

W245-E1

C20HB-TS

Actuator Controller

INSTRUCTION MANUAL

OMRON

C20HB-TS Actuator Controller


Instruction Manual


Produced November 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 relatively 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.

© OMRON, 1994

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

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

TABLE OF CONTENTS

SECTION 1

Introduction	1
1-1 Features of the C20HB-TS	2
1-2 Overview	3
1-3 The Origins of PC Logic	3
1-4 PC Terminology	4
1-5 OMRON Product Terminology	4
1-6 Overview of PC Operation	4

SECTION 2

Hardware Considerations	7
2-1 PC Components	8
2-2 PC Configuration	10
2-3 Specifications	12
2-4 Installation Environment	20
2-5 Wiring	22

SECTION 3

Memory Areas	29
3-1 Introduction	30
3-2 IR (Internal Relay) Area	31
3-3 SR (Special Relay) Area	31
3-4 AR (Auxiliary Relay) Area	33
3-5 DM (Data Memory) Area	37
3-6 HR (Holding Relay) Area	42
3-7 TC (Timer/Counter) Area	42
3-8 LR (Link Relay) Area	42
3-9 TR (Temporary Relay) Area	43

SECTION 4

Writing and Entering Programs	45
4-1 Basic Procedure	46
4-2 Instruction Terminology	46
4-3 Ladder Diagrams	47
4-4 The Programming Console	50
4-5 Preparation for Operation	54
4-6 Inputting, Modifying, and Checking the Program	58
4-7 Controlling Bit Status	76
4-8 Work Bits (Internal Relays)	77
4-9 Programming Precautions	79
4-10 Program Execution	81

SECTION 5

Instruction Set	83
5-1 Notation	84
5-2 Instruction Format	84
5-3 Data Areas, Definer Values, and Flags	84
5-4 Differentiated Instructions	86
5-5 C20HB-TS Instruction Set	87
5-6 Ladder Diagram Instructions	94
5-7 Bit Control Instructions	97
5-8 Timer and Counter Instructions	101

TABLE OF CONTENTS

SECTION 6

Program Execution Timing 113

6-1	Cycle Time	114
6-2	Instruction Execution Times	117
6-3	I/O Response Time	121
6-4	Host Link Response Time	122

SECTION 7

Program Debugging and Execution 125

7-1	Displaying and Clearing Error Messages	126
7-2	Monitoring Operation and Modifying Data	127
7-3	C250HL-PRO31-TS PROM Writer Operations	144

SECTION 8

RS-485 Interface 151

8-1	RS-485 System Configuration and Settings	152
8-2	Connection to a Host Computer's RS-232C Port	154
8-3	RS-485 Interface Flags and Control Bits	154
8-4	RS-485 Communications Protocol	155
8-5	Host Link Commands and Responses	159
8-6	Command Levels	176

SECTION 9

Troubleshooting 177

9-1	Alarm Indicators	178
9-2	Programmed Alarms and Error Messages	178
9-3	Reading and Clearing Errors and Messages	178
9-4	Error Messages	179
9-5	Error History Function	181
9-6	Host Link Error Processing	181

Appendices 183

A.	Error and Arithmetic Flag Operation	183
B.	Word Assignment Recording Sheets	185
C.	Program Coding Sheet	191
D.	Data Conversion Table	193
E.	Parameter Area Coding Charts	195

Revision History 197

About this Manual:

This manual describes the installation and operation of the C20HB-TS Actuator Controller and includes the sections described below. The OMRON C20HB-TS offers a simple but effective way to automate processing. Manufacturing, assembly, packaging, and many other processes can be automated to save time and money.

Please read this manual carefully and be sure you understand the information provided before attempting to install and operate the C20HB-TS.

Section 1 Introduction compares the C20HB-TS to the Mini H-type PCs and 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.

Section 2 Hardware Considerations explains basic aspects of the overall PC configuration and describes the components that are referred to in other sections of this manual. A complete set of specifications and instructions for installing and wiring the PC are also provided.

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 also provides information on System DM, a special area in the C20HB-TS that provides the user with flexible control of PC operating parameters.

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

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

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

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

Section 8 RS-485 Interface describes the modes, settings, and procedures essential for making use of the built-in RS-485 interface. It also lists all of the commands that can be downloaded from a host computer connected to the RS-485 interface.

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 coding sheets 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 an introduction to the C20HB-TS, 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.

1-1	Features of the C20HB-TS	2
1-2	Overview	3
1-3	The Origins of PC Logic	3
1-4	PC Terminology	4
1-5	OMRON Product Terminology	4
1-6	Overview of PC Operation	4

1-1 Features of the C20HB-TS

The C20HB-SC001-TS PC (hereafter referred to as the C20HB-TS) is based on the Mini H-type PCs and has many of the same features.

Instruction Set

In most cases, programs written for Mini H-type and C200H PCs can be used in the C20HB-TS with minor modifications. The instruction set of the C20HB-TS is smaller than those of the Mini H-type and C200H PCs, so instructions that aren't supported by the C20HB-TS will not be executed in programs imported from Mini H-type and C200H PCs. Refer to *Section 5 Instruction Set* for a complete list of the C20HB-TS instruction set.

Instructions are executed at the same high-speed as they are in Mini H-type PCs. Basic ladder diagram instructions are executed in as little as 0.75 μ s.

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.

Peripheral Device	Features
Programming Console (C200H-PRO27-E)	The simplest form of programming device for OMRON PCs. It is connected directly to the CPU without requiring a separate interface.
Programming Console (C250HL-PRO31-TS-E)	Has the same programming and monitoring functions as the C200H-PRO27-E and is also equipped with a built-in PROM Writer.
Data Access Console (C200H-DAC01-E)	The DAC can be used to monitor or change data in the TC, IR and SR areas.
Factory Intelligent Terminal (FIT10/FIT20)	The FIT is an OMRON computer with specially designed software that allows you to perform all of the operations that are available with LSS. Programs can also be output directly to an EPROM chip, floppy disk drive, or printing device without any additional interface.
Ladder Support Software (C500-SF□10-V4)	LSS is designed to run on IBM AT/XT compatibles to enable all of 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 screen, making confirmation and modification quick and easy. Syntax checks may also be performed on the programs before they are downloaded to the PC.

Comparing the C20HB-TS and Mini H-type PCs

The following table compares the capabilities of the C20HB-TS with those of the Mini H-type PCs.

Function		C20HB-TS	Mini H-type
User Memory	Size	8K (6974 words for the program)	4K (2878 words for the program)
	IC type	EEPROM	RAM, EEPROM, or EPROM
Serial Interface	Type	RS-485	RS-232C
	Baud rate	300 to 19,200 baud	Baud rate: 300 to 9,600 baud
	Modes	Host Link only	Host Link, Download/Upload, or ASCII I/O
	Host Link unit numbers	00 to 15 (set by DIP switch)	00 to 31 (set in System DM)
I/O points		128 ¹	20 to 240
High-speed timers (Interrupt-type)		16 permitted	4 permitted
Instruction set		78 instructions	142 instructions
High-speed counters		None	One permitted
Battery backup		None ²	Built-in ²
Programming Console languages		English or Japanese	English, Japanese, French, German, Italian, or Spanish
C250HL-PRO31-TS-E Programming Console		Supported (Built-in PROM Writer.)	Not supported
C200H Special I/O Units		Not supported	Most supported

Note 1. The 128 I/O points include: 48 24-VDC Inputs, 48 Transistor Outputs, and 32 Relay Outputs.

2. The C20HB-TS has a capacitor backup that maintains counters, DM, HR, and AR data for 5 minutes. Mini H-type PCs have a battery backup that maintains counters, DM, HR, and AR data for up to 5 years at 25°C.

1-2 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 C20HB-TS, like the other C-series PCs, uses a form of PC logic called ladder-diagram programming. This manual is written to explain ladder-diagram programming and to prepare the reader to program and operate the C20HB-TS.

1-3 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.

1-4 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.

1-5 OMRON Product Terminology

OMRON products are divided into several functional groups that have generic names. The term Unit is used to refer to all of the OMRON PC products.

Product groups include Programming Devices and Peripheral Devices.

1-6 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. On paper, assign all input and output devices to I/O points on the PC and determine which I/O bits will be allocated to each.

3. 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*)
4. Input the program and all required operating parameters into the PC. (Refer to *Section 7 Program Debugging and Execution*)
5. 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*)
6. Wire the PC to the controlled system. This step can actually be started as soon as step 3 has been completed.
7. 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. Refer to *3-2 I/R 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.

SECTION 2

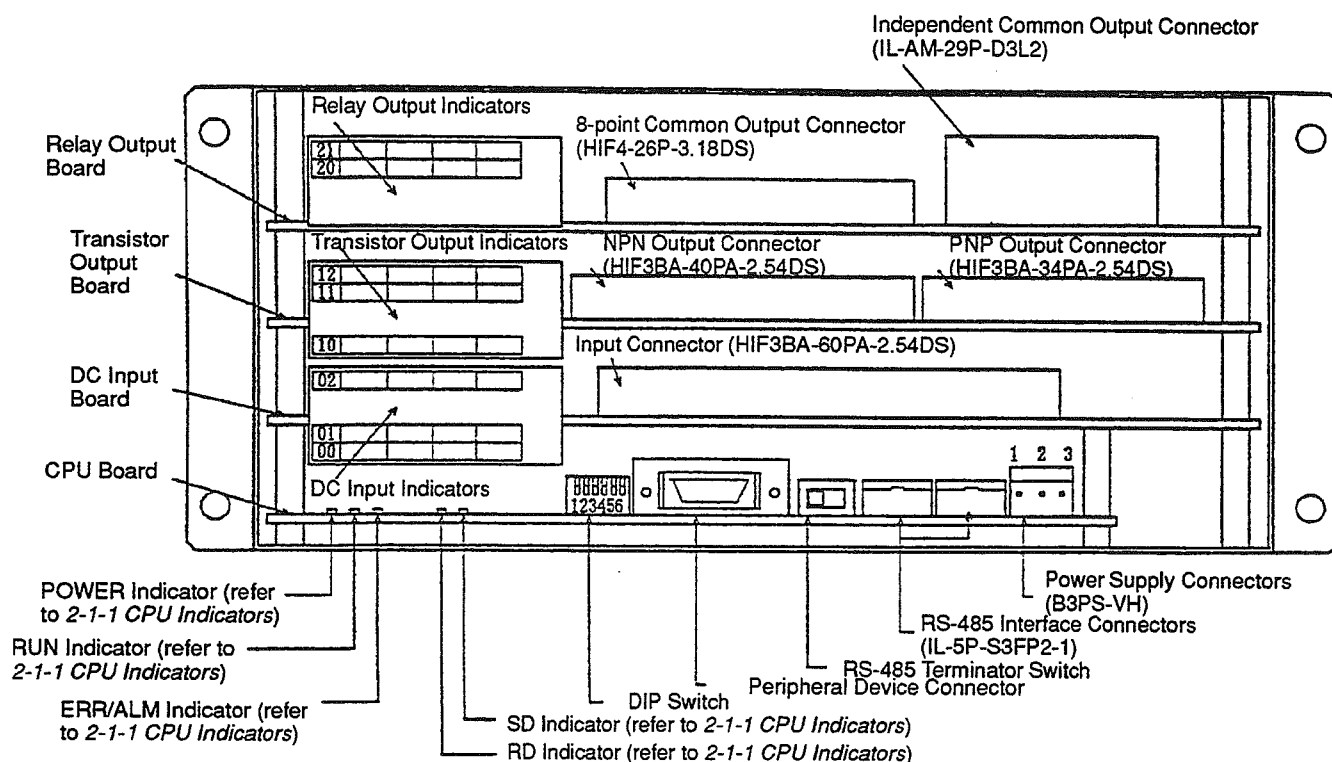
Hardware Considerations

This section provides information on hardware aspects of the C20HB-TS, including PC components, PC configuration, and specifications.

2-1	PC Components	8
2-1-1	CPU Indicators	8
2-1-2	I/O Indicators	9
2-1-3	Switch Settings	9
2-2	PC Configuration	10
2-3	Specifications	12
2-3-1	General Ratings	12
2-3-2	CPU Specifications	13
2-3-3	DC Input Specifications	14
2-3-4	PNP Transistor Output Specifications	15
2-3-5	NPN Transistor Output Specifications	16
2-3-6	Relay Output Specifications (8-point Commons)	17
2-3-7	Relay Output Specifications (Independent Commons)	18
2-3-8	RS-485 Interface Specifications	19
2-4	Installation Environment	20
2-4-1	PC Location	20
2-4-2	PC Dimensions	21
2-5	Wiring	22
2-5-1	General Wiring Precautions	22
2-5-2	Input Wiring Precautions	23
2-5-3	Output Wiring Precautions	24

2-1 PC Components

The following diagram shows the components on the front of the C20HB-TS.



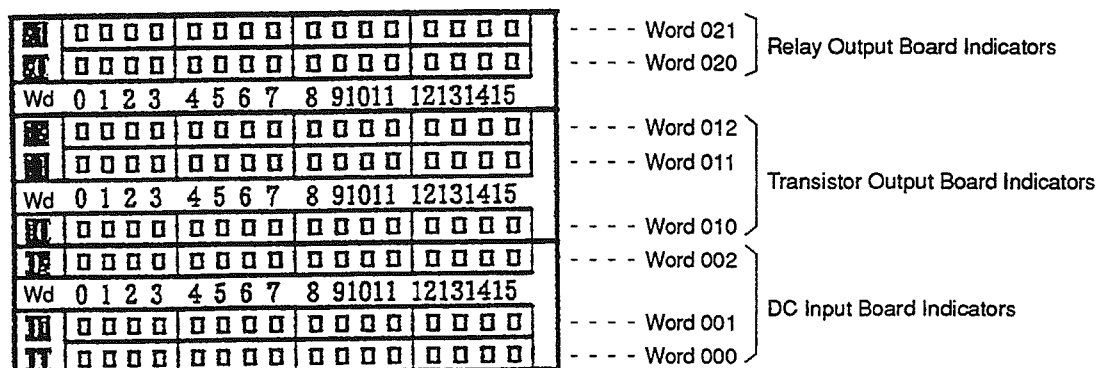
2-1-1 CPU Indicators

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

Indicator	Color	Function
POWER	Green	Lights when power (5 VDC) is supplied to the CPU.
RUN	Green	Lights when the CPU is operating normally.
ERR/ALM	Red	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. CPU operation will be stopped.
SD	Orange	Lights when transmitting data through the RS-485 interface.
RD	Orange	Lights when receiving data through the RS-485 interface.

2-1-2 I/O Indicators

I/O indicators reflect the ON/OFF status of I/O points. The following diagram shows the location of the indicators in detail.

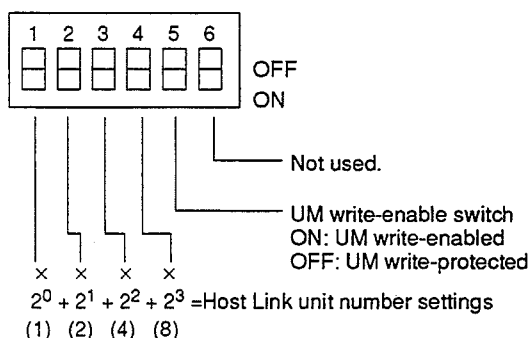


2-1-3 Switch Settings

There are two switches on the front of the C20HB-TS, the DIP switch and the RS-485 terminator switch.

DIP Switch

Pins 1 through 4 of the DIP switch are used to set the Host Link unit number and pin 5 is the UM write-enable switch.



The unit number is set in binary on pins 1 to 4.

Pin	Value
1	0 or 1 (2^0)
2	0 or 2 (2^1)
3	0 or 4 (2^2)
4	0 or 8 (2^3)

RS-485 Terminator Switch

Turn this switch on to connect the termination resistance. This should be done only on the last PC from the host computer.

Power Supply Connector

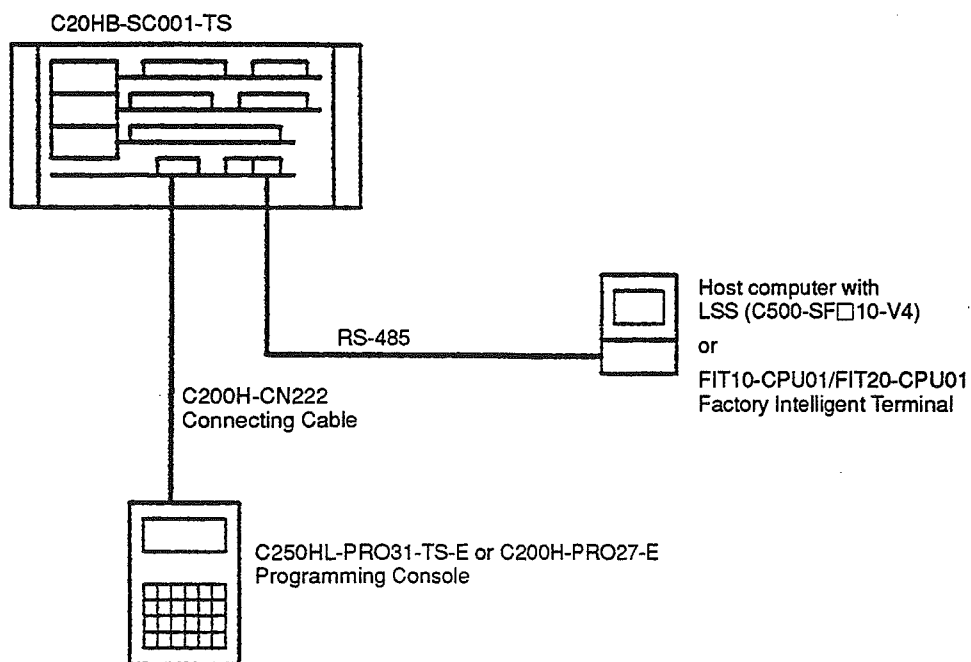
The B3PS-VH power supply connector requires a connector with a VHR-3N/VHR-3M housing and SVH-21T-P1.1 connector. The power supply is 5 VDC with a ground.

Pin	Connection	Location
1	5 VDC	
2	0 VDC	
3	FG	

2-2 PC Configuration

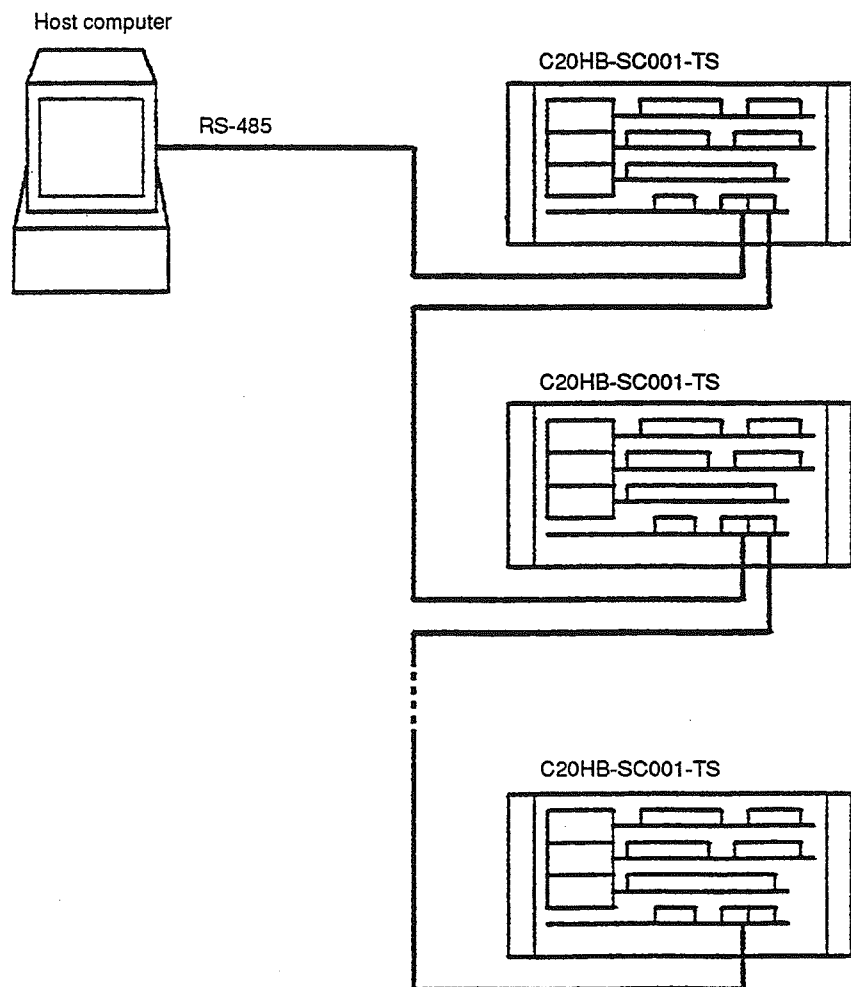
Peripheral Devices

The following diagram shows the Peripheral Devices which can be connected to the C20HB-TS and the location of their connectors.



Host Link System

You can use the RS-485 interface to connect a host computer to up to 16 C20HB-TS PCs, as shown in the following diagram. The total cable length can be up to 500 m.



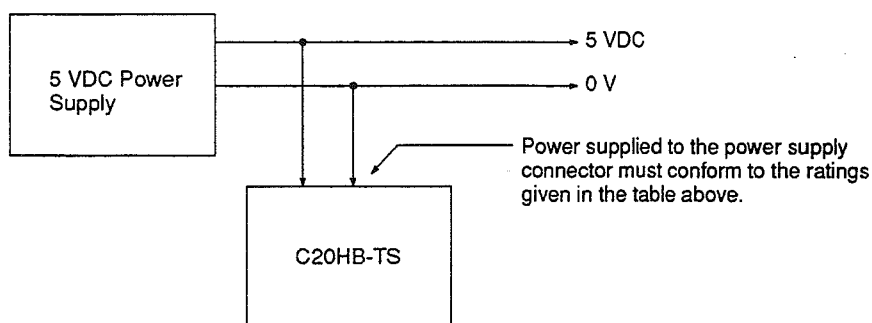
2-3 Specifications

This section provides general specifications and specifications for each PC component.

2-3-1 General Ratings

Item		Rating
Supply voltage		5 VDC $\pm 5\%$, 2A. Ripple must be less than 100 mV _{P-P} (See note.)
Power consumption		10 W max. (5 VDC, 2A)
Insulation resistance		20 M Ω min. (at 500 VDC) between I/O and FG terminals
Dialectic strength		2,000 VAC for 1 minute between AC and FG terminals (leakage current 10 mA max.) 1,000 VAC for 1 minute between DC and FG terminals (leakage current 10 mA max.)
Noise immunity	Power supply	20E (100 V _{P-P}) in normal mode, 1000 V _{P-P} in common mode Pulse width: 100 ns to 1 μ s, rise time 1 ns
	I/O	20E (480 V _{P-P}) in normal mode, 1000 V _{P-P} in common mode Pulse width: 100 ns to 1 μ s, rise time 1 ns
Vibration		10 to 35 Hz, 1 mm double amplitude, in X, Y, and Z directions: 2 hours each.
Shock		10 G (about 98 m/s ²) in X, Y, and Z directions, 3 times each
Ambient temperature		Operating 0° to 55°C Storage -20° to 65°C
Humidity		35% to 85% (non-condensing)
Atmosphere		No corrosive gases
Grounding		Less than 100 Ω
Construction		Conforms to IEC IP-30 (when panel-mounted)
Dimensions		235 × 80 × 150 mm (W × H × D)

Note Supply power to the C20HB-TS as shown in the following diagram.



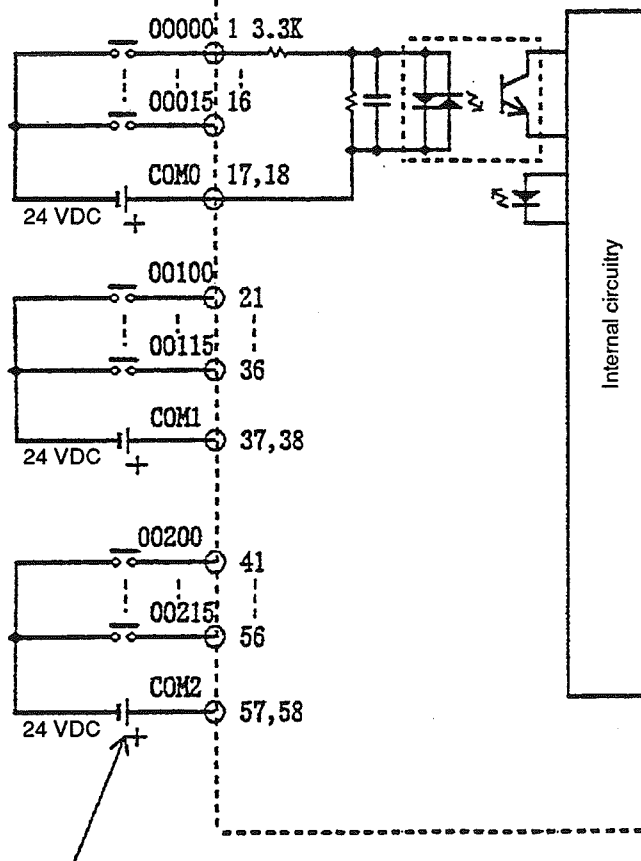
2-3-2 CPU Specifications

Item	Specification
Control Method	Stored program
I/O Control Method	Cyclical scan with interrupt processing
Programming method	Ladder diagram
Instruction length	1 address/instruction, 1 to 4 words/instruction
Number of instructions	78 (12 basic instructions, 66 special instructions)
Execution time	Basic instructions 0.75 to 2.25 μ s
Memory Capacity	6,974 words (EEPROM)
I/O bits	128
IR bits	3,824
SR bits	136 (Normally ON, normally OFF, first scan, 0.1 s clock, 0.2 s clock, 1.0 s clock, etc.)
HR bits	1,600 (HR 0000 through 9915)
TR bits	8 (TR 0 through 7)
AR bits	448 (AR 0000 through 2715)
LR bits	1,024 (LR 0000 through 6315)
Timers/Counters	512 (TIM/CNT 000 through 511) TIMs 0 through 999.9 s TIMHs 0 through 99.99 s CNT 0 through 9999 counts
DM words	Read/write 1000 words (DM 0000 through DM 0999) Read only 1000 words (DM 1000 through DM 1999) Words DM 0900 through DM 0999 and DM 1900 through DM 1999 are allocated as the system setting area.
Memory protection	The data in HR, AR, CNT, and DM areas has a capacitor backup which will maintain the data for 5 minutes at 25°C.
Self diagnostic features	CPU failure (watchdog timer) I/O bus failure Host Link error Memory error, etc.
Program check	Program checked (at the start of program execution) No END(01) instruction, Instruction errors The program can also be checked with a Programming Console. The program can be checked at three levels.

2-3-3 DC Input Specifications

Item	Specification
Input voltage	24 VDC +10%/-15%
Input impedance	3.3 k Ω
Input current	7 mA typical at 24 VDC
ON voltage	16 VDC min.
OFF voltage	5 VDC max.
ON delay time	2.5 ms max.
OFF delay time	2.5 ms max.
Number of circuits	48 (3 circuits with 16-point commons)
Input indicators	LED indicators (orange)
External connectors	Board mounting plug: HIF3BA-60PA-2.54DS (Hirose) Connector socket: HIF3BA-60D-2.54R (Hirose) Wire: AWG#28 (The external connector socket, wire, and assembly tools are not included.)

Circuit configuration



The polarity of the power supply for sensors or switches can be in either direction.

NC	C2	14	12	10	08	06	04	02	00	NC	C1	14	12	10	08	06	04	02	00	NC	C0	14	12	10	08	06	04	02	00
59	57	55	53	51	49	47	45	43	41	39	37	35	33	31	29	27	25	23	21	19	17	15	13	11	9	7	5	3	1
60	58	56	54	52	50	48	46	44	42	40	38	36	34	32	30	28	26	24	22	20	18	16	14	12	10	8	6	4	2
NC	C2	15	13	11	09	07	05	03	01	NC	C1	15	13	11	09	07	05	03	01	NC	C0	15	13	11	09	07	05	03	01

Word 02

Word 01

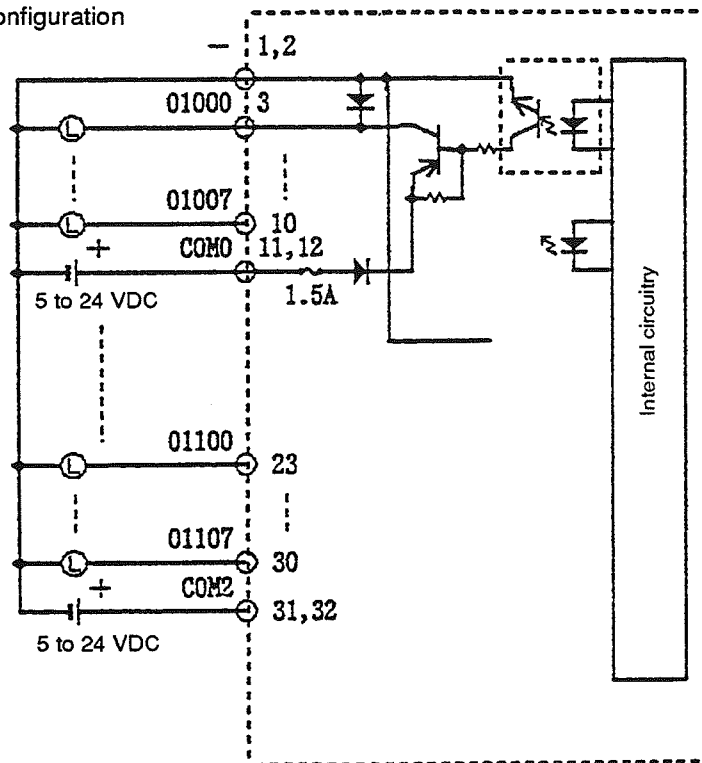
Word 00

Refer to page 27 for details on the number of I/O points that can be on at the same time.

2-3-4 PNP Transistor Output Specifications

Item	Specification
Max. switching capacity	16 mA/4.5 VDC to 100 mA/26.4 VDC (800 mA/common)
Leakage current	0.1 mA max.
Residual voltage	1.5 mV max.
ON delay time	0.2 ms max.
OFF delay time	0.6 ms max.
Number of circuits	24 points (3 circuits with 8-point commons)
External power supply	5 to 26.4 VDC, 80 mA min. (3.2 mA × number of ON outputs)
Output indicators	LED indicators (orange)
Fuses	1 fuse/common (125 V, 1.5 A)
External connectors	Board mounting plug: XG4A-3434 (OMRON) or HIF3BA-34PA-2.54DS (Hirose) Connector socket: XG4M-3430+XG4T-3404 (OMRON) or HIF3BA-34D-2.54R (Hirose) Wire: XY3A-340□ (OMRON) or equivalent (AWG#28) (The external connector socket, wire, and assembly tools are not included.)

Circuit configuration



NC	C2	06	04	02	00	C1	14	12	10	08	C0	06	04	02	00	-
33	31	29	27	25	23	21	19	17	15	13	11	9	7	5	3	1
34	32	30	28	26	24	22	20	18	16	14	12	10	8	6	4	2
NC	C2	07	05	03	01	C1	15	13	11	09	C0	07	05	03	01	-

Word 11

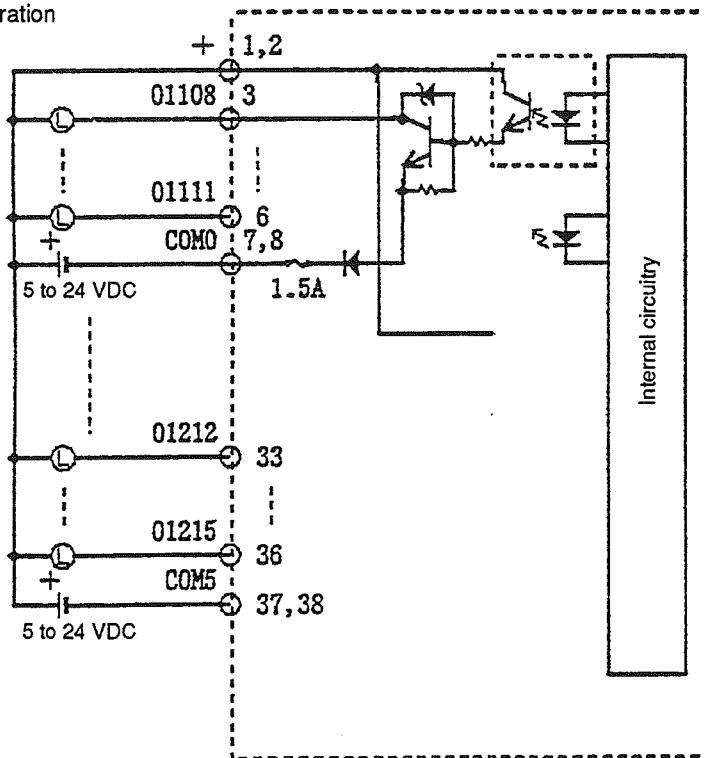
Word 10

Refer to page 27 for details on the number of outputs that can be on at the same time.

2-3-5 NPN Transistor Output Specifications

Item	Specification
Max. switching capacity	16 mA/4.5 VDC to 100 mA/26.4 VDC (400 mA/common)
Leakage current	0.1 mA max.
Residual voltage	1.5 mV max.
ON delay time	0.2 ms max.
OFF delay time	0.6 ms max.
Number of circuits	24 points (6 circuits with 4-point commons)
External power supply	5 to 26.4 VDC, 80 mA min. (3.2 mA × number of ON outputs)
Output indicators	LED indicators (orange)
Fuses	1 fuse/common (125 V, 1.5 A)
External connectors	Board mounting plug: XG4A-4034 (OMRON) or HIF3BA-40PA-2.54DS (Hirose) Connector socket: XG4M-4030+XG4T-3404 (OMRON) or HIF3BA-40D-2.54R (Hirose) Wire: XY3A-400□ (OMRON) or equivalent (AWG#28) (The external connector socket, wire, and assembly tools are not included.)

Circuit configuration



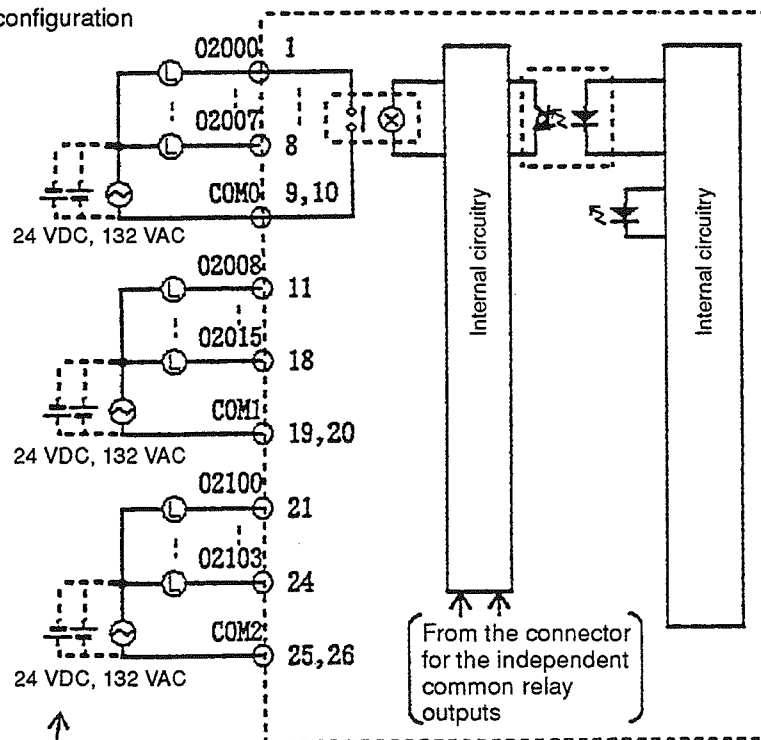
NC	C5	14	12	C4	10	08	C3	06	04	C2	02	00	C1	14	12	C0	10	08	+
39	37	35	33	31	29	27	25	23	21	19	17	15	13	11	9	7	5	3	1
40	38	36	34	32	30	28	26	24	22	20	18	16	14	12	10	8	6	4	2
NC	C5	15	13	C4	11	09	C3	07	05	C2	03	01	C1	15	13	C0	11	09	+

Refer to page 27 for details on the number of outputs that can be on at the same time.

2-3-6 Relay Output Specifications (8-point Commons)

Item	Specification
Max. switching capacity	132 VAC/0.2 A, 24 VDC/0.2 A
Min. switching capacity	5 mA/24 VDC
ON delay time	15 ms max.
OFF delay time	15 ms max.
Number of circuits	20 points (2 circuits with 8-point commons, 1 circuit with a 4-point common)
External power supply	24 VDC \pm 10%, 200 mA min. (10 mA \times number of ON outputs) (Power is supplied from the connector for the independent common relay outputs.)
Output indicators	LED indicators (orange)
Fuses	None
Relay life	Resistive load: 100,000 times electrical, 1,000,000 times mechanical
External connectors	Board mounting plug: HIF4-26P-3.18DS (Hirose) Connector socket: HIF4-26P-3.18R (Hirose) Wire: AWG#28 (The external connector socket, wire, and assembly tools are not included.)

Circuit configuration



The polarity of the power supply for the load can be in either direction.

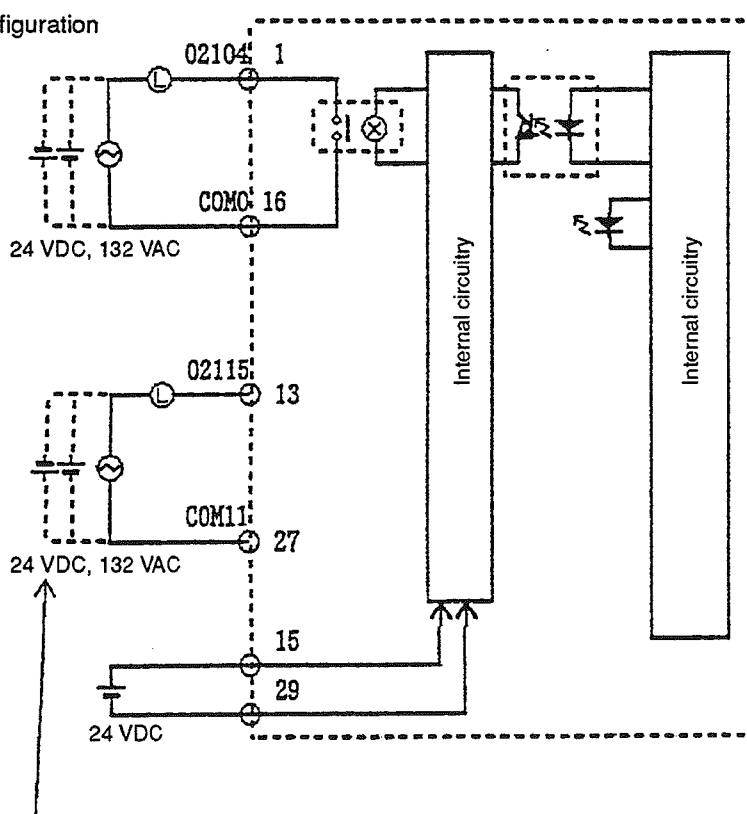
C2	02	00	C1	14	12	10	08	C0	06	04	02	00
25	23	21	19	17	15	13	11	9	7	5	3	1
26	24	22	20	18	16	14	12	10	8	6	4	2
C2	03	01	C1	15	13	11	09	C0	07	05	03	01
Word 21				Word 20								

Refer to page 27 for details on the number of outputs that can be on at the same time.


2-3-7 Relay Output Specifications (Independent Commons)

Item	Specification
Max. switching capacity	132 VAC/0.2 A, 24 VDC/0.2 A
Min. switching capacity	5 mA/24 VDC
ON delay time	15 ms max.
OFF delay time	15 ms max.
Number of circuits	12 points (12 circuits with independent commons)
External power supply	24 VDC $\pm 10\%$, 200 mA min. (10 mA \times number of ON outputs)
Output indicators	LED indicators (orange)
Fuses	None
Relay life	Resistive load: 100,000 times electrical, 1,000,000 times mechanical
External connectors	Board mounting plug: IL-AM-29P-D3L2 (JAE) Connector socket: IL-AM-29S-D3C1 (JAE) Connector contact: IL-C1-5000 (JAE) Wire: AWG#22 to #18, finished dimension 2.4 mm max. (The external connector socket, wire, and assembly tools are not included.)

Circuit configuration



The polarity of the power supply for the load can be in either direction.

24	VNC	15	14	13	12	11	NC	10	09	08	07	06	05	04	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
29	28	27	26	25	24	23		22	21	20	19	18	17	16	
0	V	NC	C11	C10	C9	C8	C7		C6	C5	C4	C3	C2	C1	C0

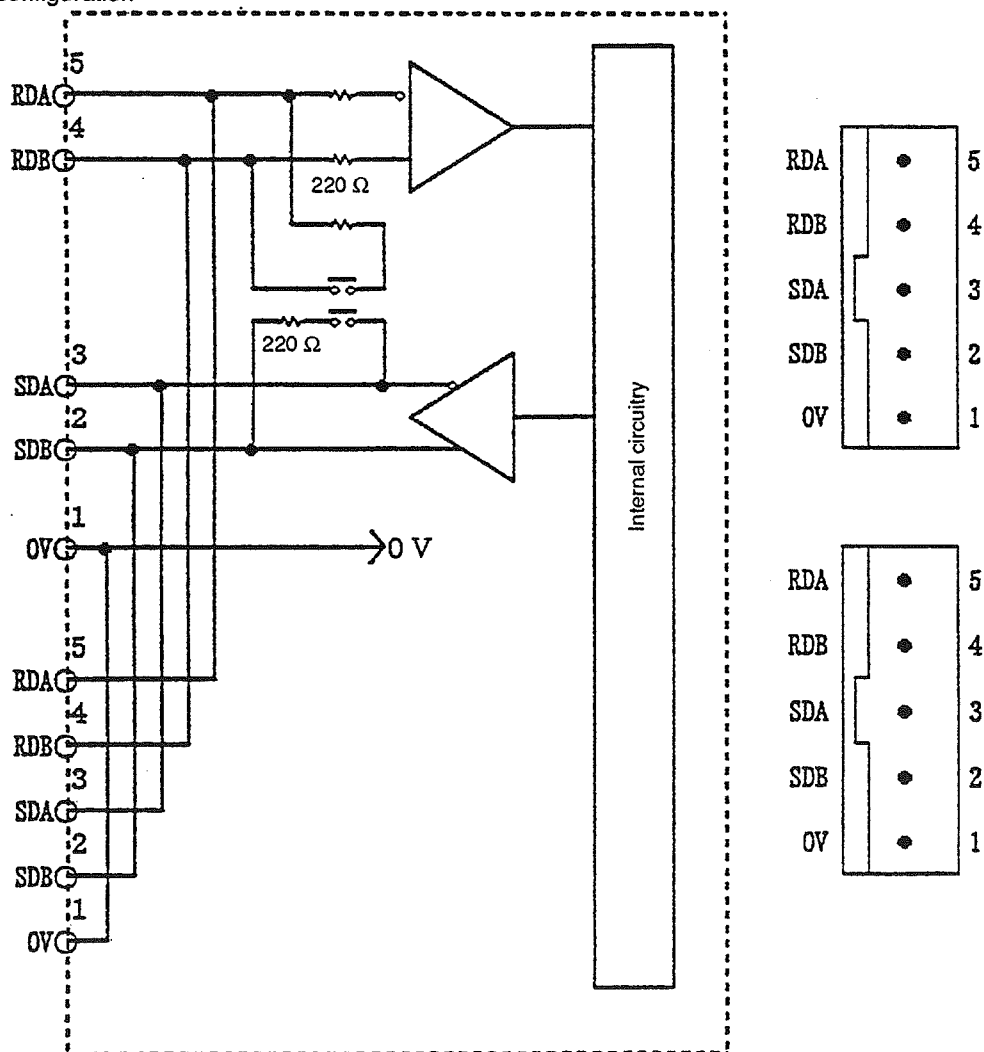
Refer to page 27 for details on the number of outputs that can be on at the same time.

Word 21

2-3-8 RS-485 Interface Specifications

Item	Specification
Interface	Conforms to RS-485 standards (internal circuitry and non-isolated)
Communication method	Half-duplex (4-wire type)
Synchronization	Start-stop method
Transmission rate	300 to 19,200 baud (300/600/1200/2400/4800/9600/19,200)
Transmission distance	500 m max.
Error detection	Parity check (positive, negative, none), and FCS
Transmission coding	7-unit ASCII or 8-unit JIS
Allowed number of nodes	16 max.
Transmission indicators	Two orange LED indicators: SD (sending data) and RD (receiving data)
External connectors	Board mounting plug: IL-5P-S3FP2-(N)-1 (JAE) Connector socket: IL-5S-S3L-(N) (JAE) Connector contact: IL-C2-1-10000 (JAE) Wire: AWG#28 to #22, wire cross-section 0.08 to 0.33 mm, outer diameter 1.0 to 1.7 mm (The external connector socket, wire, and assembly tools are not included.)

Circuit configuration



2-4 Installation Environment

This section provides information on the necessary environment and conditions for installing the PC.

2-4-1 PC Location

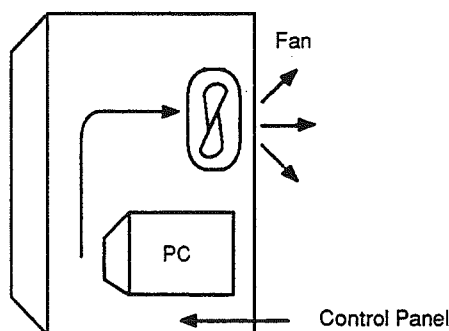
The boards that make up the C20HB-TS are equipped with LED indicators to show the operating status of the PC and the ON/OFF status of I/O points. Be sure to install the PC so that these indicators are visible.

For safety and easy maintenance, install the PC as far as possible from high-voltage and high-power wiring.

Cooling Requirements

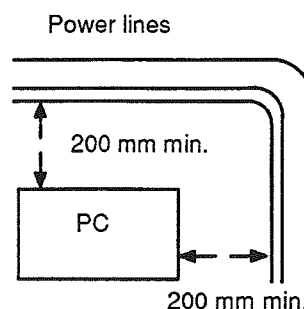
The operating temperature range for the C20HB-TS is 0° to 55°C. There are several factors to consider in order to ensure that the PC does not overheat.

- 1, 2, 3... 1. Provide enough clearance around the PC to provide enough room for I/O wiring and additional room to ensure that the I/O wiring does not restrict cooling airflow.
2. Do not install the PC above equipment that generates large amounts of heat, such as heaters, transformers, or large capacitors.
3. A cooling fan is not always necessary, but may be needed if the PC is mounted in a warm or enclosed area or over a source of heat. Although it is best to avoid installing the PC in a warm area, use a cooling fan, as shown in the following illustration, to maintain the ambient temperature within specifications.



Noise Prevention

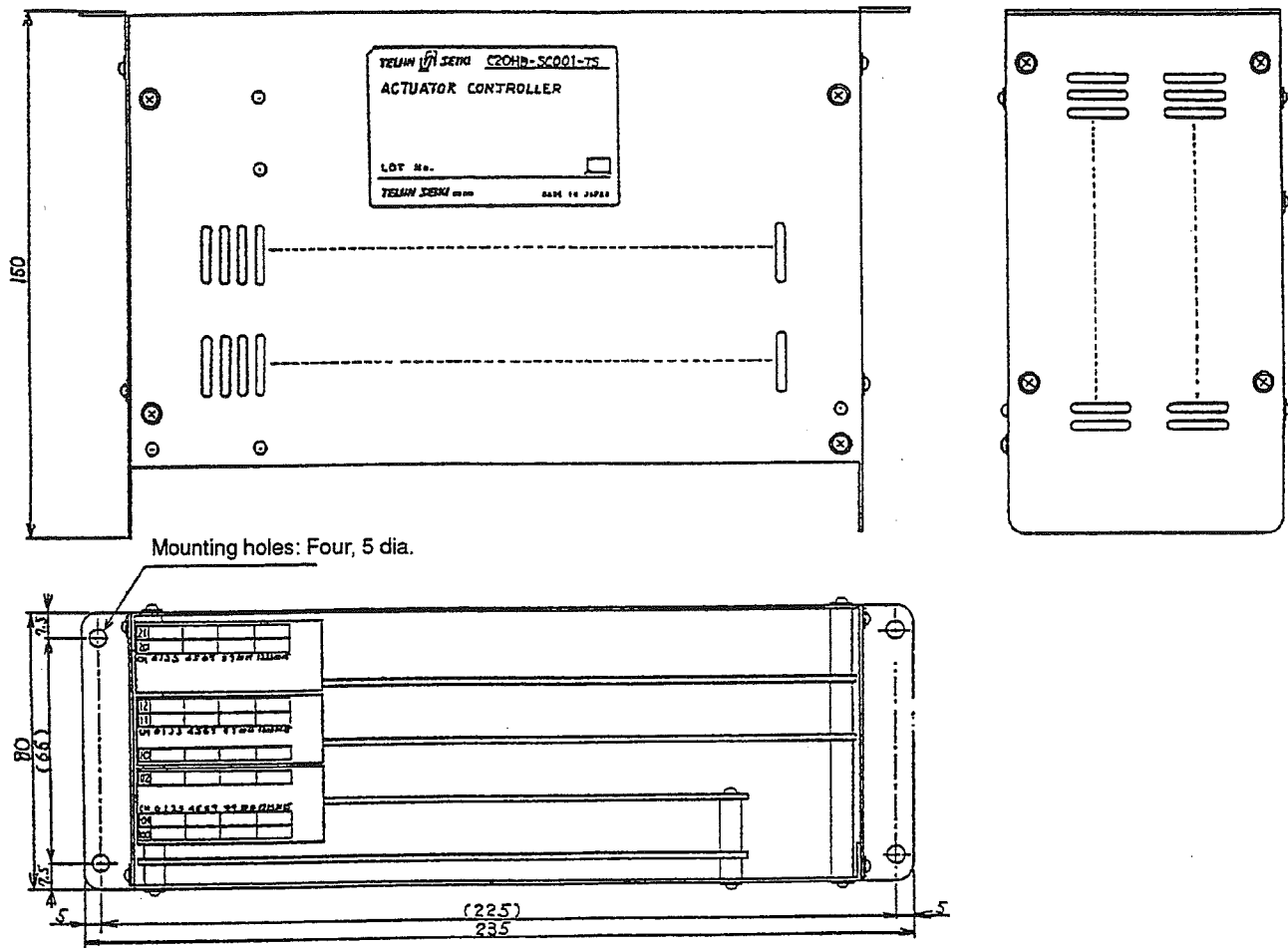
Avoid mounting the PC close to high-power equipment, and make sure the point of installation is at least 200 mm away from power cables, as shown below.



Whenever possible, use wiring conduit to hold the I/O wiring. Standard wiring conduit should be used, and it should be long enough to completely contain the I/O wiring and keep it separated from other cables.

2-4-2 PC Dimensions

The following diagram shows the dimensions of the C20HB-TS as well as the location of the mounting holes required to installation.



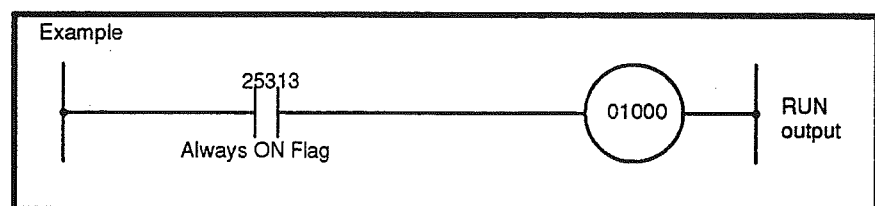
2-5 Wiring

This section provides information about wiring power supplies, grounding, and I/O.

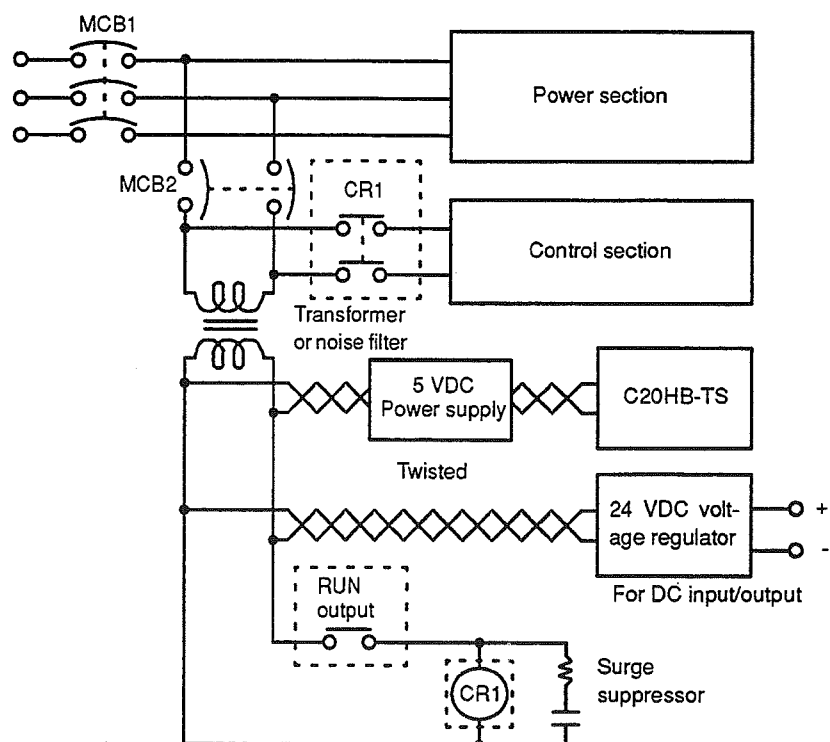
2-5-1 General Wiring Precautions

Emergency Stop Circuit

To prevent a CPU breakdown or malfunction from damaging the entire system, construct an external relay circuit so that SR bit 25313 is always ON when the CPU is operating. If the program is set up as shown in the following diagram, output 01000 will be ON whenever the CPU is in either RUN or MONITOR mode, and it will function as an output to monitor whether the CPU is operating properly.

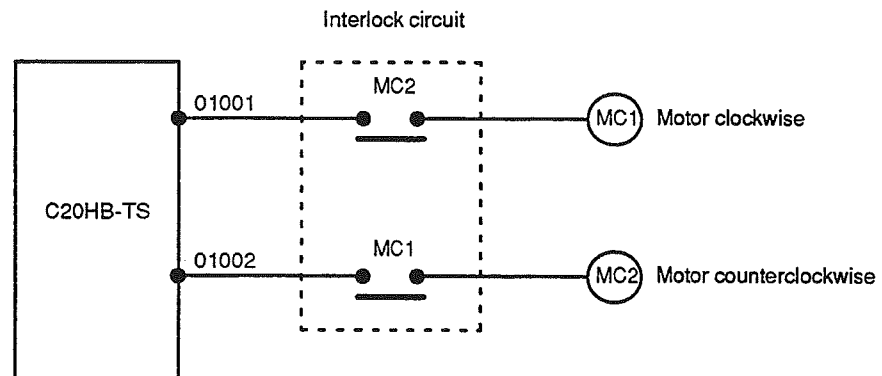


The following diagram shows a wiring example for an emergency stop circuit that turns the power to the PC OFF in the event of an emergency.



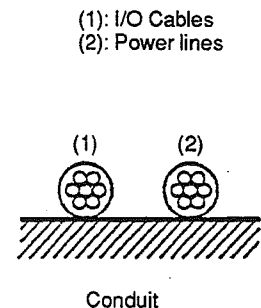
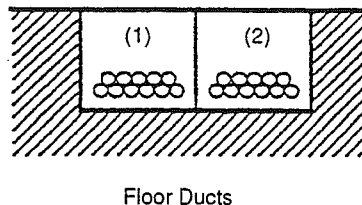
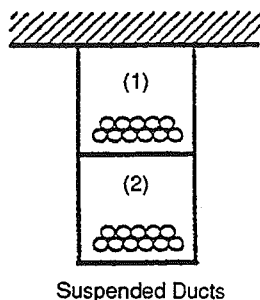
Interlock Circuits

When the PC controls an operation such as the clockwise and counterclockwise operation of a motor, provide an external interlock such as the one shown below to prevent both the forward and reverse outputs from turning ON at the same time. Even if the PC is programmed improperly or malfunctions, the motor is protected.



Electrical Noise

Be sure to ground the C20HB-TS independently to a ground less than 100 Ω , and separate I/O cables from power lines as shown in the following diagrams.

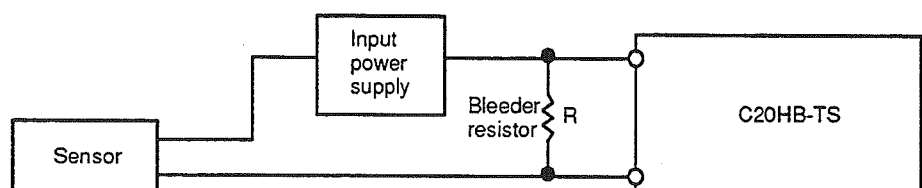


If the I/O cables and power lines must be wired in the same duct, noise-resistance can be improved by using shielded cable. In this case, ground the cables shielding to the FG terminal on the C20HB-TS.

2-5-2 Input Wiring Precautions

Input Leakage Current

When two-wire sensors, such as photoelectric sensors, proximity sensors, or limit switches with LEDs, are connected to the PC as input devices, the input bit may be turned ON erroneously by leakage current. In order to prevent this, connect a bleeder resistor across the input as shown below.



If the leakage current is less than 1.5 mA, there should be no problem. If the leakage current is greater than 1.5 mA, determine the value and rating for the bleeder resistor using the following formulas.

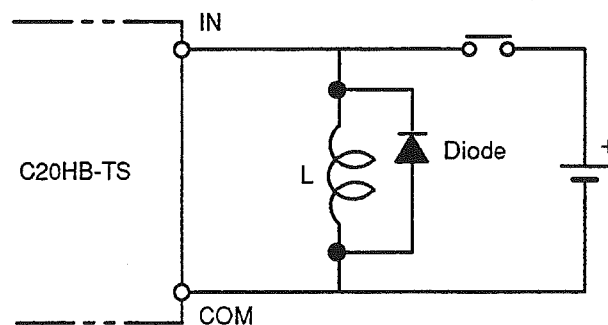
$$R = \frac{17.15}{3.43 \times I - 5} \text{ k}\Omega \text{ max.}$$

$$W = \frac{2.3}{R} \text{ W min.}$$

I = leakage current in mA
R = Bleeder resistor k Ω
W = Bleeder resistor Watts

Inductive Load Surge Suppressors

When an inductive load is connected to an input, it is necessary to connect a surge suppressor or a diode in parallel with the load, as shown below, to absorb the counter-electromotive force produced by the load.



The counter-electromotive force is a minimum of 3 times the load voltage and the average rectifying current is 1 A.

2-5-3 Output Wiring Precautions

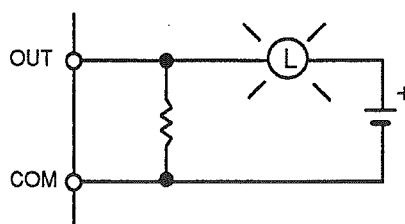
Listed below are a few items to keep in mind when wiring outputs.

- When connecting the output devices, use a fuse to protect the printed circuit board from a power surge.
- Do not put output wires near high-voltage power lines. Doing so may cause a malfunction or may damage output devices and the Units.
- The durability of a relay depends on the amount of current passing through it, the temperature, and the switching capacity. At higher temperatures and higher switching capacities the durability is lowered by as much as 50 per cent.

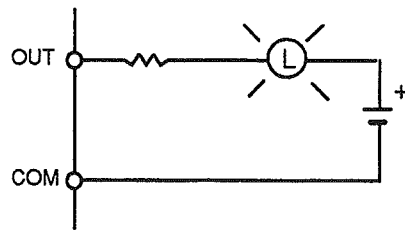
Reducing Surge Current

When connecting a transistor output to a device that allows a high surge current to flow, the current may exceed the rated current, causing damage to the transistor. Use one of the circuits shown below to reduce the inrush current.

This circuit allows a slight current (about 1/3 the rated current) to flow through the load (i.e., the lamp), thus eliminating any initial surge of current.

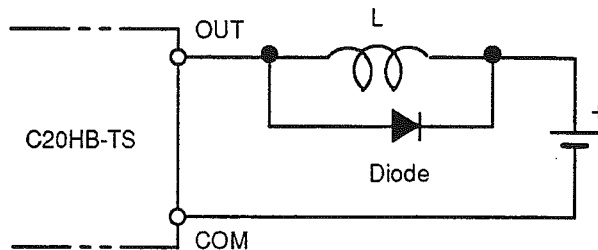


This circuit acts directly on the inrush current to limit it, but also reduces the voltage across the load.



Inductive Load Surge Suppressors

When an inductive load is connected to an output, it is necessary to connect a surge suppressor or a diode in parallel with the load, as shown below, to absorb the counter-electromotive force produced by the load.



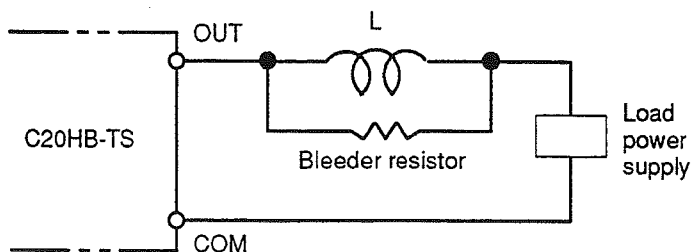
The counter-electromotive force is a minimum of 3 times the load voltage and the average rectifying current is 1 A.

Transistor Output Residual Voltage

When connecting TTL circuits to transistor outputs, it is necessary to connect a pull-up resistor and a CMOS IC between the two because of the transistor's residual voltage.

Output Leakage Current

Output devices can be triggered by the leakage current. If this occurs, connect a bleeder resistor as shown in the following diagram.

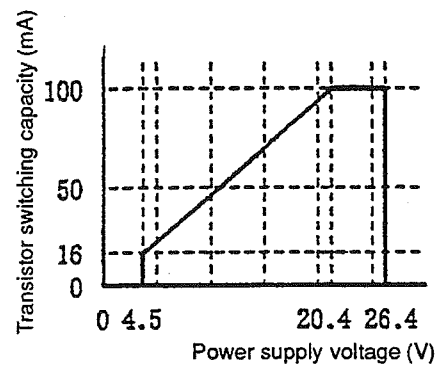


Determine the value for the bleeder resistor using the following formula.

$$R < \frac{V_{ON}}{I}$$

V_{ON} = Load voltage when ON
 I = leakage current (0.1 mA)
 R = Bleeder resistor KW

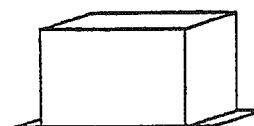
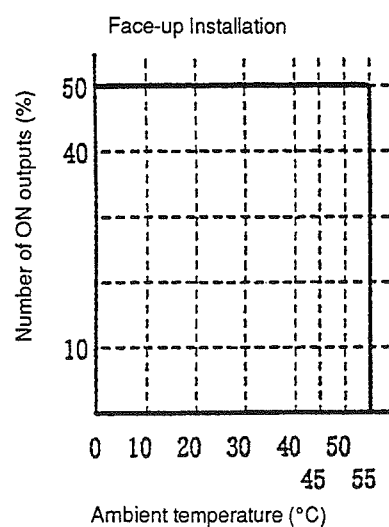
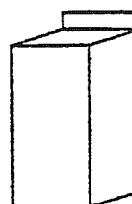
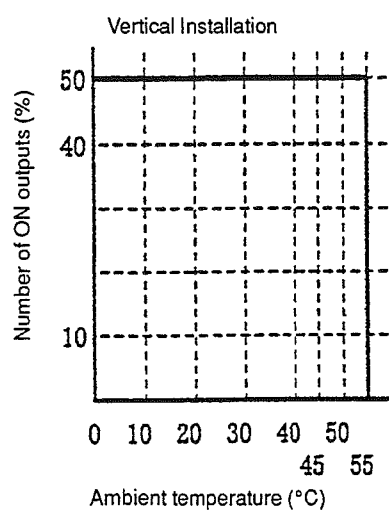
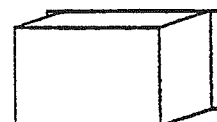
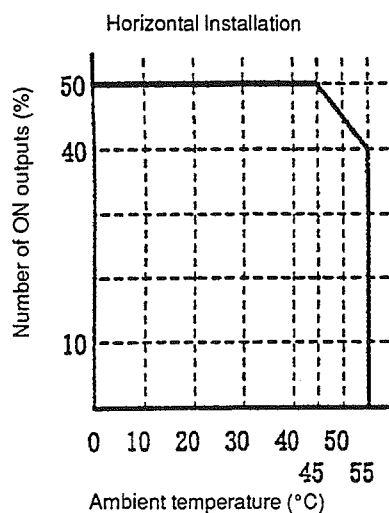
Maximum Switching Capacity As shown in the following chart, the maximum load current for transistor outputs depends on the voltage of the power supply, which can be between 5 and 24 VDC $\pm 10\%$. Use this chart as a guide when selecting the power supply.



⚠ Caution The transistor output will be damaged if the switching current exceeds the maximum value given in the chart.

Maximum Number of ON Outputs

The maximum number of outputs that can be ON at the same time depends on the orientation of the PC and the ambient temperature, as shown in the following charts.



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	30
3-2	IR (Internal Relay) Area	31
3-3	SR (Special Relay) Area	31
3-3-1	Forced Status Hold Bit	32
3-3-2	I/O Status Hold Bit	32
3-3-3	FAL (Failure Alarm) Area	32
3-3-4	Instruction Execution Error Flag, ER	33
3-4	AR (Auxiliary Relay) Area	33
3-4-1	System Parameter Flags	35
3-4-2	System Command Bits	35
3-5	DM (Data Memory) Area	37
3-5-1	Indirect Addressing	37
3-5-2	Parameter and Parameter Backup Areas	38
3-5-3	User Program Header Area	41
3-6	HR (Holding Relay) Area	42
3-7	TC (Timer/Counter) Area	42
3-8	LR (Link Relay) Area	42
3-9	TR (Temporary Relay) Area	43

3-1 Introduction

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

The AR, DM, HR, and TC areas are backed up by a capacitor that maintains data for 5 minutes (at 25°C) after a power interruption. Data will not be reliable after power interruptions longer than 5 minutes.

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. (Capacitor backup.)
Data Memory	DM	Read/write: DM 0000 to DM 0999 Read only: DM 1000 to DM 1999	Used for internal data storage and manipulation. (Capacitor backup. See note.)
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. (Capacitor backup.)
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 are 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. Timers are reset when PC operation is started, but counter PVs are maintained through momentary power interruptions. (Capacitor backup.)
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.

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.

Data Area Acronyms

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.

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

3-2 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. The following table shows the usage of IR area in the C20HB-TS.

Word(s)	Bits	Function
000 to 002	00000 to 00215	24 VDC Inputs (three 16-point commons)
003 to 009	00300 to 00915	Available for use as work bits in the program.
010 to 012	01000 to 01107	PNP Transistor Outputs (three 8-point commons)
	01108 to 01215	NPN Transistor Outputs (six 4-point commons)
013 to 019	01300 to 01915	Available for use as work bits in the program.
020	02000 to 02015	Relay Outputs (two 8-point commons)
021	02100 to 02103	Relay Outputs (one 4-point common)
	02104 to 02115	Relay Outputs (twelve independent commons)
022 to 246	02200 to 24615	Available for use as work bits in the program.

3-3 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. Bits in the SR area which are listed as "not used" cannot be used as work bits in the program.

Word(s)	Bit(s)	Function
247 to 251	00 to 15	Not used.
252	00 to 07	Not used.
	08	RS-485 Communications Error Flag
	09	RS-485 Interface Restart Bit: Turn this bit OFF → ON → OFF to restart the interface.
	10	Not used.
	11	Forced Status Hold Bit (See 3-3-1 <i>Forced Status Hold Bit</i> for details.)
	12	I/O Status Hold Bit (See 3-3-2 <i>I/O Status Hold Bit</i> for details.)
	13 to 15	Not used.

Word(s)	Bit(s)	Function
253	00 to 07	FAL number output area (See 3-3-3 FAL (Failure Alarm) Area for details.)
	08	Not used.
	09	Cycle Time Error Flag: This flag is turned ON if the cycle time exceeds 100 ms.
	10 to 12	Not used.
	13	Always ON Flag
	14	Always OFF Flag
	15	First Cycle Flag: This flag is turned ON when PC operation begins and then OFF after one cycle of the program. It is useful in initializing counter values and other operations.
254	00	1-minute clock pulse bit (30 s ON, 30 s OFF)
	01	0.02-second clock pulse bit (10 ms ON, 10 ms OFF)
	02 to 06	Not used.
	07	Step Flag: This flag is turned ON for one cycle when step execution is started with STEP(08).
	08 to 15	Not used.
255	00	0.1-second clock pulse bit (50 ms ON, 50 ms OFF)
	01	0.2-second clock pulse bit (100 ms ON, 100 ms OFF)
	02	1.0-second clock pulse bit (0.5 s ON, 0.5 s OFF)
	03	Instruction Execution Error (ER) Flag (See 3-3-4 Instruction Execution Error Flag, ER for details.)
	04	Carry (CY) Flag: This flag 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.
	05	Greater Than (GR) Flag: This flag is turned ON when the result of a comparison shows the first of two operands to be greater than the second.
	06	Equals (EQ) Flag: This flag 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.
	07	Less Than (LE) Flag: This flag is turned ON when the result of a comparison shows the first of two operands to be less than the second.

3-3-1 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.

SR 25211 is not effective when switching to RUN mode.

3-3-2 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.

3-3-3 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. This area can be reset by executing the FAL instruction with an operand of 00.

3-3-4 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 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. An overview of the AR area is provided in the following table. Bits in the AR area which are listed as "not used" cannot be used as work bits in the program.

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

Word(s)	Bit(s)	Function
00 to 03	00 to 15	Not used.
04	00 to 07	RS-485 Communications Error Code: When an error has occurred in RS-485 communications, the RS-485 Communications Error Flag (SR25208) is turned ON, and a code that indicates the type of error is output to AR 0400 to AR 0407. These codes are as follows: 01: Parity error 02: Framing error 03: Overrun error 04: FCS error These bits are refreshed every cycle.
	08 to 15	Not used.
05	00 to 07	RS-485 Reception Counter: When a transmission is received on the RS-485 interface, the number of characters is counted in hexadecimal and then output to AR 0500 to AR 0507 to assist the user in debugging RS-485 interface communications. This counter is used for all RS-485 interface operating modes, as well as for error characters. The counter can be reset by turning SR 25209 OFF → ON → OFF. These bits are refreshed every cycle.
	08 to 15	RS-485 Transmission Counter (refreshed each cycle): This counter operates the same as the RS-485 Reception Counter, except that it counts the number of characters transmitted from the PC through the RS-485 interface. These bits are refreshed every cycle.
06	00 to 15	Not used.
07	00 to 12	Not used.
	13	Error History Overwrite Bit: This bit controls 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 after the history area is cleared. This bit is refreshed every cycle.
	14	Error History Reset Bit: Turn this bit OFF → ON → OFF to reset the Error Record Pointer (DM 0969) and thus restart recording error records at the beginning of the history area. This bit is refreshed every cycle.
	15	Error History Enable Bit: Turn this bit ON to enable error history storage and OFF to disable error history storage. This bit is refreshed every cycle.
08 to 11	00 to 15	Not used.

Word(s)	Bit(s)	Function
12	00 to 15	System Parameter Warning Flags (See 3-4-1 System Parameter Flags.)
13	00 to 13	
	14	System Parameter Backup Flag (See 3-4-1 System Parameter Flags.)
	15	System Parameter/Backup Area Checksum Flag (See 3-4-1 System Parameter Flags.)
14	00 to 06	System Command Response Code (See 3-4-2 System Command Bits.)
	07	System Command Completion Flag (See 3-4-2 System Command Bits.)
	08 to 14	System Command Command Code (See 3-4-2 System Command Bits.)
	15	System Command Execution Bit (See 3-4-2 System Command Bits.)
15	00 to 07	PC Operating Mode: These bits contain a hexadecimal code that indicates the operating mode of the PC. If the Preserve Mode code (01) is set in bits 08 through 15 of DM 0900 (backup: DM 1900), the PC will be started in the mode indicated by the code in AR 1500 to AR 1507. The possible codes are: 00: PROGRAM mode 01: MONITOR mode 02: RUN mode These bits are refreshed every cycle.
	08 to 15	Not used.
16 to 22	00 to 15	Not used.
23	00 to 15	Power-off Counter: AR 23 indicates in 4-digit BCD the number of times that the PC power has been turned off. This data has a capacitor backup, so it will be unreliable after power interruptions longer than 5 minutes. This counter can be reset as necessary using the Hex/BCD Change operation from the Programming Console or other Peripheral Device. The Power-off Counter is refreshed every time the power is turned on.
24	00 to 04	Not used.
	05	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.
	06 to 14	Not used.
	15	Programming Console or 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. It is refreshed every cycle.
25	00 to 15	FALS-generating Address: AR 25 contains (in 4-digit BCD) the address generating a user-programmed FALS code (from FAL(06) or FALS(07)) or a system FALS code 9F (cycle time error). AR 25 is refreshed each cycle while an FALS code is being generated.
26	00 to 15	Maximum Cycle Time Area: AR 26 contains the maximum cycle time that has occurred since program execution began. This time is recorded in tenths of a millisecond in 4-digit BCD. It is refreshed every cycle.
27	00 to 15	Current Cycle Time Area: AR 27 contains the present cycle time in tenths of a millisecond in 4-digit BCD. It is refreshed every cycle.

3-4-1 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 0915	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 3-4-2 *System Command Bits* for details on system command bits and to 3-5 *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.

3-4-2 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 write-protected (i.e., the write enable switch is ON).
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.
3	Backup disable	The data contained in the Parameter Backup Area is disabled and AR 1314 (System Parameter Backup Flag) is turned OFF.
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-485 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 is executed by manipulating the content of AR 14 with a Programming Device. The following procedure demonstrates how to execute the system command from a 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.

3-5 DM (Data Memory) Area

Although composed of 16-bit words like any other data area, the DM area is accessible by word only. Individual bits within a DM word cannot be designated, so DM words 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 is divided into various parts as described in the following table.

Addresses	Usage	Memory type
DM 0000 to DM 0899	General User Area	RAM (Read/write)
DM 0900 to DM 0929	Parameter Area (PC operating mode)	
DM 0930 to DM 0968	Not used.	
DM 0969 to DM 0999	Error History Area (up to 10 entries)	
DM 1000 to DM 1899	General User Area	EEPROM (Read-only)
DM 1900 to DM 1929	Parameter Area backup	
DM 1930 to DM 1989	Not used.	
DM 1990 to DM 1999	User Program Header	

The DM area retains status during power interruptions.

General User Areas

The General User Areas can be used for general data storage and manipulation in the program.

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.

System DM

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. These DM words are used to control various aspects of PC operation and should never be used for any purposes other than their designed purposes.

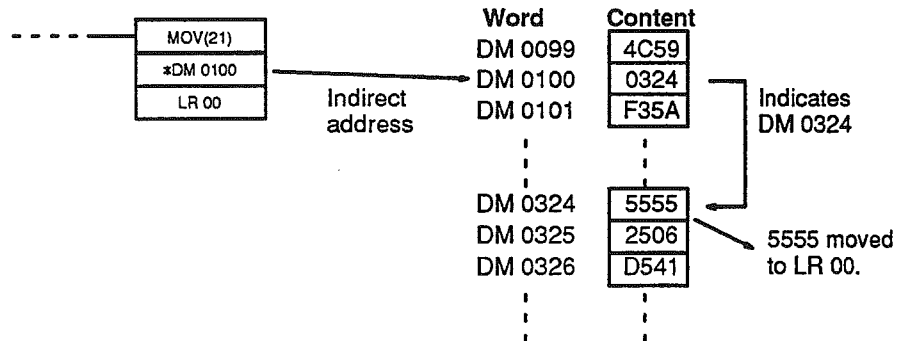
Refer to 3-5-2 *Parameter and Parameter Backup Areas* for details on the Parameter Area and Parameter Backup Area.

Refer to 3-5-3 *User Program Header* for details on the User Program Header.

3-5-1 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.



3-5-2 Parameter and Parameter Backup Areas

The Parameter Area (DM 0900 to DM 0929) and the Parameter Backup Area (DM 1900 to DM 1929) contain parameters that control various aspects of PC operation, enabling greater system flexibility. These parameters are written to the Parameter Area with a Programming Device such as a Programming Console, and then copied to the Parameter Backup Area.

In the event of a power interruption, the contents of the Parameter Area will be maintained for only 5 minutes by a capacitor backup. Always backup the parameters in the Parameter Backup Area, which will retain its data indefinitely because it is in EEPROM.

System Command

The system command executes the 7 basic operations that control the Parameter Area. The system command is executed by manipulating the content of AR 14 with a Programming Device. Refer to 3-4-2 *System Command Bits* 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 its 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.

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)	Host Link Service Time The percent of the cycle time allocated to Host Link servicing 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 minimum service time is 1 ms regardless of the cycle time and this setting.		5%
	00 to 07	Percent of cycle time allocated to Host Link 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	

Address	Name/Description		Default
	Bit	Content/Meaning	
DM 0905 to DM 0919 (DM 1905 to DM 1919)	Not used.		---
DM 0920 (DM 1920)	00 to 07	Host Link Communications Format Selection 00: Standard (1 start bit, 7-bit data length, even parity, 2 stop bits, 19,200 baud) 01: Custom settings (i.e., according to contents of DM 0921)	Standard
	08 to 15	Not used.	---
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 06: 19,200 bps	19200 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	Host Link Transmission Delay In tenths of milliseconds between 00 and 99 (BCD, correspond to 000 and 990 ms delays, respectively)	0 ms
	08 to 15	Not used.	---
DM 0923 to DM 0929 (DM 1923 to DM 1929)	Not used.		---

3-5-3 User Program Header Area

DM 1990 to DM 1999 can be used to record the program's title, version and the date that the program was created, as shown below.

Address*	Bits	Contents
DM 1990	00 to 07	Program version number in BCD without the decimal (For example, a value of 12 indicates version 1.2.)
	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.	

This information (except for the seconds) can also be displayed on the Programming Console by pressing CLR, SFT, and MONTR (see 7-2-6 *Program Header Display* for details).

3-6 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 or PC operation is stopped, but the C20HB-TS has a capacitor backup, not a battery backup, so the HR area retains status for only 5 minutes when power is interrupted. Since the HR area status isn't reliable when the PC has been off for 5 minutes or more, the HR area must be initialized when the PC is turned on if HR area data is required in the program.

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*.

3-7 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, or CNTR(12). 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.

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. The PVs of counters are not reset at these times.

Note Use the first 16 TC numbers (TC 000 through TC 015) for TIMH(15) when the cycle time is longer than 10 ms. The high-speed timer might not count accurately if higher TC numbers are used, because the higher TC numbers do not generate interrupts.

3-8 LR (Link Relay) Area

The LR area is used as a common data area by PCs that support a PC Link System, but the C20HB-TS does not support the PC Link System, so the LR area is available for use in the program as work words and bits.

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.

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.

3-9 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 through 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. The entire set of instructions used in programming is described in *Section 5 Instruction Set*.

4-1	Basic Procedure	46
4-2	Instruction Terminology	46
4-3	Ladder Diagrams	47
4-3-1	Basic Terms	48
4-3-2	Mnemonic Code	48
4-4	The Programming Console	50
4-4-1	The Keyboard	51
4-4-2	PC Modes	53
4-5	Preparation for Operation	54
4-5-1	Entering the Password	54
4-5-2	Clearing Memory	55
4-5-3	Clearing Error Messages	57
4-6	Inputting, Modifying, and Checking the Program	58
4-6-1	Setting and Reading from Program Memory Address	58
4-6-2	Entering or Editing Programs	59
4-6-3	Checking the Program	62
4-6-4	Displaying the Cycle Time	64
4-6-5	Program Searches	65
4-6-6	Inserting and Deleting Instructions	67
4-6-7	Branching Instruction Lines	69
4-6-8	Jumps	74
4-7	Controlling Bit Status	76
4-7-1	DIFFERENTIATE UP and DIFFERENTIATE DOWN	76
4-7-2	KEEP	76
4-7-3	Self-maintaining Bits (Seal)	77
4-8	Work Bits (Internal Relays)	77
4-9	Programming Precautions	79
4-10	Program Execution	81

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 B Word Assignment Recording Sheets* and *Appendix C 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 numbers are described in *5-8 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:

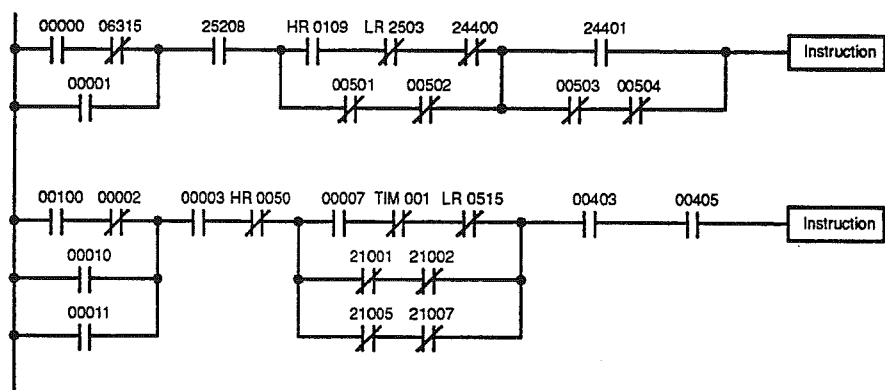
- 1, 2, 3...**
1. Instructions that correspond to the conditions on the ladder diagram and are used in instruction form only when converting a program to mnemonic code.
 2. Instructions that are used on the right side of the ladder diagram and are executed according to the conditions on the instruction 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 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.



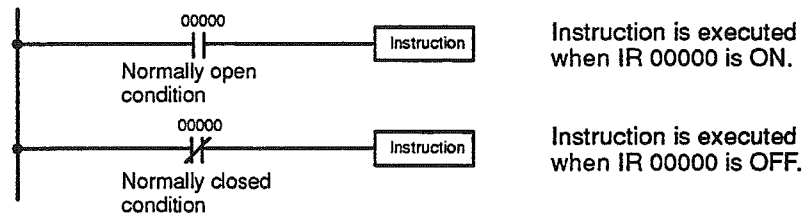
As shown in the diagram above, instruction lines can branch apart and they can join back together. The vertical pairs of lines are called conditions. Conditions without diagonal lines through them are called normally open conditions and correspond to a LOAD, AND, or OR instruction. The conditions with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



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 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 is 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 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.

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.

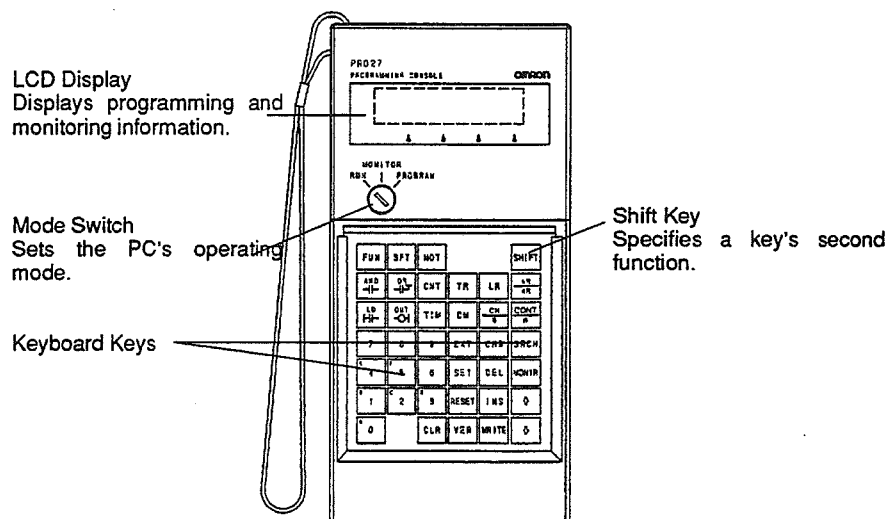
4-4 The Programming Console

Once a program has been written, it must be input into the PC. This can be done in graphic (ladder diagram) form using a FIT or LSS. The most common way of inputting a program, however, is through a Programming Console using mnemonic code. This section describes 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.

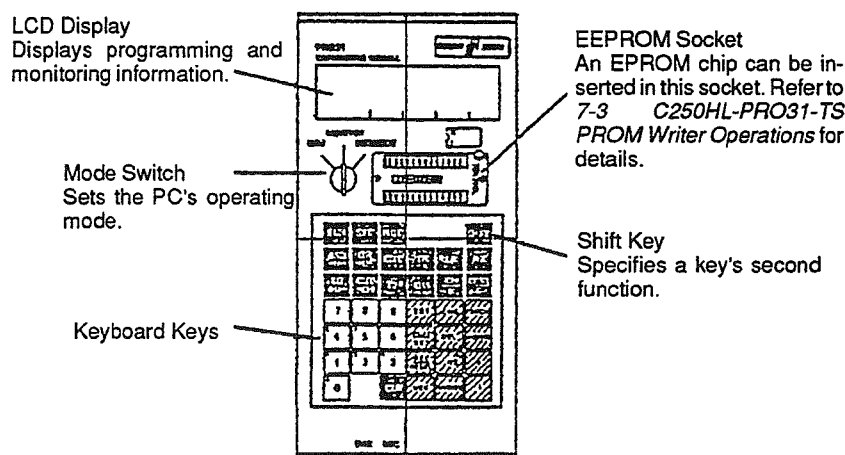
C200H-PRO27-E Programming Console

The following diagram shows the main components of the C200H-PRO27-E Programming Console.



C250HL-PRO31-TS Programming Console

The following diagram shows the main components of the C250HL-PRO31-TS Programming Console.



4-4-1 The Keyboard

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

White: Numeric Keys

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

Red: CLR Key

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






















Yellow: Operation Keys

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

Gray: Instruction and Data Area Keys

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

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

	Pressed before the function code when inputting an instruction via its function code.
	Pressed to enter SFT (the Shift Register instruction).
	Input either after a function code to designate the differentiated form of an instruction or after a ladder instruction to designate an inverse condition.
	Pressed to enter AND (the AND instruction) or used with NOT to enter AND NOT.
	Pressed to enter OR (the OR instruction) or used with NOT to enter OR NOT.
	Pressed to enter CNT (the Counter instruction) or to designate a TC number that has already been defined as a counter.
	Pressed to enter LD (the Load instruction) or used with NOT to enter LD NOT. Also pressed to indicate an input bit.
	Pressed to enter OUT (the Output instruction) or used with NOT to enter OUT NOT. Also pressed to indicate an output bit.
	Pressed to enter TIM (the Timer instruction) or to designate a TC number that has already been defined as a timer.
	Pressed before designating an address in the TR area.
	Pressed before designating an address in the LR area.
	Pressed before designating an address in the HR area.
 	Pressed before designating an address in the AR area.
	Pressed before designating an address in the DM area.
	Pressed before designating an indirect DM address.
 	Pressed before designating a word address.
	Pressed before designating an operand as a constant.
 	Pressed before designating a bit address.

4-4-2 PC Modes

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

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

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

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



DANGER!

Do not leave the Programming Console connected to the PC by an extension cable when in RUN mode. Noise 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 CPU's write-enable switch is ON.
 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.

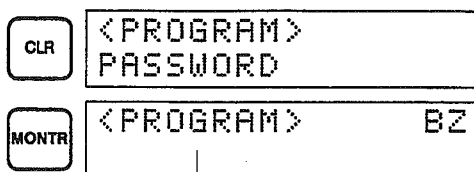
Each of these operations from entering the password on is described in detail in the following subsections. All operations should be done in PROGRAM mode unless otherwise noted.

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.



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.

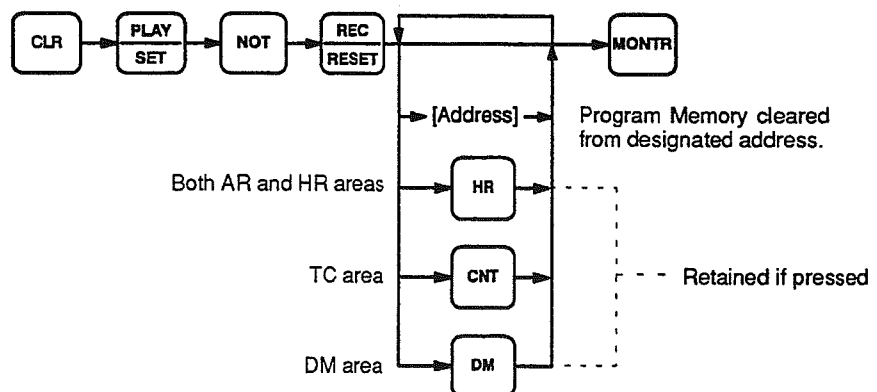
This beeper also will also sound whenever an error occurs during PC operation. Beeper operation for errors is not affected by the above setting.

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 CPU's write-enable switch is ON. If the write-enable switch is OFF, Program Memory and DM 1000 through DM 1999 will not be cleared. (The system DM areas, DM 0900 to DM 0999 and DM 1900 to DM 1999, will not be cleared in any circumstances.)

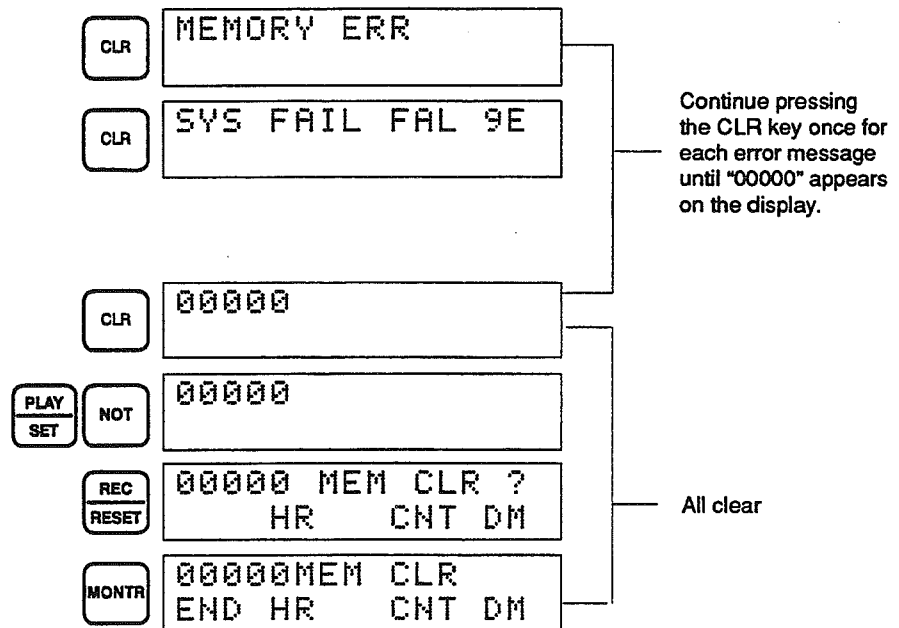
Before beginning to programming for the first time or when installing a new program, all areas should normally be cleared. Before clearing memory, check to see whether there is already a program loaded. If there is one, determine whether it is one that you need. If you do need the program, clear only 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.

**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:

CLR	00000
PLAY SET	00000
NOT	00000
REC RESET	00000MEM CLR ? HR CNT DM
CNT	00000MEM CLR ? HR DM
^B 1 ^C 2 ^D 3	00123MEM CLR ? HR DM
MONTR	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

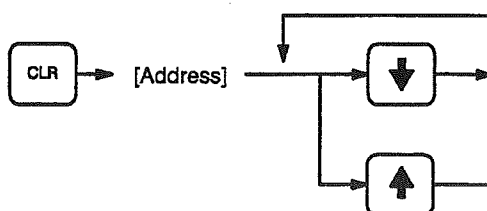
When inputting a program for the first time, it is generally written to Program Memory starting from address 00000. Because this address appears when the display is cleared, it is not necessary to specify it.

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.

CLR	00000
C2 A0 A0	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.

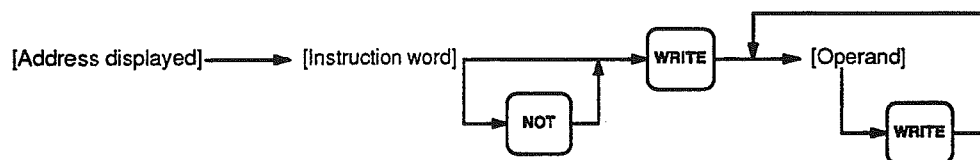
Designating Instructions

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 HL	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 write-protected EEPROM. Be sure the write-enable switch (pin 5 of the DIP switch) is set to ON.
****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.

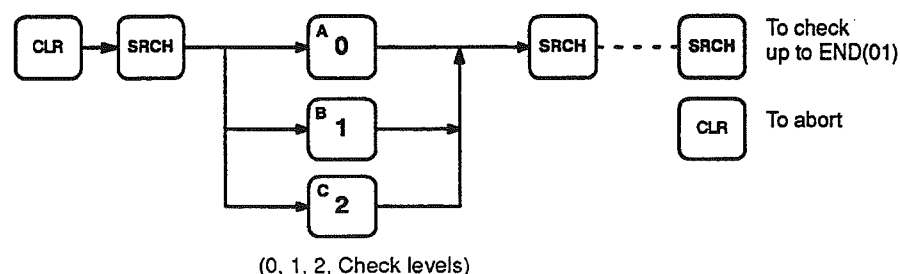
4-6-3 Checking the Program

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**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.

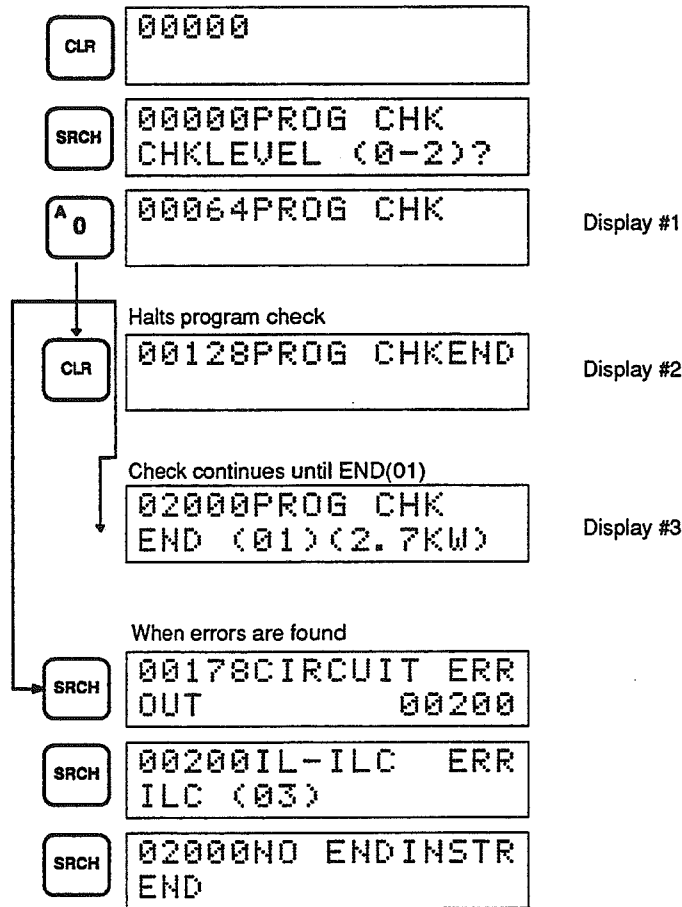
Many of the following errors are for instructions that have not yet been described. Refer to 4-7 *Controlling Bit Status* or *Section 5 Instruction Set* for details on these.

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 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.)
	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 be 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.
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.

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

Example

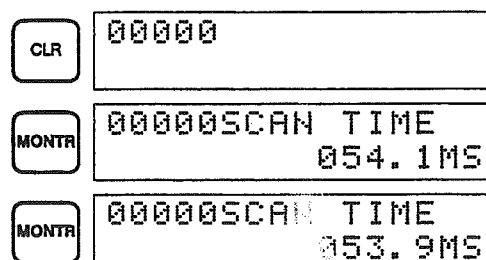
The following example shows some of the displays that can appear as a result of a program check.

**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

4-6-5 Program Searches

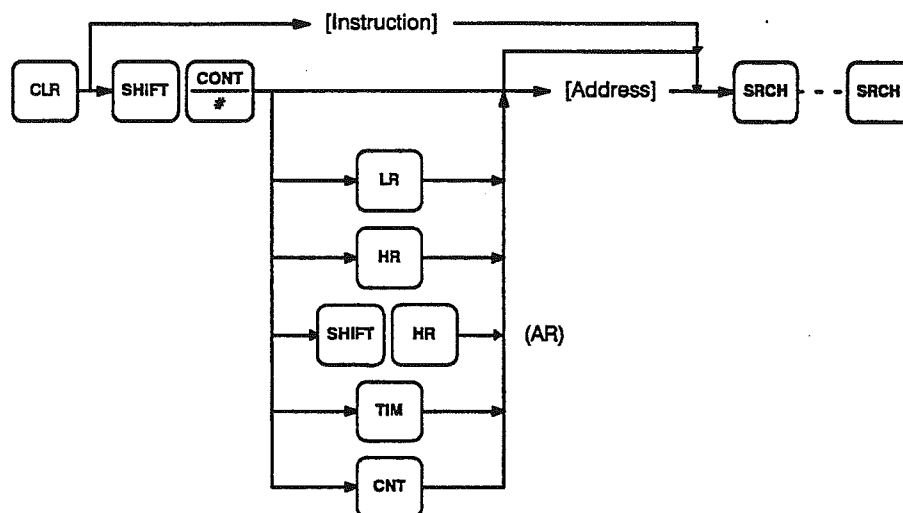
The program can be searched for occurrences of any designated instruction or data area address used in an instruction. Searches can be performed from any currently displayed address or from a cleared display.

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 HI	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

Locate position
in program
then enter:



Instruction
currently
displayed



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.

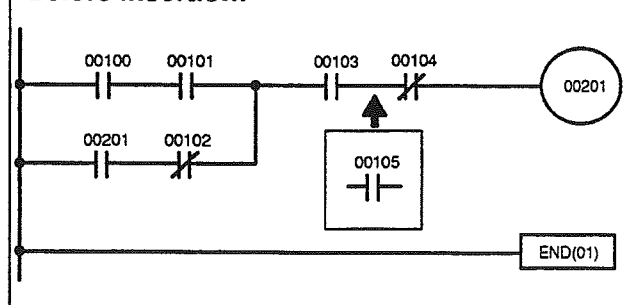
Example

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

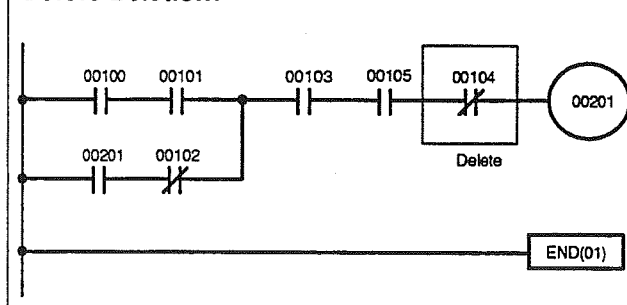
Original Program

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

CLR	00000	
OUT -O-	00000	00000
C 2 A 0 B 1	00000	00201
SRCH	00207SRCH	00201
↑	00206READ	AND NOT 00104
AND -H-	00206	AND 00000
B 1 A 0 F 5	00206	AND 00105
INS	00206INSERT?	AND 00105
↓	00207INSERT END	AND NOT 00104
↑	00206READ	AND 00105

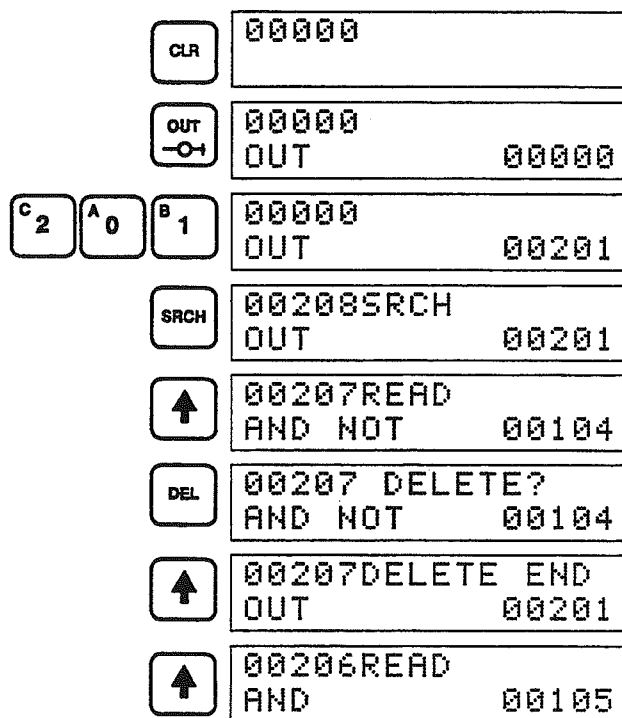
Find the address
prior to the inser-
tion point

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)	-

Insert the
instruction

Deleting an Instruction



Find the instruction that requires deletion.

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)	-

Confirm that this is the instruction to be deleted.

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

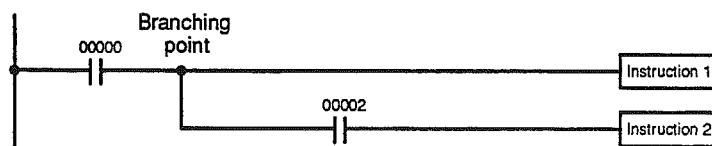


Diagram A: Correct Operation

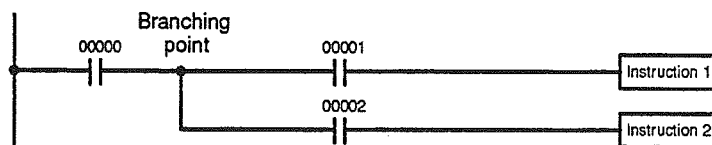


Diagram B: Incorrect Operation

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

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

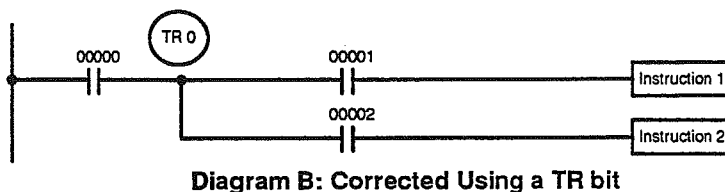
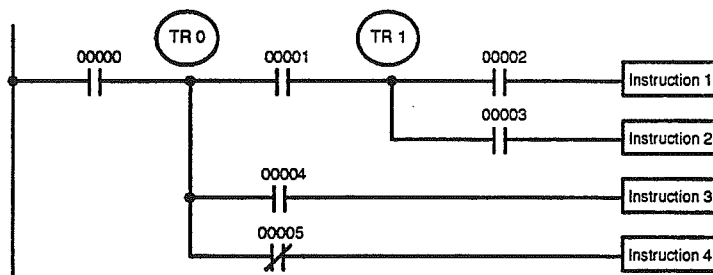


Diagram B: Corrected Using a TR bit

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.



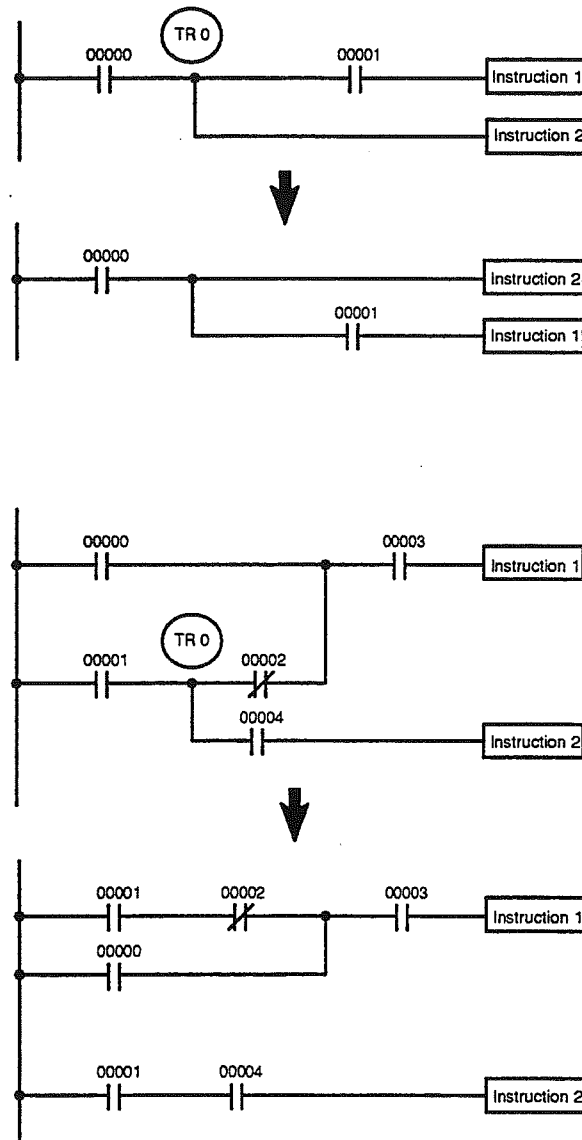
Address	Instruction	Operands
00000	LD	00000
00001	OUT	TR 0
00002	AND	00001
00003	OUT	TR 1
00004	AND	00002
00005	OUT	00500
00006	LD	TR 1
00007	AND	00003
00008	OUT	00501
00009	LD	TR 0
00010	AND	00004
00011	OUT	00502
00012	LD	TR 0
00013	AND NOT	00005
00014	OUT	00503

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.



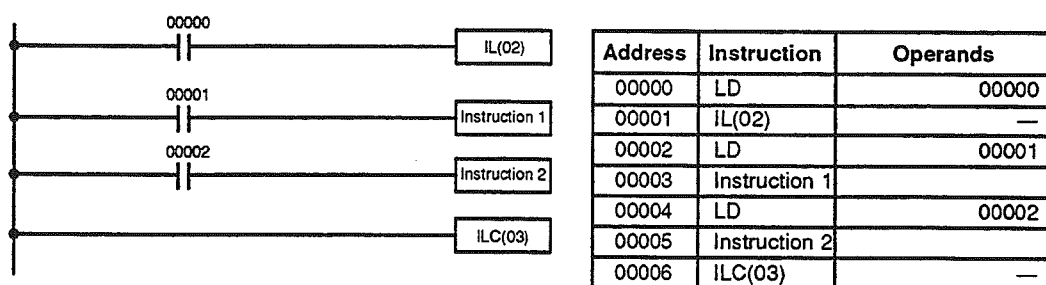
Note TR bits are only used when programming using mnemonic code. They are not necessary when inputting ladder diagrams directly, as is possible from a host computer running LSS. The above limitations on the number of branching points requiring TR bits, and considerations on methods to reduce the number 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.

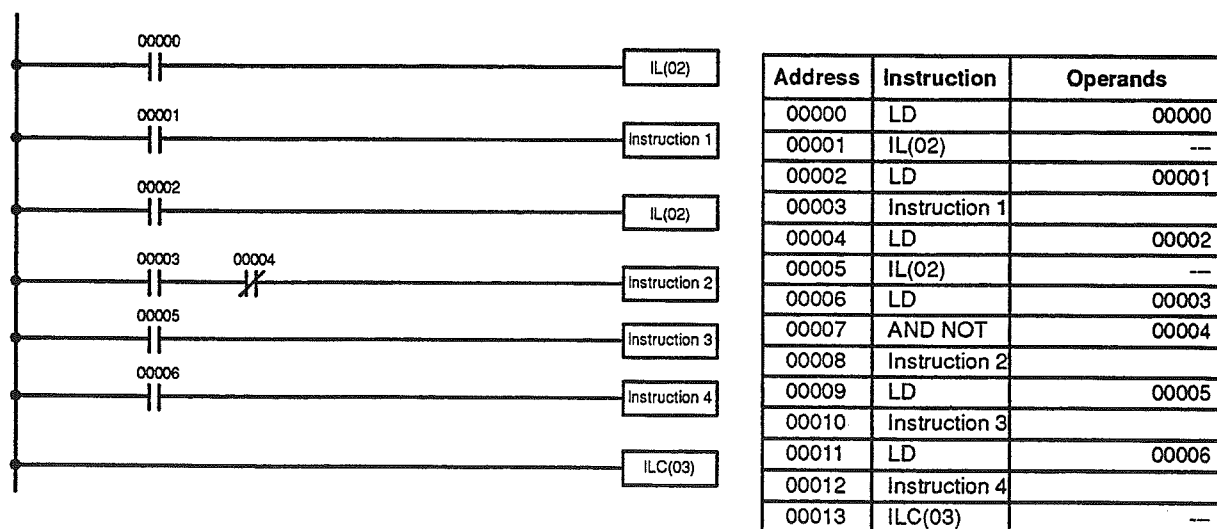
When an INTERLOCK instruction is placed before a section of a ladder program, the execution condition for the INTERLOCK instruction will control the execution of all instruction up to the next INTERLOCK CLEAR instruction. If the execution condition for the INTERLOCK instruction is OFF, all right-hand instructions through the next INTERLOCK CLEAR instruction will be executed with OFF execution conditions to reset the entire section of the ladder diagram.

Diagram B on page 69 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.



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.



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 require 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.

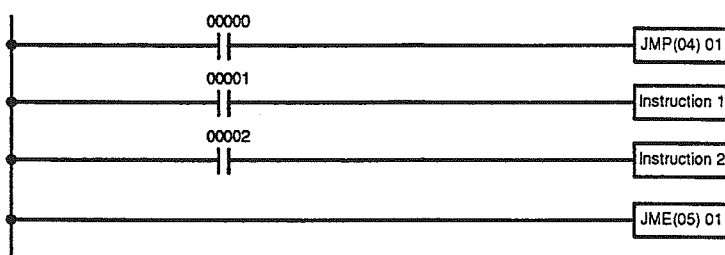


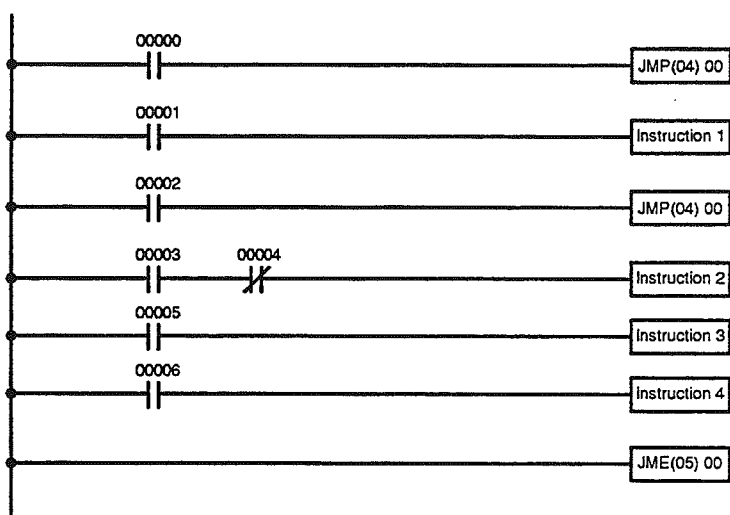
Diagram B: Corrected with a Jump

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).



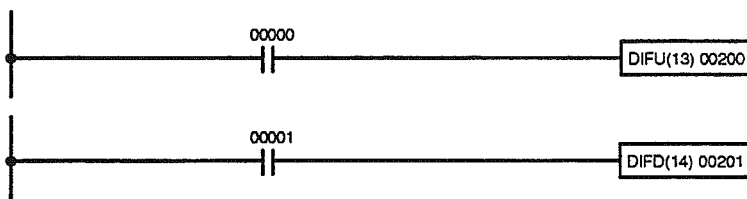
Address	Instruction	Operands
00000	LD	00000
00001	JMP(04)	00
00002	LD	00001
00003	Instruction 1	
00004	LD	00002
00005	JMP(04)	00
00006	LD	00003
00007	AND NOT	00004
00008	Instruction 2	
00009	LD	00005
00010	Instruction 3	
00011	LD	00006
00012	Instruction 4	
00013	JME(05)	00

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 *5-7 Bit Control Instructions*, these instructions (except for OUTPUT and OUTPUT NOT, which have already been introduced) are described here because of their importance in most programs. Although these instructions are used to turn ON and OFF output bits in the 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 cycle at a time. The DIFFERENTIATE UP instruction turns ON the operand bit for one cycle after the execution condition for it goes from OFF to ON; the DIFFERENTIATE DOWN instruction turns ON the operand bit for one cycle after the execution condition for it goes from ON to OFF. Both of these instructions require only one line of mnemonic code.



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

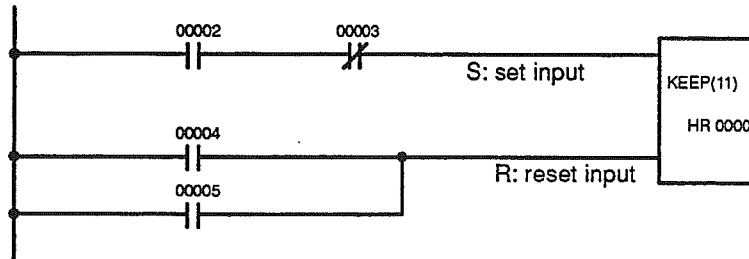
Address	Instruction	Operands
00000	LD	00001
00001	DIFD(14)	00201

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

4-7-2 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.



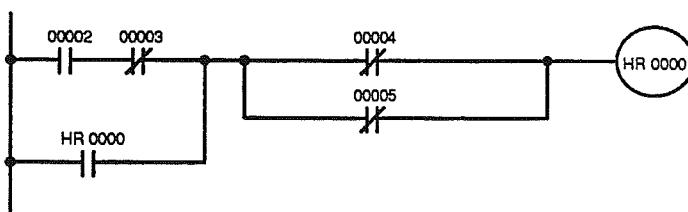
Address	Instruction	Operands
00000	LD	00002
00001	AND NOT	00003
00002	LD	00004
00003	OR	00005
00004	KEEP(11)	HR 0000

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 work 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

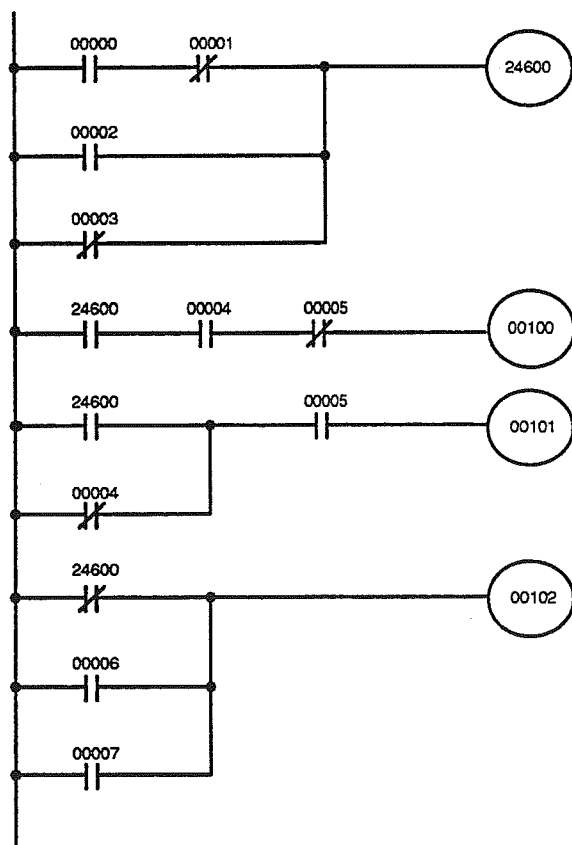
Examples given later in this subsection show two of the most common ways to employ work bits. These should act as a guide to the almost limitless number of ways in which the work bits can be used. Whenever difficulties arise in programming a control action, consideration should be given to work bits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

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

Reducing Complex Conditions

Work bits can be used to simplify programming when a certain combination of conditions is repeatedly used in combination with other conditions. In the following example, IR 00000, IR 00001, IR 00002, and IR 00003 are combined in a logic block that stores the resulting execution condition as the status of IR 24600. IR 24600 is then combined with various other conditions to determine output conditions for IR 00100, IR 00101, and IR 00102, i.e., to turn the outputs allocated to these bits ON or OFF.

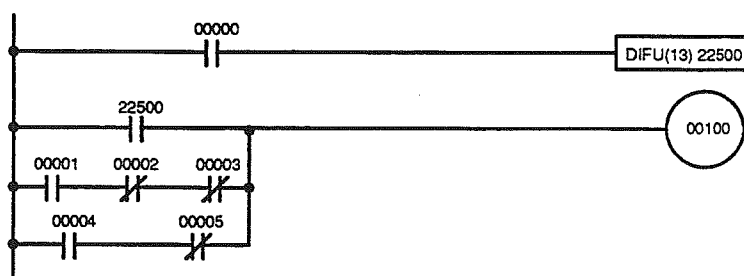


Address	Instruction	Operands
00000	LD	00000
00001	AND NOT	00001
00002	OR	00002
00003	OR NOT	00003
00004	OUT	24600
00005	LD	24600
00006	AND	00004
00007	AND NOT	00005
00008	OUT	00100
00009	LD	24600
00010	OR NOT	00004
00011	AND	00005
00012	OUT	00101
00013	LD NOT	24600
00014	OR	00006
00015	OR	00007
00016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

Differentiated Conditions

Work bits can also be used if differential treatment is necessary for some, but not all, of the conditions required for execution of an instruction. In this example, IR 00100 must be left ON continuously as long as IR 00001 is ON and both IR 00002 and IR 00003 are OFF, or as long as IR 00004 is ON and IR 00005 is OFF. It must be turned ON for only one cycle each time IR 00000 turns ON (unless one of the preceding conditions is keeping it ON continuously).

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



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

4-9 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.

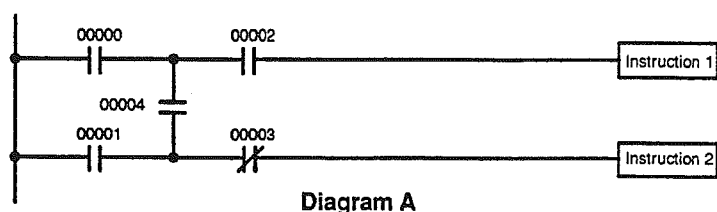


Diagram A

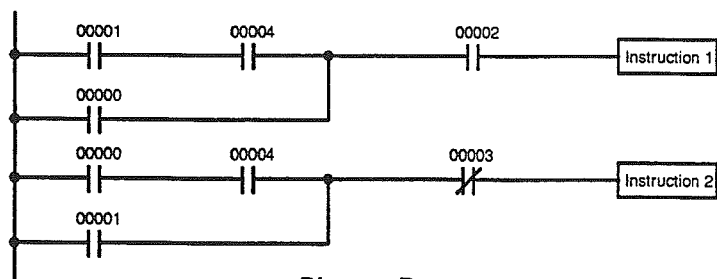
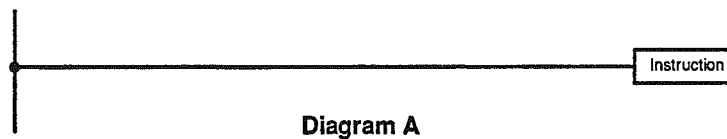


Diagram B

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.

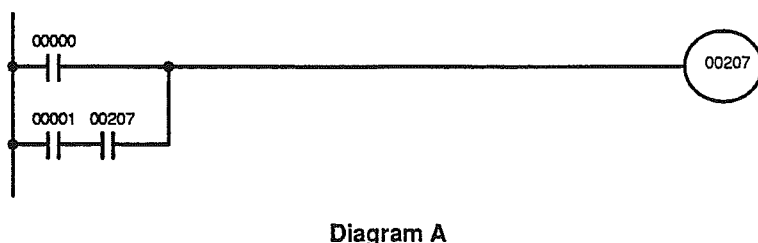
Except for instructions for which conditions are not allowed (e.g., INTERLOCK CLEAR and JUMP END, see below), every instruction line must also have at least one condition on it to determine the execution condition for the instruction at the right. Again, diagram A, below, must be drawn as diagram B. 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.



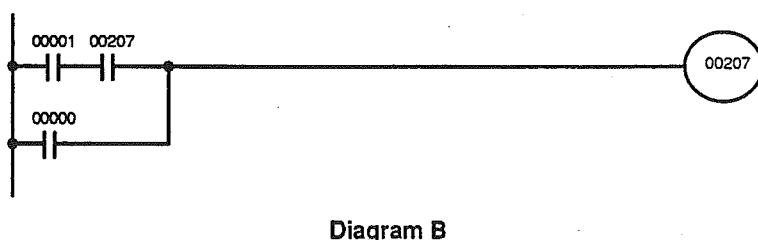
Address	Instruction	Operands
00000	LD	25313
00001	Instruction	

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 5-6-2 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



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

4-10 Program Execution

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

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

SECTION 5

Instruction Set

This section provides a brief summary of each instruction in the C20HB-TS instruction set and provides a more detailed description of the ladder diagram instructions, bit control instructions, and timer and counter instructions.

5-1	Notation	84
5-2	Instruction Format	84
5-3	Data Areas, Definer Values, and Flags	84
5-4	Differentiated Instructions	86
5-5	C20HB-TS Instruction Set	87
5-6	Ladder Diagram Instructions	94
5-6-1	LOAD, LOAD NOT, AND, AND NOT, OR, and OR NOT	94
5-6-2	AND LOAD and OR LOAD	95
5-6-3	Coding Conditions and Other Instructions	95
5-7	Bit Control Instructions	97
5-7-1	OUTPUT and OUTPUT NOT – OUT and OUT NOT	97
5-7-2	DIFFERENTIATE UP and DOWN – DIFU(13) and DIFD(14)	98
5-7-3	KEEP – KEEP(11)	99
5-8	Timer and Counter Instructions	101
5-8-1	TIMER – TIM	102
5-8-2	HIGH-SPEED TIMER – TIMH(15)	106
5-8-3	COUNTER – CNT	107
5-8-4	REVERSIBLE COUNTER – CNTR(12)	110

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 *5-5 C20HS-TS Instruction Set*.

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 provided in *5-5 C20HS-TS Instruction Set*.

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 use. Examples of definers are TC numbers, which are used in timer and counter instructions to create timers and counters, as well as jump numbers (which define which Jump instruction is paired with which Jump End instruction). Bit operands are also contained in the same word as the instruction itself, although these are not considered definers.

5-3 Data Areas, Definer Values, and Flags

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

Not all addresses in the specified data areas are necessarily allowed for an operand, e.g., if an operand requires two words, the last word in a data area cannot be designated as the first word of the operand because all words for a single operand must be 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.

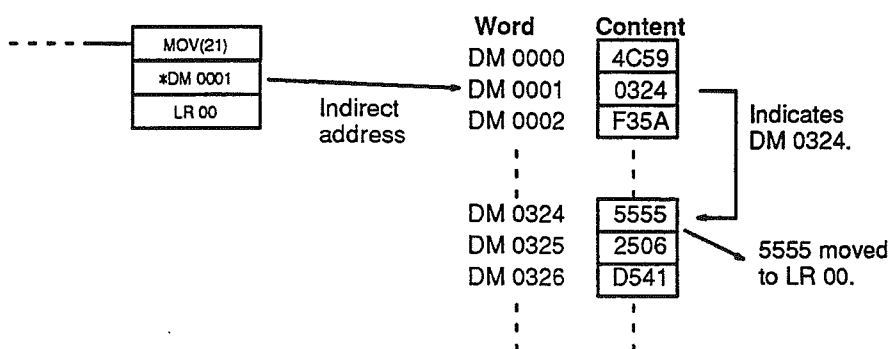
Abbreviation	Name	Bit
ER	Instruction Execution Error Flag	25503
CY	Carry Flag	25504
GR	Greater Than Flag	25505
EQ	Equals Flag	25506
LE	Less Than Flag	25507

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 A 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).

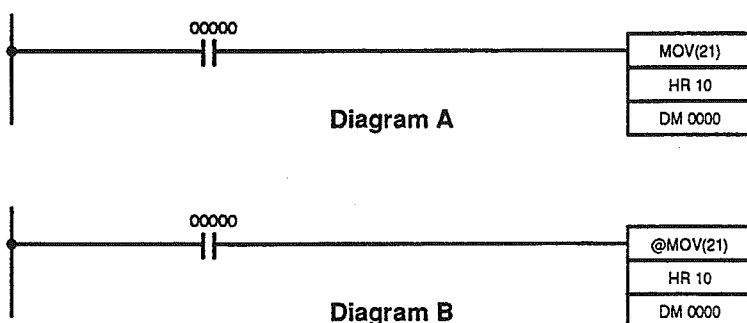
Designating Constants

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.



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

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.

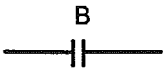
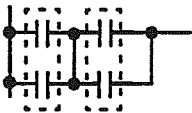
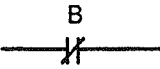
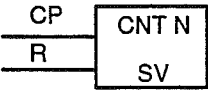
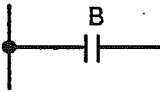
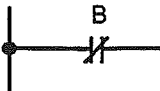

In diagram B, the differentiated @MOV(21) will move the content of HR 10 to DM 0000 only once after 00000 goes ON. Even if 00000 remains ON for 2.0 seconds with the same 80 ms cycle time, the move operation will be executed only during the first cycle in which 00000 has changed from OFF to ON. Because 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.

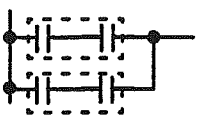


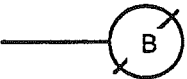
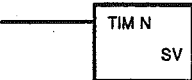

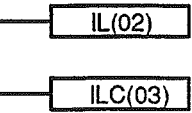
All operands, ladder diagram symbols, and other specifications for instructions are the same regardless of whether the differentiated or non-differentiated form of an instruction is used. When inputting, the same function codes are also used, but NOT is input after the function code to designate the differentiated form of an instruction. Most, but not all, instructions have differentiated forms.




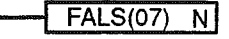


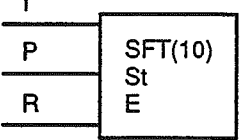
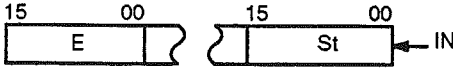
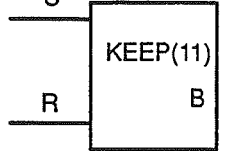
The C20HB-TS also provides differentiation instructions: DIFU(13) and DIFD(14). DIFU(13) operates the same as a differentiated instruction, but is used to turn ON a bit for one cycle. DIFD(14) also turns ON a bit for one cycle, but does it when the execution condition has changed from ON to OFF. Refer to 5-7-2 DIFFERENTIATE UP and DOWN – DIFU(13) and DIFD(14) for details.

5-5 C20HB-TS Instruction Set

The following table provides a brief summary of each instruction in the C20HB-TS instruction set. The first 12 instructions do not have function codes. These 12 instructions are listed in alphabetic order. The remaining instructions are listed in numerical order of their function codes.


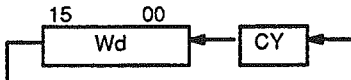

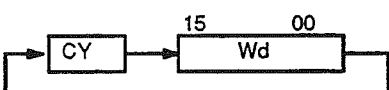

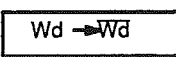
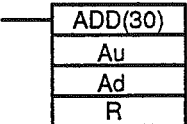
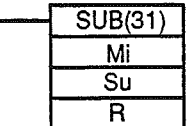
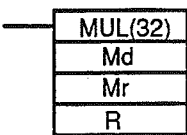
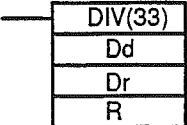
Name Mnemonic	Symbol	Function	Operand Data Areas
AND AND		Logically ANDs the status of the designated bit with the current execution condition.	B: IR SR HR AR LR TC
AND LOAD AND LD		Logically ANDs the resultant execution conditions of the preceding logic blocks.	None
AND NOT AND NOT		Logically ANDs the inverse of the designated bit with the current execution condition.	B: IR SR HR AR LR TC
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 #
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
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
OR OR		Logically ORs the status of the designated bit with the current execution condition.	B: IR SR HR AR LR TC

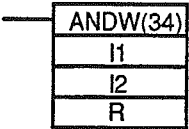
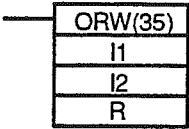
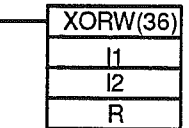
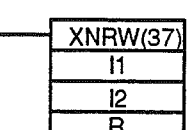
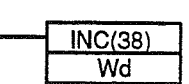
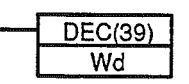


Name Mnemonic	Symbol	Function	Operand Data Areas
OR LOAD OR LD		Logically ORs the resultant execution conditions of the preceding logic blocks.	None
OR NOT OR NOT		Logically ORs the inverse of the designated bit with the execution condition.	B: IR SR HR AR LR TC
OUTPUT OUT		Turns ON B for an ON execution condition; turns OFF B for an OFF execution condition.	B: IR SR HR AR LR TR
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
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 #
NO OPERATION NOP(00)	None	Nothing is executed and program operation moves to the next instruction.	None
END END(01)		Required at the end of each program. Instructions located after END(01) will not be executed.	None
INTERLOCK IL(02) INTERLOCK CLEAR 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

Name Mnemonic	Symbol	Function	Operand Data Areas
JUMP JMP(04) JUMP END JME(05)	 	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
FAILURE ALARM (@)FAL(06)		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
SEVERE FAILURE ALARM FALS(07)		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
STEP DEFINE STEP(08)		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
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
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
KEEP KEEP(11)		Defines a bit (B) as a latch, controlled by the set (S) and reset (R) inputs.	B: IR HR AR LR

Name Mnemonic	Symbol	Function	Operand Data Areas
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 AR LR DM #
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
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 #
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
REVERSIBLE WORD SHIFT (@)RWS(17)		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: IR SR HR AR LR TC DM # St/E: IR SR HR AR LR TC DM
CYCLE TIME (@)SCAN(18)		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 SR HR AR LR TC DM # --- Not used.
COMPARE (@)CMP(20)		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 #

Name Mnemonic	Symbol	Function	Operand Data Areas	
MOVE (@)MOV(21)	<div><div>MOV(21)</div><div>S</div><div>D</div></div>	Transfers data from source word, (S) to destination word (D).	S: IR SR HR AR LR TC DM #	D: IR HR AR LR DM
MOVE NOT (@)MVN(22)	<div><div>MVN(22)</div><div>S</div><div>D</div></div>	Transfers the inverse of the data in the source word (S) to destination word (D).	S: IR SR HR AR LR TC DM #	D: IR HR AR LR DM
BCD-TO-BINARY (@)BIN(23)	<div><div>BIN(23)</div><div>S</div><div>R</div></div> <div><div><div>S (BCD)</div><div></div><div></div><div></div><div></div><div>x10⁰</div><div>x10¹</div><div>x10²</div><div>x10³</div></div><div>→</div><div><div>R (BIN)</div><div></div><div></div><div></div><div></div><div>x16⁰</div><div>x16¹</div><div>x16²</div><div>x16³</div></div></div>	Converts 4-digit, BCD data in source word (S) into 16-bit binary data, and outputs converted data to result word (R).	S: IR SR HR AR LR TC DM	R: IR HR AR LR DM
BINARY-TO-BCD (@)BCD(24)	<div><div>BCD(24)</div><div>S</div><div>R</div></div> <div><div><div>S (BIN)</div><div></div><div></div><div></div><div></div><div>x16⁰</div><div>x16¹</div><div>x16²</div><div>x16³</div></div><div>→</div><div><div>R (BCD)</div><div></div><div></div><div></div><div></div><div>x10⁰</div><div>x10¹</div><div>x10²</div><div>x10³</div></div></div>	Converts binary data in source word (S) into BCD, and outputs converted data to result word (R).	S: IR SR HR AR LR DM	R: IR HR AR LR DM
ARITHMETIC SHIFT LEFT (@)ASL(25)	<div><div>ASL(25)</div><div>Wd</div></div> <div><div>CY</div><div>←</div><div><div>15</div><div>00</div><div>Wd</div></div><div>←</div><div>0</div></div>	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	
ARITHMETIC SHIFT RIGHT (@)ASR(26)	<div><div>ASR(26)</div><div>Wd</div></div> <div><div>0</div><div>→</div><div><div>15</div><div>00</div><div>Wd</div></div><div>→</div><div>CY</div></div>	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	

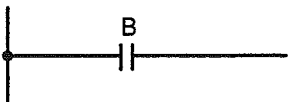
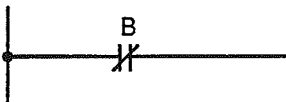
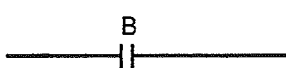
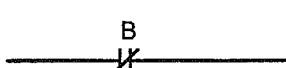
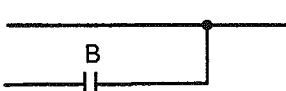
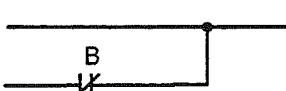
Name Mnemonic	Symbol	Function	Operand Data Areas
ROTATE LEFT (@)ROL(27)		Each bit within a single word of data (Wd) is moved one bit to the left, with bit 15 moving to carry (CY), and CY moving to bit 00. 	Wd: IR HR AR LR DM
ROTATE RIGHT (@)ROR(28)		Each bit within a single word of data (Wd) is moved one bit to the right, with bit 00 moving to carry (CY), and CY moving to bit 15. 	Wd: IR HR AR LR DM
COMPLEMENT (@)COM(29)		Inverts bit status of one word (Wd) of data, changing 0s to 1s, and vice versa. 	Wd: IR HR AR LR DM
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
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
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. $Md \times Mr \rightarrow \boxed{R + 1} \quad \boxed{R}$	Md/Mr: IR SR HR AR LR TC DM # R: IR HR AR LR DM
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. $Dd \div Dr \rightarrow \boxed{R + 1} \quad \boxed{R}$	Dd/Dr: IR SR HR AR LR TC DM # R: IR HR AR LR DM

Name Mnemonic	Symbol	Function	Operand Data Areas
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
LOGICAL OR (@)ORW(35)		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
EXCLUSIVE OR (@)XORW(36)		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
EXCLUSIVE NOR (@)XNRW(37)		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
INCREMENT (@)INC(38)		Increments the value of a 4-digit BCD word (Wd) by one, without affecting carry (CY).	Wd: IR HR AR LR DM
DECREMENT (@)DEC(39)		Decrements the value of a 4-digit BCD word by 1, without affecting carry (CY).	Wd: IR HR AR LR DM
SET CARRY (@)STC(40)		Sets the Carry Flag (i.e., turns CY ON).	None
CLEAR CARRY (@)CLC(41)		Clears the Carry Flag (i.e., turns CY OFF).	None

5-6 Ladder Diagram Instructions

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

5-6-1 LOAD, LOAD NOT, AND, AND NOT, OR, and OR NOT

	Ladder Symbols	Operand Data Areas		
LOAD – LD		<table><tr><td>B: Bit</td></tr><tr><td>IR, SR, AR, HR, TC, LR, TR</td></tr></table>	B: Bit	IR, SR, AR, HR, TC, LR, TR
B: Bit				
IR, SR, AR, HR, TC, LR, TR				
LOAD NOT – LD NOT		<table><tr><td>B: Bit</td></tr><tr><td>IR, SR, AR, HR, TC, LR</td></tr></table>	B: Bit	IR, SR, AR, HR, TC, LR
B: Bit				
IR, SR, AR, HR, TC, LR				
AND – AND		<table><tr><td>B: Bit</td></tr><tr><td>IR, SR, AR, HR, TC, LR</td></tr></table>	B: Bit	IR, SR, AR, HR, TC, LR
B: Bit				
IR, SR, AR, HR, TC, LR				
AND NOT – AND NOT		<table><tr><td>B: Bit</td></tr><tr><td>IR, SR, AR, HR, TC, LR</td></tr></table>	B: Bit	IR, SR, AR, HR, TC, LR
B: Bit				
IR, SR, AR, HR, TC, LR				
OR – OR		<table><tr><td>B: Bit</td></tr><tr><td>IR, SR, AR, HR, TC, LR</td></tr></table>	B: Bit	IR, SR, AR, HR, TC, LR
B: Bit				
IR, SR, AR, HR, TC, LR				
OR NOT – OR NOT		<table><tr><td>B: Bit</td></tr><tr><td>IR, SR, AR, HR, TC, LR</td></tr></table>	B: Bit	IR, SR, AR, HR, TC, LR
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

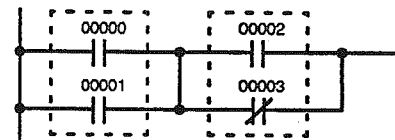
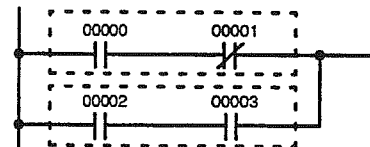
These six basic instructions correspond to the conditions on a ladder diagram. As described in *Section 4 Writing and Entering Programs*, the status of the bits assigned to each instruction determines the execution conditions for all other instructions. Each of these instructions and each bit address 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.

Flags

There are no flags affected by these instructions.

5-6-2 AND LOAD and OR LOAD**AND LOAD – AND LD****Ladder Symbol****OR LOAD – OR LD****Ladder Symbol****Description**

When instructions are combined into blocks that cannot be logically combined using only OR and AND operations, AND LD and OR LD are used. Whereas AND and OR operations logically combine a bit status and an execution condition, AND LD and OR LD logically combine two execution conditions, the current one and the last unused one.

In order to draw ladder diagrams, it is not necessary to use AND LD and OR LD instructions, nor are they necessary when inputting ladder diagrams directly, as is possible from LSS. They are required, however, to convert the program to and input it in mnemonic form. The procedures for these, limitations for different procedures, and examples are provided in *4-6 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.

Flags

There are no flags affected by these instructions.

5-6-3 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 96 for a sample ladder diagram and corresponding mnemonic code written on program coding sheet from *Appendix C*.

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.

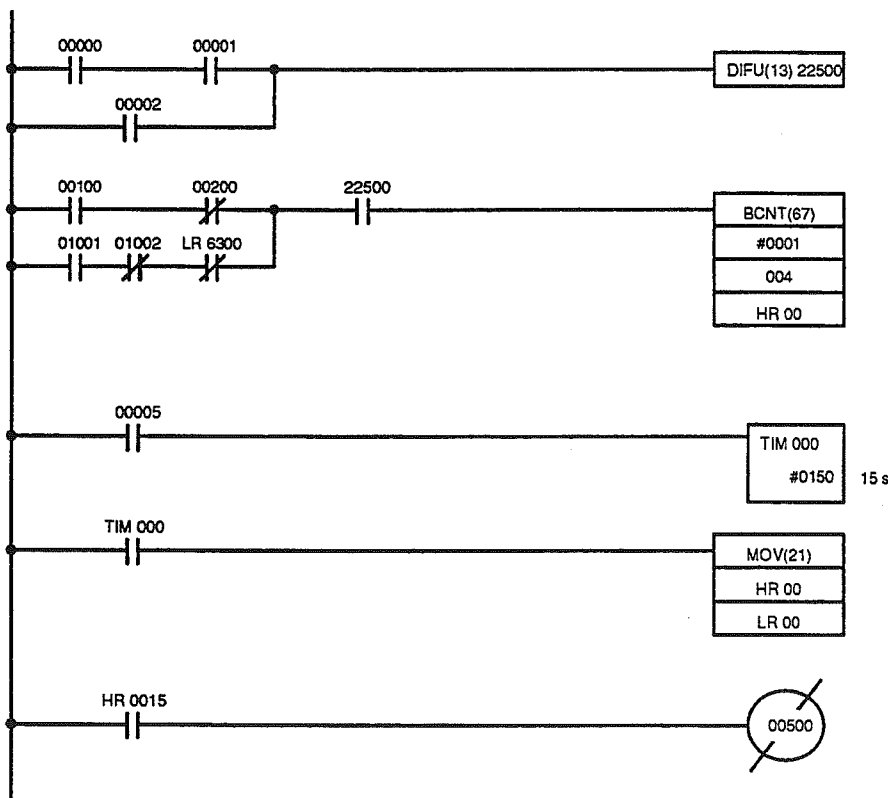
The address and instruction columns of the mnemonic code table are filled in for the instruction word only. For all other lines, the left two columns are left

blank. If the instruction requires no definier 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 placed on the right side. If a constant is to be input, the number symbol (#) is placed on the left side of the data column and the number to be input is placed on the right side. Any numbers input as definiers 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.

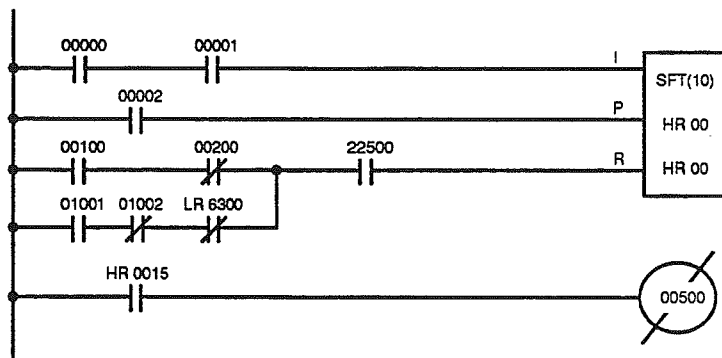
The following diagram and corresponding mnemonic code illustrate the points described above.



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.



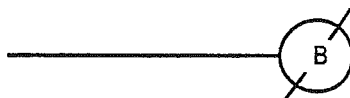
Address	Instruction	Operands
00000	LD	00000
00001	AND	00001
00002	LD	00002
00003	LD	00100
00004	AND NOT	00200
00005	LD	01001
00006	AND NOT	01002
00007	AND NOT	LR 6300
00008	OR LD	—
00009	AND	22500
00010	SFT(10)	—
		HR 00
		HR 00
00011	LD	HR 0015
00012	OUT NOT	00500

5-7 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.

5-7-1 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.

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.

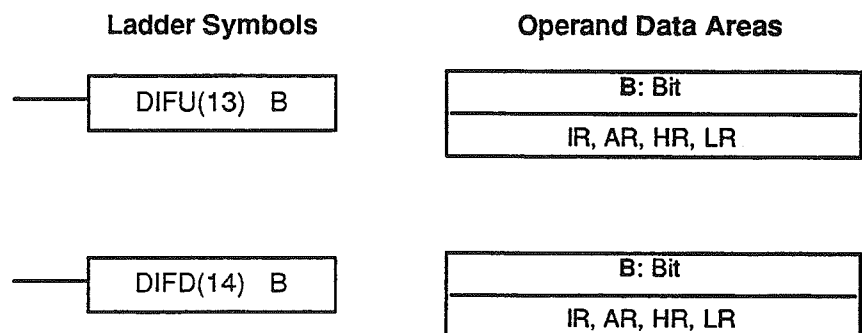
OUT NOT turns ON the designated bit for a OFF execution condition, and turns OFF the designated bit for an ON execution condition.

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 5-8-1 *TIMER – TIM* for details.

Flags

There are no flags affected by these instructions.

5-7-2 DIFFERENTIATE UP and DOWN – DIFU(13) and DIFD(14)**Limitations**

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

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

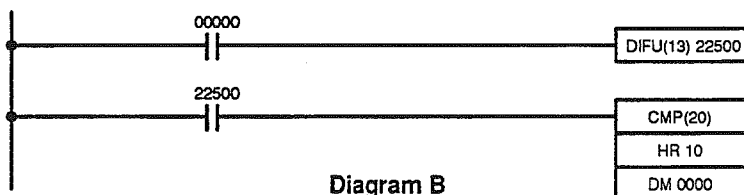
DIFU(13) and DIFD(14) operation can be uncertain when the instructions are programmed between IL and ILC or between JMP and JME.

**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.



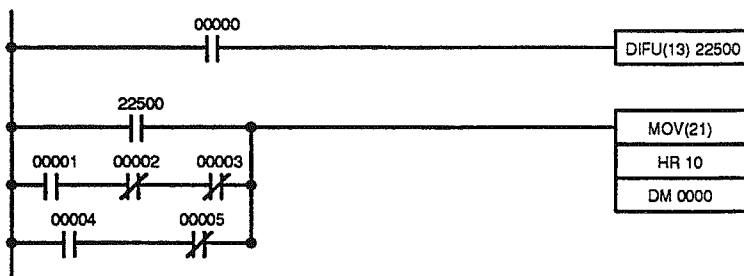
Address	Instruction	Operands
00000	LD	00000
00001	CMP(20)	
		HR 10
		DM 0000



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**

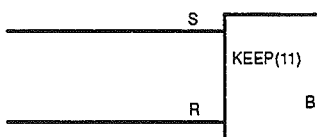
Although a differentiated form of MOV(21) is available, the following diagram would be very complicated to draw using it because only one of the conditions determining the execution condition for MOV(21) requires differentiated treatment.



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

5-7-3 KEEP – KEEP(11)

Ladder Symbol



Operand Data Areas

B: Bit
IR, AR, HR, LR

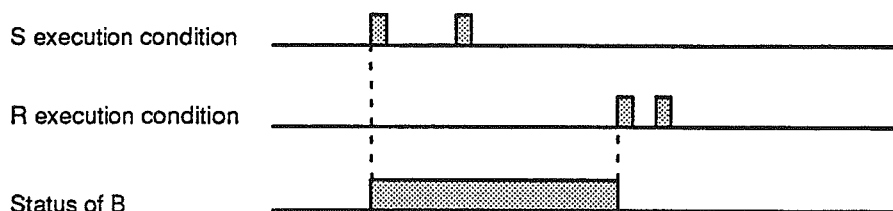
Limitations

Any output bit can generally be used in only one instruction that controls its status. Refer to 3-2 IR Area for details.

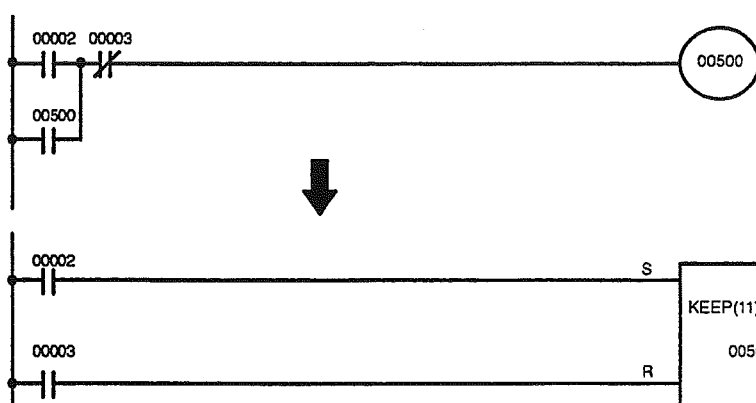
Description

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

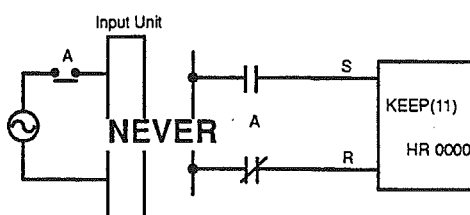
Address	Instruction	Operands
00000	LD	00002
00001	LD	00003
00002	KEEP(11)	00500

Flags

There are no flags affected by this instruction.

Precautions

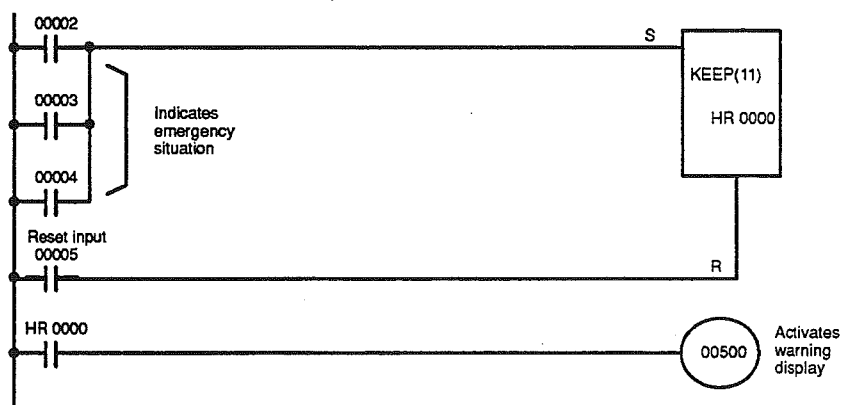
Never use an input bit in an inverse condition on the reset (R) for KEEP(11) when the input device uses an AC power supply. The delay in shutting down the PC's DC power supply (relative to the AC power supply to the input device) can cause the designated bit of KEEP(11) to be reset. This situation is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



Bits used in KEEP are not reset in interlocks.

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 5-8-1 *TIMER – TIM* for details.

5-8 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.

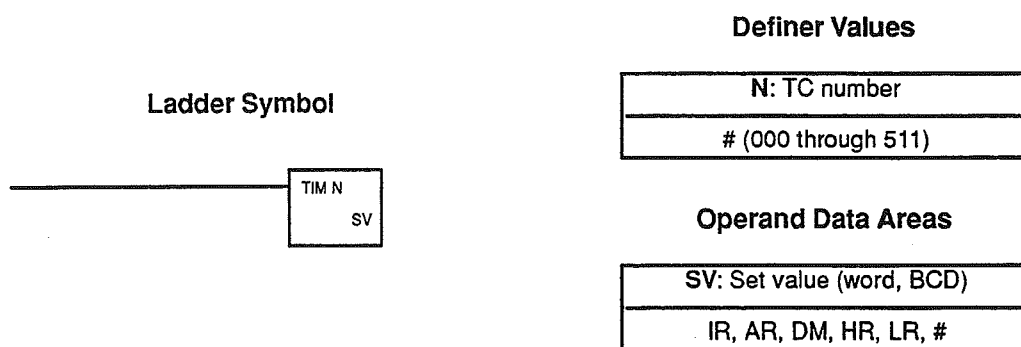
TC numbers run from 000 through 511. No prefix is required when using a TC number as a definer in a timer or counter instruction. Once defined as a timer, a TC number can be prefixed with TIM for use as an operand in certain instructions. The TIM prefix is used regardless of the timer instruction that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

5-8-1 TIMER – TIM



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.

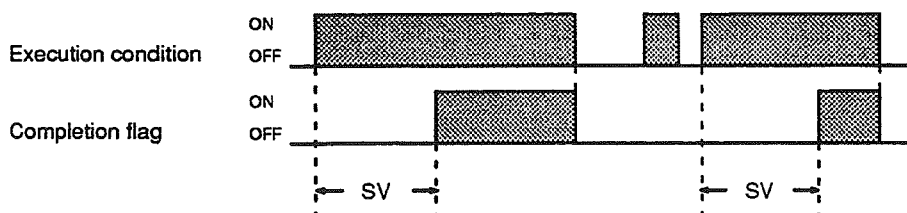
TC 000 through TC 003 should not be used in TIM if they are required for TIMH(15). Refer to 5-8-2 *HIGH-SPEED TIMER – TIMH(15)* for details.

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 5-8-3 *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.

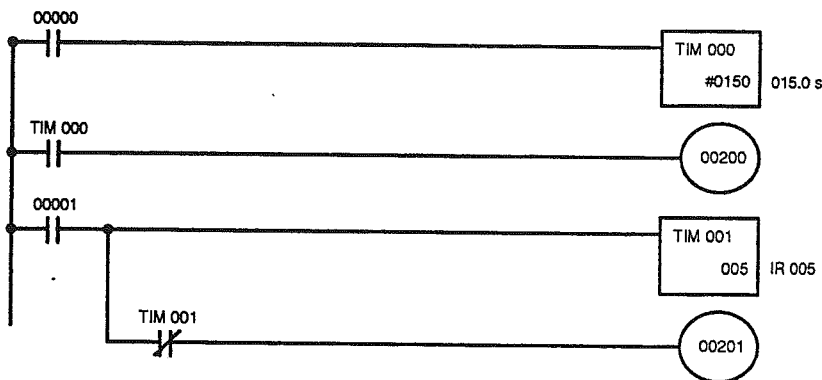
Indirectly addressed DM word is non-existent. (Content of *DM word is not BCD, or the DM area boundary has been exceeded.)

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

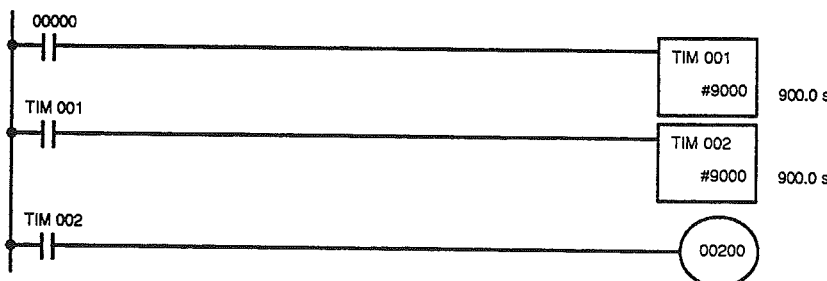
The following example shows two timers, one set with a constant and one set via input word 005. Here, 00200 will be turned ON after 00000 goes ON and stays ON for at least 15 seconds. When 00000 goes OFF, the timer will be reset and 00200 will be turned OFF. When 00001 goes ON, TIM 001 is started from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

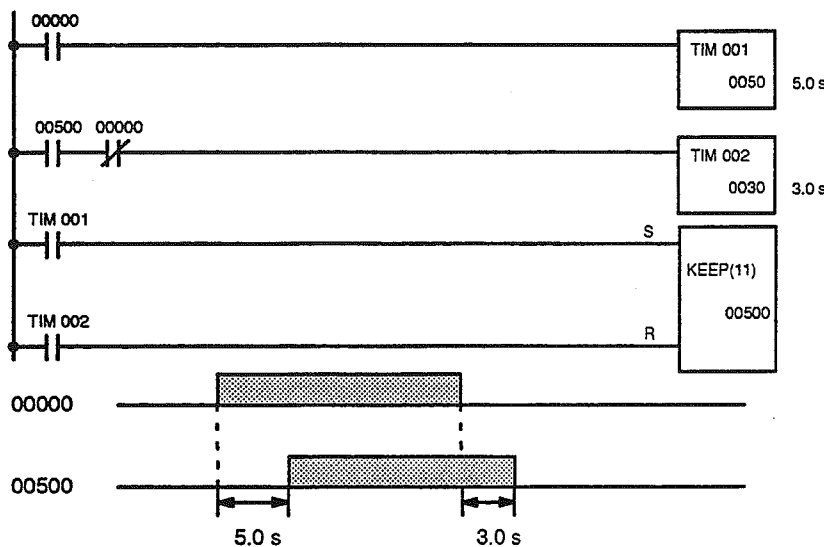
In this example, 00200 will be turned ON 30 minutes after 00000 goes ON. TIM can also be combined with CNT or CNT can be used to count SR area clock pulse bits to produce longer timers. An example is provided in 5-8-3 COUNTER – CNT.

**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 5-7-3 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).

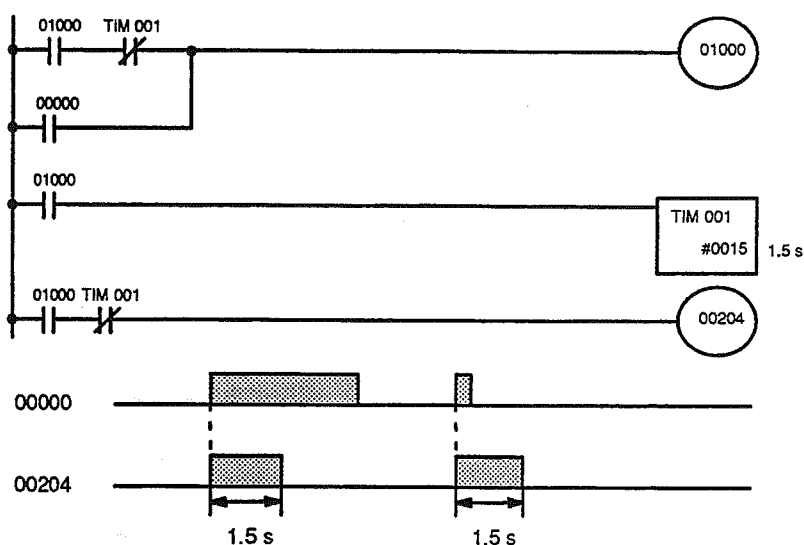


Address	Instruction	Operands
00000	LD	00000
00001	TIM	001
		# 0050
00002	LD	00500
00003	AND NOT	00000
00004	TIM	002
		# 0030
00005	LD	TIM 001
00006	LD	TIM 002
00007	KEEP(11)	00500

**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

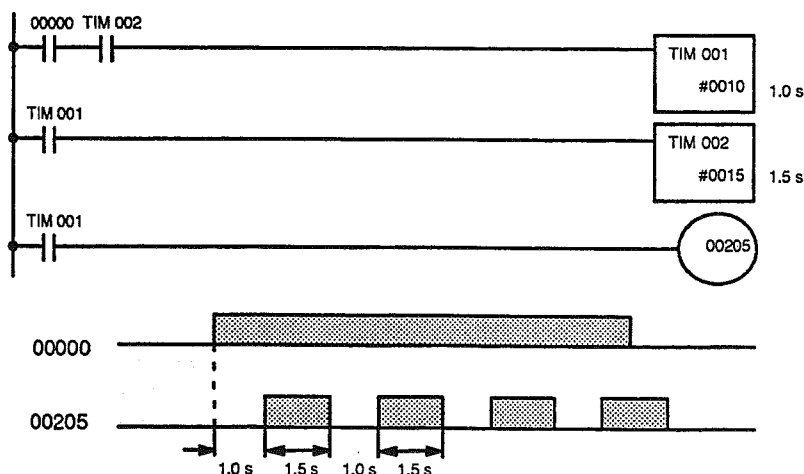
turn ON which, as an inverse condition, creates an OFF execution condition for OUT 00204).



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.

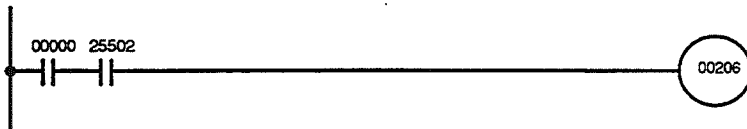


Address	Instruction	Operands
00000	LD	00000
00001	AND	TIM 002
00002	TIM	001
		# 0010
00003	LD	TIM 001
00004	TIM	002
		# 0015
00005	LD	TIM 001
00006	OUT	00205

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

5-8-2 HIGH-SPEED TIMER – TIMH(15)

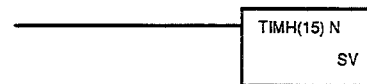
Definer Values

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

Operand Data Areas

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

Ladder Symbol



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 5-8-1 *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 5-8-3 *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.

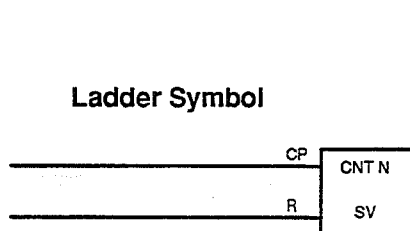
Program execution will continue even if a non-BCD SV is used, but timing will not be accurate.

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.)

5-8-3 COUNTER – CNT



Definer Values

N: TC number
(000 through 511)

Operand Data Areas

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

Limitations

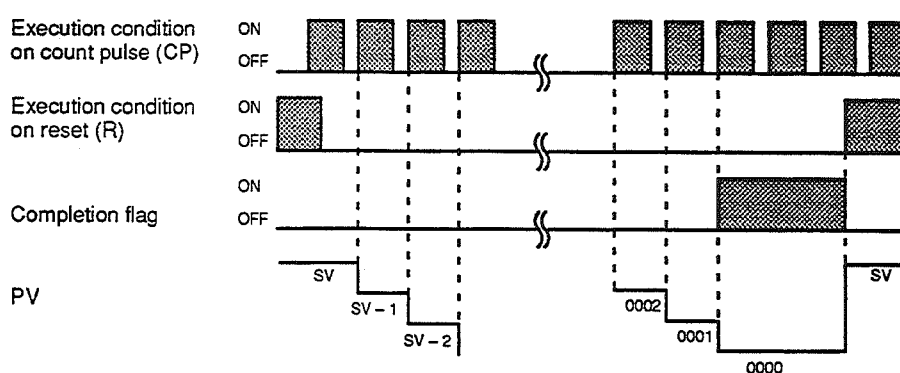
Each TC number can be used as the definer in only one TIMER or COUNTER instruction.

Description

CNT is used to count down from SV when the execution condition on the count pulse, CP, goes from OFF to ON, i.e., the present value (PV) will be decremented by one whenever CNT is executed with an ON execution condition for CP and the execution condition was OFF for the last execution. If the execution condition has not changed or has changed from ON to OFF, the PV of CNT will not be changed. The completion flag for a counter is turned ON when the PV reaches zero and will remain ON until the counter is reset.

CNT is reset with a reset input, R. When R goes from OFF to ON, the PV is reset to SV. The PV will not be decremented while R is ON. Counting down from SV will begin again when R goes OFF. The PV for CNT will not be reset in interlocked program sections or by power interruptions.

Changes in execution conditions, the completion flag, and the PV are illustrated below. PV line height is meant only to indicate changes in the PV.



Precautions

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

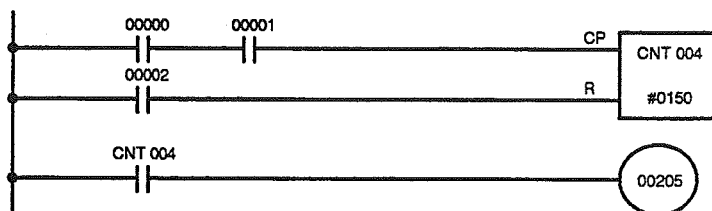
Flags

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**

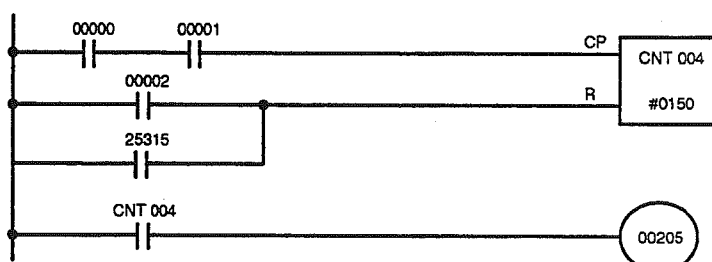
In the following example, the PV will be decremented whenever both 00000 and 00001 are ON provided that 00002 is OFF and either 00000 or 00001 was OFF the last time CNT 004 was executed. When 150 pulses have been counted down (i.e., when PV reaches zero), 00205 will be turned ON.



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

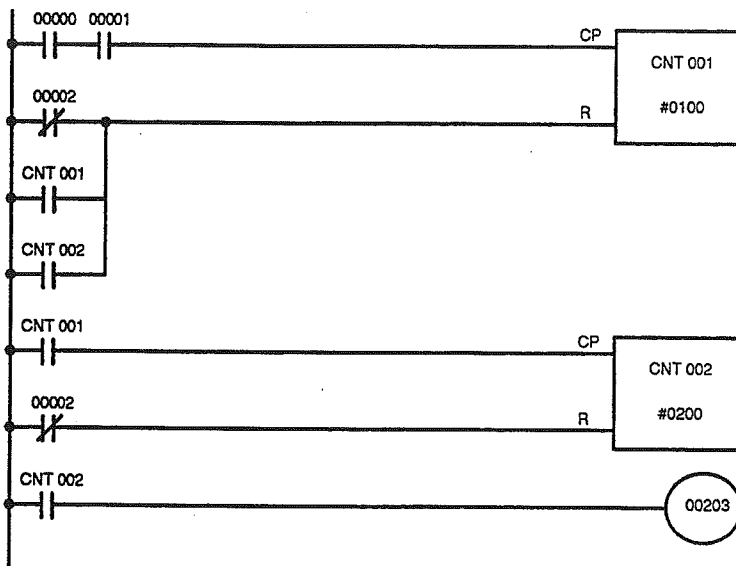
**Example 2:
Extended Counter**

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

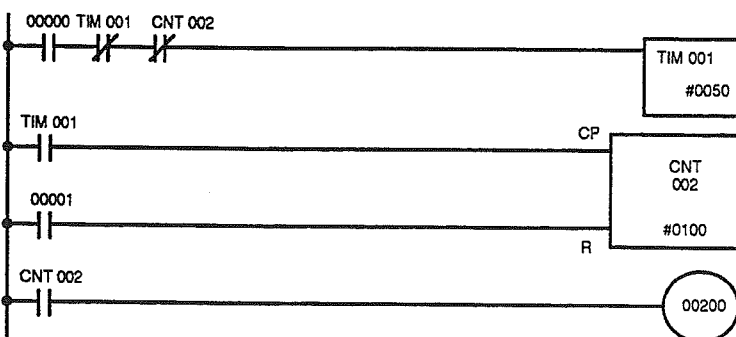
CNT can be used in sequence as many times as required to produce counters capable of counting any desired values.

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 it runs continuously and CNT 002 counts the number of times the completion flag for TIM 001 goes ON (CNT 002 would be executed once each time 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.

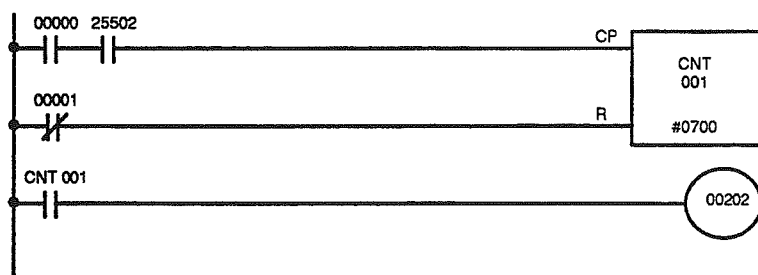
Because in this example the SV for TIM 001 is 5.0 seconds and the SV for CNT 002 is 100, the completion flag for CNT 002 turns ON when 5 seconds x 100 times, i.e., 500 seconds (or 8 minutes and 20 seconds) have expired. This would result in 00201 being turned ON.



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.

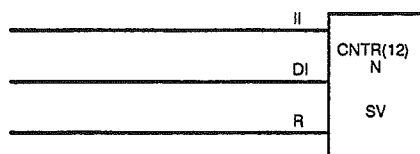


Address	Instruction	Operands
00000	LD	00000
00001	AND	25502
00002	LD NOT	00001
00003	CNT	001
		# 0700
00004	LD	CNT 001
00005	OUT	00202

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.

5-8-4 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

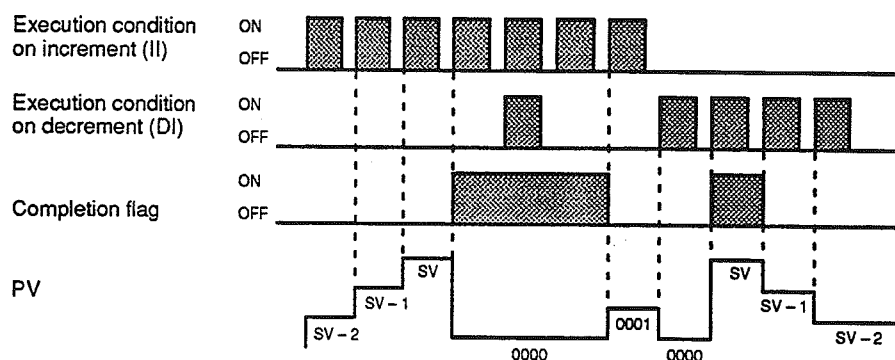
The CNTR(12) is a reversible, up/down circular counter, i.e., it is used to count between zero and SV according to changes in two execution conditions, those in the increment input (II) and those in the decrement input (DI). The present value (PV) will be incremented by one whenever CNTR(12) is executed 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.



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.)

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.

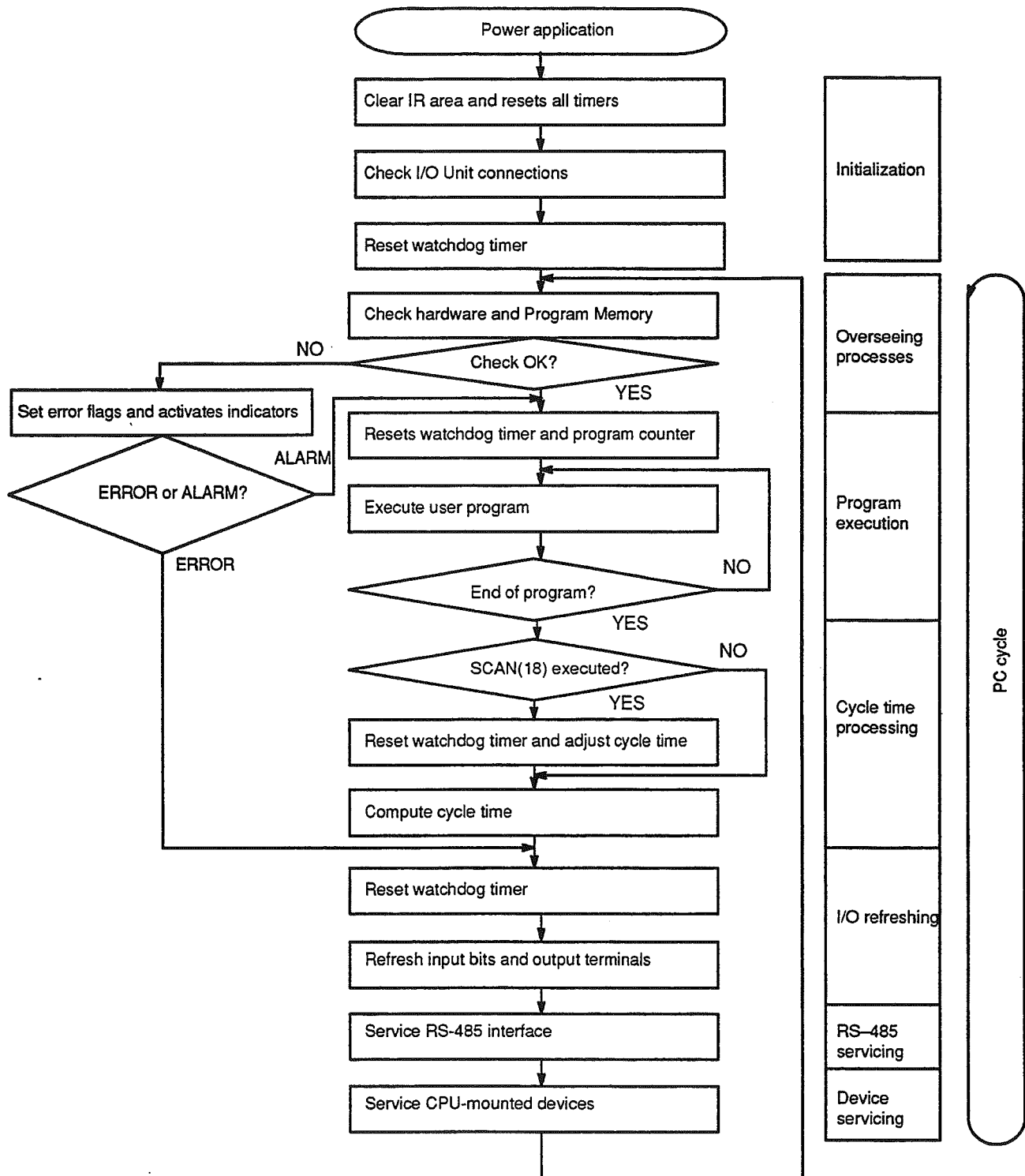
6-1	Cycle Time	114
6-2	Instruction Execution Times	117
6-3	I/O Response Time	121
6-4	Host Link Response Time	122

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.



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 six 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	3.0 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 6-2 <i>Instruction Execution Times</i> 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.
RS-485 servicing	Built-in RS-485 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%.
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.

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 Instruction Execution Times

This following table lists the execution times for all instructions that are available for the C20HB-TS. 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 (μ s)*	OFF execution time (μ s)*
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
	*DM for SV	265	JMP: 49
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
	Normal cycle, Constant for SV	169	JMP: 73
	Interrupt, *DM for SV	149	R: 291 IL: 291
	Normal cycle, *DM for SV	169	JMP: 73
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 → word	166	3.75
	*DM + *DM → *DM	593	
SUB(31)	Constant + word → word	192	3.75
	*DM – *DM → *DM	600	
MUL(32)	Constant x word → word	634	3.75
	*DM x *DM → word	1045	
DIV(33)	Word ÷ constant → word	737	3.75
	*DM ÷ *DM → *DM	1156	
ANDW(34)	Constant < word → word	162	3.75
	*DM < *DM → *DM	557	
ORW(35)	Constant > word → word	162	3.75
	*DM > *DM → *DM	560	
XORW(36)	Constant XORW word → word	162	3.75
	*DM XORW *DM → *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 → word	163	3.75
	*DM XNRW *DM → *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

Note: * The execution time is given in microseconds unless otherwise stated.

6-3 I/O Response Time

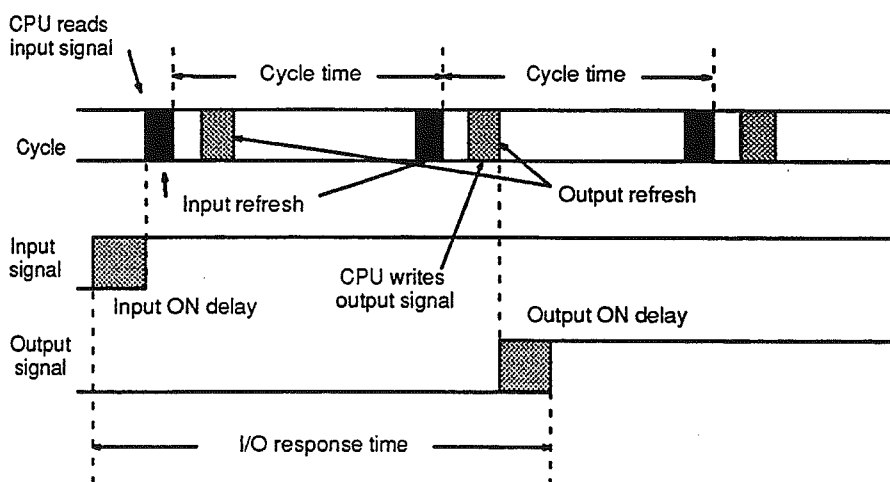
The I/O response time is the time it takes for the PC to output a control signal after it has received an input signal. The time it takes to respond depends on the cycle time and when the CPU receives the input signal relative to the input refresh period.

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

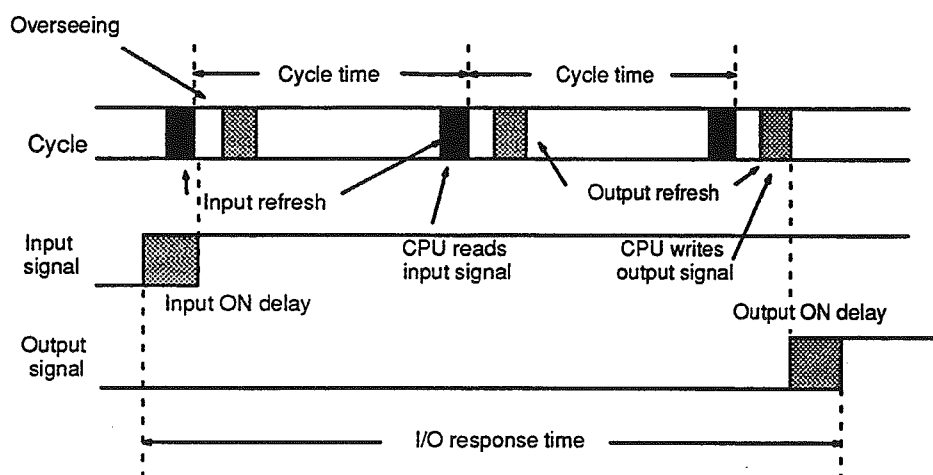
The PC responds most quickly when it receives an input signal just prior to the input refresh period in the cycle. Once the input bit corresponding to the signal has been turned ON, the program would have to be executed once to turn ON the output bit for the desired output signal and then the input refresh 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 output ON-delay time. This situation is illustrated below.



Minimum I/O response time = input ON delay + cycle time + I/O refresh time + output ON delay

Maximum I/O Response Time

The PC takes longest to respond when it receives the input signal just after the 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) + 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
Output ON-delay	15 ms

Minimum I/O response time = 1.5 + 20 + 0.23 + 15 = 36.73 ms

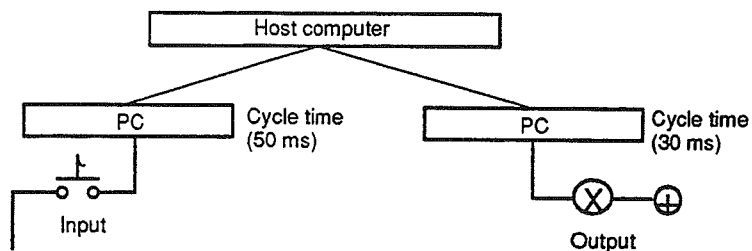
Maximum I/O response time = 1.5 + (20 x 2) + 15 = 56.5 ms

6-4 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 C20HB-TS. 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 FIT or a computer running LSS, refer to the applicable *Operation Manual* for procedures on these.

7-1	Displaying and Clearing Error Messages	126
7-2	Monitoring Operation and Modifying Data	127
7-2-1	Bit/Word Monitor	128
7-2-2	Forced Set/Reset	131
7-2-3	Forced Set/Reset Cancel	133
7-2-4	Hexadecimal/BCD Data Modification	134
7-2-5	Hex/ASCII Display Change	135
7-2-6	Program Header Display	136
7-2-7	3-word Monitor	136
7-2-8	3-word Data Modification	137
7-2-9	Binary Monitor	138
7-2-10	Binary Data Modification	139
7-2-11	Changing Timer/Counter SV	141
7-3	C250HL-PRO31-TS PROM Writer Operations	144
7-3-1	Switching to and from PROM Writer Mode	145
7-3-2	EPROM Type Setting	145
7-3-3	EPROM Erase Check	146
7-3-4	EPROM Data Read	147
7-3-5	EPROM Data Write	147
7-3-6	EPROM Data Compare	148

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.

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.

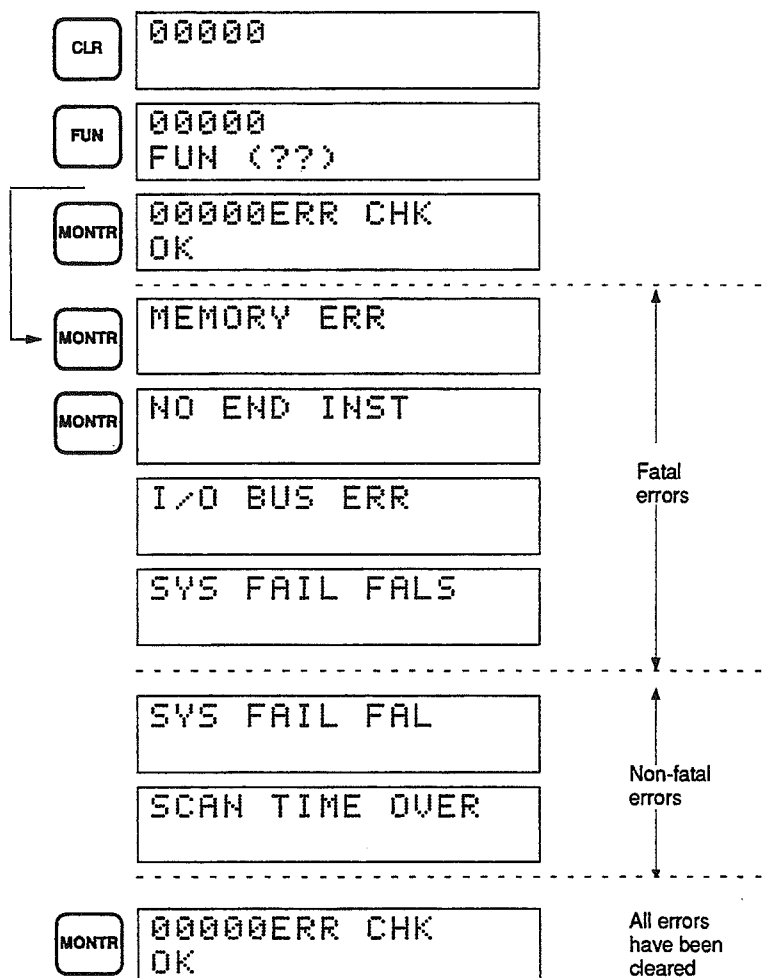
Although error messages can be displayed in any mode, they can be cleared only in PROGRAM mode. There is no way to restart the PC following a fatal error without first clearing the error message in PROGRAM mode.

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.



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 performing 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.

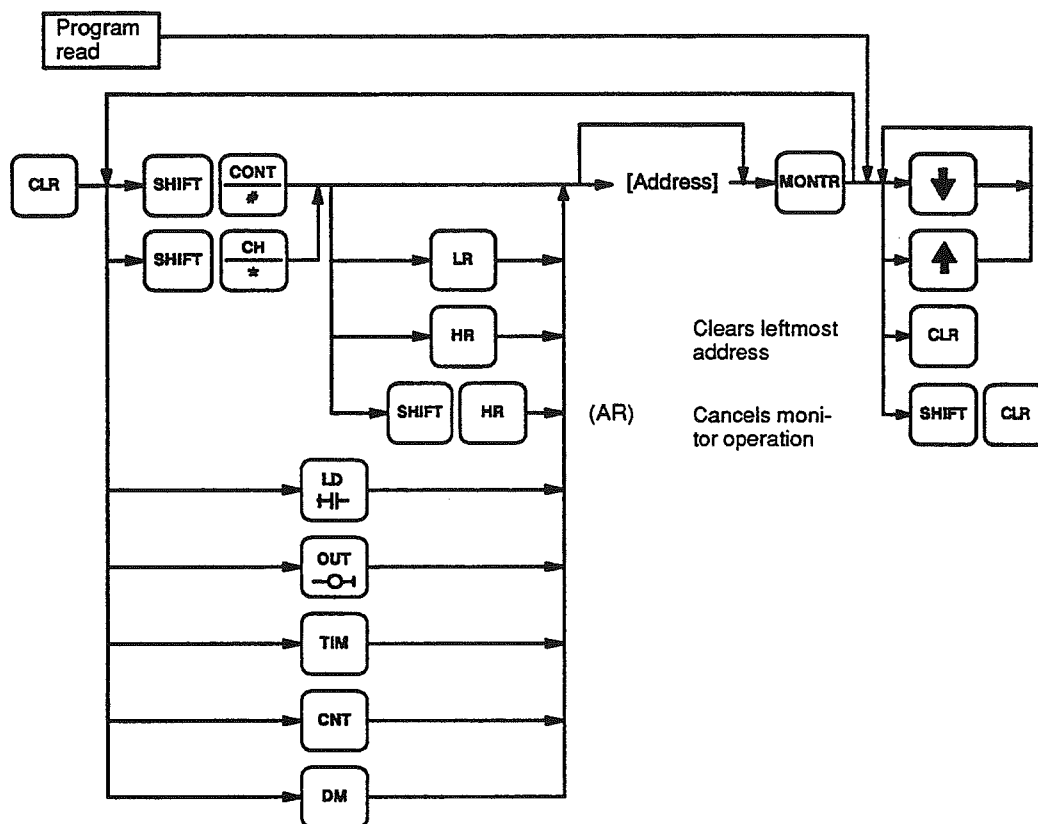
Up to six memory addresses, either bits, words, or a combination of both, can be monitored at once, although only three of these are displayed at any one time. To monitor more than one address, return to the start of the procedure and continue designating addresses. Monitoring of all designated addresses 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.

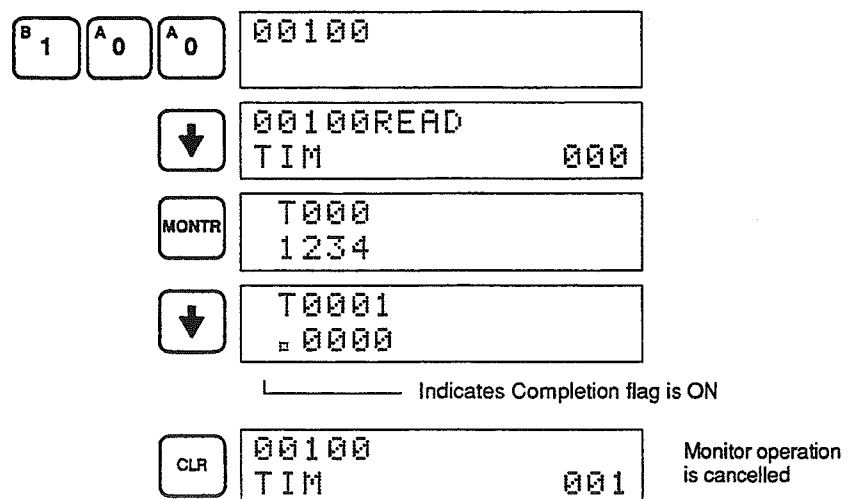
Key Sequence



Examples

The following examples show various applications of this monitor operation.

Program Read then Monitor



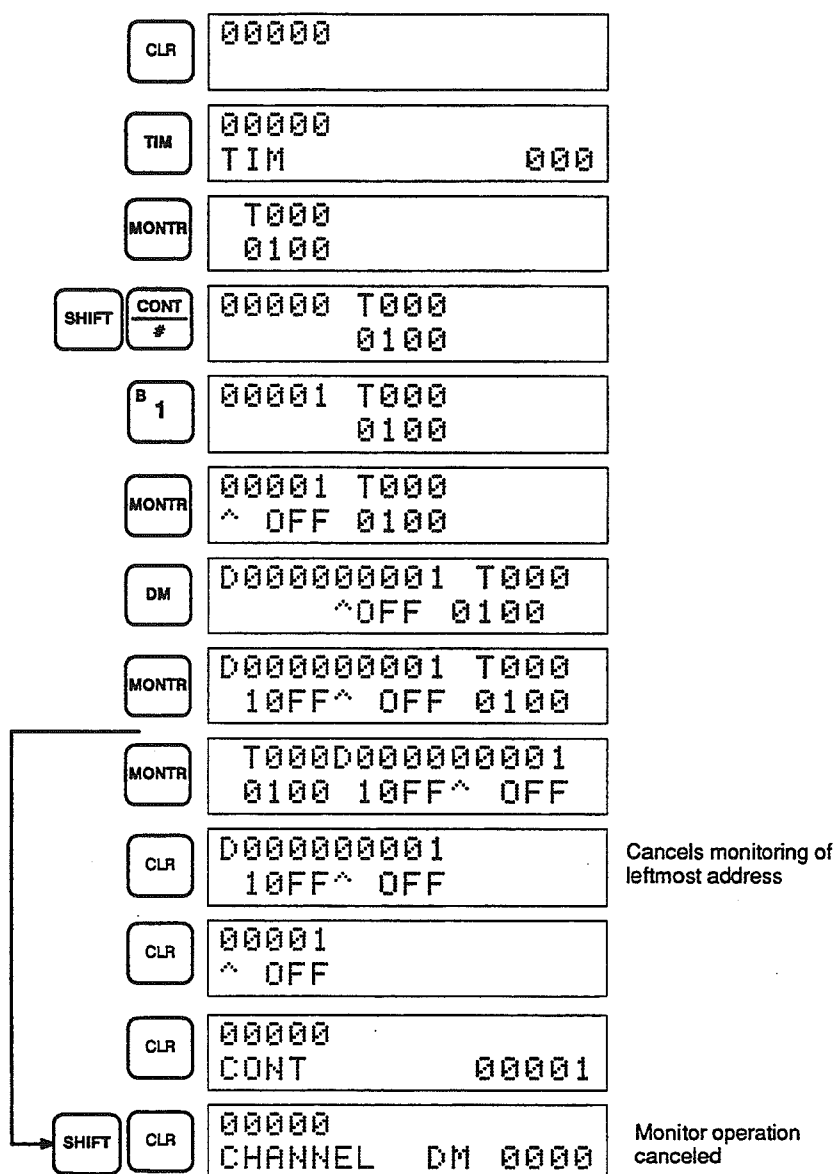
Bit Monitor

CLR		00000
LD HI	B 1	00000 LD 00001
MONTR		00001 ^ ON
CLR		00000 CONT 00001

Word Monitor

CLR		00000
SHIFT	CH *	00000 CHANNEL 000
LR	B 1	00000 CHANNEL LR 01
MONTR		cL01 FFFF
↑		cL00 0000

Multiple Address Monitoring



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.

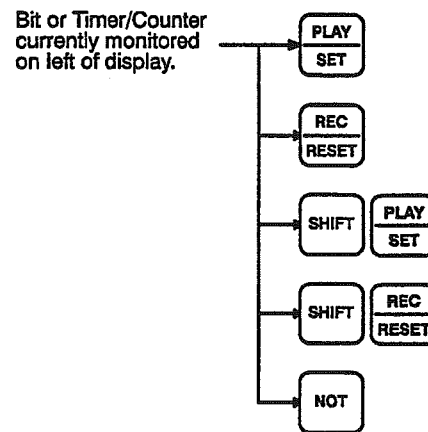
Without using NOT, the operation may be cancelled in any of the following four ways:

- 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

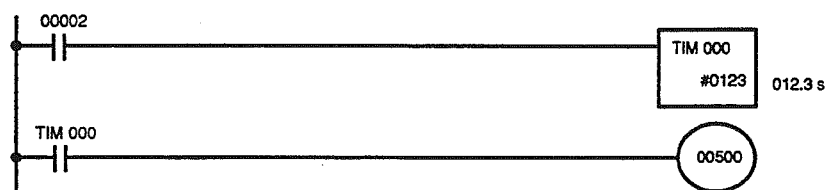
This operation can be used in MONITOR mode to check wiring of outputs from the PC prior to actual program execution. This operation cannot be used in RUN mode.

Key Sequence



Example

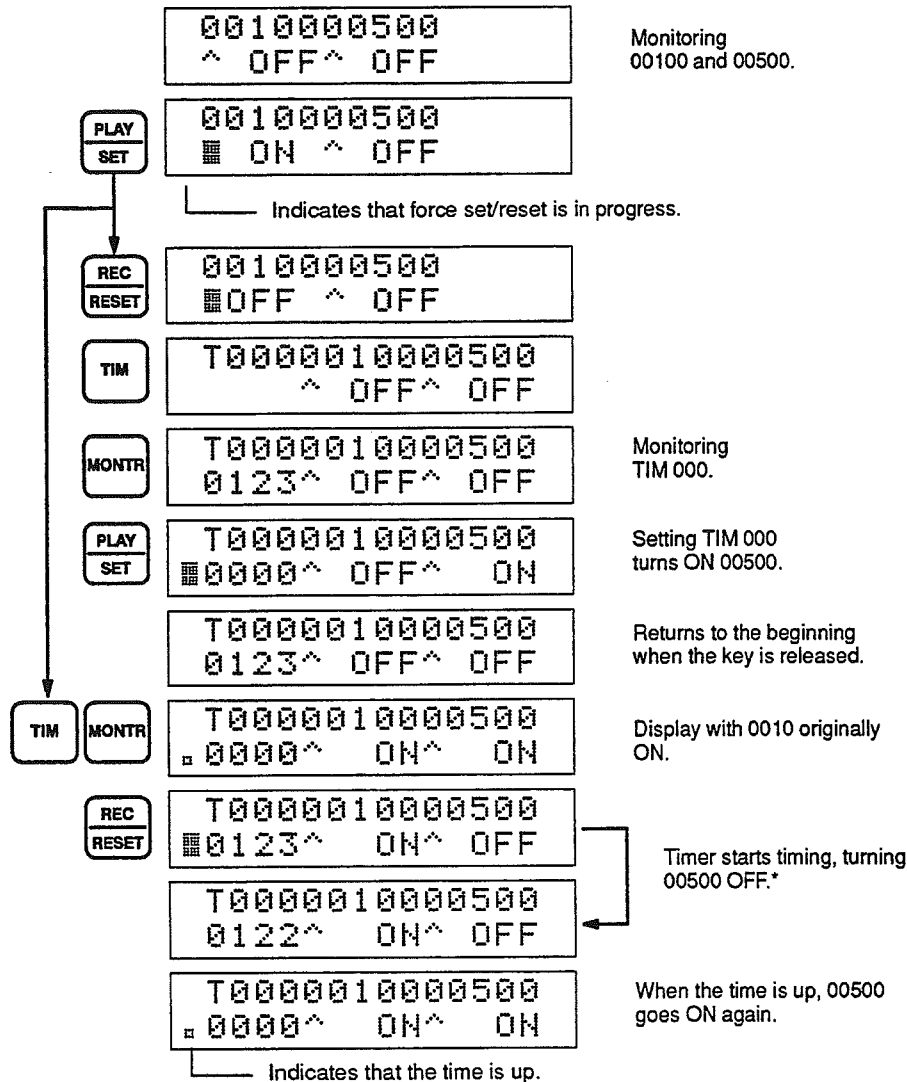
The following example shows how either bits or timers can be controlled with the Force Set/Reset operation. The displays shown below are for the following program section.



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.)



*Timing not done in PROGRAM mode.

7-2-3 Forced Set/Reset Cancel

This operation restores the status of all bits in the IR, TIM, CNT, HR, AR, or LR areas which have been force set or reset. It can be performed in PROGRAM or MONITOR mode.

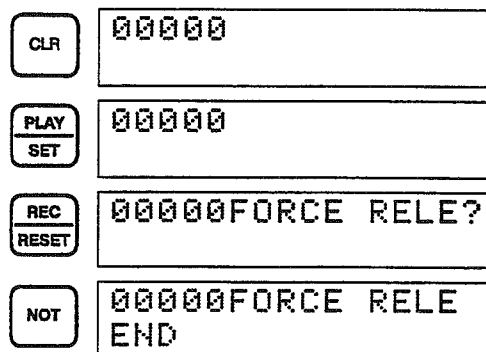
Key Sequence



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.

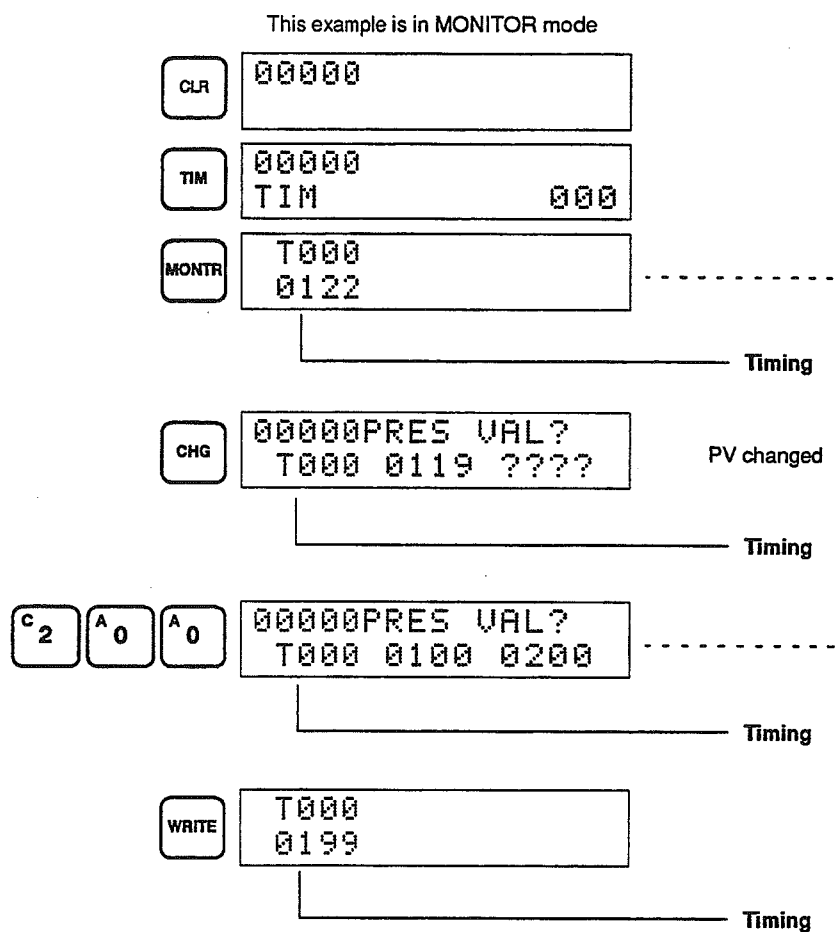
If a timer or counter is leftmost on the display, the PV will be displayed and will be the value changed. See 7-2-11 *Changing Timer/Counter SV* for the procedure to change SV. PV can be changed in MONITOR mode only when the timer or counter is operating.

To change contents of the leftmost word address, press CHG, input the desired value, and press WRITE.

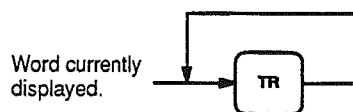
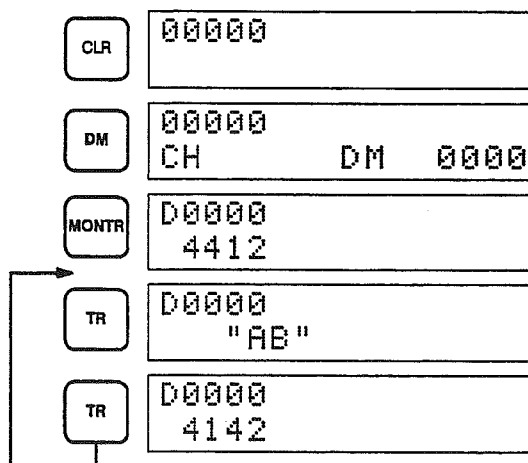
Key Sequence

Example

The following example shows the effects of changing the PV of a timer.

**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**

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.

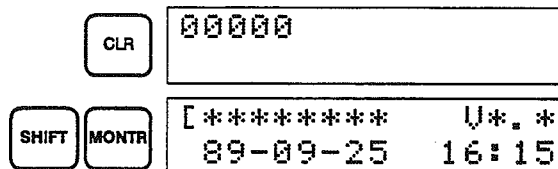
On models with a clock function, the revision time is generated automatically whenever there is an insertion, deletion, or addition to the program, or when memory is cleared or timer/counter SVs are set. For models without the clock function, it is necessary to set the revision time.

For more detail, refer to 3-5 DM (Data Memory) Area.

Key Sequence



Example

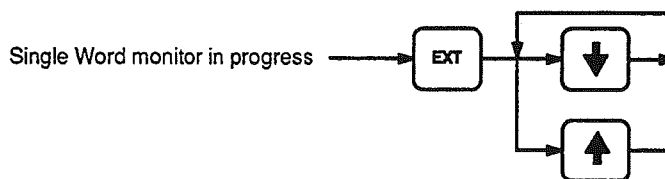


7-2-7 3-word Monitor

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



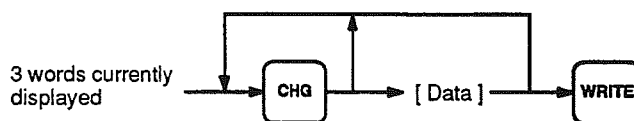
Example

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	D0003 D0002 ABCD 0123

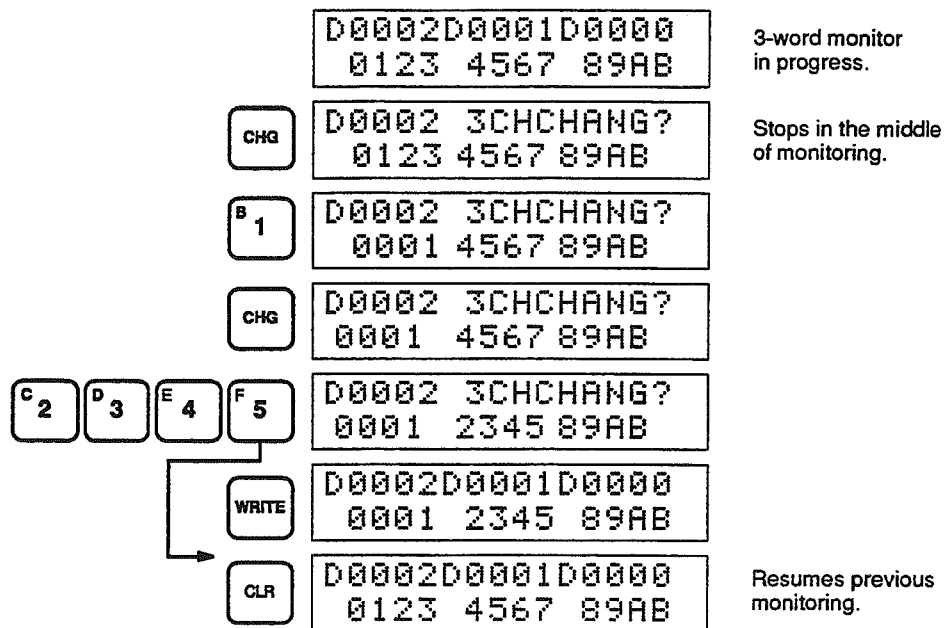
7-2-8 3-word Data Modification

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



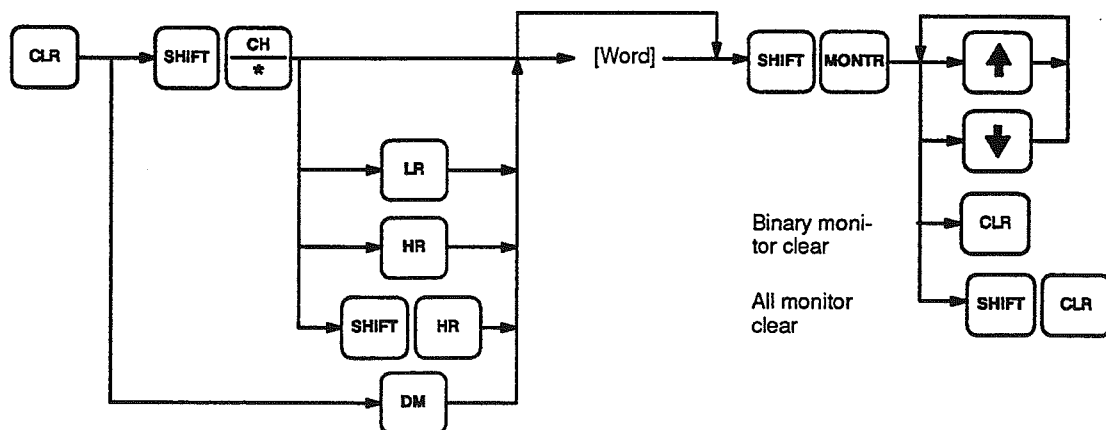
Example



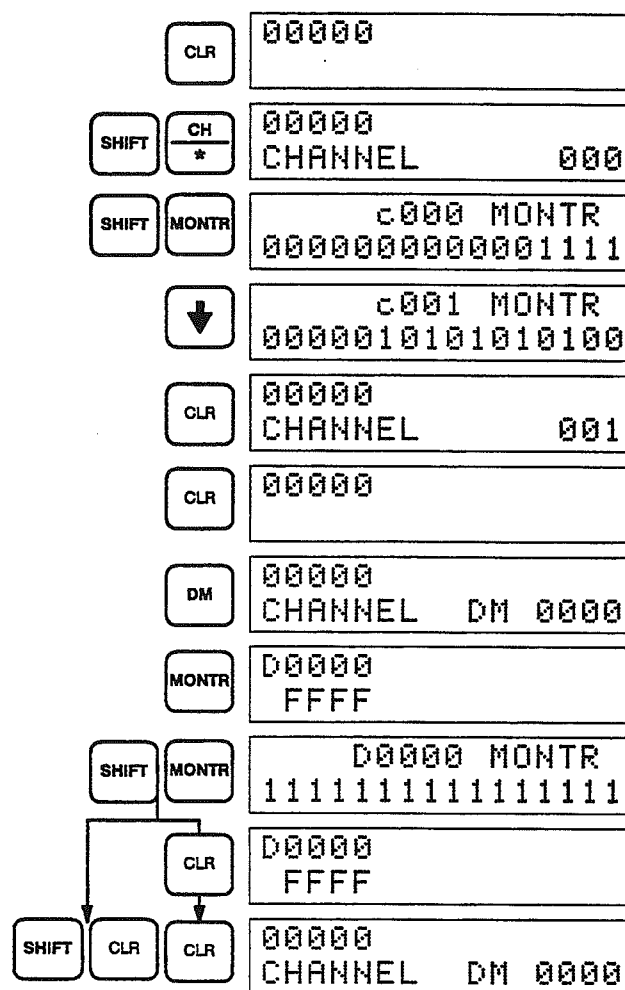
7-2-9 Binary Monitor

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



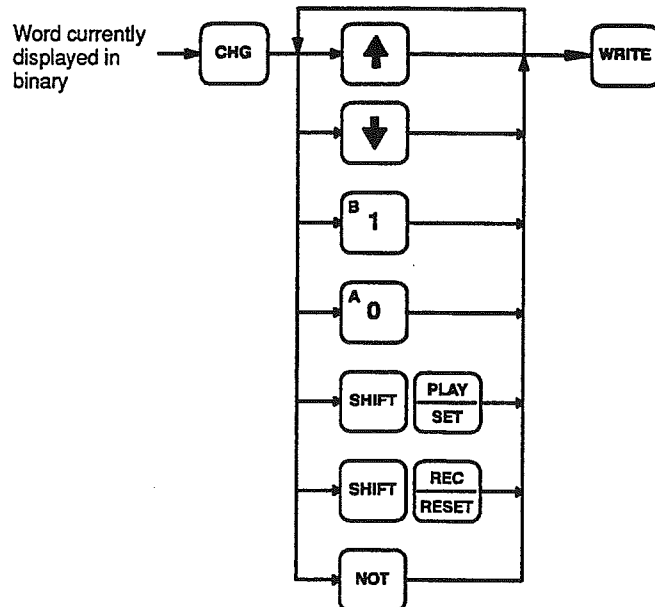
7-2-10 Binary Data Modification

This operation assigns a new 16-digit binary value to an IR, HR, AR, LR, or DM word.

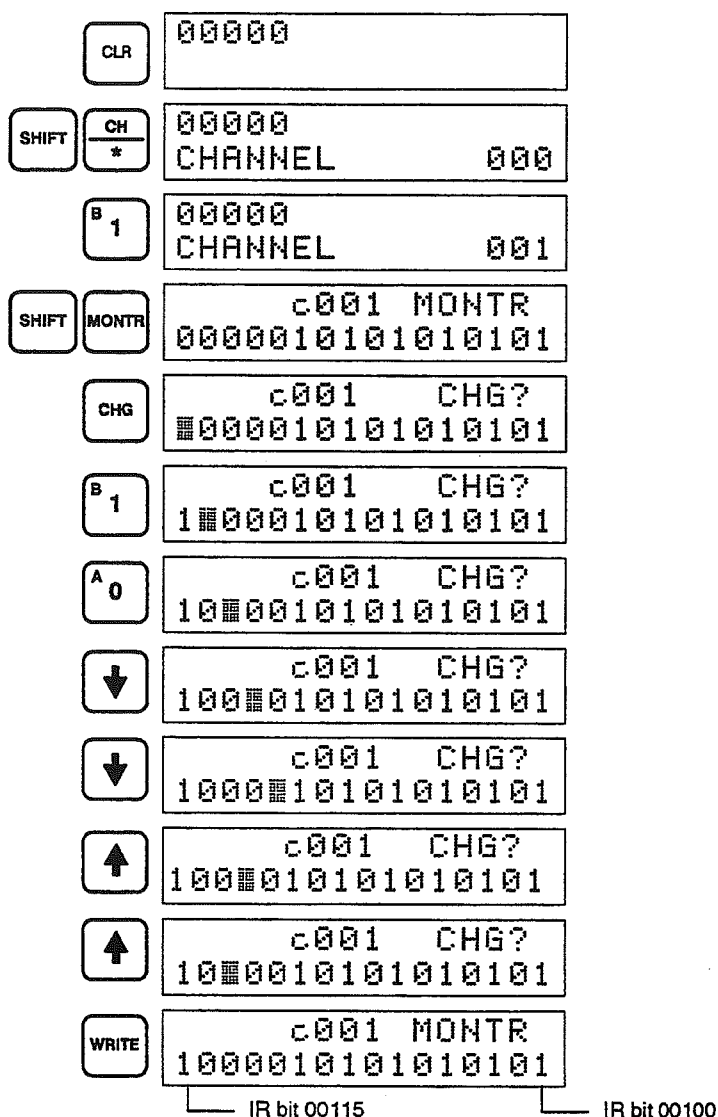
The blinking square, which can be shifted to the left with the up key and to the right with the down key, indicates the position of the bit that can be changed. After positioning to the desired bit, a 0 or a 1 can then be entered as the new bit value. Bits can be force set or force reset by pressing **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



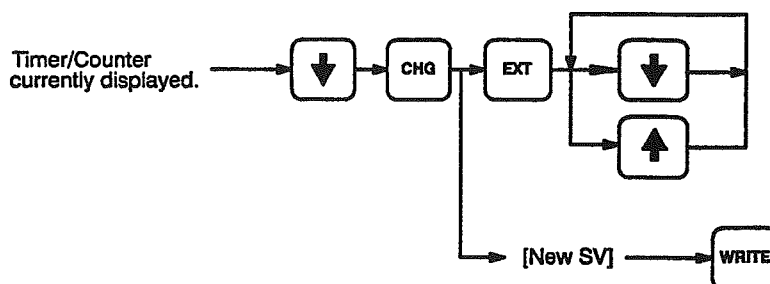
7-2-11 Changing Timer/Counter SV

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.

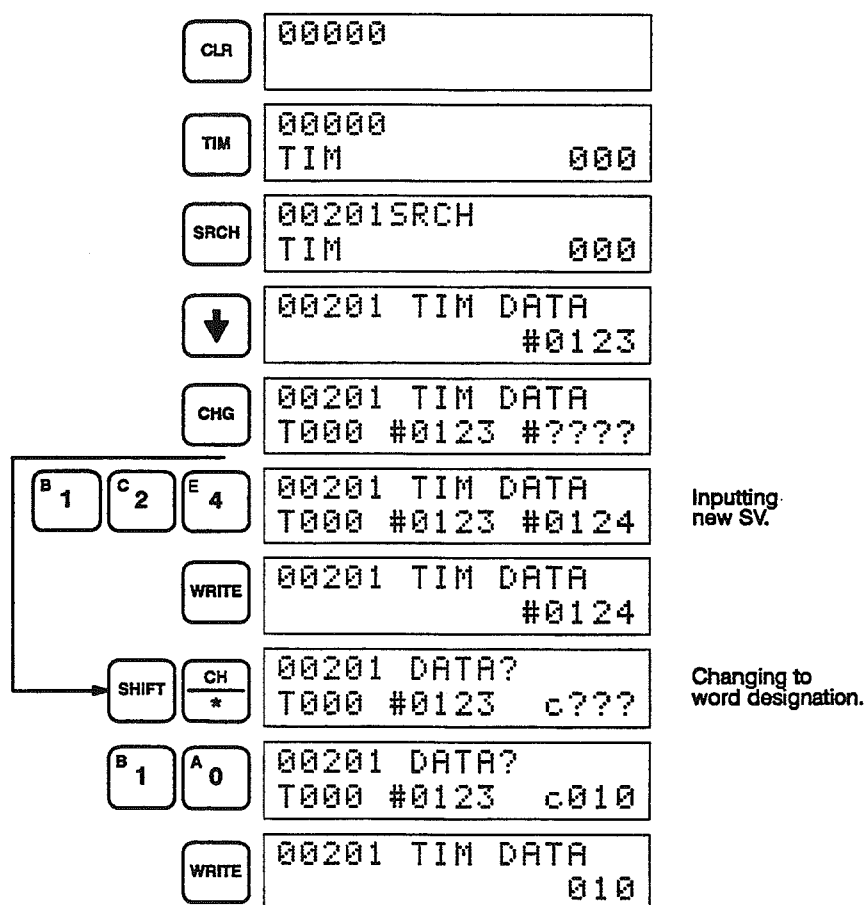
Key Sequence



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



Incrementing and
Decrementing

CLR	00000
TIM	00000 TIM 000
SRCH	00201SRCH TIM 000
↓	00201 TIM DATA #0123
CHG	00201 TIM DATA T000 #0123 #????
EXT	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
CLR	00201DATA ? T000 #0124 #????
CLR	00201 TIM DATA #0124

Returns to original display
with new SV

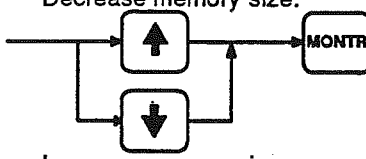
7-3 C250HL-PRO31-TS PROM Writer Operations

In addition to all of the functions of the C200H-PRO27-E Programming Console, the C250HL-PRO31-TS Programming Console has been equipped with a built-in PROM Writer.

The C250HL-PRO31-TS's Programming Console operations are identical to those of the C200H-PRO27-E. Refer to 7-1 *Displaying and Clearing Error Messages* and 7-2 *Monitoring Operation and Modifying Data* for details on the Programming Console operations.

PROM Writer Operations

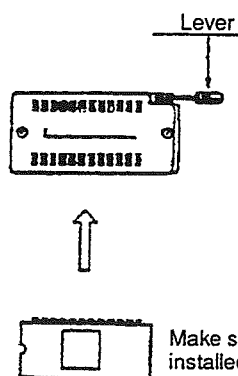
The PROM Writer can be used to transfer the Program Memory and DM 1000 to DM 1999 to PROM, transfer the Program Memory and DM 1000 to DM 1999 from EPROM to the PC, or compare the EPROM data to the PC data. The five PROM Writer operations are described briefly in the following table.

Operation	Function	Key Sequence
EPROM Type Setting	Specifies the type of EPROM being used (2764, 64A, 128, 128A, or 256). (There are 8K words of User Memory in the C20HB-SC001-TS, so a 64K-bit EPROM cannot be used. Use either 27128, 27128A, or 27256 ROM.)	Decrease memory size.  Increase memory size.
EPROM Erase Check	Performs an erase check.	DEL → MONTR
EPROM Data Read	Reads data from the EPROM chip and transfers it to the PC.	PLAY SET → MONTR
EPROM Data Write	Copies PC data to the EPROM chip if it passes an erase check.	WRITE → MONTR
EPROM Data Compare	Compares the EPROM chip data to the PC data.	VER → MONTR

Installing an EPROM Chip

The following procedure explains how to install and remove an EPROM chip from the PROM Writer socket in the C250HL-PRO31-TS.

- 1, 2, 3...
1. Raise the socket's lever to release the socket armature.
 2. Insert the EPROM chip into the socket. When inserting the chip, hold it up-right and be sure not to touch the terminal pins. The notch on the chip should be on the right side.



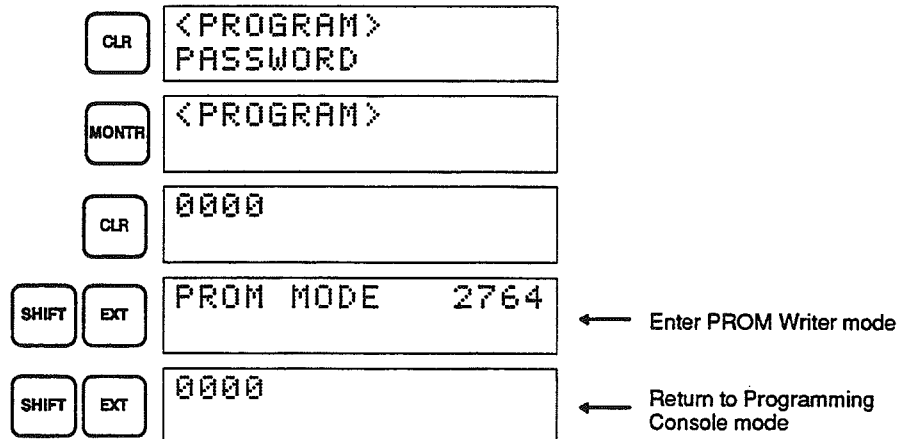
Make sure that the chip is being installed in the right direction.

3. Check that the chip is inserted properly and depress the lever.

7-3-1 Switching to and from PROM Writer Mode

The C250HL-PRO31-TS can be switched between Programming Console mode and PROM Writer mode by pressing the SHIFT + EXT keys. The Programming Console must be set to PROGRAM mode in order to switch from Programming Console mode to PROM Writer mode.

Example

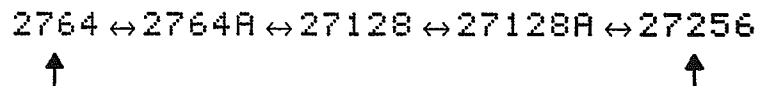


7-3-2 EPROM Type Setting

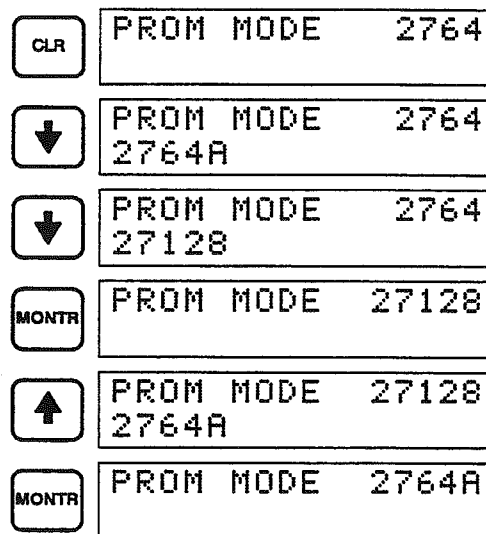
This operation is used to select the type of EPROM chip being used. The five possible settings are: 2764, 2764A, 27128, 27128A, and 27256.

To change the EPROM setting, press the Up and Down Arrow keys to display different settings and then press the MON key to select the displayed setting.

The Up Arrow key will change the setting to a smaller memory size and the Down Arrow key will change the setting to a larger memory size, in the order shown below.



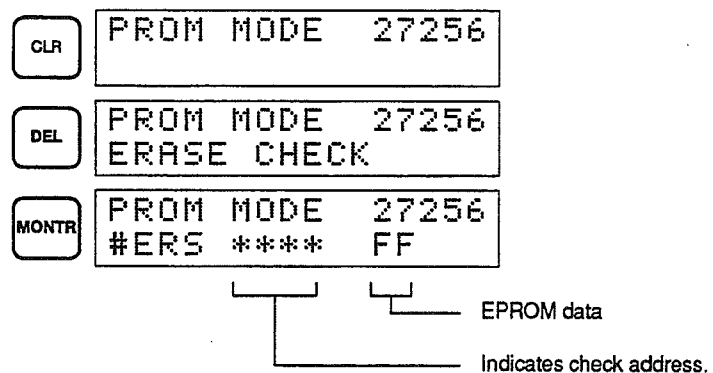
Example



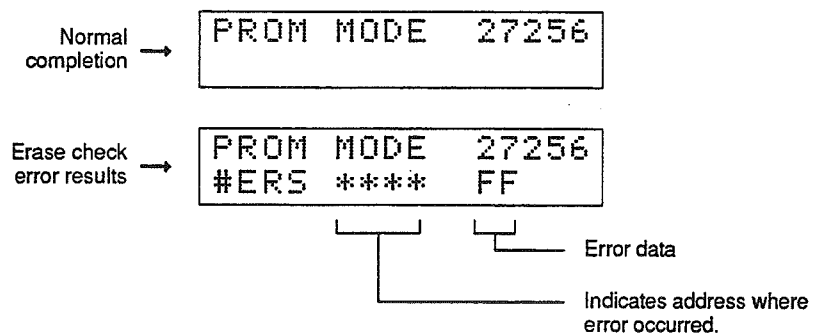
7-3-3 EPROM Erase Check

This operation performs an erase check on the EPROM chip in the PROM Writer socket.

Example



Erase Check Results

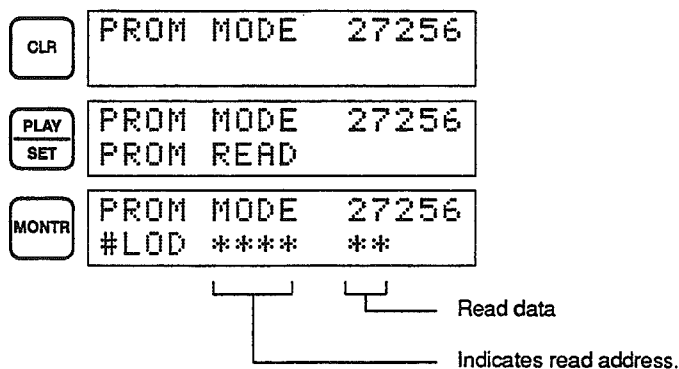


7-3-4 EPROM Data Read

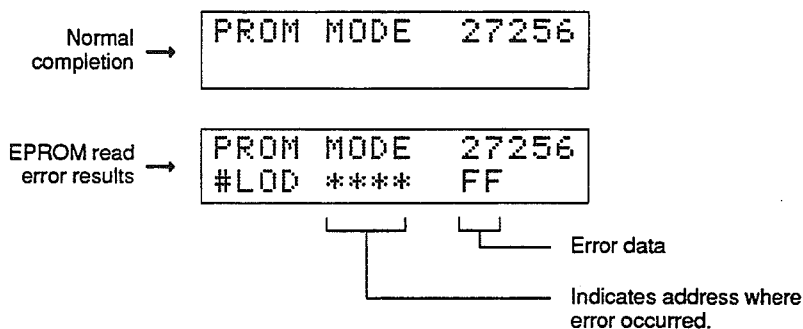
The EPROM Data Read operation reads data from the EPROM chip and transfers it to the PC. This operation is possible only when the PC's User Memory write-protect switch (pin 5 of DIP switch 2) is set to ON.

When the EPROM setting is set for 27256 EPROM, only the last 16K bytes of the chip's data will be transferred. (The 16K bytes of data in addresses \$2000 through \$3FFF will be transferred, but the 16K bytes of data in addresses \$0000 through \$1FFF will not be transferred.)

Example



EPROM Read Results

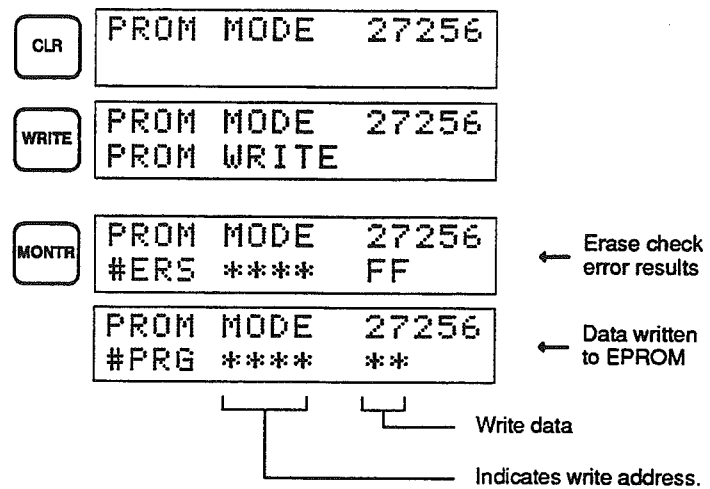


7-3-5 EPROM Data Write

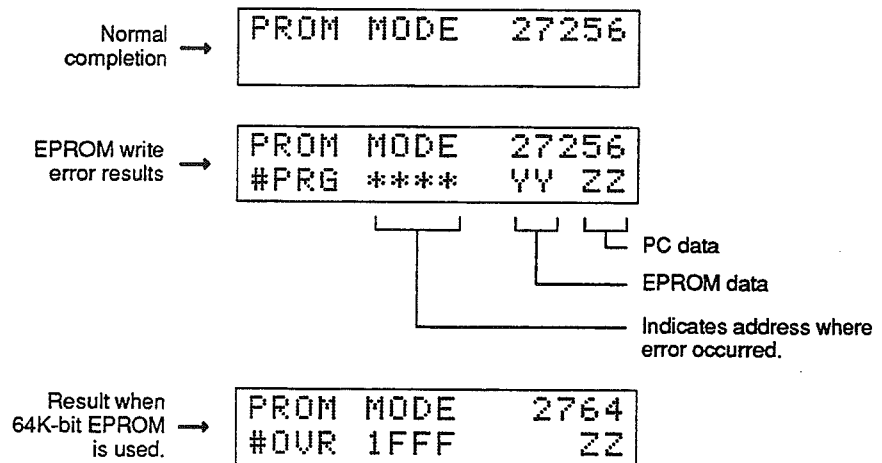
The EPROM Data Write operation writes PC data to the EPROM chip if it passes an erase check.

When the EPROM setting is set for 27256 EPROM, the 16K bytes of PC data will be transferred to the last 16K bytes of the chip's memory. (16K bytes of data will be transferred to addresses \$2000 through \$3FFF. Data will not be transferred to addresses \$0000 through \$1FFF.)

Example



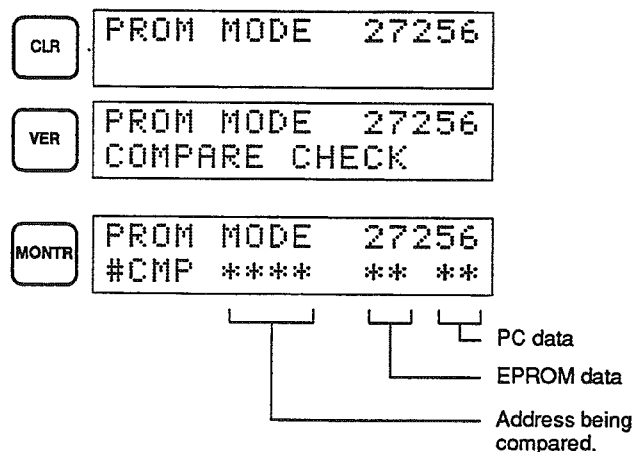
EPROM Write Results



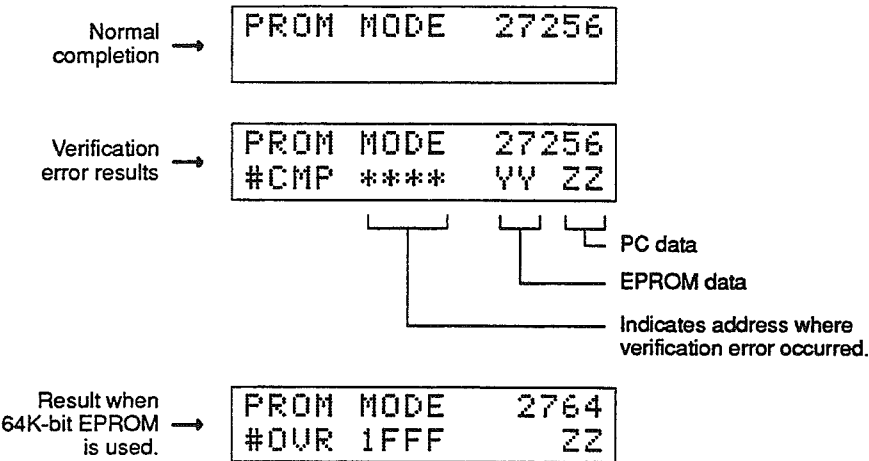
7-3-6 EPROM Data Compare

The EPROM Data Compare operation compares PC's User Memory data to the data in the EPROM chip.

Example



EPROM Compare Results



SECTION 8

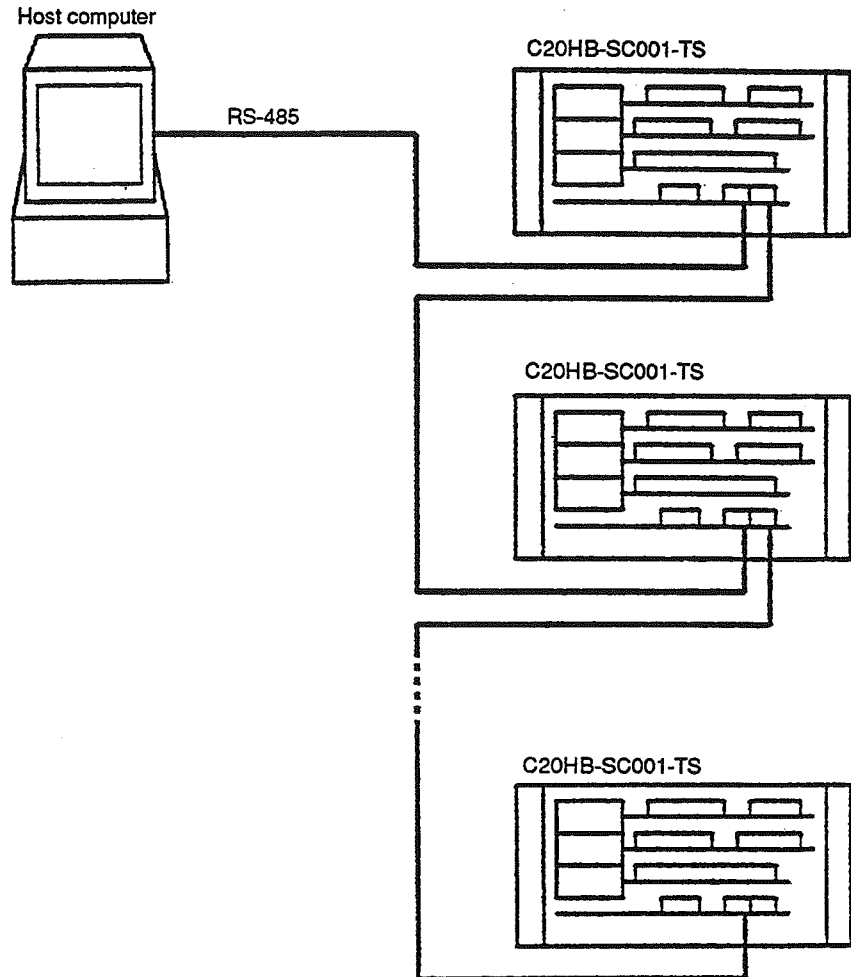
RS-485 Interface

The C20HB-TS has a built-in RS-485 interface which allows it to be connected to a Host Link System. This section describes modes, settings, and procedures used with the RS-485 interface. 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-485 System Configuration and Settings	152
8-2	Connection to a Host Computer's RS-232C Port	154
8-3	RS-485 Interface Flags and Control Bits	154
8-4	RS-485 Communications Protocol	155
8-4-1	Block Format	155
8-4-2	Block Format With More Than One Frame	155
8-4-3	Data Representation	157
8-4-4	FCS Calculation	158
8-4-5	FCS Calculation Program Example	158
8-5	Host Link Commands and Responses	159
8-5-1	TEST	159
8-5-2	STATUS READ	160
8-5-3	ERROR READ	160
8-5-4	IR AREA READ	161
8-5-5	HR AREA READ	162
8-5-6	AR AREA READ	162
8-5-7	LR AREA READ	162
8-5-8	TC STATUS READ	163
8-5-9	DM AREA READ	163
8-5-10	PV READ	163
8-5-11	SV READ 1	164
8-5-12	SV READ 2	164
8-5-13	STATUS WRITE	165
8-5-14	IR AREA WRITE	165
8-5-15	HR AREA WRITE	165
8-5-16	AR AREA WRITE	166
8-5-17	LR AREA WRITE	166
8-5-18	TC STATUS WRITE	166
8-5-19	DM AREA WRITE	167
8-5-20	PV WRITE	167
8-5-21	SV CHANGE 1	168
8-5-22	SV CHANGE 2	168
8-5-23	FORCED SET	169
8-5-24	FORCED RESET	169
8-5-25	MULTIPLE FORCED SET/RESET	170
8-5-26	MULTIPLE FORCED SET/RESET STATUS READ	170
8-5-27	FORCED SET/RESET CANCEL	171
8-5-28	PC MODEL READ	171
8-5-29	ABORT and INITIALIZE	172
8-5-30	Response to an Undefined Command	172
8-5-31	Response Indicating an Unprocessed Command	172
8-5-32	PROGRAM READ	172
8-5-33	PROGRAM WRITE	173
8-5-34	I/O REGISTER	173
8-5-35	I/O READ	174
8-5-36	Response Code List	175
8-6	Command Levels	176

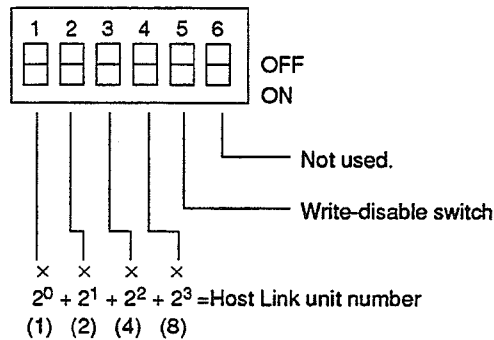
8-1 RS-485 System Configuration and Settings

You can use the RS-485 interface to connect a host computer to up to 16 C20HB-TS PCs, as shown in the following diagram. The total cable length can be up to 500 m.



Unit Number Settings

Each PC in a Host Link System must have a unique unit number, which is set on the DIP switch (SW2) located next to the PC's Peripheral Device port. The unit number is set in binary on pins 1 to 4.



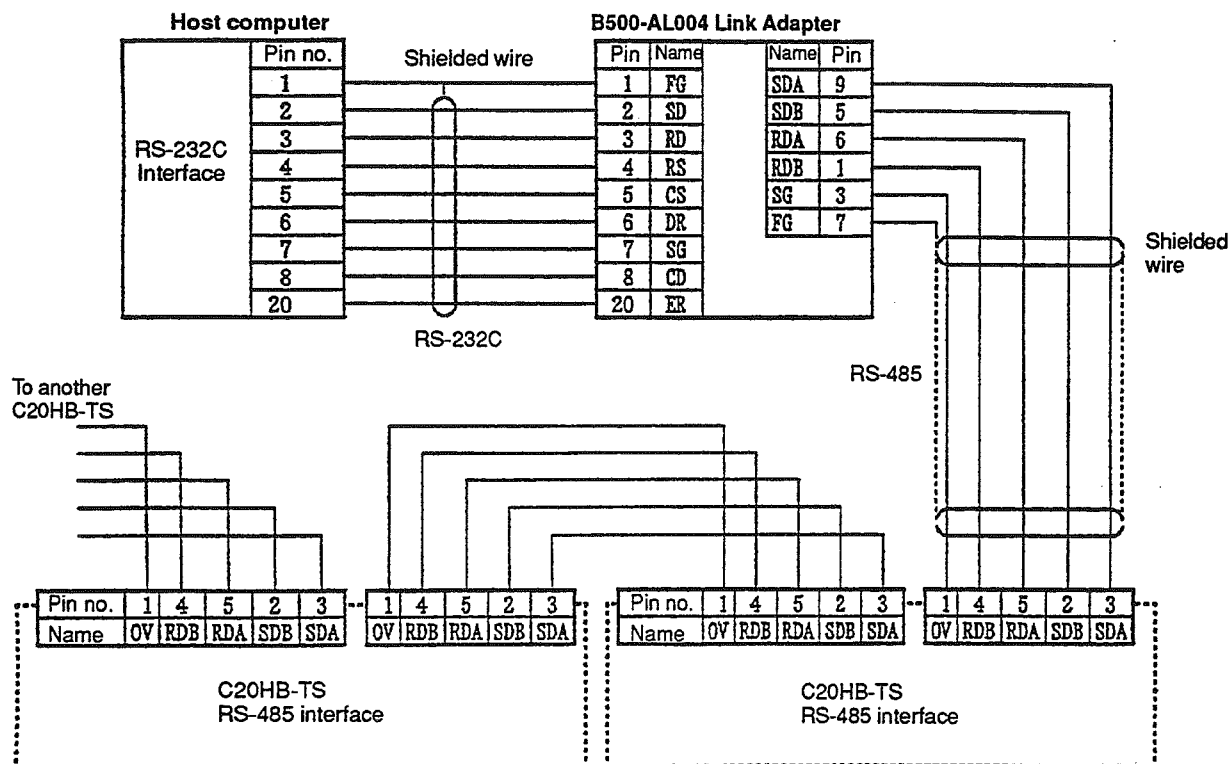
Pin	Value
1	0 or 1 (2^0)
2	0 or 2 (2^1)
3	0 or 4 (2^2)
4	0 or 8 (2^3)

Communications Settings

The RS-485 communications settings are set in DM 0920 to DM 0922. The default settings are 1 start bit, 7-bit data length, even parity, 2 stop bits, and 19,200 baud, but these settings can be changed. Refer to *3-5-2 Parameter and Parameter Backup Areas* for details.

8-2 Connection to a Host Computer's RS-232C Port

A B500-AL004 Link Adapter must be used when C20HB-TS PCs are connected to a host computer's RS-232C port, as shown in the following diagram. Up to 16 C20HB-TS PCs can be connected to a single host computer. The maximum allowable cable length is 500 m. Refer to the B500-AL004 Link Adapter's Operation Manual for details on the Link Adapter.



8-3 RS-485 Interface Flags and Control Bits

There are certain flags and control bits that indicate or control the status of the RS-485 interface.

RS-485 Communications Error Flag (SR 25208)

This flag is turned on to indicate that error has occurred in RS-485 communications. When a communications error occurs, a code indicating the type of error is output to AR 0400 to AR 0407.

RS-485 Communications Restart Bit (SR 25209)

This control bit is used to clear errors that have occurred in RS-485 communications. When the Restart Bit is turned ON and then OFF, the RS-485 Interface will be restarted and SR 25208, AR 0400 to AR 0407, and word AR 05 will be cleared.

RS-485 Communications Error Code (AR 04)

When an error has occurred in RS-485 communications, the RS-485 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:

- 01: Parity error
- 02: Framing error
- 03: Overrun error
- 04: FCS error

RS-485 Reception Counter

When a transmission is received on the RS-485 interface, the number of characters is counted in hexadecimal up to 255 (FF_{hex}) and then output to

AR 0500 to AR 0507 (RS-485 Reception Counter) to assist the user in debugging RS-485 interface communications. The counter automatically resets to 0 when the count exceeds 255. The counter can be reset manually by turning SR 25209 ON and then OFF.

RS-485 Transmission Counter

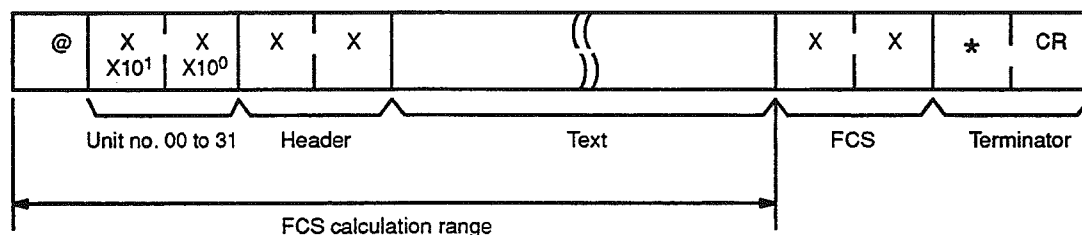
AR 0508 to AR 0515 (RS-485 Transmission Counter) operates the same as AR 0500 to AR 0507, except that it operates for transmissions sent from the PC through the RS-485 interface.

8-4 RS-485 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.

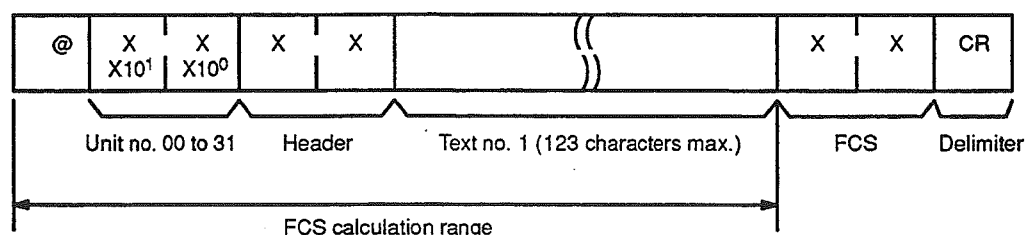
8-4-1 Block Format



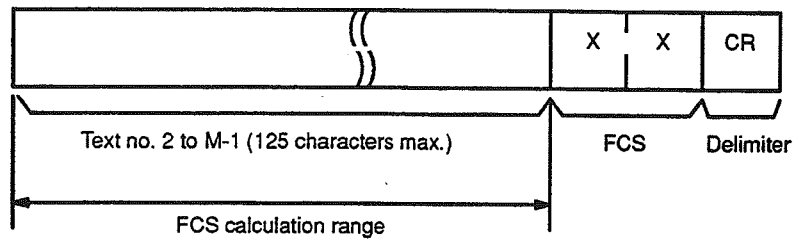
A block is usually made up of one unit called a frame, but long blocks of data (over 131 characters) must be divided into more than one frame before transmission. The first frame can have up to 131 characters, and subsequent frames can have up to 128 characters. The data must thus be divided into more 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

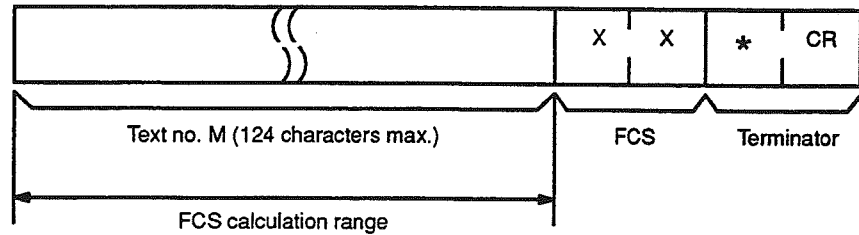
First Frame (131 Characters or Less)



Intermediate Frame(s) (128 Characters or Less)

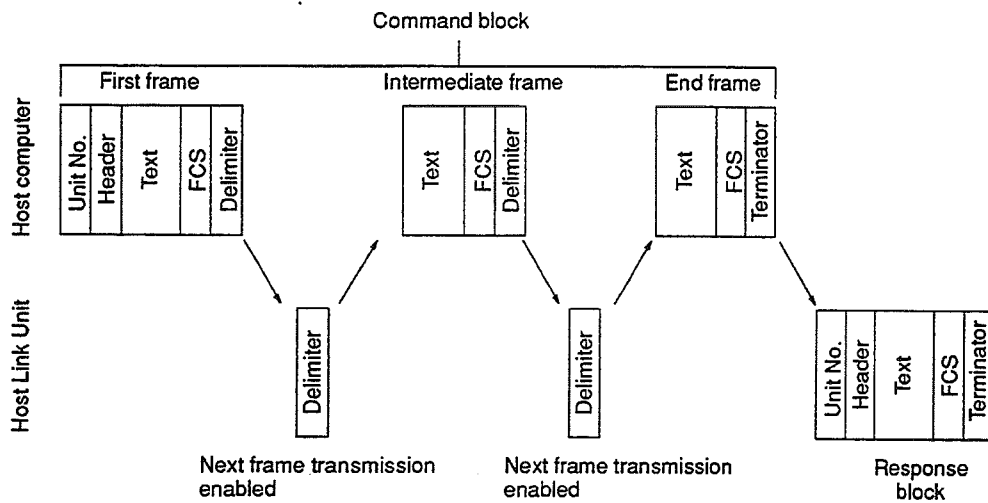


Last Frame (128 Characters or Less)



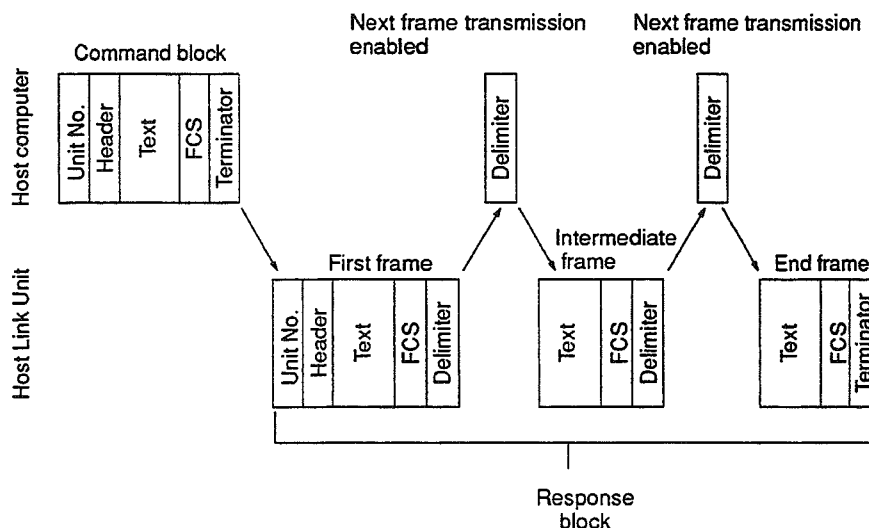
Sending Commands

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 X16³ to X16⁰ 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, X10³ to X10⁰ 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

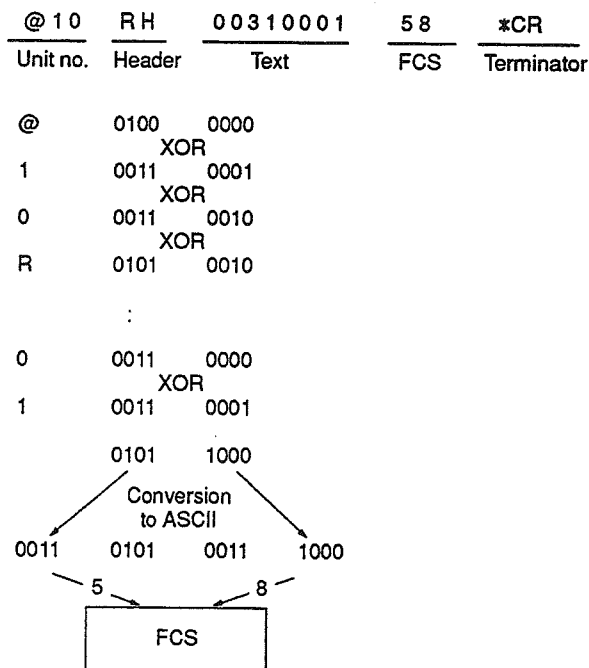
	ON/ OFF X2 ³	ON/ OFF X2 ²	ON/ OFF X2 ¹	ON/ OFF X2 ⁰	

In the above figure, the ON/OFF and X2³ to X2⁰ 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 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.

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.



```

400 *FSCHECK
405 L=LEN(RESPONSE$) ----- Transmit/receive data
410 Q=0:FCSCK$=" "
415 A$=RIGHT$(RESPONSE$,1)
417 PRINT RESPONSE$,A$,L
420 IF A$="*" THEN LENG$=LEN(RESPONSE$)-3 ELSE LENG$=LEN(RESPONSE$)-2
430 FCSP$=MID$(RESPONSE$,LENG$+1,2)
440 FOR I=1 TO LENG$ ----- Number of characters in FCS calculation range.
450 Q=ASC(MID$(RESPONSE$,I,1)) XOR Q Receive data contains an FCS, delimiter,
460 NEXT I terminator, etc. The ABORT command,
470 FCSD$=HEX$(Q) however, does not contain an FCS.
480 IF LEN(FCSD$)=1 THEN FCSD$="0"+FCSD$ ----- FCS calculation result
490 IF FCSD$<>FCSP$ THEN FCSCK$="ERR" ----- Receive FCS data
495 PRINT "FCSD$=";FCSD$,"FCSP$=";FCSP$,"FCSCK$=";-- A space follows the semicolon if the
500 RETURN FCS reception is performed normally. If
it is not performed, ERR is displayed.

```

Note: in this example, CR (CHRS(13)) is not included in RESPONSES.

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 in all communications, it cannot monitor host computer errors, it can only check for communication errors existing in the data it receives by checking the parity and frame check sequence of the data.

The response time will vary in accordance with the transfer speed, the amount of data, and the PC cycle time. The RS-485 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. X10 ¹	X10 ⁰	T	S	Any characters (122 max.) other than a carriage return	FCS	*	CR
---	------------------------------	------------------	---	---	--	-----	---	----

Response Format

@	Unit no. X10 ¹	X10 ⁰	T	S	Any characters (122 max.) other than a carriage return	FCS	*	CR
---	------------------------------	------------------	---	---	--	-----	---	----

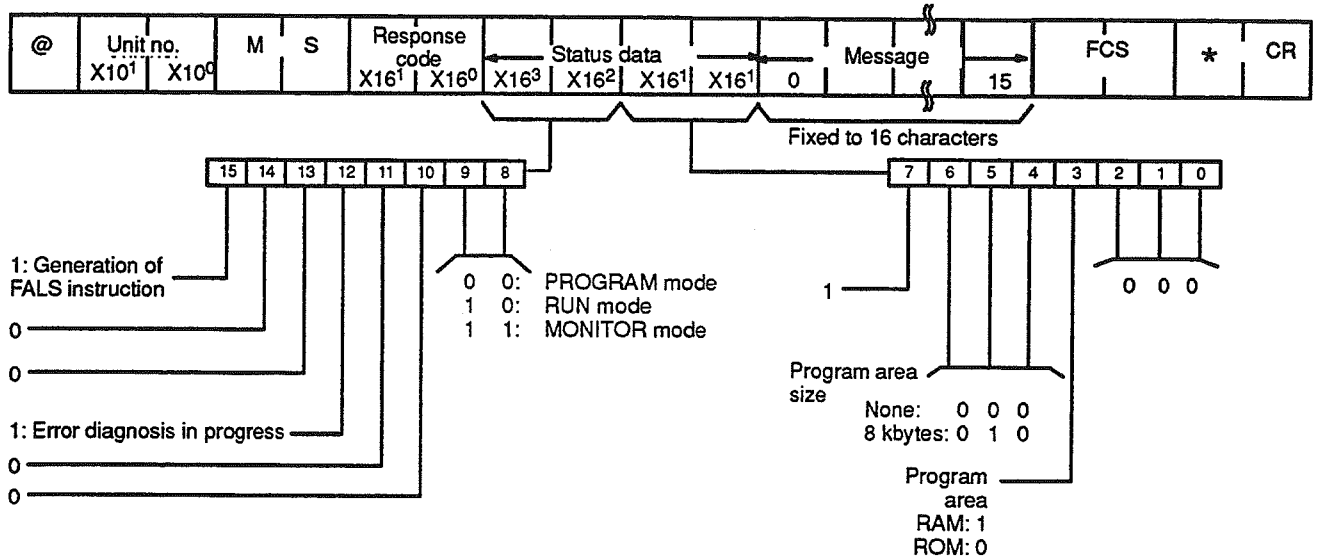
8-5-2 STATUS READ

This command causes the PC to read the operating status of the PC.

Command Format

@	Unit no. X10 ¹	X10 ⁰	M	S	FCS	*	CR
---	------------------------------	------------------	---	---	-----	---	----

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

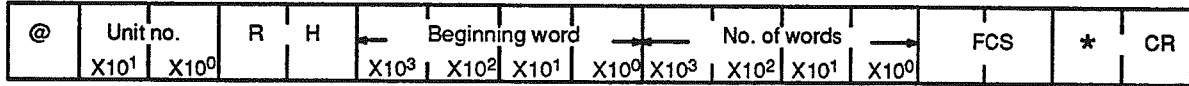
@	Unit no. X10 ¹	X10 ⁰	M	F	Error clear X10 ¹	X10 ⁰	FCS	*	CR
---	------------------------------	------------------	---	---	---------------------------------	------------------	-----	---	----

00: Don't clear
01: Clear

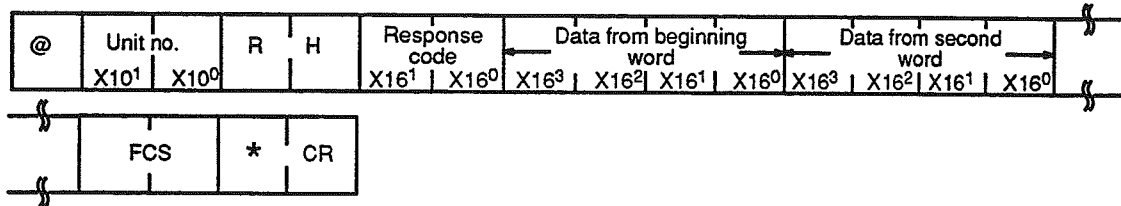
8-5-5 HR AREA READ

Reads the contents of the specified number of HR words, starting from the specified word.

Command Format



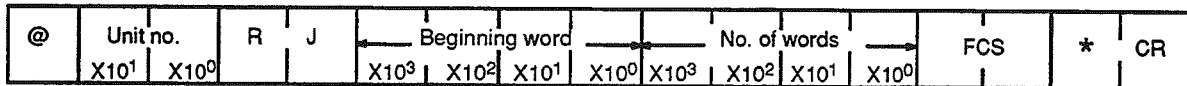
Response Format



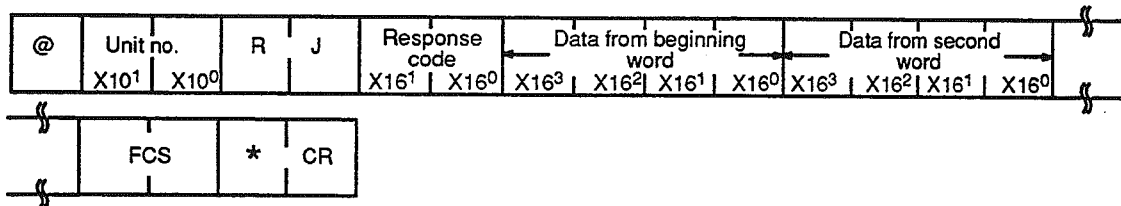
8-5-6 AR AREA READ

Reads the contents of the specified number of AR words, starting from the specified word.

Command Format



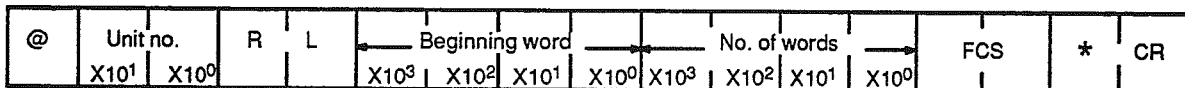
Response Format



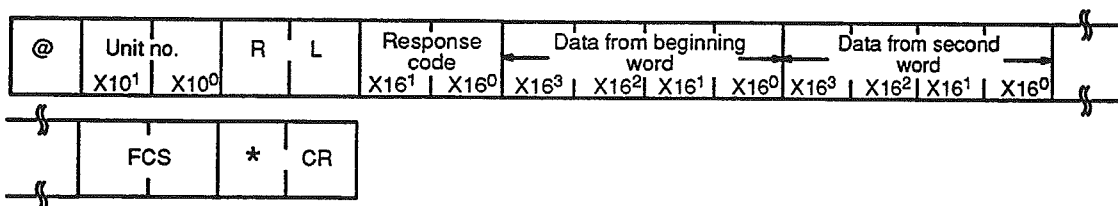
8-5-7 LR AREA READ

Reads the contents of the specified number of LR words, starting from the specified word.

Command Format



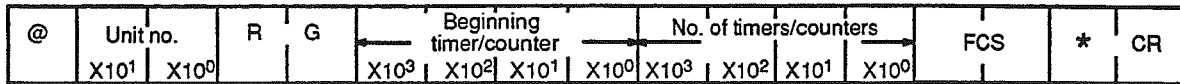
Response Format



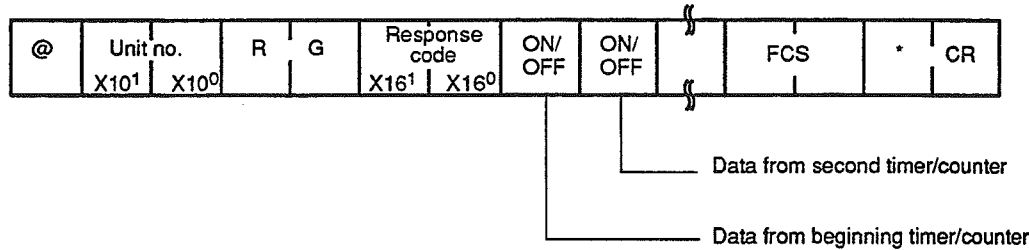
8-5-8 TC STATUS READ

Reads the status of the Completion Flags of the specified number of timers/counters, starting from the specified timer/counter.

Command Format



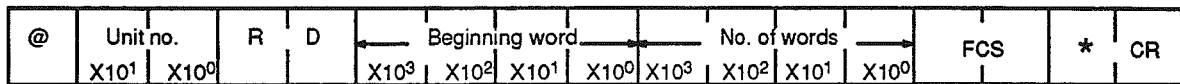
Response Format



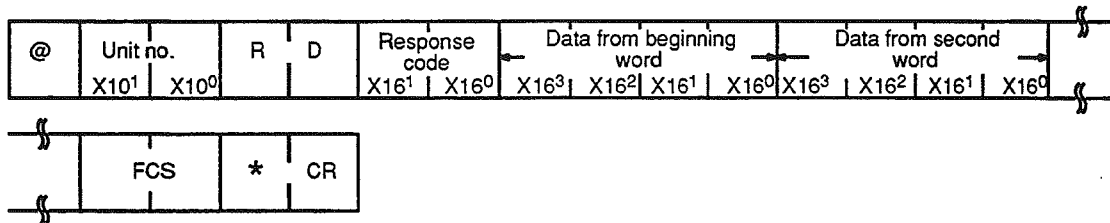
8-5-9 DM AREA READ

Reads the contents of the specified number of DM words, starting from the specified word.

Command Format



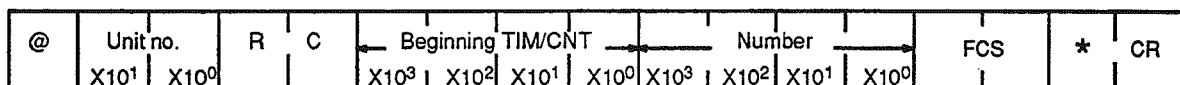
Response Format



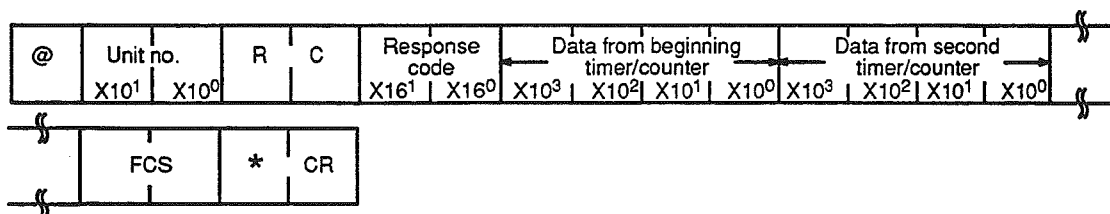
8-5-10 PV READ

Reads the specified number of timer/counter PVs (present values), starting from the specified timer/counter.

Command Format



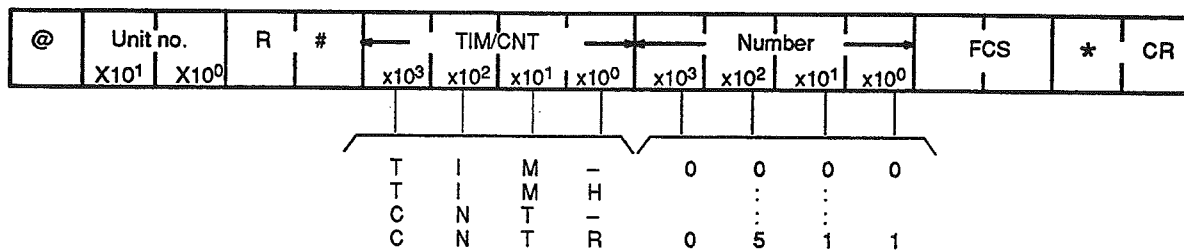
Response Format



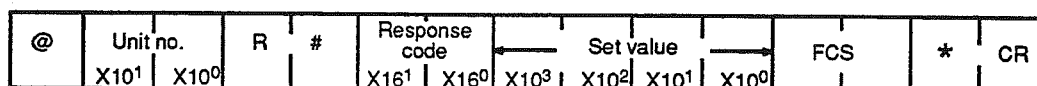
8-5-11 SV READ 1

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



Response Format

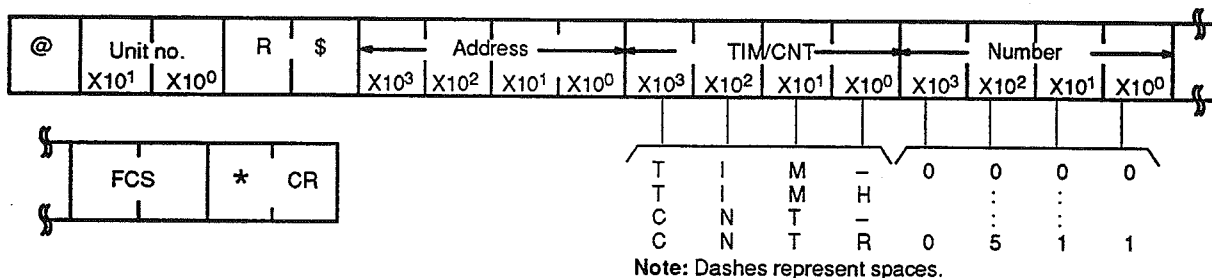


If the command is used more than once, the set value of only the first instruction will be read. If the second word (the operand) is not a constant, an error response (16) will be returned.

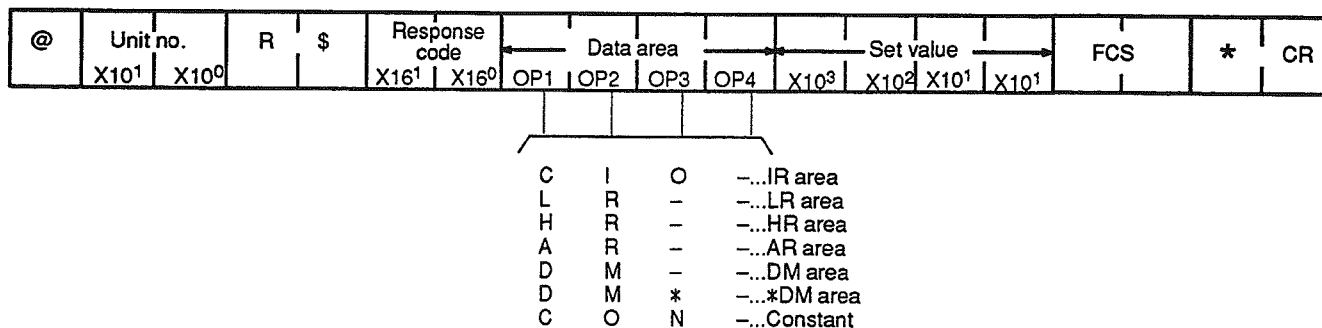
8-5-12 SV READ 2

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



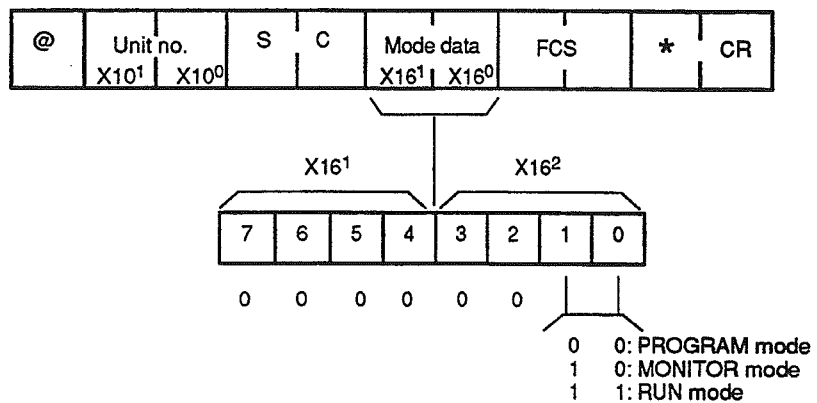
Response Format



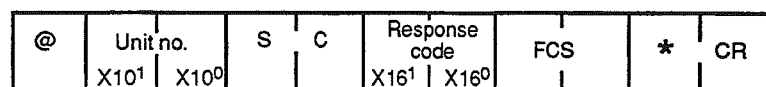
8-5-13 STATUS WRITE

Changes the operating mode of the PC according to the information input into digit x16⁰.

Command Format



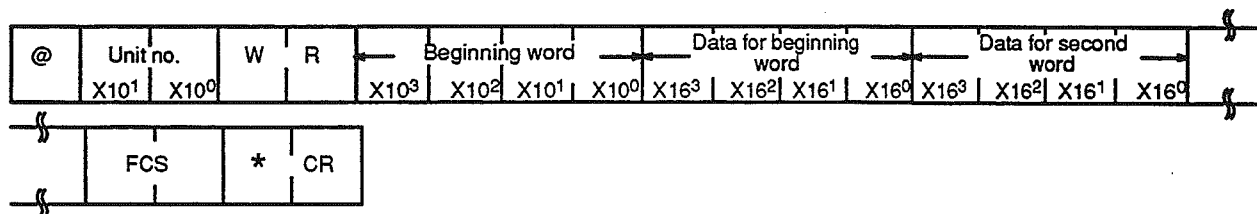
Response Format



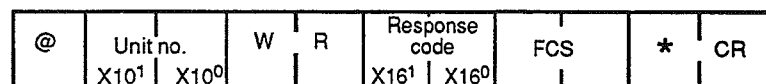
8-5-14 IR AREA WRITE

Writes data to the IR area, starting from the specified word. Writing is done word by word.

Command Format



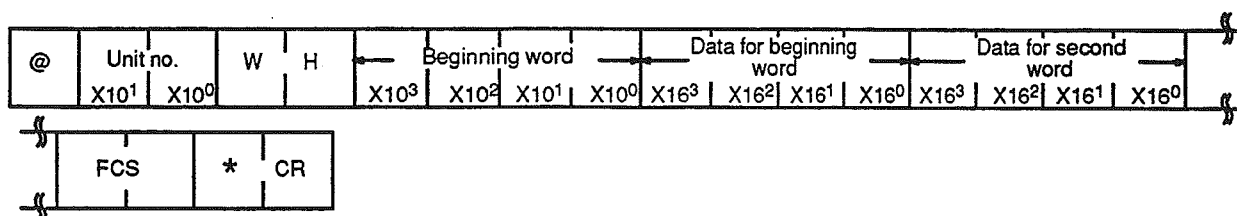
Response Format



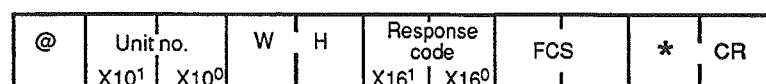
8-5-15 HR AREA WRITE

Writes data to the HR area, starting from the specified word. Writing is done word by word.

Command Format



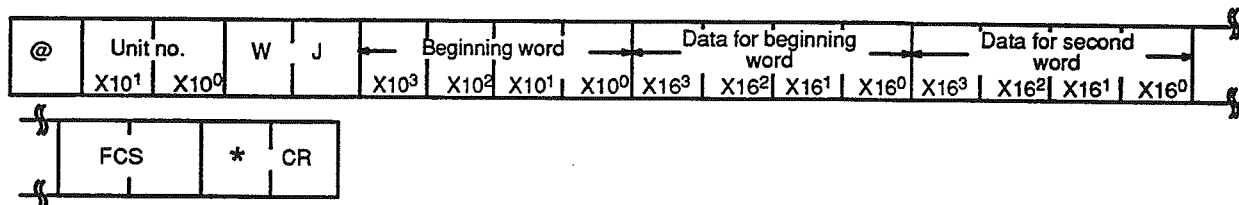
Response Format



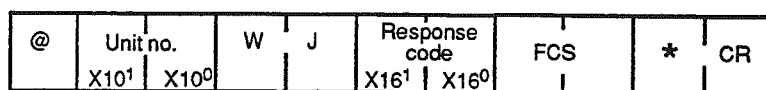
8-5-16 AR AREA WRITE

Writes data to the AR area, starting from the specified word. Writing is done word by word.

Command Format



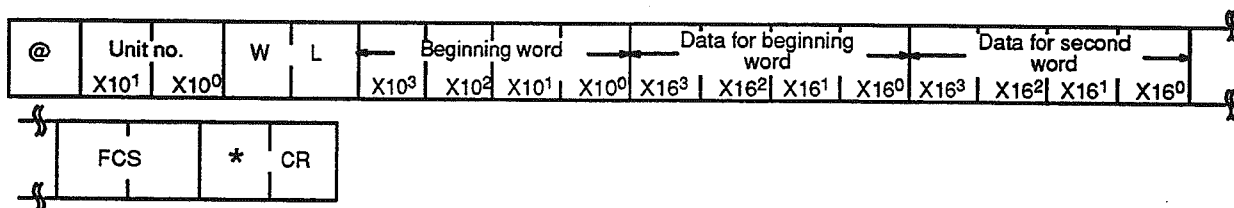
Response Format



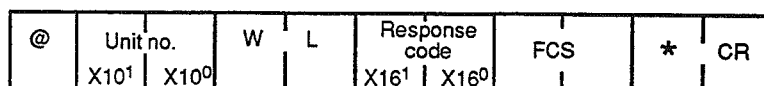
8-5-17 LR AREA WRITE

Writes data to the LR area, starting from the specified word. Writing is done word by word.

Command Format



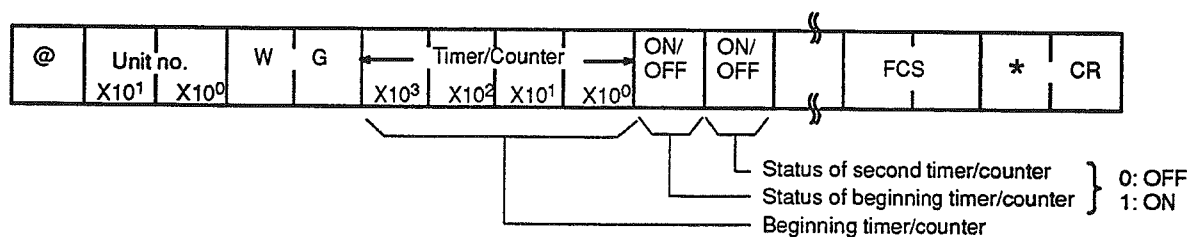
Response Format



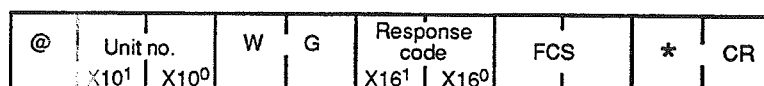
8-5-18 TC STATUS WRITE

Writes the status of Completion Flags to the TC area, starting from the specified timer/counter.

Command Format



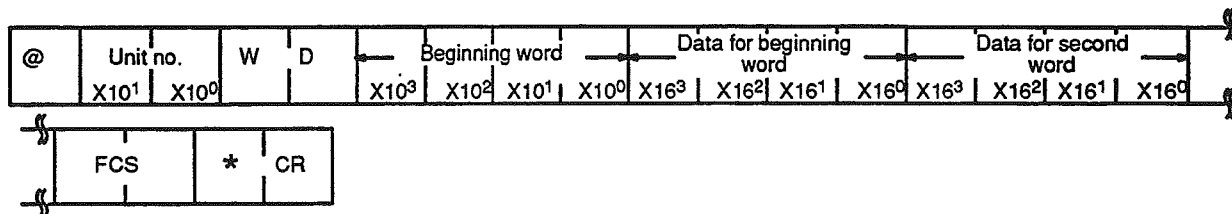
Response Format



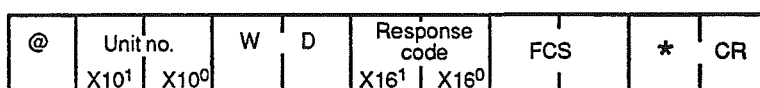
8-5-19 DM AREA WRITE

Writes data to the DM area, starting from the specified word. Writing is done word by word. If the write enable switch is set to OFF, the the writing range only extends up to DM 0999.

Command Format



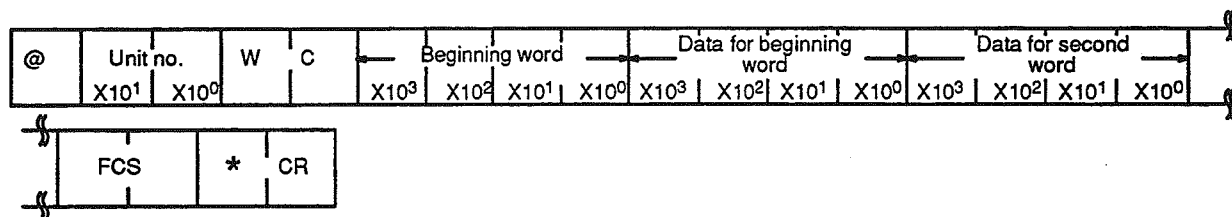
Response Format



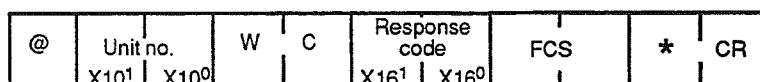
8-5-20 PV WRITE

Writes the PVs (present values) of timers/counters starting from the specified timer/counter.

Command Format



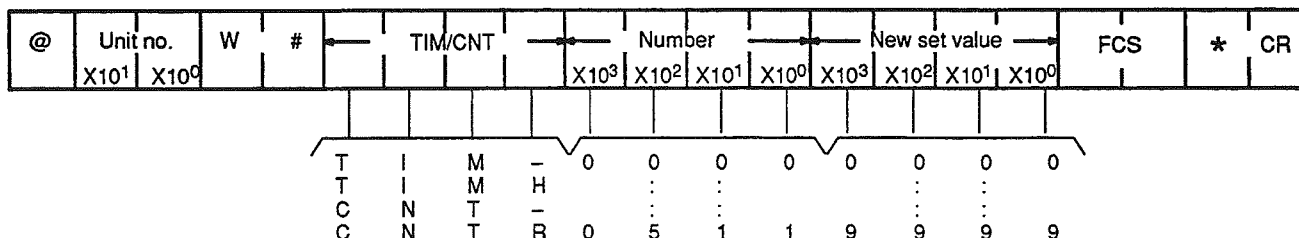
Response Format



8-5-21 SV CHANGE 1

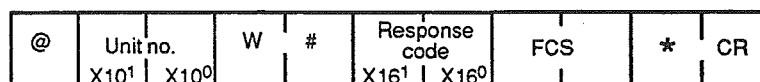
Changes the set value (constant only) of the specified timer/counter instruction. Reads from the beginning of the program and may therefore require up to about 20 seconds to produce a response. Refer also to SV CHANGE 2.

Command Format



Note: Dashes represent spaces.

Response Format



Changes the set value (a constant, or data area and word) of the specified timer/counter instruction. The instruction is specified by program address.

Diagram illustrating the 16-bit data format for the 16C160:

The top part shows the 16-bit structure:

- 1 bit: @
- 2 bits: Unit no. (X10¹, X10⁰)
- 1 bit: W
- 1 bit: \$
- 4 bits: Address (X10³, X10², X10¹, X10⁰)
- 4 bits: TIM/CNT
- 4 bits: Number (X10³, X10², X10¹, X10⁰)

Below the TIM/CNT field, a 4-bit code is shown:

T	I	M	-
C	N	T	H
C	N	T	R

Below the Number field, a 4-bit code is shown:

0	0	0	0
0	5	1	1

The bottom part shows the 16-bit structure:

- 1 bit: Data area
- 1 bit: New set value
- 4 bits: FCS (X10³, X10², X10¹, X10⁰)
- 1 bit: *
- 1 bit: CR

Below the Data area field, a 4-bit code is shown:

C	I	O	-
L	R	-	-
H	R	-	-
A	R	-	-
D	M	-	-
D	M	*	-
C	O	N	-

Note: Dashes represent spaces.

@	Unit no.		W	\$	Response code		FCS	*	CR
	X10 ¹	X10 ⁰			X16 ¹	X16 ⁰			

Forced sets a bit in an IR, LR, HR, AR, or TC area. Bits need to be force set one at a time.

Diagram of the data format of the transmission frame. The frame structure is as follows:

@	Unit no. X10 ¹ X10 ⁰		K	S	Data area				Word X10 ³ X10 ² X10 ¹			Bit X10 ¹ X10 ⁰		FCS	*	CR
---	---	--	---	---	-----------	--	--	--	--	--	--	--	--	-----	---	----

Below the Data area, the following fields are defined:

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	B	

@	Unit no.		K	S	Response code		FCS	*	CR
	X10 ¹	X10 ⁰			X16 ¹	X16 ⁰			

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

@	Unit no.		K	R	Data area				Word				Bit		FCS	*	CR
	X10 ¹	X10 ⁰							X10 ³	X10 ²	X10 ¹	X10 ⁰	X10 ¹	X10 ⁰			

C	I	O	-	IR area
L	R	-	-	LR area
H	R	-	-	HR area
A	R	-	-	AR area
T	I	M	-		
T	I	M	H		
C	N	T	-		
C	N	T	R		

TC area

Note: Dashes represent spaces.

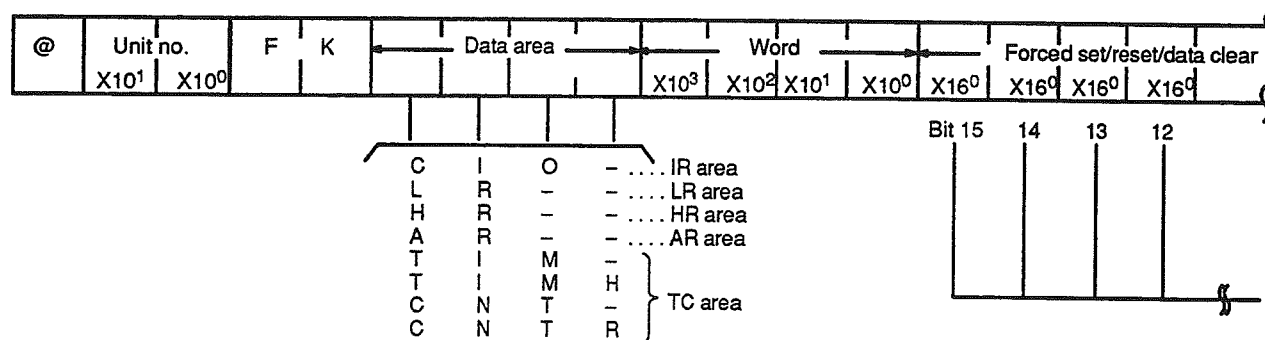
Response Format

@	Unit no.		K	R	Response code		FCS	*	CR
	X10 ¹	X10 ⁰			X16 ¹	X16 ⁰			

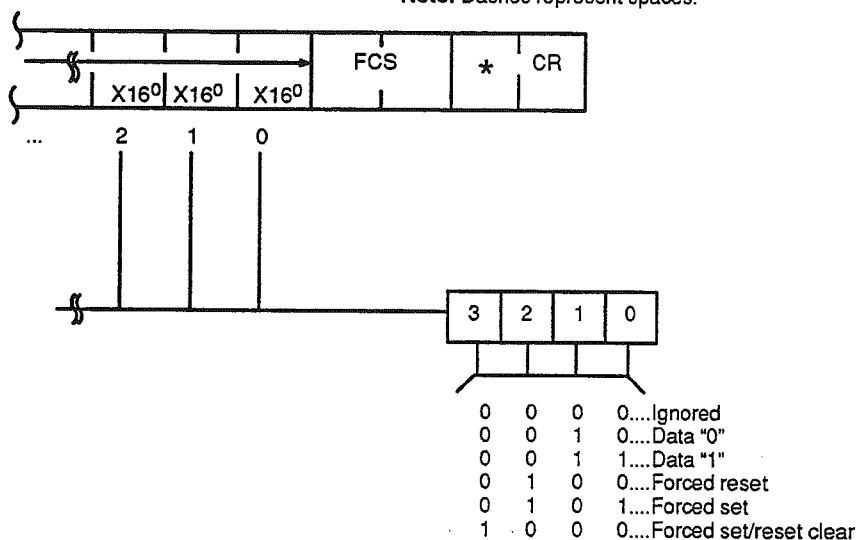
8-5-25 MULTIPLE FORCED SET/RESET

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.

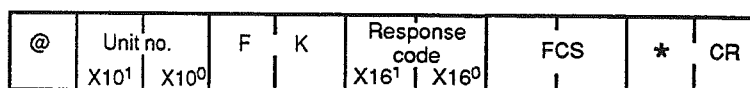
Command Format



Note: Dashes represent spaces.



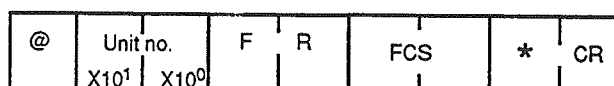
Response Format



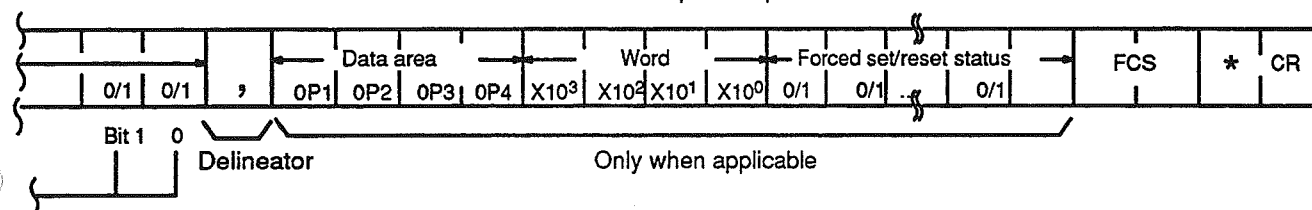
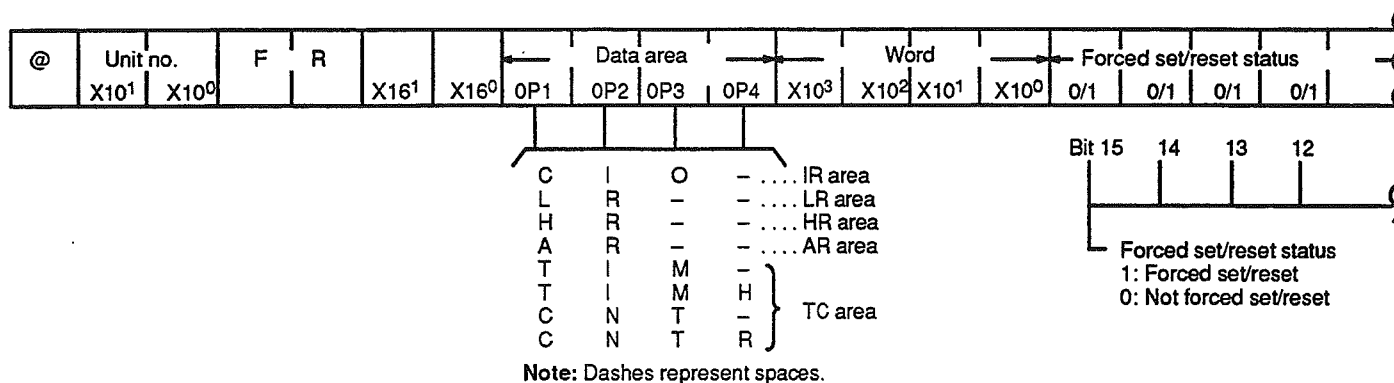
8-5-26 MULTIPLE FORCED SET/RESET STATUS READ

Reads the forced set or forced reset status of the PC.

Command Format



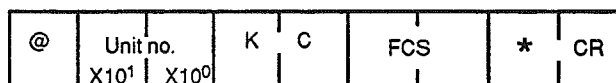
Response Format



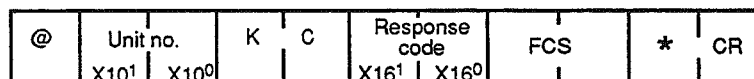
8-5-27 FORCED SET/RESET CANCEL

Cancels all forced set and forced reset bits (including those achieved via MULTIPLE FORCED SET/RESET.

Command Format



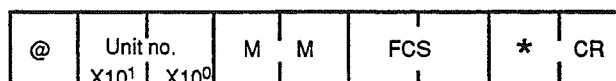
Response Format



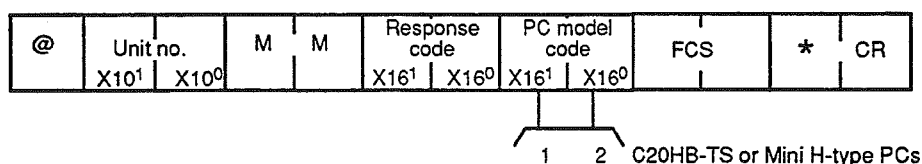
8-5-28 PC MODEL READ

Reads the model type of the PC.

Command Format



Response Format

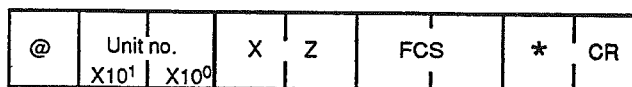


8-5-29 ABORT and INITIALIZE

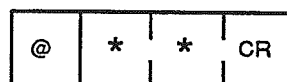
The ABORT command is used to abort the process being performed by the Host Link function and to then enable reception of the next command. The INITIALIZE command initializes the transmission control procedure of all the PCs connected to the host computer. Neither command receives a response.

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



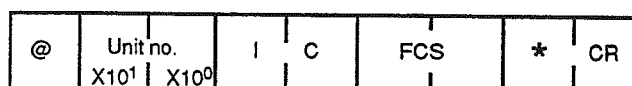
INITIALIZE Command Format



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.

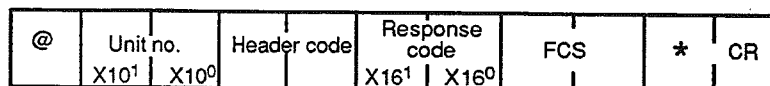
Response Format



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

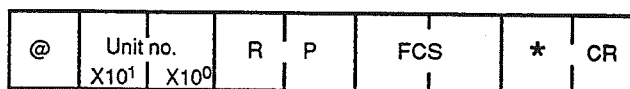


The header code varies according to the command which was sent. The headers of some commands include subheader codes (e.g., I/O REGISTER, I/O READ, and DM SIZE CHANGE).

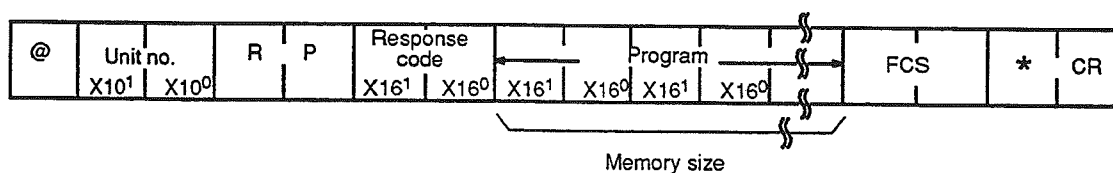
8-5-32 PROGRAM READ

Transmits the contents of the PC program memory.

Command Format



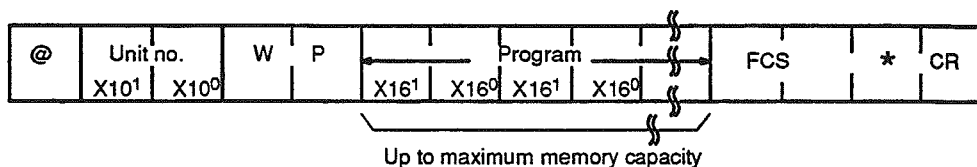
Response Format



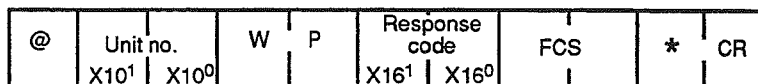
8-5-33 PROGRAM WRITE

Writes the received program into the PC program memory.

Command Format



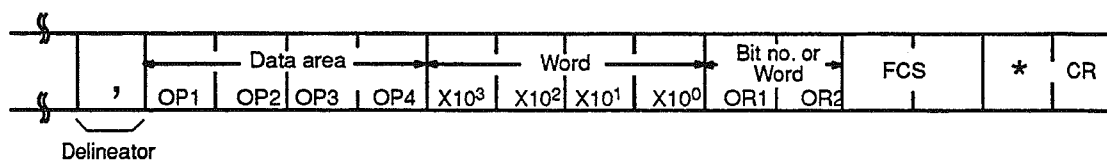
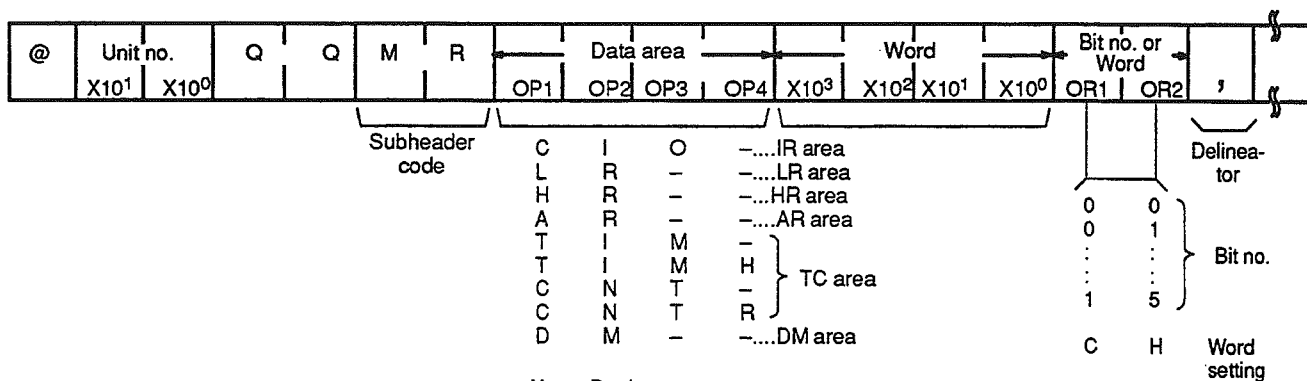
Response Format



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



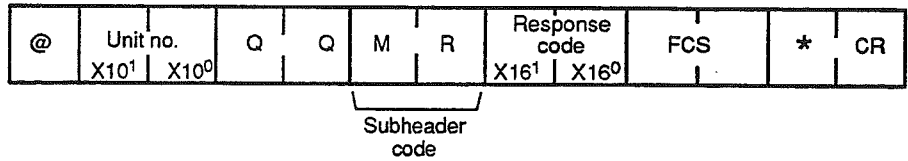
Setting Table

	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.

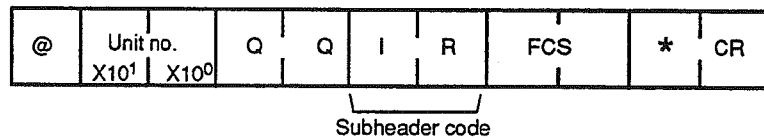
Response Format



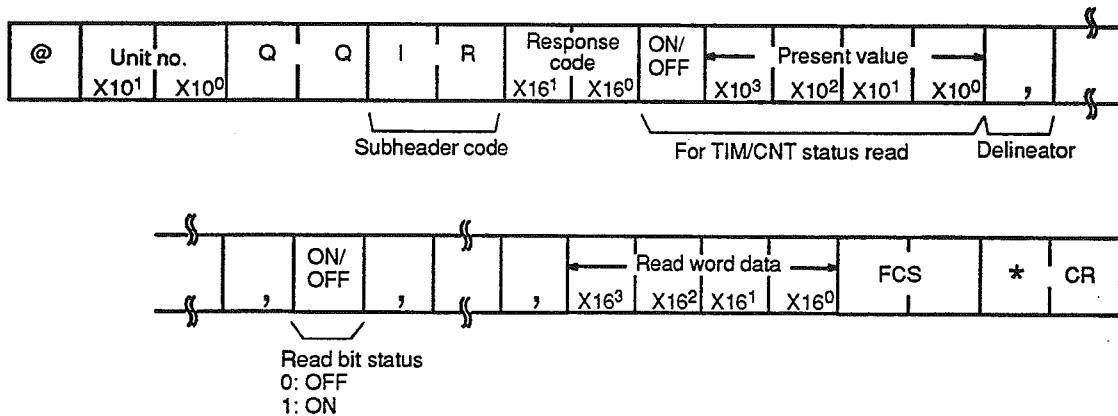
8-5-35 I/O READ

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	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	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.

8-6 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 C20HB-TS provides self-diagnostic functions to identify many types of abnormal system conditions. These functions minimize downtime and enable quick, smooth error correction.

This section provides information on hardware and software errors that occur during PC operation. 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 *3-3 SR Area* and *3-4 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	178
9-2	Programmed Alarms and Error Messages	178
9-3	Reading and Clearing Errors and Messages	178
9-4	Error Messages	179
9-5	Error History Function	181
9-6	Host Link Error Processing	181
9-6-1	Error Control	181
9-6-2	Invalid Processing	182
9-6-3	Process Interruption	182
9-6-4	Time Monitoring	182
9-6-5	Retries	182

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 table on page 180). 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) and FALS(07) can be used in the program to provide user-programmed information on error conditions. With these instructions, the user can tailor error diagnosis to aid in troubleshooting.

FAL(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.

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 LSS and FIT are provided in the relevant operation manuals.

9-4 Error Messages

There are basically two types of errors for which messages are displayed: 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.

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 <div>SYS FAIL FAL</div>	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 <div>SCAN TIME OVER</div>	F8	Watchdog timer has exceeded 100 ms.	Program cycle time is longer than recommended. Reduce cycle time if possible.
Host Link Error <div>No message</div>	None	<ul style="list-style-type: none"> Error exists between host computer and built-in Host Link Interface. 	<ul style="list-style-type: none"> 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.

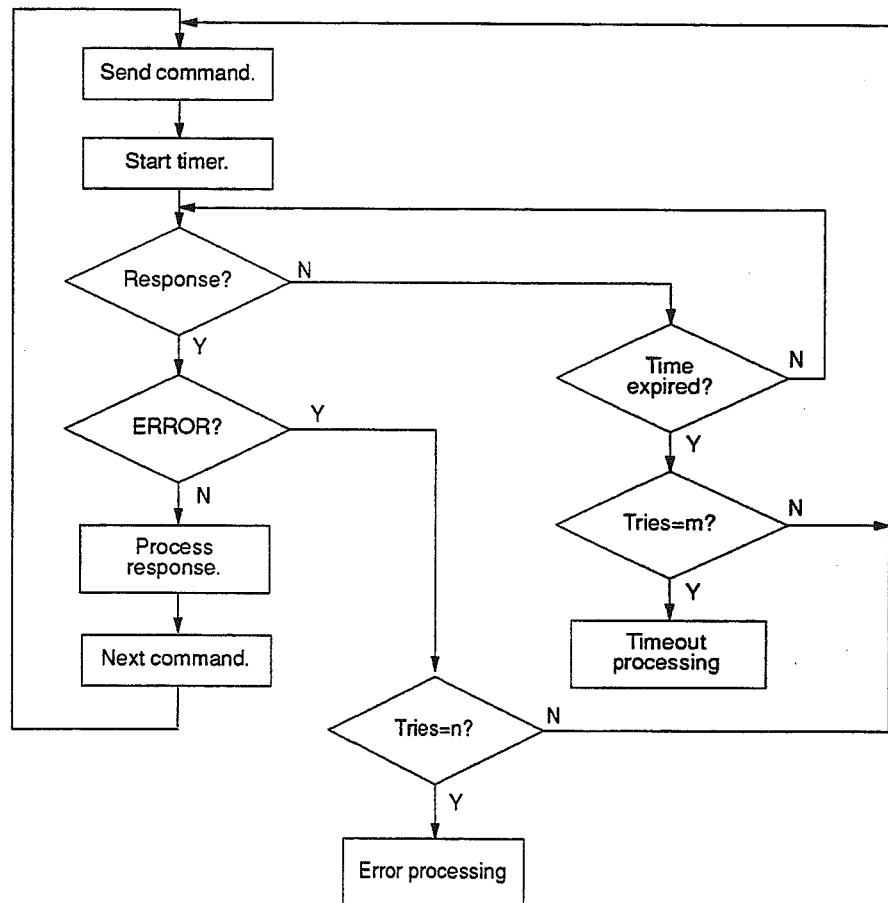
Error and message	FAL no.	Probable cause	Possible correction
Power interruption <div>No message</div>	None	Power has been interrupted for at least 10 ms.	Check power supply voltage and power lines. Try to power-up again.
CPU error <div>No message</div>	None	Watchdog timer has exceeded maximum setting (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 <div>MEMORY ERR</div>	F1	Checksum error has occurred or incorrect instruction exists.	Perform a Program Check Operation to locate cause of error. If error not correctable, try inputting program again.
No END(01) instruction <div>NO END INST</div>	F0	END(01) is not written anywhere in program.	Write END(01) at the last address of the program.
I/O bus error <div>I/O BUS ERR</div> <div>Rack no.</div>	C0 to C2	Error has occurred in the bus line between the Units.	Check cable connections between Units.
FALS error <div>SYS FAIL FAL**</div>	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.	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 the FAL no. is sent to the System DM when an error occurs.

9-6 Host Link Error Processing

This section describes errors that can occur in a computer-linked system employing the RS-485 interface, including errors processed by the host computer (see 9-6-1 Error Control and 8-4 Host Link Communications Protocol).



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

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

Appendix B

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

Page:

[illegible]

Timers and Counters

Page:

[illegible]

Appendix C

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 through a Programming Console or other Peripheral Device.

Program Coding Sheet

Programmer:

Program:

Date:

Page:

[illegible]

Appendix D

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 E

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. Set between 00 and 99 in BCD for 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 (00 to 99).	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): Host Link Service Time		5%: 0000	
00 to 07	Percent of cycle time allocated to Host Link servicing (00 to 99).	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	0000	
DM 0905 to DM 0919 (DM 1905 to DM 1919) Not used.		0000	0000
DM 0920 (DM 1920)		0000	
00 to 07	Host Link Communications Format Selection 00: Standard (1 start bit, 7-bit data, even parity, 2 stop bits, 19,200 bps) 01: Custom settings (i.e., according to contents of DM 0921)	Standard: 00	
08 to 15	Not used.	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 06: 19,200 bps	19,200 bps: 06	
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	Host Link 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	Not used.	00	
DM 0923 to DM 0929 (DM 1923 to DM 1929) Not used.		0000	0000

Revision History

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	November 1994	Original production based on the Mini H-type PCs Operation Manual (W176-E1-4) and the Mini H-type PCs Installation Guide (W175-E1-4).

OMRON CORPORATION

FA Systems Division H.Q.

66 Matsumoto

Mishima-city, Shizuoka 411-8511

Japan

Tel: (81)55-977-9181/Fax: (81)55-977-9045

Regional Headquarters

OMRON EUROPE B.V.

Wegalaan 67-69, NL-2132 JD Hoofddorp

The Netherlands

Tel: (31)2356-81-300/Fax: (31)2356-81-388

OMRON ELECTRONICS LLC

1 East Commerce Drive, Schaumburg, IL 60173

U.S.A.

Tel: (1)847-843-7900/Fax: (1)847-843-8568

OMRON ASIA PACIFIC PTE. LTD.

83 Clemenceau Avenue,

#11-01, UE Square,

Singapore 239920

Tel: (65)6835-3011/Fax: (65)6835-2711

OMRON

Authorized Distributor:

Note: Specifications subject to change without notice.

Printed in Japan
0405-0.01M

This manual is printed on 100% recycled paper.