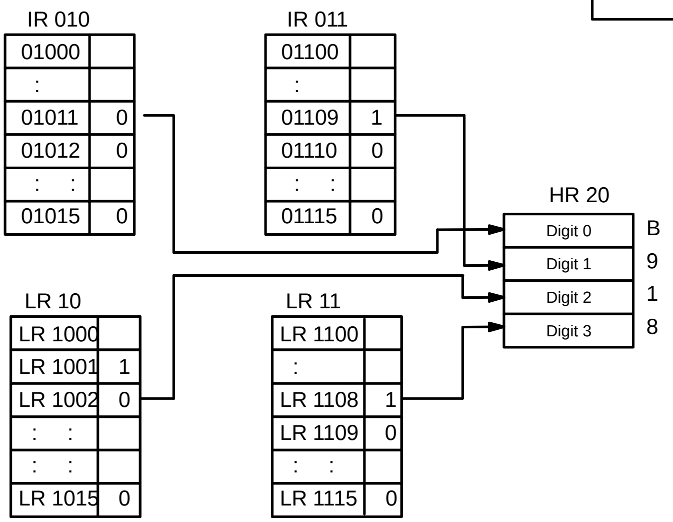
Indirectly addressed DM word is non-existent. (Content of *DM word is not BCD, or the DM area boundary has been exceeded.)

Example

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.

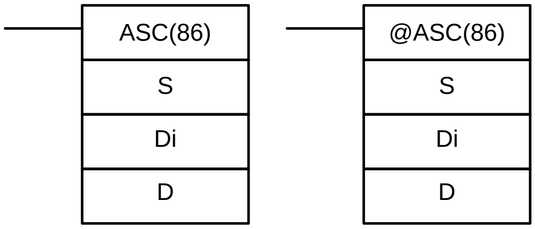


Address	Instruction	Operands
00000	LD	00000
00001	DMPX(77)	
		010
		HR 20
		# 0010
00002	DMPX(77)	
		LR 10
		HR 20
		# 0012



5-14-8 ASCII CONVERT – ASC(86)

Ladder Symbols



Operand Data Areas

S: Source word

IR, SR, AR, DM, HR, TC, LR

Di: Digit designator

IR, AR, DM, HR, TC, LR, #

D: First destination word

IR, AR, DM, HR, LR

Limitations

Di must be within the values given in the following.

All destination words must be in the same data area.

Description

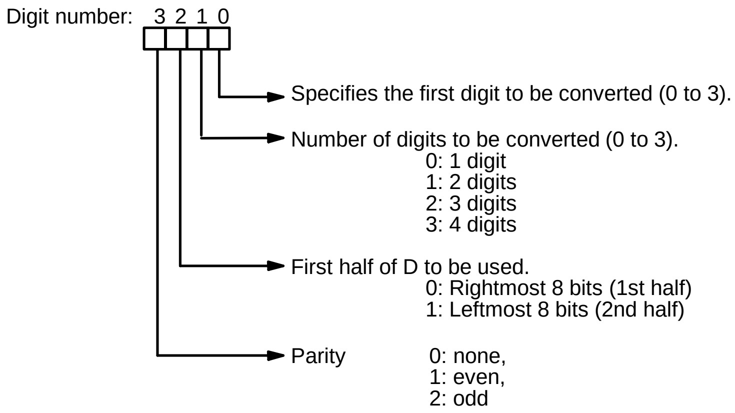
When the execution condition is OFF, ASC(86) is not executed. When the execution condition is ON, ASC(86) converts the designated digit(s) of S into the equivalent 8-bit ASCII code and places it into the destination word(s) beginning with D.

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.

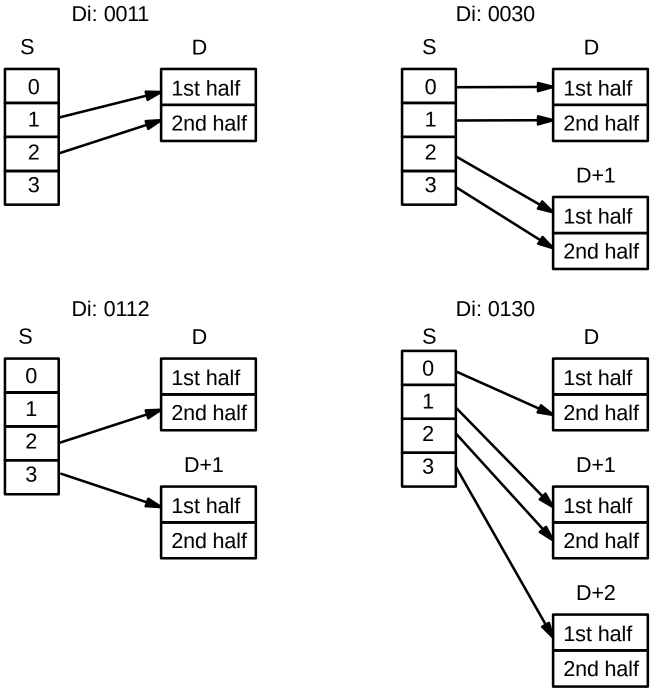
Refer to *Appendix I* for a table of extended ASCII characters.

Digit Designator

The digits of Di are set as shown below.


Digit number: 3 2 1 0
Specifies the first digit to be converted (0 to 3).
Number of digits to be converted (0 to 3).
0: 1 digit
1: 2 digits
2: 3 digits
3: 4 digits
First half of D to be used.
0: Rightmost 8 bits (1st half)
1: Leftmost 8 bits (2nd half)
Parity
0: none,
1: even,
2: odd

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 2-digit ASCII character 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.)

5-15 BCD Calculations

The BCD calculation instructions – INC(38), DEC(39), ADD(30), SUB(31), MUL(32), and DIV(33) – all perform arithmetic operations on BCD data.

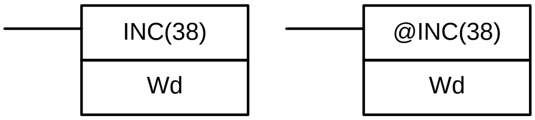
For INC(38) and DEC(39) the source and result words are the same. That is, the content of the source word is overwritten with the instruction result. All other instructions change only the content of the words in which results are placed, i.e., the contents of source words are the same before and after 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.

5-15-1 INCREMENT – INC(38)

Ladder Symbols



Operand Data Areas

Wd: Increment word (BCD)
IR, 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

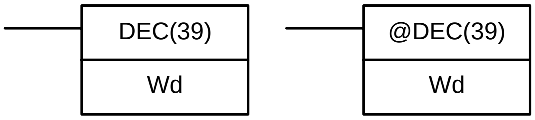
ER: Wd is not BCD

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 result of the increment is 0.

5-15-2 DECREMENT – DEC(39)

Ladder Symbols



Operand Data Areas

Wd: Decrement word (BCD)
IR, 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

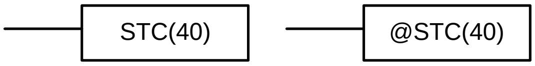
ER: Wd is not BCD

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.

5-15-3 SET CARRY – STC(40)

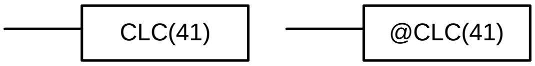
Ladder Symbols



When the execution condition is OFF, STC(40) is not executed.When the execution condition is ON, STC(40) turns ON CY (SR 25504).

5-15-4 CLEAR CARRY – CLC(41)

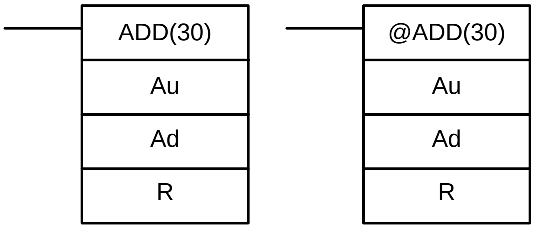
Ladder Symbols



When the execution condition is OFF, CLC(41) is not executed.When the execution condition is ON, CLC(41) turns OFF CY (SR 25504).

5-15-5 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, 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.

$$\boxed{Au} + \boxed{Ad} + \boxed{CY} \rightarrow \boxed{CY} \quad \boxed{R}$$

Flags

ER: Au and/or Ad is not BCD.

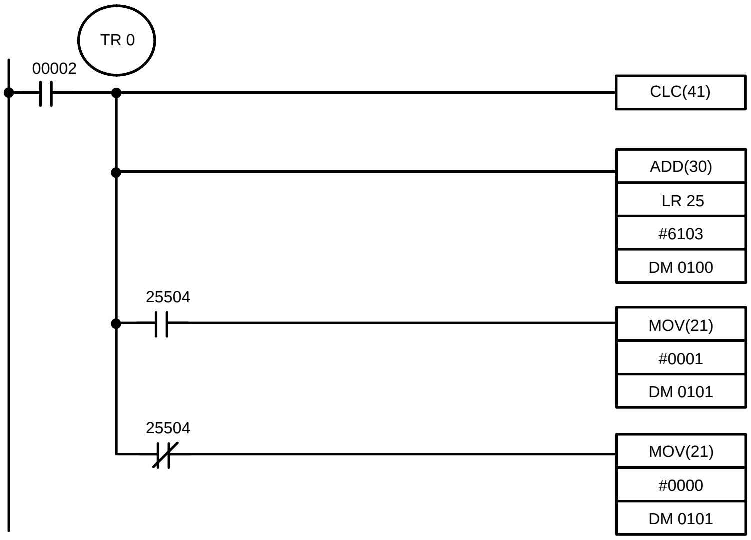
Indirectly addressed DM word is non-existent. (Content of *DM word is not BCD, or the DM area boundary has been exceeded.)

CY: ON when there is a carry in the result.

EQ: ON when the result is 0.

Example

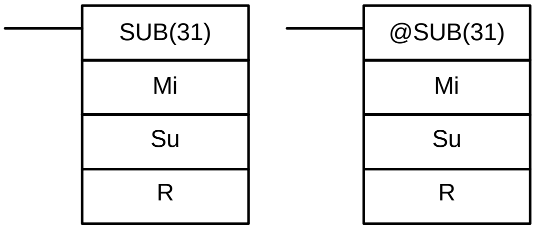
If 00002 is ON, the program represented by the following diagram clears CY with CLC(41), adds the content of LR 25 to a constant (6103), places the result in DM 0100, and then moves either all zeros or 0001 into DM 0101 depending on the status of CY (25504). This ensures that any carry from the last 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	ADD(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

5-15-6 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, 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).

$$\boxed{\text{Mi}} - \boxed{\text{Su}} - \boxed{\text{CY}} \rightarrow \boxed{\text{CY}} \quad \boxed{\text{R}}$$

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.

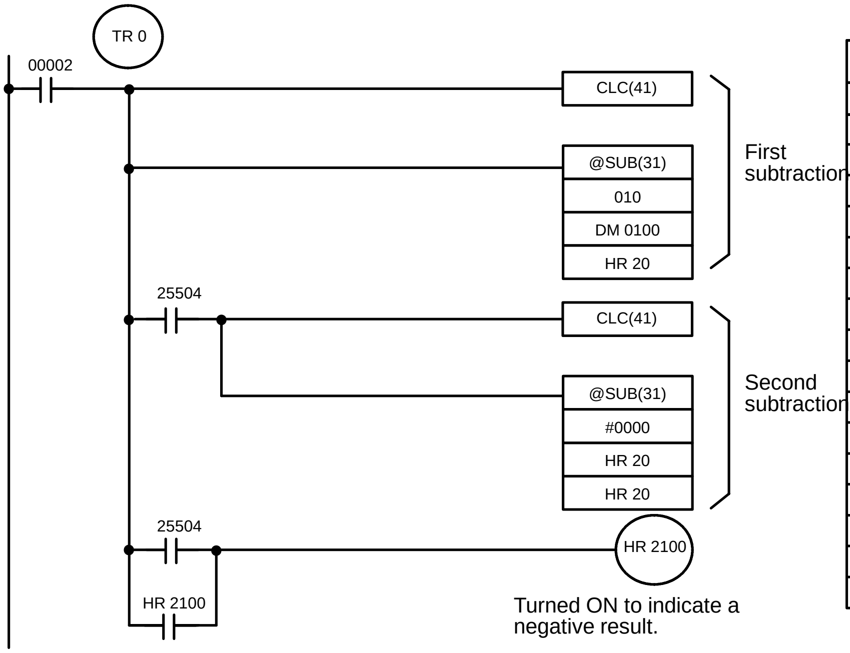
Example 1: Subtraction of 4-digit Numbers

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

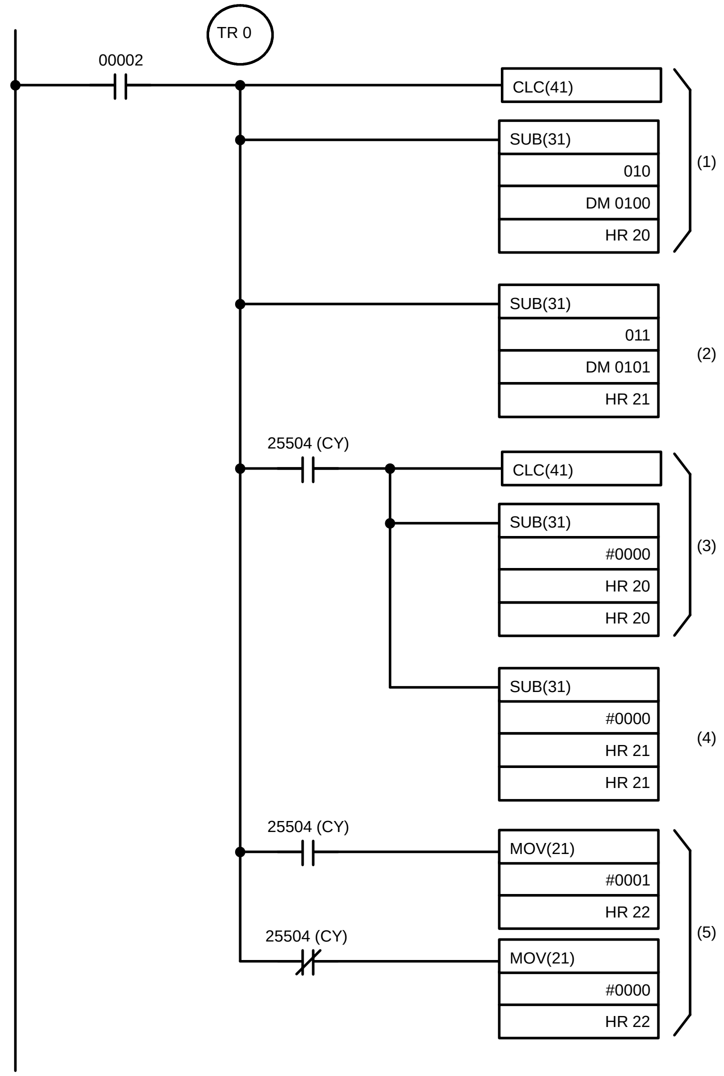
	0000	
HR 20	–7577	
CY	–0	
HR 20	2423	(0000 + (10000 – 7577))
CY	1	(negative result)

In the above case, the program would turn ON HR 2100 to indicate that the value held in HR 20 is negative.

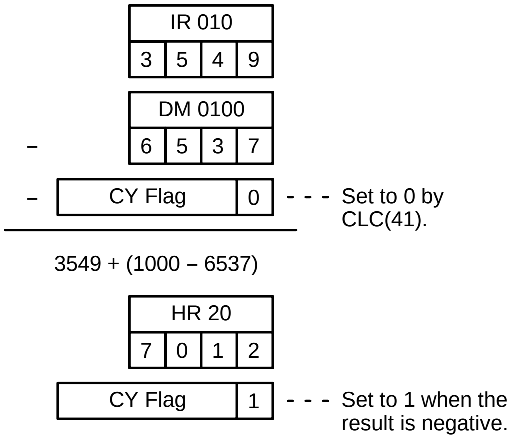
Example 2: Subtraction of 8-digit Numbers

In this example, the 8-digit content of DM 0100 and DM 0101 is subtracted from the 8-digit content of 010 and 011 and the result is placed in HR 20, HR

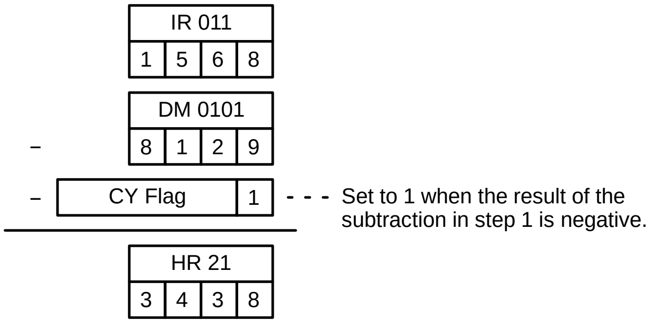
21, and HR 22. (The content of HR 22 is set to #0001 to indicate a negative result.)



1, 2, 3... 1. When 00002 is ON, CY is set to zero, the content of DM 0100 is subtracted from the content of 010, and the result is placed in HR 20.



2. The content of DM 0101 and CY are then subtracted from the content of 011 and the result is placed in HR 21.

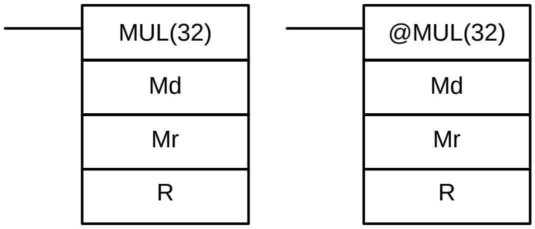

IR 011
1 5 6 8
DM 0101
–
8 1 2 9
–
CY Flag
1
Set to 1 when the result of the subtraction in step 1 is negative.
HR 21
3 4 3 8

3. If the result of the subtraction in step 2 is negative, CY will be turned ON and the result placed in HR 20 and HR 21 will be the 10's complement of the actual result. Here the 10's complement of the 4 digits in HR 20 is converted to the actual result.
   #0000–7012=0000+(10000–7012)=2988
4. The 10's complement of the 4 digits in HR 21 is converted to the actual result. The 1 in the equation comes from the CY Flag set in step 3.
   #0000–3438–1=0000+(10000–3438–1)=6561
5. HR 22 will contain #0001 if the result is negative and #0000 if the result is positive. The final result is shown below.

HR 22				HR 21				HR 20			
0	0	0	1	6	5	6	1	2	9	8	8

5-15-7 BCD MULTIPLY – MUL(32)

Ladder Symbols



Operand Data Areas

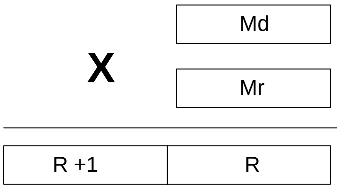
Md: Multiplicand (BCD)
IR, SR, AR, DM, HR, TC, LR, #

Mr: Multiplier (BCD)
IR, SR, AR, DM, HR, TC, LR, #

R: First result word
IR, AR, DM, HR LR

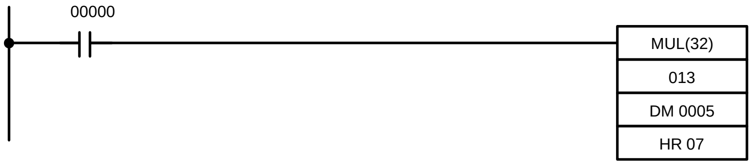
Description

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 1: Multiplication of 4-digit Numbers

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

Md: IR 013			
3	3	5	6

X

Mr: DM 0005			
0	0	2	5

R+1: HR 08				R: HR 07			
0	0	0	8	3	9	0	0

Example 2: Multiplication of 8-digit Numbers

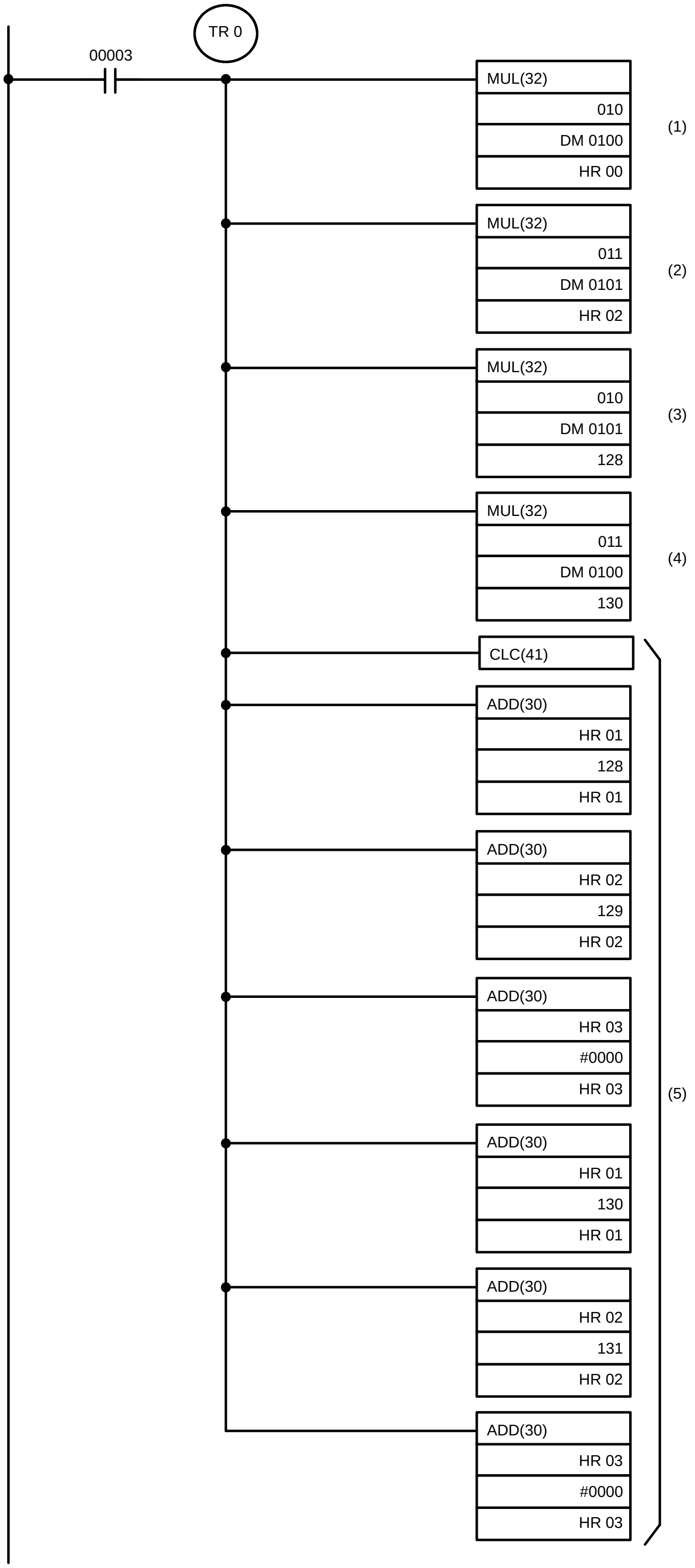
In this example, the 8-digit content of 010 and 011 is multiplied by the 8-digit content of DM 0100 and DM 0101 and the result is placed in HR 00 to HR 03.

The letters in the table below are assigned to the 4 source words to make later calculations easier to follow.

Letter	Source word	Contents
A	IR 011	1234
B	IR 010	5678
C	DM 0101	8765
D	DM 0100	4321

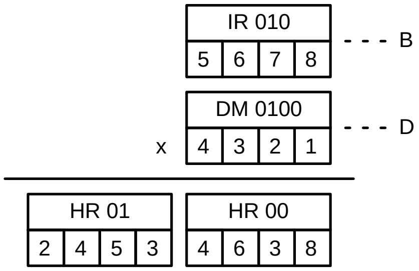
Fewer calculations and thus fewer instructions will be required if the multiplication is rearranged as shown below:

$(A\times10^4+B) \times (C\times10^4+D) = A\times C\times10^4 + (B\times C + A\times D) \times 10^4 + B\times D$

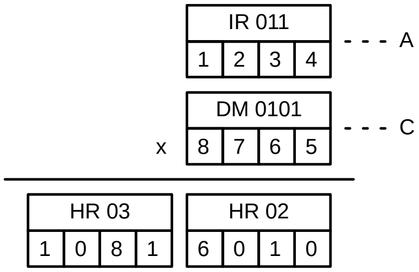


1, 2, 3...

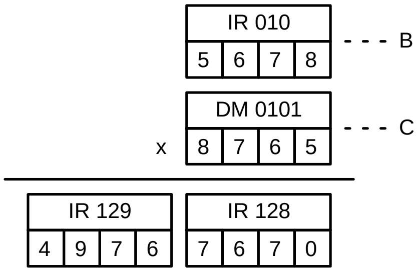
1. The least significant 4-digits in each number are multiplied first (B×D).



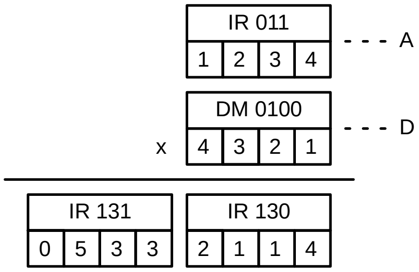
2. The most significant 4-digits in each number are multiplied (A×C).



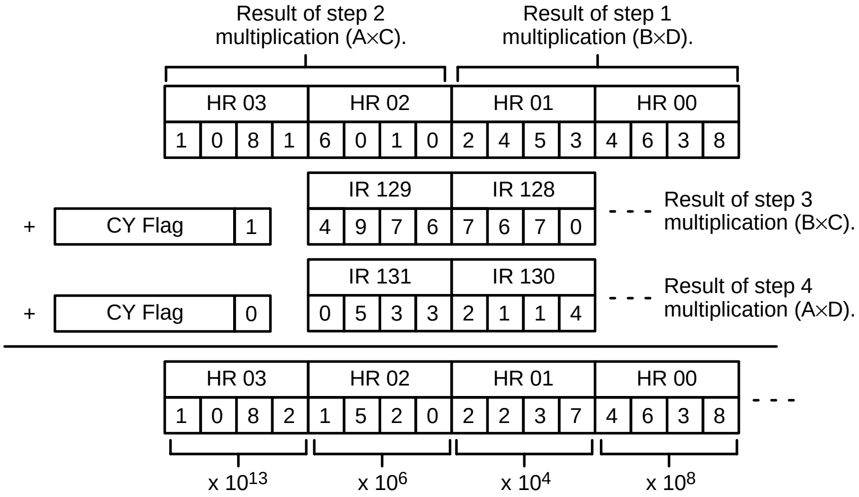
3. The least significant 4-digits of the first number are multiplied by the most significant 4-digits of the second (B×C).



4. The most significant 4-digits of the first number are multiplied by the least significant 4-digits of the second (A×D).



5. The results of the multiplications in steps 1 to 4 are combined. The 10^4 terms and 10^8 terms are arranged in columns and added. The result of addition in the 10^8-term column exceeds 9999, so the 10^{12} term is incremented by 1.



Flags

ER: Md and/or Mr not BCD.

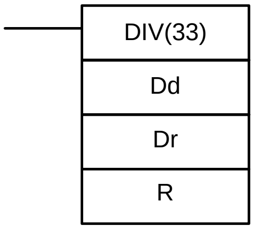
Indirectly addressed DM word is non-existent. (Content of *DM word is not BCD, or the DM area boundary has been exceeded.)

CY: ON when there is a carry in the result.

EQ: ON when the result is 0.

5-15-8 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, #

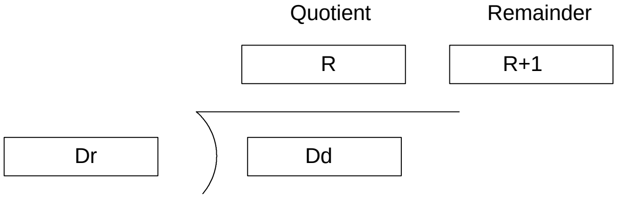
R: First result word (BCD)
IR, AR, DM, HR, LR

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

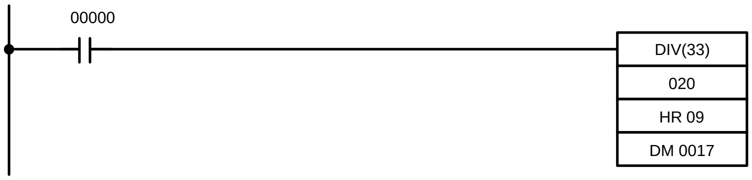
ER: Dd or Dr is not in BCD.

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

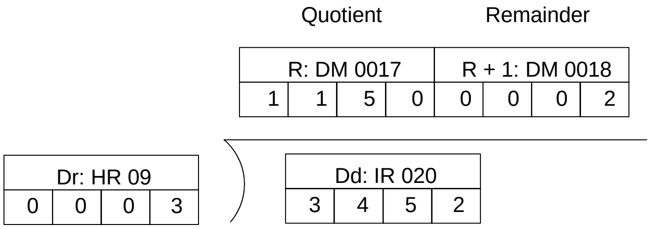
EQ: ON when the result is 0.

Example 1: Division of 4-digit Numbers

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



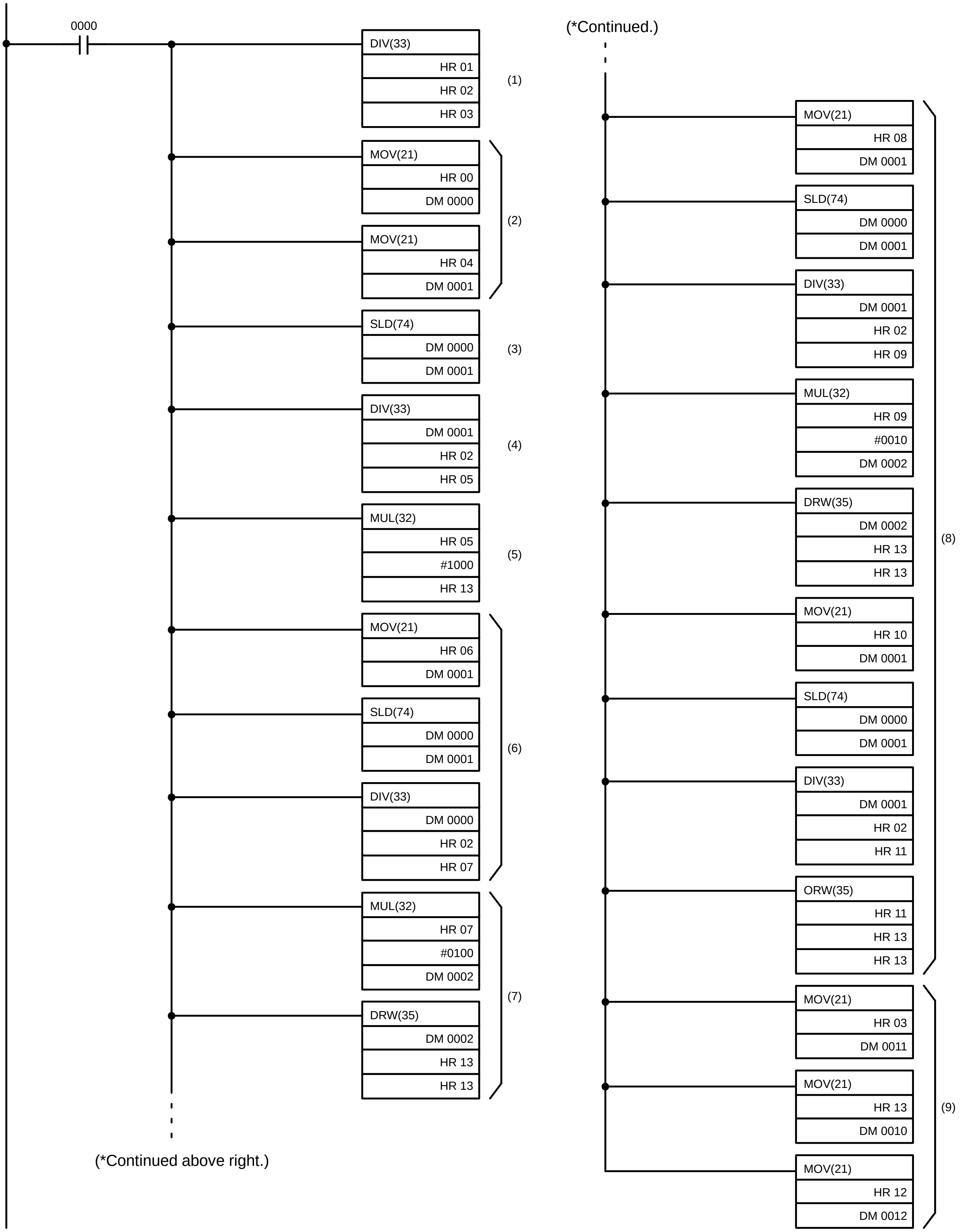
Quotient				Remainder			
R: DM 0017				R + 1: DM 0018			
1	1	5	0	0	0	0	2

Dr: HR 09			
0	0	0	3

Dd: IR 020			
3	4	5	2

Example 2: Division of 8-digits by 3-digits

In this example, the 8-digit content of HR 00 and HR 01 is divided by a 3-digit number in HR 02, the quotient is placed in DM 0010 and DM 0011, and the remainder is placed in DM 0012.



1, 2, 3... 1. The most significant 4-digits are divided first.

HR 01			
1	2	3	4

/

HR 02			
0	3	2	1

HR 03			
0	0	0	3

HR 04			
0	2	7	1

2. The quotient is placed in DM 0000 and the remainder is placed in DM 0001.

DM 0001			
0	2	7	1

DM 0000			
5	6	7	8

3. The 8 digits in DM 0000 and DM 0001 are shifted one digit to the left.

DM 0001			
2	7	1	5

DM 0000			
6	7	8	0

4. The fifth digit is divided, the quotient is placed in HR 05 and the remainder in HR 06.

DM 0001			
2	7	1	5

/

HR 02			
0	3	2	1

HR 05			
0	0	0	8

HR 06			
0	1	4	7

5. The quotient is multiplied by 1000 and placed in HR 13.

HR 13			
8	0	0	0

6. The sixth digit is divided in the same way, the quotient is placed in HR 07 and the remainder in HR 08.

HR 07			
0	0	0	4

HR 08			
0	1	9	2

7. The quotient is multiplied by 100 and placed in DM 0002. DM 0002 is logically ORed with HR 13 and the result is placed back in HR 13.

HR 13			
8	4	0	0

8. The same calculations are carried out on the seventh and eighth digits, and the results output to HR 13.

HR 13			
8	4	6	0

HR 12			
0	0	1	8

9. The contents of HR 03, HR 13, and HR 12 are transferred to DM 0011, DM 0010, and DM 0012, respectively. The final result of the division has the overall structure shown below.

HR 01				HR 00			
1	2	3	4	5	6	7	8

	HR 020			
/	0	3	2	1

DM0011				DM0010			
0	0	0	3	8	4	6	0

Remainder:	DM0012			
	0	0	1	8

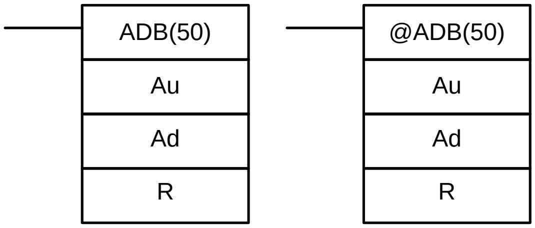
5-16 Binary Calculations

The binary calculation instructions – ADB(50), SBB(51), MLB(52) and DVB(53) – all perform arithmetic operations on binary (hexadecimal) data.

The addition and subtraction instructions include CY in the calculation as well as in the result. Be sure to clear CY if its previous status is not required in the calculation, and to use the result placed in CY, if required, before it is changed by the execution of any other instruction. STC(40) and CLC(41) can be used to control CY. Refer to *5-15 BCD Calculations*.

5-16-1 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, 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.

$$\boxed{Au} + \boxed{Ad} + \boxed{CY} \rightarrow \boxed{CY}\ \boxed{R}$$

Flags

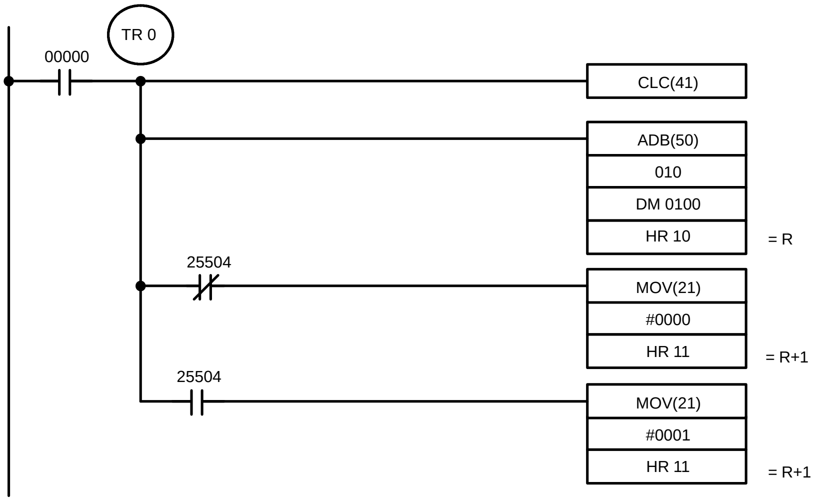
ER: Indirectly addressed DM word is non-existent. (Content of *DM word is not BCD, or the DM area boundary has been exceeded.)

CY: ON when the result is greater than FFFF.

EQ: ON when the result is 0.

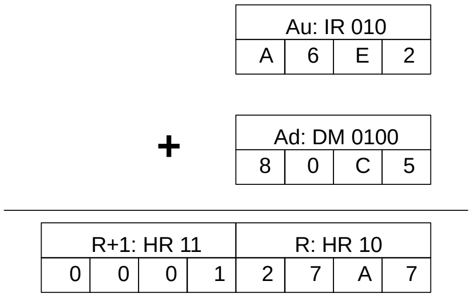
Examples

The following example shows a four-digit addition with CY used to place either 0000 or 0001 into R+1 to ensure that any Carry is preserved.

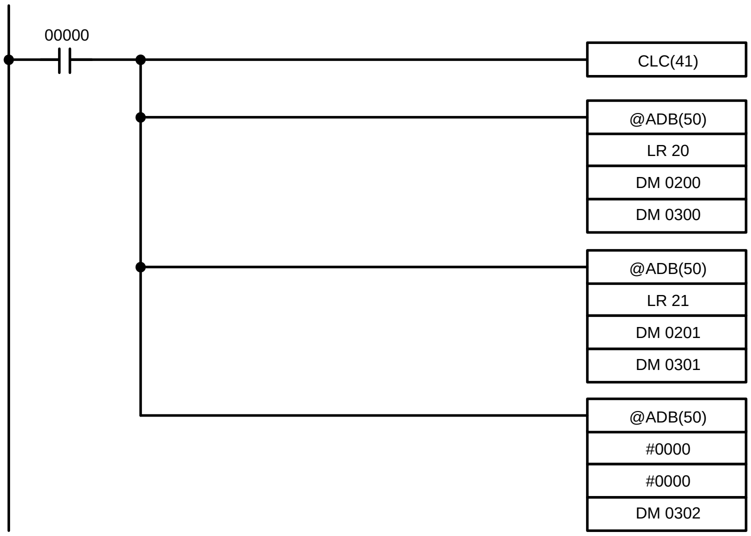


Address	Instruction	Operands
00000	LD	00000
00001	OUT	TR 0
00002	CLC(41)	
00003	ADB(50)	
		010
		DM 0100
		HR 10
00004	AND NOT	25504
00005	MOV(21)	
		# 0000
		HR 11
00006	LD	TR 0
00007	AND	25504
00008	MOV(21)	
		# 00001
		HR 11

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.

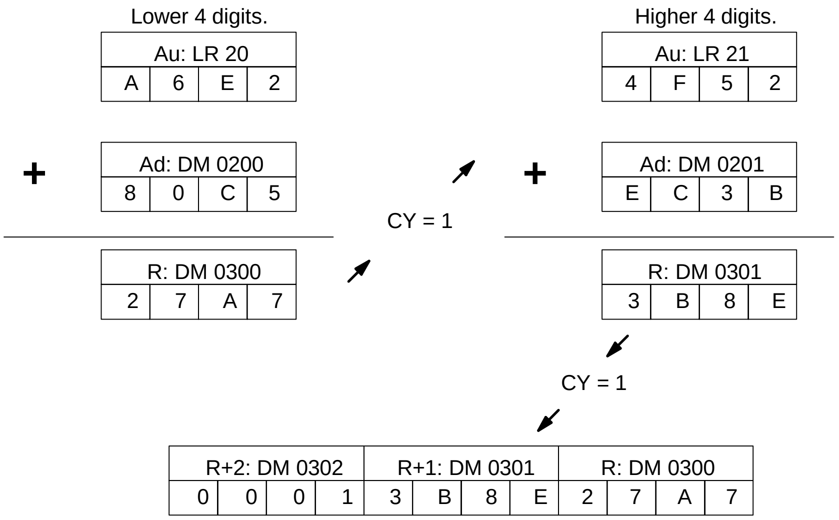


The following example performs eight-digit addition by using ADB(50) twice. ADB(50) is also used to place the carry into DM 0302 (one word greater than the rest of the answer). The complete answer thus ends up in DM 0300 through DM 0302.



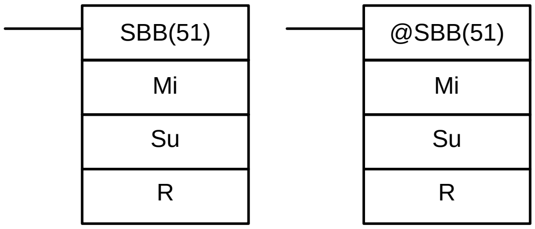
Address	Instruction	Operands	
00000	LD		00000
00001	CLC(41)		
00002	@ADB(50)		
		LR	20
		DM	0200
		DM	0300
00003	@ADB(50)		
		LR	21
		DM	0201
		DM	0301
00004	@ADB(50)		
		#	0000
		#	0000
		DM	0302

In the case below, 4F52A6E2 + EC3B80C5 = 13B8E27A7. The sum of the addition of the lower 4 digits is a 5-digit number, so CY (SR 25504) = 1, and the sum of the higher 4-digit addition is incremented by 1.



5-16-2 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, 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.

$$\boxed{\text{Mi}} - \boxed{\text{Su}} - \boxed{\text{CY}} \rightarrow \boxed{\text{CY}} \quad \boxed{\text{R}}$$

Flags

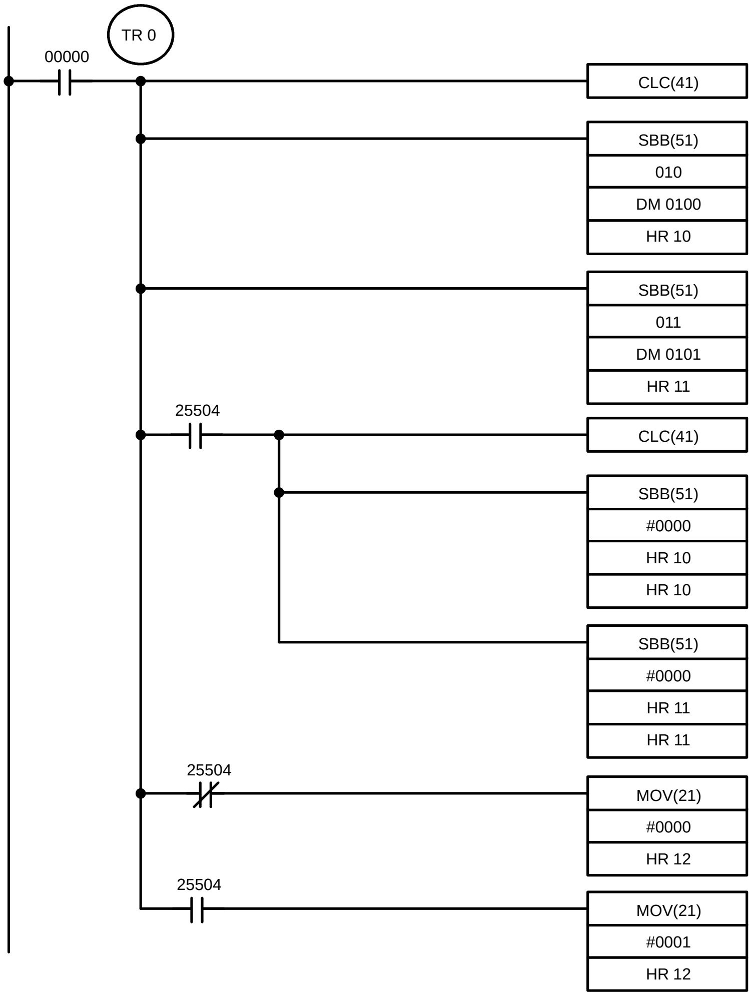
ER: Indirectly addressed DM word is non-existent. (Content of *DM word is not BCD, or the DM area boundary has been exceeded.)

CY: ON when the result is negative, i.e., when Mi is less than Su plus CY.

EQ: ON when the result is 0.

Example

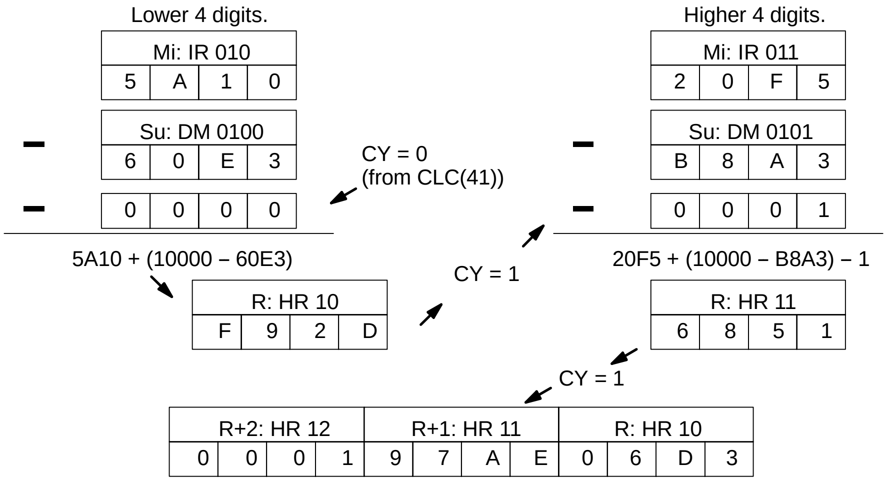
The following example shows eight-digit subtraction. CY is tested following the first two subtractions to see if the result is negative. If it is, the first result is subtracted from zero to obtain the true result, which is placed in HR 10 and HR 11, and either 0000 or 0001 is placed in HR 12 (0001 indicates a negative answer).



Address	Instruction	Operands	
00000	LD		00000
00001	OUT	TR	0
00002	CLC(41)		
00003	SBB(51)		
			010
		DM	0100
		HR	10
00004	SBB(51)		
			011
		DM	0101
		HR	11
00005	AND		25504
00006	CLC(41)		
00007	SBB(51)		
		#	0000
		HR	10
		HR	10
00008	SBB(51)		
		#	0000
		HR	11
		HR	11
00009	LD	TR	0
00010	AND NOT		25504
00011	MOV(21)		
		#	0000
		HR	12
00012	LD	TR	0
00013	AND		25504
00014	MOV(21)		
		#	0001
		HR	12

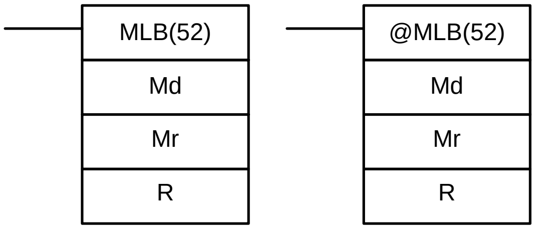
In the case below, 20F55A10 – B8A360E3 = 97AE06D3. In the the lower 4-digit subtraction, Su > Mi, so CY(SR 25504) becomes 1, and the result of the higher 4-digit subtraction is decremented by 1. In the final calculations, #0000 – F9D2 = 0000 + (10000 – F9D2) = 06D3.

#0000 – 6851 –1 (from CY = 1) = 0000 + (10000 – 6851 – 1) = 97AE.
The content of HR 12, #0001, indicates a negative result.



5-16-3 BINARY MULTIPLY – MLB(52)

Ladder Symbols



Operand Data Areas

Md: Multiplicand word (binary)

IR, SR, AR, DM, HR, TC, LR, #

Mr: Multiplier word (binary)

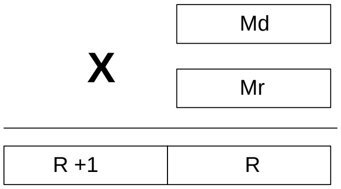
IR, SR, AR, DM, HR, TC, LR, #

R: First result word

IR, AR, DM, HR LR

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.



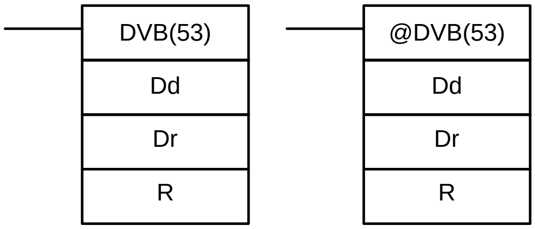
Flags

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.

5-16-4 BINARY DIVIDE – DVB(53)

Ladder Symbols



Operand Data Areas

Dd: Dividend word (binary)

IR, SR, AR, DM, HR, TC, LR, #

Dr: Divisor word (binary)

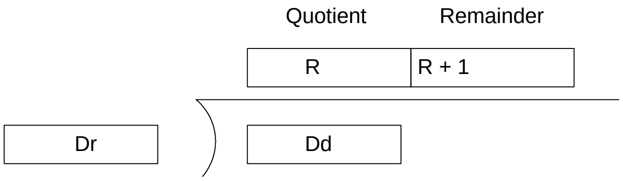
IR, SR, AR, DM, HR, TC, LR, #

R: First result word

IR, 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.



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.)

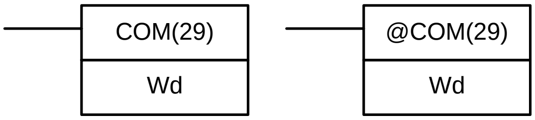
EQ: ON when the result is 0.

5-17 Logic Instructions

The logic instructions – COM(29), ANDW(34), ORW(35), XORW(36), and XNRW(37) – perform logic operations on word data.

5-17-1 COMPLEMENT – COM(29)

Ladder Symbols



Operand Data Areas

Wd: Complement word

IR, 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

	15															00
Original	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1

	15															00
Complement	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0

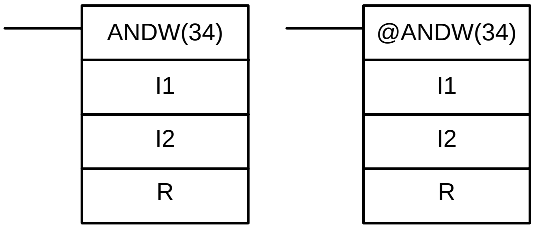
Flags

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.

5-17-2 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, 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

	15															00
I1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1

	15															00
I2	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

⬇

	15															00
R	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

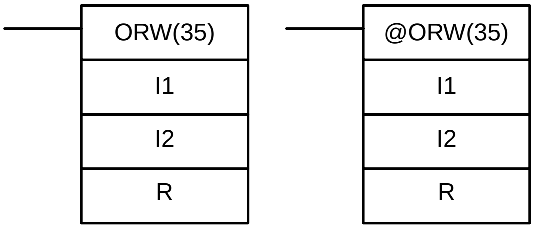
Flags

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.

5-17-3 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, 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

	15															00
I1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
I2	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
R	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1

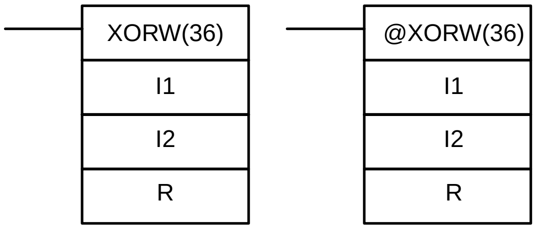
Flags

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.

5-17-4 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, 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

	15															00
I1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
I2	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
R	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0

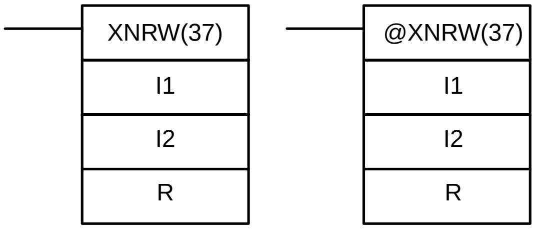
Flags

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.

5-17-5 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, 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.

	15															00
I1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1

	15															00
I2	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

⬇

	15															00
R	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1

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.

5-18 Subroutines

5-18-1 Overview

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 the main program code. When all the subroutine instructions have been executed, control returns to the main program to the point just after the point from which the subroutine was entered (unless otherwise specified in the subroutine).

5-18-2 SUBROUTINE START and RETURN – SBN(92)/RET(93)

Ladder Symbols

Operand Data Areas

N: Subroutine number
(00 to 49)

Limitations

Each subroutine number can be used in SBN(92) once only, i.e., up to 50 subroutines may be programmed.

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 *5-18-3 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. (Refer to *5-18-3 SUBROUTINE ENTER – SBS(91)* for further details.)

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.

5-18-3 SUBROUTINE ENTER – SBS(91)

Ladder Symbol

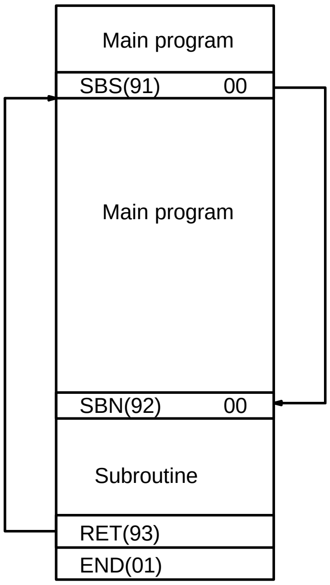
Operand Data Areas

N: Subroutine number
(00 to 49)

Description

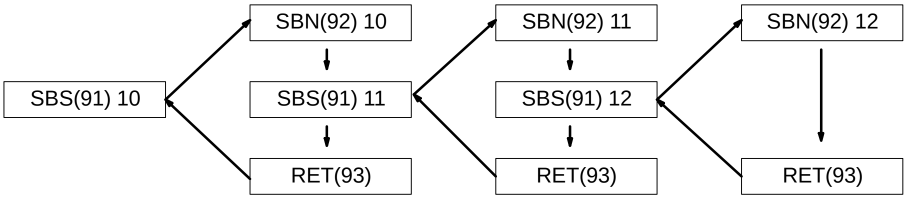
A subroutine can be executed by placing SBS(91) in the main program at the point where the subroutine is desired. The subroutine number used in

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.

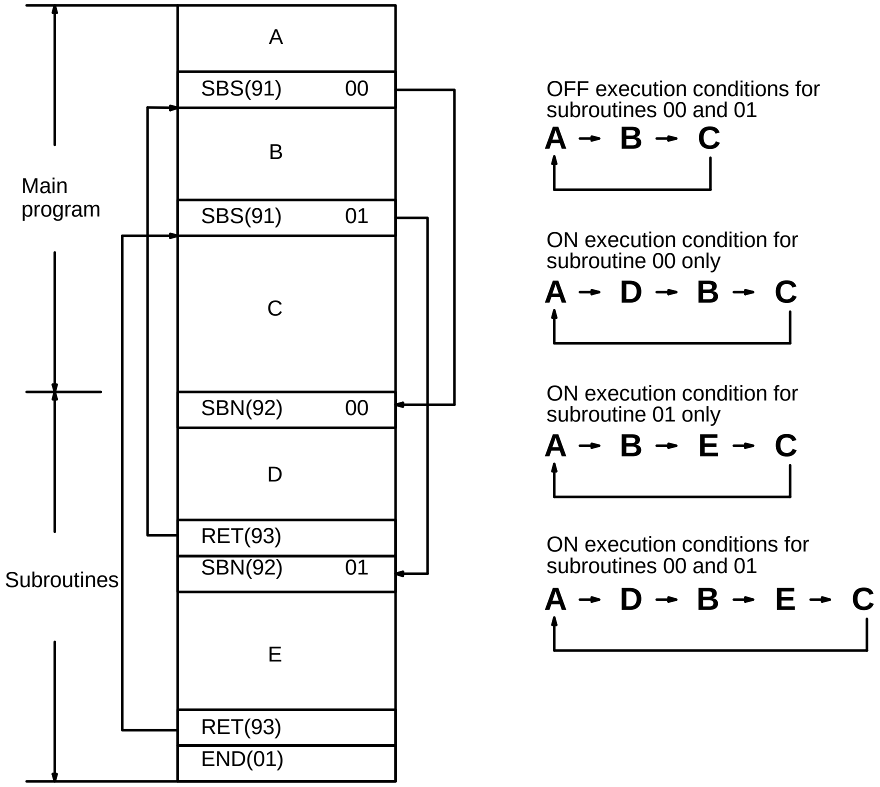


SBS(91) may be used as many times as desired in the program, i.e., the same subroutine may be called from different places in the program).

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 of up to sixteen levels is possible. 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).


A
SBS(91) 00
B
SBS(91) 01
C
SBN(92) 00
D
RET(93)
SBN(92) 01
E
RET(93)
END(01)
Main program
Subroutines
OFF execution conditions for subroutines 00 and 01
A → B → C
ON execution condition for subroutine 00 only
A → D → B → C
ON execution condition for subroutine 01 only
A → B → E → C
ON execution conditions for subroutines 00 and 01
A → D → B → E → C

Flags

ER: A subroutine does not exist for the specified subroutine number.

A subroutine has called itself.

Subroutines have been nested to more than sixteen levels.

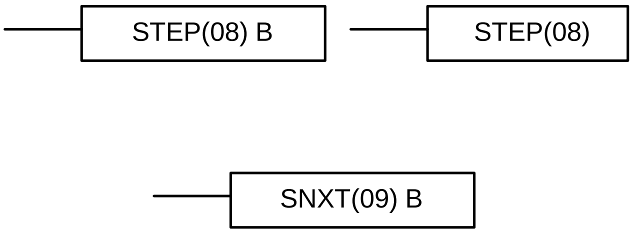
Caution SBS(91) will not be executed and the subroutine will not be called when ER is ON.

5-19 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 normal programming code, except that certain instructions (e.g. IL(02)/ILC(03), JMP(04)/JME(05)) may not be included.

5-19-1 STEP DEFINE and STEP START–STEP(08)/SNXT(09)

Ladder Symbols



Definer Data Areas

B: Control bit
IR, AR, HR, LR

B: Control bit
IR, AR, HR, LR

Limitations

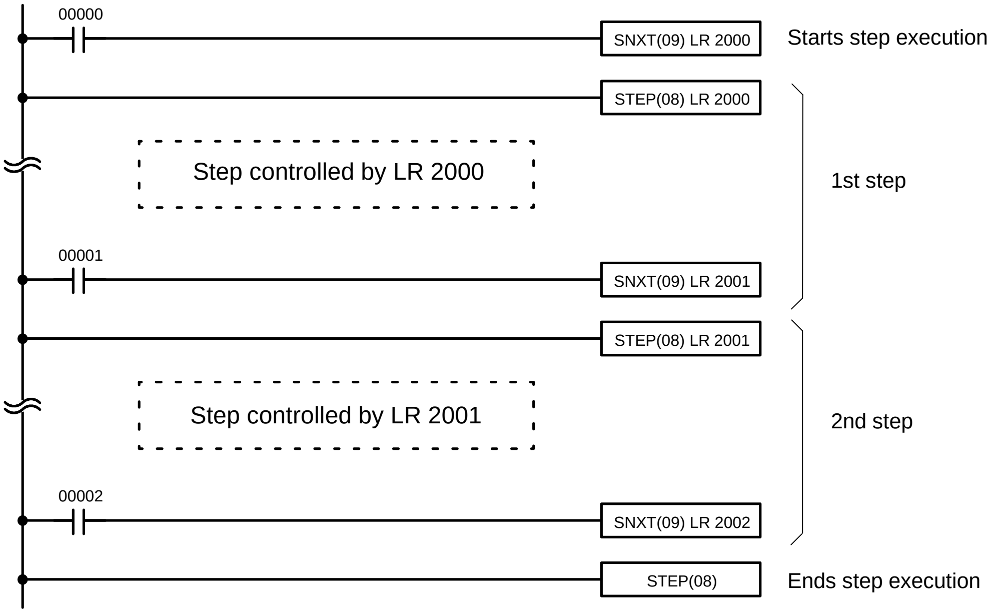
All control bits must be in the same word and must be consecutive.

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 and all timers in the step are reset to their SVs. Counters, shift registers, and bits used in KEEP(11) maintain status. Two simple examples are shown below.



Address	Instruction	Operands
00000	LD	00000
00001	SNXT(09)	LR 2000
00002	STEP(08)	LR 2000
	Step controlled by LR 2000.	
00100	LD	00001
00101	SNXT(09)	LR 2001
00102	STEP(08)	LR 2001
	Step controlled by LR 2001.	
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

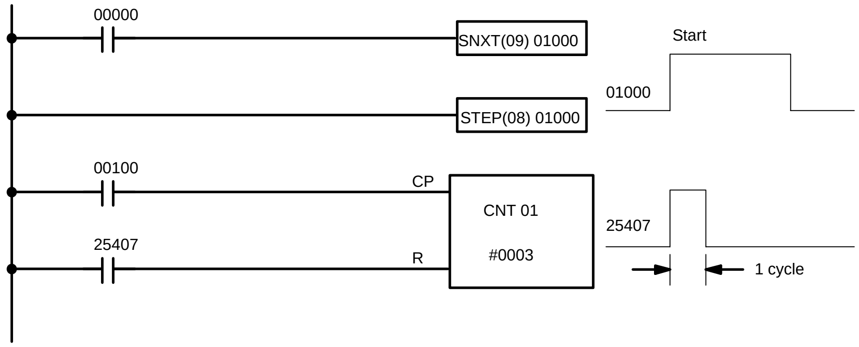
Interlocks, jumps, SBN(92), and END(01) cannot be used within step programs.

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

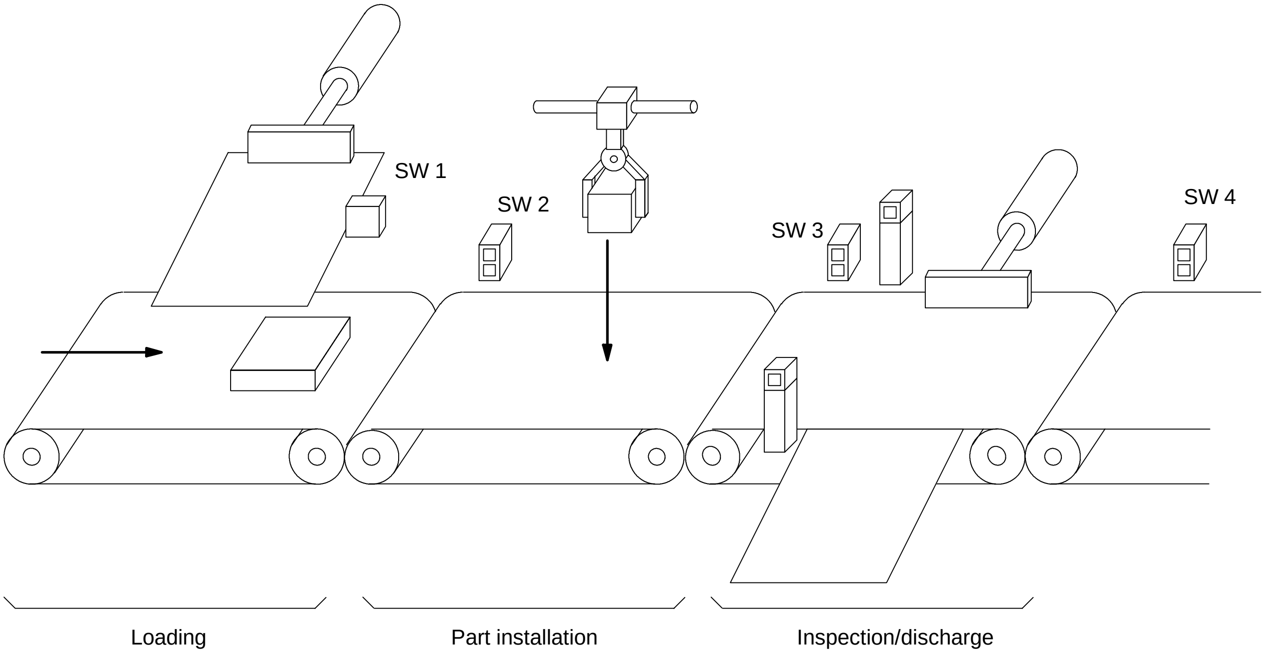
Address	Instruction	Operands
00004	LD	25407
00005	CNT	01
		# 0003

Examples

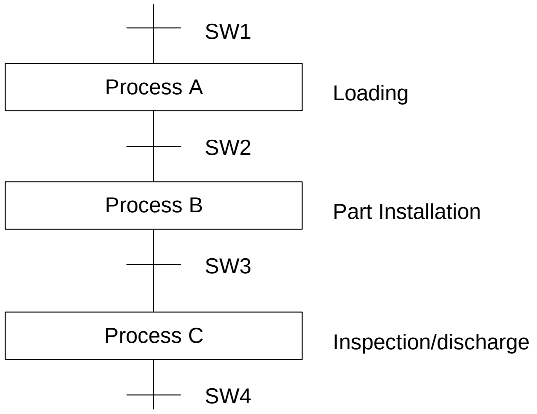
The following three examples demonstrate the three types of execution control possible with step programming. Example 1 demonstrates sequential execution; example 2, branching execution; and example 3, parallel execution.

Example 1: Sequential Execution

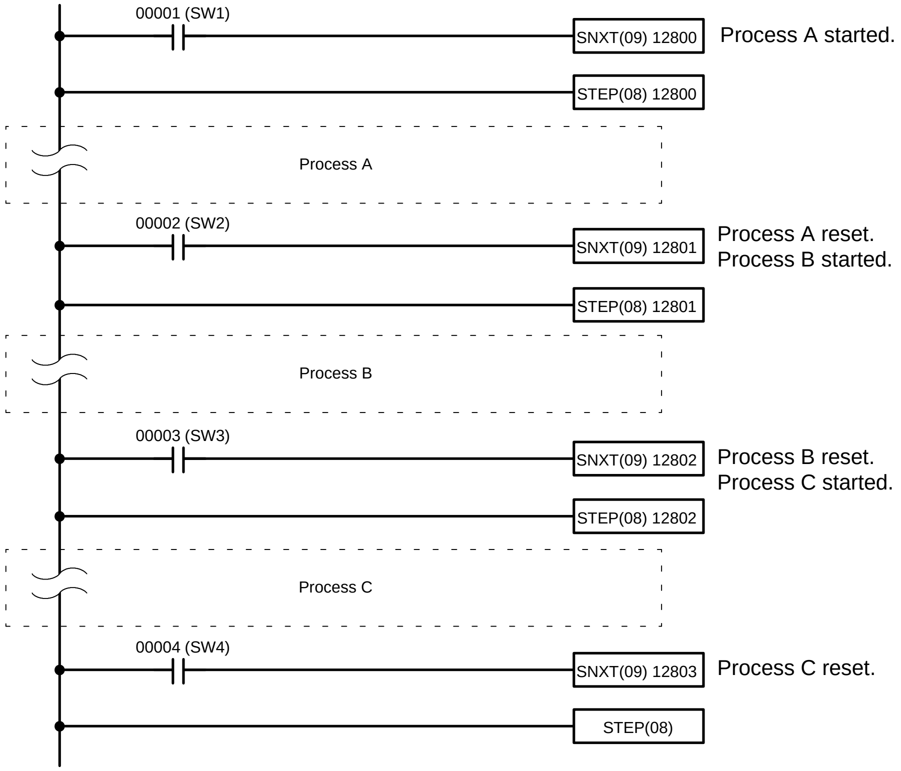
The following process requires that three processes, loading, part installation, and inspection/discharge, be executed in sequence with each process being reset before continuing on the the next process. Various sensors (SW1, SW2, SW3, and SW4) are positioned to signal when processes are to start and 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.

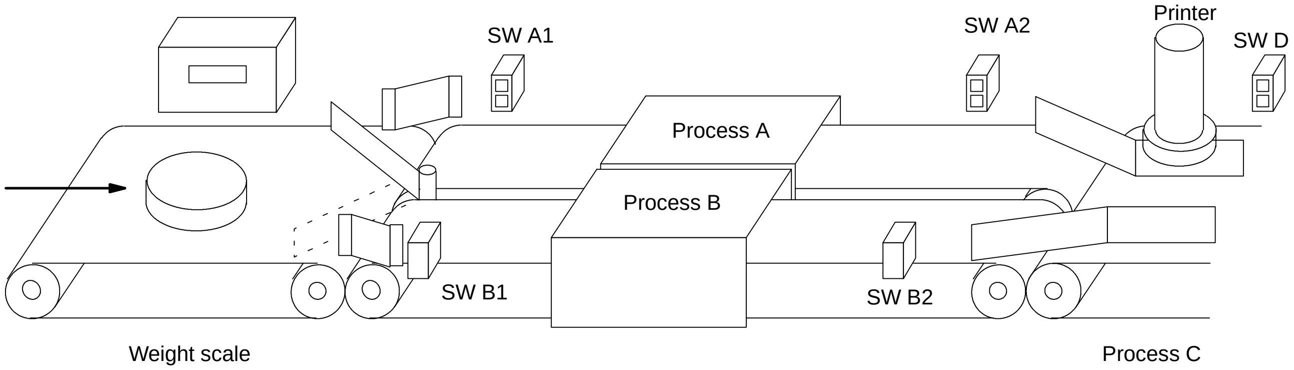


Address	Instruction	Operands
00000	LD	00001
00001	SNXT(09)	12800
00002	STEP(08)	12800
	Process A	
00100	LD	00002
00101	SNXT(09)	12801
00102	STEP(08)	12801

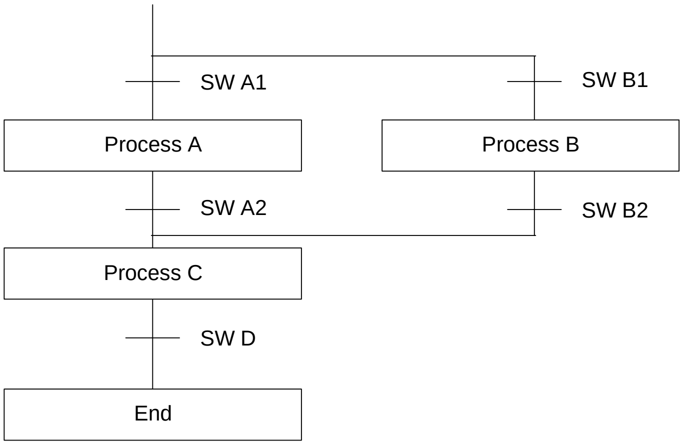
Address	Instruction	Operands
	Process B	
00100	LD	00003
00101	SNXT(09)	12802
00102	STEP(08)	12802
	Process C	
00200	LD	00004
00201	SNXT(09)	12803
00202	STEP(08)	---

Example 2: Branching Execution

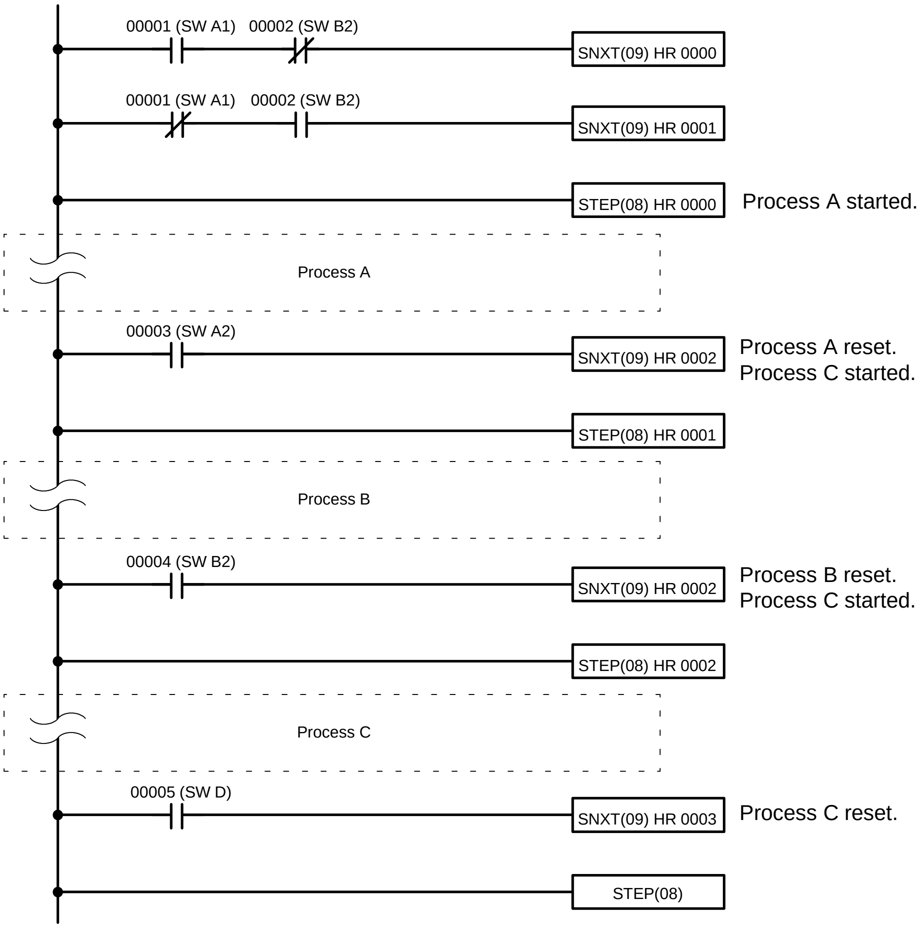
The following process requires that a product is processed in one of two ways, depending on its weight, before it is printed. The printing process is the same regardless of which of the first processes is used. Various sensors are positioned to signal when processes are to start and end.



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.

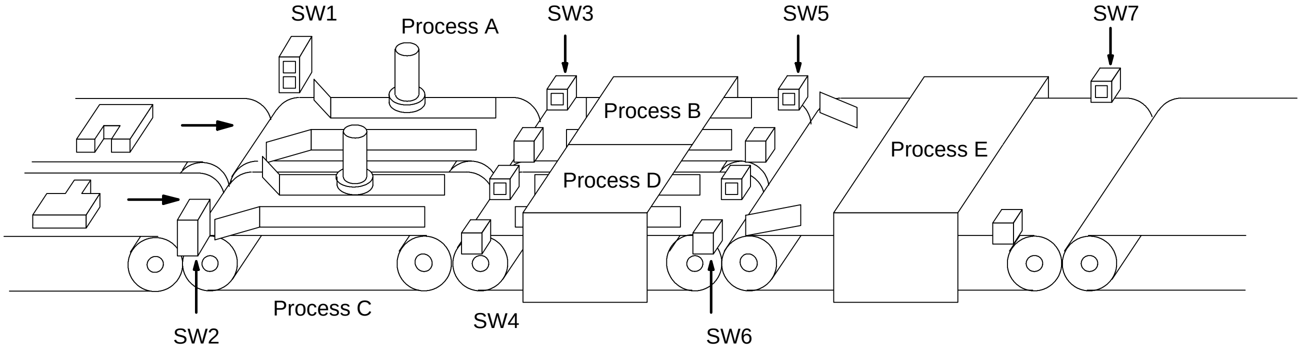


Address	Instruction	Operands	
00000	LD		00001
00001	AND NOT		00002
00002	SNXT(09)	HR	0000
00003	LD NOT		00001
00004	AND		00002
00005	SNXT(09)	HR	0001
00006	STEP(08)	HR	0000
	Process A		
00100	LD		00003
00101	SNXT(09)	HR	0002
00102	STEP(08)	HR	0001

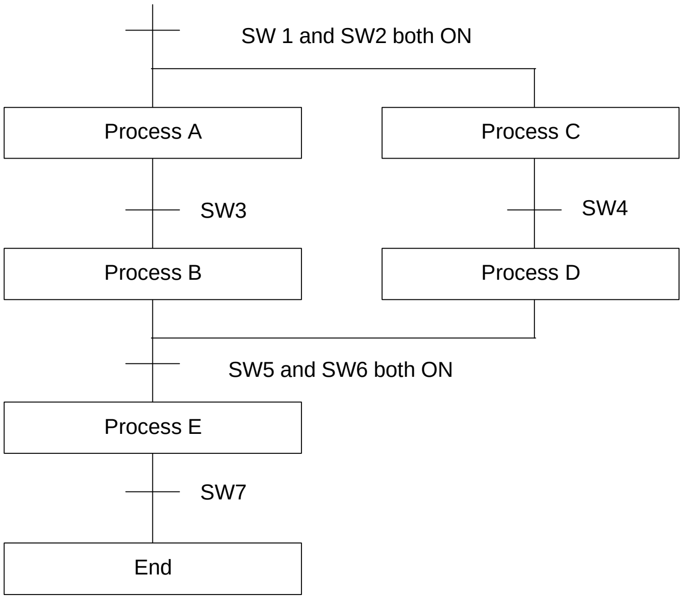
Address	Instruction	Operands	
	Process B		
00100	LD		00004
00101	SNXT(09)	HR	0002
00102	STEP(08)	HR	0002
	Process C		
00200	LD		00005
00201	SNXT(09)	HR	0003
00202	STEP(08)	---	

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.



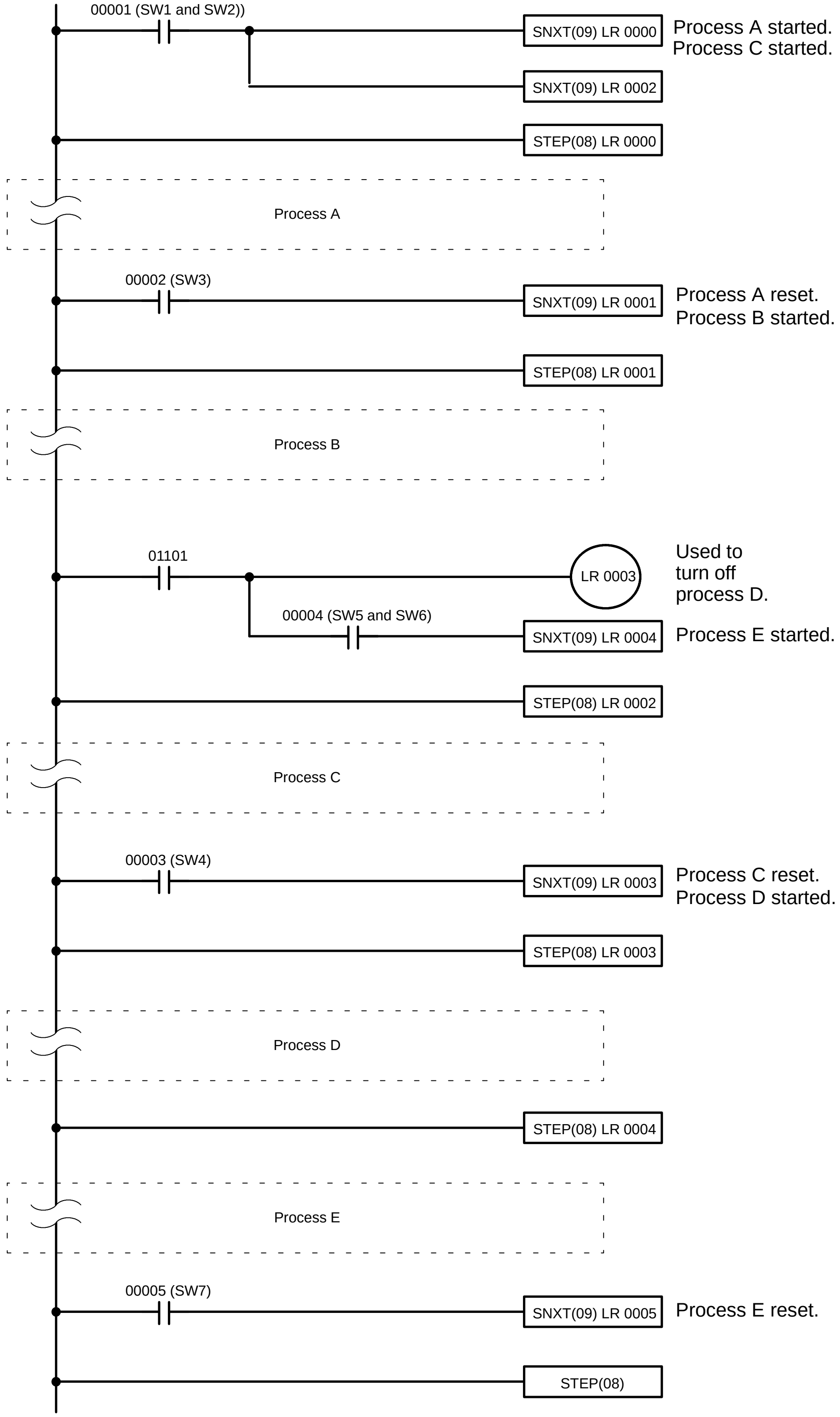
The following diagram demonstrates the flow of processing and the switches that are used for execution control. Here, process A and process C are started together. When process A finishes, process B starts; when process C finishes, process D starts. When both processes B and D have finished, process 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 SW5 and SW6 turn 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.



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

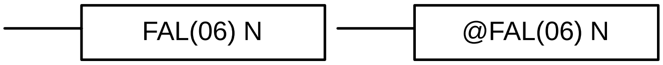
Address	Instruction	Operands
00102	STEP(08)	LR 0002
	Process C	
00200	LD	00003
00201	SNXT(09)	LR 0003
00202	STEP(08)	LR 0003
	Process D	
00300	STEP(08)	LR 0004
	Process E	
00400	LD	00005
00401	SNXT(09)	LR 0005
00402	STEP(08)	---

5-20 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.

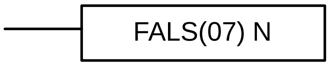
5-20-1 FAILURE ALARM – FAL(06) and SEVERE FAILURE ALARM – FALS(07)

Ladder Symbols



Definer Data Areas

N: FAL number
(00 to 99)

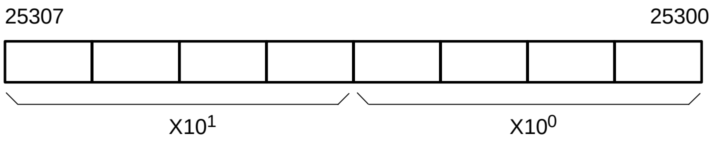


N: FAL number
(01 to 99)

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 instruction 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



When FAL(06) is executed with an ON execution condition, the warning indicator on the front of the CPU will light, but PC operation will continue. When FALS(07) is executed with an ON execution condition, the alarm indicator will light and PC operation will stop.

The system also generates error codes to the FAL area.

Resetting Errors

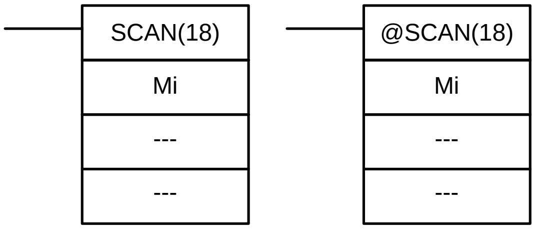
A maximum of three FAL error codes will be retained in memory, although only one of these is available in the FAL area. To access the other FAL codes, reset the FAL area by executing FAL(06) 00. Each time FAL(06) 00 is executed, another FAL error will be moved to the FAL area, clearing the one that is already there.

FAL(06) 00 is also used to clear messages 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 *7-1 Displaying and Clearing Error Messages*).

5-20-2 CYCLE TIME – SCAN(18)

Ladder Symbols



Operand Data Areas

Mi: Multiplier (BCD)

IR, AR, DM, HR, TC, LR, #

---: Not used.

---: Not used.

Limitations

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 0120, the minimum cycle time will be 12.0 ms. The possible setting range is from 0 to 999.9 milliseconds.

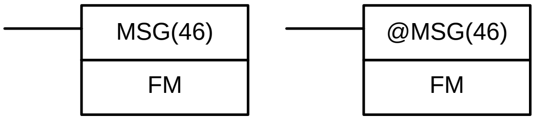
If the actual cycle time is less than the cycle time set with SCAN(18) the CPU will wait until the designated time has elapsed before starting the next cycle. If the actual cycle time is greater than the set time, the set time will be ignored and the program will be executed to completion.

Flags

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

5-20-3 DISPLAY MESSAGE – MSG(46)

Ladder Symbols



Operand Data Areas

FM: First message word
IR, AR, DM, HR, LR

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 or on the GPC. 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.

Note If different messages are being used, some characters from longer messages will remain on the display when a shorter message is output. Input spaces after shorter messages so that the messages are the same length and all of the characters of the longer messages will be overwritten.

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 > 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 *7-1 Displaying and Clearing Error Messages*.

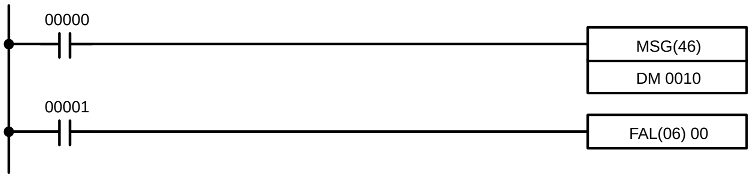
If the message data changes while the message is being displayed, the display will also change.

Flags

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

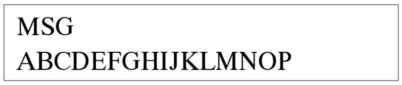
Example

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.



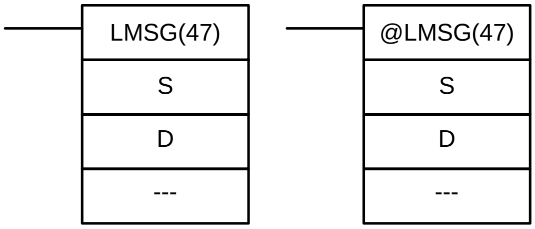
Address	Instruction	Operands	
00000	LD		00000
00001	MSG(46)		
		DM	0010
00002	LD		00001
00003	FAL(06)		00

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


```
MSG
ABCDEFGHIJKLMNOP
```

5-20-4 LONG MESSAGE – LMSG(47)

Ladder Symbols



Operand Data Areas

S: First source word (ASCII)

IR, AR, DM, HR, LR

D: Destination

(000, 001, or 002)

---: Not used.

Limitations

A maximum of 32 characters are permitted. S through S+15 must be in the same data area and must be in ASCII code. The message data string is terminated with a null character (00) placed in the S and S+15 range. The last character before the null character must always be the ASCII code for a carriage return (0D).

Description

LMSG(47) is used to output from 1 to 32 ASCII characters through either the Programming Console or the built-in RS-232C port. The message to be output must be in ASCII beginning in S and ending in S+15. The last character in the message must be the ASCII code for a carriage return (0D) or the message will not be sent. If a message shorter than 32 characters is desired, placing a null character (00) after the carriage return (0D) will truncate the message at that point. No characters after the null character will be output.

The destination of the message is designated in D. (Note that D appears to be a word number address. However in this case it is not.) Provided that the Programming Console is set to the TERMINAL mode (refer to *4-4-1 TERMINAL and CONSOLE Mode* for details), if D is 000, the output will be to the

Programming Console. Provided that the System Parameter for the RS-232C port has been set to ASCII I/O mode (refer to *3-8-2 System DM* and *3-5-11 System Command Bits*), if D is 001, the output will be from the leftmost byte through the serial port. If D is 002, the output will be from the rightmost byte through the serial port. If neither TERMINAL nor ASCII mode is set, the instruction will still execute but no message data will appear.

When D is 001 or 002 all valid ASCII characters can be sent through the serial port. Refer to *Appendix I* for ASCII codes.

Flags

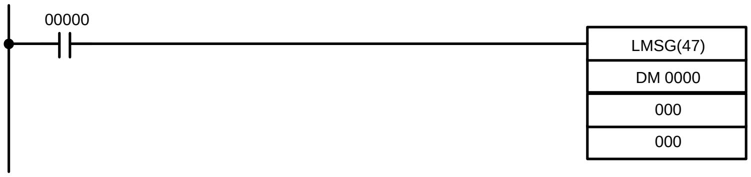
ER: S and S+15 are not in the same data area.

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

The communications buffer of the RS-232C interface is being used by another instruction.

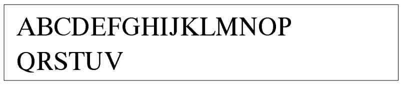
Example

The following example shows the display that would be produced for the instruction and data given when 00000 was ON with the Programming Console in TERMINAL mode. In TERMINAL mode the screen is cleared by sending two consecutive carriage return (0D) codes.



Address	Instruction	Operands	
00000	LD		00000
00001	LMSG(47)		
		DM	0000
			000
			000

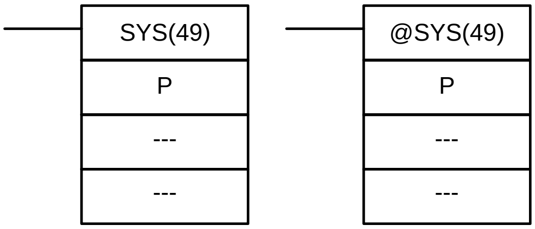
DM contents					ASCII equivalent	
DM 0000	4	1	4	2	A	B
DM 0001	4	3	4	4	C	D
DM 0002	4	5	4	6	E	F
DM 0003	4	7	4	8	G	H
DM 0004	4	9	4	A	I	J
DM 0005	4	B	4	C	K	L
DM 0006	4	D	4	E	M	N
DM 0007	4	F	5	0	O	P
DM 0008	5	1	5	2	Q	R
DM 0009	5	3	5	4	S	T
DM 0010	5	5	5	6	U	V
DM 0011	0	D	0	0	CR	0



```
ABCDEFGHIJKLMNOP
QRSTUV
```

5-20-5 SET SYSTEM – SYS(49)

Ladder Symbols



Operand Data Areas

P: Parameters
#

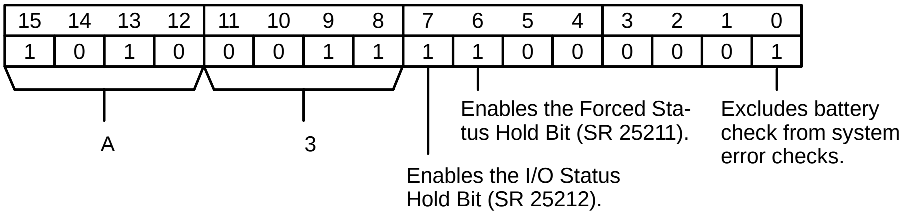
---: Not used.

---: Not used.

Limitations Only specific values are valid for P (see below).

Description SYS(49) has two functions; it can be used for bit control of certain operating parameters, or to execute the same system commands that are possible from the AR area. The contents bits 08 to 15 of P determine which function SYS(49) will have.

Bit Control If bits 08 to 15 of P contain A3, SYS(49) is used to set system operating parameters. To be effective, it must be programmed at program address 00001 with LD AR 1001 at program address 00000.



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

Enables the Forced Status Hold Bit (SR 25211).

Excludes battery check from system error checks.

Enables the I/O Status Hold Bit (SR 25212).

Only bits 00, 06, and 07 are used. If bit 00 is 1, the battery check will be excluded from system error checks when PC power is turned ON. If bit 06 is 1, the Forced Status Hold Bit (SR 25211) will be turned ON. If bit 07 is 1, the I/O Status Hold Bit (SR 25212) will ensure that I/O bit status is maintained when PC power is turned ON.

System Commands

If bits 08 to 15 of P contain 00, the system command indicated by the command code in bits 00 to 07 will be executed. As shown in the following table.

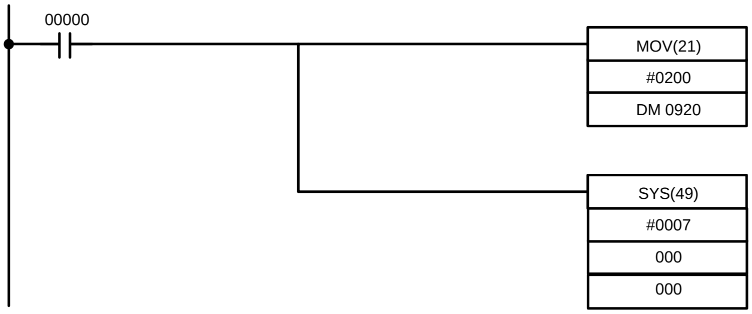
Command code	Name	Meaning
01	Parameter set	The contents of the Parameter Area (DM 0900 to DM 0929) are set into the system, the value of each parameter is checked for validity, all invalid values are replaced with the default values, and a checksum is generated.
02	Parameter backup	The contents of the Parameter Area (DM 0900 to DM 0929) is transferred to the Parameter Backup Area (DM 1900 to 1929), a checksum is generated, the data in the Parameter Backup Area is enabled, and AR 1314 (System Parameter Backup Flag) is turned ON. Because the Parameter Backup Area is contained in the Memory Unit, parameter backup will not be possible if the Memory Unit is write-protected or EPROM.
03	Backup disable	The data contained in the Parameter Backup Area is disabled and AR 1314 (System Parameter Backup Flag) is turned OFF.
04	Parameter clear	All words in the Parameter Area (DM 0900 to DM 0929) are turned OFF (i.e., set to zero).
05	General parameter set	Works in the same way as 01, but only DM 0900 to DM 0905 are set.
06	High-speed counter parameter set	Works in the same way as 01, but only DM 0905 to DM 0919 are set.
07	RS-232C parameter set	Works in the same way as 01, but only DM 0920 to DM 0929 are set.

Flags

ER: When bits 08 to 15 of P contain 00, bits 00 to 07 do not contain a valid command code (00 to 07).

Example

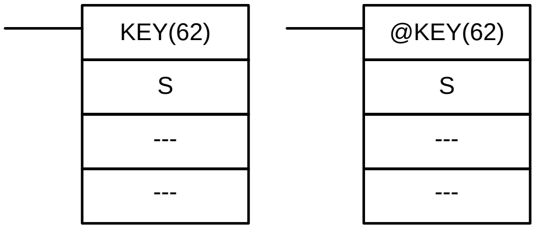
The following example shows how to change the RS-232C port from factory-set mode to ASCII output mode.



Address	Instruction	Operands
00000	LD	00000
00001	MOV(21)	
		# 0200
		DM 0920
00002	SYS(49)	
		# 0007
		000
		000

5-20-6 KEY INPUT – KEY(62)

Ladder Symbols



Operand Data Areas

S: First source word (key codes)

IR, AR, DM, HR, LR

---: Not used.

---: Not used.

Limitations

S and S+15 must be in the same data area and must contain key codes. SR 25503 must be OFF and a Programming Console must be mounted.

Description

KEY(62) is used to perform Programming Console operations from within the program. S designates the first word containing a key code. When KEY(62) is executed with an ON execution condition, the key codes will produce the same effect as pressing the equivalent Programming Console keys.

Key codes are executed from bits 08 to 15 first, then 00 to 07, then 08 to 15 of the next word, and so on. The key code string can be up to 16 words (i.e., 32 key equivalents) in length, but can be ended at any point by inputting a null code (00).

The Programming Console display can be reset to the initial display by inputting the reset code, 40.

Key codes are provided in *Appendix J*.

Flags

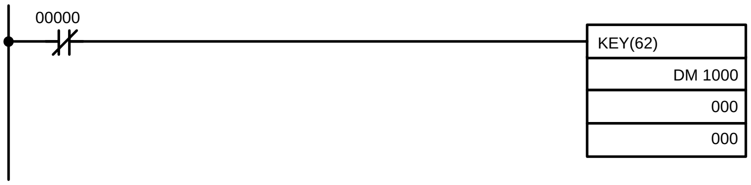
ER: The key code string is not within the same data area.

A Programming Console is not attached.

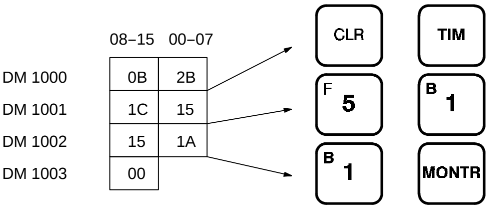
Indirectly addressed DM word is non-existent. (Content of *DM word is not BCD, or the DM area boundary has been exceeded.)

Example

When IR 00000 is ON, the following instruction would produce the key sequence shown for the contents given below for the operand words. When this instruction is executed, the ‘key sequence will display the current time of timer TIM 511 onto the Peripheral Device (e.g., Programming Console or Data Access Console).

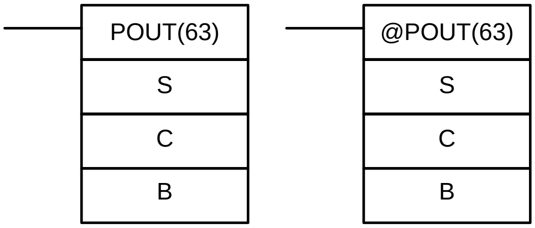


Address	Instruction	Operands
00000	LD NOT	00000
00001	KEY(62)	
		DM 1000
		000
		000



5-20-7 RS-232C PORT OUTPUT – POUT(63)

Ladder Symbols



Operand Data Areas

S: Source beginning word

IR, AR, DM, HR, TC, LR

C: Control number

# (0000 or 0001)

B: Number of bytes

IR, AR, DM, HR, TC, LR, #

Limitations

S through S+(B÷2)–1 must be within the same data area. C must be either #0000 or #0001. B must be BCD between 0000 and 0200.

The RS-232C Mode in DM 0920 must be set to ASCII I/O mode.

Description

When the execution condition is OFF, POUT(63) is not executed. When the execution condition is ON, POUT(63) outputs the B bytes of data in S to S+(B÷2)–1 through the RS-232C port. The control number, C, determines whether the leftmost (C=#0000) or rightmost (C=#0001) bytes in the words will be output first. Refer to *Section 8 RS-232C Interface* for details on the operation of the interface.

Start and End Codes

Bits 08 to 15 of DM 0925 specify whether or not there is a start code. If the content of these bits is 01, the start code (contained in DM 0925 bits 00 to 07) will be output before the data. If the content of bits 08 to 15 of DM 0925 is 00, the start code won't be output.

Likewise, bits 08 to 15 of DM 0926 specify whether or not there is an end code. If the content of these bits is 01, the end code (contained in DM 0926 bits 00 to 07) will be output after the data. If the content of bits 08 to 15 of DM 0926 is 00, the end code won't be output.

Communications Enable Flag

POUT(63) will not be executed unless the Communications Enable Flag (AR 0415) is ON. This flag will be ON if the RS-232C Mode in DM 0920 is set to ASCII I/O mode and the communications buffer is empty. If AR 0415 is OFF because the communications buffer is being used by LMSG(47) or another POUT(63) instruction, it will be turned ON again when the buffer is empty.

Precautions

When the RS-232C is set to ASCII I/O mode in DM 0920, communications are half duplex, so data cannot be received while POUT(63) is being executed and the reception buffer will be cleared when execution of POUT(63) is completed.

Do not use the same code for both the start and end codes.

Flags

ER: S and S+(B÷2)–1 aren't in the same data area.

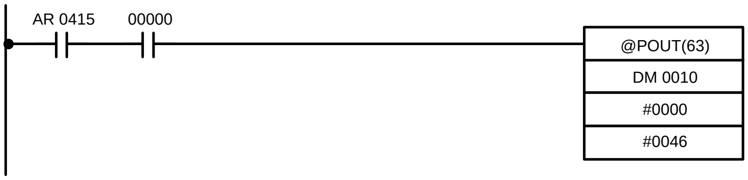
The POUT(63) instruction was executed while a LMSG(47) or POUT(63) instruction was being executed.

The RS-232C interface is not set to ASCII I/O mode.

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

Example 1

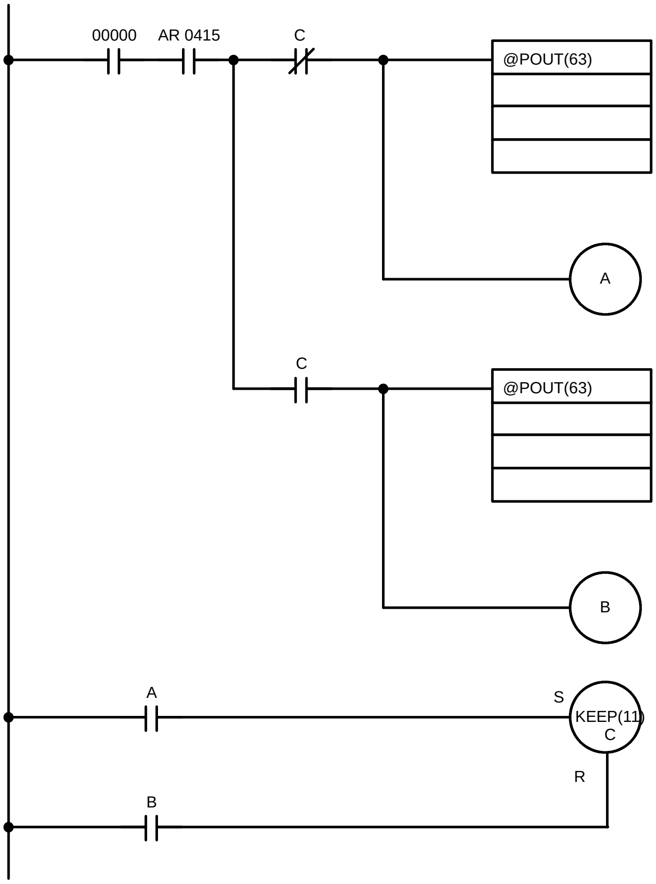
When AR 0415 and 00000 are ON, the following program outputs the 46 characters of ASCII data in DM 0010 to DM 0032 through the RS-232C interface.



Address	Instruction	Operands
00000	LD	AR 0415
00001	AND	00000
00002	@POUT(63)	
		DM 0010
		# 0000
		# 0046

Example 2

When more than one POUT(63) instruction is used in a program (to transmit more than 200 bytes of data, for example), it is necessary to write the program so that the instructions are executed consecutively. In this example, two POUT(63) instructions are used.

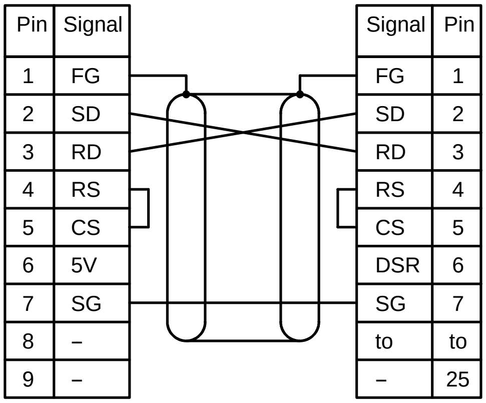


Example 3: Communication with an ID Controller

The POUT(63) instruction can be used to transmit commands and data to an ID Controller. The OMRON products used in this example are listed below.

Name	Model number
ID Controller	V620-CA1A
Read/Write Head	V620-H01
Data Carrier	V620-D2KR01
Monitor Unit	V600-P01

The PC is connected to the ID Controller with a 9-pin to 25-pin serial cable. (The PC has a 9-pin RS-232C interface and the ID Controller has a 25-pin RS-232C interface.) The following diagram shows the pin assignments and cable connections.



Pin	Signal
1	FG
2	SD
3	RD
4	RS
5	CS
6	5V
7	SG
8	–
9	–

Signal	Pin
FG	1
SD	2
RD	3
RS	4
CS	5
DSR	6
SG	7
to	to
–	25

Setting Communications Parameters

After establishing the physical connection between the PC and ID Controller, it is necessary to set the communications parameters. Set the communications parameters of the PC in the DM Area as shown below.

Word	Setting
DM 0920 (DM 1920)	0201 (ASCII I/O mode)
DM 0921 (DM 1921)	0205 (9,600 bps, 8 data bits, 1 stop bit, no parity)
DM 0922 (DM 1922)	0000 (no transmission delay, no RTS/CTS control)
DM 0926 (DM 1926)	010D (end code = 0D)

Set the communications parameters of the ID Controller with the DIP switch on the front of the Controller. Set the parameters for 9,600 bps, 1 start bit, 1 stop bit, 8 data bits, and no parity. Refer to the ID Controller's *System Manual* for details.

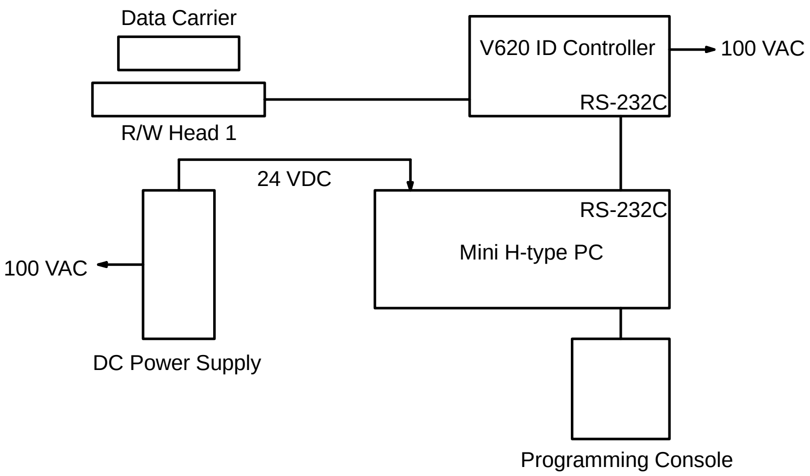
Setting Data for Transmission

The data to be transmitted to the ID Controller is set in the PC’s memory. In this example, the data and commands are set in the DM area as shown below. Refer to *Appendix I* for the extended ASCII table.

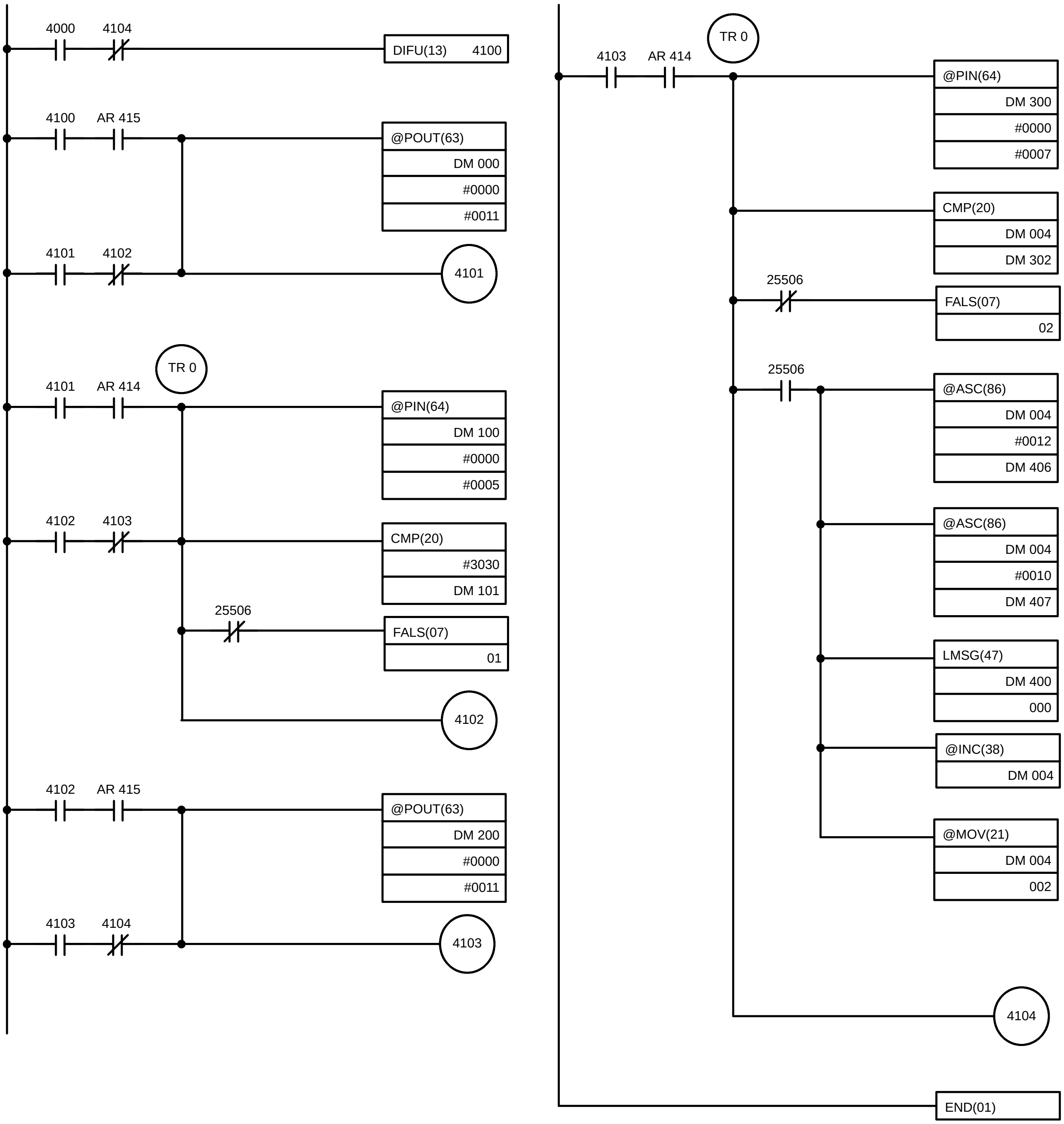
DM0000	5	7	5	4	⇨	“WT”	Write command header
DM0001	4	1	3	1	⇨	“AI”	ASCII code, R/W Head 1
DM0002	3	0	3	0	⇨	“00”	Beginning address (0006)
DM0003	3	0	3	6	⇨	“06”	
DM0004	0	0	0	0	⇨		Data to be written
DM0005	2	A	0	0	⇨	“* ”	
DM0200	5	2	4	4	⇨	“RD”	Read command header
DM0201	4	1	3	1	⇨	“AI”	ASCII code, R/W Head 1
DM0202	3	0	3	0	⇨	“00”	Beginning address (0006)
DM0203	3	0	3	6	⇨	“06”	
DM0204	3	0	3	2	⇨	“02”	Number of bytes to be read
DM0205	2	A	0	0	⇨	“* ”	
DM0400	4	3	5	5	⇨	“CU”	Programming Console* display data
DM0401	5	2	5	2	⇨	“RR”	
DM0402	4	5	4	E	⇨	“EN”	
DM0403	5	4	2	0	⇨	“T ”	
DM0404	4	4	4	1	⇨	“DA”	
DM0405	5	4	4	1	⇨	“TA”	
DM0406	3	A	2	0	⇨	“ : ”	
DM0407	0	0	0	0	⇨		
DM0100					⇨		Write command response
DM0101					⇨		
DM0102					⇨		
DM0300					⇨		
DM0301					⇨		Read command response
DM0302					⇨		
DM0303					⇨		

Note *Use the Programming Console in TERMINAL mode.

System Configuration

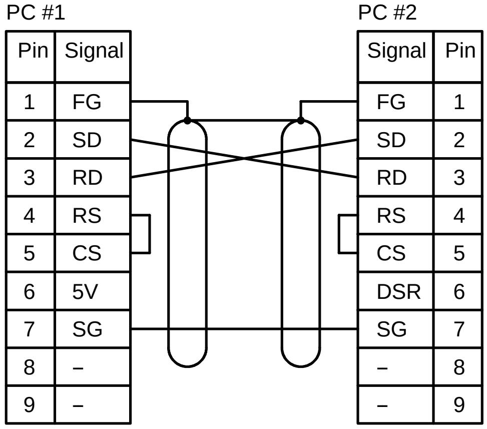


Sample Ladder Program



Example 4: Communications between Mini H-type PCs

In this example, the POUT(63) and PIN(64) instructions are used to transfer data between two PCs. The PCs are connected with a 25-pin to 25-pin serial cable. Refer to *5-20-8 RS-232C PORT INPUT – PIN(64)* for details on PIN(64). The following diagram shows the pin assignments and cable connections.



PC #1

Pin	Signal
1	FG
2	SD
3	RD
4	RS
5	CS
6	5V
7	SG
8	–
9	–

PC #2

Signal	Pin
FG	1
SD	2
RD	3
RS	4
CS	5
DSR	6
SG	7
–	8
–	9

Setting Communications Parameters

Set the communications parameters of the PCs as shown below.

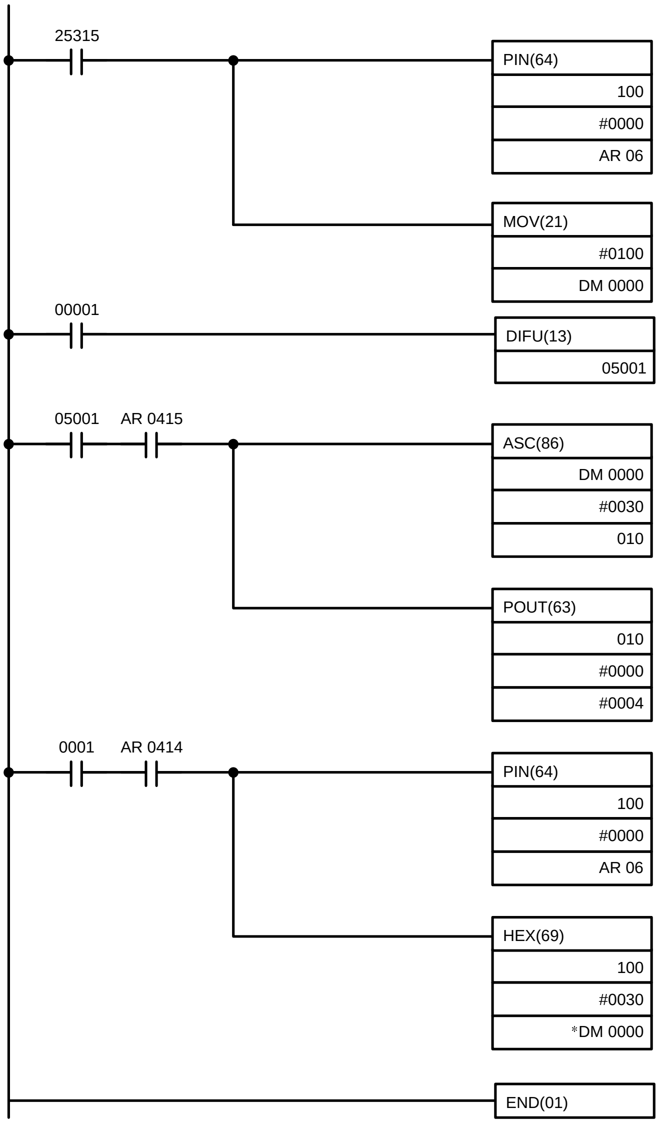
Word	Setting
DM 0920 (DM 1920)	0200 (ASCII I/O mode, standard settings)
DM 0926 (DM 1926)	010D (end code = 0D)

Data Link Program

The following ladder programs are used to change the content of a DM address in PC #1 to agree with the corresponding DM address in PC #2.

To accomplish this, the DM address is transmitted from PC #1 to PC #2, the content of the transmitted DM address is read in PC #2 and transmitted back to PC#1, and the content of the corresponding DM address in PC #1 is then changed to agree with the content of that address in PC #2.

The program below is written in PC #1.



Clears the communications buffer (dummy input).

Sets the DM address (0100) in DM 0000.

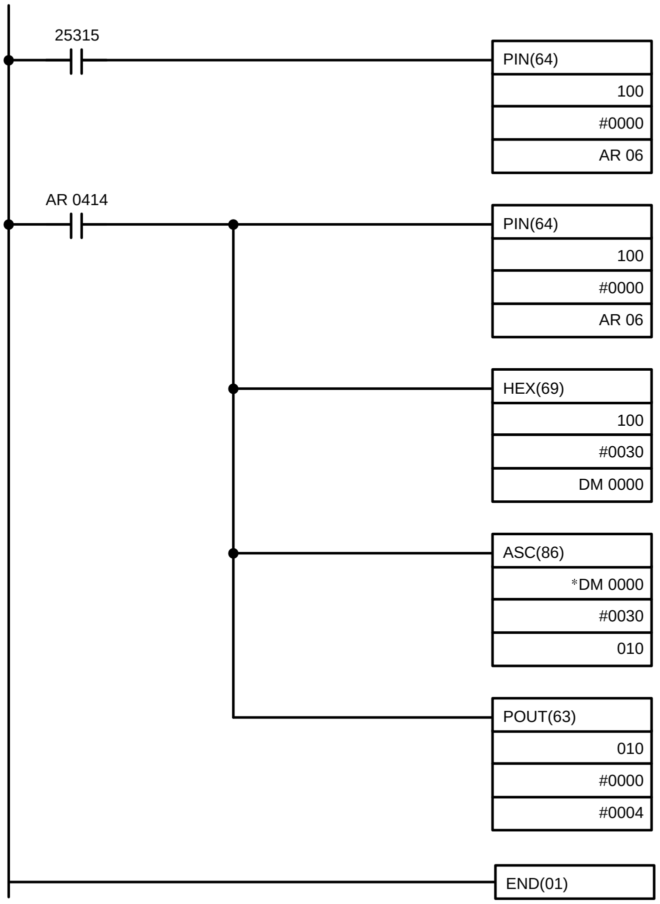
Converts the transmission data to ASCII code and outputs the result to IR 010 and IR 011.

Transmits the address (in ASCII code) to PC #2.

Places data received from PC #2 in IR 100 and IR 101.

Converts the received data from ASCII code to hexadecimal and outputs the result to the same DM address.

The program below is written in PC #2.



Clears the communications buffer (dummy input).

Places the DM address received from PC #1 (in ASCII code) in IR 100 and IR 101.

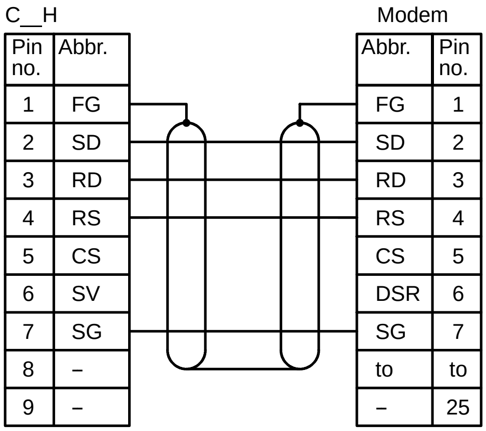
Converts the received DM address from ASCII code to hexadecimal and outputs the result to DM 0000.

Converts the content of the specified DM address to ASCII code and outputs the result to IR 010 and IR 011.

Transmits the contents of the specified DM address (in ASCII code) to PC #1.

Example 4: Communications through a Modem

In this example, the POUT(63) and PIN(64) instructions are used to transfer data to and from a modem. The PC is connected to the modem with a 9-pin to 25-pin serial cable. The modem used in this example is the OMRON MD24FP5V Intelligent Modem. Refer to *5-20-8 RS-232C PORT INPUT – PIN(64)* for details on PIN(64). The following diagram shows the pin assignments and cable connections.



C__H

Pin no.	Abbr.
1	FG
2	SD
3	RD
4	RS
5	CS
6	SV
7	SG
8	–
9	–

Modem

Abbr.	Pin no.
FG	1
SD	2
RD	3
RS	4
CS	5
DSR	6
SG	7
to	to
–	25

Cables and Connectors

The following cables and connectors are recommended for this example.

Cable
CO-MA-VV-SB 5PX28AWG (Hitachi)

Connectors

Mini H-type PC	Hood:	XM2A-0901 (OMRON)
	Plug:	XM2S-0911 (OMRON)
Modem	Hood:	XM2D-2501 (OMRON)
	Plug:	XM2S-2511 (OMRON)
	Bracket:	XM2Z-0001 (OMRON)

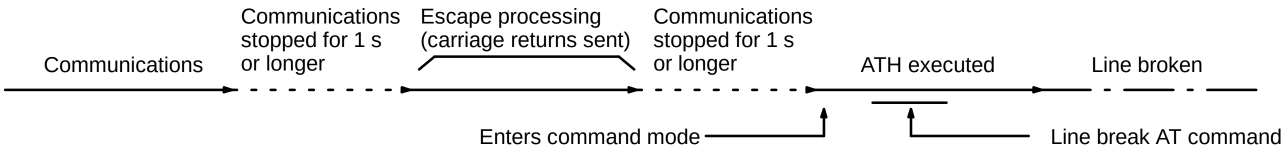
Modem Settings

Place the following settings into the modem's memory. Refer to the modem's operation manual for details.

Item	Setting
ER (DTR) signal	ER signal is ignored. Set to to remain ON. (Not supported by PC.)
Command echo	None
Result code	Set to use as a numeric parameter.
S register	S2 = 13 (carriage return) Although S2 is set to a carriage return, you may have difficulties in operation if the carriage return is also set to be treated as a character, e.g., when using the fall-back character feature.

Breaking Communications

The following procedure is used to break the communications line. This procedure is necessary because the PC does not support an ER signal.


Communications
Communications stopped for 1 s or longer
Escape processing (carriage returns sent)
Communications stopped for 1 s or longer
ATH executed
Line broken
Enters command mode
Line break AT command

Communications Procedures

The following procedures are used for transmission and reception. In either case, NCU must be set to AA in the modem.

Transmission The phone number is output and then data is output after connection of the communications line has been confirmed. A response of "1" is returned to the PC when the line has been connected.

Reception When a request is received, connection of the communications line is confirmed and data is received. A response of "2" is is returned to the PC as a request for reception and a response of "1" is returned to the PC when the line has been connected.

Setting Communications Parameters

Set the communications parameters of the PCs as shown below.

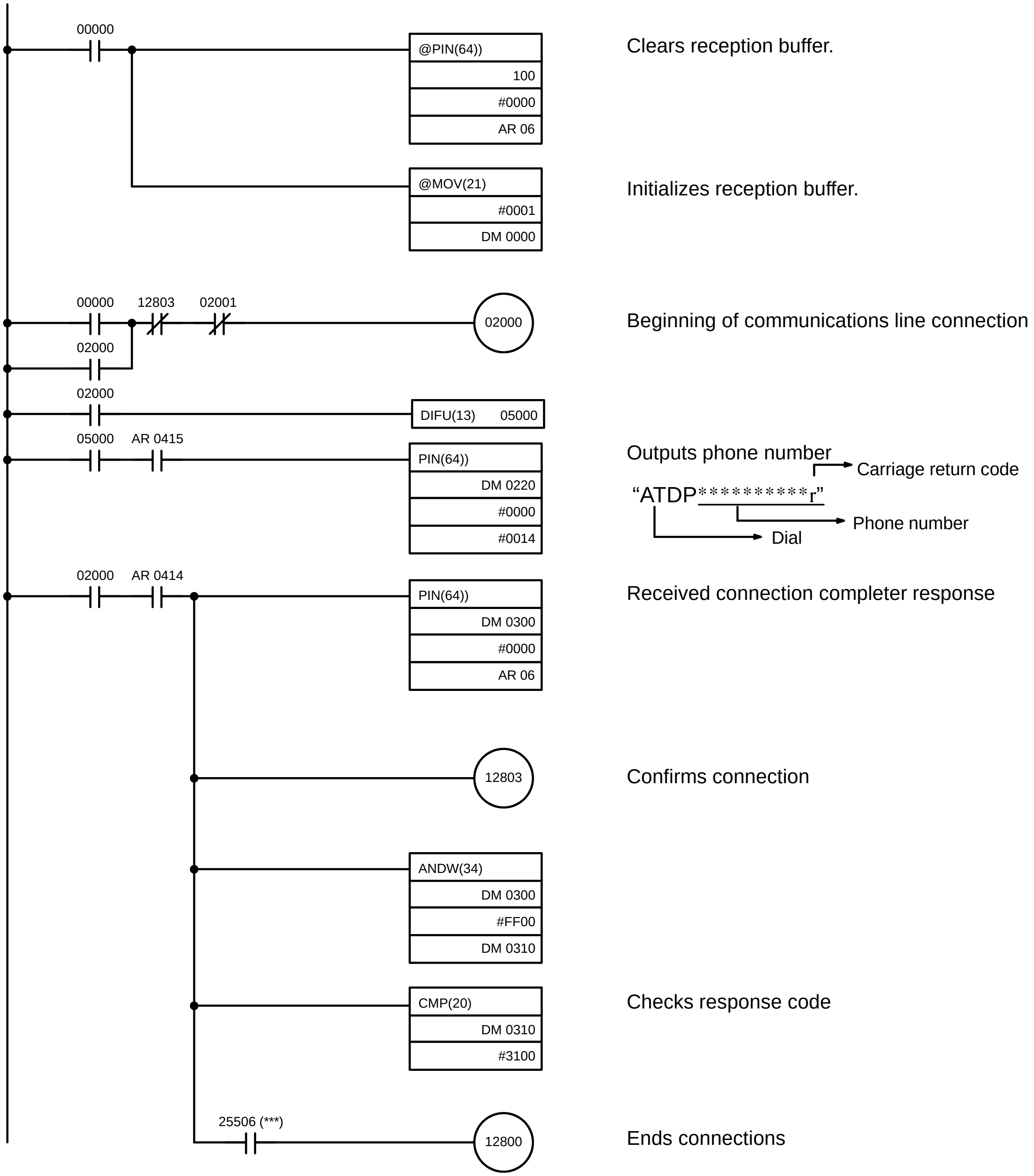
Word	Setting
DM 0920 (DM 1920)	0201 (ASCII I/O mode)
DM 0921 (DM 1921)	0203 (2,400 bps, 8 data bits, 1 stop bit, no parity)
DM 0922 (DM 1922)	0000 (no transmission delay, no RTS/CTS control)
DM 0926 (DM 1926)	010D (end code = 0D)

The following data is set in data memory.

Word	Setting
DM 0200	ODOD (rr)
DM 0210	4154 (AT)
DM 0211	4800 (H “null”)
DM 0220	4154 (AT)
DM 0221	4450 (DP)
DM 0222 to DM 0206	Phone number (in ASCII)

Data Link Program

The following ladder program is used to transmit data from the PC through the modem.



Clears reception buffer.

Initializes reception buffer.

Beginning of communications line connection

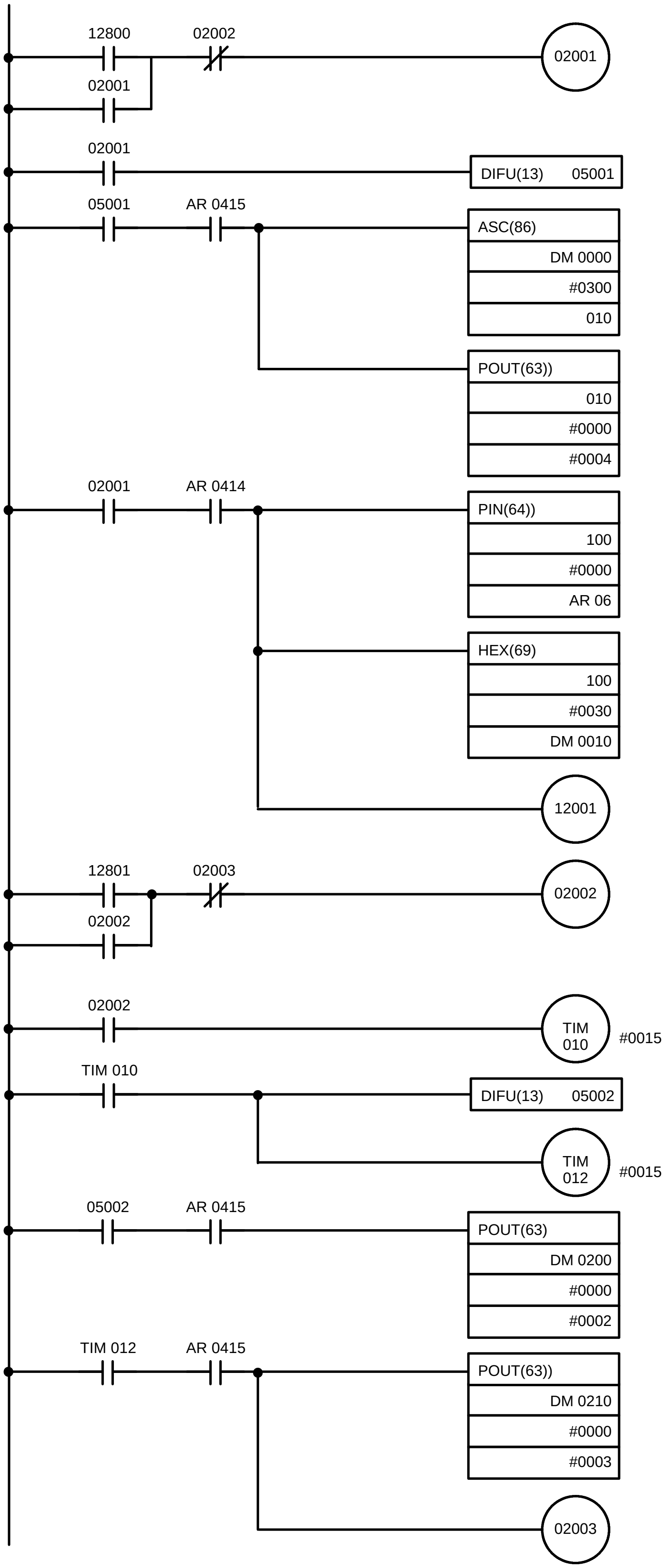
Outputs phone number

Received connection completer response

Confirms connection

Checks response code

Ends connections



Converts transmission data to ASCII

Transmits 4 bytes

Reads received data

Converts received data to hexadecimal

Communications Completed Flag

Stops communications for 1.5 s

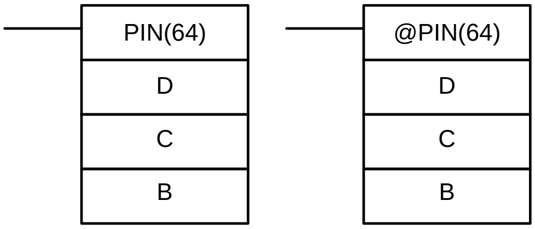
Stops communications for 1.5 s

Outputs two carriage returns to switch modem to command mode (third carriage return provided by end code).

Outputs command to break line (ATH)

5-20-8 RS-232C PORT INPUT – PIN(64)

Ladder Symbols



Operand Data Areas

D: Destination beginning word

IR, SR, AR, DM, HR, TC, LR

C: Control number

# (0000 or 0001)

B: Number of bytes

IR, SR, AR, DM, HR, TC, LR, #

Limitations

D through D+(B÷2)–1 must be within the same data area. C must be either #0000 or #0001. B must be BCD between 0000 and 0200.

Description

When the execution condition is OFF, PIN(64) is not executed. When the execution condition is ON, PIN(64) writes B bytes of data received through the RS-232C port to words beginning at D. If fewer than B bytes have been received at the communications buffer, only those bytes received will be input. The control number, C, determines whether the leftmost (C=#0000) or rightmost (C=#0001) byte in the words was input first. Refer to *Section 8 RS-232C Interface* for details on the operation of the interface. Refer to *5-20-7 RS-232C PORT OUTPUT – POUT(63)* for examples.

RS-232C Bytes Input Area

AR 08 contains the number of bytes of data (BCD) actually input to the PC by PIN(64). The number of bytes input will vary depending on conditions when the instruction is executed, as described below.

1, 2, 3...

1. If the number of bytes to input (operand B) is greater than or equal to the number of bytes received (content of AR 06), the number of bytes received will be input.
2. If B is less than the content of AR 06, B bytes will be input.
3. PIN(64) might be executed before the RS-232C Reception Completed Flag (AR 0414) is turned ON. In this case, the number of bytes received would equal the content of AR 06 just before execution. AR 06 would continue increasing as more data was received after execution.

Start and End Codes

Bits 08 to 15 of DM 0925 specify whether or not there is a start code. If the content of these bits is 01, data following the start code will be input. If the content of these bits is 00, all received data will be input.

Likewise, bits 08 to 15 of DM 0926 specify whether or not there is an end code. If the content of these bits is 01, data up to the end code will be input. If the content of bits 08 to 15 of DM 0926 is 00, data will be input until the reception buffer is full (200 bytes). If no end code is specified, data should be recorded as it arrives, before the reception buffer is full.

RS-232C Bytes Received Area

The number of bytes of data received through the RS-232C port (excluding the start and end codes) is output to AR 06 every cycle. To input all of the data received, use AR 06 as the number of bytes operand, B, when entering the instruction.

Reception Completed Flag

The Reception Completion Flag (AR 0414) is turned ON when the end code is received or the reception buffer is full (200 bytes received). Data cannot be received while this flag is ON. AR 0414 is turned OFF after PIN(64) has been executed.

Reception Impossible Flag

The RS-232C Reception Impossible Flag (AR 0413) is turned ON when newly received data cannot be input. New data cannot be input if the previously received data has not yet been input by PIN(64), or an error occurred during the previous reception.

RS-232C Communications Error Flag

SR 25208 functions as the Communications Error Flag for both the RS-232C Interface and the CPU-mounting Host Link Unit. It is turned ON when an error (a parity, overrun, or framing error) occurs during reception. This flag can be can be turned OFF by toggling the RS-232C Restart Bit (SR 25209) or executing PIN(64) with the number of bytes, B, equal to zero. SR 25209 also functions as the Restart Bit for the CPU-mounting Host Link Unit.

Flags

ER: D and D+(B÷2)–1 aren't in the same data area.

The RS-232C interface is busy. (I.e., the RS-232C Transmission Possible Flag (AR 0415) is OFF.)

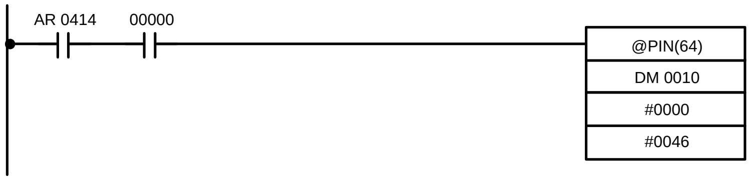
The RS-232C interface is not set to ASCII I/O mode.

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

Example

When AR 0414 and 00000 are ON, the following program inputs all of the characters received through the RS-232C interface (total number contained in AR 06) beginning at DM 0010.

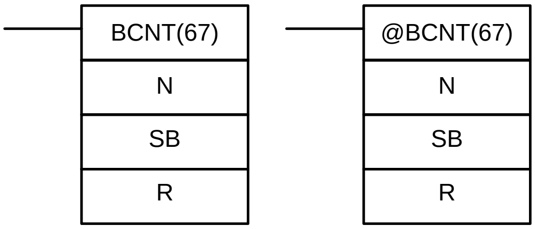
In this case, the instruction is executed when the Reception Completed Flag (AR 0414) is ON, but the received data can also be input when AR 0414 is OFF, i.e., when the end code has not been received and the reception buffer is not full.



Address	Instruction	Operands	
00000	LD	AR	0415
00001	AND		00000
00002	@PIN(64)		
		DM	0010
		#	0000
		AR	46

5-20-9 BIT COUNTER – BCNT(67)

Ladder Symbols



Operand Data Areas

N: Number of words (BCD)
IR, AR, DM, HR, TC, LR, #

SB: Source beginning word
IR, SR, AR, DM, HR, TC, LR

R: Destination word
IR, AR, DM, HR, TC, LR

Limitations

N cannot be 0.

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 R.

Flags

ER: N is not BCD, or N is 0; SB and SB+(N–1) are not in the same area.

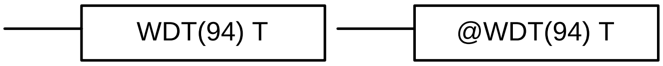
The resulting count value exceeds 9999.

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.

5-20-10 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.

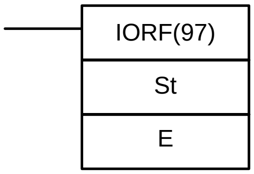
Timers might not function properly when the cycle time exceeds 100 ms. When using WDT(94), the same timer should be repeated in the program at intervals that are less than 100 ms apart.

Flags

There are no flags affected by this instruction.

5-20-11 I/O REFRESH – IORF(97)

Ladder Symbol



Operand Data Areas

St: Starting word

IR (I/O word only)

E: End word

IR (I/O word only)

Limitations

This instruction is only effective for I/O words. IR 000 to IR 039 are allocated to I/O Units.

St must be less than or equal to E.

Description

When the execution condition is OFF, IORF(97) is not executed. When the execution condition is ON, all words between St and E will be refreshed. This will be in addition to the normal I/O refresh performed during the CPU's cycle.

Execution Time

The execution time for IORF(97), T_{IORF}, is computed as follows:

Inputs	T = 600 µs + (70 µs x 2 x number of words refreshed)
Outputs	T = 600 µs + (40 µs x 2 x number of words refreshed)

Flags

There are no flags affected by this instruction.

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.

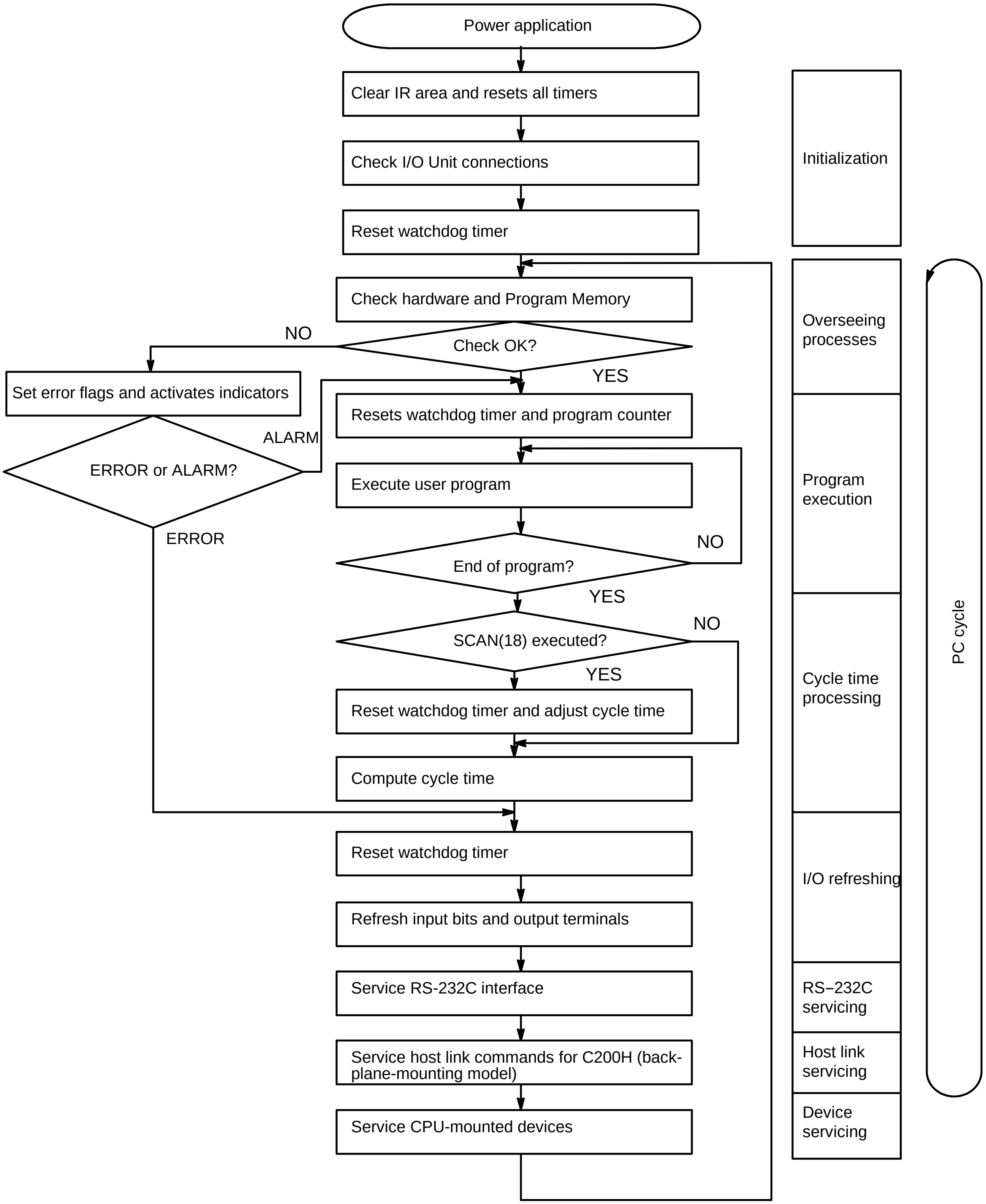
I/O response times in Link Systems are described in the individual System Manuals. These are listed at the end of*Section 1 Introduction*.

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 essential to effective programming and PC operations.

The major factors in determining program timing are the cycle time and the I/O response time. One cycle of CPU operation is called a cycle; the time required for one cycle is called the cycle time. The time required to produce a control output signal following reception of an input signal is called the I/O response time.

The overall flow of CPU operation is as shown in the following flowchart.


Power application
Clear IR area and resets all timers
Check I/O Unit connections
Reset watchdog timer
Check hardware and Program Memory
NO
Check OK?
YES
Set error flags and activates indicators
ALARM
ERROR or ALARM?
ERROR
Resets watchdog timer and program counter
Execute user program
NO
End of program?
YES
NO
SCAN(18) executed?
YES
Reset watchdog timer and adjust cycle time
Compute cycle time
Reset watchdog timer
Refresh input bits and output terminals
Service RS-232C interface
Service host link commands for C200H (back-plane-mounting model)
Service CPU-mounted devices
Initialization
Overseeing processes
Program execution
Cycle time processing
I/O refreshing
RS−232C servicing
Host link servicing
Device servicing
PC cycle

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, with each cycle forming one cycle. The cycle time is the time that is required for the CPU to complete one of these cycles. This cycle includes basically seven operations.

The following table shows the breakdown of the PC cycle, time requirements, and reasons for variations in the cycle time. The total cycle time will be the sum of all the following.

Process	Content	Time requirements
Overseeing	Resetting watchdog timer, I/O bus check, UM check, refreshing clock	2.9 ms
Program execution	Execution of user program	Total time for executing all instructions according to current execution conditions. Varies with instructions used and execution conditions. Refer to *6-3 Instruction Execution Times* for details.
Cycle time adjustments	Computation of cycle time and required adjustment for SCAN(18)	Almost instantaneous (less than 1 ms) if no adjustment is necessary. Time required is determined by operand of SCAN(18) if adjustment is necessary.
I/O refresh	Output terminal status updated according to output bit status and input bits status updated according to input terminal status.	Inputs: 0.07 ms per 8 pts. Outputs: 0.04 ms per 8 pts. (I/O Units with 12 outputs are treated as having 16 output points).
RS-232C servicing	Built-in RS-232C interface serviced.	If not set in System DM, 5% of the calculation cycle time will be used (but the minimum time is 1 ms); otherwise, the percentage set in System DM (between 0% and 99%) will be used. The minimum servicing time is 0.2 ms even if System DM is set to 0%.
Host link servicing	Commands from host computer connected to C200H Host Link Unit (backplane-mounting model) serviced.	8 ms max.
Peripheral Device servicing	Peripheral Devices connect to CPU serviced (e.g., Programming Devices)	If not set in System DM, 5% of the calculation cycle time will be used (but the minimum time is 1 ms); otherwise, the percentage set in System DM (between 0% and 99%) will be used. The minimum servicing time is 0.2 ms even if System DM is set to 0%.

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.
20 or greater	0.02-second clock pulse not accurately readable.
100 or greater	0.1-second clock pulse not accurately readable and Cycle Timer Error flag (25309) turns ON.
200 or greater	0.2-second clock pulse not accurately readable.
6,500 or greater	FALS code 9F generated regardless of watchdog timer setting and the system halts.

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 example, 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 0.94 µs. (Operating times are given in the table in *Section 6-3*.)

In this example, we'll compute the cycle time for a C20H CPU. It is assumed that the program contains 500 instructions requiring an average of 0.94 µs each to execute.

The equation for the cycle time from above is given below. The portions not required for this simple PC have been left out. It is assumed that there is no adjustment to the cycle time.

Cycle time = overseeing time
+ execution time
+ I/O refresh time
+ RS-232C servicing time
+ peripheral device servicing time

In this example, as shown in the table below, the total cycle time is between 3.6 and 5.6 ms depending on RS-232C and Peripheral Device connections.

Process	Calculation	RS-232C and Peripheral Device not connected	RS-232C and Peripheral Device processing times set to 5%*	RS-232C and Peripheral Device processing times set to 0%*
Overseeing time	Fixed	2.9 ms	2.9 ms	2.9 ms
Execution time	0.94 µs x 500 instructions	0.5 ms	0.5 ms	0.5 ms
I/O refresh time	0.07 ms x 2 + 0.04 ms x 1	0.2 ms	0.2 ms	0.2 ms
RS-232C servicing time	Zero or percentage of execution time	0 ms	1.0 ms	0.2 ms
Peripheral device servicing time	Zero or percentage of execution time	0 ms	1.0 ms	0.2 ms
Cycle time	**Total of above**	**3.6 ms**	**5.6 ms**	**4.0 ms**

*If the servicing time for RS-232C and peripheral devices is set to 0%, the servicing time will be 0.2 ms and the response time will be extremely slow.

6-3 Instruction Execution Times

This following table lists the execution times for all instructions that are available for the C20H/C28H/C40H/C60H. 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.

Table: Instruction Execution Times

Instruction	Conditions	ON execution time (µs)*	OFF execution time (µs)*
LD	---	0.75	1.5
LD NOT	---	0.75	1.5
AND	---	0.75	1.5
AND NOT	---	0.75	1.5
OR	---	0.75	1.5
OR NOT	---	0.75	1.5
AND LD	---	0.75	1.5
OR LD	---	0.75	1.5
OUT	---	1.13	2.25
OUT NOT	---	1.13	2.25
TIM	Constant for SV	2.25	R: 2.25 IL: 2.25 JMP: 2.25
	*DM for SV		R: 259 IL: 2.25 JMP: 2.25
CNT	Constant for SV	2.25	R: 2.25 IL: 2.25 JMP: 2.25
	*DM for SV		R: 255 IL: 2.25 JMP: 2.25
NOP(00)	---	0.75	---
END(01)	---	85	---
IL(02)	---	32	35
ILC(03)	---	59	35
JMP(04)	---	35	35
JME(05)	---	45	35
FAL(06)	FAL(06) 00 (reset)	357	2.25
	FAL(06) 01 to 99	247	2.25
FALS(07)	---	11.1 ms	2.25
STEP(08)	---	364	2.25
SNXT(09)	---	22	2.25
SFT(10)	With 1-word shift register	227	R: 191 IL: 30 JMP: 30
	With 250-word shift register	8.06 ms	R: 1.81 ms IL: 30 JMP: 30
KEEP(11)	---	1.13	---
CNTR(12)	Constant for SV	107	R: 85 IL: 49 JMP: 49
	*DM for SV	265	
DIFU(13)	---	105	Normal: 93 IL: 93 JMP: 84

Note: * The execution time is given in microseconds unless otherwise stated.

Instruction	Conditions	ON execution time (µs)*	OFF execution time (µs)*
DIFD(14)	---	104	Normal: 92 IL: 92 JMP: 84
TIMH(15)	Interrupt, Constant for SV	149	R: 199 IL: 199 JMP: 73
	Normal cycle, Constant for SV	169	
	Interrupt, *DM for SV	149	R: 291 IL: 291 JMP: 73
	Normal cycle, *DM for SV	169	
WSFT(16)	When shifting 1 word	260	3
	When shifting 1000 words using *DM	17.3 ms	
RWS(17)	When shifting 1 word	558	3.75
	When shifting 1000 words using *DM	57.4 ms	
SCAN(18)	---	Cycle time set in instruction – actual cycle time	3.75
CMP(20)	When comparing a constant to a word	162	3
	When comparing two *DM	447	
MOV(21)	When transferring a constant to a word	113	3
	When transferring *DM to *DM	321	
MVN(22)	When transferring a constant to a word	115	3
	When transferring *DM to *DM	392	
BIN(23)	When converting a word to a word	197	3
	When converting *DM to *DM	465	
BCD(24)	When converting a word to a word	198	3
	When converting *DM to *DM	451	
ASL(25)	When shifting a word	62	2.25
	When shifting *DM	190	
ASR(26)	When shifting a word	62	2.25
	When shifting *DM	190	
ROL(27)	When rotating a word	66	2.25
	When rotating *DM	194	
ROR(28)	When rotating a word	66	2.25
	When rotating *DM	194	
COM(29)	When inverting a word	379	2.25
	When inverting *DM	506	
ADD(30)	Constant + word A word	166	3.75
	*DM + *DM A *DM	593	
SUB(31)	Constant + word A word	192	3.75
	*DM – *DM A *DM	600	
MUL(32)	Constant x word A word	634	3.75
	*DM x *DM A word	1045	
DIV(33)	Word ÷ constant A word	737	3.75
	*DM ÷ *DM A *DM	1156	
ANDW(34)	Constant < word A word	162	3.75
	*DM < *DM A *DM	557	
ORW(35)	Constant > word A word	162	3.75
	*DM > *DM A *DM	560	
XORW(36)	Constant XORW word A word	162	3.75
	*DM XORW *DM A *DM	560	

Note: * The execution time is given in microseconds unless otherwise stated.

Instruction	Conditions	ON execution time (µs)*	OFF execution time (µs)*
XNRW(37)	Constant XNRW word A word	163	3.75
	*DM XNRW *DM A *DM	561	
INC(38)	When incrementing a word	79	2.25
	When incrementing *DM	207	
DEC(39)	When decrementing a word	72	2.25
	When decrementing *DM	260	
STC(40)	---	21	1.5
CLC(41)	---	21	1.5
MSG(46)	---	88	2.25
LMSG(47)	When outputting character string to Programming Console from word	334	3.75
	When outputting character string to Programming Console set by *DM	414	
	When outputting character string to RS-232C from word	751	
	When outputting character string to RS-232C set by *DM	1679	
SYS(49)	When using command code 01.	1998	3.75
ADB(50)	Constant + word A word	208	3.75
	*DM + *DM A *DM	604	
SBB(51)	Constant – word A word	208	3.75
	*DM – *DM A *DM	604	
MLB(52)	Constant x word A word	283	3.75
	*DM x *DM A *DM	658	
DVB(53)	Word ÷ constant A word	516	3.75
	*DM ÷ *DM A *DM	927	
RDM(60)	When comparing 1 range with words	719	719
	When comparing max. ranges with *DM	18.0 ms	18.0 ms
HDM(61)	When comparing 1 range with words	1079	3.75
	When comparing max. ranges with *DM	18.5 ms	
KEY(62)	When using words	464	3.75
	When using *DM	489	
POUT(63)	When outputting 0 bytes	464	3.75
	When outputting 200 bytes of *DM	4.78 ms	
PIN(64)	When inputting 0 bytes	646	3.75
	When inputting 200 bytes of *DM	4.95 ms	
HTS(65)	Word conversion	468	3.75
	*DM conversion (converting max. time)	60.6 ms	
STH(66)	Word conversion	572	3.75
	*DM conversion (converting max. seconds)	175.3 ms	
BCNT(67)	When counting 1 word	531	3.75
	When counting 1000 words using *DM	253.6 ms	
BCMP(68)	Comparing constant to word-designated table	823	3.75
	Comparing *DM A *DM-designated table	17 ms	
HEX(69)	Converting word to word	433	3.75
	Converting *DM to *DM	1.20 ms	
XFER(70)	When transferring 1 word	433	3.75
	When transferring 1000 words using *DM	47.1 ms	
BSET(71)	When setting a constant to 1 word	280	3.75

Note: * The execution time is given in microseconds unless otherwise stated.

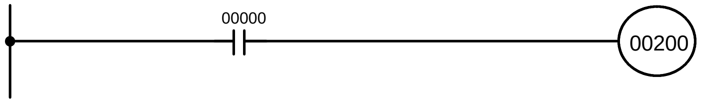
Instruction	Conditions	ON execution time (µs)*	OFF execution time (µs)*
	When setting *DM ms to 1,000 words using *DM	1.56 ms	
XCHG(73)	Between words	215	3
	Between *DM	408	
SLD(74)	Shifting 1 word	211	3
	Shifting 1,000 words using *DM	25.3 ms	
SRD(75)	Shifting 1 word	208	3
	Shifting 1,000 words using *DM	25.3 ms	
MLPX(76)	When decoding word to word	337	3.75
	When decoding *DM to *DM	708	
DMPX(77)	When encoding a word to a word	404	3.75
	When encoding *DM to *DM	758	
MOVB (82)	When transferring word to a word	172	3.75
	When transferring *DM to *DM	557	
MOVD(83)	When transferring word to a word	210	3.75
	When transferring *DM to *DM	459	
SFTR(84)	When shifting 1 word	475	3.75
	When shifting 1000 DM words using *DM	18.7 ms	
ASC(86)	Word A word	385	3.75
	*DM A *DM	746	
SBS(91)	---	320	2.25
SBN(92)	---	---	---
RET(93)	---	207	1.5
WDT(94)	---	27	2.25
IORF(97)	1-word refresh	675	3
	30-word refresh	4 ms	

Note: * The execution time is given in microseconds unless otherwise stated.

6-4 I/O Response Time

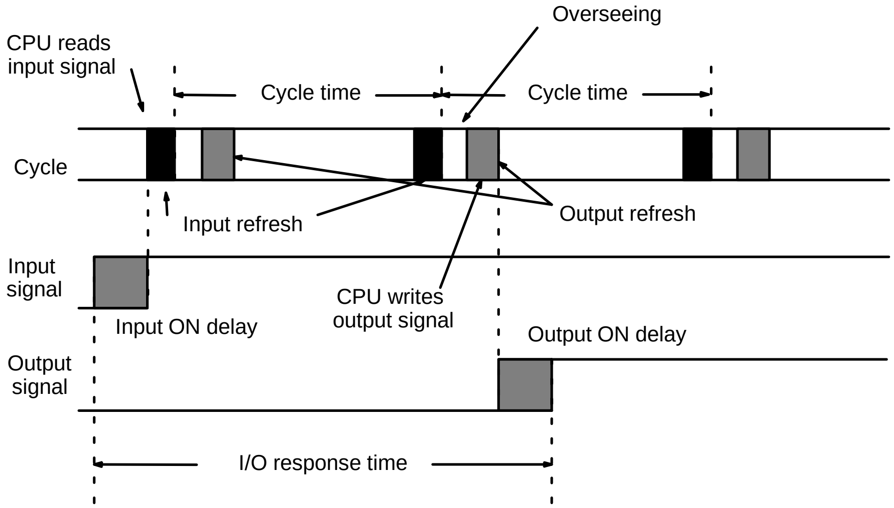
The I/O response time is the time it takes for the PC to output a control signal after it has received an input signal. The time it takes to respond depends on the cycle time and when the CPU receives the input signal relative to the input refresh period.

The minimum and maximum I/O response time calculations described below are for where 00000 is the input bit that receives the signal and 00200 is the output bit corresponding to the desired output point.



Minimum I/O Response Time

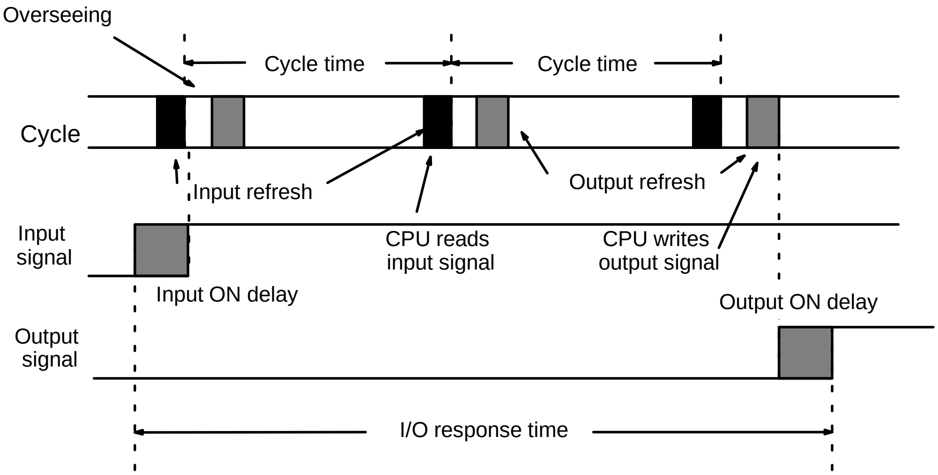
The PC responds most quickly when it receives an input signal just prior to the input refresh period in the cycle. Once the input bit corresponding to the signal has been turned ON, the program would have to be executed once to turn ON the output bit for the desired output signal and then the input refresh and overseeing operations would have to be repeated before the output refresh operation refreshes the output bit. The I/O response time in this case is thus found by adding the input ON-delay time, the cycle time (including the I/O refresh times and the overseeing 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 + overseeing time + output ON delay

Maximum I/O Response Time

The PC takes longest to respond when it receives the input signal just after the input refresh phase of the cycle. In this case the CPU does not recognize the input signal until the end of the next cycle. The maximum response time is thus one cycle longer than the minimum I/O response time, except that the input 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) + overseeing time + output ON delay

Calculation Example

The data in the following table would produce the minimum and maximum cycle times shown calculated below.

Input ON-delay	1.5 ms
Cycle time	20 ms
Input refresh time	0.23 ms
Overseeing time	3.0 ms
Output ON-delay	15 ms

Minimum I/O response time = 1.5 + 20 + 0.23 + 3.0 +15 = 39.73 ms

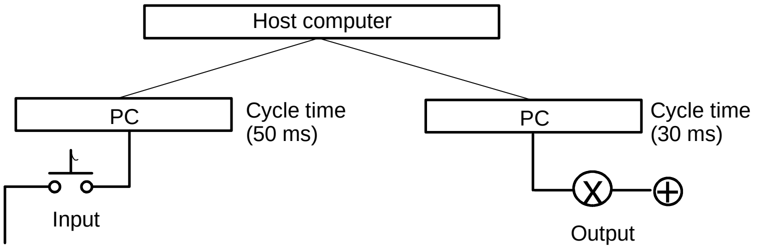
Maximum I/O response time = 1.5 + (20 x 2) + 3.0 +15 = 59.5 ms

6-5 Host Link Response Time

The processing that determines and the methods for calculating the minimum and maximum times required from an input on one PC in a Host Link System to an output on another PC in the same Host Link System are described below. The transfer between the PCs is handled through a host computer connected to both these PCs. Although more precise equations may be written if required, those used in the following calculations do not consider fractions of a cycle.

In considering response times, it is important to remember the sequence of processing that occurs during the PC cycle. The main factor that affects the response time is the timing of inputs and outputs, the length of the transmission, and the time required for host computer processing.

The following diagram illustrates the setup used in response time calculations.



The following equations are used to calculate the minimum and maximum response times for the C20H/C28H/C40H/C60H. The maximum response time is an approximation.

Minimum response time = Input ON delay + Command transmission time + (Cycle time of PC for Unit #0 x 2) + Response transmission time + Host computer processing time + Command transmission time + (Cycle time of PC for Unit #31 x 2) + 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

SECTION 7
Program Debugging and Execution

This section provides the procedures for debugging a program, and for monitoring and controlling the PC through a Programming Console.

If you are using a GPC, a FIT, or a computer running LSS, refer to the *Operation Manual* for procedures on these.

7-1 Displaying and Clearing Error Messages

After inputting a program and correcting it for syntax errors, it must be executed and all execution errors must be eliminated. Execution errors include an excessively long cycle, errors in settings for various Units in the PC, and inappropriate control actions, i.e., the program not doing what it is designed to do.

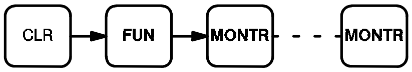
If desired, the program can first be executed isolated from the actual control system and wired to dummy inputs and outputs to check for certain types of errors before actual trial operation with the controlled system.

When an error occurs during program execution, it can be displayed for identification by pressed CLR, FUN, and then MONTR. If an error message is displayed, MONTR can be pressed to access any other error messages that are stored by the system in memory. If MONTR is pressed in PROGRAM mode, the error message will be cleared from memory. Be sure to write down the error message when required before pressing MONTR. OK will be displayed when the last message has been cleared. This procedure can also be used to clear messages produced by the program through MSG(46).

If a beeper sounds and the error cannot be cleared by pressing MONTR, the cause of the error still exists and must be eliminated before the error message can be cleared. If this happens, take the appropriate corrective action to eliminate the error. Refer to *Section 9 Troubleshooting* for all details on all error messages. The sequence in which error messages are displayed depends on the priority levels of the errors. The messages for fatal errors (i.e., those that stop PC operation) are displayed before non-fatal ones.

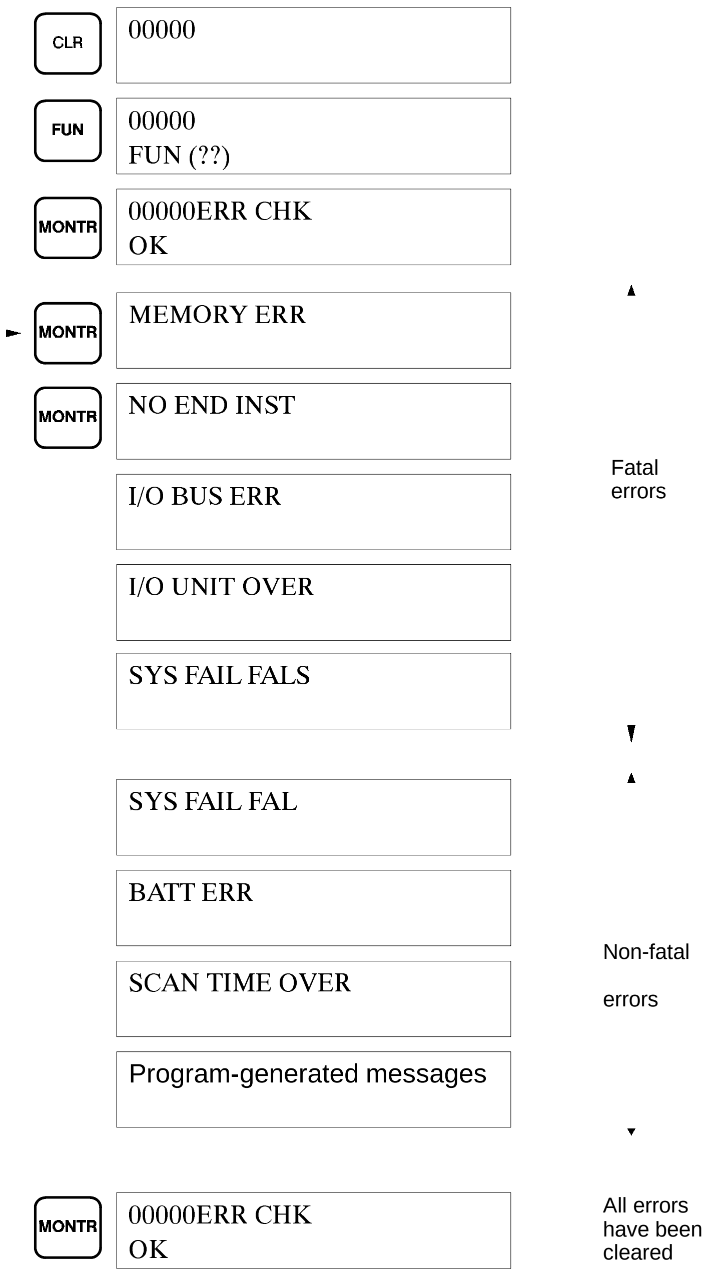
Although error messages can be displayed in any mode, they can be cleared only in PROGRAM mode. There is no way to restart the PC following a fatal error without first clearing the error message in PROGRAM mode.

Key Sequence



Example

The following displays show some of the messages that may appear. Refer to *Section 9 Troubleshooting* for an extensive list of error messages, their meanings, and the appropriate responses.



```
00000

00000
FUN (??)

00000ERR CHK
OK

MEMORY ERR

NO END INST

I/O BUS ERR

I/O UNIT OVER

SYS FAIL FALS

SYS FAIL FAL

BATT ERR

SCAN TIME OVER

Program-generated messages

00000ERR CHK
OK
```

7-2 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 preforming one of the monitor operations. Data modification is possible in either MONITOR or PROGRAM mode, but cannot be performed in RUN mode.

7-2-1 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/Word 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.

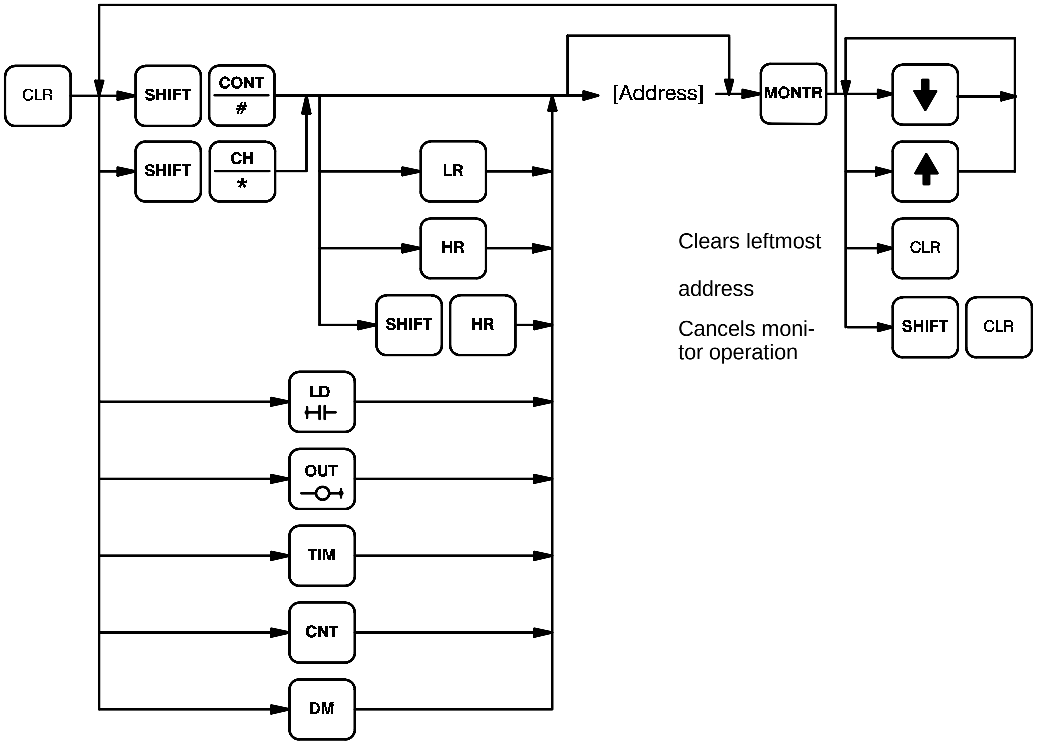
Up to six memory addresses, either bits, words, or a combination of both, can be monitored at once, although only three of these are displayed at any one time. To monitor more than one address, return to the start of the procedure and continue designating addresses. Monitoring of all designated addresses 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



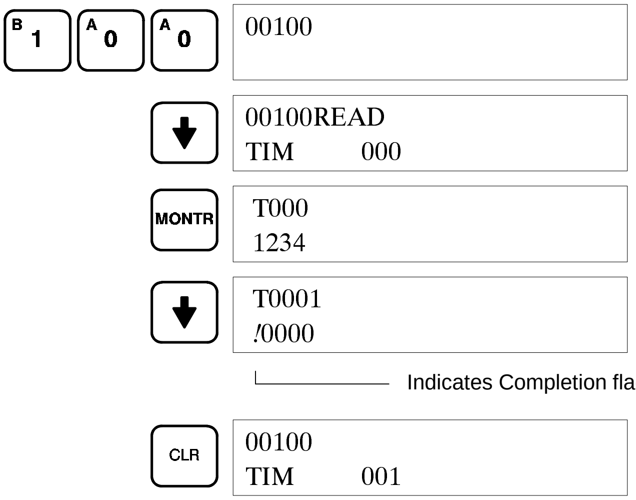
Clears leftmost address

Cancels monitor operation

Examples

The following examples show various applications of this monitor operation.

Program Read then Monitor



```
00100
```

```
00100READ
TIM         000
```

```
T000
1234
```

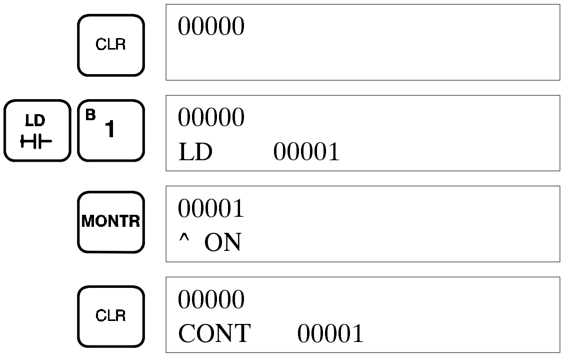
```
T0001
/0000
```

Indicates Completion flag is ON

```
00100
TIM         001
```

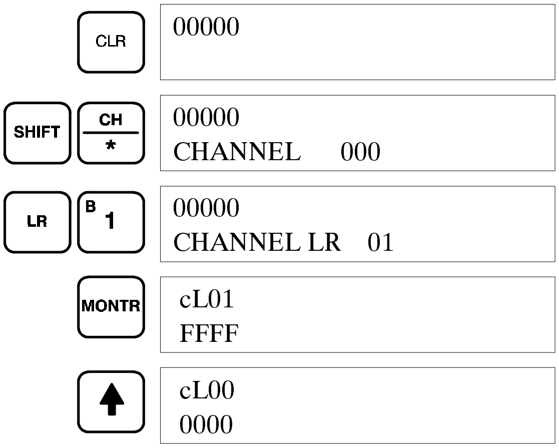
Monitor operation is cancelled

Bit Monitor



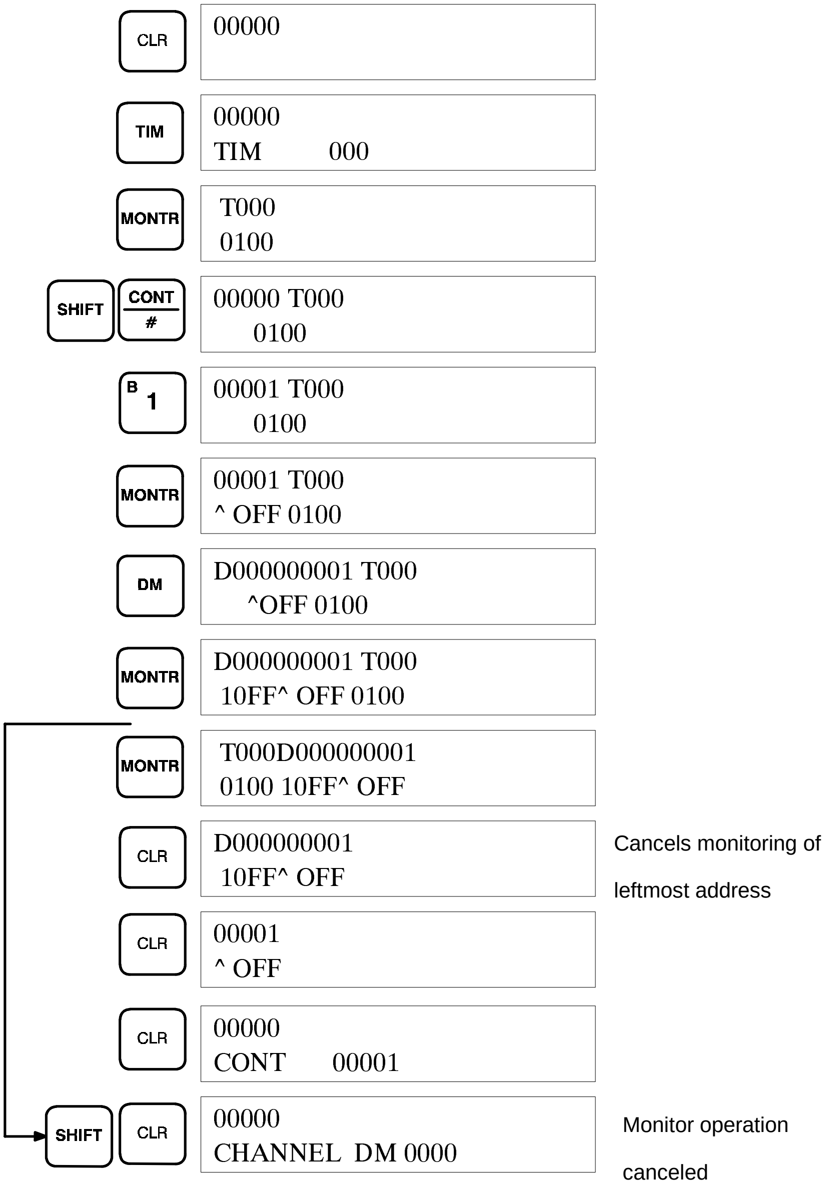
```
00000

00000
LD        00001

00001
^  ON

00000
CONT      00001
```

Word Monitor



```
00000

00000
CHANNEL     000

00000
CHANNEL LR   01

cL01
FFFF

cL00
0000
```

Multiple Address Monitoring



```
CLR            00000

TIM            00000
               TIM        000

MONTR           T000
                0100

SHIFT CONT/#   00000 T000
                    0100

B 1            00001 T000
                    0100

MONTR          00001 T000
               ^ OFF 0100

DM             D000000001 T000
                   ^OFF 0100

MONTR          D000000001 T000
                10FF^ OFF 0100

MONTR           T000D000000001
                0100 10FF^ OFF

CLR            D000000001
                10FF^ OFF

CLR            00001
               ^ OFF

CLR            00000
               CONT      00001

SHIFT CLR      00000
               CHANNEL  DM 0000
```

Cancels monitoring of leftmost address

Monitor operation canceled

7-2-2 Forced Set/Reset

When the Bit/Word 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.

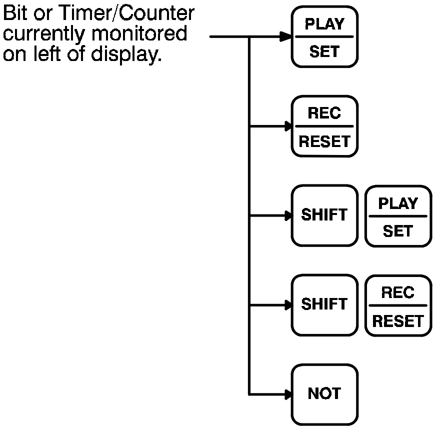
If you press PLAY/SET or REC/RESET alone (i.e., without SHIFT), then Force Set/Reset will continue only as long as the key is held down. If you press either of these with SHIFT, however, then the operation will continue until cancelled with NOT.

Without using NOT, the operation may be cancelled in any of the following four ways:

- With the Restore Status operation
- With a PC mode change
- When operation halts due to an error
- When operation halts due to a power failure

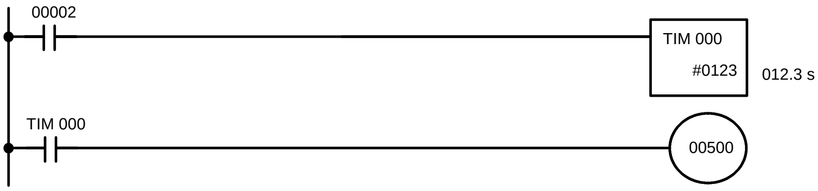
This operation can be used in MONITOR mode to check wiring of outputs from the PC prior to actual program execution. This operation cannot be used in RUN mode.

Key Sequence


Bit or Timer/Counter currently monitored on left of display.
PLAY SET
REC RESET
SHIFT
PLAY SET
SHIFT
REC RESET
NOT

Example

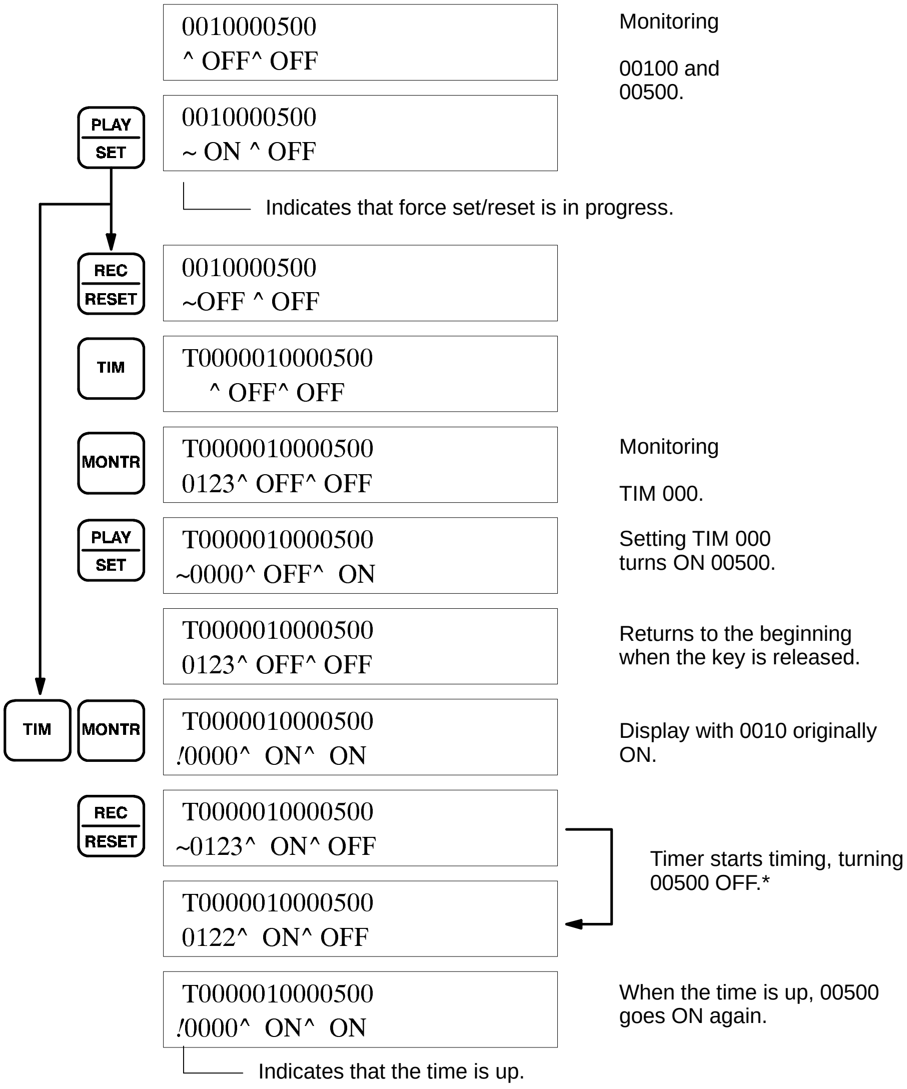
The following example shows how either bits or timers can be controlled with the Force Set/Reset operation. The displays shown below are for the following program section.



Address	Instruction	Operands	
00200	LD		00002
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).

(This example is in MONITOR mode.)



```
0010000500
^ OFF^ OFF
```

Monitoring

00100 and 00500.

```
0010000500
~ ON ^ OFF
```

Indicates that force set/reset is in progress.

```
0010000500
~OFF ^ OFF
```

```
T0000010000500
   ^ OFF^ OFF
```

```
T0000010000500
0123^ OFF^ OFF
```

Monitoring

TIM 000.

```
T0000010000500
~0000^ OFF^  ON
```

Setting TIM 000 turns ON 00500.

```
T0000010000500
0123^ OFF^ OFF
```

Returns to the beginning when the key is released.

```
T0000010000500
!0000^  ON^  ON
```

Display with 0010 originally ON.

```
T0000010000500
~0123^  ON^ OFF
```

```
T0000010000500
0122^  ON^ OFF
```

Timer starts timing, turning 00500 OFF.*

```
T0000010000500
!0000^  ON^  ON
```

When the time is up, 00500 goes ON again.

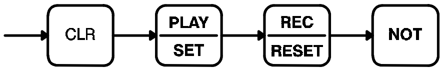
Indicates that the time is up.

*Timing not done in PROGRAM mode.

7-2-3 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.

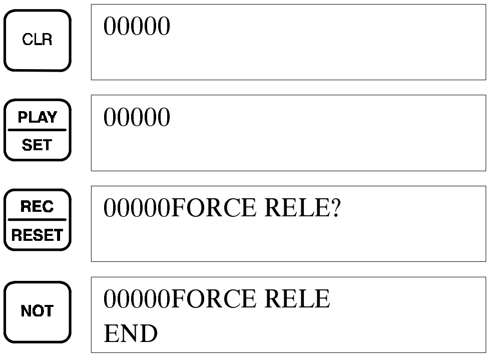
Key Sequence



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.



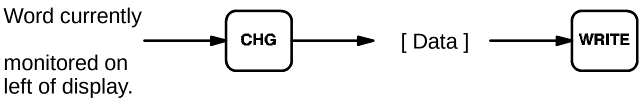
7-2-4 Hexadecimal/BCD Data Modification

When the Bit/Word 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 *7-2-11 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

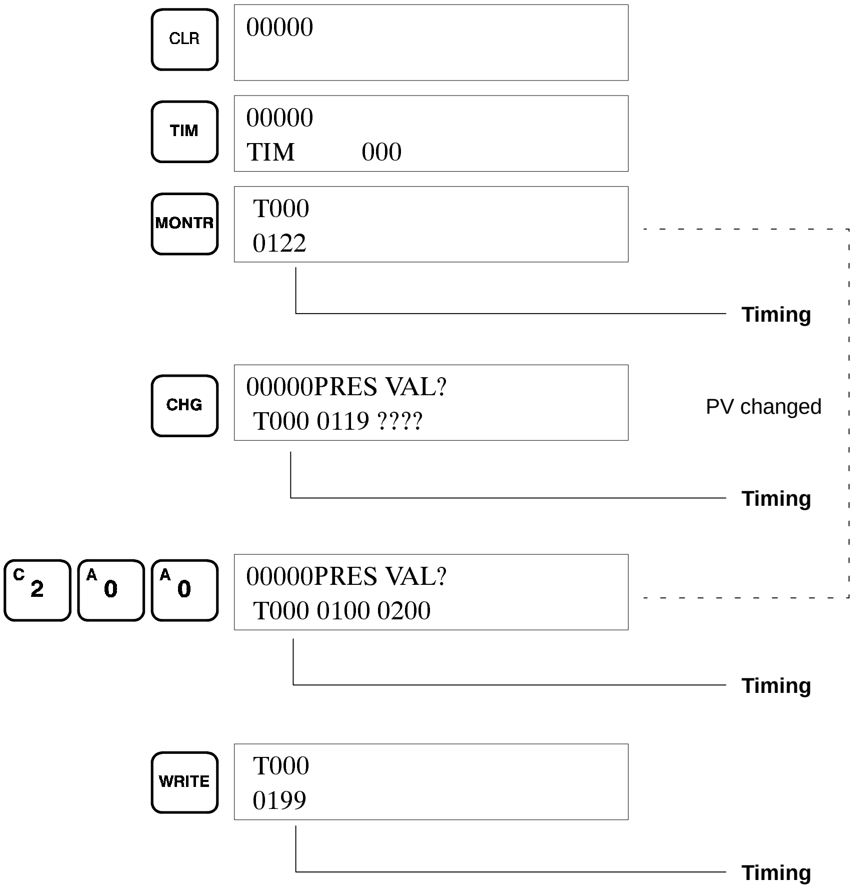
Key Sequence



Example

The following example shows the effects of changing the PV of a timer.

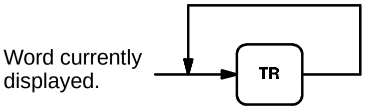
This example is in MONITOR mode



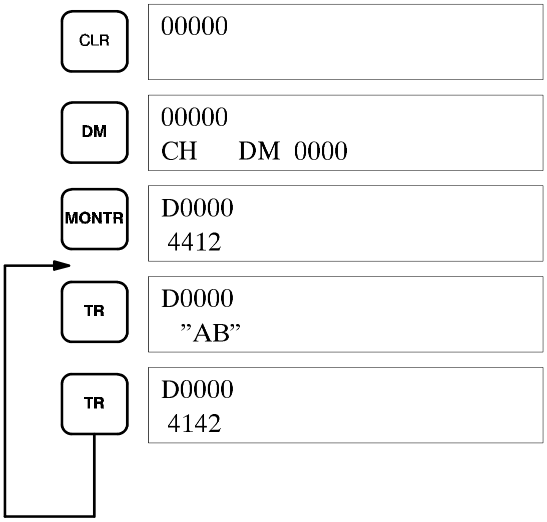
7-2-5 Hex/ASCII Display Change

This operation converts DM data displays back and forth between 4-digit hexadecimal data and ASCII.

Key Sequence



Example



```
00000

00000
CH      DM  0000

D0000
 4412

D0000
   ”AB”

D0000
 4142
```

7-2-6 Program Header Display

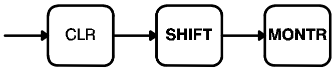
With this operation you can display the name of the program, along with the version number and the time it was last revised (given in year, month, day, hour, and minute).

When the SHIFT and MONITOR keys are pressed, the Programming Console displays the program name, version number, and so on, which have previously been stored in the DM System area. If the title/version enable (5A) in DM 1990 is OFF, then asterisks will be displayed.

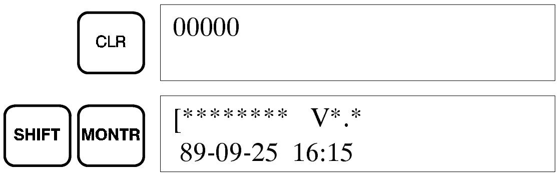
On models with a clock function, the revision time is generated automatically whenever there is an insertion, deletion, or addition to the program, or when memory is cleared or timer/counter SVs are set. For models without the clock function, it is necessary to set the revision time.

For more detail, refer to *3-6 DM (Data Memory) Area*.

Key Sequence



Example



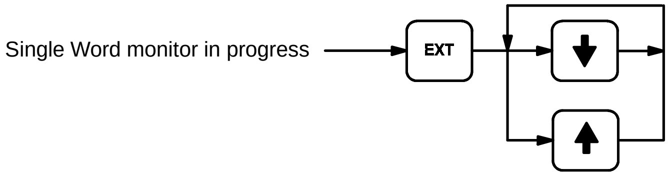
```
00000

[********   V*.*
 89-09-25  16:15
```

7-2-7 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



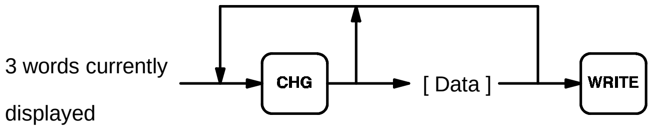
Example

CLR	00000
DM	00000 CHANNEL DM 0000
MONTR	D0000 89AB
EXT	D0002D0001D0000 0123 4567 89AB
↓	D0003D0002D0001 ABCD 0123 4567
↓	D0004D0003D0002 EF00 ABCD 0123
↓	D0005D0004D0003 1111 EF00 ABCD
↑	D0004D0003D0002 EF00 ABCD 0123
CLR	D0002 0123

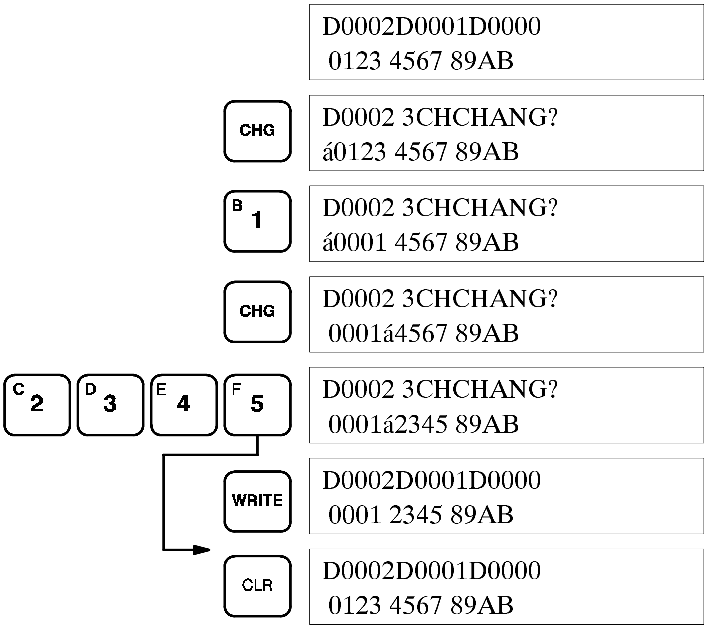
7-2-8 3-word Data Modification

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.

Key Sequence



Example



```
D0002D0001D0000
 0123 4567 89AB
```

3-word monitor in progress.

```
D0002 3CHCHANG?
á0123 4567 89AB
```

Stops in the middle of monitoring.

```
D0002 3CHCHANG?
á0001 4567 89AB
```

```
D0002 3CHCHANG?
 0001á4567 89AB
```

```
D0002 3CHCHANG?
 0001á2345 89AB
```

```
D0002D0001D0000
 0001 2345 89AB
```

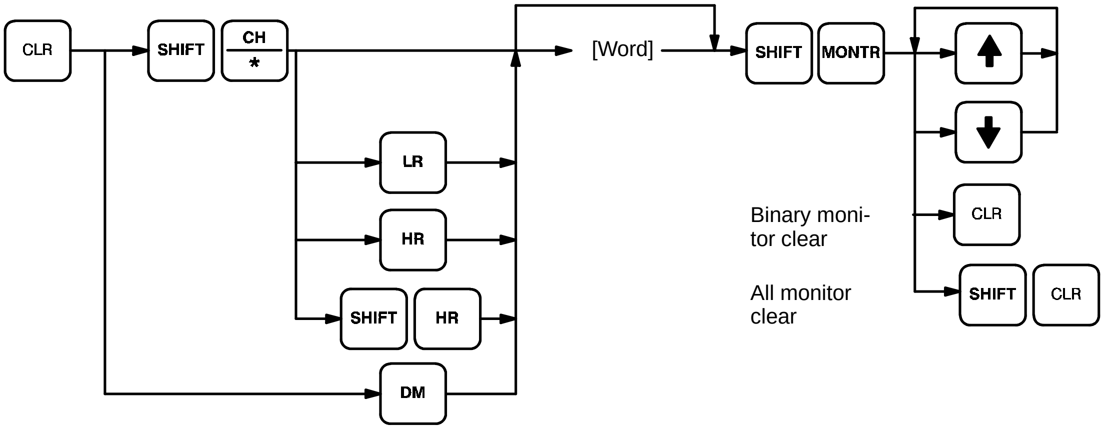
```
D0002D0001D0000
 0123 4567 89AB
```

Resumes previous monitoring.

7-2-9 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

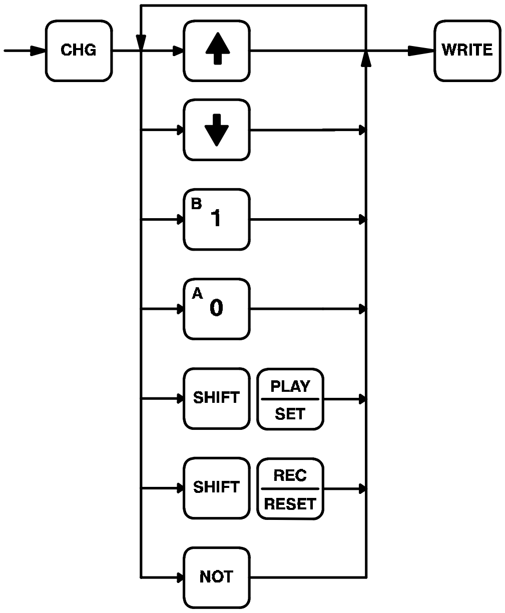
7-2-10 Binary Data Modification

This operation assigns a new 16-digit binary value to an IR, HR, AR, LR, or DM word.

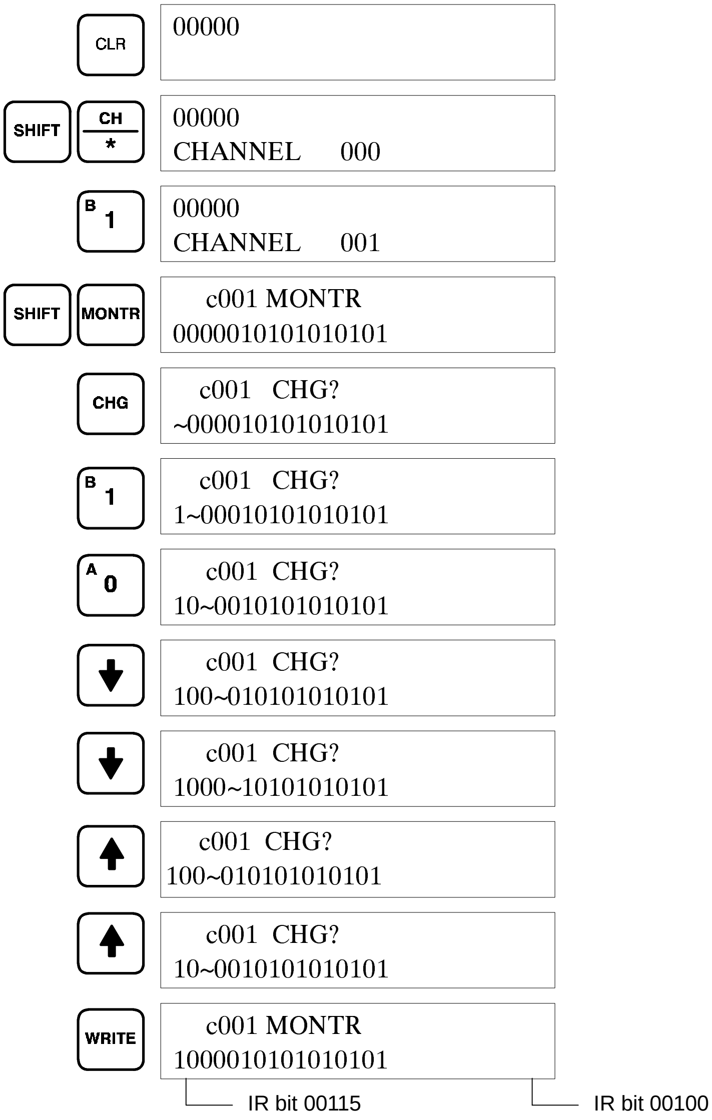
The blinking square, which can be shifted to the left with the up key and to the right with the down key, indicates the position of the bit that can be changed. After positioning to the desired bit, a 0 or a 1 can then be entered as the new bit value. Besides 0 or 1, a change can be executed with SHIFT and PLAY/SET or SHIFT and REC/RESET.

Forced Set and Forced Reset are indicated by S and R, and set at 1 and 0. They are cancelled by NOT. (See *Force Set/Reset Indicators* in *7-2-1 Bit/ Word Monitor*. 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



```
CLR             00000

SHIFT  CH/*     00000
                CHANNEL     000

B 1             00000
                CHANNEL     001

SHIFT  MONTR      c001 MONTR
                0000010101010101

CHG               c001   CHG?
                ~000010101010101

B 1               c001   CHG?
                1~00010101010101

A 0               c001  CHG?
                10~0010101010101

↓                 c001  CHG?
                100~010101010101

↓                 c001  CHG?
                1000~10101010101

↑                c001  CHG?
               100~010101010101

↑                 c001  CHG?
                10~0010101010101

WRITE             c001 MONTR
                1000010101010101
```

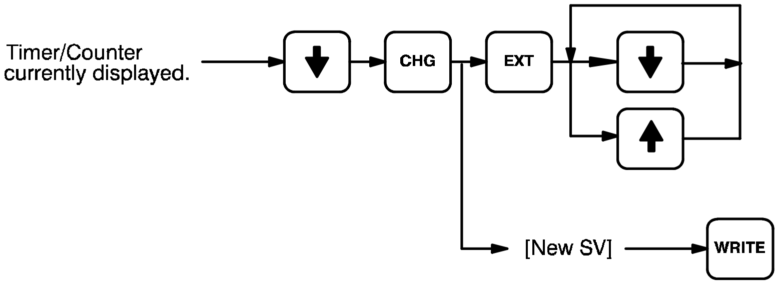
7-2-11 Changing Timer/Counter SV

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 versa.

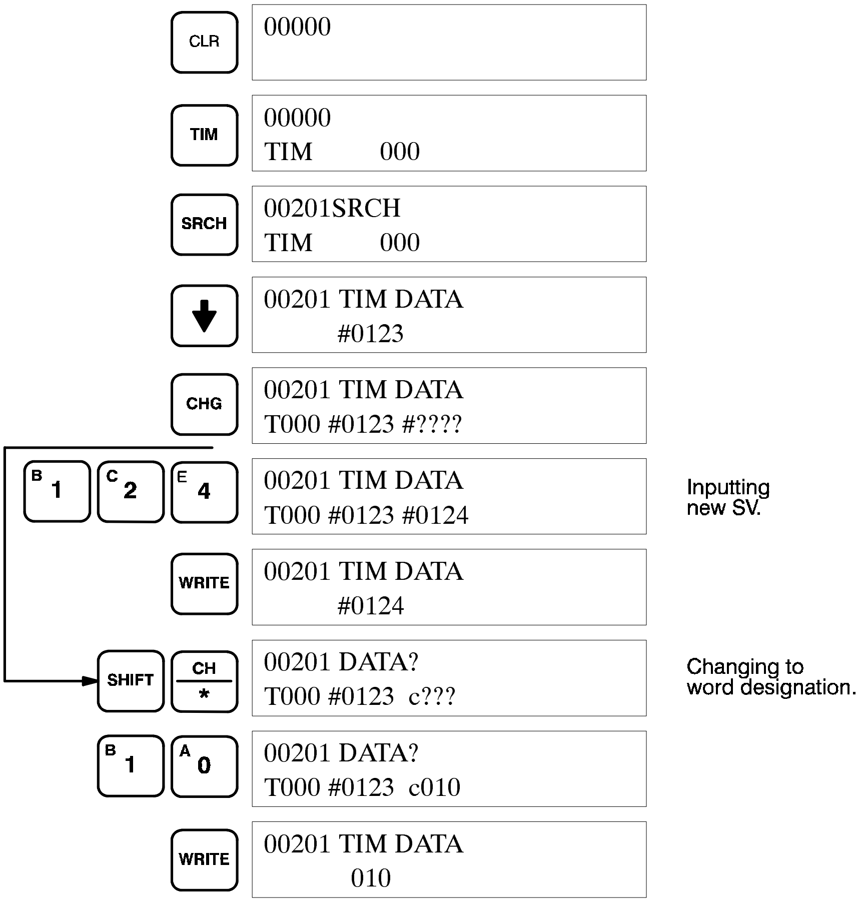
Key Sequence



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

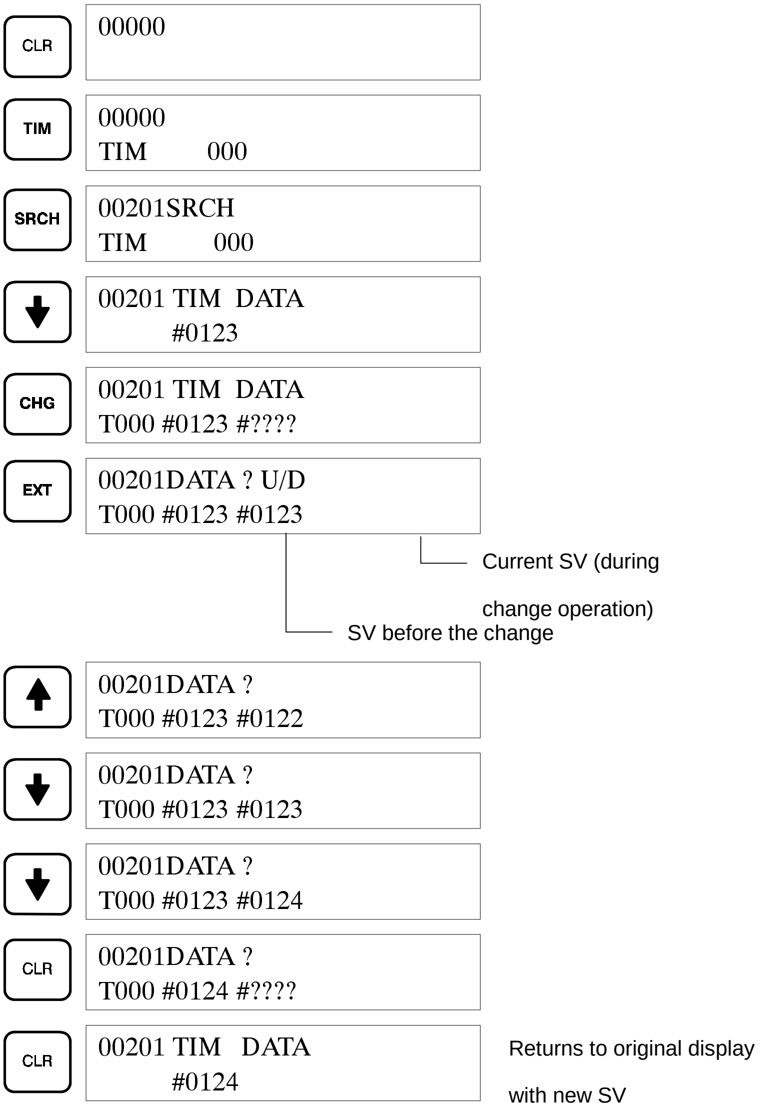


```
00000

00000
TIM        000

00201SRCH
TIM        000

00201 TIM DATA
       #0123

00201 TIM DATA
T000 #0123 #????

00201 TIM DATA
T000 #0123 #0124

00201 TIM DATA
       #0124

00201 DATA?
T000 #0123  c???

00201 DATA?
T000 #0123  c010

00201 TIM DATA
        010
```

Inputting new SV.

Changing to word designation.

Incrementing and Decrementing



```
00000

00000
TIM        000

00201SRCH
TIM         000

00201 TIM  DATA
          #0123

00201 TIM  DATA
T000 #0123 #????

00201DATA ? U/D
T000 #0123 #0123
```

Current SV (during change operation)

SV before the change

```
00201DATA ?
T000 #0123 #0122

00201DATA ?
T000 #0123 #0123

00201DATA ?
T000 #0123 #0124

00201DATA ?
T000 #0124 #????

00201 TIM   DATA
          #0124
```

Returns to original display with new SV

7-2-12 PROM Writer Operations

You can use the built-in RS-232C interface to download the Program Memory and DM 1000 to DM 1999 to a commercially available PROM writer and to upload the ROM contents of the PROM writer back to the PC.

For this operation you must set the RS-232C interface mode to download/ upload at both the PC and the PROM writer. (See *8-1 RS-232C Interface Modes*.) The PC must be in PROGRAM mode. Also, if a baud rate greater than 2,400 bps is used, reception may not be normal unless a transmission delay is used at the PROM writer.

For the transfer format, you can use either Intel’s HEX or Motorola’s S.

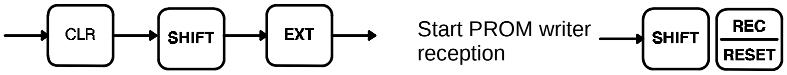
With the C20H/C28H/C40H, a cassette tape recorder cannot be connected directly to the cassette jacks on the Programming Console, although it is possible to connect a cassette recorder to the GPC to store, retrieve, and verify programs on tape.

The GPC not only allows programs to be stored and retrieved from tape, but also allows programs written for the C120 and C500 PCs to be converted to the format used with the C20H/C28H/C40H.

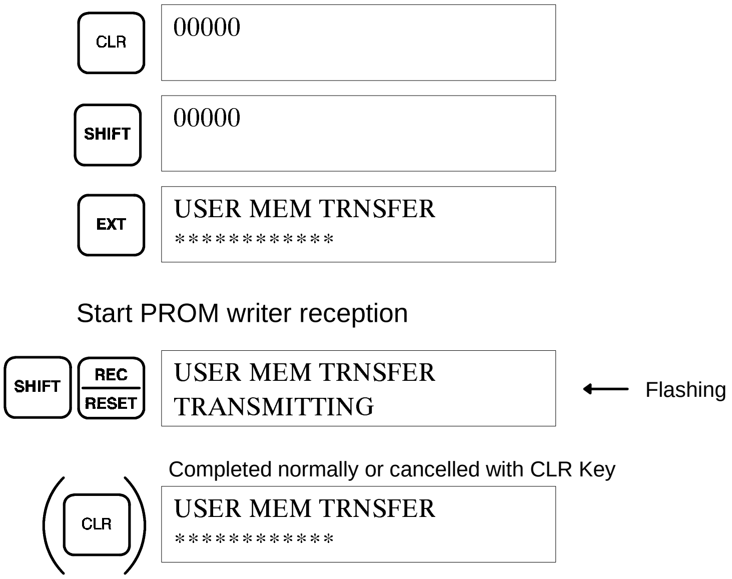
PC to PROM Writer

You can use the following key sequence to transfer Program Memory from the PC to the PROM writer. Where ********* appears below, the transfer format will be indicated as either Intel's HEX or Motorola's S. When writing to 27256 or 27128 chips, the first 8K bytes (0000 to 1FFF) are used.

Key Sequence



Example



```
CLR
00000

SHIFT
00000

EXT
USER MEM TRNSFER
************
```

Start PROM writer reception

```
SHIFT REC/RESET
USER MEM TRNSFER
TRANSMITTING
```

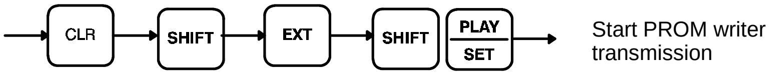
Completed normally or cancelled with CLR Key

```
(CLR)
USER MEM TRNSFER
************
```

PROM Writer to PC

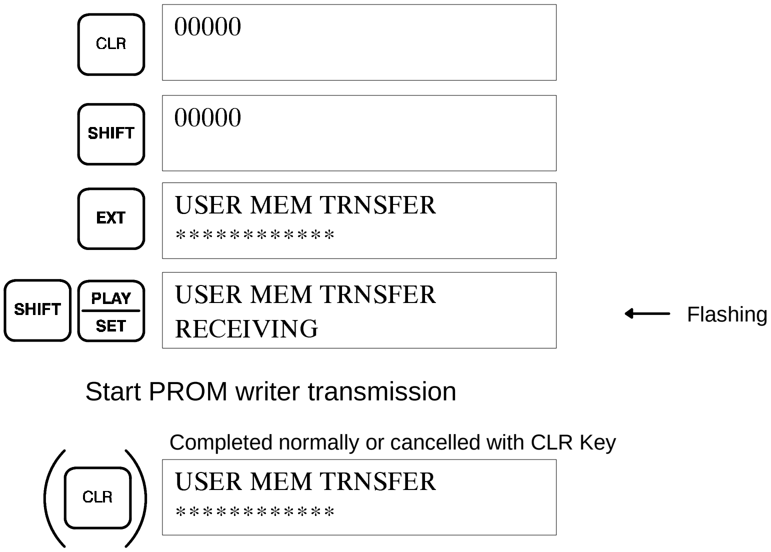
You can use the following key sequence to transfer Program Memory from the PROM writer to the PC. Where ********* appears below, the transfer format will be indicated as either Intel's HEX or Motorola's S-Record.

Key Sequence



Start PROM writer transmission

Example

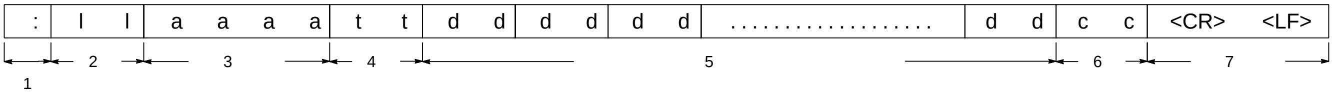


Start PROM writer transmission

Completed normally or cancelled with CLR Key

Intel's HEX Format

Intel's HEX format is completely configured in ASCII characters. Except for the start mark and the load address, it is expressed as two-digit hexadecimal bytes of data. The format is as shown below.



:	l	l	a	a	a	a	t	t	d	d	d	d	d	d	....	d	d	c	c	<CR>	<LF>
1	2		3				4		5									6		7	

1, 2, 3...

Start mark (:)
Indicates the start of a record.

Code number (two digits, hexadecimal)
Indicates the number of bytes in the record, from 01 to FF (1 to 255 bytes). If the code number is 00, it indicates the final record.

Load address (four digits, hexadecimal)
Indicates the address where the data is stored.

Record type (two digits, hexadecimal)
00 and 81 are recognized as data records, and the data beginning with the load address is loaded to the buffer RAM. 01 is recognized as the final record, and other types are ignored.

Data (two digits, hexadecimal)
This is the data which is stored in the buffer RAM. Addresses increase one by one.

Checksum
This is the two's complement of the total value of (2) through (5) after they are added together in hexadecimal.The rightmost eight bits are valid.

Carriage return and line feed

Motorola's S-Record

Motorola's S-Record are completely configured in ASCII characters. Except for the start mark, the record type and the load address, it is expressed as two-digit hexadecimal bytes of data. The format is as shown below.

S	t	l	l	a	a	a	a	d	d	d	d	d	d	....	d	d	c	c	<CR>	<LF>
1	2	3		4				5									6		7	

1, 2, 3...

1. Start mark (S)
Indicates the start of a record.

2. Record type (0 to 9, excluding 4 and 6)
0: Ignored. Record type at output time is output as 1.
1: Ordinary data record, with 2-byte (4-character) load address
2: Ordinary data record, with 3-byte (6-character) load address
3: Ordinary data record, with 4-byte (8-character) load address
5: Ignored. Record type at output time is output as 1.
7: Indicates final type 3 record.
8: Indicates final type 2 record.
9: Indicates final type 1 record.

3. Code number (two digits, hexadecimal)
Indicates one half the total number of characters from (4) load address to (6) checksum.

4. Load address (4, 6, or 8 digits, hexadecimal)
Indicates the address where the data is stored. When the load address is 6 or 8, the rightmost four digits are valid.

5. Data (two digits, hexadecimal)
This is the data which is stored in the buffer RAM.

6. Checksum
This is the one's complement of the total value of (3) through (5) after they are added together in hexadecimal.The rightmost eight bits are valid.

7. Carriage return and line feed

Transmission Error

The following table gives the probable causes and the procedures for correcting errors which may occur during transmission between the PC and the PROM writer.

Programming Console Message	Probable Cause	Correction
USER MEM TRNSFER RECEIVE ERR	Communication mode setting of RS-232C interface does not match that of PROM writer. RS-232C cable connection is faulty. PC reception is late.	Verify that the settings of the PC and the PROM writer match the specifications. Check the Communications Counter (AR 05) to see if the lines are transmitting properly. If there is a transmission delay at the PROM writer, lower the baud rate or insert a delay at the PROM writer.

SECTION 8
RS-232C Interface

The C20H/C28H/C40H/C60H have a built-in RS-232C interface which allows them to connect directly to a Host Link System, a Display Terminal, commercially available printers, and other devices which employ an RS-232C interface.

This section describes modes, settings, and procedures used with the RS-232C interface. For information on hardware connections, configuration, and installation, refer to the *Mini H-type PCs: C20H, C28H, C40H, C60H Installation Guide.* For information on errors which occur while using this interface in a Host Link System, refer to *9-6 Host Link Error Processing*. For information on Host Link Systems including other C-series PCs, refer to the *Host Link System Manuals.*

8-1 RS-232C Interface Modes

You can use the RS-232C interface in any of three modes:

1, 2, 3...

1. **Host Link Mode:** Select this mode to connect to a Host Link System or a FIT. For details, refer to *8-4 Host Link Communications Protocol* and *8-5 Host Link Commands and Responses.*
2. **Download/Upload Mode:** Select this mode to download the Program Memory and DM 1000 to DM 1999 from the PC to a commercial PROM writer and upload the ROM contents of the PROM writer back to the PC. For the transfer format, you can use either Intel's HEX or Motorola's [S]. For details, refer to *7-2-12 PROM Writer Operations*.
3. **ASCII I/O Mode:** ASCII I/O mode can be used to communicate from the user program with external devices that are equipped with an RS-232C interface. The following instructions can be used for 2-way half-duplex communications in ASCII I/O mode: LONG MESSAGE – LMSG(47), RS-232C PORT OUTPUT – POUT(63), and RS-232C PORT INPUT – PIN(64). For hardware configurations, refer to the *C20H, C28H, C40H, C60H Installation Guide*.

The interface mode is set in System DM. Refer to DM 0920/DM 1920 in *3-6 DM (Data Memory) Area*.

8-2 DM and AR Area Settings

Certain settings in the DM and AR areas are relevant to the RS-232C interface. For information on these settings, refer to *3-5-3 Built-in Host Link Communications Error Code*, *3-5-4 RS-232C Interface Communications Counters*, and *3-8-2 System DM.* With respect to these settings, the following points should be noted:

DM Area

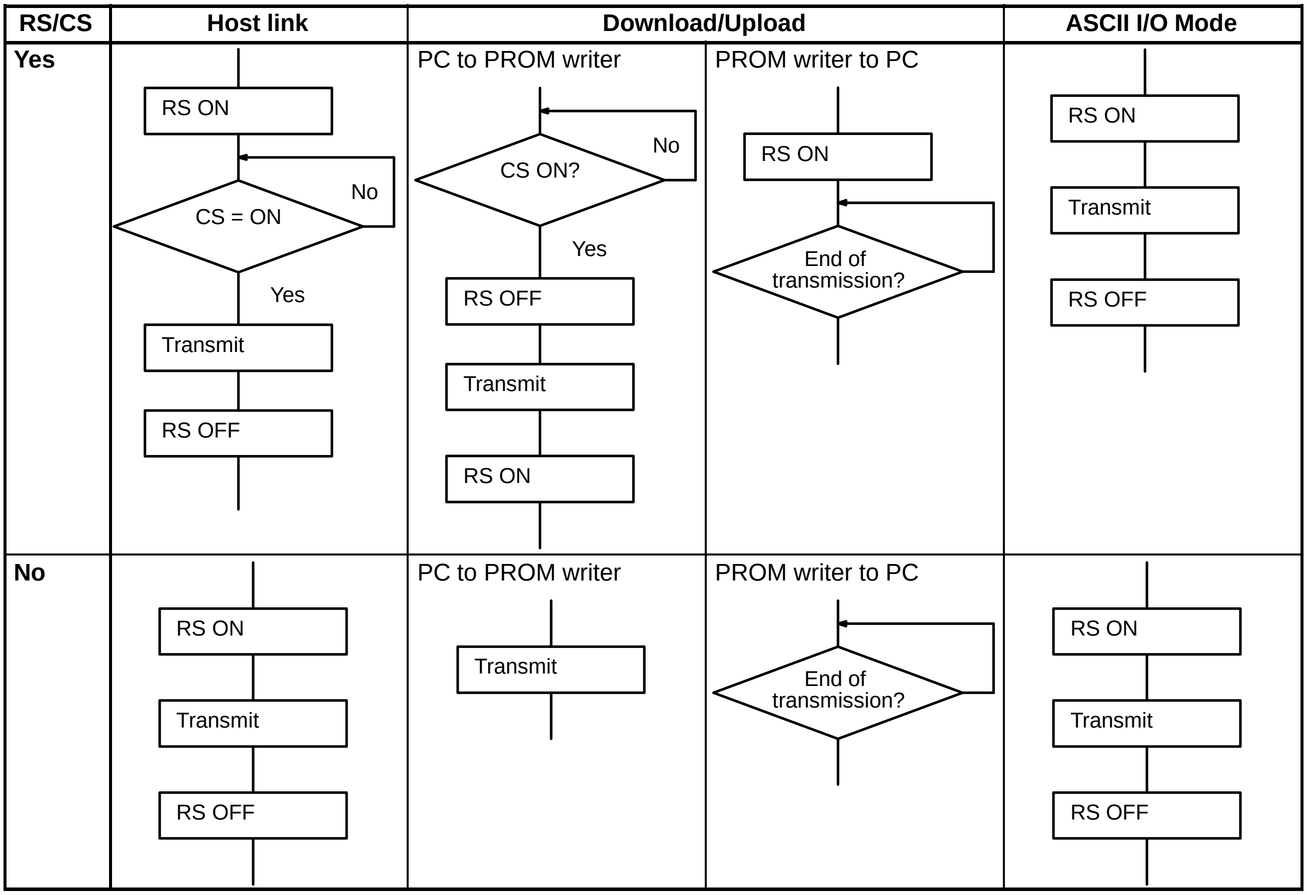
The interface mode is set in DM 0920/DM1920, bits 08 to 15. Any setting other than 00, 01, or 02 is regarded as Host Link Mode.

With a setting in DM 0920/DM 1920, bits 00 to 07, you designate the communications format as standard or custom. If you designate it as standard, then any baud rate or format settings in DM 0921/DM 1921 will be invalid. When connecting to a FIT at a baud rate of 9,600 bps, it is most expedient to use the standard format.

If you set the communications format to custom, then you can select a format in DM 0921/DM 1921. When using the Host Link Mode, do not select a no-parity format. Doing so may adversely affect the quality of the communications data.

You can set a transmission delay in DM 0922/DM 1922, bits 00 to 07. This can be useful in cases where the receiving party has an overrun, but it will slow down the transmission time. Always set these bits to 00 when connecting to a Programming Device (e.g., FIT).

You can designate RS/CS control with a setting in DM 0922/DM1922, bits 08 to 15. Set this to “without RS/CS” when using RS-422 or in other cases where RS/CS control is unnecessary. Do not use RS/CS control in Host link or ASCII I/O mode. Communications will proceed as shown below with and without RS/CS control.



If bits 08 to 15 of DM 0925 are set to 01, the code set in bits 00 to 07 of DM 0925 will be used as the starting code for POUT(63) and PIN(64) transmissions. If bits 08 to 15 are set to 00, no starting code will be used. If bits 08 to 15 of DM 0926 are set to 01, the code set in bits 00 to 07 of DM 0926 will be used as the end code for POUT(63) and PIN(64) transmissions. If bits 08 to 15 are set to 00, no end code will be used. Refer to *Section 5 Instruction Set* for details.

AR Area

The following parts of the AR area are used with the RS-232C interface.

Word(s)	Bit(s)	Function
04	00 to 07	RS-232C Communications Error Code
	13	RS-232C Reception Impossible Flag
	14	RS-232C Reception Completed Flag
	15	RS-232C Transmission Possible Flag
05	00 to 07	RS-232C Reception Counter
	08 to 15	RS-232C Transmission Counter
06	00 to 15	RS-232C Bytes Received Area
08	00 to 15	RS-232C Bytes Input Area

RS-232C Communications Error Code

When an error has occurred in RS-232C communications, the RS-232C Interface Communications Error Flag is turned ON, and a code that indicates the type of error is output to AR 0400 to AR 0407. These codes are in hexadecimal and are as follows:

00:	Parity error
01:	Framing error
02:	Overrun error
03:	FCS error

These bits are refreshed each cycle while using the RS-232C interface.

Reception Impossible Flag

The RS-232C Reception Impossible Flag (AR 0413) is turned ON when newly received data cannot be input. There are two possible reasons that the new data cannot be input:

1, 2, 3...

1. The previously received data has not yet been input by PIN(64).
2. An error occurred during the previous reception.

This flag will be turned OFF when PIN(64) is executed.

Reception Completed Flag

The RS-232C Reception Completed Flag (AR 0414) is turned ON when an end code or 200 bytes of data have been received (not including the starting and end codes if used) at the RS-232C interface. Use this flag in ASCII I/O mode to confirm that data has been received before execution PIN(64) and is turned OFF when PIN(64) is executed.

Transmission Possible Flag

The RS-232C Transmission Possible Flag (AR 0415) is turned ON when data can be transmitted from the RS-232C interface (i.e., when data is not being transmitted). This bit is turned OFF whenever the transmission buffer is being used by LMSG(47), POUT(63), or PIN(64). Be sure to use check this Flag when executing these instructions.

RS-232C Reception Counter

When a transmission is received on the RS-232C interface, the number of characters is counted in hexadecimal and then output to AR 0500 to AR 0507 (RS-232C Reception Counter) to assist the user in debugging RS-232C interface communications. This counter is used for all RS-232C interface operating modes, as well as for error characters. The counter can be reset by turning SR 25209 ON and then OFF.

RS-232C Transmission Counter

AR 0508 to AR 0515 (RS-232C Transmission Counter) operates the same as AR 0500 to AR 0507, except that it operates for transmisions sent from the PC through the RS-232C interface.

These counters are refreshed every cycle.

RS-232C Bytes Received Area

When a transmission is received in ASCII I/O Mode at the RS-232C interface, the number of bytes of data is counted in BCD and then output to AR 06. The start and end codes are not counted.

Counting stops when the RS-232C Reception Completed Flag (AR 0414) is turned ON. The value stored in AR 06 will not be updated after this flag is turned on even if new data is received at the RS-232C interface. The content of AR 06 is reset to 0 when PIN(64) is executed.

AR 06 is refreshed every cycle while the RS-232C interface is being used in ASCII I/O mode.

RS-232C Bytes Input Area

While AR 06 contains the number of bytes received in ASCII I/O Mode at the RS-232C interface, AR 08 contains the number of bytes of data (BCD) actually input to the PC by PIN(64).

Normally, the bytes input will equal the bytes received, but in some cases the numbers will differ. The two most likely cases are described below.

1, 2, 3...

1. One operand of PIN(64) specifies the number of bytes to input. If the value of this operand is less than the content of AR 06, then the bytes input will not equal the bytes received.
2. PIN(64) might be executed before the RS-232C Reception Completed Flag (AR 0414) is turned ON. In this case, the number of bytes received would continue increasing as more data was received.

The content of AR 08 is updated each time that PIN(64) is executed.

8-3 SR Area Bits and Flags

The SR area contains several bits and flags which are used in conjunction with the RS-232C interface and Host Link Systems.

SR bit	Functions
25208	1. RS-232C Interface Error Flag 2. CPU-mounting Host Link Unit Communications Error Flag
25209	1. RS-232C Restart Bit 2. CPU-mounting Host Link Unit Restart Bit

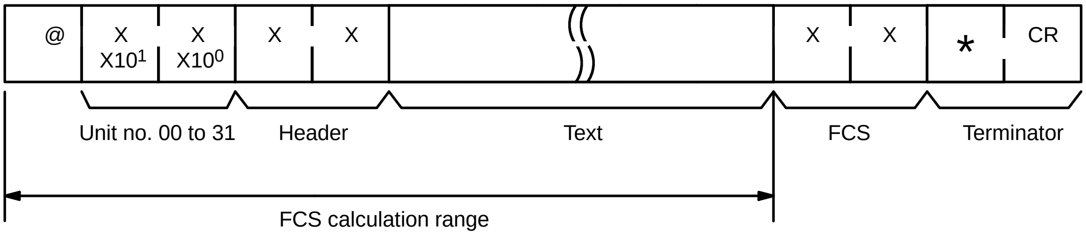
The Error Flag is turned ON when an error (e.g., parity, framing, overrun, or FCS error) has occurred either in communications via the RS-232 Interface or between the CPU and the CPU-mounting Host Link Unit. When the Restart Bit is turned ON and then OFF, both the RS-232C Interface and the CPU-mounting Host Link Unit will be restarted and SR 25208, AR 0400 to AR 0407, and AR 05 will be cleared.

8-4 Host Link Communications Protocol

The host computer has initial transmission priority. Data transfer between the host computer and the Host Link System is, therefore, initiated when the computer sends a command to a PC in the Host Link System.

A set of data in a transmission is called a block. The data block sent from the host computer to the PC is called a command block. The block sent from the PC to the computer is called a response block. Each block starts with a unit number and a header, and ends with a Frame Check Sequence (FCS) code and a terminator (* and CR). The terminator in the command block enables the PC to send a response. The terminator in the response block enables the host computer to send another command.

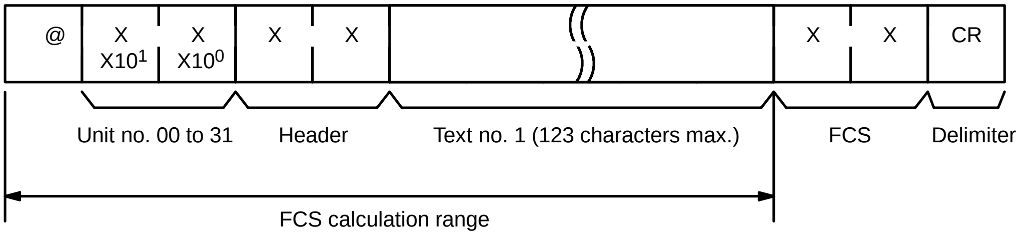
8-4-1 Block Format



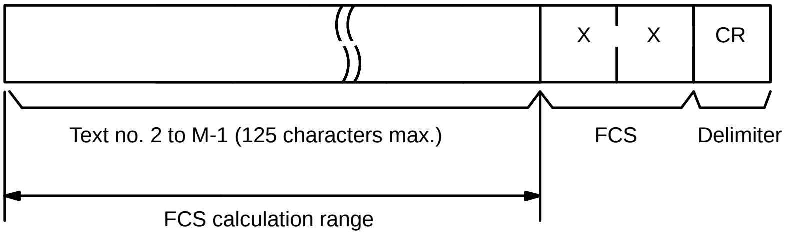
A block is usually made up of one unit called a frame, but long blocks of data (over 131 characters) must be divided into more than one frame before transmission. The first frame can have up to 131 characters, and subsequent frames can have up to 128 characters. The data must thus be divided into more than one frame when there is a block consisting of more than 131 characters. When multiple frames are used, the beginning and intermediate frames end with a delimiter (CR), instead of a terminator (*CR).

8-4-2 Block Format With More Than One Frame

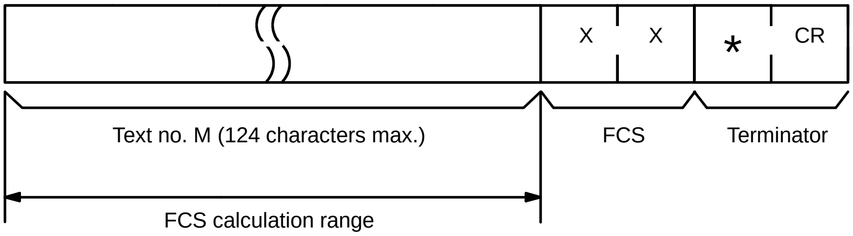
First Frame (131 Characters or Less)



Intermediate Frame(s) (128 Characters or Less)

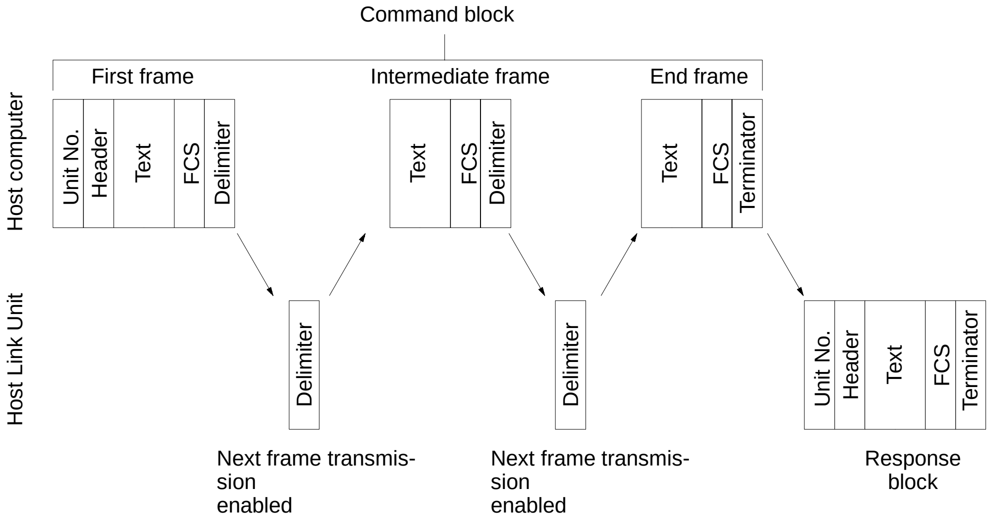


Last Frame (128 Characters or Less)



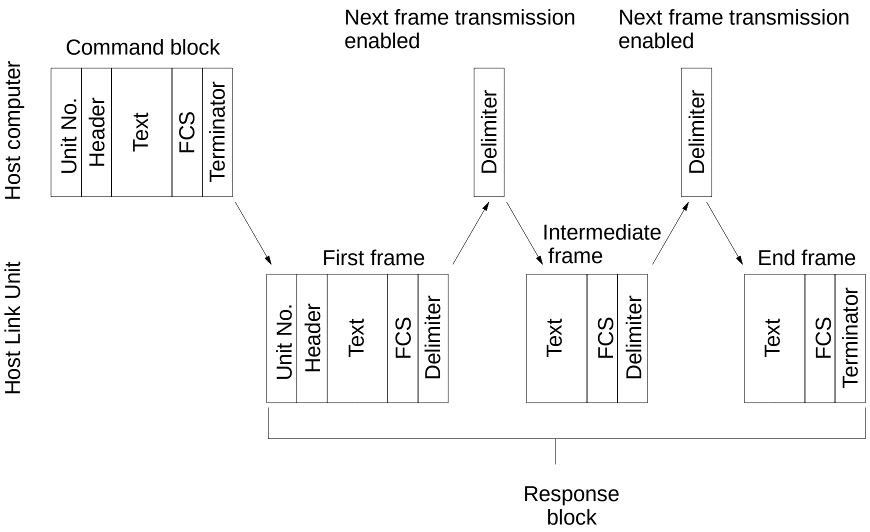
Sending Commands

To send a command block with more than one frame from the computer, initially send only the first frame in the block. Do not send the next frame until the host computer has received the delimiter which should have been sent back from the PC. Do not separate data from a single word into different frames for any write command.



Receiving Commands

To receive a response block consisting of more than one frame from the PC, the host computer must send the carriage return code (delimiter) to the PC after receiving the delimiter from the PC. This enables the PC to send the next frame.



8-4-3 Data Representation

Numerical data within a transmission is expressed in hexadecimal, decimal, or binary format. Refer to the format example of each command in *8-5 Host Link Commands and Responses* for details. The appropriate range is indicated in the following manner.

Hexadecimal Data

	$X16^3$	$X16^2$	$X16^1$	$X16^0$	

In the above diagram, the elements $X16^3$ to $X16^0$ indicate that the data is expressed in hexadecimal. Each digit can, therefore, be in the range from 0 (ASCII 48_{dec}, binary 0000) to 9 (ASCII 49_{dec}, binary 1001), or A (ASCII 65_{dec}, binary 1010) to F (ASCII 70_{dec}, binary 1111).

Decimal Data

	$X10^3$	$X10^2$	$X10^1$	$X10^0$	

In this figure, $X10^3$ to $X10^0$ indicate that the data is expressed in decimal. Each digit can, therefore, be in the range from 0 (binary 0000) to 9 (binary 1001).

Binary Data

	ON/ OFF $X2^3$	ON/ OFF $X2^2$	ON/ OFF $X2^1$	ON/ OFF $X2^0$	

In the above figure, the ON/OFF and $X2^3$ to $X2^0$ indicate that the data is binary. Each box therefore represents either 0 or 1 as follows:

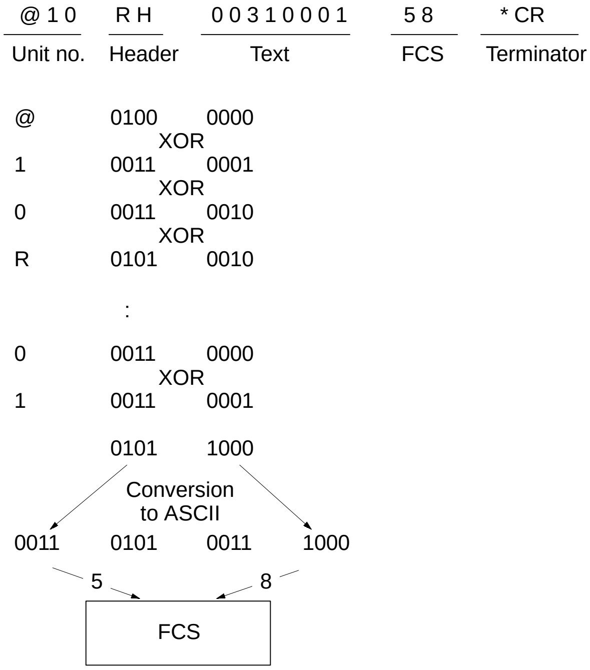
0 (ASCII 48_{dec}): OFF
1 (ASCII 49_{dec}): ON.

Data Areas

Data area codes must be entered in capital letters and must be 4 characters wide. Names shorter than 4 characters must be followed by spaces (ASCII 32_{dec}) to make up the 4 characters. Data areas valid for each command are listed with the command.

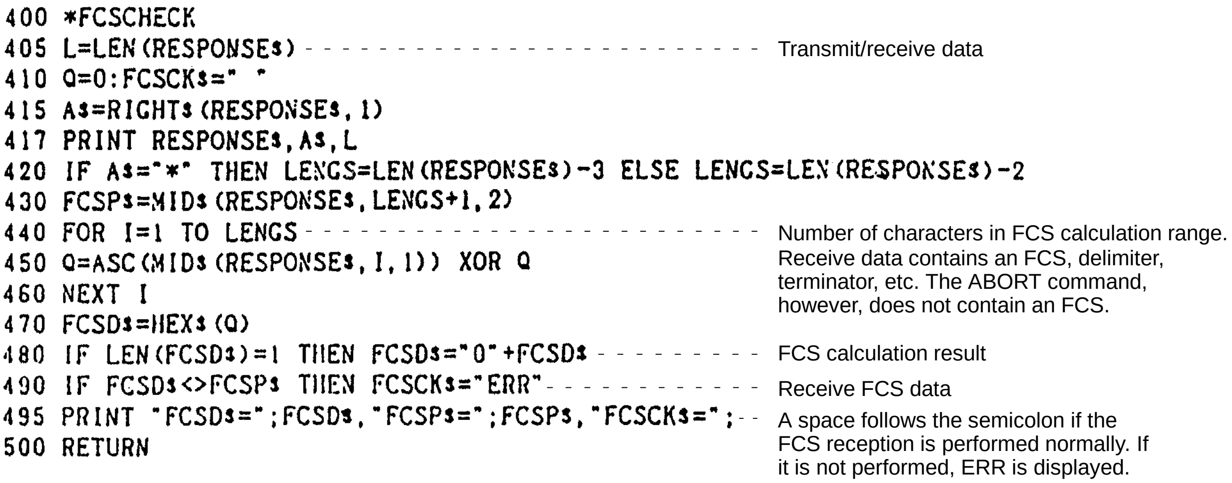
8-4-4 FCS Calculation

The FCS is 8-bit data converted into two ASCII characters. The 8-bit data is the result of an EXCLUSIVE OR sequentially performed between each character, from the first character in the frame to the last character of the text in that frame.



8-4-5 FCS Calculation Program Example

The following program is an example of how FCS calculation can be performed on received data.



```
400 *FCSCHECK
405 L=LEN(RESPONSE$)
410 Q=0:FCSCK$=" "
415 A$=RIGHT$(RESPONSE$,1)
417 PRINT RESPONSE$,A$,L
420 IF A$="*" THEN LENGS=LEN(RESPONSE$)-3 ELSE LENGS=LEN(RESPONSE$)-2
430 FCSP$=MID$(RESPONSE$,LENGS+1,2)
440 FOR I=1 TO LENGS
450 Q=ASC(MID$(RESPONSE$,I,1)) XOR Q
460 NEXT I
470 FCSD$=HEX$(Q)
480 IF LEN(FCSD$)=1 THEN FCSD$="0"+FCSD$
490 IF FCSD$<>FCSP$ THEN FCSCK$="ERR"
495 PRINT "FCSD$=";FCSD$,"FCSP$=";FCSP$,"FCSCK$=";
500 RETURN
```

Transmit/receive data

Number of characters in FCS calculation range. Receive data contains an FCS, delimiter, terminator, etc. The ABORT command, however, does not contain an FCS.

FCS calculation result

Receive FCS data

A space follows the semicolon if the FCS reception is performed normally. If it is not performed, ERR is displayed.

Note: in this example, CR (CHR$(13)) is not included in RESPONSE$.

8-5 Host Link Commands and Responses

In the Host Link Mode, the host computer can both monitor and control the PC. The host computer monitors the PC by sending commands to the PC requesting various types of data: program data, I/O data, and error data. The host computer controls the PC by writing various types of data: data that changes the PC operating mode, program data, I/O data, and memory area data. In either case it is the host computer that initiates all communications.

Because the PC is passive is all communications, it cannot monitor host computer errors, it can only check communication errors existing in the data it receives. These are checked for through parity check and frame check sequence.

The response time will vary in accordance with the transfer speed, the amount of data, and the PC cycle time. The RS-232C interface servicing time can be set in the System DM. The longer the servicing time, the faster the response time. Long service times will increase the cycle time, possibly causing inaccuracies in timers (see *Section 6*). If the servicing time is extremely short, then the response time will be extremely slow. The following data shows the actual times required to read 10 words of DM data from a PC with a cycle time of 30 ms for various settings of the servicing time.

0% servicing time:	7.0 s
1% servicing time:	0.90s
2% servicing time:	0.28 s
10% servicing time:	0.19 s
50% servicing time:	0.14 s
99% servicing time:	0.14 s

The rest of this section describes the commands sent from the host computer to the PC. Tables summarizing the complete set of instructions according to their command level are included at the end of this section (see *Section 8-6*).

8-5-1 TEST

Transmits one block of data to the PC and then returns it, unaltered, to the host computer. Each frame is treated as a block regardless of whether it used a terminator or delimiter.

Command Format

@	Unit no.		T	S	Any characters (118 max.) other than a carriage return	FCS	*	CR
	$X10^1$	$X10^0$						

Response Format

@	Unit no.		T	S	Any characters (118 max.) other than a carriage return	FCS	*	CR
	$X10^1$	$X10^0$						

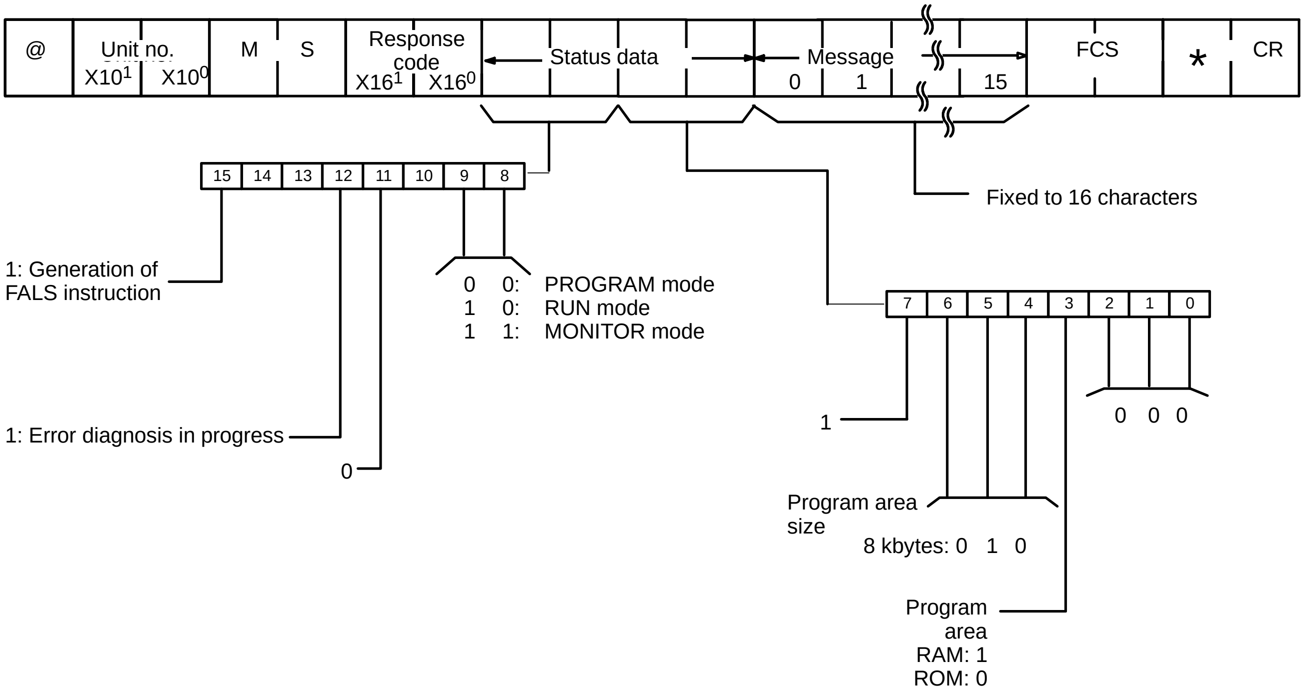
8-5-2 STATUS READ

This command causes the PC to read the operating status of the PC. A message is entered only when MSG(46) has been executed.

Command Format

@	Unit no.		M	S	FCS	*	CR
	$X10^1$	$X10^0$					

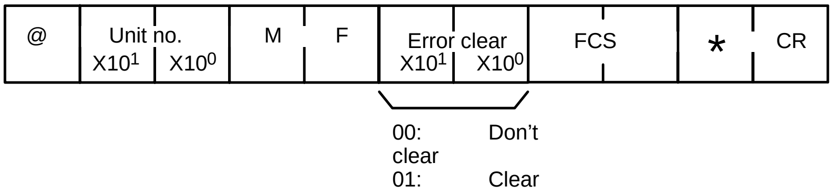
Response Format



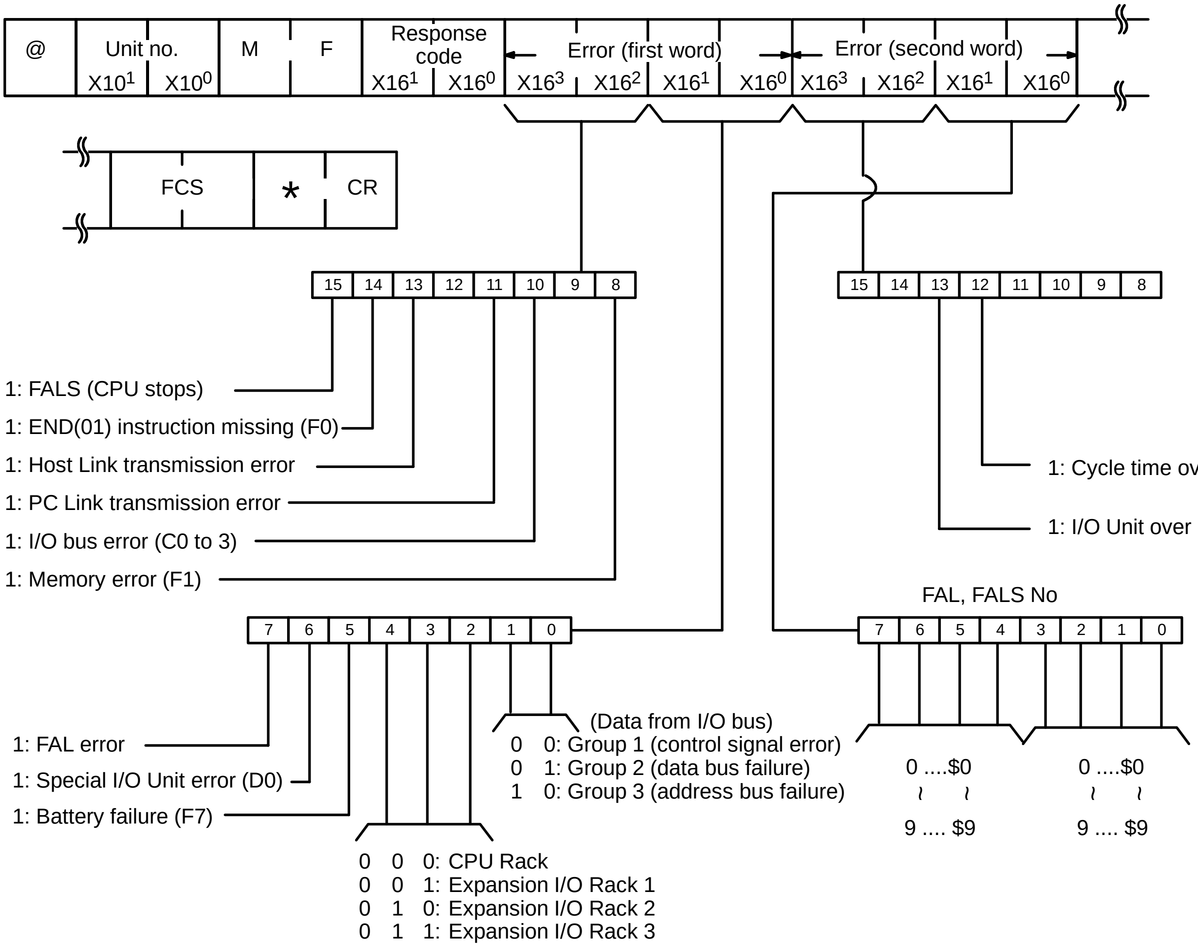
8-5-3 ERROR READ

Reads and clears errors in the PC. Also checks whether previous errors have already been cleared.

Command Format



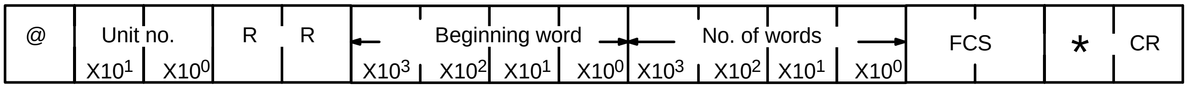
Response Format



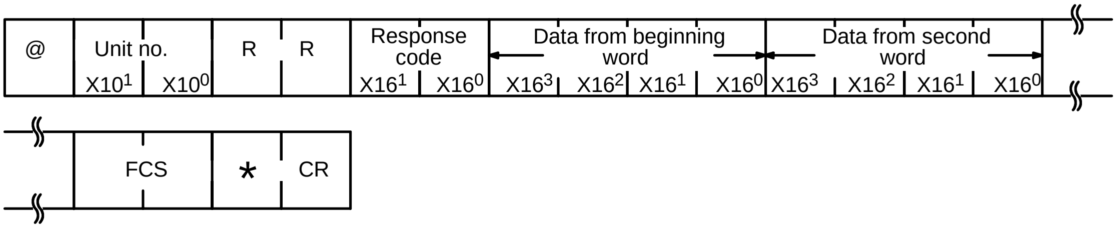
8-5-4 IR AREA READ

Reads the contents of the specified number of IR words, starting from the specified word.

Command Format



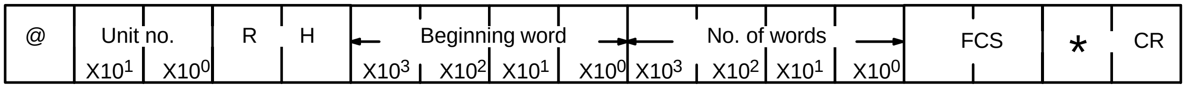
Response Format



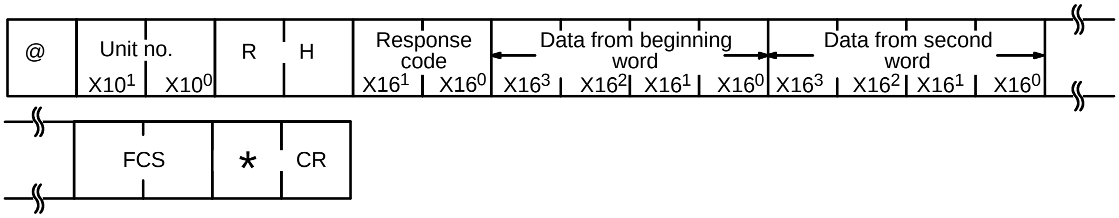
8-5-5 HR AREA READ

Reads the contents of the specified number of HR words, starting from the specified word.

Command Format



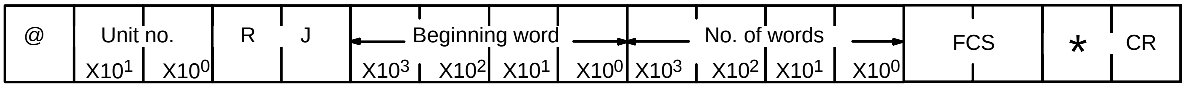
Response Format



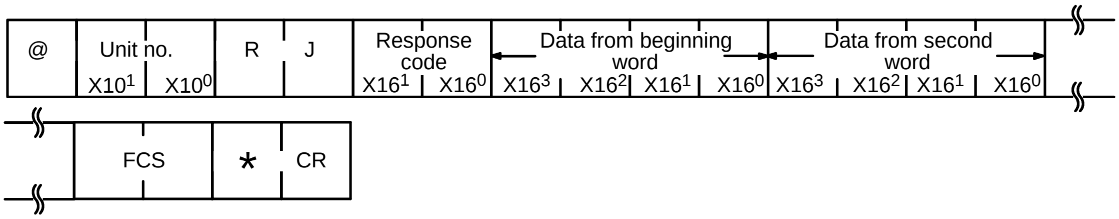
8-5-6 AR AREA READ

Reads the contents of the specified number of AR words, starting from the specified word.

Command Format



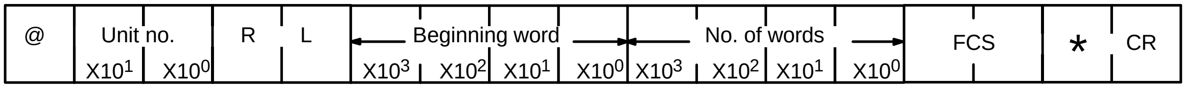
Response Format



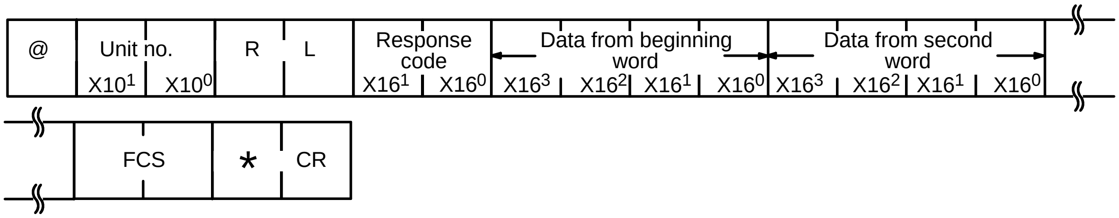
8-5-7 LR AREA READ

Reads the contents of the specified number of LR words, starting from the specified word.

Command Format



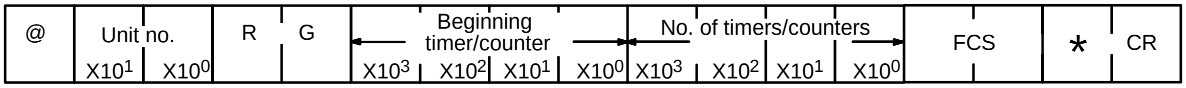
Response Format



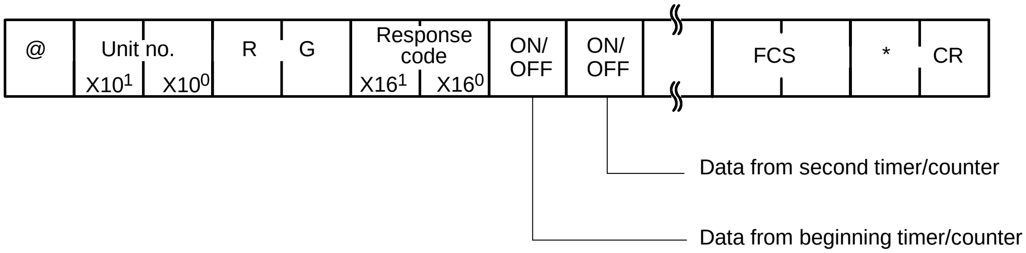
8-5-8 TC STATUS READ

Reads the status of the Completion Flags of the specified number of timers/counters, starting from the specified timer/counter.

Command Format



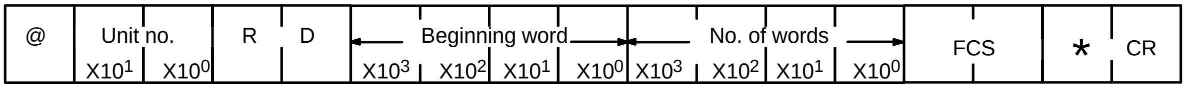
Response Format



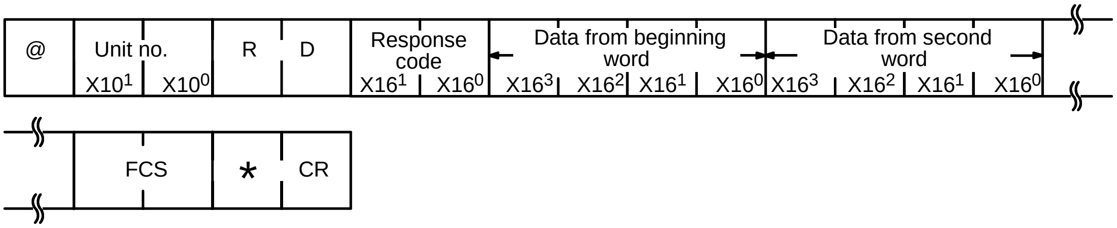
8-5-9 DM AREA READ

Reads the contents of the specified number of DM words, starting from the specified word.

Command Format



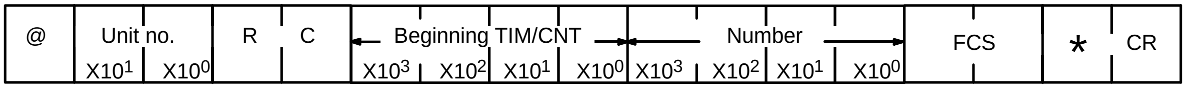
Response Format



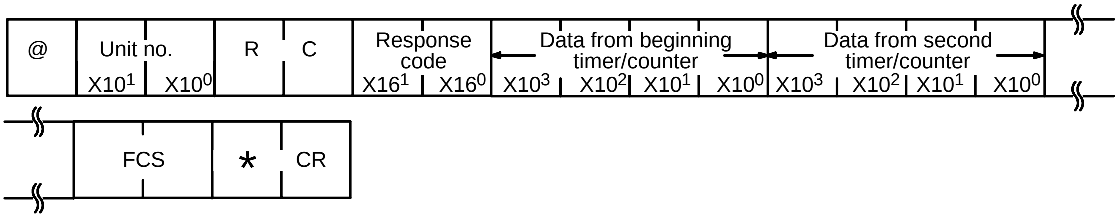
8-5-10 PV READ

Reads the specified number of timer/counter PVs (present values), starting from the specified timer/counter.

Command Format



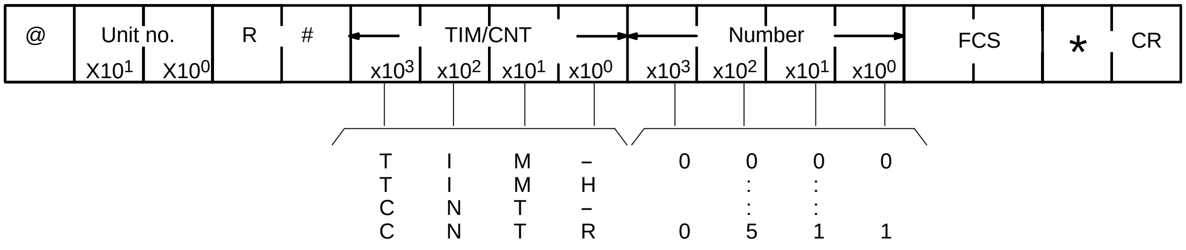
Response Format



8-5-11 SV READ 1

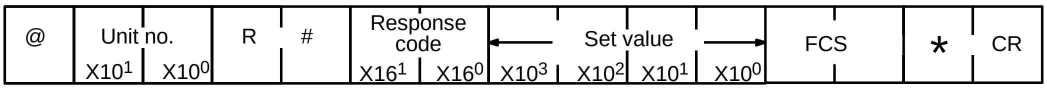
Reads the set value (a constant) of the specified timer/counter instruction. Reads from the beginning of the program and may therefore require about 20 seconds or more to produce a response. Refer also to SV READ 2.

Command Format



Note: Dashes represent spaces.

Response Format

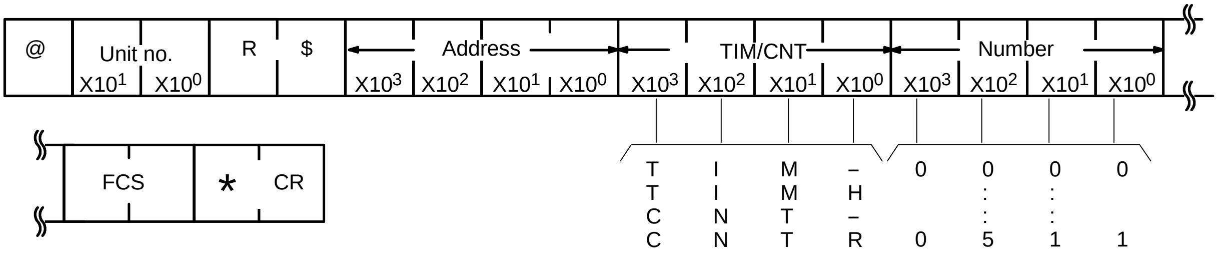


If the command is used more than once, the set value of only the first instruction will be read. If the second word (the operand) is not a constant, an error response (16) will be returned.

8-5-12 SV READ 2

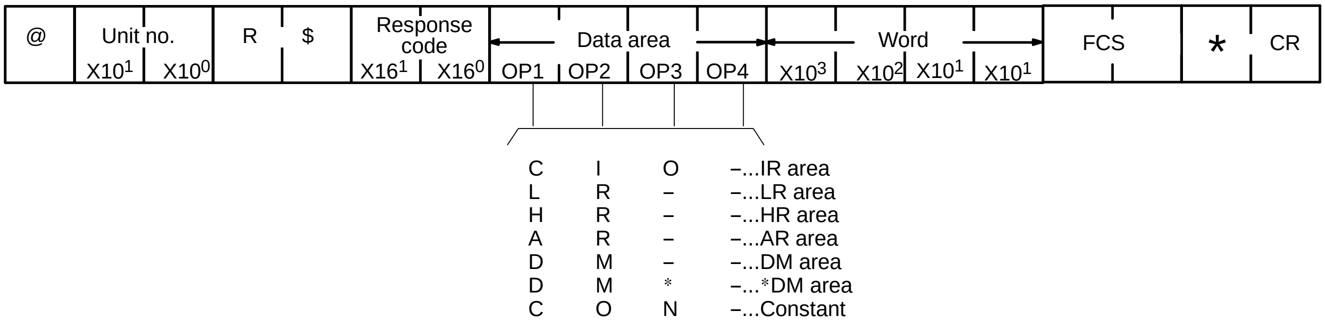
Reads the set value (a constant, or data area and word) of the specified timer/counter instruction. The timer/counter instruction is designated by program address.

Command Format



Note: Dashes represent spaces.

Response Format

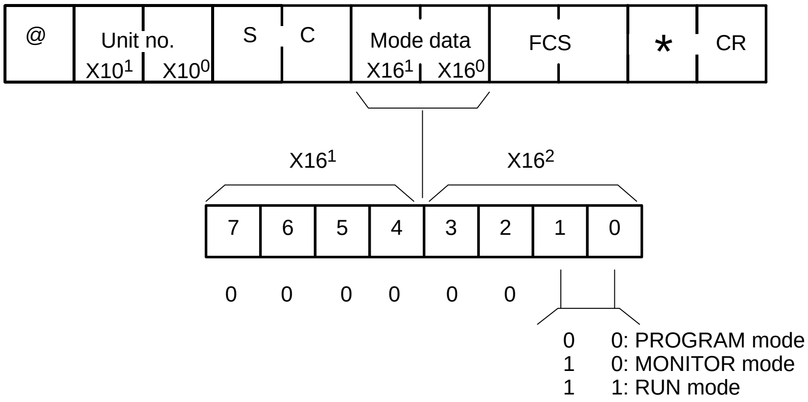


Note: Dashes represent spaces.

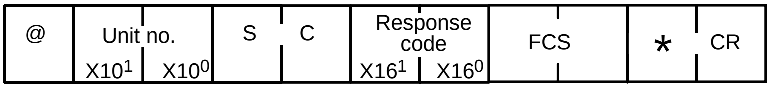
8-5-13 STATUS WRITE

Changes the operating mode of the PC according to the information input into digit X16^0.

Command Format



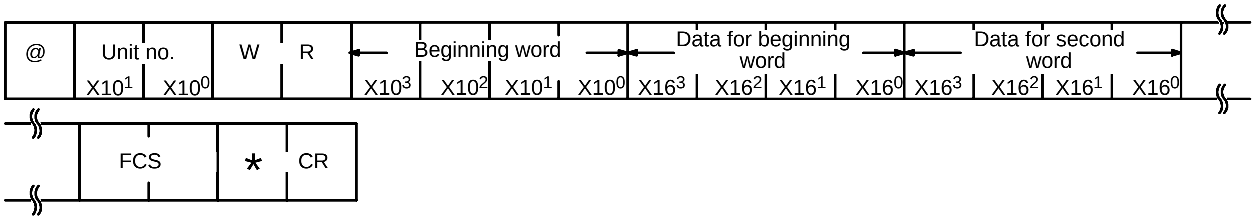
Response Format



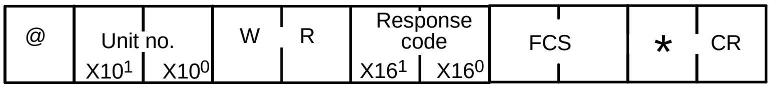
8-5-14 IR AREA WRITE

Writes data to the IR area, starting from the specified word. Writing is done word by word.

Command Format



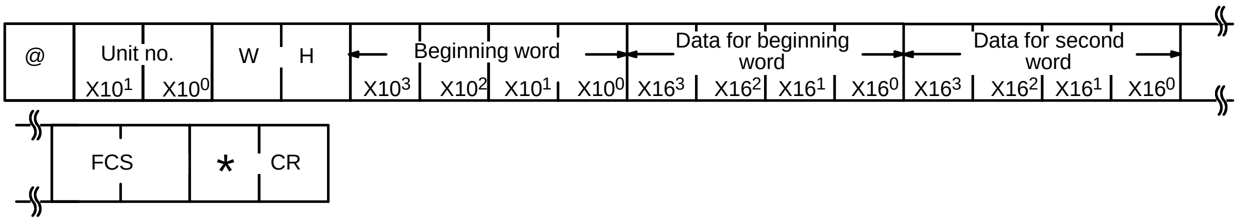
Response Format



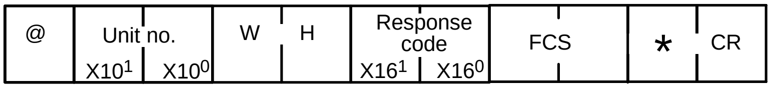
8-5-15 HR AREA WRITE

Writes data to the HR area, starting from the specified word. Writing is done word by word.

Command Format



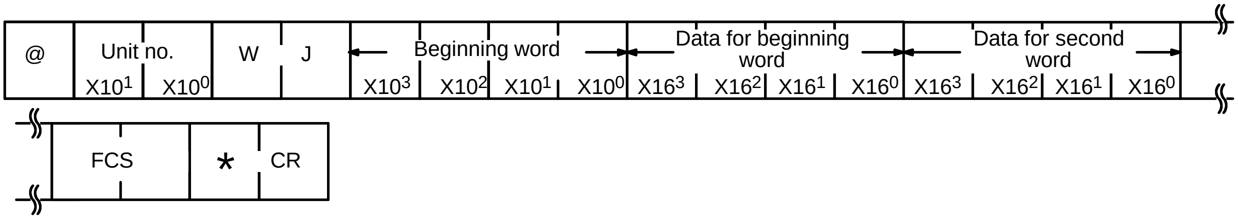
Response Format



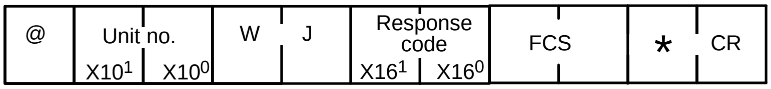
8-5-16 AR AREA WRITE

Writes data to the AR area, starting from the specified word. Writing is done word by word.

Command Format



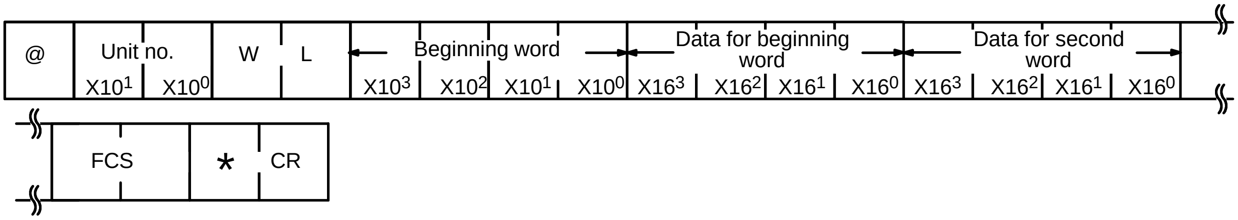
Response Format



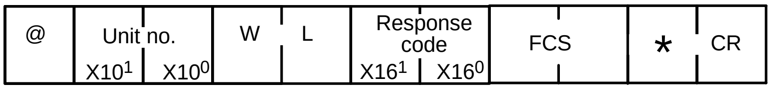
8-5-17 LR AREA WRITE

Writes data to the LR area, starting from the specified word. Writing is done word by word.

Command Format



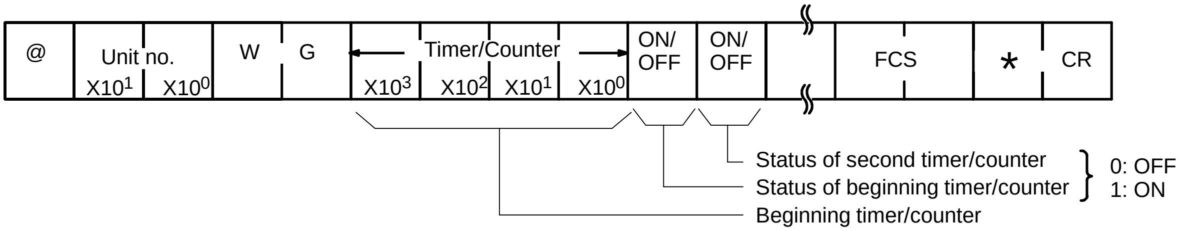
Response Format



8-5-18 TC STATUS WRITE

Writes the status of Completion Flags to the TC area, starting form the specified timer/counter.

Command Format



Response Format

@	Unit no.		W	G	Response code		FCS		*	CR
	$X10^1$	$X10^0$			$X16^1$	$X16^0$				

8-5-19 DM AREA WRITE

Writes data to the DM area, starting from the specified word. Writing is done word by word. If the Program Memory is in an EPROM chip, or if the write enable switch is set to OFF, the the writing range only extends up to DM 0999.

Command Format

@	Unit no.		W	D	Beginning word				Data for beginning word				Data for second word			
	$X10^1$	$X10^0$			$X10^3$	$X10^2$	$X10^1$	$X10^0$	$X16^3$	$X16^2$	$X16^1$	$X16^0$	$X16^3$	$X16^2$	$X16^1$	$X16^0$

FCS		*	CR

Response Format

@	Unit no.		W	D	Response code		FCS		*	CR
	$X10^1$	$X10^0$			$X16^1$	$X16^0$				

8-5-20 PV WRITE

Writes the PVs (present values) of timers/counters starting from the specified timer/counter.

Command Format

@	Unit no.		W	C	Beginning word				Data for beginning word				Data for second word			
	$X10^1$	$X10^0$			$X10^3$	$X10^2$	$X10^1$	$X10^0$	$X10^3$	$X10^2$	$X10^1$	$X10^0$	$X10^3$	$X10^2$	$X10^1$	$X10^0$

FCS		*	CR

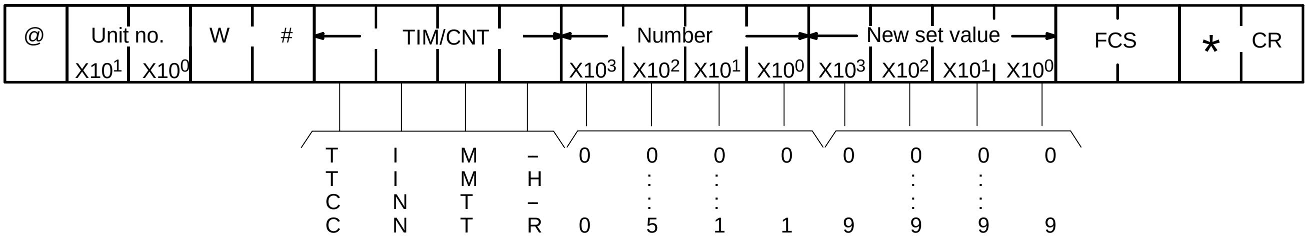
Response Format

@	Unit no.		W	C	Response code		FCS		*	CR
	$X10^1$	$X10^0$			$X16^1$	$X16^0$				

8-5-21 SV CHANGE 1

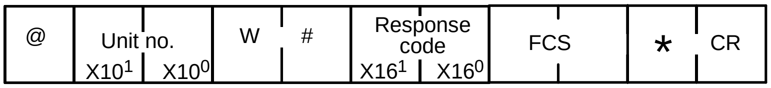
Changes the set value (constant only) of the specified timer/counter instruction. Reads from the beginning of the program and may therefore require up to about 20 seconds to produce a response. Refer also to SV CHANGE 2.

Command Format



Note: Dashes represent spaces.

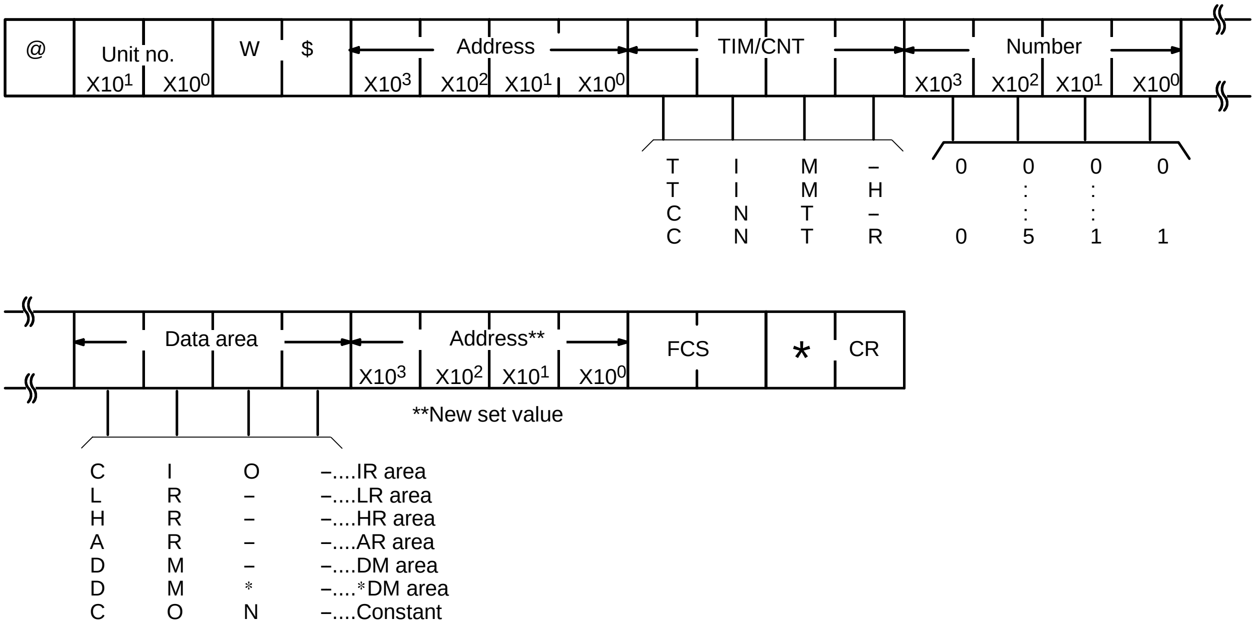
Response Format



8-5-22 SV CHANGE 2

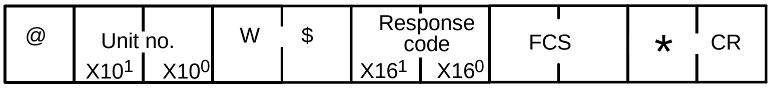
Changes the set value (a constant, or data area and word) of the specified timer/counter instruction. The instruction is specified by program address.

Command Format



Note: Dashes represent spaces.

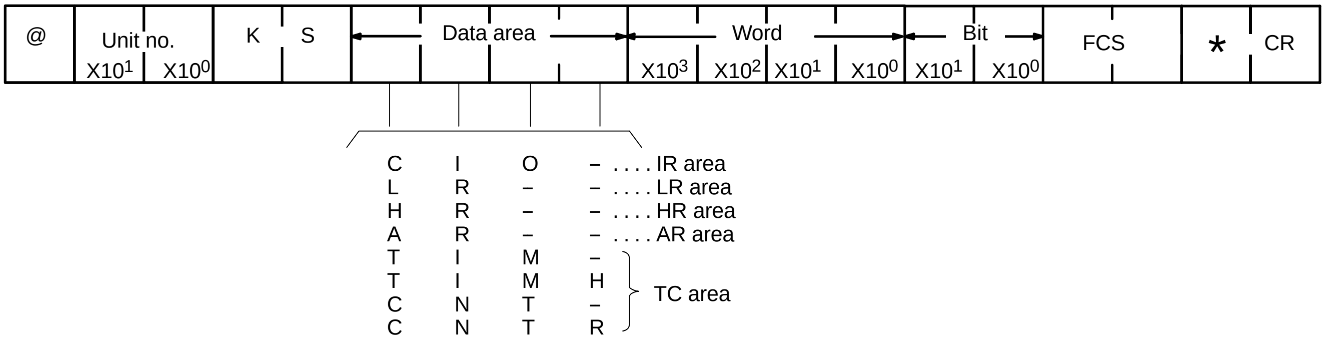
Response Format



8-5-23 FORCED SET

Forced sets a bit in an IR, LR, HR, AR, or TC area. Bits need to be force set one at a time.

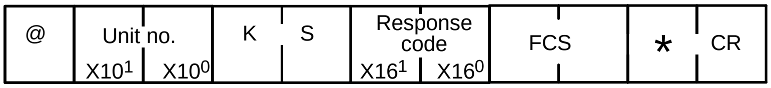
Command Format



C	I	O	–	. . . . IR area
L	R	–	–	. . . . LR area
H	R	–	–	. . . . HR area
A	R	–	–	. . . . AR area
T	I	M	–	TC area
T	I	M	H	
C	N	T	–	
C	N	T	R	

Note: Dashes represent spaces.

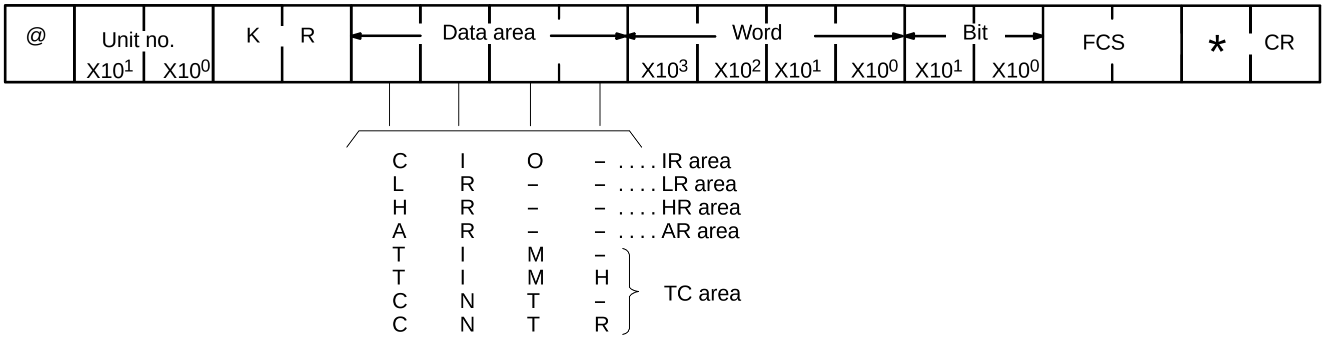
Response Format



8-5-24 FORCED RESET

Force resets a bit in an IR, LR, HR, AR, or TC area. Bits can only be force set one at a time. If an attempt is made to simultaneously force reset more than one bit, none of the bits will reset.

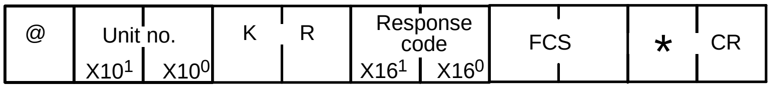
Command Format



C	I	O	–	. . . . IR area
L	R	–	–	. . . . LR area
H	R	–	–	. . . . HR area
A	R	–	–	. . . . AR area
T	I	M	–	TC area
T	I	M	H	
C	N	T	–	
C	N	T	R	

Note: Dashes represent spaces.

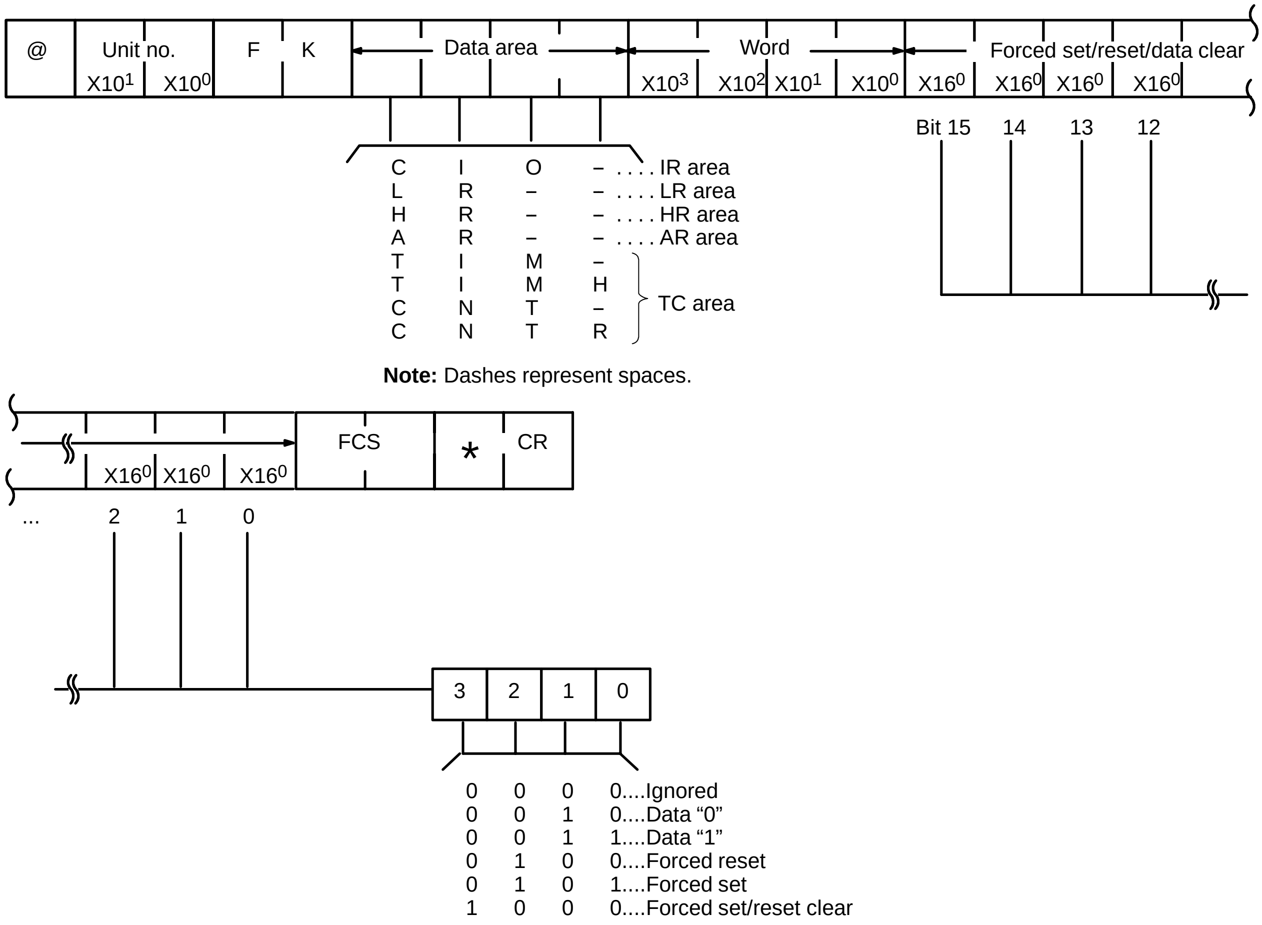
Response Format



8-5-25 MULTIPLE FORCED SET/RESET

This command force sets or resets bits in the IR, LR, HR, AR, or TC areas. All forced status will be lost if the PC is switched to RUN mode.

Command Format

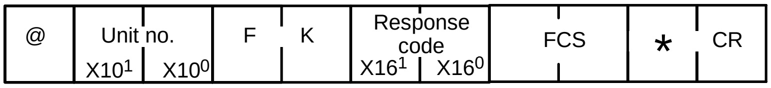


C	I	O	–	. . . . IR area
L	R	–	–	. . . . LR area
H	R	–	–	. . . . HR area
A	R	–	–	. . . . AR area
T	I	M	–	TC area
T	I	M	H	
C	N	T	–	
C	N	T	R	

Note: Dashes represent spaces.

0	0	0	0	....Ignored
0	0	1	0	....Data “0”
0	0	1	1	....Data “1”
0	1	0	0	....Forced reset
0	1	0	1	....Forced set
1	0	0	0	....Forced set/reset clear

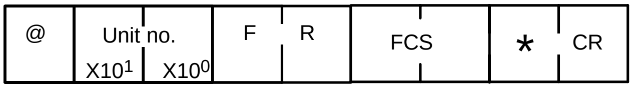
Response Format



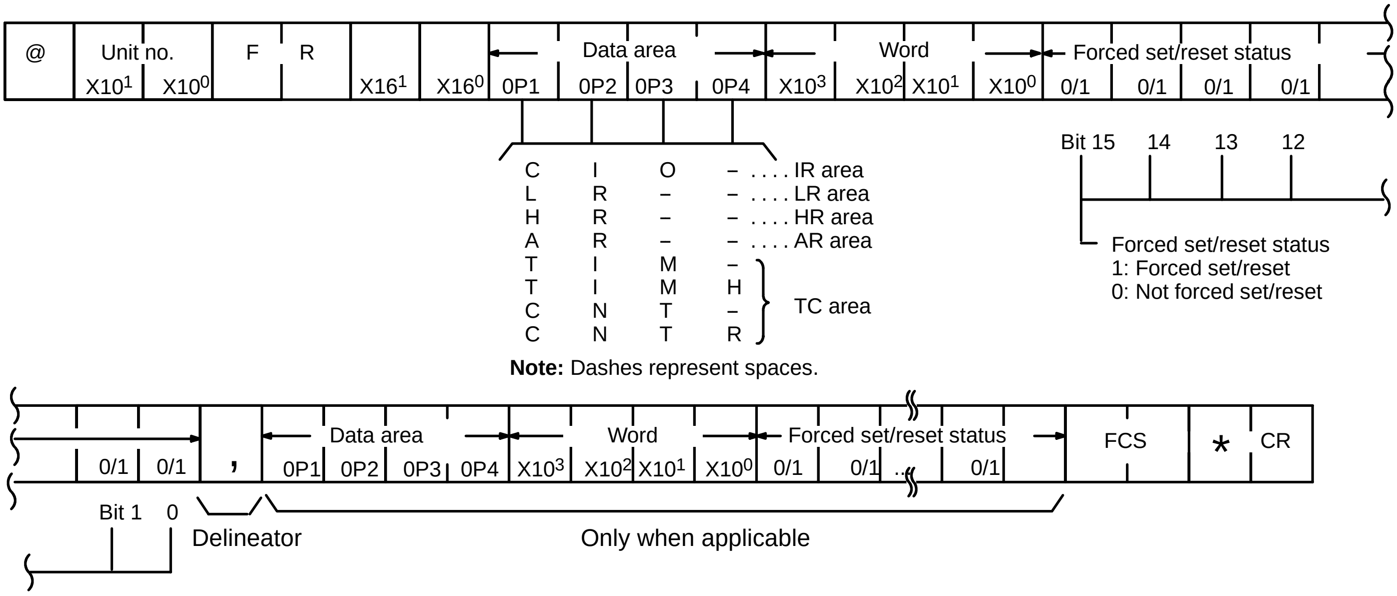
8-5-26 MULTIPLE FORCED SET/RESET STATUS READ

Reads the forced set or forced reset status of the PC.

Command Format



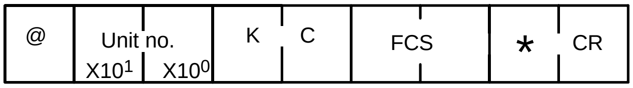
Response Format



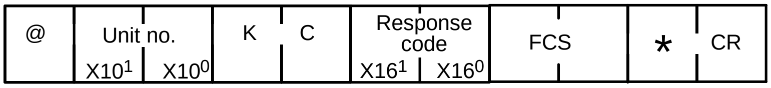
8-5-27 FORCED SET/RESET CANCEL

Cancels all forced set and forced reset bits (including those achieved via MULTIPLE FORCED SET/RESET.

Command Format



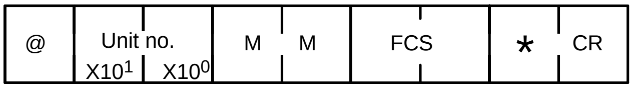
Response Format



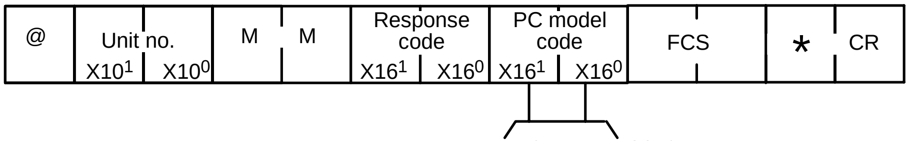
8-5-28 PC MODEL READ

Reads the model type of the PC.

Command Format



Response Format



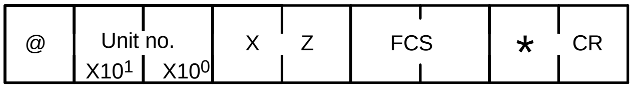
0	1	C250 or P-type
0	2	C500
0	3	C120 or C50
0	9	C250F
0	A	C500F
0	B	C120F
0	E	C2000
1	0	C1000H
1	1	C2000H or K-type
1	2	C200H or C20H/C28H/C40H/C60H
1	3	C1000HF

8-5-29 ABORT and INITIALIZE

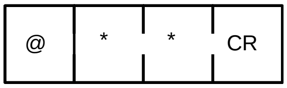
The ABORT command is used to abort the process being performed by the Host Link function and to then enable reception of the next command. The INITIALIZE command initializes the transmission control procedure of all the PCs connected to the host computer. Neither command receives a response.

A processing time of 100 ms is required between reception of the ABORT or INITIALIZE commands, and reception of the next command. If INITIALIZE is used in a single-link system, it will be regarded as undefined.

ABORT Command Format



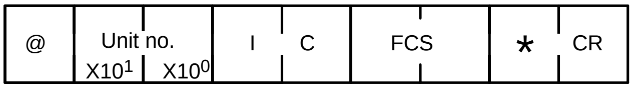
INITIALIZE Command Format



8-5-30 Response to an Undefined Command

This response is sent if the PC cannot read the command's header code, or if the specified command is not valid for the command level or model of PC. If this response is received check the header code, command level, and PC model, then execute the correct command.

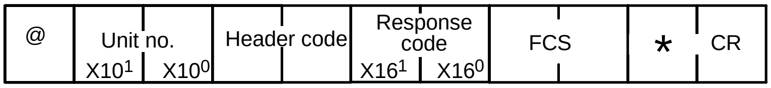
Response Format



8-5-31 Response Indicating an Unprocessed Command

This response is sent when the PC cannot process a command. The type of error encountered by the PC can be identified via the response code. (See *Section 8-5-36*.)

Response Format

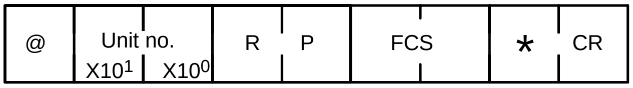


The header code varies according to the command which was sent. The headers of some commands include subheader codes (e.g., I/O REGISTER, I/O READ, and DM SIZE CHANGE).

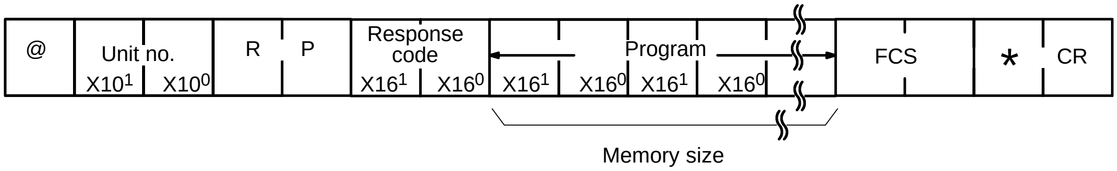
8-5-32 PROGRAM READ

Transmits the contents of the PC program memory.

Command Format



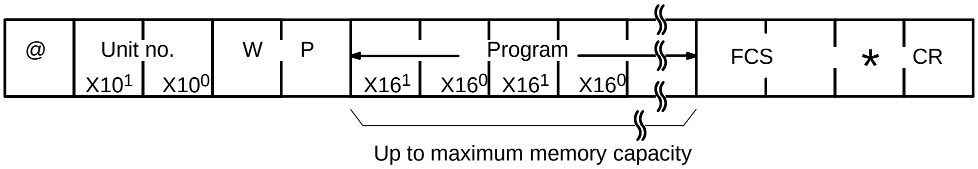
Response Format



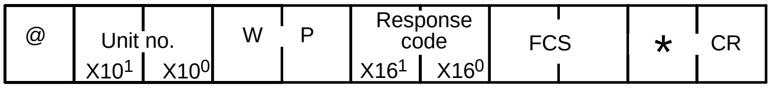
8-5-33 PROGRAM WRITE

Writes the received program into the PC program memory.

Command Format



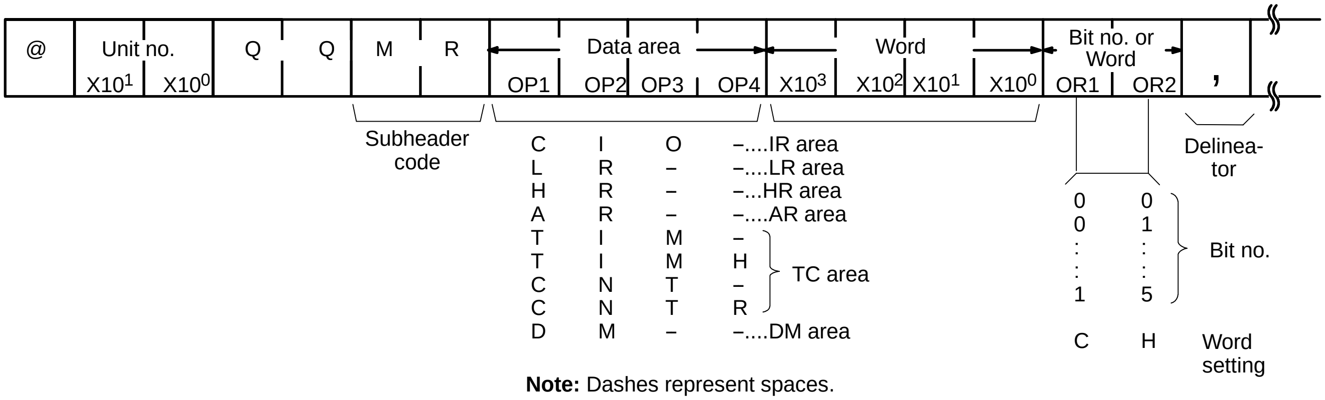
Response Format



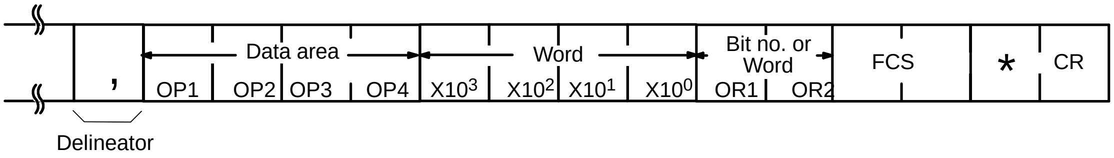
8-5-34 I/O REGISTER

Registers the IR, LR, HR, AR, or TC area bit, or the DM word that is to be read via I/O READ (described in the next subsection). Registered data is retained until new data is registered, or the power is turned OFF.

Command Format



Note: Dashes represent spaces.



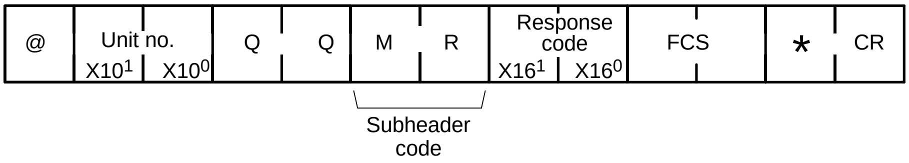
Setting Table

	Data Area	Word Address	Bit or Word Setting	Response
Bit	IR	0000 to 0255	00 to 15	ON/OFF
	LR	0000 to 0063	00 to 15	ON/OFF
	HR	0000 to 0099	00 to 15	ON/OFF
	AR	0000 to 0027	00 to 15	ON/OFF
	TIM/CNT	0000 to 0511	Anything other than CH	ON/OFF
Wd	IR	0000 to 0255	CH	Word data
	LR	0000 to 0063	CH	Word data
	HR	0000 to 0099	CH	Word data
	AR	0000 to 0027	CH	Word data
	TIM/CNT	0000 to 0511	CH	ON/OFF and PV
	DM	0000 to 1999	Any character	Word data

The maximum number of data items is 128. Count the TC area word specification as two items.

The data is registered in the same sequence in which it was specified.

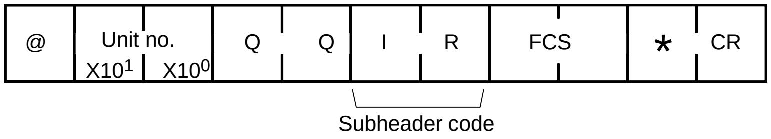
Response Format



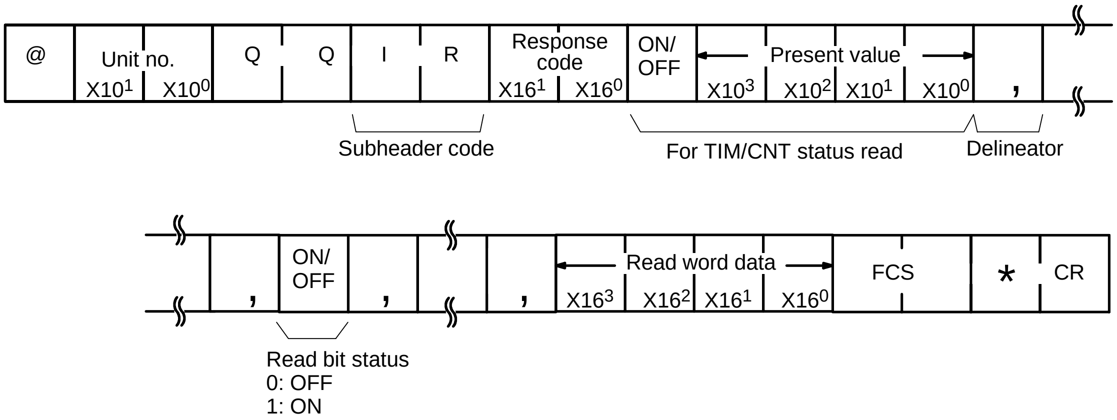
8-5-35 I/O READ

Reads the data specified by I/O REGISTER.

Command Format



Response Format



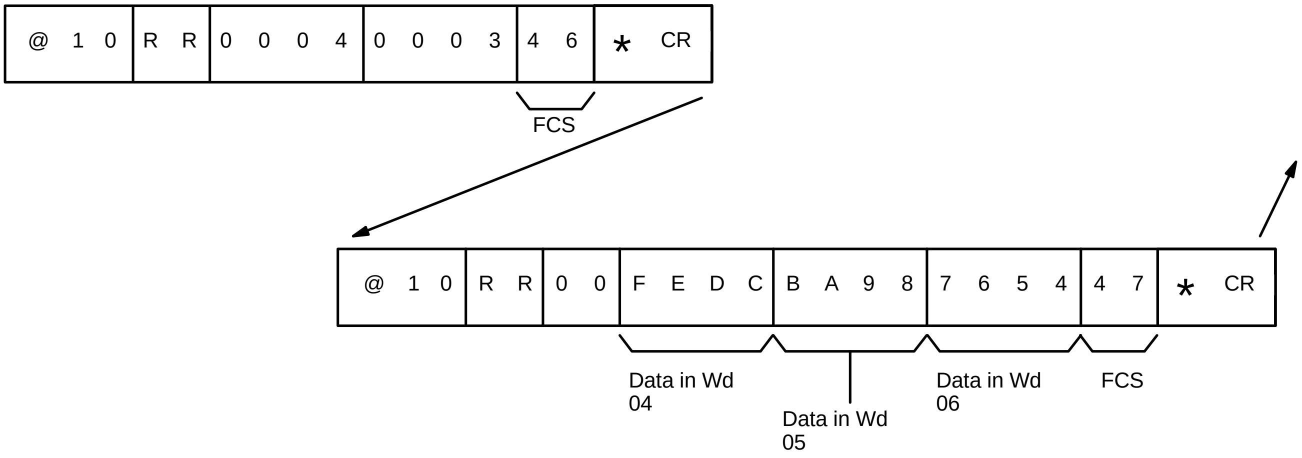
8-5-36 Response Code List

$X16^1$	$X16^0$	Description
0	0	Normal Completion
0	1	Not executable in RUN mode
0	2	Not executable in MONITOR mode
0	3	Not executable with PROM mounted
0	4	Address over (data overflow)
0	B	Not executable in PROGRAM mode
1	0	Parity error
1	1	Framing error
1	2	Overrun
1	3	FCS error
1	4	Format error (parameter length error)
1	5	Entry number data error (parameter error, data code error, data length error)
1	6	Instruction not found
1	8	Frame length error
1	9	Not executable (due to unclearable error, memory error, unwriteable EEPROM, missing I/O table, etc.)
2	2	No Memory Unit mounted
2	3	User memory is write-protected
A	0	Aborted due to parity error in transmit data
A	1	Aborted due to framing error in transmit data
A	2	Aborted due to overrun in transmit data
A	4	Aborted due to format error in transmit data
A	5	Aborted due to entry number data error in transmit data
A	8	Aborted due to frame length error in transmit data
Other		Probably produced by noise. Execute command again.

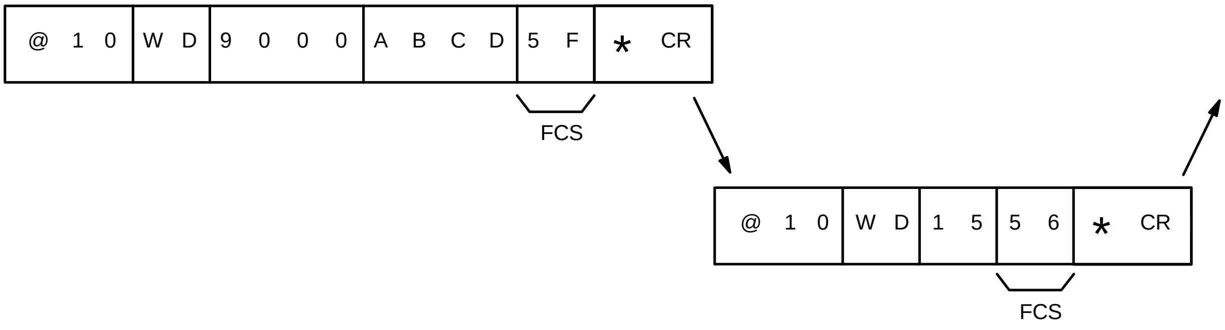
8-5-37 Communications Examples

The following are examples of commands from the host computer (first line) and the responses that would be given by the PC (second line). The arrows indicate the transfer of the right to transmit.

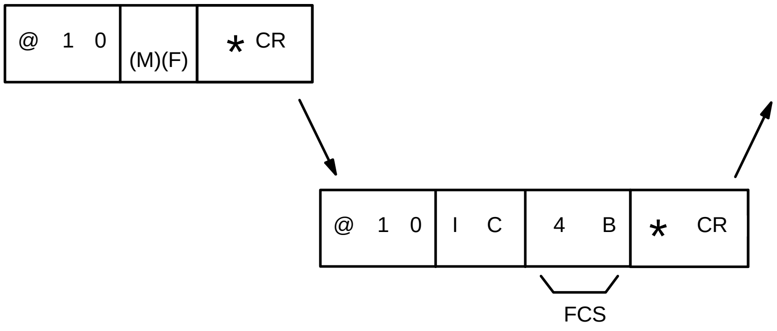
Reading Data from Multiple PC Words (IR 04 to IR 06) (Normal Completion)


@ 1 0 | R R | 0 0 0 4 | 0 0 0 3 | 4 6 | * CR
FCS
@ 1 0 | R R | 0 0 | F E D C | B A 9 8 | 7 6 5 4 | 4 7 | * CR
Data in Wd 04
Data in Wd 05
Data in Wd 06
FCS

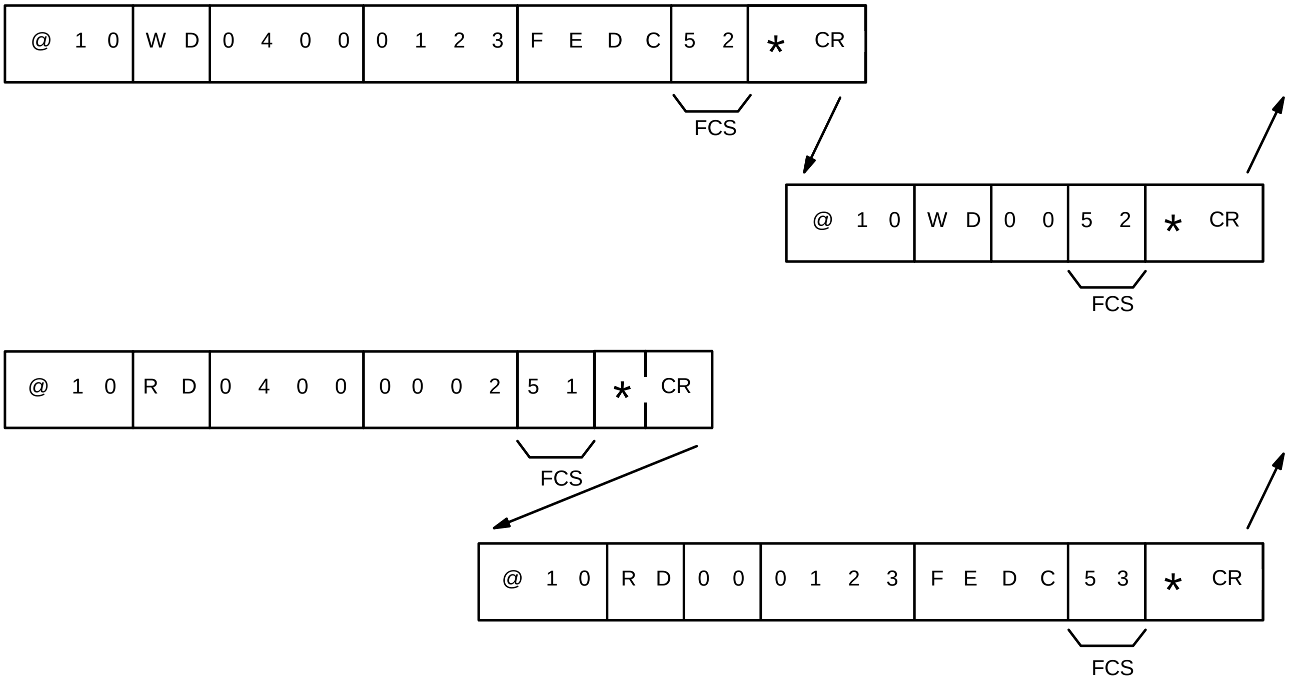
Wd 9000 Specified by Mistake in a DM AREA WRITE Command


@ 1 0 | W D | 9 0 0 0 | A B C D | 5 F | * CR
FCS
@ 1 0 | W D | 1 5 | 5 6 | * CR
FCS

Header Code Destroyed During ERROR READ Operation



Writing Data into PC Words (“0123” to DM 0400 and “FEDC” to DM 0401) and Confirming with DM AREA READ.


@ 1 0 | W D | 0 4 0 0 | 0 1 2 3 | F E D C | 5 2 | * CR
FCS
@ 1 0 | W D | 0 0 | 5 2 | * CR
FCS
@ 1 0 | R D | 0 4 0 0 | 0 0 0 2 | 5 1 | * CR
FCS
@ 1 0 | R D | 0 0 | 0 1 2 3 | F E D C | 5 3 | * CR
FCS

8-6 Command Levels

There are three levels of Host Link Unit commands. These different operating levels allow the user to establish hierarchical protocols to give more sophisticated control.

Level 1

Header Code	Name	PC Mode		
		RUN	MONITOR	PROGRAM
TS	TEST	Valid	Valid	Valid
MS	STATUS READ	Valid	Valid	Valid
MF	ERROR READ	Valid	Valid	Valid
RR	IR AREA READ	Valid	Valid	Valid
RH	HR AREA READ	Valid	Valid	Valid
RJ	AR AREA READ	Valid	Valid	Valid
RL	LR AREA READ	Valid	Valid	Valid
RG	TC STATUS READ	Valid	Valid	Valid
RD	DM AREA READ	Valid	Valid	Valid
RC	PV READ	Valid	Valid	Valid
R#	SV READ 1	Valid	Valid	Valid
R$	SV READ 2	Valid	Valid	Valid
SC	STATUS WRITE	Valid	Valid	Valid
WR	IR AREA WRITE	Not Valid	Valid	Valid
WH	HR AREA WRITE	Not Valid	Valid	Valid
WJ	AR AREA WRITE	Not Valid	Valid	Valid
WL	LR AREA WRITE	Not Valid	Valid	Valid
WG	TC STATUS WRITE	Not Valid	Valid	Valid
WD	DM AREA WRITE	Not Valid	Valid	Valid
WC	PV WRITE	Not Valid	Valid	Valid
W#	SV CHANGE 1	Not Valid	Valid	Valid
W$	SV CHANGE 2	Not Valid	Valid	Valid
KS	FORCED SET	Not Valid	Valid	Not Valid
KR	FORCED RESET	Not Valid	Valid	Not Valid
FK	MULTIPLE FORCED SET/RESET	Not Valid	Valid	Not Valid
FR	MULTIPLE FORCED SET/RESET STATUS READ	Not Valid	Valid	Not Valid
KC	FORCED SET/RESET CANCEL	Not Valid	Valid	Not Valid
MM	PC MODEL READ	Valid	Valid	Valid
IC	Undefined command (response only)	Valid	Valid	Valid
	Unprocessed command (response only)	Valid	Valid	Valid
XZ	ABORT (command only)	Valid	Valid	Valid

Level 2

Header Code	Name	PC Mode		
		RUN	MONITOR	PROGRAM
RP	PROGRAM READ	Valid	Valid	Valid
WP	PROGRAM WRITE	Not valid	Not valid	Valid

Level 3

Header Code	Name	PC Mode		
		RUN	MONITOR	PROGRAM
QQ	I/O REGISTER	Valid	Valid	Valid
QQ	I/O READ	Valid	Valid	Valid

SECTION 9
Troubleshooting

The C20H/C28H/C40H/C60H provide 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. For information on displaying errors, see *7-1 Displaying and Clearing Error Messages*. For information on error flags which can be used in troubleshooting, refer to *3-4 SR Area* and *3-5 AR Area*. For information on errors which can occur when inputting the program, refer to *4-6-3 Checking the Program*.

9-1 Alarm Indicators

There are two indicators on the front of the CPU that provide visual indication of an abnormality in the PC. The error indicator (ERR) indicates fatal errors (i.e., ones that will stop PC operation); the alarm indicator (ALARM) indicates nonfatal ones. These indicators are shown in *2-1 Indicators*.

Caution The PC will turn ON the error indicator (ERR), 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 pages 287 to 288). 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.

9-2 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 contained 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*.

9-3 Reading and Clearing Errors and Messages

System error messages can be displayed onto the Programming Console or any 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 *7-2 Monitoring Operation and Modifying Data*. Procedures for the GPC, LSS, and FIT are provided in the relevant operation manuals.

9-4 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. In addition, there are errors which can occur when inputting the program. For information on these, and their message displays, refer to *4-6-3 Checking the Program*.

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 Error

The following error message appears before program execution has been started. The POWER indicator will be lit, the RUN indicator will not be lit, and the RUN output will be OFF for this error.

Error and message	FAL no.	Probable cause	Possible remedy
Waiting for Units CPU WAITãG	None	Units have not started.	Check all system components to be sure they are working properly and replace all faulty Units.

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 light and the ALARM/ERROR indicator will flash. 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).
	9E	Checksum Flag (AR 1315) is ON.	Check Parameter and Parameter Backup areas. Set and back up parameters with system command.
Cycle time overrun SCAN TIME OVER	F8	Watchdog timer has exceeded 100 ms.	Program cycle time is longer than recommended. Reduce cycle time if possible.
Battery error BATT LOW	F7	Backup battery is missing or its voltage has dropped.	Check battery and replace if necessary.
Host Link Error No message	None	• Error exists between host computer and built-in Host Link Interface.	• Check link set-up and requirements.

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. All CPU indicators will not be lit for the power interruption error. For all other fatal operating errors, the POWER and ALARM/ERROR indicators will be lit. The RUN output will be OFF. For power interruptions, all indicators will not be lit.

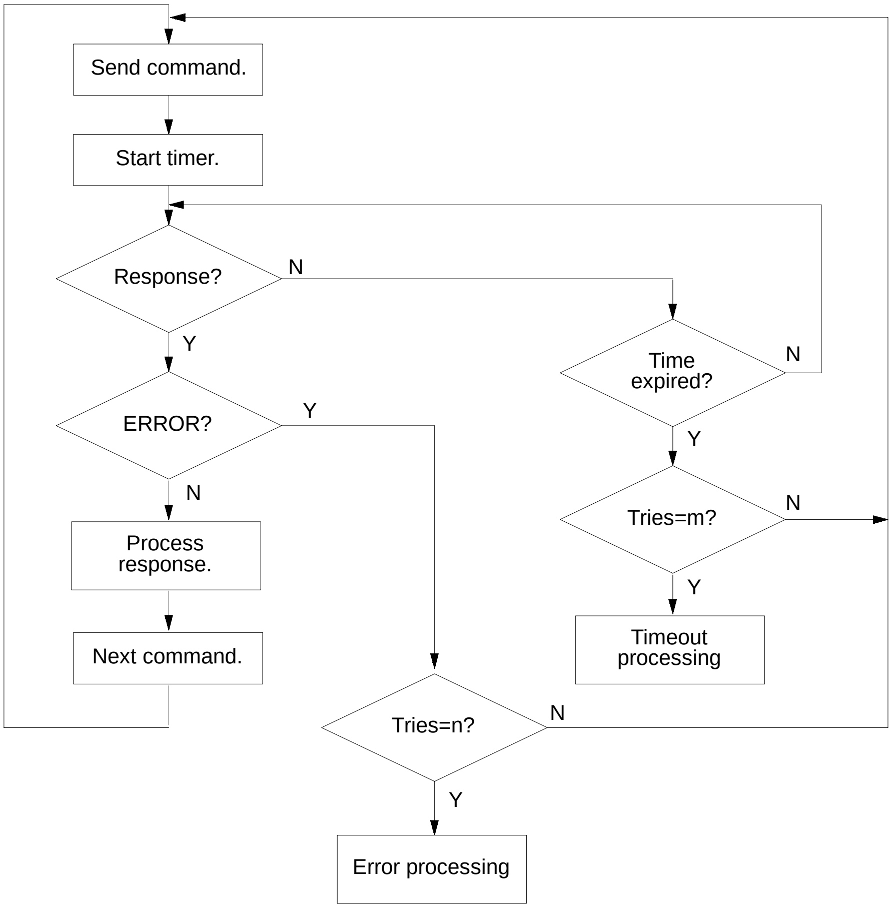
Error and message	FAL no.	Probable cause	Possible correction
Power interruption No message	None	Power has been interrupted for at least 10 ms.	Check power supply voltage and power lines. Try to power-up again.
CPU error No message	None	Watchdog timer has exceeded maximum setting (default setting: 130 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	Memory Unit is incorrectly mounted or missing, checksum error has occurred, or incorrect instruction exists.	Check Memory Unit to make sure it is mounted and backed up properly. Perform a Program Check Operation to locate cause of error. If error not correctable, try inputting program again.
No END(01) instruction NO END INST	F0	END(01) is not written anywhere in program.	Write END(01) at the last address of the program.
I/O Unit over I/O UNIT OVER	E1	The I/O capacity of the PC has been exceeded.	Reduce the number of I/O points and restart the PC.
I/O bus error I/O BUS ERR Rack no.	C0 to C3	Error has occurred in the bus line between the Units.	The rightmost digit of the FAL number will indicate the number of the Unit where the error was detected. Check cable connections between Units.
FALS error SYS FAIL FAL**	01 to 99	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 too long.
	9F	The cycle time is over 120 ms (or 130 ms to 6.5 s if watchdog timer instruction is used.)	9F will be output when FALS(07) is executed and the cycle time is over 120 ms. Check the program.

9-5 Error History Function

If the Error History Enable Bit (AR 0715) has been turned ON, then, when an error occurs, the FAL no., the date, and the time (in hours, minutes, and seconds, i.e., the values stored in AR 18 and AR 19) are sent to the System DM. For details on how this function operates, refer to *3-6 DM Area*.

9-6 Host Link Error Processing

This section describes errors that can occur in a computer-linked system employing the RS-232C interface, including errors processed by the host computer (see *9-6-1 Error Control* and *8-4 Host Link Communications Protocol*).



Programs to monitor communication time and error handling need to be developed on the host computer. Be sure that these include processes that can respond appropriately to errors and other abnormalities from the PC by taking into consideration the kinds of errors described in the sections listed above.

9-6-1 Error Control

The host computer is responsible for ensuring system recovery after errors occur in the Host Link System.

The host link interface runs the following checks to detect errors:

1, 2, 3...

1. Parity check
2. Framing check
3. Overrun check
4. Format check
5. Entry data check (The start word, read word, etc., in the command format.)
6. FCS (An Exclusive OR check is performed on all command or response data, from the unit number to the end of the text.)

Of the above commands, 1 to 3 are performed on a character by character basis. Checks 4 to 6, however, are performed on each block (frame).

Transmit data in a multiple-link system is checked by means of a parity check and a Frame Check Sequence (FCS). The FCS check is not performed in single-link systems.

9-6-2 Invalid Processing

If the host link interface detects an error in a single-frame command or the first frame of a command block, it will regard the command as invalid. The command will not be processed and, after the terminator is received, an error response will be sent to the host computer.

9-6-3 Process Interruption

If the host link interface detects an error in an intermediate frame, the commands up to that point will be processed normally. Those following the erroneous frame, however, will not be processed. After the host link interface has received the terminator of the erroneous block, it responds with a response code that informs the host computer of the process interruption.

9-6-4 Time Monitoring

If the host link interface does not receive a delimiter or terminator, it cannot send a response to the host computer. Similarly, if the computer does not receive a delimiter or terminator, it cannot transmit further commands to the host link interface. To allow transmission to alternate smoothly between the computer and the host link interface, the process times need to be monitored. It is therefore necessary to have a time-monitoring program on the host computer side. Its purpose is to initiate remedial action if the right to transmit is not transferred quickly enough.

9-6-5 Retries

An error response will be returned to the originating device if the host link interface detects any communications line data that has been destroyed (e.g., by noise). If, however, the Unit number has also been lost, no response will be made at all. It is therefore necessary to have response monitoring and retry processing in the host computer to check for error responses.

Appendix A
Standard Models

CPUs

Name	Power supply	Inputs	Outputs	Memory*	Clock	Model number
C20H	24 VDC	12, 24-VDC inputs 2 commons One with 2 pts. One with 10 pts.	8 relay outputs with sockets 5 commons 4 with 1 pt. each 1 with 4 pts.	RAM	None	C20H-C1DR-DE-V1
				IC socket		C20H-C2DR-DE-V1
				EEPROM		C20H-C3DR-DE-V1
				RAM	Built in	C20H-C5DR-DE-V1
				IC socket		C20H-C6DR-DE-V1
				EEPROM		C20H-C7DR-DE-V1
			8 transistor outputs without sockets 5 commons 4 with 1 pt. each 1 with 4 pts.	RAM	None	C20H-C1DT-DE-V1
				IC socket		C20H-C2DT-DE-V1
				EEPROM		C20H-C3DT-DE-V1
				RAM	Built in	C20H-C5DT-DE-V1
				IC socket		C20H-C6DT-DE-V1
				EEPROM		C20H-C7DT-DE-V1
C28H	24 VDC	16, 24-VDC inputs 2 commons One with 2 pts. One with 14 pts.	12 relay outputs with sockets 6 commons 4 with 1 pt. each 2 with 4 pts.	RAM	None	C28H-C1DR-DE-V1
				IC socket		C28H-C2DR-DE-V1
				EEPROM		C28H-C3DR-DE-V1
				RAM	Built in	C28H-C5DR-DE-V1
				IC socket		C28H-C6DR-DE-V1
				EEPROM		C28H-C7DR-DE-V1
			12 transistor outputs without sockets 6 commons 4 with 1 pt. each 2 with 4 pts.	RAM	None	C28H-C1DT-DE-V1
				IC socket		C28H-C2DT-DE-V1
				EEPROM		C28H-C3DT-DE-V1
				RAM	Built in	C28H-C5DT-DE-V1
				IC socket		C28H-C6DT-DE-V1
				EEPROM		C28H-C7DT-DE-V1
C40H	24 VDC	24, 24-VDC inputs 3 commons One with 2 pts. One with 8 pts. One with 14 pts.	16 relay outputs with sockets 7 commons 4 with 1 pt. each 3 with 4 pts.	RAM	None	C40H-C1DR-DE-V1
				IC socket		C40H-C2DR-DE-V1
				EEPROM		C40H-C3DR-DE-V1
				RAM	Built in	C40H-C5DR-DE-V1
				IC socket		C40H-C6DR-DE-V1
				EEPROM		C40H-C7DR-DE-V1
			16 transistor outputs without sockets 7 commons 4 with 1 pt. each 3 with 4 pts.	RAM	None	C40H-C1DT-DE-V1
				IC socket		C40H-C2DT-DE-V1
				EEPROM		C40H-C3DT-DE-V1
				RAM	Built in	C40H-C5DT-DE-V1
				IC socket		C40H-C6DT-DE-V1
				EEPROM		C40H-C7DT-DE-V1
C60H	24 VDC	32, 24-VDC inputs 3 commons One with 2 pts. One with 14 pts. One with 16 pts.	28 relay outputs with sockets 8 commons 4 with 1 pt. each 2 with 4 pts. 2 with 8 pts.	RAM	None	C60H-C1DR-DE-V1
				IC socket		C60H-C2DR-DE-V1
				EEPROM		C60H-C3DR-DE-V1
				RAM	Built in	C60H-C5DR-DE-V1
				IC socket		C60H-C6DR-DE-V1
				EEPROM		C60H-C7DR-DE-V1
			28 transistor outputs without sockets 8 commons 4 with 1 pt. each 2 with 4 pts. 2 with 8 pts	RAM	None	C60H-C1DT-DE-V1
				IC socket		C60H-C2DT-DE-V1
				EEPROM		C60H-C3DT-DE-V1
				RAM	Built in	C60H-C5DT-DE-V1
				IC socket		C60H-C6DT-DE-V1
				EEPROM		C60H-C7DT-DE-V1

*CPUs with IC sockets are not provided with Memory Chips; one of those listed in the following table must be ordered separately. The memory in other types of CPUs cannot be changed.

Memory Chips

Memory chips can be set on on CPU models equipped with I/O sockets.

Name	Specifications	Model number
EPROM Chip	2764, 200 ns, write voltage: 21 V	ROM-HD
	2764, 200 ns, write voltage: 12.5 V	ROM-HB-B
	27128, 200 ns, write voltage: 12.5 V	ROM-IB-B
RAM Chip	6264, 150 ns	RAM-H
EEPROM Chip	28C64, 200 ns	EEPROM-H

Note: Only 27128 EPROM chips (ROM-I_-B) can be used if writing via the FIT's PROM writer or via the C500-PRW06 connected to the GPC.

I/O Units

Name	Power supply	Inputs	Outputs	Accessories	Model number
C20H	24 VDC	12, 24-VDC inputs 2 commons One with 2 pts. One with 10 pts.	8 relay outputs with sockets 5 commons 4 with 1 pt. each 1 with 4 pts.	C20H-CN311 Connecting Cable included.	C20H-EDR-D
			8 transistor outputs without sockets 5 commons 4 with 1 pt. each 1 with 4 pts.		C20H-EDT-D
C28H	24 VDC	16, 24-VDC inputs 2 commons One with 2 pts. One with 14 pts.	12 relay outputs with sockets 6 commons 4 with 1 pt. each 2 with 4 pts.	C20H-CN311 Connecting Cable included.	C28H-EDR-D
			12 transistor outputs without sockets 6 commons 4 with 1 pt. each 2 with 4 pts.		C28H-EDT-D
C40H	24 VDC	24, 24-VDC inputs 3 commons One with 2 pts. One with 8 pts. One with 14 pts.	16 relay outputs with sockets 7 commons 4 with 1 pt. each 3 with 4 pts.	C20H-CN311 Connecting Cable included.	C40H-EDR-D
			16 transistor outputs without sockets 7 commons 4 with 1 pt. each 3 with 4 pts.		C40H-EDT-D
C60H	24 VDC	32, 24-VDC inputs 3 commons One with 2 pts. One with 14 pts. One with 16 pts.	28 relay outputs with sockets 8 commons 4 with 1 pt. each 2 with 4 pts. 2 with 8 pts.	C20H-CN311 Connecting Cable included.	C60H-EDR-D
			28 transistor outputs without sockets 8 commons 4 with 1 pt. each 2 with 4 pts. 2 with 8 pts.		C60H-EDT-D

Connecting Cables

Name	Specifications		Model number
Connecting Cable*	Cable length: 30 cm	To connect CPUs and I/O Units.	C20H-CN311
	Cable length: 60 cm		C20H-CN611
	Cable length: 1 m		C20H-CN121
	Cable length: 2 m		C20H-CN221

*Total length must be 6m or less.

DIN Products

Product	Specifications	Model No.
DIN Track	Length: 50 cm; height: 7.3 mm	PFP-50N
	Length: 1 mm; height: 7.3 mm	PFP-100N
	Length: 1 m; height: 16 mm	PFP-100N2
End Plate	-	PFP-M
Spacer	-	PFP-S

Peripheral Devices

Product	Description		Model No.
Programming Console	Vertical, w/backlight Connecting cable required; sold separately		C200H-PR027-E
Data Access Console	Vertical, w/backlight Connecting cable required; sold separately		C200H-DAC01
Programming and Data Access Console Connecting Cable	For Handheld console, 2 m		C200H-CN222
	For Handheld console, 4 m		C200H-CN422
Panel Mounting Bracket	For Handheld Programming Console or Data Access Console		C200H-ATT01
Cassette Tape Recorder Connecting Cable*	1 m		SCYP0R-PLG01
PROM Writer*	For C-series PCs (12.5/21 V)		C500-PRW06
Floppy Disk Interface Unit*	For C-series PCs		3G2C5-FDI03
Printer Interface Unit*	For C-series PCs		3G2A5-PRT01-E
Memory Pack (for Printer Interface)	With comment printing function		C2000-MP103-EV3
	M1R/M5R		3G2A5-MP003-E
	POR		3G2A5-MP004-E
	S6		3G2A5-MP005-E
	V8		3G2A5-MP006-E
Printer Connecting Cable	2 m (also used for X-Y plotter)		SCY-CN201
CPU-mounting Host Link Unit	Connects to optical fiber cable (APF/PCF)		3G2A6-LK101-PEV1
	Connects to optical fiber cable (PCF)		3G2A6-LK101-EV1
	RS-232C		3G2A6-LK201-EV1
	RS-422		3G2A6-LK202-EV1
Peripheral Interface Unit	For connecting PC to GPC or FIT		C200H-IP004**
	Connecting cable required; sold separately		C200H-IP006
Connecting Cable	To connect GPC to Peripheral Interface Unit	2 m	3G2A2-CN221
		5 m	C500-CN523
		10 m	C500-CN131
		20 m	C500-CN 231
		30 m	C500-CN331

Product	Description		Model No.
		40 m	C500-CN431
		50 m	C500-CN531
Graphic Programming Console	100 to 120 VAC, w/comment		3G2C5-GPC03-E
	200 to 240 VAC, w/comment		3G2C5-GPC04-E
CRT Interface Unit*	For connection between GPC and CRT		3G2C5-GDI01
Programming Console Base Unit	Required to mount 3G2A5-IP004/ 3G2A6-LK___-(P)EV1 to C20H or C28H CPU.	30 m	C200H-BP001
		50 m	C200H-BP002
FIT	Factory Intelligent Terminal		FIT 10-SET11-E

*Used with GPC only.

**Programming Console Base Unit required to mount to C20H or C28H.

Graphic Programming Console (GPC)

Name	Specifications		Model number
GPC (LCD display)	32 kW, with comments	100 VAC	3G2C5-GPC03-E
		200 VAC	3G2C5-GPC04-E
GPC Carrying Case	With side pocket		C500-CS001
GPC System Memory Cassette	For C20H, C40H, C200H, C1000H, C2000H; printing capability for DM and FM lists.		3G2C5-MP304-EV3

Factory Intelligent Terminal (FIT)

Name	Specifications	Model number
FIT	1. FIT Computer (FIT10-CPU01) 2. SYSMATE Ladder Pack (2 system disks, 1 data disk) (FIT10-MF101-EV4) 3. MS-DOS 4. GPC Communications Adapter (C500-IF001) 5. Peripheral Connecting Cable (3G2A2-CN221) 6. Power Cord and 3-pin or 2-pin plug 7. Carrying Case	FIT10-SET11-E

Ladder Support Software (LSS)

Name	Specifications	Model number
Ladder Support Software	5.25”, 2D	C500-SF711-EV3
	3.5”, 2DD	C500-SF312-EV3

Appendix B
Programming Instructions

This appendix provides tables listing the full range of ladder diagram programming instructions used with the C20H/C28H/C40H/C60H. the first table summarizes all instructions and gives page references where more detailed information can be found in the body of the manual. The second table gives the execution times for the instructions for both ON and OFF execution conditions. The third part is divided into two tables and summarizes the instructions, giving the ladder diagram symbol, a brief description, and the applicable data areas. In all tables, the entries are listed alphanumerically. Instructions without function codes are given first in alphabetical order, according to the mnemonic. These are followed by the instructions with function codes which are listed numerically, according to the function code.

Standard Instructions

The following table lists the standard programming instructions available for C-series PCs. A PC instruction is entered either using the appropriate Programming Console key(s) (e.g., LD, AND, OR, NOT), or by using function codes. To input an instruction using its function code, press FUN, the function code, and then WRITE.

Function Code	Name	Mnemonic	Page
––	AND	AND	50, 100
––	AND LOAD	AND LD	53, 101
––	AND NOT	AND NOT	50, 100
––	COUNTER	CNT	117
––	LOAD	LD	50, 100
––	LOAD NOT	LD NOT	50, 100
––	OR	OR	51, 100
––	OR LOAD	OR LD	54, 101
––	OR NOT	OR NOT	51, 100
––	OUTPUT	OUT	52, 103
––	OUTPUT NOT	OUT NOT	52, 103
––	TIMER	TIM	112
00	NO OPERATION	NOP	---
01	END	END	52, 111
02	INTERLOCK	IL	84, 108
03	INTERLOCK CLEAR	ILC	84, 108
04	JUMP	JMP	86, 110
05	JUMP END	JME	86, 110
06	FAILURE ALARM	FAL	197
07	SEVERE FAILURE ALARM	FALS	197
08	STEP DEFINE	STEP	189
09	STEP START	SNXT	189
10	SHIFT REGISTER	SFT	129
11	KEEP	KEEP	106
12	REVERSIBLE COUNTER	CNTR	120
13	DIFFERENTIATE UP	DIFU	88, 104
14	DIFFERENTIATE DOWN	DIFD	88, 104
15	HIGH-SPEED TIMER	TIMH	116
16	WORD SHIFT	WSFT	131
17	REVERSIBLE WORD SHIFT	RWS	132
18	CYCLE TIME	SCAN	198

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 24615	24700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	Read/Wr: DM 0000 to DM 0999 Rd only: DM 1000 to DM 1999	0000 to 9999 or 0000 to FFFF

Function Code	Name	Mnemonic	Page
20	COMPARE	CMP	145
21	MOVE	MOV	139
22	MOVE NOT	MVN	139
23	BCD-TO-BINARY	BIN	151
24	BINARY-TO-BCD	BCD	151
25	ARITHMETIC SHIFT LEFT	ASL	134
26	ARITHMETIC SHIFT RIGHT	ASR	134
27	ROTATE LEFT	ROL	135
28	ROTATE RIGHT	ROR	135
29	COMPLEMENT	COM	182
30	BCD ADD	ADD	164
31	BCD SUBTRACT	SUB	165
32	BCD MULTIPLY	MUL	168
33	BCD DIVIDE	DIV	172
34	LOGICAL AND	ANDW	183
35	LOGICAL OR	ORW	183
36	EXCLUSIVE OR	XORW	184
37	EXCLUSIVE NOR	XNRW	185
38	INCREMENT	INC	162
39	DECREMENT	DEC	163
40	SET CARRY	STC	163
41	CLEAR CARRY	CLC	163
46	DISPLAY MESSAGE	MSG	199
47	LONG MESSAGE	LMSG	200
49	SET SYSTEM	SYS	202
50	BINARY ADD	ADB	177
51	BINARY SUBTRACT	SBB	179
52	BINARY MULTIPLY	MLB	181
53	BINARY DIVIDE	DVB	182
60	REVERSIBLE DRUM COUNTER	RDM	121
61	HIGH-SPEED COUNTER	HDM	124
62	KEY INPUT	KEY	63, 204
63	RS-232C PORT OUTPUT	POUT	205
64	RS-232C PORT INPUT	PIN	216
65	HOURS-TO-SECONDS	HTS	152
66	SECONDS-TO-HOURS	STH	153
67	BIT COUNTER	BCNT	218
68	BLOCK COMPARE	BCMP	148
69	HEXADECIMAL CONVERT	HEX	154
70	BLOCK TRANSFER	XFER	140
71	BLOCK SET	BSET	141
73	DATA EXCHANGE	XCHG	142
74	ONE DIGIT SHIFT LEFT	SLD	136
75	ONE DIGIT SHIFT RIGHT	SRD	136
76	4-TO-16 DECODER	MLPX	156
77	16-TO-4 ENCODER	DMPX	158
82	MOVE BIT	MOVB	143
83	MOVE DIGIT	MOVD	143
84	REVERSIBLE SHIFT REGISTER	SFTR	137

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 24615	24700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	Read/Wr: DM 0000 to DM 0999 Rd only: DM 1000 to DM 1999	0000 to 9999 or 0000 to FFFF

Function Code	Name	Mnemonic	Page
86	ASCII CONVERT	ASC	160
91	SUBROUTINE ENTER	SBS	186
92	SUBROUTINE START	SBN	186
93	RETURN	RET	186
94	WATCHDOG TIMER REFRESH	WDT	218
97	I/O REFRESH	IORF	219

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 24615	24700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	Read/Wr: DM 0000 to DM 0999 Rd only: DM 1000 to DM 1999	0000 to 9999 or 0000 to FFFF

Table: Instruction Execution Times

The execution time is given in microseconds unless otherwise stated.

Instruction	Conditions	ON execution time	OFF execution time	Page
LD	---	0.75	1.5	50, 100
LD NOT	---	0.75	1.5	50, 100
AND	---	0.75	1.5	50, 100
AND NOT	---	0.75	1.5	50, 100
OR	---	0.75	1.5	50, 100
OR NOT	---	0.75	1.5	50, 100
AND LD	---	0.75	1.5	53, 101
OR LD	---	0.75	1.5	54, 101
OUT	---	1.13	2.25	52, 103
OUT NOT	---	1.13	2.25	52, 103
TIM	Constant for SV	2.25	R: 2.25 IL: 2.25 JMP: 2.25	112
	*DM for SV		R: 259 IL: 2.25 JMP: 2.25	
CNT	Constant for SV	2.25	R: 2.25 IL: 2.25 JMP: 2.25	117
	*DM for SV		R: 255 IL: 2.25 JMP: 2.25	
NOP(00)	---	0.75	---	---
END(01)	---	85	---	52, 111
IL(02)	---	32	35	84, 108
ILC(03)	---	59	35	84, 108
JMP(04)	---	35	35	86, 110
JME(05)	---	45	35	86, 110
FAL(06)	FAL(06) 00 (reset)	357	2.25	197
	FAL(06) 01 to 99	247	2.25	
FALS(07)	---	11.1 ms	2.25	197
STEP(08)	---	364	2.25	189
SNXT(09)	---	22	2.25	189
SFT(10)	With 1-word shift register	227	R: 191 IL: 30 JMP: 30	129
	With 250-word shift register	8.06 ms	R: 1.81 ms IL: 30 JMP: 30	
KEEP(11)	---	1.13	---	106
CNTR(12)	Constant for SV	107	R: 85 IL: 49 JMP: 49	120
	*DM for SV	265		

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 24615	24700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	Read/Wr: DM 0000 to DM 0999 Rd only: DM 1000 to DM 1999	0000 to 9999 or 0000 to FFFF

Instruction	Conditions	ON execution time	OFF execution time	Page
DIFU(13)	---	105	Normal: 93 IL: 93 JMP: 84	88, 104
DIFD(14)	---	104	Normal: 92 IL: 92 JMP: 84	88, 104
TIMH(15)	Interrupt, Constant for SV	149	R: 199 IL: 199 JMP: 73	116
	Normal cycle, Constant for SV	169		
	Interrupt, *DM for SV	149	R: 291 IL: 291 JMP: 73	
	Normal cycle, *DM for SV	169		
WSFT(16)	When shifting 1 word	260	3	131
	When shifting 1000 words using *DM	17.3 ms		
RWS(17)	When shifting 1 word	558	3.75	132
	When shifting 1000 words using *DM	57.4 ms		
SCAN(18)	---	Actual cycle time	3.75	198
CMP(20)	When comparing a constant to a word	162	3	145
	When comparing two *DM	447		
MOV(21)	When transferring a constant to a word	113	3	139
	When transferring *DM to *DM	321		
MVN(22)	When transferring a constant to a word	115	3	139
	When transferring *DM to *DM	392		
BIN(23)	When converting a word to a word	197	3	151
	When converting *DM to *DM	465		
BCD(24)	When converting a word to a word	198	3	151
	When converting *DM to *DM	451		
ASL(25)	When shifting a word	62	2.25	134
	When shifting *DM	190		
ASR(26)	When shifting a word	62	2.25	134
	When shifting *DM	190		
ROL(27)	When rotating a word	66	2.25	135
	When rotating *DM	194		
ROR(28)	When rotating a word	66	2.25	135
	When rotating *DM	194		
COM(29)	When inverting a word	379	2.25	182
	When inverting *DM	506		
ADD(30)	Constant + word A word	166	3.75	164
	*DM + *DM A *DM	593		
SUB(31)	Constant + word A word	192	3.75	165
	*DM – *DM A *DM	600		
MUL(32)	Constant x word A word	634	3.75	168
	*DM x *DM A word	1045		
DIV(33)	Word ÷ constant A word	737	3.75	172
	*DM ÷ *DM A *DM	1156		

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 24615	24700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	Read/Wr: DM 0000 to DM 0999 Rd only: DM 1000 to DM 1999	0000 to 9999 or 0000 to FFFF

Instruction	Conditions	ON execution time	OFF execution time	Page
ANDW(34)	Constant < word A word	162	3.75	183
	*DM < *DM A *DM	557		
ORW(35)	Constant > word A word	162	3.75	183
	*DM > *DM A *DM	560		
XORW(36)	Constant XORW word A word	162	3.75	184
	*DM XORW *DM A *DM	560		
XNRW(37)	Constant XNRW word A word	163	3.75	185
	*DM XNRW *DM A *DM	561		
INC(38)	When incrementing a word	79	2.25	162
	When incrementing *DM	207		
DEC(39)	When decrementing a word	72	2.25	163
	When decrementing *DM	260		
STC(40)	---	21	1.5	163
CLC(41)	---	21	1.5	163
MSG(46)	---	88	2.25	199
LMSG(47)	When outputting character string to Programming Console from word	334	3.75	200
	When outputting character string to Programming Console set by *DM	414		
	When outputting character string to RS-232C from word	751		
	When outputting character string to RS-232C set by *DM	1679		
SYS(49)	When using command code 01.	1998	3.75	202
ADB(50)	Constant + word A word	208	3.75	177
	*DM + *DM A *DM	604		
SBB(51)	Constant – word A word	208	3.75	179
	*DM – *DM A *DM	604		
MLB(52)	Constant x word A word	283	3.75	181
	*DM x *DM A *DM	658		
DVB(53)	Word ÷ constant A word	516	3.75	182
	*DM ÷ *DM A *DM	927		
RDM(60)	When comparing 1 range with words	719	719	121
	When comparing max. ranges with *DM	18.0 ms	18.0 ms	
HDM(61)	When comparing 1 range with words	1079	3.75	124
	When comparing max. ranges with *DM	18.5 ms		
KEY(62)	When using words	464	3.75	63, 204
	When using *DM	489		
POUT(63)	When outputting 0 bytes	464	3.75	205
	When outputting 200 bytes of *DM	4.78 ms		
PIN(64)	When inputting 0 bytes	646	3.75	216
	When inputting 200 bytes of *DM	4.95 ms		
HTS(65)	Word conversion	468	3.75	152
	*DM conversion (converting max. time)	60.6 ms		
STH(66)	Word conversion	572	3.75	153
	*DM conversion (converting max. seconds)	175.3 ms		

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 24615	24700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	Read/Wr: DM 0000 to DM 0999 Rd only: DM 1000 to DM 1999	0000 to 9999 or 0000 to FFFF

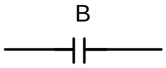
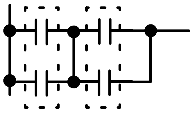
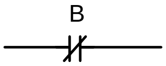
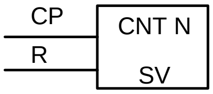
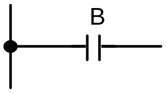
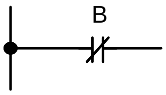
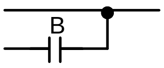
Instruction	Conditions	ON execution time	OFF execution time	Page
BCNT(67)	When counting 1 word	531	3.75	218
	When counting 1000 words using *DM	253.6 ms		
BCMP(68)	Comparing constant to word-designated table	823	3.75	148
	Comparing *DM A *DM-designated table	17 ms		
HEX(69)	Converting word to word	433	3.75	154
	Converting *DM to *DM	1.20 ms		
XFER(70)	When transferring 1 word	433	3.75	140
	When transferring 1000 words using *DM	47.1 ms		
BSET(71)	When setting a constant to 1 word	280	3.75	141
	When setting *DM ms to 1,000 words using *DM	1.56 ms		
XCHG(73)	Between words	215	3	142
	Between *DM	408		
SLD(74)	Shifting 1 word	211	3	136
	Shifting 1,000 words using *DM	25.3 ms		
SRD(75)	Shifting 1 word	208	3	136
	Shifting 1,000 words using *DM	25.3 ms		
MLPX(76)	When decoding word to word	337	3.75	156
	When decoding *DM to *DM	708		
DMPX(77)	When encoding a word to a word	404	3.75	158
	When encoding *DM to *DM	758		
MOVB (82)	When transferring word to a word	172	3.75	143
	When transferring *DM to *DM	557		
MOVD(83)	When transferring word to a word	210	3.75	143
	When transferring *DM to *DM	459		
SFTR(84)	When shifting 1 word	475	3.75	137
	When shifting 1000 DM words using *DM	18.7 ms		
ASC(86)	Word A word	385	3.75	160
	*DM A *DM	746		
SBS(91)	---	320	2.25	186
SBN(92)	---	---	---	186
RET(93)	---	207	1.5	186
WDT(94)	---	27	2.25	218
IORF(97)	1-word refresh	675	3	219
	30-word refresh	4 ms		

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 24615	24700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	Read/Wr: DM 0000 to DM 0999 Rd only: DM 1000 to DM 1999	0000 to 9999 or 0000 to FFFF

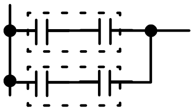
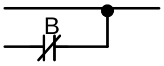
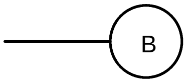
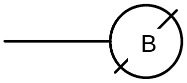
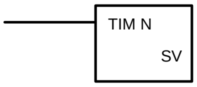
Basic Instructions

Name Mnemonic	Symbol	Function	Operand Data Areas	Page
AND AND		Logically ANDs the status of the designated bit with the current execution condition.	**B:** IR SR HR AR LR TC	50, 100
AND LOAD AND LD		Logically ANDs the resultant execution conditions of the preceding logic blocks.	None	53, 101
AND NOT AND NOT		Logically ANDs the inverse of the designated bit with the current execution condition.	**B:** IR SR HR AR LR TC	50, 100
COUNTER CNT		A decrementing counter. SV: 0 to 9999; CP: count pulse; R: reset input. The TC bit is entered as a constant.	**N:** TC **SV:** IR HR AR LR DM #	117
LOAD LD		Defines the status of bit B as the execution condition for subsequent operations in the instruction line.	**B:** IR SR HR AR LR TC TR	50, 100
LOAD NOT LD NOT		Defines the status of the inverse of bit B as the execution condition for subsequent operations in the instruction line.	**B:** IR SR HR AR LR TC	50, 100
OR OR		Logically ORs the status of the designated bit with the current execution condition.	**B**: IR SR HR AR LR TC	51, 100

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 24615	24700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	Read/Wr: DM 0000 to DM 0999 Rd only: DM 1000 to DM 1999	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas	Page
OR LOAD OR LD		Logically ORs the resultant execution conditions of the preceding logic blocks.	None	54, 101
OR NOT OR NOT		Logically ORs the inverse of the designated bit with the execution condition.	**B:** IR SR HR AR LR TC	51, 100
OUTPUT OUT		Turns ON B for an ON execution condition; turns OFF B for an OFF execution condition.	**B:** IR SR HR AR LR TR	52, 103
OUTPUT NOT OUT NOT		Turns OFF B for an ON execution condition; turns ON B for an OFF execution condition.	**B:** IR SR HR AR LR	52, 103
TIMER TIM		ON-delay (decrementing) timer operation. Set value: 000.0 to 999.9 s. The same TC bit cannot be assigned to more than one timer/counter. The TC bit is entered as a constant.	**N:** TC **SV**: IR HR AR LR DM #	112

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 24615	24700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	Read/Wr: DM 0000 to DM 0999 Rd only: DM 1000 to DM 1999	0000 to 9999 or 0000 to FFFF

Special Instructions

Name Mnemonic	Symbol	Function	Operand Data Areas	Page
NO OPERATION NOP(00)	None	Nothing is executed and program operation moves to the next instruction.	None	---

END
END(01)
	END(01)	Required at the end of each program. Instructions located after END(01) will not be executed.	None	52,
				111

INTERLOCK
IL(02)
INTERLOCK CLEAR
ILC(03)
	IL(02)
	ILC(03)
		If an interlock condition is OFF, all outputs and all timer PVs between the current IL(02) and the next ILC(03) are turned OFF or reset, respectively. Other instructions are treated as NOP. Counter PVs are maintained. If the execution condition is ON, execution continues normally.	None	84,
				108

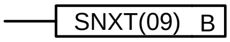
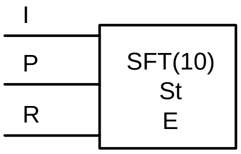
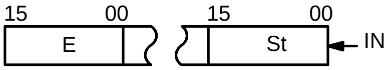
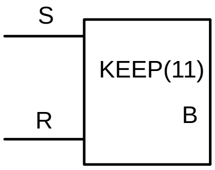
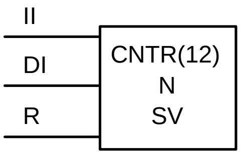
JUMP
JMP(04)
JUMP END
JME(05)
	JMP(04) N
	JME(05) N
		When the execution condition for the JMP(04) instruction is ON, all instructions between JMP(04) and the corresponding JME(05) are to be ignored or treated as NOP(00). For direct jumps, the corresponding JMP(04) and JME(05) instructions have the same N value in the range 01 through 49. Direct jumps are usable only once each per program (i.e., N is 01 through 49 can be used only once each) and the instructions between the JUMP and JUMP END instructions are ignored; 00 may be used as many times as necessary, instructions between JMP 00 and the next JME 00 are treated as NOP, thus increasing cycle time, as compared with direct jumps.	**N:**
			00 to 49
				86,
				110

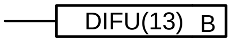
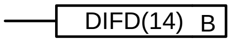
FAILURE ALARM (@)FAL(06)	FAL(06) N	Assigns a failure alarm code to the given execution condition.When N can be given a value between 01 and 99 to indicate that a non-fatal error (i.e., one that will not stop the CPU) has occurred. This is indicated by the PLC outputting N (the FAL number) to the FAL output area. To reset the FAL area, N can be defined as 00. This will cause all previously recorded FAL numbers in the FAL area to be deleted. FAL data sent after a 00 will be recorded in the normal way. The same code numbers can be used for both FAL(06) and FALS(07).	**N:** 00 to 99	197
SEVERE FAILURE ALARM FALS(07)	FALS(07) N	A fatal error is indicated by outputting N to the FAL output area and the CPU is stopped. The same FAL numbers are used for both FAL(06) and FALS(07).	**N:** 01 to 99	197
STEP DEFINE STEP(08)	STEP(08)B	When used with a control bit (B), defines the start of a new step and resets the previous step. When used without B, it defines the end of step execution.	**B:** IR HR AR LR	189

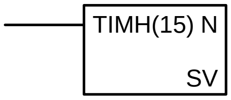
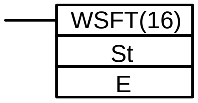
Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 24615	24700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	Read/Wr: DM 0000 to DM 0999 Rd only: DM 1000 to DM 1999	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas	Page
STEP START SNXT(09)		Used with a control bit (B) to indicate the end of the step, reset the step, and start the next step which has been defined with the same control bit.	**B:** IR HR AR LR	189
SHIFT REGISTER SFT(10)		Creates a bit shift register for data from the starting word (St) through to the ending word (E). I: input bit; P: shift pulse; R: reset input. St must be less than or equal to E. St and E must be in the same data area. 	**St/E:** IR HR AR LR	129
KEEP KEEP(11)		Defines a bit (B) as a latch, controlled by the set (S) and reset (R) inputs.	**B:** IR HR AR LR	106
REVERSIBLE COUNTER CNTR (12)		Increases or decreases the PV by one whenever the increment input (II) or decrement input (DI) signals, respectively, go from OFF to ON. SV: 0 to 9999; R: reset input. Each TC bit can be used for one timer/ counter only. The TC bit is entered as a constant.	**N:** TC **SV:** IR SR HR AR LR DM #	120

DIFFERENTIATE UP
DIFU(13)
DIFFERENTIATE DOWN
DIFD(14)
	
	
		DIFU(13) turns ON the designated bit (B) for one cycle on reception of the leading (rising) edge of the input signal; DIFD(14) turns ON the bit for one cycle on reception of the trailing (falling) edge.	**B:**
			IR
			HR
			AR
			LR
				88,
				104

HIGH-SPEED TIMER TIMH(15)		A high-speed, ON-delay (decrementing) timer. SV: 00.02 to 99.99 s. Each TC bit can be assigned to only one timer or counter. The TC bit is entered as a constant.	**N:** TC **SV**: IR SR HR AR LR HR #	116
WORD SHIFT (@)WSFT(16)		The data in the words from the starting word (St) through to the ending word (E), is shifted left in word units, writing all zeros into the starting word. St must be less than or equal to E, and St and E must be in the same data area.	**St/E:** IR HR AR LR DM	131

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

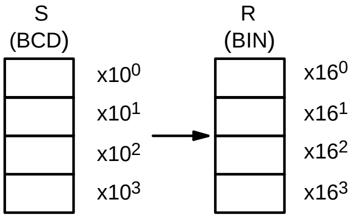
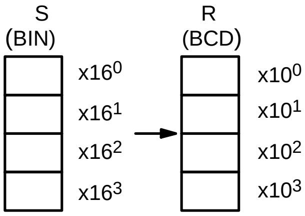
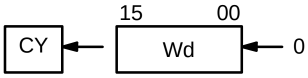
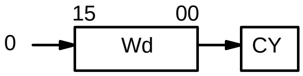
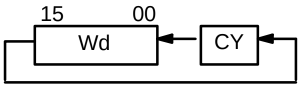
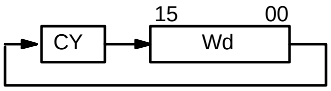
IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 24615	24700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	Read/Wr: DM 0000 to DM 0999 Rd only: DM 1000 to DM 1999	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas	Page
REVERSIBLE WORD SHIFT (@)RWS(17)	RWS(17) C St E	Creates and controls a reversible non-synchronous word shift register between St and E. Exchanges the content of a word containing zero with the content of either the preceding or following word, depending on the shift direction. Bits 13, 14, and 15 of control word C determine the mode of operation of the register according to the following: The shift direction is determined by bit 13 (OFF shifts the non-zero data to higher addressed words; ON to lower addressed words). Bit 14 is the register enable bit (ON for shift enabled). Bit 15 is the reset bit (if bit 15 is ON, the register will be set to zero between St and E when the instruction is executed with bit 14 also ON). St and E must be in the same data area.	**C:** **St/E:** IR IR SR SR HR HR AR AR LR LR TC TC DM DM #	132
CYCLE TIME (@)SCAN(18)	SCAN(18) Mi --- ---	Sets the minimum cycle time, Mi, in tenths of milliseconds. The possible setting range is from 0 to 999.0 ms. If the actual cycle time is less than the time set using SCAN(18), the CPU will wait until the designated time has elapsed before starting the next cycle.	**Mi:** ---: IR Not used. SR HR AR LR TC DM #	198
COMPARE (@)CMP(20)	CMP(20) Cp1 Cp2	Compares the data in two 4-digit hexadecimal words (Cp1 and Cp2) and outputs result to the GR, EQ, or LE Flags.	**Cp1/Cp2:** IR SR HR AR LR TC DM #	145
MOVE (@)MOV(21)	MOV(21) S D	Transfers data from source word, (S) to destination word (D).	**S:** **D:** IR IR SR HR HR AR AR LR LR DM TC DM #	139
MOVE NOT (@)MVN(22)	MVN(22) S D	Transfers the inverse of the data in the source word (S) to destination word (D).	**S:** **D:** IR IR SR HR HR AR AR LR LR DM TC DM #	139

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

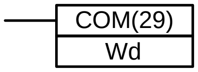
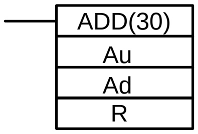
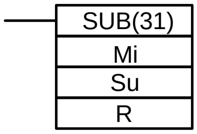
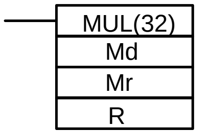
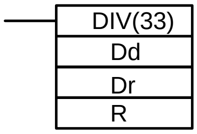
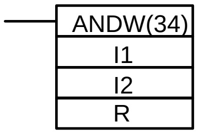
IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 24615	24700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	Read/Wr: DM 0000 to DM 0999 Rd only: DM 1000 to DM 1999	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas	Page
BCD-TO-BINARY (@)BIN(23)	BIN(23) S R	Converts 4-digit, BCD data in source word (S) into 16-bit binary data, and outputs converted data to result word (R).  S (BCD) R (BIN) x10^0 x10^1 x10^2 x10^3 x16^0 x16^1 x16^2 x16^3	**S:** IR SR HR AR LR TC DM **R:** IR HR AR LR DM	151
BINARY-TO-BCD (@)BCD(24)	BCD(24) S R	Converts binary data in source word (S) into BCD, and outputs converted data to result word (R).  S (BIN) R (BCD) x16^0 x16^1 x16^2 x16^3 x10^0 x10^1 x10^2 x10^3	**S:** IR SR HR AR LR DM **R:** IR HR AR LR DM	151
ARITHMETIC SHIFT LEFT (@)ASL(25)	ASL(25) Wd	Each bit within a single word of data (Wd) is shifted one bit to the left, with zero written to bit 00 and bit 15 moving to CY. 	**Wd:** IR HR AR LR DM	134
ARITHMETIC SHIFT RIGHT (@)ASR(26)	ASR(26) Wd	Each bit within a single word of data (Wd) is shifted one bit to the right, with zero written to bit 15 and bit 00 moving to CY. 	**Wd:** IR HR AR LR DM	134
ROTATE LEFT (@)ROL(27)	ROL(27) Wd	Each bit within a single word of data (Wd) is moved one bit to the left, with bit 15 moving to carry (CY), and CY moving to bit 00. 	**Wd:** IR HR AR LR DM	135
ROTATE RIGHT (@)ROR(28)	ROR(28) Wd	Each bit within a single word of data (Wd) is moved one bit to the right, with bit 00 moving to carry (CY), and CY moving to bit 15. 	**Wd:** IR HR AR LR DM	135

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 24615	24700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	Read/Wr: DM 0000 to DM 0999 Rd only: DM 1000 to DM 1999	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas	Page
COMPLEMENT (@)COM(29)		Inverts bit status of one word (Wd) of data, changing 0s to 1s, and vice versa. $\boxed{Wd \rightarrow \overline{Wd}}$	**Wd:** IR HR AR LR DM	182
BCD ADD (@)ADD(30)		Adds two 4-digit BCD values (Au and Ad) and content of CY, and outputs the result to the specified result word (R). Au + Ad + $\boxed{CY} \rightarrow$ R $\boxed{CY}$	**Au/Ad:** IR SR HR AR LR TC DM # **R:** IR HR AR LR DM	164
BCD SUBTRACT (@)SUB(31)		Subtracts both the 4-digit BCD subtrahend (Su) and content of CY, from the 4-digit BCD minuend (Mi) and outputs the result to the specified result word (R). Mi – Su – $\boxed{CY} \rightarrow$ R $\boxed{CY}$	**Mi/Su:** IR SR HR AR LR TC DM # **R:** IR HR AR LR DM	165
BCD MULTIPLY (@)MUL(32)		Multiplies the 4-digit BCD multiplicand (Md) and 4-digit BCD multiplier (Mr), and outputs the result to the specified result words (R and R + 1). R and R + 1 must be in the same data area. Md x Mr $\rightarrow \boxed{R + 1}\ \boxed{R}$	**Md/Mr:** IR SR HR AR LR TC DM # **R:** IR HR AR LR DM	168
BCD DIVIDE (@)DIV(33)		Divides the 4-digit BCD dividend (Dd) by the 4-digit BCD divisor (Dr), and outputs the result to the specified result words. R receives the quotient; R + 1 receives the remainder. R and R + 1 must be in the same data area. Dd ÷ Dr $\rightarrow \boxed{R + 1}\ \boxed{R}$	**Dd/Dr:** IR SR HR AR LR TC DM # **R:** IR HR AR LR DM	172
LOGICAL AND (@)ANDW(34)		Logically ANDs two 16-bit input words (I1 and I2) and sets the bits in the result word (R) if the corresponding bits in the input words are both ON.	**I1/I2:** IR SR HR AR LR TC DM # **R:** IR HR AR LR DM	183

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

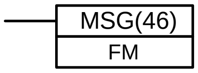
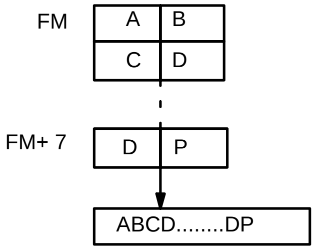
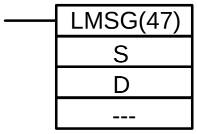
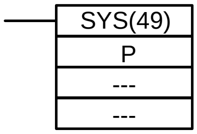
IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 24615	24700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	Read/Wr: DM 0000 to DM 0999 Rd only: DM 1000 to DM 1999	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas	Page
LOGICAL OR (@)ORW(35)	ORW(35) I1 I2 R	Logically ORs two 16-bit input words (I1 and I2) and sets the bits in the result word (R) when one or both of the corresponding bits in the input words is/are ON.	**I1/I2:** IR SR HR AR LR TC DM # **R:** IR HR AR LR DM	183
EXCLUSIVE OR (@)XORW(36)	XORW(36) I1 I2 R	Exclusively ORs two 16-bit input words (I1 and I2) and sets the bits in the result word (R) when the corresponding bits in input words differ in status.	**I1/I2:** IR SR HR AR LR TC DM # **R:** IR HR AR LR DM	184
EXCLUSIVE NOR (@)XNRW(37)	XNRW(37) I1 I2 R	Exclusively NORs two 16-bit input words (I1 and I2) and sets the bits in the result word (R) when the corresponding bits in both input words have the same status.	**I1/I2:** IR SR HR AR LR TC DM # **R:** IR HR AR LR DM	185
INCREMENT (@)INC(38)	INC(38) Wd	Increments the value of a 4-digit BCD word (Wd) by one, without affecting carry (CY).	**Wd:** IR HR AR LR DM	162
DECREMENT (@)DEC(39)	DEC(39) Wd	Decrements the value of a 4-digit BCD word by 1, without affecting carry (CY).	**Wd:** IR HR AR LR DM	163
SET CARRY (@)STC(40)	STC(40)	Sets the Carry Flag (i.e., turns CY ON).	None	163
CLEAR CARRY (@)CLC(41)	CLC(41)	Clears the Carry Flag (i.e, turns CY OFF).	None	163

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 24615	24700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	Read/Wr: DM 0000 to DM 0999 Rd only: DM 1000 to DM 1999	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas	Page
DISPLAY MESSAGE (@)MSG(46)		Displays eight words of ASCII code, starting from FM, on the Programming Console or GPC. All eight words must be in the same data area. 	**FM:** IR HR AR LR TC DM #	199
LONG MESSAGE (@)LMSG(47)		Outputs a 32-character message to either a Programming Console, or a device connected via the RS-232C interface. The output message must be in ASCII beginning at address S. The destination of the message is designated in D: 000 specifies that the message is to be output to the GPC; 001 specifies the RS-232C interface, starting with the leftmost byte; and 002 specifies the RS-232C interface, starting from the rightmost byte.	**S:** IR HR AR LR TC DM **D:** #000 #001 #002 **---:** Not used.	200
SET SYSTEM (@)SYS(49)		Used to either control certain operating parameters, or to execute the system commands that can be executed from the AR area. The contents of the leftmost 8 bits of P determine which function SYS(49) will have. If they contain A3, bit 00 specifies whether the battery will be checked, and bit 07 specifies whether I/O status will be maintained on start up. Bit 06 specifies whether the Force Status Hold Bit is set. To be effective SYS(49) must be programmed at address 00001 with LD AR 1001 at address 00000. If the leftmost 8 bits of P are 00, one of seven possible System Commands, as specified by the rightmost 8 bits, will be executed (this option is not available with C200H PLCs). These commands can be used to set or back-up the contents of the Parameter Area (in the DM area).	**P:** # **---:** Not used.	202

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 24615	24700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	Read/Wr: DM 0000 to DM 0999 Rd only: DM 1000 to DM 1999	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas	Page
BINARY ADD (@)ADB(50)	ADB(50) Au Ad R	Adds the 4-digit augend (Au), 4-digit addend (Ad), and content of CY and outputs the result to the specified result word (R). $\begin{array}{rl} & Au \\ + & Ad \\ + & \boxed{CY} \\ \hline & \boxed{R} \\ & \boxed{CY} \end{array}$	**Au/Ad:** **R:** IR IR SR HR HR AR AR LR LR DM TC DM #	177
BINARY SUBTRACT (@)SBB(51)	SBB(51) Mi Su R	Subtracts the 4-digit hexadecimal subtrahend (Su) and content of carry, from the 4-digit hexadecimal minuend (Mi), and outputs the result to the specified result word (R). $\begin{array}{rl} & Mi \\ - & Su \\ - & \boxed{CY} \\ \hline & \boxed{R} \\ & \boxed{CY} \end{array}$	**Mi/Su:** **R:** IR IR SR HR HR AR AR LR LR DM TC DM #	179
BINARY MULTIPLY (@)MLB(52)	MLB(52) Md Mr R	Multiplies the 4-digit hexadecimal multiplicand (Md) and 4-digit multiplier (Mr), and outputs the 8-digit hexadecimal result to the specified result words (R and R+1). R and R+1 must be in the same data area. $\begin{array}{rl} & Md \\ \mathbf{X} & Mr \\ \hline \text{Quotient} & \boxed{R} \\ \text{Remainder} & \boxed{R+1} \end{array}$	**Md/Mr:** **R:** IR IR SR HR HR AR AR LR LR DM TC DM #	181
BINARY DIVIDE (@)DVB(53)	DVB(53) Dd Dr R	Divides the 4-digit hexadecimal dividend (Dd) by the 4-digit divisor (Dr), and outputs result to the designated result words (R and R + 1). R and R + 1 must be in the same data area. $\begin{array}{rl} & D \\ & d \\ \div & Dr \\ \hline \text{Quotient} & \boxed{R} \\ \text{Remainder} & \boxed{R+ 1} \end{array}$	**Dd/Dr:** **R:** IR IR SR HR HR AR AR LR LR TC DM #	182

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

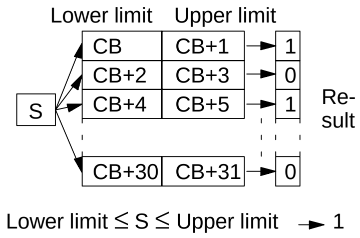
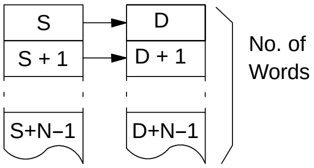
IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 24615	24700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	Read/Wr: DM 0000 to DM 0999 Rd only: DM 1000 to DM 1999	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas			Page
REVERSIBLE DRUM COUNTER (@)RDM(60)	RDM(60) N T R	Creates a reversible ring counter in TC 500 to TC 511 that counts from 0 to 9999, compares the PV to a table of ranges, and turns ON corresponding bits in R whenever the PV is within one of the ranges given in the table. The size of the table is determined by the value of n, specified by bits 00 to 07 of T. The table starting at T must be within one data area, and all table words from T+1 on must be in BCD.	**N:** TC 500 through TC 511	**T:** IR SR HR AR LR TC DM	**R:** IR SR HR AR LR TC DM	121
HIGH-SPEED DRUM COUNTER HDM(61)	HDM(61) T R ---	Compares the PV of the high-speed counter (CNT 511) to a table of ranges, and turns ON corresponding bits in R whenever the PV is within one of the ranges given in the table. The size of the table is determined by the value of n, specified by bits 00 to 07 of T. The table starting at T must be within one data area, and all table words from T+1 on must be in BCD.	**T:** IR SR HR AR LR TC DM	**R:** IR SR HR AR LR TC DM	---: Not used.	124
KEY INPUT (@)KEY(62)	KEY(62) S --- ---	Performs Programming Console operations from within the program. S designates the first word containing a key code. The key codes will produce the same effect as pressing the equivalent Programming Console keys.	**S:** #	---: Not used.		63, 204
RS-232C PORT OUTPUT (@)POUT(63)	POUT(63) S C B	Outputs the B bytes of data in S to S+(B÷2)−1 through the RS-232C port. The control number, C, determines whether the leftmost (C=#0000) or rightmost (C=#0001) bytes in the words will be output first.	**S:** IR SR HR AR LR TC DM	**C:** #	**B:** IR HR AR LR TC DM #	205
RS-232C PORT INPUT (@)PIN(64)	PIN(64) D C B	Writes B bytes of data received through the RS-232C port to words beginning at D. The control number, C, determines whether the leftmost (C=#0000) or rightmost (C=#0001) byte in the words was input first.	**D:** IR HR AR LR TC DM	**C:** #	**B:** IR HR AR LR TC DM #	216
HOURS-TO-SECONDS (@)HTS(65)	HTS(65) S R ---	Converts a time given in hours/minutes/seconds (S and S+1) to an equivalent time in seconds only (R and R+1). S and S+1 must be BCD and within one data area. R and R+1 must also be within one data area.	**S:** IR SR HR AR LR TC DM	**R:** IR SR HR AR LR TC DM	---: Not used.	152
SECONDS-TO-HOURS (@)STH(66)	STH(66) S R ---	Converts a time given in seconds (S and S+1) to an equivalent time in hours/minutes/seconds (R and R+1). S and S+1 must be BCD between 0 and 35,999,999, and within the same data area. R and R+1 must also be within one data area.	**S:** IR SR HR AR LR TC DM	**R:** IR SR HR AR LR TC DM	---: Not used.	153

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

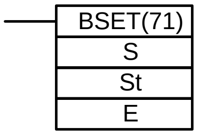
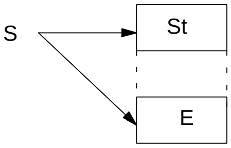
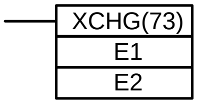
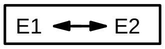
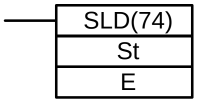
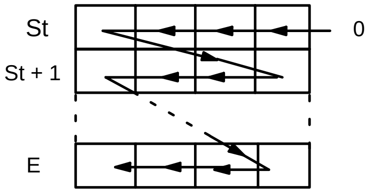
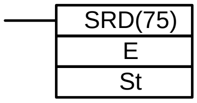
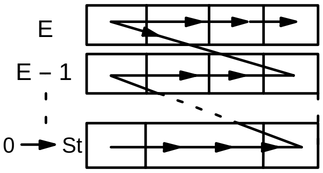
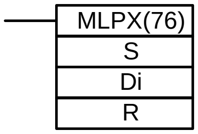
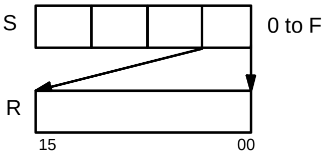
IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 24615	24700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	Read/Wr: DM 0000 to DM 0999 Rd only: DM 1000 to DM 1999	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas			Page
BIT COUNTER (@)BCNT(67)	BCNT(67) N SB R	Counts the number of ON bits in one or more words (SB is the beginning source word) and outputs the result to the specified result word (R). N gives the number of words to be counted. All words in which bit are to be counted must be in the same data area.	**N:** IR SR HR AR LR TC DM	**R:** IR HR AR LR TC DM	**SB:** IR SR HR AR LR TC DM	218
BLOCK COMPARE (@)BCMP(68)	BCMP(68) S CB R	Compares a 1-word binary value (S) with the 16 ranges given in the comparison table (CB is the starting word of the comparison block). If the value falls within any of the ranges, the corresponding bits in the result word (R) will be set. The comparison block must be within one data area.  Lower limit Upper limit CB CB+1 CB+2 CB+3 CB+4 CB+5 CB+30 CB+31 S 1 0 1 0 Re-sult Lower limit ≤ S ≤ Upper limit → 1	**S:** IR SR HR AR LR TC DM #	**CB:** IR SR HR LR TC DM	**R:** IR HR AR LR TC DM	148
HEXADECIMAL CONVERT (@)HEX(69)	HEX(69) S Di D	Converts the ASCII codes in words beginning with S according to specifications in Di and outputs the hexadecimal equivalents to words beginning with D. Di specifications: Digit 0: first digit to convert; digit 1: number of ASCII codes to convert; digit 2: first half of S to convert; and digit 3: parity (0: none; 1: even; 2: odd).	**S:** IR SR HR AR LR TC DM	**Di:** IR HR AR LR TC DM #	**D:** IR HR AR LR DM	154
BLOCK TRANSFER (@)XFER(70)	XFER(70) N S D	Moves the content of several consecutive source words (S gives the address of the starting source word) to consecutive destination words (D is the starting destination word). All source words must be in the same data area, as must all destination words. Transfers can be within one data area or between two data areas, but the source and destination words must not overlap.  	**N:** IR SR HR AR LR TC DM #	**S :** IR HR AR LR TC DM	**D:** IR SR HR AR LR TC DM #	140

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

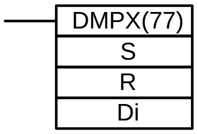
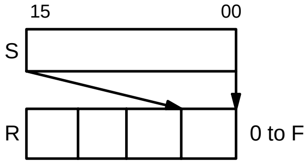
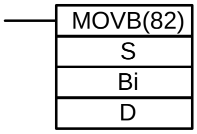
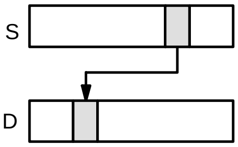
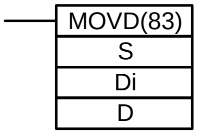
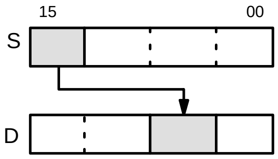
IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 24615	24700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	Read/Wr: DM 0000 to DM 0999 Rd only: DM 1000 to DM 1999	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas	Page
BLOCK SET (@)BSET(71)		Copies the content of one word or constant (S) to several consecutive words (from the starting word, St, through to the ending word, E). St and E must be in the same data area. 	**St/E:** IR HR AR LR TC DM **S:** IR SR HR AR LR TC DM #	141
DATA EXCHANGE (@)XCHG(73)		Exchanges the contents of two words (E1 and E2). 	**E1/E2:** IR HR AR LR TC DM	142
ONE DIGIT SHIFT LEFT (@)SLD(74)		Shifts all data, between the starting word (St) and ending word (E), one digit (four bits) to the left, writing zero into the rightmost digit of the starting word. St and E must be in the same data area. 	**St/E:** IR HR AR LR DM	136
ONE DIGIT SHIFT RIGHT (@)SRD(75)		Shifts all data, between starting word (St) and ending word (E), one digit (four bits) to the right, writing zero into the leftmost digit of the ending word. St and E must be in the same data area. 	**St/E:** IR HR AR LR DM	136
4-TO-16 DECODER (@)MLPX(76)		Converts up to four hexadecimal digits in the source word (S), into decimal values from 0 to 15, and turns ON the corresponding bit(s) in the result word(s) (R). There is one result word for each converted digit. Digits to be converted are designated by Di. (The rightmost digit specifies the first digit. The next digit to the left gives the number of digits to be converted minus 1. The two leftmost digits are not used.) 	**S:** IR SR HR AR LR TC DM **Di:** IR HR AR LR TC DM # **R:** IR HR AR LR DM	156

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

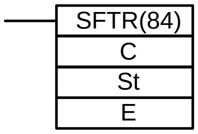
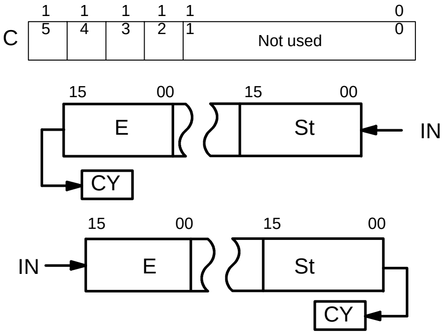
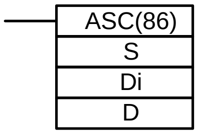
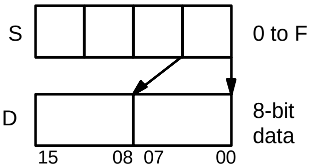
IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 24615	24700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	Read/Wr: DM 0000 to DM 0999 Rd only: DM 1000 to DM 1999	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas			Page
16-TO-4 ENCODER (@)DMPX(77)		Determines the position of the leftmost ON bit in the source word(s) (starting word: S) and turns ON the corresponding bit(s) in the specified digit of the result word (R). One digit is used for each source word. Digits to receive the converted values are designated by Di. (The rightmost digit specifies the first digit. The next digit to left gives the number of words to be converted minus 1. The two leftmost digits are not used.) 	**S:** IR SR HR AR LR TC DM	**R:** IR HR AR LR DM	**Di:** IR HR AR LR TC DM #	158
MOVE BIT (@)MOVB(82)		Transfers the designated bit of the source word or constant (S) to the designated bit of the destination word (D). The rightmost two digits of the bit designator (Bi) specify the source bit. The two leftmost digits specify the destination bit. 	**S:** IR SR HR AR LR DM #	**Bi:** IR HR AR LR TC DM #	**D:** IR HR AR LR DM	143
MOVE DIGIT (@)MOVD(83)		Moves hexadecimal content of up to four specified 4-bit source digit(s) from the source word to the specified destination digit(s) (S gives the source word address. D specifies the destination word). Specific digits within the source and destination words are defined by the Digit Designator (Di) digits. (The rightmost digit gives the first source digit. The next digit to the left gives the number of digits to be moved. The next digit specifies the first digit in the destination word.) 	**S:** IR SR HR AR LR TC DM #	**Di:** IR HR AR LR TC DM #	**D:** IR HR AR LR TC DM	143

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 24615	24700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	Read/Wr: DM 0000 to DM 0999 Rd only: DM 1000 to DM 1999	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas	Page
REVERSIBLE SHIFT REGISTER (@)SFTR(84)		Shifts bits in the specified word or series of words either left or right. Starting (St) and ending words (E) must be specified. Control word (C) contains shift direction, reset input, and data input. (Bit 12: 0 = shift right, 1 = shift left. Bit 13 is the value shifted into the source data, with the bit at the opposite end being moved to CY. Bit 14: 1 = shift enabled, 0 = shift disabled. If bit 15 is ON when SFTR(89) is executed with an ON condition, the entire shift register and CY will be set to zero.) St and E must be in the same data area and St must be less than or equal to E.  C 15 14 13 12 11 00 Not used 15 00 15 00 E St IN CY 15 00 15 00 IN E St CY	**St/E/C:** IR HR AR TC LR DM	137
ASCII CONVERT (@)ASC(86)		Converts hexadecimal digits from the source word (S) into 8-bit ASCII values, starting at leftmost or rightmost half of the starting destination word (D). The rightmost digit of Di designates the first source digit. The next digit to the left gives the number of digits to be converted. The next digit specifies the whether the data is to be transferred to the rightmost (0) or leftmost (1) half of the first destination word. The leftmost digit specifies parity: 0: none, 1: even, or 2: odd. 	**S:** IR, SR, HR, AR, LR, TC, DM **Di:** IR, HR, LR, TC, DM, # **D:** IR, HR, LR, DM	160
SUBROUTINE ENTER (@)SBS(91)		Calls subroutine N. Moves program operation to the specified subroutine.	**N:** 00 to 49	186
SUBROUTINE START SBN(92)		Marks the start of subroutine N.	**N:** 00 to 49	186

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 24615	24700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	Read/Wr: DM 0000 to DM 0999 Rd only: DM 1000 to DM 1999	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas	Page

RETURN
RET(93)
	RET(93)	Marks the end of a subroutine and returns control to the main program.	None	186

WATCHDOG TIMER REFRESH (@)WDT(94)	WDT(94) T	Sets the maximum and minimum limits for the watchdog timer (normally 0 to 130 ms). New limits: Maximum time = 130 + (100 x T) Minimum time = 130 + (100 x (T–1))	**T:** 0 to 63	218
I/O REFRESH (@)IORF(97)	IORF(97) St E	Refreshes all I/O words between the start (St) and end (E) words. Only I/O words may be designated. Normally these words are refreshed only once per cycle, but refreshing words before use in an instruction can increase execution speed. St must be less than or equal to E.	**St/E:** IR	219

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 24615	24700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	Read/Wr: DM 0000 to DM 0999 Rd only: DM 1000 to DM 1999	0000 to 9999 or 0000 to FFFF

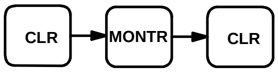
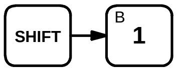
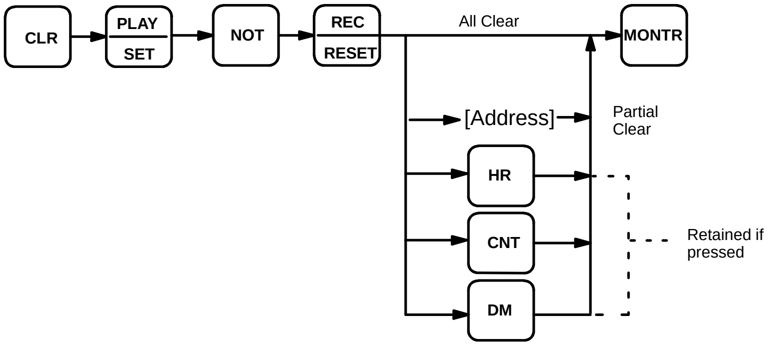
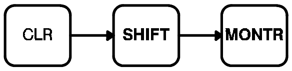
Appendix C
Programming Console Operations

The table below lists the Programming Console operations, a brief description, and the page on which they appear in the body of this manual. All operations are described briefly, and the key sequence for inputting them given, in the tables which form the second part of this appendix.

Name	Function	Page
Password Input	Prompts the user for the access password.	66
Buzzer ON/OFF	Controls whether the buzzer will sound for keystroke inputs.	67
Data Clear	Used to erase data, either selectively or totally, from the Program Memory and the IR, AR, HR, DM, and TC areas.	67
Program Header Display	Displays the program name, version number, and the date of last revision.	244
Address Designation	Displays the specified address.	70
Program Input	Used to edit or input program instructions.	71
Program Read	Allows the user to scroll through the program address-by-address. In RUN and MONITOR modes, status of bits is also given.	70
Program Search	Searches a program for the specified data address or instruction.	77
Instruction Insert Instruction Delete	Allows a new instruction to be inserted before the displayed instruction, or deletes the displayed instruction (respectively).	79
Program Check	Checks the completed program for syntax errors (up to three levels).	74
Error Message Read	Displays error messages in sequence, starting with the most severe messages.	69, 234
Bit/Word Monitor	Displays the specified address whose operand is to be monitored. In RUN or MONTR mode it will show the status of the operand for any bit or word in any data area.	236
3-word Monitor	Simultaneously monitors three consecutive words.	244
Forced Set/Reset	Set: Used to turn ON bits or timers, or to increment counters currently displayed on the left of the screen. Reset: Used to turn OFF bits, or to reset timers or counters.	239
Clear Forced Set/Reset	Simultaneously clears all forced bits within the currently displayed word.	241
Hex/BCD Data Change	Used to change the value of the leftmost BCD or hexadecimal word displayed during a Bit/Word Monitor operation.	242
Binary Data Change	Changes the value of 16-bit words bit-by-bit. Bits can be changed temporarily or permanently to the desired status.	247
SV Change SV Reset	Alters the SV of a timer or counter either by incrementing or decrementing the value, or by overwriting the original value with a new one.	249
3-word Change	Used to change the value of a word displayed during a 3-word Monitor operation.	245
Scan Time Display	Measures the duration of the current cycle. Cycle times will vary according to the execution conditions which exist in each cycle.	76
Hex/ASCII Display Change	Converts 4-digit hexadecimal data in the DM area to ASCII and vice-versa.	243
Binary Monitor	Displays the monitored area in binary format.	246
PC to PROM Writer	Outputs Program Memory to the RS-232C interface for writing to a commercial PROM writer.	252
PROM Writer to PC	Reads Program Memory data from a commercial PROM writer into the PC via the RS-232C interface.	253

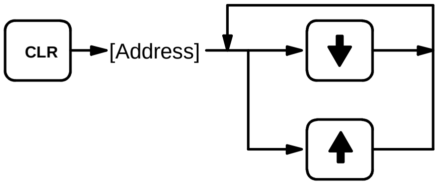
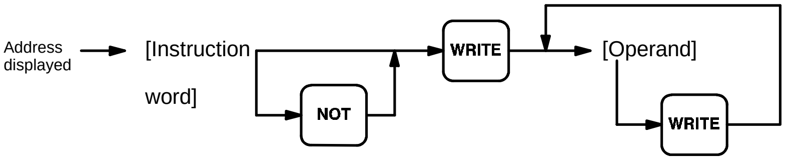
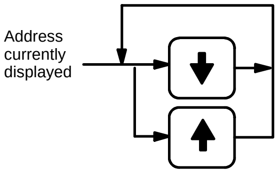
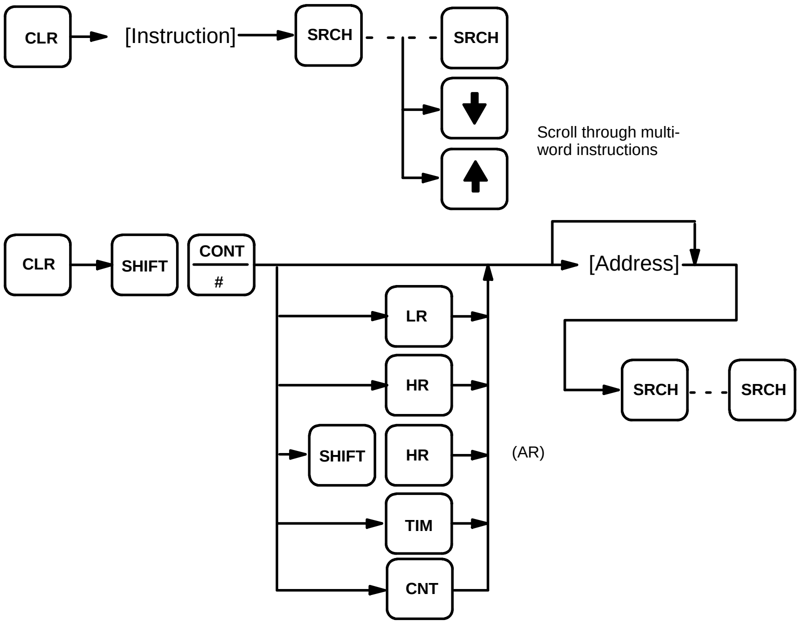
System Operations

The next set of tables lists the Programming Console operations according to their function. A brief description of each operation is given, along with the allowable modes in which it can be implemented, and the keystroke sequence used to enter it.

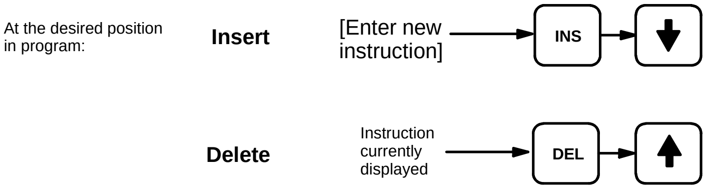
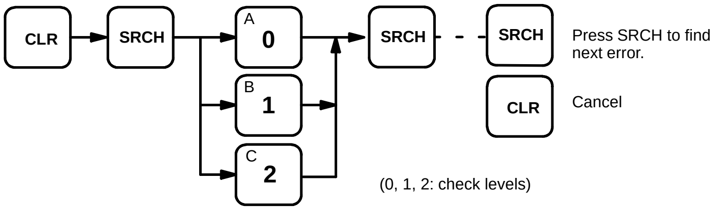
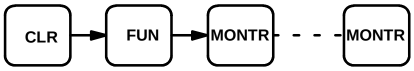
Operation/Description	Modes*	Key sequence
Password Input Controls access to the PC's programming functions. To gain access to the system once "PASSWORD" has been displayed, press CLR, MONTR, and then CLR.	R M P	
Buzzer ON/OFF The buzzer can be switched to operate whenever Programming Console keys are pressed (as well as for the normal error indication). BZ is displayed in the upper right corner when the buzzer is operative. The buzzer can be enabled by pressing SHIFT and then 1 immediately after entering the password, or after changing the mode.	R M P	
Data Clear Unless otherwise specified, this operation will clear all erasable memory in Program Memory and IR, HR, AR, DM, and TC areas. To clear EPROM memory the write enable switch must be ON (i.e., enabled). The branch lines shown are used only when performing a partial memory clear, with each of the memory areas entered being retained. Specifying an address will result in the Program Memory after and including that address being deleted. All memory up to that address will be retained.	P	 CLR PLAY SET NOT REC RESET All Clear MONTR [Address] Partial Clear HR CNT DM Retained if pressed
Program Header Display Displays the name of the program, along with the version number and the time it was last revised (given in year, month, day, hour, and minute).	R P M	

*Modes in which the given instruction is applicable: R = RUN, M = MONITOR, P = PROGRAM

Programming and Debugging Operations

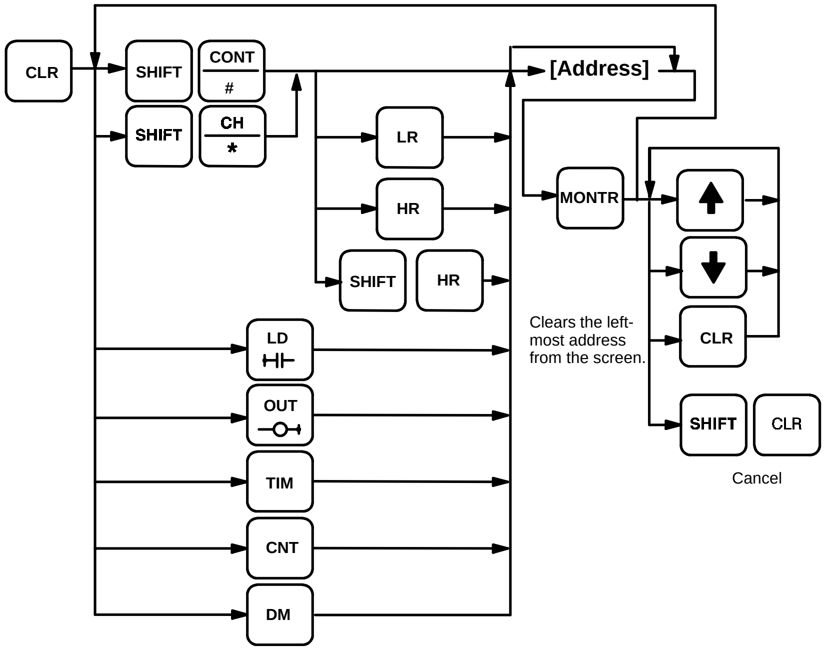
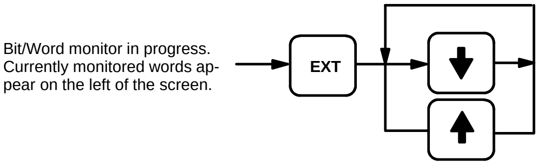
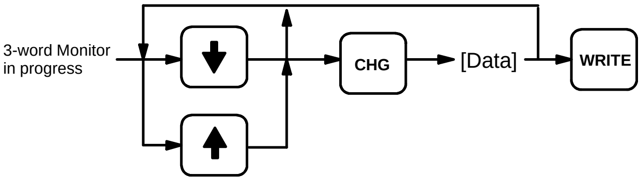
Operation/Description	Modes*	Key sequence
Address Designation Displays the specified address. Can be used to start programming from a non-zero address or to access an address for editing. Leading zeros need not be entered. The contents of the address will not be displayed until the down key is pressed. The up and down keys can then be used to scroll through the Program Memory.	R P M	
Program Input Used to enter or edit program instructions. This operation over-writes the contents of the memory at the displayed address. Once at the desired address, enter the new instruction word and then press WRITE (preceded by NOT for differentiated instructions). Input the required operands, and press WRITE after each.	P	 Address displayed [Instruction word] NOT WRITE [Operand] WRITE
Program Read Allows the user to scroll through the program address-by-address. If the Program Memory is read in RUN or MONITOR mode, the ON/OFF status of each displayed bit is also shown.	R P M	
Program Search Allows the program to be searched for occurrences of any designated instruction or data area address. To designate a bit address, press SHIFT, CONT/#, and then input the address. Then press SRCH. Pressing SRCH again will find the next occurrence. For multi-word instructions, the up and down keys can be used to scroll through the words before continuing the search. In RUN or MONITOR mode, the ON/OFF status of each monitored bit will also be displayed.	R P M	 CLR [Instruction] SRCH SRCH ↓ ↑ Scroll through multi-word instructions CLR SHIFT CONT # LR HR SHIFT HR (AR) TIM CNT [Address] SRCH SRCH

*Modes in which the given instruction is applicable: R = RUN, M = MONITOR, P = PROGRAM

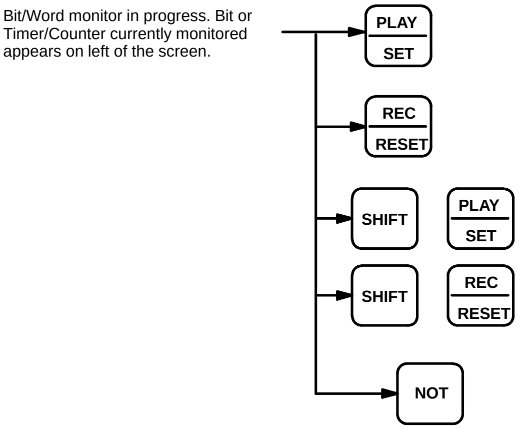
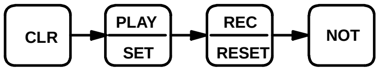
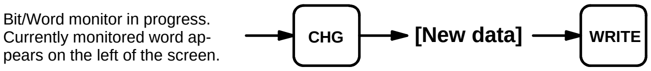
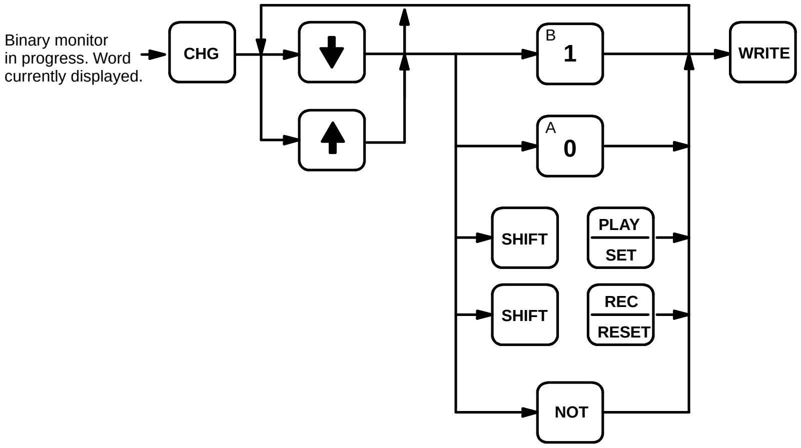
Operation/Description	Modes*	Key sequence
Instruction Insert and Instruction Delete The displayed instruction can be deleted, or another instruction can be inserted before it. Care should be taken to avoid inadvertent deletions as there is no way of recovering the instructions other than to re-enter them. When an instruction is deleted all subsequent instruction addresses are automatically adjusted so that there are no empty addresses, or instructions without addresses.	P	 At the desired position in program: Insert [Enter new instruction] INS Delete Instruction currently displayed DEL
Program Check Once a program has been entered, it should be checked for errors. This program check can be used to search for three levels of syntax errors. Details of the errors covered by each level are given in *Section 4-6-3 Checking the Program*. The address where the error was generated will also be displayed.	P	 CLR SRCH A 0 B 1 C 2 SRCH SRCH Press SRCH to find next error. CLR Cancel (0, 1, 2: check levels)
Error Message Read Displays error messages in sequence with most severe messages displayed first. Press monitor to access remaining messages. In PROGRAM mode, pressing MONTR clears the displayed message from memory and the next message is displayed.	R P M	

*Modes in which the given instruction is applicable: R = RUN, M = MONITOR, P = PROGRAM

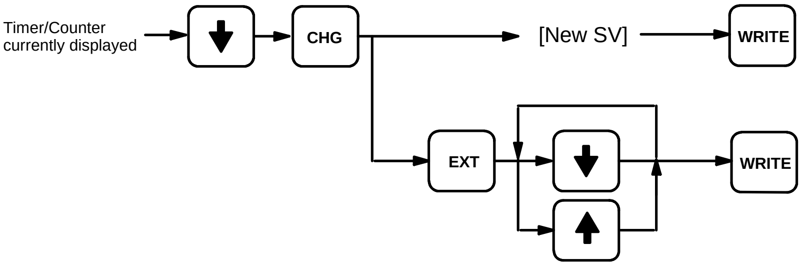
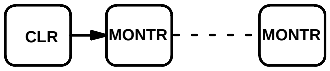
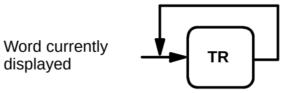
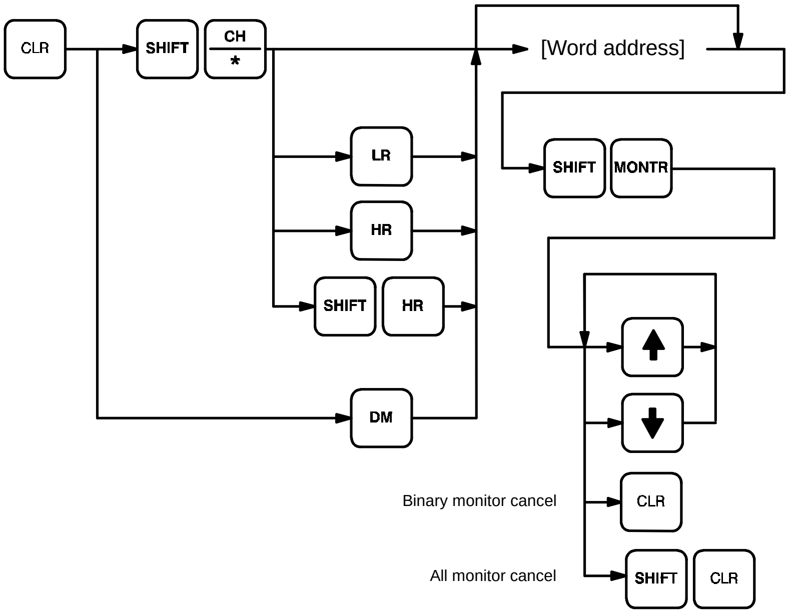
Monitoring and Data Changing Operations

Operation/Description	Modes*	Key sequence
Bit/Word Monitor Up to six memory addresses, containing either words or bits, or a combination of the two, can be monitored at once. Only three can be displayed at any one time. If operated in RUN or MONITOR mode, the status of monitored bits will also be displayed. The operation can be started from a cleared display by entering the address of the first word or bit to be monitored and pressing MONTR, or from any address in the program by displaying the address of the bit or word to be monitored and pressing MONTR. When a timer or counter is monitored, its PV will be displayed and a box is displayed in the bottom left hand corner if the Completion Flag is ON.	R P M	 CLR SHIFT CONT # SHIFT CH * [Address] LR HR SHIFT HR MONTR LD OUT TIM CNT DM Clears the left-most address from the screen. CLR SHIFT CLR Cancel
3-word Monitor Monitors three consecutive words simultaneously. Specify the lowest valued address of the three words, press MONTR, and then press EXT to display the data contents of the specified word and the two words that follow. Pressing CLR will change the three-word monitor operation into a single-word display.	R P M	 Bit/Word monitor in progress. Currently monitored words appear on the left of the screen. EXT
3-word Change This operation changes the value of a word displayed during a 3-word Monitor operation. The blinking cursor indicates the word that will be affected by the operation. The cursor can be moved by using the up and down keys. When the cursor is at the desired location, press CHG. After entering the new data, pressing WRITE causes the original data to be overwritten.	P M	

*Modes in which the given instruction is applicable: R = RUN, M = MONITOR, P = PROGRAM

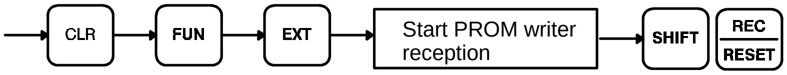
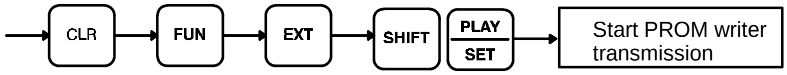
Operation/Description	Modes*	Key sequence
Forced Set/Reset If a bit, timer, or counter address is leftmost on the screen during a Bit/Word Monitor operation, pressing PLAY/SET will turn ON the bit, start the timer, or increment the counter. Pressing REC/RESET will turn OFF the bit, or reset the timer or counter. These force-sets and force-resets are effective while the key is held down. Permanent sets and resets can be implemented. By pressing SHIFT first, the force operations will be effective until NOT is pressed, or until a Clear Forced Set/Reset operation is performed. Timers will not operate in PROGRAM mode. SR bits are not affected by this operation.	P M	 Bit/Word monitor in progress. Bit or Timer/Counter currently monitored appears on left of the screen. PLAY SET REC RESET SHIFT PLAY SET SHIFT REC RESET NOT
Clear Forced Set/Reset Simultaneously clears all forced set and forced reset bits within the word currently displayed.	P M	
Hex/BCD Data Change Used to edit the leftmost BCD or hexadecimal value displayed during a Bit/Word Monitor operation. If a timer or counter is leftmost on the display, the PV will be the value displayed and affected by this operation. It can only be changed in MONITOR mode and only while the timer or counter is operating. SR words cannot be changed using this operation.	P M	 Bit/Word monitor in progress. Currently monitored word appears on the left of the screen. CHG [New data] WRITE
Binary Data Change This operation is used to change the value of IR, HR, AR, LR, or DM words bit-by-bit. The cursor can be moved left by using the up key, and right by using the down key. The position of the cursor is the bit that will be overwritten. There are two types of changes, temporary and permanent. Temporary changes result if 1 or 0 is entered. Permanent changes are made by pressing SHIFT and SET, or SHIFT and RESET. The former will result in an S being displayed in that bit position. Similarly, SHIFT and RESET will produce an R in the display. During operation of the PC, the bits having 1 or 0 values will change according to the program conditions. Bits with S or R, however, will always be treated as a 1 or 0, respectively. NOT cancels S and R settings and the bits will become 1 or 0, respectively.	P M	 Binary monitor in progress. Word currently displayed. CHG B 1 WRITE A 0 SHIFT PLAY SET SHIFT REC RESET NOT

*Modes in which the given instruction is applicable: R = RUN, M = MONITOR, P = PROGRAM

Operation/Description	Modes*	Key sequence
SV Change, SV Reset There are two ways of modifying the SVs for timers and counters. One method is to enter a new value. The second is to increment or decrement the existing SV. In MONITOR mode the SV can be changed while the program is being executed. Incrementing and decrementing can only be carried out if the SV has been entered as a constant.	P M M	 Timer/Counter currently displayed CHG [New SV] WRITE EXT WRITE
Scan (Cycle) Time Display This operation should be performed after all syntax errors have been corrected. The cycle time can only be checked in RUN or MONITOR mode and while the program is being executed. The cycle time displayed after pressing CLR and MONTR is that for the current cycle. Pressing MONTR again will display a new cycle time. Any difference between successive cycle times is due to the different execution conditions that exist during each cycle.	R M	
Hex/ASCII Display Change Converts 4-digit hexadecimal DM data to ASCII and vice-versa.	R P M	
Binary Monitor The contents of a monitored word can be specified to be displayed in binary by pressing SHIFT and MONTR after entering the word address. Words can be scrolled by pressing the up and down keys to increment and decrement the displayed address. To terminate the binary display, press CLR.	R P M	 CLR SHIFT CH * [Word address] LR HR SHIFT HR DM SHIFT MONTR Binary monitor cancel CLR All monitor cancel SHIFT CLR

*Modes in which the given instruction is applicable: R = RUN, M = MONITOR, P = PROGRAM

PROM Writer Operations (Non-OMRON Products Only)

Operation/Description	Modes*	Key sequence
PC to PROM Writer Copies Program Memory to the RS-232C interface for writing to a commercial PROM writer.	P	 CLR FUN EXT Start PROM writer reception SHIFT REC RESET
PROM Writer to PC Reads Program Memory data from a commercial PROM writer into the PC via the RS-232C interface. This operation overwrites the PC's existing program memory.	P	 CLR FUN EXT SHIFT PLAY SET Start PROM writer transmission

*Modes in which the given instruction is applicable: R = RUN, M = MONITOR, P = PROGRAM

Appendix D
Error and Arithmetic Flag Operation

The following table shows the instructions that affect the ER, CY, GT, LT 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.

Instructions	25503 (ER)	25504 (CY)	25505 (GR)	25506 (EQ)	25507 (LE)
TIM	↕	Unaffected	Unaffected	Unaffected	Unaffected
CNT					
END(01)	OFF	OFF	OFF	OFF	OFF
CNTR(12)	↕	Unaffected	Unaffected	Unaffected	Unaffected
TIMH(15)					
WSFT(16)					
CMP(20)	↕	Unaffected	↕	↕	↕
MOV(21)	↕	Unaffected	Unaffected	↕	Unaffected
MVN(22)					
BIN(23)					
BCD(24)					
ASL(25)	↕	↕	Unaffected	↕	Unaffected
ASR(26)					
ROL(27)					
ROR(28)					
COM(29)	↕	Unaffected	Unaffected	↕	↕
ADD(30)	↕	↕	Unaffected	↕	Unaffected
SUB(31)					
MUL(32)	↕	Unaffected	Unaffected	↕	Unaffected
DIV(33)					
ANDW(34)					
ORW(35)					
XORW(36)					
XNRW(37)					
INC(38)					
DEC(39)					
STC(40)	Unaffected	ON	Unaffected	Unaffected	Unaffected
CLC(41)	Unaffected	OFF	Unaffected	Unaffected	Unaffected
MSG(46)	↕	Unaffected	Unaffected	Unaffected	Unaffected
LMSG(47)					
ADB(50)	↕	↕	Unaffected	↕	Unaffected
SBB(51)					
MLB(52)	↕	Unaffected	Unaffected	↕	Unaffected
DVB(53)					

Instructions	25503 (ER)	25504 (CY)	25505 (GR)	25506 (EQ)	25507 (LE)
RDM(60)	↕	Unaffected	Unaffected	Unaffected	Unaffected
HDM(61)					
KEY(62)					
POUT(63)					
PIN(64)					
HTS(65)	↕	Unaffected	Unaffected	↕	Unaffected
STH(66)					
BCNT(67)					
BCMP(68)					
HEX(69)	↕	Unaffected	Unaffected	Unaffected	Unaffected
XFER(70)	↕	Unaffected	Unaffected	Unaffected	Unaffected
BSET(71)					
XCHG(73)	↕	Unaffected	Unaffected	Unaffected	Unaffected
SLD(74)					
SRD(75)					
MLPX(76)					
DMPX(77)					
MOVB(82)	↕	Unaffected	Unaffected	Unaffected	Unaffected
MOVD(83)					
SFTR(84)	↕	↕	Unaffected	Unaffected	Unaffected
ASC(86)	↕	Unaffected	Unaffected	Unaffected	Unaffected
SBS(91)	↕	Unaffected	Unaffected	Unaffected	Unaffected

Appendix E
Memory Areas

This appendix contains tables from *Section 3 Memory Areas*. They have been duplicated here for convenience. Refer to *Section 3* for details. The Parameter Area of System DM is outlined in *Appendix K Parameter Area Coding Charts*.

Memory Areas

Area	Acronym	Range
Internal Relay*	IR	Words: 040 to 246 Bits: 04000 to 24615
Special Relay	SR	Words: 247 to 255 Bits: 24700 to 25507
Auxiliary Relay	AR	Words: AR 00 to AR 27 Bits: AR 0000 to AR 2715
Data Memory	DM	Read/Write: DM 0000 to DM 0999 Read only: DM 1000 to DM 1999
Holding Relay	HR	Words: HR 00 to HR 99 Bits: HR 0000 to HR 9915
Timer/Counter	TC	TC 000 to TC 511 (TC numbers used to access other information)
Link Relay	LR	Words: LR 00 to LR 63 Bits: LR 0000 to 6315
Temporary Relay	TR	TR 00 to TR 07 (bits only)
Program Memory	UM	UM: Depends on Memory Unit used.

* Including I/O words/bits (see next table).

Word Allocations in IR Area

Connected Unit	I/O Designation	C20H or C28H	C40H
CPU	Input words	IR 000	IR 000 to IR 001
	Output words	IR 002	IR 002 to IR 003
1st Expansion I/O Unit	Input words	IR 010	IR 010 to IR 011
	Output words	IR 012	IR 012 to IR 013
2nd Expansion I/O Unit	Input words	IR 020	IR 020 to IR 021
	Output words	IR 022	IR 022 to IR 023
3rd Expansion I/O Unit	Input words	IR 030	IR 030 to IR 031
	Output words	IR 032	IR 032 to IR 033

CPUs

Model	Input bits	Outputs bits	Work bits
C20H	IR 00000 to IR 00011	IR 00200 to IR 00207	IR 00208 to IR 00215
C28H	IR 00000 to IR 00015	IR 00200 to IR 00211	IR 00212 to IR 00215
C40H	IR 00000 to IR 00015 IR 00100 to IR 00107	IR 00200 to IR 00211 IR 00300 to IR 00303	IR 00212 to IR 00215 IR 00304 to IR 00315
C60H	IR 00000 to IR 00015 IR 00100 to IR 00115	IR 00200 to IR 00211 IR 00300 to IR 00315	IR 00312 to IR 00315

Expansion I/O Units

Model	Input bits	Outputs bits	Work bits
C20H	Bits 00 to 11 of IR n	Bits 00 to 07 of IR n+2	Bits 08 to 15 of IR n+2
C28H	Bits 00 to 15 of IR n	Bits 00 to 11 of IR n+2	Bits 12 to 15 of IR n+2
C40H	Bits 00 to 15 of IR n Bits 00 to 07 of IR n+1	Bits 00 to 11 of IR n+2 Bits 00 to 03 of IR n+3	Bits 12 to 15 of IR n+2 Bits 04 to 15 of IR n+3
C60H	Bits 00 to 15 of IR n Bits 00 to 15 of IR n+1	Bits 00 to 11 of IR n+2 Bits 00 to 15 of IR n+3	Bits 12 to 15 of IR n+3

SR Area

Word(s)	Bit(s)	Function
247 to 250	00 to 07	Reserved
	08 to 15	Reserved
251	00 to 15	Not used
252	00 to 05	Not used
	06	Reserved
	07	Not used
	08	RS-232C Communications Error Flag and CPU-mounting Host Link Unit Communications Error Flag
	09	RS-232C Interface Restart Bit and CPU-mounting Host Link Unit Restart Bit
	10	Calendar/clock Area Refresh Bit
	11	Forced Status Hold Bit
	12	I/O Status Hold Bit
	13	Reserved
	14 to 15	Not used
253	00 to 07	FAL number output area.
	08	Battery Alarm Flag
	09	Cycle Time Error Flag
	10	Not used
	11	Not used
	12	Not used
	13	Always ON Flag
	14	Always OFF Flag
	15	First Cycle
254	00	1-minute clock pulse bit
	01	0.02-second clock pulse bit
	02 to 06	Reserved for future use
	07	Step Flag
	08 to 14	Reserved for future use
	15	Reserved
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

AR Area

Word(s)	Bit(s)	Function
02	00 to 10	Reversible Drum Counter (RDM(60)) Reset Bits
	11	High-speed Counter Reset Bit
	12	High-speed Counter Reset Flag
03	00 to 10	Reversible Drum Counter (RDM(60)) Direction Bits
	11	High-speed Counter Bank Bit
04	00 to 07	RS-232C Communications Error Code
	13	RS-232C Reception Impossible Flag
	14	RS-232C Reception Completed Flag
	15	RS-232C Transmission Possible Flag
05	00 to 07	RS-232C Reception Counter
	08 to 15	RS-232C Transmission Counter
06	00 to 15	RS-232C Bytes Received Area
07	08	TERMINAL Mode Input Cancel Bit
	13	Error History Overwrite Bit
	14	Error History Reset Bit
	15	Error History Enable Bit
08	00 to 15	RS-232C Bytes Input Area
12	00 to 15	System Parameter Warning Flags
13	00 to 13	System Parameter Warning Flags
	14	System Parameter Backup Flag
	15	System Parameter/Backup Area Checksum Flag
14	00 to 03	System Command Response Code
	04 to 06	Not used
	07	System Command Completion Flag
	08 to 11	System Command Command Code
	12 to 14	Not used
	15	System Command Execution Bit
15	00 to 07	Startup Operating Mode
17	00 to 15	Current Time Area
18 to 21	00 to 15	Calendar/clock Area (AR 2113: Seconds Round-off Bit; AR 2114: Stop Bit; AR 2115: Set Bit)
22	00 to 15	TERMINAL Mode Key Bits
23	00 to 15	Power-off Counter
24	05	SCAN(18) Scan Time Flag
	15	Programming Console or Peripheral Interface Unit Mounted Flag
25	00 to 15	FALS-generating Address
26	00 to 15	Maximum Cycle Time Area
27	00 to 15	Current Cycle Time Area

DM Area

Addresses	User program read/write	Usage
DM 0000 to DM 0899	Read/write	General User Area
DM 0900 to DM 0929	Read/write	Parameter Area
DM 0930 to DM 0968	---	Not used.
DM 0969 to DM 0999	Read/write	Error History Area
DM 1000 to DM 1899	Read only	General User Area
DM 1900 to DM 1929	Read only	Parameter Backup Area
DM 1930 to DM 1989	---	Not used.
DM 1990 to DM 1999	Read only	User Program Header

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		

Data Storage

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 Table

Decimal	BCD	Hex	Binary
00	00000000	00	00000000
01	00000001	01	00000001
02	00000010	02	00000010
03	00000011	03	00000011
04	00000100	04	00000100
05	00000101	05	00000101
06	00000110	06	00000110
07	00000111	07	00000111
08	00001000	08	00001000
09	00001001	09	00001001
10	00010000	0A	00001010
11	00010001	0B	00001011
12	00010010	0C	00001100
13	00010011	0D	00001101
14	00010100	0E	00001110
15	00010101	0F	00001111
16	00010110	10	00010000
17	00010111	11	00010001
18	00011000	12	00010010
19	00011001	13	00010011
20	00100000	14	00010100
21	00100001	15	00010101
22	00100010	16	00010110
23	00100011	17	00010111
24	00100100	18	00011000
25	00100101	19	00011001
26	00100110	1A	00011010
27	00100111	1B	00011011
28	00101000	1C	00011100
29	00101001	1D	00011101
30	00110000	1E	00011110
31	00110001	1F	00011111
32	00110010	20	00100000

Appendix I
Extended ASCII

Programming Console and Data Access 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		0	@	P	‘	p
0001	**1**	SOH	DC$_1$	!	1	A	Q	a	q	!	1	A	Q	a	q
0010	**2**	STX	DC$_2$	”	2	B	R	b	r	”	2	B	R	b	r
0011	**3**	ETX	DC$_3$	#	3	C	S	c	s	#	3	C	S	c	s
0100	**4**	EOT	DC$_4$	$	4	D	T	d	t	$	4	D	T	d	t
0101	**5**	ENQ	NAK	%	5	E	U	e	u	%	5	E	U	e	u
0110	**6**	ACK	SYN	&	6	F	V	f	v	&	6	F	V	f	v
0111	**7**	BEL	ETB	’	7	G	W	g	w	’	7	G	W	g	w
1000	**8**	BS	CAN	(	8	H	X	h	x	(	8	H	X	h	x
1001	**9**	HT	EM	)	9	I	Y	i	y	)	9	I	Y	i	y
1010	**A**	LF	SUB	*	:	J	Z	j	z	*	:	J	Z	j	z
1011	**B**	VT	ESC	+	;	K	[	k	{	+	;	K	[	k	{
1100	**C**	FF	FS	,	<	L	\	l	|	,	<	L	\	l	|
1101	**D**	CR	GS	-	=	M	]	m	}	-	=	M	]	m	}
1110	**E**	S0	RS	.	>	N	^	n	~	.	>	N	^	n	
1111	**F**	S1	US	/	?	O	_	o	«	/	?	O	_	o	~

Appendix J
Programming Console Key Codes

The following codes can be used in KEY(62) to execute Programming Console operations from the user program. Refer to *Section 5 Instruction Set* for programming details and to *Section 7 Program Input, Debugging, and Execution* for Programming Console details.

The keys shown below are activated with the hexadecimal code given with it. The Programming Console display can be reset to the password display by inputting **40**$_{\text{hex}}$.

3D FUN	**3C** SFT	**3B** NOT	**38** None	**39** None	**3A** SHIFT
35 AND ⊣⊢	**34** OR ⊣⊢	**33** CNT	**30** TR	**31** LR	**32** HR
2D LD ⊣⊢	**2C** OUT –○	**2B** TIM	**28** DM	**29** CH / *	**2A** CONT / #
25 7	**24** 8	**23** 9	**20** EXT	**21** CHG	**22** SRCH
1D E 4	**1C** F 5	**1B** 6	**18** PLAY / SET	**19** DEL	**1A** MONTR
15 B 1	**14** C 2	**13** D 3	**10** REC / RESET	**11** INS	**12** ↑
0D A 0	**0C** None	**0B** CLR	**08** VER	**09** WRITE	**0A** ↓

Appendix K
Parameter Area Coding Charts

This appendix is provided for you to use in determining and inputting settings into the parameter area of System DM. The default settings are given for convenience. Only those settings which differ from the defaults need to be noted. DM addresses not in parentheses are those in the parameter area; those within parentheses are those of the corresponding words in the parameter backup area.

Word and parameter		**Default**	**Setting**
Bit	**Content/Meaning**		
DM 0900 (DM 1900): PC Mode on Startup		Key switch: 0000	
00 to 07	00: PROGRAM 01: MONITOR 02: RUN	00	
08 to 15	00: As set on Programming Console key switch 01: Mode when PC was last turned off (in AR 15) 02: Mode set in bits 00 to 07, above	00	
DM 0901 (DM 1901): Cycle Time Limit		100 ms: 1001	
00 to 07	Cycle time limit in units of ten milliseconds. Setting is between 00 and 99 in BCD resulting in cycle time limits between 000 and 990 ms, respectively.	10	
08 to 15	00: Bits 00 to 07 disabled (i.e., cycle time limit is 100 ms) 01: Bits 00 to 07 enabled	01	
DM 0902 (DM 1902): Peripheral Device Service Time		5%: 0000	
00 to 07	Percent of cycle time allocated to Device servicing between 00 and 99 in BCD.	00	
08 to 15	00: Bits 00 to 07 disabled (i.e., servicing set to 5%) 01: Bits 00 to 07 enabled	00	
DM 0903 (DM 1903): RS-232C Interface Service Time		5%: 0000	
00 to 07	Percent of cycle time allocated to RS-232C interface servicing between 00 and 99 in BCD.	00	
08 to 15	00: Bits 00 to 07 disabled (i.e., servicing set to 5%) 01: Bits 00 to 07 enabled	00	
DM 0904 (DM 1904): Programming Console Message Language Bits		English:	
08 to 15	00: English 01: Japanese 02: German 03: French 04: Italian 05: Spanish	0000	
DM 0905 (DM 1905): General High-speed Counter Bits		No counter: 0000	
00	IR 00200 Enable Bit	0	
01	IR 00201 Enable Bit	0	
02	IR 00202 Enable Bit	0	
03	IR 00203 Enable Bit	0	
04	IR 00204 Enable Bit	0	
05	IR 00205 Enable Bit	0	
06	IR 00206 Enable Bit	0	
07	IR 00207 Enable Bit	0	
08 to 10	Final step for bank 0 (0 to 7)	0	
11 to 13	Final step for bank 1 (0 to 7)	0	
14	Hard Reset Enable Bit (Turn ON to enable hard reset.)	0	
15	High-speed Counter Enable Bit (Turn ON to enable counter.)	0	

Word and parameter		Default	Setting
Bit	Content/Meaning		
DM 0906 (DM 1906): High-speed Counter Interrupt Output Table		0000	
00	Step 0 output status for IR 00200	0	
01	Step 0 output status for IR 00201	0	
02	Step 0 output status for IR 00202	0	
03	Step 0 output status for IR 00203	0	
04	Step 0 output status for IR 00204	0	
05	Step 0 output status for IR 00205	0	
06	Step 0 output status for IR 00206	0	
07	Step 0 output status for IR 00207	0	
08	Step 1 output status for IR 00200	0	
09	Step 1 output status for IR 00201	0	
10	Step 1 output status for IR 00202	0	
11	Step 1 output status for IR 00203	0	
12	Step 1 output status for IR 00204	0	
13	Step 1 output status for IR 00205	0	
14	Step 1 output status for IR 00206	0	
15	Step 1 output status for IR 00207	0	
DM 0907 (DM 1907): **High-speed Counter Interrupt Output Table (continued)** This word continues the table started in DM 0906.		0000	
00	Step 2 output status for IR 00200	0	
01	Step 2 output status for IR 00201	0	
02	Step 2 output status for IR 00202	0	
03	Step 2 output status for IR 00203	0	
04	Step 2 output status for IR 00204	0	
05	Step 2 output status for IR 00205	0	
06	Step 2 output status for IR 00206	0	
07	Step 2 output status for IR 00207	0	
08	Step 3 output status for IR 00200	0	
09	Step 3 output status for IR 00201	0	
10	Step 3 output status for IR 00202	0	
11	Step 3 output status for IR 00203	0	
12	Step 3 output status for IR 00204	0	
13	Step 3 output status for IR 00205	0	
14	Step 3 output status for IR 00206	0	
15	Step 3 output status for IR 00207	0	

Word and parameter		Default	Setting
Bit	**Content/Meaning**		
DM 0908 (DM 1908): **High-speed Counter Interrupt Output Table (continued)** This word continues the table started in DM 0906.		0000	
00	Step 4 output status for IR 00200	0	
01	Step 4 output status for IR 00201	0	
02	Step 4 output status for IR 00202	0	
03	Step 4 output status for IR 00203	0	
04	Step 4 output status for IR 00204	0	
05	Step 4 output status for IR 00205	0	
06	Step 4 output status for IR 00206	0	
07	Step 4 output status for IR 00207	0	
08	Step 5 output status for IR 00200	0	
09	Step 5 output status for IR 00201	0	
10	Step 5 output status for IR 00202	0	
11	Step 5 output status for IR 00203	0	
12	Step 5 output status for IR 00204	0	
13	Step 5 output status for IR 00205	0	
14	Step 5 output status for IR 00206	0	
15	Step 5 output status for IR 00207	0	
DM 0909 (DM 1909): **High-speed Counter Interrupt Output Table (continued)** This word continues the table started in DM 0906.		0000	
00	Step 6 output status for IR 00200	0	
01	Step 6 output status for IR 00201	0	
02	Step 6 output status for IR 00202	0	
03	Step 6 output status for IR 00203	0	
04	Step 6 output status for IR 00204	0	
05	Step 6 output status for IR 00205	0	
06	Step 6 output status for IR 00206	0	
07	Step 6 output status for IR 00207	0	
08	Step 7 output status for IR 00200	0	
09	Step 7 output status for IR 00201	0	
10	Step 7 output status for IR 00202	0	
11	Step 7 output status for IR 00203	0	
12	Step 7 output status for IR 00204	0	
13	Step 7 output status for IR 00205	0	
14	Step 7 output status for IR 00206	0	
15	Step 7 output status for IR 00207	0	

Word and parameter		Default	Setting
Bit	**Content/Meaning**		
High-speed Counter Step Table These words contain the width of each step of the counter. The total width of the counter is the sum of all the widths set below.		---	---
Step 0: DM 0910		0000 (10000) when counter is enabled.	
Step 1: DM 0911			
Step 2: DM 0912			
Step 3: DM 0913			
Step 4: DM 0914			
Step 5: DM 0915			
Step 6: DM 0916			
Step 7: DM 0917			
DM 0918 and DM 0919 (DM 1918 and DM 1919) Not used.		0000	0000
DM 0920 (DM 1920)		0000	
00 to 07	**Standard/Custom Communications Format Selectio**n 00: Standard (1 start bit, 7-bit data length, even parity, 2 stop bits, 9,600 baud) 01: Custom settings (i.e., according to contents of DM 0921)	Standard: 00	
08 to 15	**RS-232C Mode** 00: Host link 01: Memory upload/download 02: ASCII output mode	Host link: 00	
DM 0921 (DM 1921)		0000	
00 to 07	**Baud Rate (if DM 0920 bits 00 to 07 are 01)** 00: 300 bps 01: 600 bps 02: 1,200 bps 03: 2,400 bps 04: 4,800 bps* 05: 9,600 bps*	300 bps: 00	
08 to 15	**Data Format (if DM 0920 bits 00 to 07 are 01)** 00: 1 start bit, 7-bit data, 2 stop bits, even parity 01: 1 start bit, 7-bit data, 2 stop bits, odd parity 02: 1 start bit, 8-bit data, 1 stop bits, no parity 03: 1 start bit, 8-bit data, 2 stop bits, no parity 04: 1 start bit, 8-bit data, 1 stop bits, even parity 05: 1 start bit, 8-bit data, 1 stop bits, odd parity	1 start bit, 7-bit data, 2 stop bits, even parity: 00	
DM 0922 (DM 1922)		0000	
00 to 07	**Transmission Delay** In tenths of milliseconds between 00 and 99 (BCD, correspond to 000 and 990 ms delays, respectively)	0 ms: 00	
08 to 15	**RTS/CTS Control** 00: Without RTS/CTS 01: With RTS/CTS	Without RTS/CTS: 00	
DM 0923 (DM 1923) (00 to 07 not used.)		0000	00
08 to 15	Unit number for host link mode between 00 and 31 in BCD	#0: 00	
DM 0924 (DM 1924) (00 to 07 not used.)		0000	00
08 to 15	**Transmission Format for Memory Upload/Download** 00: Intel HEX 01: Motorola S	Intel HEX: 00	
DM 0925 (DM 1925)		0000	
00 to 07	**Starting Code** This byte contains the starting code used in POUT(63) and PIN(64) transmissions when bits 08 to 15 contain 01.	No starting code	
08 to 15	**Starting Code/No Starting Code Selectio**n 00: No starting code 01: Starting code set in bits 00 to 07		

Word and parameter		Default	Setting
Bit	**Content/Meaning**		
DM 0926 (DM 1926)		0000	
00 to 07	**End Code** This byte contains the end code used in POUT(63) and PIN(64) transmissions when bits 08 to 15 contain 01.	No end code	
08 to 15	**End Code/No End Code Selectio**n 00: No end code 01: End code set in bits 00 to 07		
DM 0927 to DM 0929 (DM 1927 to DM 1929)		0000	0000
Not used.		---	**---**

*Higher baud rates may produce errors in RS-232C communications if both RS-232C interface and Peripheral Interface Unit are used.

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 optical fiber 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.

BCD Short for binary-coded decimal.

BCD calculation An arithmetic calculation that uses numbers expressed in binary-coded decimal.

binary A number system where all numbers are expressed in base 2, i.e., numbers are written using only 0’s and 1’s. Each group of four binary bits is equivalent to one hexadecimal digit.

binary 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.

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.

CONSOLE mode A Programming Console mode that enables key sequences to be input. See TERMINAL mode.

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.

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.

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.

Error History Area An area in System DM that is used to store records indicating the time and nature of up to ten errors that have occurred in the system.

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.

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 x 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.

GPC
Acronym for Graphic Programming Console.

Graphic Programming Console
A programming device with advanced programming and debugging capabilities to facilitate PC operation. A Graphic Programming Console is provided with a large display onto which ladder-diagram programs can be written directly in ladder-diagram symbols for input into the PC without conversion to mnemonic form.

hardware error
An error originating in the hardware structure (electronic components) of the PC, as opposed to a software error, which originates in software (i.e., programs).

hexadecimal
A number system where all numbers are expressed to the base 16. In a PC all data is ultimately stored in binary form, however, displays and inputs on Programming Devices are often expressed in hexadecimal to simplify operation. Each group of four binary bits is numerically equivalent to one hexadecimal digit.

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 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 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 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.

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.

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.

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.

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.

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.

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.

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.

Parameter Area

A part of System DM used to designate various PC operating parameters.

Parameter Backup Area

A part of System DM used to back up the Parameter Area.

parity

Adjustment of the number of ON bits in a word or other unit of data so that the total is always an even number or always an odd number. Parity is generally used to check the accuracy of data after being transmitted by confirming that the number of ON bits is still even or still odd.

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.

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. Programmable Controllers are used to automate control of external devices.

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.

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.

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.

scan time *See cycle 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.

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.

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.

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 unwritable SR bits as a destination), and mistakes in actual application of instructions (e.g., a call to a subroutine that does not exist).

System Command Control bits in the AR area used to manipulate the Parameter and Parameter Backup Areas.

system configuration The arrangement in which Units in a system are connected.

System DM A dedicated portion of the DM area that is used for special purposes in controlling and managing the PC. Includes the Program Version, Parameter Area, Parameter Backup Area, User Program Header, and Error History Area.

system error An error generated by the system, as opposed to one resulting from execution of an instruction designed to generate an error.

system error message An error message generated by the system, as opposed to one resulting from execution of an instruction designed to generate a message.

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.

TERMINAL Mode A Programming Console mode used to automatically display messages produced by the program and to input data area-mapped keys.

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.

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.

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. 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.

User DM The portion of the DM area that is available for general data storage and manipulation. Part of User DM may be dedicated if Special I/O Units or Link Units are part of the PC.

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.

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.

work bit A bit in a work word.

work word A word that can be used for data calculation or other manipulation in programming, i.e., a ‘work space’ in memory. A large portion of the IR area is always reserved for work words. Parts of other areas not required for special purposes may also be used as work words.

Index

Revision History

A manual revision code appears as a suffix to the catalog number on the front cover of the manual.

Cat. No. W176-E1-4A

Revision code

The following table outlines the changes made to the manual during each revision. Page numbers refer to the previous version.

Revision code	Date	Revised content
1	July 1990	Original production
2	June 1991	**Page 5**: PROM Writer, Floppy Disk Interface Unit, and Printer Interface Unit require interfacing via GPC. **Page 15**: First paragraph, third line: Replace “**3**-digit BCD” with “**4**-digit BCD”. **Pages 18 and 19**: Remove the first two paragraphs of *Section 3–4–1.* **Page 22**: Paragraph four rewritten. **Page 24***: Section 3–5–4* paragraph one: Replace “converted to a parity code” with “counted in hexadecimal”. **Page 27:** N0. 3 under “Executing System Command” and second sentence in paragraph one of *Section 3–5–10* corrected. **Page 28:** Number 3 under “Set Bit” corrected. **Page 29:** Table: Replace “AR **200**” with “AR **2200**”. **Page 30:** Second sentence in paragraph four, text in graphic and last paragraph corrected. **Page 31:** Paragraph two: Replace “DM 1000 to **1799**” with “DM 1000 to **1899**” in both instances. **Page 37:** Replace “RS/CS” with “RTS/CTS,” “Interlec HEX” with “Intel HEX,” and “Motorola S” with “Motorola S-Record.” **Page 41:** Paragraphs five and seven: Replaced “250 #Gms” with “250 ms”. Paragraph eight corrected. **Page 42:** Paragraph five, line one corrected. **Page 46:** Last paragraph, last sentence corrected. **Page 70:** Paragraph five, diagram, and paragraph six corrected. **Page 76:** Paragraph four, line three corrected. **Page 81:** Numerical ranges under and in paragraphs one and three corrected. **Page 83:** Numerical figures in paragraph four and six corrected. **Page 98:** Ladder symbols, paragraph two, paragraph four, and paragraph five corrected. **Page 101:** Text in first graphic corrected and paragraph two eliminated. **Page 105:** First sentence, line one of paragraph four, line three of paragraph four corrected and diagram added. **Page 110:** TCMP(85) removed (not supported by Mini H-type PCs). **Page 116:** Paragraph one: “to 7-segment display data” eliminated from lines three and four. **Page 118:** Note added to limitations to STH(66). **Page 119:** Error Flag description corrected. **Page 120:** Second diagram: Replaced “Digit number: **0123**” with “Digit number: **3210**”. **Page 125:** First paragraph of *Section 5–15*: Add **INC(38)** to the BCD calculation instructions. **Page 128:** Sentence including ADDL(54) eliminated (not supported by Mini H-type PCs). **Page 131:** Last diagram: The remainder should follow the quotient. **Page 132:** Last diagram: Replace “Dd: HR 09” with “Dr: HR 09”. **Page 140:** Sentence including INT(89) eliminated (not supported by Mini H-type PCs). **Page 152:** Leftmost column of table corrected. **Page 153:** Section 5–20–4 rewritten: **Page 155:** Last paragraph, last line: Replace “...places the result in **D**.” with “...places the result in **R**.” **Page 166:** ROOT(72) and FUN(89) eliminated from table (not supported by Mini H-type PCs). **Page 184:** Paragraph nine: Replace “402180” with “4021”, and “2180” with “21”. **Page 187:** Last key explanation corrected (block programming instructions not supported by Mini H-type PCs). **Page 189:** Remove the last sentence. **Page 196:** Paragraph four and paragraph six corrected to remove reference to block programming instructions. **Page 224:** Replace “INTELLEC HEX” with “Intel’s HEX” and “MOTOROLA [S]” with “Motorola’s S-Record”. **Pages 225, 226:** Add a seventh step added to procedures and graphic. **Page 228:** Fourth paragraph from the bottom: Eliminate “bits 08 to 15” from line two. **Page 230:** Numeric limitations corrected in first and second diagrams. **Page 258:** Table: Replaced “AR 0000 to AR **0015**” with “AR 0000 to AR **0007**” **Page 264:** Corrected model number for Cassette Recorder Connecting Cable to SCY-P0R-PLG01. **Page 289:** Program Read Protect/Clear eliminated (not supported by Mini H-type PCs). **Pages 290, 291:** Cassette Tape Operations eliminated (cassette tape recorder requires interfacing via GPC). **Page 291:** PROM Writer operations are applicable to non-OMRON products, i.e. Intel’s HEX and Motorola’s S-Record. **Page 294:** INT(89) eliminated (not supported by Mini H-type PCs). **Page 298:** Table, row five: Replace “DM 1000 to DM **1799**” with “DM 1000 to DM **1899**”. **Page 316:** Replace “RS/CS” with “RTS/CTS,” “Interlec HEX” with “Intel HEX,” and “Motorola S” with “Motorola S-Record.”
2A	August 1991	**Pages 86, 87, 110, 276** Jump number range corrected to 00 to 49. **Pages 171, 287** Subroutine number range corrected to 00 to 49. **Page 116** TC number range clarified.
2B	December 1991	**Page 196** Time required for overseeing changed to 2.9 ms. **Page 197** Times required for processing changed. **Pages 230, 231** Rewritten to remove “Single-link System” protocol.

Revision code	Date	Revised content
3	June 1992	Manual revised for V1 of CPUs and to include C60H. **Page 51** “NOT” added after “LD” in top table of mnemonic code. **Page77** First digit in last display corrected to “0.” **Page 80** Condition for IR 00105 corrected to NO and that for IR 00104 corrected to NC in *Before Insertion* portion of example ladder diagram. **Page 81** Bottom table of mnemonic code corrected. **Page 87** “5” removed from end of last operand in top table of mnemonic code. **Page 93** Last operand in bottom table of mnemonic code corrected. **Page 109** Following added to bottom table of mnemonic code: 00008 LD 00100. **Page 113** Last operand in bottom table of mnemonic code corrected. **Page 116** Second “LD” corrected to “AND” in table of mnemonic code. **Page 132** Function code for ASR corrected in illustration. **Page 131** “00514” corrected to “05014” and “@SFT(10)” corrected to SFTR(84) in table of mnemonic code. **Page 157** “AND(30)” corrected to “ADD(30)” in table of mnemonic code. **Page 165** “25505” corrected to “25504” and “#0000” corrected to ”#0001” in operands in table of mnemonic code. **Page 199** Execution times corrected for TIMH(15). **Page 200** OFF execution times for RDM(60) corrected to the same values as the ON execution times. **Page 209** Completion Flag ON mark added to display. **Page 211** Caret added to the third display from the bottom. **Page 213** Displays corrected. **Page 218** “0001” corrected to “0123” in the bottom display. **Page 221** Displays corrected. **Page 237** Response for ERROR READ corrected for bits 2, 3, and 4. **Page 240** Format for timer/counter data changed to BCD for PV READ, and maximum timer/counter number corrected to 0511 for SV READ 1/2. **Pages 242 and 243** Format for data area contents corrected to hexadecimal. **Page 245** Maximum timer/counter number corrected to 0511 for SV CHANGE 2. **Page 247** Code to clear forced set/reset status corrected from “0100” to “1000.” **Page 249** “C120 or C20” changed to “C120 or C50” for PC MODEL READ. **Page 271** Same as for page 199. **Page 273** Same as for page 200. **Page 297 and 298** Flag addresses corrected in table column headers. **Page 313** Direction of arrows reversed for E7 and F7.
4	March 1993	**Page 128** “Example” changed to “Example 1.” **Page 129** Example 2 added. **Page 294:** EPROM model number ROM-I_-B has been added to the note. **Pages 295, 296:** Peripheral Devices table slightly altered to reflect the list in the Installation Guide. **Page 296:** Programming Console Base Unit specifications were corrected.
4A	July 1994	Address change. Scan time changed to Cycle time throughout the manual. **Page 13:** Input bits for C60H Expansion I/O Units corrected. **Page 333:** Output bits for C60H CPUs corrected. **Page 334:** Output bits for C60H Expansion I/O Units corrected.