Cat. No. W140-E1-4

SYSMAC
C1000H/C2000H
Programmable Controllers

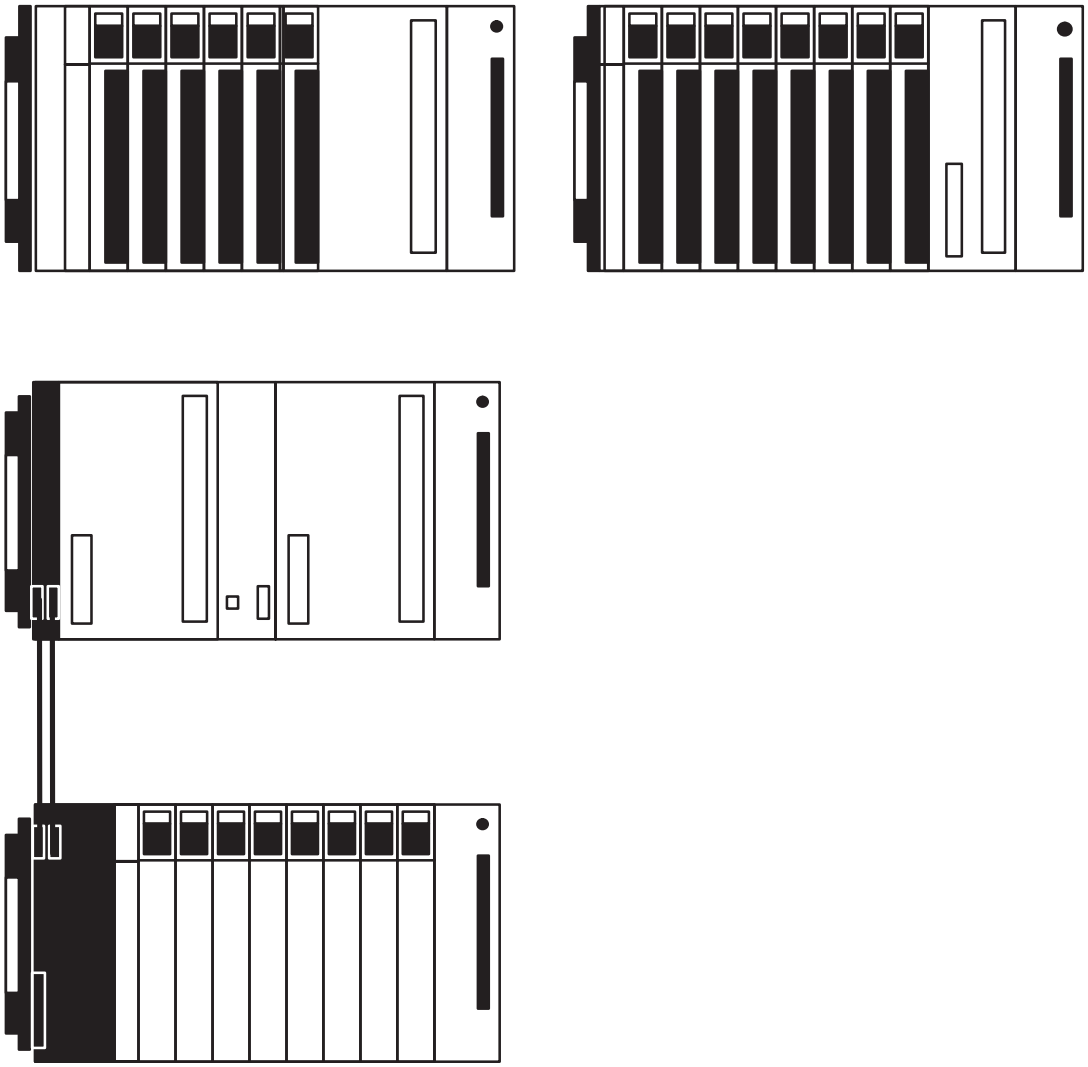
OPERATION MANUAL

OMRON®

C1000H/C2000H
Programmable Controllers

Operation Manual

Revised May 2003



Certain Terms and Conditions of Sale

1. Offer; Acceptance. These terms and conditions (these "Terms") are deemed part of all catalogs, manuals or other documents, whether electronic or in writing, relating to the sale of goods or services (collectively, the "Goods") by Omron Electronics LLC and its subsidiary companies ("Seller"). Seller hereby objects to any terms or conditions proposed in Buyer's purchase order or other documents which are inconsistent with, or in addition to, these Terms. Please contact your Omron representative to confirm any additional terms for sales from your Omron company.
2. Prices. All prices stated are current, subject to change without notice by Seller. Buyer agrees to pay the price in effect at time of shipment.
3. Discounts. Cash discounts, if any, will apply only on the net amount of invoices sent to Buyer after deducting transportation charges, taxes and duties, and will be allowed only if (i) the invoice is paid according to Seller's payment terms and (ii) Buyer has no past due amounts owing to Seller.
4. Orders. Seller will accept no order less than $200 net billing.
5. Governmental Approvals. Buyer shall be responsible for, and shall bear all costs involved in, obtaining any government approvals required for the importation or sale of the Goods.
6. Taxes. All taxes, duties and other governmental charges (other than general real property and income taxes), including any interest or penalties thereon, imposed directly or indirectly on Seller or required to be collected directly or indirectly by Seller for the manufacture, production, sale, delivery, importation, consumption or use of the Goods sold hereunder (including customs duties and sales, excise, use, turnover and license taxes) shall be charged to and remitted by Buyer to Seller.
7. Financial. If the financial position of Buyer at any time becomes unsatisfactory to Seller, Seller reserves the right to stop shipments or require satisfactory security or payment in advance. If Buyer fails to make payment or otherwise comply with these Terms or any related agreement, Seller may (without liability and in addition to other remedies) cancel any unshipped portion of Goods sold hereunder and stop any Goods in transit until Buyer pays all amounts, including amounts payable hereunder, whether or not then due, which are owing to it by Buyer. Buyer shall in any event remain liable for all unpaid accounts.
8. Cancellation; Etc. Orders are not subject to rescheduling or cancellation unless Buyer indemnifies Seller fully against all costs or expenses arising in connection therewith.
9. Force Majeure. Seller shall not be liable for any delay or failure in delivery resulting from causes beyond its control, including earthquakes, fires, floods, strikes or other labor disputes, shortage of labor or materials, accidents to machinery, acts of sabotage, riots, delay in or lack of transportation or the requirements of any government authority.
10. Shipping; Delivery. Unless otherwise expressly agreed in writing by Seller:
    a. Shipments shall be by a carrier selected by Seller;
    b. Such carrier shall act as the agent of Buyer and delivery to such carrier shall constitute delivery to Buyer;
    c. All sales and shipments of Goods shall be FOB shipping point (unless otherwise stated in writing by Seller), at which point title to and all risk of loss of the Goods shall pass from Seller to Buyer, provided that Seller shall retain a security interest in the Goods until the full purchase price is paid by Buyer;
    d. Delivery and shipping dates are estimates only.
    e. Seller will package Goods as it deems proper for protection against normal handling and extra charges apply to special conditions.
11. Claims. Any claim by Buyer against Seller for shortage or damage to the Goods occurring before delivery to the carrier must be presented in writing to Seller within 30 days of receipt of shipment and include the original transportation bill signed by the carrier noting that the carrier received the Goods from Seller in the condition claimed.
12. Warranties. (a) Exclusive Warranty. Seller's exclusive warranty is that the Goods will be free from defects in materials and workmanship for a period of twelve months from the date of sale by Seller (or such other period expressed in writing by Seller). Seller disclaims all other warranties, express or implied. (b) Limitations. SELLER MAKES NO WARRANTY OR REPRESENTATION, EXPRESS OR IMPLIED, ABOUT NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE OF THE GOODS. BUYER ACKNOWLEDGES THAT IT ALONE HAS DETERMINED THAT THE GOODS WILL SUITABLY MEET THE REQUIREMENTS OF THEIR INTENDED USE. Seller further disclaims all warranties and responsibility of any type for claims or expenses based on infringement by the Goods or otherwise of any intellectual property right. (c) Buyer Remedy. Seller's sole obligation hereunder shall be to replace (in the form originally shipped with Buyer responsible for labor charges for removal or replacement thereof) the non-complying Good or, at Seller's election, to repay or credit Buyer an amount equal to the purchase price of the Good; provided that in no event shall Seller be responsible for warranty, repair, indemnity or any other claims or expenses regarding the Goods unless Seller's analysis confirms that the Goods were properly handled, stored, installed and maintained and not subject to contamination, abuse, misuse or inappropriate modification. Return of any goods by Buyer must be approved in writing by Seller before shipment. Seller shall not be liable for the suitability or unsuitability or the results from the use of Goods in combination with any electrical or electronic components, circuits, system assemblies or any other materials or substances or environments. Any advice, recommendations or information given orally or in writing, are not to be construed as an amendment or addition to the above warranty.
13. Damage Limits; Etc. SELLER SHALL NOT BE LIABLE FOR SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, LOSS OF PROFITS OR PRODUCTION OR COMMERCIAL LOSS IN ANY WAY CONNECTED WITH THE GOODS, WHETHER SUCH CLAIM IS BASED IN CONTRACT, WARRANTY, NEGLIGENCE OR STRICT LIABILITY. Further, in no event shall liability of Seller exceed the individual price of the Good on which liability is asserted.
14. Indemnities. Buyer shall indemnify and hold harmless Seller, its affiliates and its employees from and against all liabilities, losses, claims, costs and expenses (including attorney's fees and expenses) related to any claim, investigation, litigation or proceeding (whether or not Seller is a party) which arises or is alleged to arise from Buyer's acts or omissions under these Terms or in any way with respect to the Goods. Without limiting the foregoing, Buyer (at its own expense) shall indemnify and hold harmless Seller and defend or settle any action brought against Seller to the extent that it is based on a claim that any Good made to Buyer specifications infringed intellectual property rights of another party.
15. Property; Confidentiality. The intellectual property embodied in the Goods is the exclusive property of Seller and its affiliates and Buyer shall not attempt to duplicate it in any way without the written permission of Seller. Notwithstanding any charges to Buyer for engineering or tooling, all engineering and tooling shall remain the exclusive property of Seller. All information and materials supplied by Seller to Buyer relating to the Goods are confidential and proprietary, and Buyer shall limit distribution thereof to its trusted employees and strictly prevent disclosure to any third party.
16. Miscellaneous. (a) Waiver. No failure or delay by Seller in exercising any right and no course of dealing between Buyer and Seller shall operate as a waiver of rights by Seller. (b) Assignment. Buyer may not assign its rights hereunder without Seller's written consent. (c) Amendment. These Terms constitute the entire agreement between Buyer and Seller relating to the Goods, and no provision may be changed or waived unless in writing signed by the parties. (d) Severability. If any provision hereof is rendered ineffective or invalid, such provision shall not invalidate any other provision. (e) Setoff. Buyer shall have no right to set off any amounts against the amount owing in respect of this invoice. (f) As used herein, "including" means "including without limitation".

Certain Precautions on Specifications and Use

1. Suitability of Use. Seller shall not be responsible for conformity with any standards, codes or regulations which apply to the combination of the Good in the Buyer's application or use of the Good. At Buyer's request, Seller will provide applicable third party certification documents identifying ratings and limitations of use which apply to the Good. This information by itself is not sufficient for a complete determination of the suitability of the Good in combination with the end product, machine, system, or other application or use. The following are some examples of applications for which particular attention must be given. This is not intended to be an exhaustive list of all possible uses of this Good, nor is it intended to imply that the uses listed may be suitable for this Good:
    (i) Outdoor use, uses involving potential chemical contamination or electrical interference, or conditions or uses not described in this document.
    (ii) Energy control systems, combustion systems, railroad systems, aviation systems, medical equipment, amusement machines, vehicles, safety equipment, and installations subject to separate industry or government regulations.
    (iii) Systems, machines and equipment that could present a risk to life or property. Please know and observe all prohibitions of use applicable to this Good.

    NEVER USE THE PRODUCT FOR AN APPLICATION INVOLVING SERIOUS RISK TO LIFE OR PROPERTY WITHOUT ENSURING THAT THE SYSTEM AS A WHOLE HAS BEEN DESIGNED TO ADDRESS THE RISKS, AND THAT THE SELLER'S PRODUCT IS PROPERLY RATED AND INSTALLED FOR THE INTENDED USE WITHIN THE OVERALL EQUIPMENT OR SYSTEM.
2. Programmable Products. Seller shall not be responsible for the user's programming of a programmable Good, or any consequence thereof.
3. Performance Data. Performance data given in this catalog is provided as a guide for the user in determining suitability and does not constitute a warranty. It may represent the result of Seller's test conditions, and the user must correlate it to actual application requirements. Actual performance is subject to the Seller's Warranty and Limitations of Liability.
4. Change in Specifications. Product specifications and accessories may be changed at any time based on improvements and other reasons. It is our practice to change part numbers when published ratings or features are changed, or when significant construction changes are made. However, some specifications of the Good may be changed without any notice. When in doubt, special part numbers may be assigned to fix or establish key specifications for your application. Please consult with your Seller's representative at any time to confirm actual specifications of purchased Good.
5. Errors and Omissions. The information in this catalog has been carefully checked and is believed to be accurate; however, no responsibility is assumed for clerical, typographical or proofreading errors, or omissions.

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, means "word" and is abbreviated "Wd" in documentation.

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... Indicates lists of one sort or another, such as procedures, precautions, etc.

© **OMRON, 1990**

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 Overview 2
1-2 Relay Circuits: The Roots of PC Logic 2
1-3 PC Terminology 3
1-4 OMRON Product Terminology 4
1-5 Overview of PC Operation 4
1-6 Peripheral Devices 5
1-7 Available Manuals 7

SECTION 2
Hardware Considerations 9

2-1 Indicators 10
2-2 PC Configuration 12

SECTION 3
Memory Areas 15

3-1 Introduction 16
3-2 Data Area Structure 16
3-3 IR (Internal Relay)Area 18
3-4 SR (Special Relay) Area 23
3-5 AR (Auxiliary Relay) Area 31
3-6 DM (Data Memory) Area 36
3-7 HR (Holding Relay) Area 37
3-8 TC (Timer/Counter) Area 37
3-9 LR (Link Relay) Area 38
3-10 Program Memory 39
3-11 File Memory 39
3-12 Trace Memory 39
3-13 TR (Temporary Relay) Area 39

SECTION 4
Writing and Inputting the Program 41

4-1 Basic Procedure 42
4-2 Instruction Terminology 42
4-3 Basic Ladder Diagrams 43
4-4 The Programming Console 58
4-5 Preparation for Operation 61
4-6 Inputting, Modifying, and Checking the Program 75
4-7 Controlling Bit Status 92
4-8 Work Bits (Internal Relays) 93
4-9 Programming Precautions 95
4-10 Program Execution 97

SECTION 5
Instruction Set 99
5-1 Notation 102
5-2 Instruction Format 102
5-3 Data Areas, Definer Values, and Flags 102
5-4 Differentiated Instructions 104
5-5 Coding Right-hand Instructions 104
5-6 Ladder Diagram Instructions 107
5-7 Bit Control Instructions 109
5-8 INTERLOCK and INTERLOCK CLEAR – IL(02) and ILC(03) 113
5-9 JUMP and JUMP END – JMP(04) and JME(05) 115
5-10 END – END(01) 116
5-11 NO OPERATION – NOP(00) 116
5-12 Timer and Counter Instructions 117
5-13 Data Shifting 127
5-14 Data Movement 135
5-15 Data Comparison 142
5-16 Data Conversion 148
5-17 BCD Calculations 158
5-18 Binary Calculations 174
5-19 Logic Instructions 179
5-20 Subroutines and Interrupt Control 182
5-21 Block Programming Instructions 190
5-22 Step Instructions 199
5-23 Special Instructions 208
5-24 Data Tracing (TRACE MEMORY SAMPLING – TRSM(45)) 211
5-25 File Memory Instructions 214
5-26 Intelligent I/O Instructions 217
5-27 Network Instructions 219
SECTION 6
Program Execution Timing 227
6-1 Cycle Time 228
6-2 Calculating Cycle Time 232
6-3 Instruction Execution Times 235
6-4 I/O Response Time 241
SECTION 7
Program Debugging and Execution 243
7-1 Debugging 244
7-2 Monitoring Operation and Modifying Data 252
7-3 File Memory Operations 266
7-4 Program Backup and Restore Operations 276

SECTION 8

Error Processing 285

8-1 Alarm Indicators 286
8-2 Programmed Alarms and Error Messages 286
8-3 Reading and Clearing Errors and Messages 286
8-4 Error Messages 286
8-5 Error Flags 289
8-6 Troubleshooting 290

Appendices 293

A. Standard Models 293
B. Programming Instructions 303
C. Programming Console Operations 335
D. Error and Arithmetic Flag Operation 349
E. Data Areas 353
F. I/O Assignment Records Sheets 357
G. Program Coding Sheet 363
H. Data Conversion Table 367
I. Extended ASCII 369

Glossary 371

Index 389

Revision History 395

About this Manual:

The OMRON C1000H and C2000H offer an effective way to automate processing. Manufacturing, assembly, packaging, and many other processes can be automated to save time and money. Distributed control systems can also be designed to allow centralized monitoring and supervision of several separate controlled systems. Monitoring and supervising can be done through a host computer, connecting the controlled system to a data bank. It is thus possible to have adjustments in system operation made automatically to compensate for requirement changes.

The C1000H and C2000H are Rack PCs, i.e., various Units are combined to produce the optimum control system for each application, which can just as easily be expanded up to the maximum I/O capacity of the PC by adding additional Units in the future. Additional Units include dedicated Special I/O Units that can be used for specific tacks and Link Units that can be used to build more highly integrated systems, including an optical LAN.

The C1000H and C2000H are equipped with large programming instruction sets, data areas, and other features to control processing directly or remotely. Programming utilizes ladder-diagram programming methods, which are described in detail for those unfamiliar with them.

This manual describes the characteristics and abilities of the C1000H and C2000H, programming operations and instructions, and other aspects of operation and preparation that demand attention. Before attempting to operate the PC, thoroughly familiarize yourself with the information contained herein. Hardware information is provided in detail in the *C1000H/C2000H Installation Guide*. A table of other manuals that can be used in combination with this manual is provided at the end of *Section 1 Introduction*.

Section 1 Introduction explains the background and some of the basic terms used in ladder-diagram programming. It also provides an overview of the process of programming and operating a PC and explains basic terminology used with OMRON PCs. Descriptions of peripheral devices used with the C1000H and C2000H and a table of other manuals available to use with this manual for special PC applications are also provided.

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

Section 3 Memory Areas takes a look at the way memory is divided and allocated and explains the information provided there to aid in programming. It also explains how I/O is managed in memory and how bits in memory correspond to specific I/O points.

Section 4 Writing and Inputting the Programming explains the basics of ladder-diagram programming and how the program is input into the PC using a Programming Console. The elements that make up the 'ladder' part of a ladder-diagram program and how execution of this program is controlled are explained. The user should be able to write and input a basic "input-output" program after finishing this section.

S*ection 5 Instruction Set* then goes on to describe individually 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 provides the Programming Console procedures used to debug the program and to monitor and control operation.

Finally, ***Section 8 Troubleshooting*** provides information on error indications and other means of reducing down time. Information in this section is also sometimes necessary when debugging a program.

The appendices provide tables of standard OMRON products available for the C1000H and C2000H, reference tables of instructions and Programming Console operations, and other information helpful in PC operation.

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

SECTION 1
Introduction

This section gives a brief overview of the history of Programmable Controllers and explains terms commonly used in ladder-diagram programming. It also provides an overview of the process of programming and operating a PC and explains basic terminology used with OMRON PCs. Descriptions of peripheral devices used with the C1000H and C2000H, and a table of other manuals available to use with this manual for special PC applications, are also provided.

1-1 Overview 2
1-2 Relay Circuits: The Roots of PC Logic 2
1-3 PC Terminology 3
1-4 OMRON Product Terminology 4
1-5 Overview of PC Operation 4
1-6 Peripheral Devices 5
1-7 Available Manuals 7

1-1 Overview

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

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

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

To achieve proper control, the C1000H and C2000H use a form of PC logic called ladder-diagram programming. This manual is written to explain ladder-diagram programming and to prepare the reader to program and operate the C1000H and/or C2000H.

1-2 Relay Circuits: The Roots of PC Logic

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

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

Relay vs. PC Terminology

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

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

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

Actually there is not a total equivalence between these terms. The term condition is only used to describe ladder diagram programs in general and is specifically equivalent to one of certain set of basic instructions. The terms input and output are not used in programming per se, except in reference to I/O bits that are assigned to input and output signals coming into and leaving the PC. Normally open conditions and normally closed conditions are explained in *4-2 The Ladder Diagram*.

1-3 PC Terminology

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

PC

Because the C1000H and C2000H are Rack PCs, there is no one product that is a C1000H or C2000H PC. That is why we talk about the configuration of the PC, because a PC is a configuration of smaller Units.

To have a functional PC, you would need to have a CPU Rack with at least one Unit mounted to it that provides I/O points. With a Duplex System, you would also need an CPU I/O Rack to mount the I/O Unit, because the Duplex CPU Rack does not provide slots for mounting other Units. When we refer to the PC, however, we are generally talking about the CPU and all of the Units directly controlled by it through the program. This does not include the I/O devices connected to PC inputs and outputs.

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

Inputs and Outputs

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

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

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

Controlled System and Control System

The Control System includes the PC and all I/O devices it uses to control an external system. A sensor that provides information to achieve control is an input device that is clearly part of the Control System. The controlled system is the external system that is being controlled by the PC program through

these I/O devices. I/O devices can sometimes be considered part of the controlled system, e.g., a motor used to drive a conveyor belt.

1-4 OMRON Product Terminology

OMRON products are divided into several functional groups that have generic names. A*ppendix A Standard Models* list products according to these groups. The term **Unit** is used to refer to all of the OMRON PC products. Although a Unit is any one of the building blocks that goes together to form a C1000H or C2000H PC, its meaning is generally, but not always, limited in context to refer to the Units that are mounted to a Rack. Most, but not all, of these products have names that end with the word Unit.

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

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

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

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

1-5 Overview of PC Operation

The following are the basic steps involved in programming and operating a C1000H or C2000H. Assuming you have already purchased one or more of these PCs, you must have a reasonable idea of the required information for steps one and two, which are discussed briefly below. This manual is written to explain steps three through six, eight, and nine. The relevant sections of this manual that provide more information are listed with each of these steps.

1, 2, 3...

1. Determine what the controlled system must do, in what order, and at what times.
2. Determine what Racks and what Units will be required. Refer to the *C1000H/C2000H Installation Guide*. If a Link System is required, refer to the appropriate *System Manual*.
3. On paper, assign all input and output devices to I/O points on Units and determine which I/O bits will be allocated to each. If the PC includes Special I/O Units or Link Systems, refer to the individual *Operation Manuals* or *System Manuals* for details on I/O bit allocation. (*Section 3 Memory Areas)*
4. Using relay ladder symbols, write a program that represents the sequence of required operations and their inter-relationships. Be sure to also program appropriate responses for all possible emergency situations. (*Section 4 Writing and Inputting the Program, Section 5 Instruction Set, Section 6 Program Execution Timing)*
5. Input the program and all required operating parameters into the PC. (*Section 7 Program Input, Debugging, and Execution*)
6. Debug the program, first to eliminate any syntax errors, and then to find execution errors. (*Section 7 Program Input, Debugging, and Execution* and *Section 8 Troubleshooting*)

7. Wire the PC to the controlled system. This step can actually be started as soon as step 3 has been completed. Refer to the *C1000H/C2000H Installation Guide* and to *Operation Manuals* and *System Manuals* for details on individual Units.
8. Test the program in an actual control situation and carry out fine tuning as required. (*Section 4 Writing and Inputting the Program, Section 7 Program Debugging and Execution*, and *Section 8 Troubleshooting*)
9. Record two copies of the finished program on masters and store them safely in different locations. (*Section 7 Program Debugging and Execution*)

Control System Design

Designing the Control System is the first step in automating any process. A PC can be programmed and operated only after the overall Control System is fully understood. Designing the Control System requires, first of all, a thorough understanding of the system that is to be controlled. The first step in designing a Control System is thus determining the requirements of the controlled system.

Input/Output Requirements

The first thing that must be assessed is the number of input and output points that the controlled system will require. This is done by identifying each device that is to send an input signal to the PC or which is to receive an output signal from the PC. Keep in mind that the number of I/O points available depends on the configuration of the PC. Refer to *3-2 IR Area* for details on I/O capacity and the allocation of I/O bits to I/O points.

Sequence, Timing, and Relationships

Next, determine the sequence in which control operations are to occur and the relative timing of the operations. Identify the physical relationships between the I/O devices as well as the kinds of responses that should occur between them.

For instance, a photoelectric switch might be functionally tied to a motor by way of a counter within the PC. When the PC receives an input from a start switch, it could start the motor. The PC could then stop the motor when the counter has received a specified number of input signals from the photoelectric switch.

Each of the related tasks must be similarly determined, from the beginning of the control operation to the end.

Unit Requirements

The actual Units that will be mounted or connected to PC Racks must be determined according to the requirements of the I/O devices. Actual hardware specifications, such as voltage and current levels, as well as functional considerations, such as those that require Special I/O Units or Link Systems will need to be considered. In many cases, Special I/O Units, Intelligent I/O Units, or Link Systems can greatly reduce the programming burden. Details on these Units and Link Systems are available in appropriate *Operation Manuals* and *System Manuals.*

Once the entire Control System has been designed, the task of programming, debugging, and operation as described in the remaining sections of this manual can begin.

1-6 Peripheral Devices

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

names have been placed in bold when introduced in the following descriptions.

Programming Console

A Programming Console is the simplest form of programming device for OMRON PCs. Although a **Programming Console Adapter** is sometimes required, all Programming Consoles are connected directly to the CPU without requiring a separate interface. The Programming Console also functions as an interface to transfer programs to a standard cassette tape recorder.

Various types of Programming Console are available, including both CPU-mounting and Hand-held models. Programming Console operations are described later in this manual.

Graphic Programming Console:

The GPC allows you to perform all the operations of the Programming Console as well as many additional ones. PC programs can be written on-screen in ladder-diagram form as well as in mnemonic form. As the program is written, it is displayed on a liquid crystal display, making confirmation and modification quick and easy. Syntax checks may also be performed on the programs before they are downloaded to the PC. Many other functions are available, depending on the Memory Pack used with the GPC.

A **Peripheral Interface Unit** is required to interface the GPC to the PC.

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

Ladder Support Software: LSS

LSS is designed to run on IBM AT/XT compatibles to enable all of the operations available on the GPC. Using an Optical Host Link Unit also enables the use of optical fiber cable to connect the FIT to the PC. Wired Host Link Units are available when desired. (Although FIT does not have optical connectors, conversion to optical fiber cable is possible by using **Converting Link Adapters**.)

A **Host Link Unit** is required to interface a computer running LSS to the PC.

Factory Intelligent Terminal: FIT

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

A **Peripheral Interface Unit** or **Host Link Unit** is required to interface the FIT to the PC. Using an Optical Host Link Unit also enables the use of optical fiber cable to connect the FIT to the PC. Wired Host Link Units are available when desired. (Although FIT does not have optical connectors, conversion to optical fiber cable is possible by using **Converting Link Adapters**.)

PROM Writer

Other than its applications described above, the PROM Writer can be mounted to the PC’s CPU to write programs to EPROM chips.

Floppy Disk Interface Unit

Other than its applications described above, the Floppy Disk Interface Unit can be mounted to the PC’s CPU to interface a floppy disk drive and write programs onto floppy disks.

Printer Interface Unit

Other than its applications described above, the Printer Interface Unit can be mounted to the PC’s CPU to interface a printer or X-Y plotter to print out programs in either mnemonic or ladder-diagram form.

1-7 Available Manuals

The following table lists other manuals that may be required to program and/or operate the C1000H and C2000H. *Operation Manuals* and/or *Operation Guides* are also provided with individual Units and are required for wiring and other specifications.

Name	Cat. No.	Contents
C1000H/C2000H Operation Manual	W140	Software specifications
GPC Operation Manual	W84	Programming procedures for the GPC (Graphics Programming Console)
FIT Operation Manual	W150	Programming procedures for using the FIT (Factory Intelligent Terminal
LSS Operation Manual	W237	Programming procedures for using LSS (Ladder Support Software)
Data Access Console Operation Guide	W173	Data area monitoring and data modification procedures for the Data Access Console
Printer Interface Unit Operation Guide	W107	Procedures for interfacing a PC to a printer
PROM Writer Operation Guide	W155	Procedures for writing programs to EPROM chips
Floppy Disk Interface Unit Operation Guide	W119	Procedures for interfacing a PC to a floppy disk drive
Wired Remote I/O System Manual	W120	Information on building a Wired Remote I/O System to enable remote I/O capability
Optical Remote I/O System Manual	W136	Information on building an Optical Remote I/O System to enable remote I/O capability
PC Link System Manual	W135	Information on building a PC Link System to automatically transfer data between PCs
Host Link System Manual	W143	Information on building a Host Link System to manage PCs from a 'host' computer
SYSMAC NET Link System Manual	W114	Information on building a SYSMAC NET Link System and thus create an optical LAN integrating PCs with computers and other peripheral devices
SYSMAC LINK System Manual	W174	Information on building a SYSMAC LINK System to enable automatic data transfer, programming, and programmed data transfer between the PCs in the System

SECTION 2
Hardware Considerations

This section provides information on hardware aspects of the C1000H and C2000H that are relevant to programming and software operation. These include indicators on the CPU and Duplex Unit and basic PC configuration. This information is covered in detail in the *C1000H/C2000H Installation Guide*.

2-1 Indicators 10
2-2 PC Configuration 12

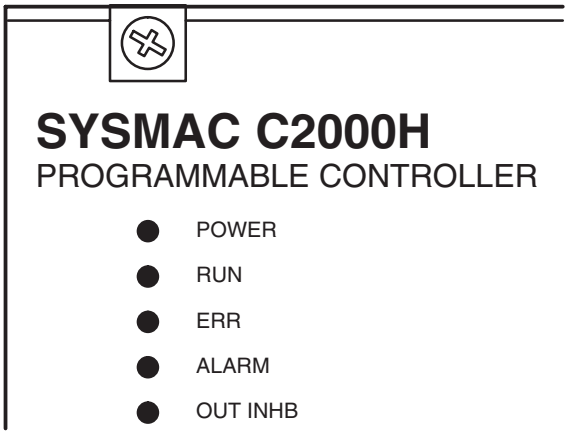
2-1 Indicators

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

CPU Indicators

CPU indicators are shown below and are described in the following table. Indicators are the same for both C1000H and C2000H.

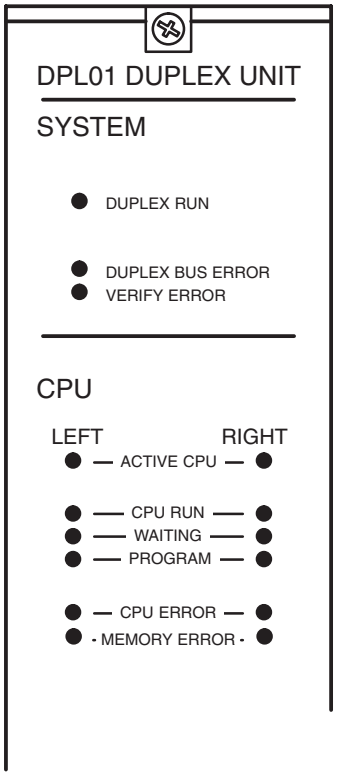
Indicator	Function
POWER	Lights when power is supplied to the CPU.
RUN	Lights when the CPU is operating normally.
ERR	Lights when an error is discovered in error diagnosis operations. When this indicator lights, the RUN indicator will go off, CPU operation will be stopped, and all outputs from the PC will be turned OFF.
ALARM	Lights when an error is discovered in error diagnosis operations. PC operation will continue.
OUT INHB	Lights when the Output OFF bit, SR bit 25215, is turned ON. All outputs from the PC will be turned OFF.



Duplex Unit Indicators

Duplex Unit indicators are shown and described below. Refer to the *C1000H/C2000H Installation Guide* for details. Duplex operation is only available on C2000H Units.

Indicator	Function
DUPLEX RUN	Lights when the Duplex Unit is operating normally.
DUPLEX BUS ERROR	Lights when an error has occurred in the Duplex Unit bus. The DUPLEX RUN indicator will go out and the active CPU will switch to simplex operation.
VERIFY ERROR	Lights when the two CPUs do not contain the same program. The DUPLEX RUN indicator will go out and the active CPU will switch to simplex operation.
ACTIVE CPU	Indicate which CPU is active.
CPU RUN	Lights whenever the RUN indicators on the CPUs are lit.
WAITING	Lights at the beginning of duplex operation until the programs have been verified (1 to 20 seconds). Will remain lit if program execution does not start properly or if an error occurs in starting duplex operation.
PROGRAM	Lights when the PCs are in PROGRAM mode.
CPU ERROR	Lights when CPU errors occur in the CPUs. CPU operation will stop and the CPU RUN indicator will go out.
MEMORY ERROR	Lights when memory errors occur in the CPUs. CPU operation will stop and the CPU RUN indicator will go out.



2-2 PC Configuration

The basic PC configuration consists of either two or three types of Rack: a CPU Rack and Expansion I/O Racks for C2000H Simplex Systems and the C1000H, and a CPU Rack, a CPU I/O Rack, and Expansion I/O Racks for C2000H Duplex Systems. The Expansion I/O Racks are not a required part of the basic system. They are used to increase the number of I/O points. An illustration of these Racks is provided in *3-2 IR Area.* A fourth type of Rack, called a Slave Rack, can be used when the PC is provided with a Remote I/O System.

CPU Racks

A C2000H Simplex CPU Rack or C1000H CPU Rack consists of four components: (1) The CPU Backplane, to which the CPU, the Power Supply, and other Units are mounted. (2) The CPU, which executes the program and controls the PC. (3) Other Units, such as I/O Units, Special I/O Units, Link Units, and Intelligent I/O Units, which provide the physical I/O terminals corresponding to I/O points. (4) The Power Supply, which provides power to the CPU Rack.

A C2000H Simplex or C1000H CPU Rack can be used alone or it can be connected to other Racks to provide additional I/O points. The C1000H CPU Rack provides five or eight slots to which these other Units can be mounted depending on the backplane used; the C2000H Simplex CPU Rack provides six slots.

A C2000H Duplex CPU Rack Consists of a Duplex CPU Backplane, two CPUs connected by a Duplex Unit, and a Power Supply. A Duplex CPU Rack is completely filled by these Units and does not provide any slots for other Units.

In a Duplex System, one of the two CPUs is active and the other is on standby as long as both are operating normally. Both CPUs must contain the same size and type of Memory Unit, and the same program. If the active CPU fails to operate normally, the standby CPU takes control with simplex operation until normal duplex operation can be restored. If an error occurs in the standby CPU or in the Duplex Unit, the active CPU switches to simplex operation. The Duplex Unit coordinates these processes.

CPU I/O Racks

To provide slots for other Units in a C2000H Duplex System, a CPU I/O Rack is connected to the CPU Rack. The CPU I/O Rack is built on a I/O Backplane, which provides eight slots for other Units. In a Duplex System, one CPU Rack and one CPU I/O Rack are used in a pair. The CPU has no means of sending and receiving I/O signals without an I/O Rack. The I/O Rack is always connected to a C2000H Duplex CPU via the connectors on the Backplanes, allowing communication between the two Racks.

Expansion I/O Racks

An Expansion I/O Rack can be thought of as an extension of the PC because it provides additional slots to which other Units can be mounted. It is built onto an Expansion I/O Backplane to which a Power Supply and up to eight other Units are mounted.

An I/O Interface Unit is also mounted to any Expansion I/O Rack to interface the Rack to the CPU Rack. Also, an I/O Control Unit must be mounted to any C1000H or C2000H Simplex CPU Rack to which one or more Expansion I/O Racks are mounted. In a C2000H Duplex System the I/O Control Unit is mounted to the CPU I/O Rack.

An Expansion I/O Rack is always connected to the CPU via the connectors on the Backplanes, allowing communication between the two Racks. In a

C1000H System, or in C2000H Simplex or Duplex Systems, up to seven Expansion I/O Racks can be connected in series to the CPU Rack or, in a C2000H Duplex System, to the CPU I/O Rack.

Unit Mounting Position

Only I/O Units and Special I/O Units can be mounted to Slave Racks. All I/O Units, Special I/O Units, Remote I/O Master Units, and I/O Link Units can be mounted to any slot on all other Racks. All other Units, including Interrupt Input Units, File Memory Units, and all other Link Units must be mounted only to certain slots on specific Racks. All Units occupy only one slot except for the PID Unit and some Position Control Units.

All Units that do not require specific slots can be mounted in any order. Units that do require specific slots can be mounted in any order within the required slots.

Refer to the *C1000H/C2000H Installation Guide* for details about which slots can be used for which Units and other details about PC configuration. The way in which I/O points on Units are allocated in memory is described in *3-2 IR Area*.

SECTION 3
Memory Areas

Various types of data are required to achieve effective and correct control. To facilitate managing this data, the PC is provided with various **memory areas** for data, each of which performs a different function. The areas generally accessible by the user for use in programming are classified as **data areas**..

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

3-1 Introduction 16
3-2 Data Area Structure 16
3-3 IR (Internal Relay)Area 18
3-4 SR (Special Relay) Area 23
3-4-1 Link System Flags and Control Bits 24
3-4-2 Data Retention Control Bit 28
3-4-3 Output OFF Bit 29
3-4-4 FAL (Failure Alarm) Area 29
3-4-5 Low Battery Flag 29
3-4-6 Cycle Time Error Flag 29
3-4-7 I/O Verification Error Flag 29
3-4-8 First Cycle Flag 29
3-4-9 Clock Pulse Bits 29
3-4-10 Step Flag 30
3-4-11 Duplex System Flags 30
3-4-12 Instruction Execution Error Flag, ER 30
3-4-13 Arithmetic Flags 31
3-5 AR (Auxiliary Relay) Area 31
3-5-1 SYSMAC LINK System Data Link Settings 33
3-5-2 Active Node Flags 33
3-5-3 SYSMAC LINK/SYSMAC NET Link System Service Time 34
3-5-4 Tracing Flags and Control Bits 34
3-5-5 File Memory Flags and Control Bits 34
3-5-6 On-line Removal Bits 35
3-5-7 Power-off Counter 35
3-5-8 Network Parameter Flags 36
3-5-9 Link Unit Mounted Flags 36
3-5-10 CPU-mounting Device Flag 36
3-5-11 FALS-generating Address 36
3-5-12 Cycle Time Indicators 36
3-6 DM (Data Memory) Area 36
3-7 HR (Holding Relay) Area 37
3-8 TC (Timer/Counter) Area 37
3-9 LR (Link Relay) Area 38
3-10 Program Memory 39
3-11 File Memory 39
3-12 Trace Memory 39
3-13 TR (Temporary Relay) Area 39

3-1 Introduction

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

Area	Acronym	Range	Function
Internal Relay	IR	Words: 000 to 236 Bits: 0000 to 23615	Used to control I/O points, other bits, timers, and counters, and to temporarily store data.
Special Relay	SR	Words: 237 to 255 Bits: 23700 to 25515	Contains system clocks, flags, control bits, and status information. Many words are dedicated for use by Link Systems
Auxiliary Relay	AR	Words: AR 00 to AR 27 Bits: AR 00 to AR 2715	Contains flags and bits for special functions, such as write-protecting the FM area.
Data Memory	DM	C1000H: DM 0000 to DM 4095 (words only) C2000H: DM 0000 to DM 6655 (words only)	Used for internal data storage and manipulation.
Holding Relay	HR	Words: HR 00 to HR 99 Bits: HR 0000 to HR 9915	Used to store data and to retain the data values when the power to the PC is turned off.
Timer/Counter	TC	TC 000 to TC 511 (TC numbers used to access other information)	Used to define timers and counters, and to access completion flags, PV, and SV.
Link Relay	LR	Words: LR 00 to LR 63 Bits: LR 0000 to 6315	Used for inter-PC communication in PC Link Systems.
Temporary Relay	TR	TR 00 to TR 07 (bits only)	Used to temporarily store execution conditions.
Program Memory	UM	UM: Depends on Memory Unit used.	Contains the program executed by the CPU.
File Memory	FM	FM: 0000 to 0999 or 0000 to 1999	Located in a File Memory Unit mounted to the CPU Rack and used to store programs or data.
Trace Memory	TM	TM: Traces of 250 instructions	Used to store results from traces of program execution.

Work Bits and Words

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

Flags and Control Bits

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

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

3-2 Data Area Structure

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

An actual data location within any data area but the TC area is designated by its address. The address designates the bit or word within the area where the desired data is located. The TC area consists of TC numbers, each of which is used for a specific timer or counter defined in the program. Refer to *3-7 TC Area* for more details on TC numbers and to *5-11 Timer and Counter Instructions* for information on their application.

The rest of the data areas (i.e., the IR, SR, HR, DM, AR, and LR areas) consist of words, each of which consists of 16 bits numbered 00 through 15 from right to left. IR words 000 and 001 are shown below with bit numbers. Here, the content of each word is shown as all zeros. Bit 00 is called the rightmost bit; bit 15, the leftmost bit.

The term least significant bit is often used for rightmost bit; the term most significant bit, for leftmost bit. These terms are not used in this manual because a single data word is often split into two or more parts, with each part used for different parameters or operands. When this is done, the rightmost bits of a word may actually become the most significant bits, i.e., the leftmost bits in another word,when combined with other bits to form a new word.

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

The DM area is accessible by word only; you cannot designate an individual bit within a DM word. Data in the IR, SR, HR, AR, and LR areas is accessible either by word or by bit, depending on the instruction in which the data is being used.

To designate one of these areas by word, all that is necessary is the acronym (if required) and the two-, three-, or four-digit word address. To designate an area by bit, the word address is combined with the bit number as a single four- or five-digit address. The following table show examples of this. The two rightmost digits of a bit designation must indicate a bit between 00 and 15, i.e., the rightmost digit must be 5 or less the next digit to the left, either 0 or 1.

The same TC number can be used to designate either the present value (PV) of the timer or counter, or a bit that functions as the Completion flag for the timer or counter. This is explained in more detail in *3-7 TC Area*.

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

Data Structure

Word data input as decimal values is stored in binary-coded decimal (BCD); word data entered as hexadecimal is stored in binary form. Each four bits of a word represents one digit, either a hexadecimal or decimal digit, numerically equivalent to the value of the binary bits. One word of data thus con-

tains four digits, which are numbered from right to left. These digit numbers and the corresponding bit numbers for one word are shown below.

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

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

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

Converting Different Forms of Data

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

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

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

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

Decimal Points

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

3-3 IR (Internal Relay)Area

The IR area is used both as data to control I/O points, and as work bits to manipulate and store data internally. It is accessible both by bit and by word. Those words that are used to control I/O points are called I/O words. Bits in I/O words are called I/O bits.

The number of I/O words varies between the C1000H and C2000H as shown in the following table. As shown, the remaining words for each PC are work

words (work bits), which can be used in programming to manipulate data and control other bits. IR area work bits are reset when power is interrupted or PC operation is stopped.

PC	I/O words	I/O bits	Work words	Work bits
C1000H	000 through 063	00000 through 06315	064 through 236	06400 through 23615
C2000H	000 though 127	00000 though 12715	128 through 236	12800 through 23615

I/O Words

The maximum number of available I/O bits is 16 (bits/word) times the number of I/O words, i.e., 1,024 bits for the C1000H; 2,048 for the C2000H. I/O bits are assigned to input or output points as described later in this section (see *Word Allocations*).

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

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

C1000H Remote I/O Words

Although normally not available as I/O words in the C1000H, IR 064 through 127 are used as I/O words by C1000H Remote I/O Systems to control I/O points located on Remote I/O Units. Refer to the *Optical Remote I/O* and *Wired Remote I/O System Manuals* for details on Remote I/O Systems.

Input Bit Usage

Input bits can be used to directly input external signals to the PC and can be used in any order in programming. Each input bit can also be used in as many instructions as required to achieve effective and proper control. They cannot be used in instructions that control bit status, e.g., the OUTPUT, DIFFERENTIATE UP, and KEEP instructions.

Output Bit Usage

Output bits are used to output program execution results and can be used in any order in programming. Because outputs are refreshed only once during each cycle (i.e., once each time the program is executed), any output bit can be used in only one instruction that controls its status, including OUT, KEEP(11), DIFU(13), DIFD(14) and SFT(10). If an output bit is used in more than one such instruction, only the status determined by the last instruction will actually be output from the PC.

See *5-12 Shift Register – SFT(10)* for an example that uses an output bit in two 'bit-control' instructions.

Word Allocations

I/O words in the IR area are allocated to Units mounted on Racks by performing the I/O Table Registration operation. This operation creates in memory a table called an I/O table that records what words and how many words are allocated to the Unit in each slot and whether these words are input or output words. The actual procedure for this operation is described in *Section 7 Program Input, Debugging, and Execution.*

When the I/O Table Registration operation is performed, the system automatically assigns word addresses to Units in the order in which they are mounted left to right on the CPU Rack (or, in a Duplex System, on the CPU I/O Rack) and then continuing left to right on any Expansion I/O Racks in the order that the Expansion I/O Racks are connected. I/O words start from IR 000 for the first Unit and continue consecutively: IR 001, IR 002, etc.

Because different Units can require a different number of words, there are no specific words associated with any particular slot. Rather, each Unit is as-

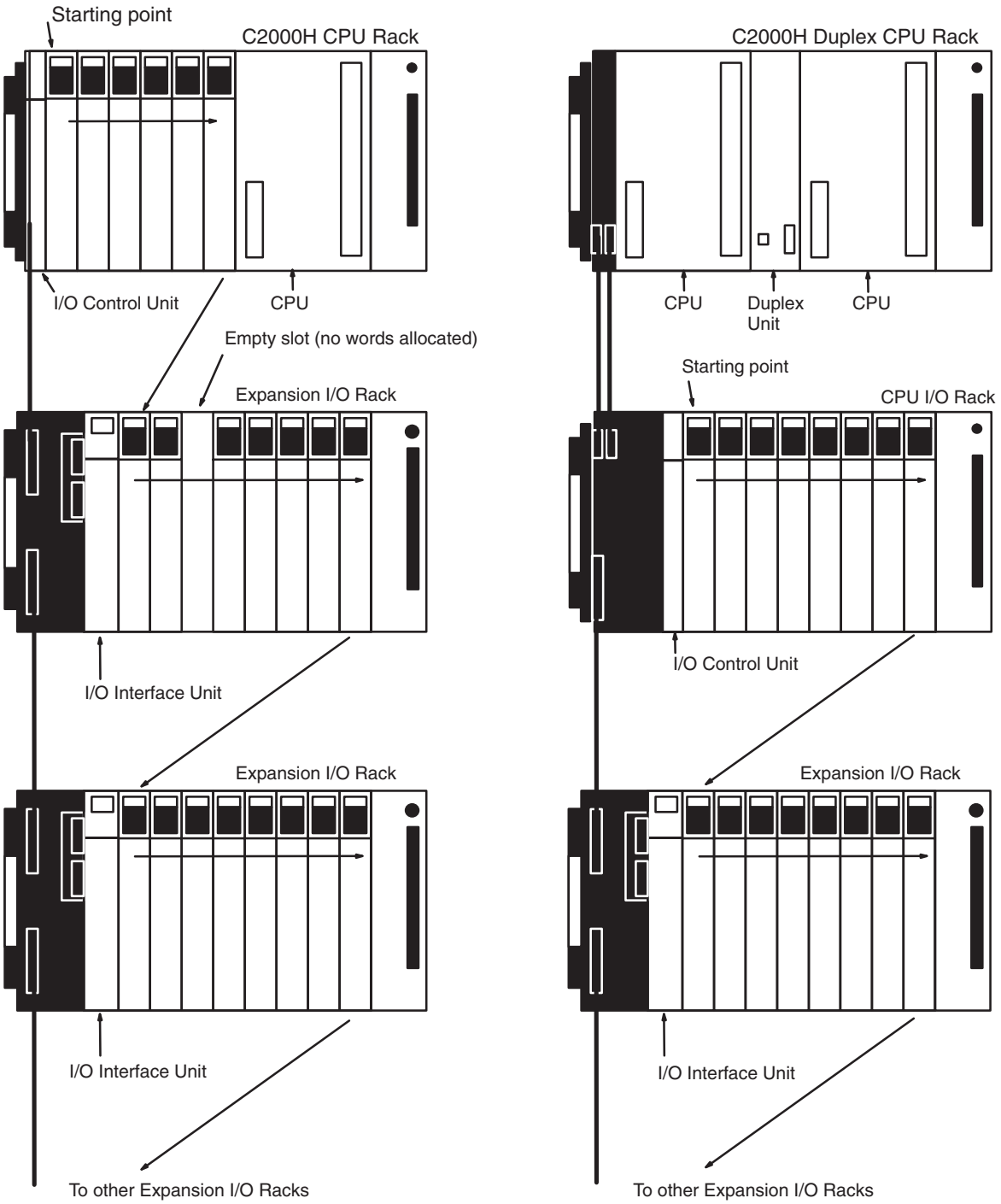
signed the next word(s) following the word(s) assigned to the previous Unit. If there are any empty slots, no words will be assigned. Words are only assigned when a Unit is mounted; all empty slots are skipped. The number of I/O words allocated to each type of Unit is shown below.

Unit	Words required
16-pt I/O Units	1 word
24- or 32-pt I/O Units	2 words
64-pt I/O Units	4 words
Interrupt Input Unit	1 word
Dummy I/O Unit	Set to 1, 2, or 4 words
Analog I/O Units	2 or 4 words
High-speed Counter Units	CT012 and CT041: 2 words; CT001: 4 words
MCR Units	4 words
PID Unit	4 words
I/O Interface Unit	None
Cam Positioner	2 or 4 words
ASCII Unit	2 or 4 words
Host Link Unit	None
PC Link Unit	None (assigned LR words)
SYSMAC LINK Unit	None (assigned LR and/or DM words)
SYSMAC NET Link Unit	None (assigned LR and/or DM words)
Remote I/O Master Unit	None*
Remote I/O Slave Unit	None*
I/O Link Unit	1 or 2 words
File Memory Unit	None
Position Control Units	2 or 4 words
I/O Control Unit	None
ID Sensor Unit	5 words

*Remote I/O Systems are described briefly below.

Once the word(s) assigned to a Unit has been determined, the use of individual bits in the word(s) is determined by the type of Unit. If the Unit is a Special I/O Unit, I/O Link Unit, or Intelligent I/O Unit, each bit will have a dedicated function. Refer to *Operation Manuals* for the relevant Units for details. With I/O Units, bits within a word are assigned to terminals starting at the top of the I/O Unit with bit 00 and going sequentially to the bottom. If the first Unit on the left of the first Rack (CPU Rack for all but C2000H Duplex Systems; CPU I/O Rack for C2000H Duplex Systems) is an Input Unit, the top terminals (i.e., the top input point) will be assigned IR 00000, the next terminals, IR 00001, and so forth for all of the terminals on the Unit. The allocation or-

der is illustrated below. Arrows indicate the order in which words are allocated to Units.



Remote I/O Systems

Although a Remote I/O Master Unit (called a Master for short) mounted to a Rack, and any Remote I/O Slave Units connected to it, are not allocated I/O words, any Units on the Slave Racks or other Remote I/O Units connected to the Master are allocated I/O words.

Units mounted to a Slave Rack are allocated words according to the slot where the Master is mounted. If IR 16 is the last word allocated to the Unit to the left of the Master, IR 17 would be allocated to the first Unit on the left end of the first Slave Rack connected to the Master. Allocations would continue to all of the Units on all Slave Racks before continuing to the Unit mounted to

the right of the Master. If 18 words we required by the Units on the Slave Racks, the first word allocated to the Unit to the right of the Master would be IR 35.

I/O word allocations for other Remote I/O Units are not according to the mounting order of their Master. They are determined, rather, by the word setting on the Unit and the word multiplier set for the Master through which they are controlled, according to the following equation.

Allocated word = word setting + (32 x word multiplier)

Word allocations for Remote I/O Systems are described in more detail in the *Wired* and *Optical Remote I/O System Manuals*.

Rack Changes

Once Units have been mounted and the I/O Table Registration operation has been performed, a change to any Unit mounted to a Rack that affects the type of I/O word, or the number of words, required by the Unit will cause an I/O verification error to occur. This includes adding Units to previously unused slots or removing Units that have already been allocated word(s). A Unit can, however, be replaced with another Unit that requires the same number of input words and the same number of output words without generating an I/O verification error.

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

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

Sometimes program changes can be avoided when a Unit is removed from a Rack by reserving words. Although designed to enable slot reservations for future use, a slot reservation can be left permanently to prevent what could be extensive program changes.

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

Word Reservations

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

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

Once an I/O table has been registered, it can be changed using the I/O Table Change operation described in *Section 7 Program Input, Debugging, and Execution*. This operation can be used to reserve up to four input words, output words, or non-defined words at a time. It cannot be used to reserve words for Units in Remote I/O Systems or for Interrupt Input Units.The I/O

Table Change operation must be performed after the I/O Table Registration operation, otherwise all word reservations will be cancelled, and I/O Table Change will have to be repeated.

3-4 SR (Special Relay) Area

The SR area contains flags and control bits used for monitoring PC operation, accessing clock pulses, and signalling errors. SR area word addresses range from 247 through 255; bit addresses, from 24700 through 25515.

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

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

SR bits 25209 through 25215 are all control bits. They can be turned ON and OFF from the program, i.e., they can be manipulated with the OUTPUT and OUTPUT NOT instructions. Any of these bits not assigned specific functions should be left OFF. Bits in words SR 237 through SR 251, only, can be used as work bits if the Systems for which these bits are dedicated are not used by the PC.

Word(s)	Bit(s)	Function
237	00 to 07	Completion code output area following execution of SEND(90)/RECV(98) for SYSMAC LINK System or NET Link System
	08 to 15	Not used.
238 to 241	00 to 15	Data link status output area for operating level 0 of SYSMAC LINK or SYSMAC NET Link System
242 to 245	00 to 15	Data link status output area for operating level 1 of SYSMAC LINK or SYSMAC NET Link System
246	00 to 15	Not used.
247 to 250	00 to 07	PC Link Unit Run Flags or data link status for operating level 1
	08 to 15	PC Link Unit Error Flags or data link status for operating level 1
251	00 to 15	Remote I/O Error Flags
252	00 and 01	Not used.
	02	Operating Level 0 Data Link Operating Flag
	03	SEND(90)/RECV(98) Error Flag
	04	SEND(90)/RECV(98) Enable Flag
	05	Operating Level 1 Data Link Operating Flag
	06	Rack-mounting Host Link Unit Level 1 Error Flag
	07	Not used.
	08	CPU-mounting Host Link Unit Error Flag
	09	CPU-mounting Host Link Unit Restart Bit
	10	Leave set to 0.
	11	Not used.
	12	Data Retention Control Bit
	13	Rack-mounting Host Link Unit Restart Bit
	14	Leave set to 0.
	15	Output OFF Bit

Word(s)	Bit(s)	Function
253	00 to 07	FAL number output area.
	08	Low Battery Flag
	09	Cycle Time Error Flag
	10	I/O Verification Error Flag
	11	Rack-mounting Host Link Unit Level 0 Error Flag
	12	Remote I/O Error Flag
	13	Normally ON Flag
	14	Normally OFF Flag
	15	First Cycle
254	00	1-minute clock pulse bit
	01	0.02-second clock pulse bit
	02 to 06	Reserved for function expansion. Do not use.
	07	Step Flag
	08 to 12	Duplex System flags
	13 to 15	Reserved for function expansion. Do not use.
255	00	0.1-second clock pulse bit
	01	0.2-second clock pulse bit
	02	1.0-second clock pulse bit
	03	Instruction Execution Error (ER) Flag
	04	Carry (CY) Flag
	05	Greater Than (GR) Flag
	06	Equals (EQ) Flag
	07	Less Than (LE) Flag

3-4-1 Link System Flags and Control Bits

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

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

Remote I/O Systems

Word 251 is used to indicate errors in Remote I/O Systems. The function of each bit is described below. Refer to *Optical* and *Wired Remote I/O System Manuals* for details.

Bit 00 - Error Check Bit

If there are errors in more than one Remote I/O Unit, word 251 will error information for only the first one. Data for the remaining Units will be stored in memory and can be accessed by turning the Error Check bit ON and OFF. Be sure to record data for the first error, which will be cleared when data for the next error is displayed.

Bits 01 and 02

Not used.

Bit 03 - Remote I/O Error Flag

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

Bits 04 to 15

If the content of bits 12 through 15 is B, an error has occurred in a Remote I/O Master or Slave Unit, and the content of bits 08 through 11 will indicate the mounting order of the Master of the Remote I/O Subsystem involved. These numbers are assigned to Masters in the order that they are mounted to the CPU and Expansion I/O Racks. If the error is in the Master, bit 7 will be ON. If the error is in a Slave, bit 07 will be OFF, and bits 04 through 06 will provide the unit number of the Slave where the error occurred.

If the content of bits 12 through 15 is other than B, an error has occurred in an Optical I/O Unit, I/O Link Unit, or Remote Terminal. Here, bits 08 through 15 will provide the word address that has been set on the Unit and bits 05 and 06 will provide the word multiplier of the Master of the Remote I/O Subsystem to which the Unit belongs. The word address setting and word multiplier can be used to find the actual word allocated to the Unit as follows:

Allocated word = Word setting + (word multiplier x 32)

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

Host Link Systems

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

Flag	Bit
Rack-mounting Host Link Unit Level 1 Error Flag	25206
CPU-mounting Host Link Unit Error Flag	25208
CPU-mounting Host Link Unit Restart Bit	25209
Rack-mounting Host Link Unit Restart Bit	25213
Rack-mounting Host Link Unit Level 0 Error Flag	25311

SYSMAC NET Link and SYSMAC LINK Systems

SR 25203 turns ON when an error has occurred in data communications using SEND(90) or RECV(98) to transfer data in either a SYSMAC NET Link or SYSMAC LINK System and SR 25204 is ON when SEND(90) or RECV(98) are enable in these Systems. SR 25202 turns ON when a data link is active in operating level 0 of either of these Systems and SR 25205 turns ON with a data link is active in operating level 1. These flags and corresponding SR bits are shown below.

Bit	Flag
25202	Operating Level 0 Data Link Operating Flag
25203	SEND(90)/RECV(98) Error Flag
25204	SEND(90)/RECV(98) Enable Flag
25205	Operating Level 1 Data Link Operating Flag

SYSMAC LINK Communications Completion Code

When SEND(90) or RECV(98) is used in a SYSMAC LINK System, a completion code is output to SR 23700 through SR 23707a to indicate whether or not the data transfer was completed successfully or not and to indicate the nature of the error when communications are not completed successfully. These error codes are as follows.

Completion code	Name	Meaning
00	Normal end	Data transfer was completed successfully.
01	Parameter error	SEND(90)/RECV(98) instruction operands are not within specified ranges.
02	Transmission impossible	The System was reset during execution of the instruction or the destination node is not in the System.
03	Destination not in System	The destination node is not in the System.
04	Busy error	The destination node is busy and cannot receive the transfer.
05	Response timeout	A response was not received within the time limit.
06	Response error	An error response was received from the destination node.
07	Communications controller error	An error occurred in the communications controller.
08	Setting error	The node address was set incorrectly.
09	CPU error	A CPU error occurred in the PC of the destination node.

SYSMAC LINK/SYSMAC NET Link Data Link Status

Data link status is output to SR 238 through SR 241 for the operating-level-0 data link in the SYSMAC NET Link or SYSMAC LINK System. Although link status is always output to SR 242 through SR 245 for the SYSMAC LINK System, the status of SW3-4 on the SYSMAC NET Link Unit determines the words used for operating level 1 of the SYSMAC NET Link System, i.e., if SW3-4 is ON, SR 242 through SR 245 are used; if SW3-4 is OFF, SR 247 through SR 250 are used. In the C2000H Duplex System, however, one one SYSMAC NET Link Unit can be mounted and SR 247 through SR 250 are always used for data link status.

The meaning of each bit in these areas differs depending on whether the data link is in a SYSMAC LINK System or SYSMAC NET Link System, as shown below. Note that SR 247 through SR 250 are also used for PC Link Systems, as described in the next section.

SYSMAC LINK Systems

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

Each of the above sets of four bits operates as shown below.

Leftmost bit	Middle bits		Rightmost bit
ON when data link is active.	ON when there is a data communications error.	ON when there is a PC error.	ON when PC is in RUN mode

SYSMAC NET Link Systems

Level 0	Level 1		Bit numbers in header/data link table entry numbers in table body															
	SW3-4	SW3-4	PC Error Flags								PC Run Flags							
	ON	OFF	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
SR 238	SR 242	SR 247	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
SR 239	SR 243	SR 248	9	10	11	12	13	14	15	16	9	10	11	12	13	14	15	16
SR 240	SR 244	SR 249	17	18	19	20	21	22	23	24	17	18	19	20	21	22	23	24
SR 241	SR 245	SR 250	25	26	27	28	29	30	31	32	25	26	27	28	29	30	31	32

PC Link Systems

PC Link Unit Error and Run Flags

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

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

Single-level PC Link Systems

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

Multilevel PC Link Systems

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

Application Example

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

Bit no	15 ←- -			0
Binary	0 0 0 0	0 0 1 0	1 1 1 1	1 1 1 1
Hex	0	2	F	F

3-4-2 Data Retention Control Bit

SR bit 25212 can be turned ON to preserve the status of IR and LR bits when shifting from PROGRAM to MONITOR or RUN mode or when shifting from MONITOR or RUN mode to PROGRAM mode. If bit 25212 is OFF, IR and LR bits will be turned OFF when switching between these modes.

The status of the Data Retention Control bit is maintained for power interruptions or when PC operation is stopped.

3-4-3 Output OFF Bit

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

The status of the Output OFF bit is maintained for power interruptions or when PC operation is stopped.

3-4-4 FAL (Failure Alarm) Area

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

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

3-4-5 Low Battery Flag

SR bit 25308 turns ON if the voltage of the CPU or File Memory backup battery drops. The warning indicator on the front of the CPU will also light.

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

3-4-6 Cycle Time Error Flag

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

3-4-7 I/O Verification Error Flag

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

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

3-4-8 First Cycle Flag

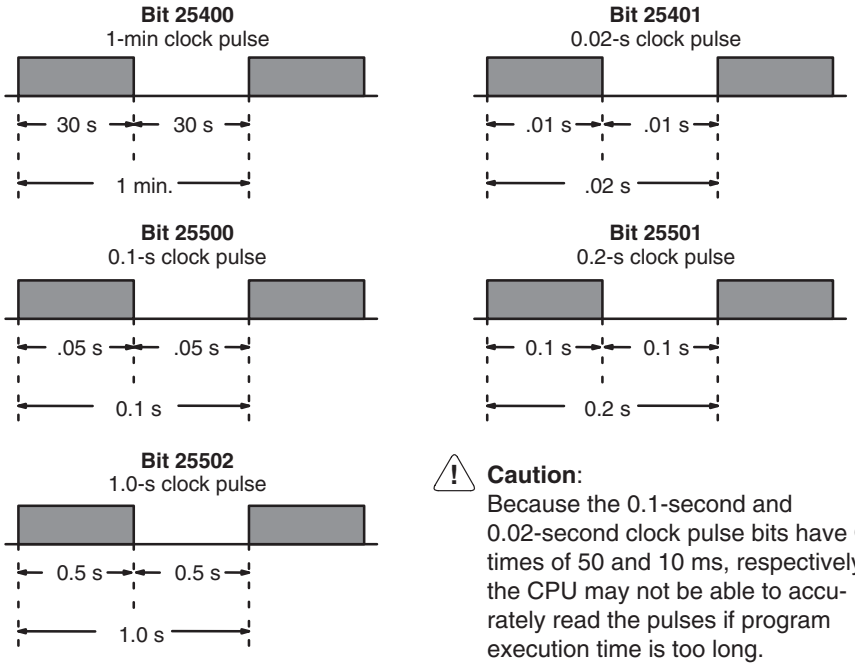
SR bit 25315 turns ON when PC operation begins and then turns OFF after one cycle of the program. The First Cycle Flag is useful in initializing counter values and other operations. An example of this is provided in *5-11 Timer and Counter Instructions*.

3-4-9 Clock Pulse Bits

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

These clock pulse bits are often used with counter instructions to create timers. Refer to *5-11 Timer and Counter Instructions* for an example of this.

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



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

3-4-10 Step Flag

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

3-4-11 Duplex System Flags

Five flags are provided in the SR area to monitor Duplex System operation. These flags are used only with a C2000H Duplex System. The following table summarizes Duplex System flags.

Name	Bit	Function
Bus Error Flag	25408	Turns ON when an error occurs in communications between the CPUs. When this flag turns ON, one of the CPUs will take over control and switch to simplex operation.
Replacement Enabled Flag	25409	Turns ON when the other CPU may be removed.
CPU Error Flag	25410	Turns ON when a CPU error has occurred in the other CPU.
Memory Error Flag	25411	Turns ON when a memory error has occurred in the other CPU.
Run Flag	25412	Stays ON during normal duplex operation.

3-4-12 Instruction Execution Error Flag, ER

SR bit 25503 turns ON if an attempt is made to execute an instruction with incorrect operand data. Common causes of an instruction error are non-BCD operand data when BCD data is required, or an indirectly addressed DM

word that is non-existent. **When the ER Flag is ON, the current instruction will not be executed.**

3-4-13 Arithmetic Flags

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

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

Refer to *5-12 Data Shifting*, *5-14 Data Comparison*, *5-16 BCD Calculations*, and *5-17 Binary Calculations* for details.

Carry Flag, CY

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

Greater Than Flag, GR

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

Equal Flag, EQ

SR bit 25506 turns ON when the result of a comparison shows two operands to be equal or when the result of an arithmetic operation is zero.

Less Than Flag, LE

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

⚠ Caution The previous four flags are cleared when END(01) is is executed.

3-5 AR (Auxiliary Relay) Area

The AR area consists of two parts. The first part, words AR 00 through AR 06 (bits AR 0000 through AR 0615), may be used by the user as work words or work bits. The rest of the AR area, words AR 07 through AR 27 (bits AR 0700 through AR 2715), is dedicated for various flags, control bits, and operating parameters.

AR 07 through AR 15 are used for the SYSMAC LINK System; AR 16 and AR 17 are used for both the SYSMAC LINK System and the SYSMAC NET Link System; and AR 19 through AR 21 are used with the File Memory Unit. When not used for their prescribed purposes, these words may be used as work bits.

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

AR Area Flags and Control Bits

Word(s)	Bit(s)	Function
07	00 to 03	Data Link setting for operating level 0 of SYSMAC LINK System
	05 to 07	Data Link setting for operating level 1 of SYSMAC LINK System
	08 to 15	Not used. May be used as work bits.
08 to 10	00 to 15	Active Node Flags for SYSMAC LINK System nodes of
11	00 to 13	operating level 0
11	14	Communications Controller Error Flag for operating level 0
	15	EEPROM Error Flag for operating level 0
12 to 14	00 to 15	Node Active Flags for SYSMAC LINK System nodes of
15	00 to 13	operating level 1
15	14	Communications Controller Error Flag for operating level 1
	15	EEPROM Error Flag for operating level 1
16	00 to 15	SYSMAC LINK/SYSMAC NET Link System operating level 0 service time per cycle
17	00 to 15	SYSMAC LINK/SYSMAC NET Link System operating level 1 service time per cycle
18	12	Trace Complete Flag
	13	Tracing Flag
	14	Trace Start Bit
	15	Sampling Start Bit
19	00	File Memory Unit Error Reset Bit
	01	FM Data Transfer Flag
	02	FM Write/Read Flag
	03	FM Blocks Different Error Flag
	04	FM Write-protected Error Flag
	05	Unsuccessful FM Write Flag
	06	FM Checksum Error Flag
	07	File Memory Unit Low Battery Flag
	08	FM Blocks 0 to 249 Write-protect Bit
	09	FM Blocks 250 to 499 Write-protect Bit
	10	FM Blocks 500 to 749 Write-protect Bit
	11	FM Blocks 750 to 999 Write-protect Bit
	12	FM Blocks 1,000 to 1,249 Write-protect Bit
	13	FM Blocks 1,250 to 1,499 Write-protect Bit
	14	FM Blocks 1,500 to 1,749 Write-protect Bit
	15	FM Blocks 1,750 to 1,999 Write-protect Bit
20	00 to 15	FM Blocks Counter
21	00 to 15	Remaining FM Blocks Counter
22	00 to 11	On-line Removal First Word Indicator
	12 to 14	Number of Words Indicator for On-line Removal
	15	On-line Removal Flag
23	00 to 15	Power-Off Counter

Word(s)	Bit(s)	Function
24	00 to 03	Leftmost digit of FALS-generating address (AR 25 contains the other four digits)
	04 and 05	Not used and not accessible by user.
	06	Level 1 Network Parameter Flag
	07	Level 0 Network Parameter Flag
	08 to 10	Not used and not accessible by user.
	11	PC Link Unit Level 1 Mounted Flag
	12	PC Link Unit Level 0 or Single-level PC Link Unit Mounted Flag
	13	SYSMAC NET Link Unit Mounted Flag
	14	Rack-mounting Host Link Unit Mounted Flag
	15	CPU-mounting Device Flag
25	00 to 15	Rightmost four digits of FALS-generating address (AR 2400 to AR 2403 contain the fifth digit)
26	00 to 15	Maximum cycle time
27	00 to 15	Present cycle time

3-5-1 SYSMAC LINK System Data Link Settings

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

External/Automatic Allocation

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

Words per Node

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

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

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

3-5-2 Active Node Flags

AR 08 through AR 11 and AR 12 through AR 15 provide flags that indicate which nodes are active in the SYSMAC LINK System at the current time.

These flags are refreshed every cycle while the SYSMAC LINK System is operating.

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

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

*Communication Controller Error Flag
**EEPROM Error Flag

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

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

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

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

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

3-5-4 Tracing Flags and Control Bits

AR 1812 through AR 1815 are used with trace operations, which are described in *5-23 Data Tracing – TRSM(45)*. AR 1812 and AR 1813 are refreshed every cycle.

Sampling begins on the rising edge of the Sampling Start Bit, AR 1815. Actual tracing begins on the rising edge of the Trace Start Bit, AR 1814. The Tracing Flag, AR 1813, turns ON then tracing has started. When tracing has completed, the Tracing Flag turns OFF and the Trace Complete Flag, AR 1812, turns ON.

3-5-5 File Memory Flags and Control Bits

AR 19 through AR 21 are used with File Memory operations, which are described in *5-24 File Memory Instructions*.

Flags and Reset Bit

The following table describes the flags and reset bits available for FM operations. The actual use of these is described in more detail in *5-24 File Memory Instructions*. AR 1903 through AR 1907 are refreshed every cycle when a File Memory Unit is mounted.

Name	Bit	Function
File Memory Unit Error Reset Bit	AR 1900	Turned ON from program to reset AR 1903 through AR 1906.
FM Data Transfer Flag	AR 1901	ON while FM data is being transferred.
FM Write/Read Flag	AR 1902	ON during transfers to FM; OFF during transfers from FM.
FM Blocks Different Error Flag	AR 1903	Turns ON if attempt is made to read different types of FM block.
FM write-protected Error Flag	AR 1904	Turns ON if attempt is made to write to a write-protected block.
Unsuccessful FM Write Flag	AR 1905	Turns ON if a write operation to FM ends unsuccessfully.
FM Checksum Error Flag	AR 1906	Turns ON if a checksum error occurs when reading from FM.
File Memory Unit Low Battery Flag	AR 1907	Turns ON if the File Memory Unit battery voltage drops.

Write-protect Bits

FM data can be write-protected in units of 250 blocks. To protect any part of FM from being overwritten, turn ON the Write-protect bits of the blocks to be protected. These are software protects. There are also switch settings on the File Memory Unit that can be set to achieve the same purpose.

Bit	Blocks
AR 1908	0 to 249
AR 1909	250 to 499
AR 1910	500 to 749
AR 1911	750 to 999
AR 1912	1000 to 1249
AR 1913	1250 to 1499
AR 1914	1500 to 1749
AR 1915	1750 to 1999

Block Counters

The FM Blocks Counter (AR 20) indicates the number of the block that is currently being transferred. The Remaining FM Blocks Counter (AR 21) indicates the number of blocks remaining to be transferred. Both of these counters provide data in 4-digit BCD and are refreshed each time transfer of a block is completed.

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

3-5-6 On-line Removal Bits

AR 22 indicates when a Unit is being removed on-line and what words are allocated to the Unit. AR 22 is refreshed every cycle while a Unit is being removed on-line.

The On-line Removal Flag, AR 2215 turns ON when an on-line removal operation is in progress. The On-line Removal First Word Indicator, AR 2200 through AR 2211, indicates the first word allocated to the Unit that is being removed. This value is in 3-digit BCD.

The Number of Words Indicator, AR 2212 through AR 2214, indicates in binary the number of words allocated to the Unit that is being removed.

3-5-7 Power-off Counter

AR 23 provides in 4-digit BCD the number of times that the PC power has been turned off. This counter can be reset as necessary using the PV

Change 1 operation from the Programming Console. (Refer to *7-6-3 Hex/ BCD Data Modification* for details.) The Power-Off Counter is refreshed every time power is turned on.

3-5-8 Network Parameter Flags

AR 2406 is ON when the actual setting of the network parameter for operating level 1 of the SYSMAC LINK System differs from the setting at the FIT.

AR 2407 is ON when the actual setting of the network parameter for operating level 0 of the SYSMAC LINK System differs from the setting at the FIT.

3-5-9 Link Unit Mounted Flags

The following flags indicate when the specified Link Units are mounted to the Racks. (Refer to *3-5-7 CPU-mounting Device Flag* for CPU-mounting Host Link Units.) These flags are refreshed every cycle.

Name	Bit	Link Unit
PC Link Unit Level 1 Mounted Flag	AR 2411	PC Link Unit in operating level 1
PC Link Unit Level 0 Mounted Flag	AR 2412	PC Link Unit in operating level 0 or in Single-level System
SYSMAC NET Link Unit Mounted Flag	AR 2413	SYSMAC NET Link Unit
Rack-mounting Host Link Unit	AR 2414	Rack-mounting Host Link Unit

3-5-10 CPU-mounting Device Flag

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

3-5-11 FALS-generating Address

AR 2400 to AR 2403 and AR 25 contain the address generating a user-programmed FALS code or a system FALS code 9F (cycle time error). The address is in 5-digit BCD with the leftmost digit given in AR 2400 to AR 2403 and the rightmost four digits given in AR 25. FALS codes are described in *5-22-1 Failure Alarm – FAL(06) and Severe Failure Alarm – FALS(07)*. The address is refreshed every cycle when an FALS code has been generated.

3-5-12 Cycle Time Indicators

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

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

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

3-6 DM (Data Memory) Area

The DM area is used for internal data storage and manipulation and is accessible only by word. Addresses range from DM 0000 through DM 4095 for the C1000H; from DM 0000 through DM 6655 for the C2000H.

Although composed of 16 bits just like any other word in memory, DM words cannot be specified by bit for use in instructions with bit-size operands, such

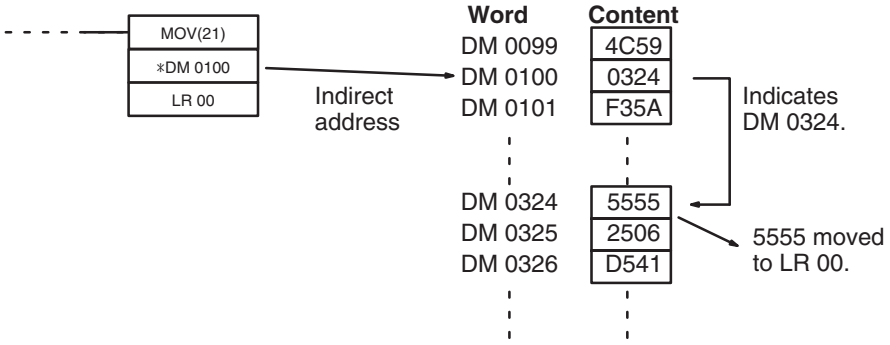
as LD, OUT, AND, and OR, nor can DM words be used with the SHIFT instruction.

The DM area retains status during power interruptions.

Indirect Addressing

Normally, when the content of a data area word is specified for an instruction, the instruction is performed directly on the content of that word. For example, suppose CMP(20) (COMPARE), with IR 005 as the first operand and DM 0010 as the second operand, is used in the program. When this instruction is executed, the content of IR 005 is compared with that of DM 0010.

It is also possible, however, to use indirect DM addresses as operands for instructions. If ∗DM 0100 is specified as the data for a programming instruction, the asterisk in front of DM indicates that it is an indirect address that specifies another DM word which contains the actual operand data. If, in this case, the content of DM 0100 is 0324, then ∗DM 0100 indicates DM 0324 as the word that contains the desired data, and the content of DM 0324 is used as the operand in the instruction. The following example shows this type of indirect addressing with the MOVE instruction (MOV(21)).



3-7 HR (Holding Relay) Area

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

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

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

When a SYSMAC LINK System is used, a certain number of HR bits is required for a routing table and monitor timer. These bits are taken from between HR 00 to HR 42. Refer to the *SYSMAC LINK System Manual* for details.

3-8 TC (Timer/Counter) Area

The TC area is used to create and program timers and counters and holds the Completion flags, set values (SV), and present values (PV) for all timers and counters. All of these are accessed through TC numbers ranging from TC 000 through TC 511. Each TC number is defined as either a timer or counter using one of the following instructions: TIM, TIMH, CNT, CNTR(12),

TIMW<13>, TMHW<15>, or CNTW<14>. No prefix is required when using a TC number as a definer in a timer or counter instruction.

Once a TC number has been defined using one of these instructions, it cannot be redefined elsewhere in the program either using the same or a different instruction. If the same TC number is defined in more than one of these instructions or in the same instruction twice, an error will be generated during the program check. There are no restrictions on the order in which TC numbers can be used.

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

TC numbers can be designated for operands that require bit data or for operands that require word data. When designated as an operand that requires bit data, the TC number accesses the completion flag of the timer or counter. When designated as an operand that requires word data, the TC number accesses a memory location that holds the PV of the timer or counter.

TC numbers are also used to access the SV of timers and counters from a Programming Device. The procedures for doing so using the Programming Console are provided in *7-6 Monitoring Operation and Modifying Data.*

The TC area retains the SVs of both timers and counters during power interruptions. The PVs of timers are reset when PC operation is begun and when reset in interlocked program sections. Refer to *5-7 Interlock and Interlock Clear – IL(02) and ILC(03)* for details on timer and counter operation in interlocked program sections. The PVs of counters are not reset at these times.

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

3-9 LR (Link Relay) Area

The LR area is used as a common data area to transfer information between PCs. This data transfer is achieved through a PC Link System, a SYSMAC LINK System, or a SYSMAC NET Link System. Certain words will be allocated as the write words of each PC. These words are written by the PC and automatically transferred to the same LR words in the other PCs in the System. The write words of the other PCs are transferred in as read words so that each PC can access the data written by the other PCs in the PC Link System. Only the write words allocated to the particular PC will be available for writing; all other words may be read only. Refer to the *PC Link System Manual*, *SYSMAC LINK System Manual*, or *SYSMAC NET Link System Manual* for details.

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

LR area data is not retained when the power is interrupted, when the PC is changed to PROGRAM mode, or when it is reset in an interlocked program

section. Refer to *5-7 Interlock and Interlock Clear – IL(02) and ILC(03)* for details on interlocks.

3-10 Program Memory

Program Memory is where the user program is stored. The amount of Program Memory available is either 8K,16K, 24K or 32K words, depending on the type of Memory Unit mounted to the CPU. These memory capacities correspond to approximately 7.7K, 15.4K, 23.1K and 30.8K instructions, but the actual number of instructions will vary depending on the instructions actually used.

Memory Units come in different types, such as RAM and ROM Units, and for each type there are different sizes. (Refer to the *Installation Guide* for details.)

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

3-11 File Memory

The File Memory (FM) is available only when a File Memory Unit is mounted to the PC. This area, contained in RAM within the File Memory Unit, can be used for data storage and retrieval. Program Memory and/or IR, SR, DM, LR, HR, AR, and TC area data can be stored in or retrieved from FM.

The area is accessible in block units only. The block numbers are 4-digit BCD, and each block consists of 128 words.

File Memory addresses are by block and range from 0000 through 0999 or from 0000 through 1999, depending on the model of File Memory Unit that is used. Refer to *Appendix A Standard Models* for File Memory Unit models and to *5-24 File Memory Instructions* for instructions to transfer data to and from File Memory.

The File Memory retains data during power interruptions or when PC operation is stopped.

3-12 Trace Memory

The Trace Memory is used to store the results of execution traces. Trace operations are explained in *5-23 Trace Operations*.

3-13 TR (Temporary Relay) Area

The TR area provides eight bits that are used only with the LD and OUT instructions to enable certain types of branching ladder diagram programming. The use of TR bits is described in *Section 4 Writing and Inputting the Program*.

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

SECTION 4
Writing and Inputting the Program

This section explains the basic steps and concepts involved in writing a basic ladder diagram program, inputting the program into memory, and executing it. It introduces the instructions that are used to build the basic structure of the ladder diagram and control its execution. The entire set of instructions used in programming is described in *Section 5 Instruction Set*.

4-1 Basic Procedure 42
4-2 Instruction Terminology 42
4-3 Basic Ladder Diagrams 43
4-3-1 Basic Terms 43
4-3-2 Mnemonic Code 44
4-3-3 Ladder Instructions 45
4-3-4 OUTPUT and OUTPUT NOT 48
4-3-5 The END Instruction 48
4-3-6 Logic Block Instructions 49
4-3-7 Coding Multiple Right-hand Instructions 57
4-4 The Programming Console 58
4-4-1 The Keyboard 58
4-4-2 PC Modes 59
4-4-3 The Display Message Switch 61
4-5 Preparation for Operation 61
4-5-1 Entering the Password 61
4-5-2 Clearing Memory 62
4-5-3 Registering the I/O Table 64
4-5-4 Clearing Error Messages 66
4-5-5 Transferring the I/O Table 67
4-5-6 Changing the I/O Table 67
4-5-7 Changing I/O Units On-line (C2000H Only) 69
4-5-8 Verifying the I/O Table 70
4-5-9 Reading the I/O Table 71
4-6 Inputting, Modifying, and Checking the Program 75
4-6-1 Setting and Reading from Program Memory Address 75
4-6-2 Entering or Editing Programs 76
4-6-3 Checking the Program 79
4-6-4 Displaying the Cycle Time 81
4-6-5 Program Searches 82
4-6-6 Inserting and Deleting Instructions 83
4-6-7 Branching Instruction Lines 86
4-6-8 Jumps 90
4-7 Controlling Bit Status 92
4-7-1 DIFFERENTIATE UP and DIFFERENTIATE DOWN 92
4-7-2 KEEP 92
4-7-3 Self-maintaining Bits (Seal) 93
4-8 Work Bits (Internal Relays) 93
4-9 Programming Precautions 95
4-10 Program Execution 97

4-1 Basic Procedure

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

1, 2, 3...

1. Obtain a list of all I/O devices and the I/O points that have been assigned to them and prepare a table that shows the I/O bit allocated to each I/O device.
2. If the PC has any Units that are allocated words in data areas other than the IR area or are allocated IR words in which the function of each bit is specified by the Unit, prepare similar tables to show what words are used for which Units and what function is served by each bit within the words. These Units include Special I/O Units and Link Units.
3. Determine what words are available for work bits and prepare a table in which you can allocate these as you use them.
4. Also prepare tables of TC numbers and jump numbers so that you can allocate these as you use them. Remember, the function of a TC number can be defined only once within the program; jump numbers 01 through 99 can be used only once each. (TC number are described in *5-11 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.

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 8 Troubleshooting* also provides information required for debugging.

4-2 Instruction Terminology

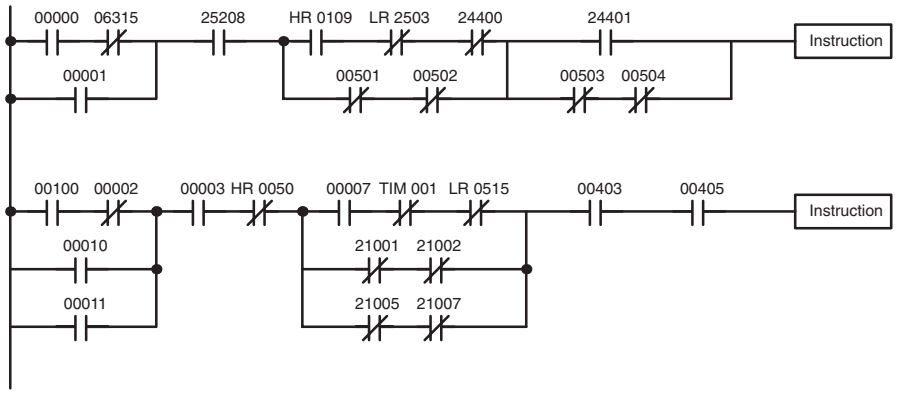
There are basically two types of instructions used in ladder-diagram programming: instructions that correspond to the conditions on the ladder diagram and are used in instruction form only when converting a program to mnemonic code and instructions that are used on the right side of the ladder diagram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

Most instructions have at least one or more operands associated with them. Operands indicate or provide the data on which an instruction is to be performed. These are sometimes input as the actual numeric values, but are usually the addresses of data area words or bits that contain the data to be used. For instance, a MOVE instruction that has IR 000 designated as the source operand will move the contents of IR 000 to some other location. The other location is also designated as an operand. A bit whose address is designated as an operand is called an operand bit; a word whose address is designated as an operand is called an operand word. If the actual value is entered as a constant, it is preceded by # to indicate that it is not an address.

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

4-3 Basic Ladder Diagrams

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



As shown in the diagram above, instruction lines can branch apart and they can join back together. The vertical pairs of lines are called conditions. Conditions without diagonal lines through them are called normally open conditions and correspond to a LOAD, AND, or OR instruction. The conditions with diagonal lines through them are called normally closed conditions and correspond to a LOAD NOT, AND NOT, or OR NOT instruction. The number above each condition indicates the operand bit for the instruction. It is the status of the bit associated with each condition that determines the execution condition for following instructions. The way the operation of each of the instructions corresponds to a condition is described below. Before we consider these, however, there are some basic terms that must be explained.

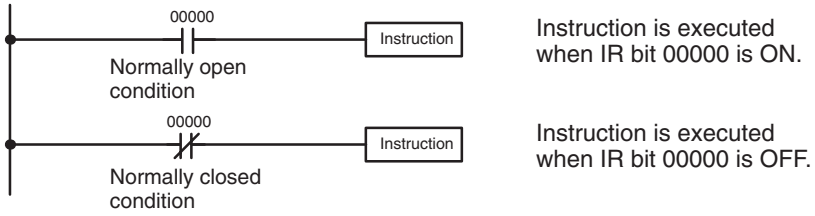
Note When displaying ladder diagrams with a GPC, a FIT, or LSS, a second bus bar will be shown on the right side of the ladder diagram and will be connected to all instructions on the right side. This does not change the ladder-diagram program in any functional sense. No conditions can be placed between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

4-3-1 Basic Terms

Normally Open and Normally Closed Conditions

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

something to happen when a bit is ON, and a normally closed condition when you want something to happen when a bit is OFF.



Execution Conditions

In ladder diagram programming, the logical combination of ON and OFF conditions before an instruction determines the compound condition under which the instruction is executed. This condition, which is either ON or OFF, is called the execution condition for the instruction. All instructions other than LOAD instructions have execution conditions.

Operand Bits

The operands designated for any of the ladder instructions can be any bit in the IR, SR, HR, AR, LR, or TC areas. This means that the conditions in a ladder diagram can be determined by I/O bits, flags, work bits, timers/counters, etc. LOAD and OUTPUT instructions can also use TR area bits, but they do so only in special applications. Refer to *4-6-7 Branching Instruction Lines* for details.

Logic Blocks

The way that conditions correspond to what instructions is determined by the relationship between the conditions within the instruction lines that connect them. Any group of conditions that go together to create a logic result is called a logic block. Although ladder diagrams can be written without actually analyzing individual logic blocks, understanding logic blocks is necessary for efficient programming and is essential when programs are to be input in mnemonic code.

4-3-2 Mnemonic Code

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

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

Program Memory Structure

The program is input into addresses in Program Memory. Addresses in Program Memory are slightly different to those in other memory areas because each address does not necessarily hold the same amount of data. Rather, each address holds one instruction and all of the definers and operands (described in more detail later) required for that instruction. Because some instructions require no operands, while others require up to three operands, Program Memory addresses can be from one to four words long.

Program Memory addresses start at 00000 and run until the capacity of Program Memory has been exhausted. The first word at each address defines the instruction. Any definers used by the instruction are also contained in the first word. Also, if an instruction requires only a single bit operand (with no definer), the bit operand is also programmed on the same line as the instruction. The rest of the words required by an instruction contain the operands that specify what data is to be used. When converting to mnemonic code, all but ladder diagram instructions are written in the same form, one word to a line, just as they appear in the ladder diagram symbols. An example of mnemonic code is shown below. The instructions used in it are described later in the manual.

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

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

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

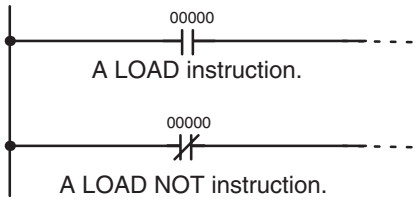
4-3-3 Ladder Instructions

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

LOAD and LOAD NOT

The first condition that starts any logic block within a ladder diagram corresponds to a LOAD or LOAD NOT instruction. Each of these instruction requires one line of mnemonic code. “Instruction” is used as a dummy instruc-

tion in the following examples and could be any of the right-hand instructions described later in this manual.

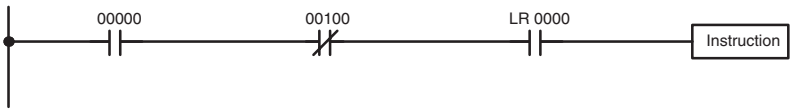


Address	Instruction	Operands
00000	LD	00000
00001	Instruction	
00002	LD NOT	00000
00003	Instruction	

When this is the only condition on the instruction line, the execution condition for the instruction at the right is ON when the condition is ON. For the LOAD instruction (i.e., a normally open condition), the execution condition would be ON when IR 00000 was ON; for the LOAD NOT instruction (i.e., a normally closed condition), it would be ON when 00000 was OFF.

AND and AND NOT

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



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

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

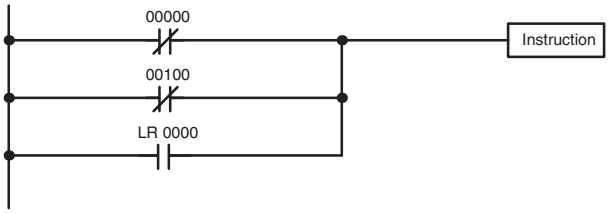
AND instructions in series can be considered individually, with each taking the logical AND of the execution condition (i.e., the total of all conditions up to that point) and the status of the AND instruction's operand bit. If both of these are ON, an ON execution condition will be produced for the next instruction. If either is OFF, the result will also be OFF. The execution condition for the first AND instruction in a series is the first condition on the instruction line.

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

OR and OR NOT

When two or more conditions lie on separate instruction lines running in parallel and then joining together, the first condition corresponds to a LOAD or LOAD NOT instruction; the rest of the conditions correspond to OR or OR NOT instructions. The following example shows three conditions which corre-

spond in order from the top to a LOAD NOT, an OR NOT, and an OR instruction. Again, each of these instructions requires one line of mnemonic code.



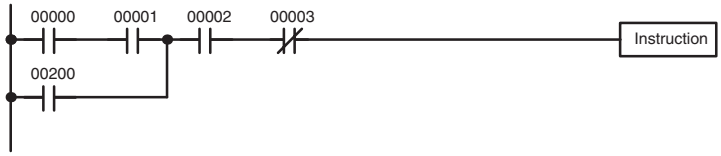
Address	Instruction	Operands
00000	LD	00000
00001	OR NOT	00100
00002	OR	LR 0000
00003	Instruction	

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

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

Combining AND and OR Instructions

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



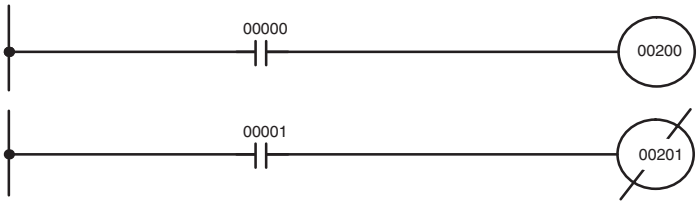
Address	Instruction	Operands
00000	LD	00000
00001	AND	00001
00002	OR	00200
00003	AND	00002
00004	AND NOT	00003
00005	Instruction	

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

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

4-3-4 OUTPUT and OUTPUT NOT

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



Address	Instruction	Operands
00000	LD	00000
00001	OUT	00200

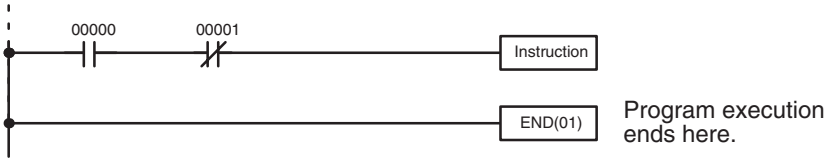
Address	Instruction	Operands
00000	LD	00001
00001	OUT NOT	00201

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

The length of time that a bit is ON or OFF can be controlled by combining the OUTPUT or OUTPUT NOT instruction with Timer instructions. Refer to Examples under *5-11-1 Timer – TIM* for details.

4-3-5 The END Instruction

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



Address	Instruction	Operands
00500	LD	00000
00501	AND NOT	00001
00502	Instruction	
00503	END(01)	---

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

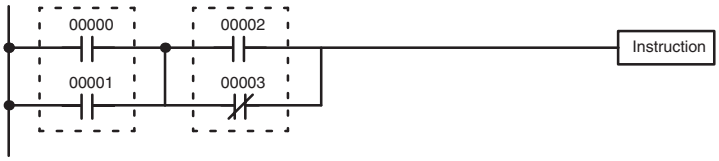
Now you have all of the instructions required to write simple input-output programs. Before we finish with ladder diagram basic and go onto inputting the program into the PC, let's look at logic block instruction (AND LOAD and OR LOAD), which are sometimes necessary even with simple diagrams.

4-3-6 Logic Block Instructions

Logic block instructions do not correspond to specific conditions on the ladder diagram; rather, they describe relationships between logic blocks. The AND LOAD instruction logically ANDs the execution conditions produced by two logic blocks. The OR LOAD instruction logically ORs the execution conditions produced by two logic blocks.

AND LOAD

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



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

The two logic blocks are indicated by dotted lines. Studying this example shows that an ON execution condition will be produced when: either of the conditions in the left logic block is ON (i.e., when either IR 00000 or IR 00001 is ON), **and** when either of the conditions in the right logic block is ON (i.e., when either IR 00002 is ON or IR 00003 is OFF).

The above ladder diagram cannot, however, be converted to mnemonic code using AND and OR instructions alone. If an AND between IR 00002 and the results of an OR between IR 00000 and IR 00001 is attempted, the OR NOT between IR 00002 and IR 00003 is lost and the OR NOT ends up being an OR NOT between just IR 00003 and the result of an AND between IR 00002 and the first OR. What we need is a way to do the OR (NOT)'s independently and then combine the results.

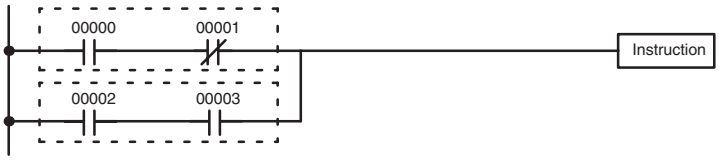
To do this, we can use the LOAD or LOAD NOT instruction in the middle of an instruction line. When LOAD or LOAD NOT is executed in this way, the current execution condition is saved in special buffers and the logic process is begun over. To combine the results of the current execution condition with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

Analyzing the above ladder diagram in terms of mnemonic instructions, the condition for IR 00000 is a LOAD instruction and the condition below it is an OR instruction between the status of IR 00000 and that of IR 00001. The condition at IR 00002 is another LOAD instruction and the condition below is an OR NOT instruction, i.e., an OR between the status of IR 00002 and the inverse of the status of IR 00003. To arrive at the execution condition for the

instruction at the right, the logical AND of the execution conditions resulting from these two blocks would have to be taken. AND LOAD does this. The mnemonic code for the ladder diagram is shown below. The AND LOAD instruction requires no operands of its own, because it operates on previously determined execution conditions. Here too, dashes are used to indicate that no operands needs designated or input.

OR LOAD

The following diagram requires an OR LOAD instruction between the top logic block and the bottom logic block. An ON execution condition would be produced for the instruction at the right either when IR 00000 is ON and IR 00001 is OFF or when IR 00002 and IR 00003 are both ON. The operation of and mnemonic code for the OR LOAD instruction is exactly the same as those for a AND LOAD instruction except that the current execution condition is ORed with the last unused execution condition.



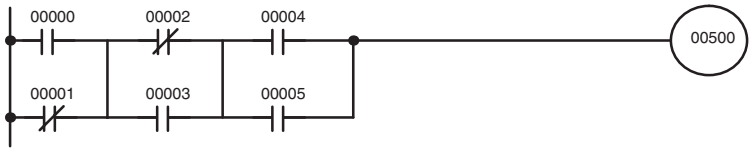
Address	Instruction	Operands
00000	LD	00000
00001	AND NOT	00001
00002	LD	00002
00003	AND	00003
00004	OR LD	---

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

Logic Block Instructions in Series

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

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

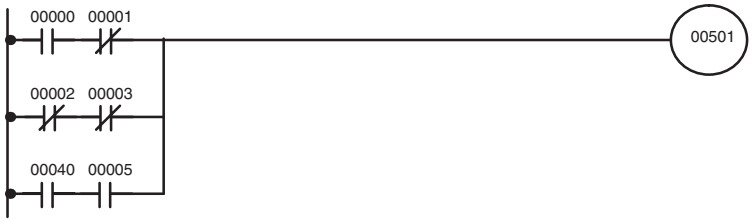


Address	Instruction	Operands
00000	LD	00000
00001	OR NOT	00001
00002	LD NOT	00002
00003	OR	00003
00004	AND LD	—
00005	LD	00004
00006	OR	00005
00007	AND LD	—
00008	OUT	00500

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

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

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



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

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

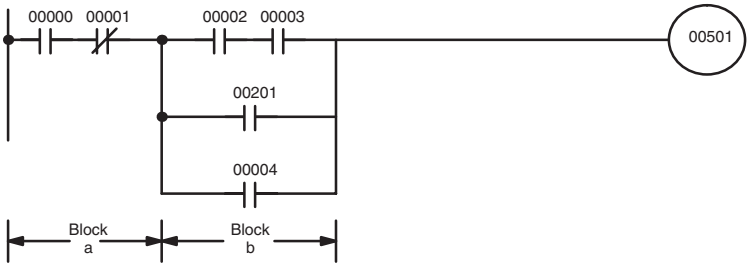
Address	Instruction	Operands
00000	LD	00000
00001	AND NOT	00001
00002	LD NOT	00002
00003	AND NOT	00003
00004	LD	00004
00005	AND	00005
00006	OR LD	—
00007	OR LD	—
00008	OUT	00501

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

Combining AND LOAD and OR LOAD

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

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

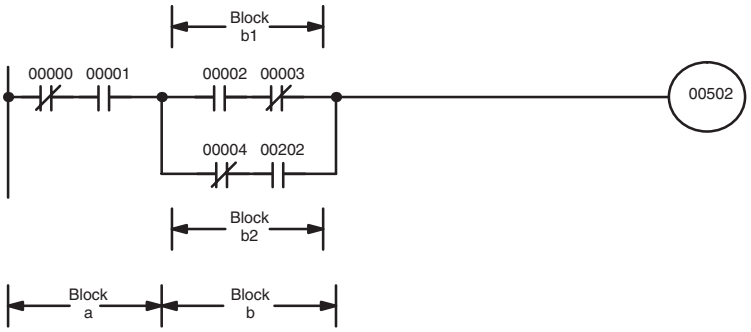


Address	Instruction	Operands
00000	LD	00000
00001	AND NOT	00001
00002	LD	00002
00003	AND	00003
00004	OR	00201
00005	OR	00004
00006	AND LD	—
00007	OUT	00501

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

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

logic block instruction and the execution condition of the logic block third from the end, and on back to the first logic block that is being combined.



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

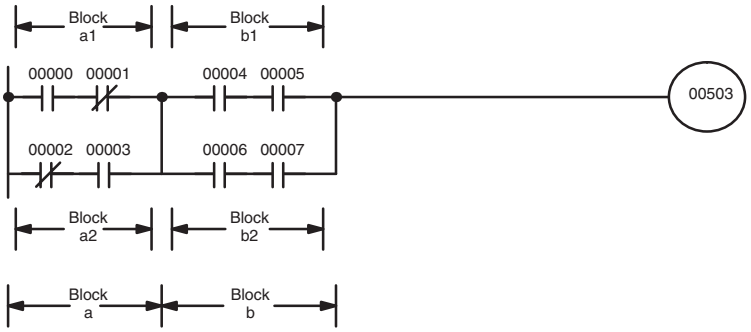
Complicated Diagrams

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

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

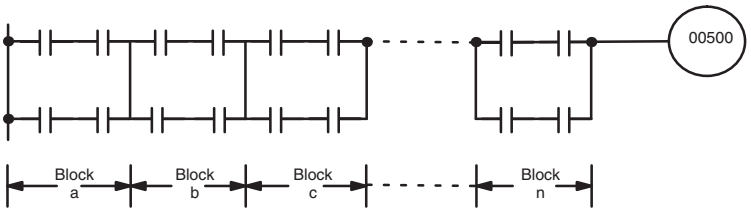
The following diagram must be broken down into two blocks and each of these then broken into two blocks before it can be coded. As shown below, blocks a and b require an AND LOAD. Before AND LOAD can be used, how-

ever, OR LOAD must be used to combine the top and bottom blocks on both sides, i.e., to combine a1 and a2; b1 and b2.

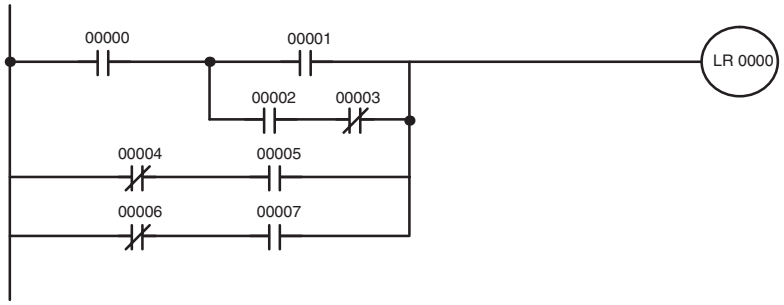


Address	Instruction	Operands	
00000	LD	00000	
00001	AND NOT	00001	
00002	LD NOT	00002	
00003	AND	00003	
00004	OR LD	—	Blocks a1 and a2
00005	LD	00004	
00006	AND	00005	
00007	LD	00006	
00008	AND	00007	
00009	OR LD	—	Blocks b1 and b2
00010	AND LD	—	Blocks a and b
00011	OUT	00503	

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

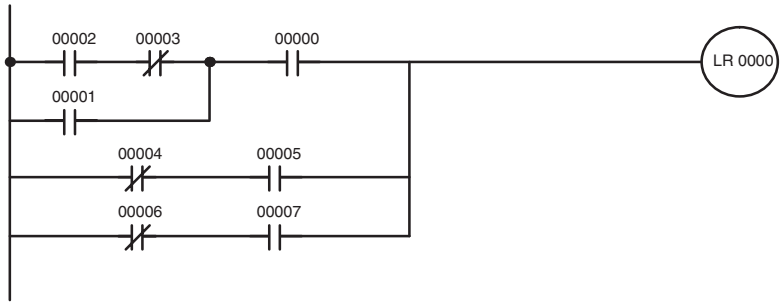


The following diagram requires an OR LOAD followed by an AND LOAD to code the top of the three blocks, and then two more OR LOADs to complete the mnemonic code.



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

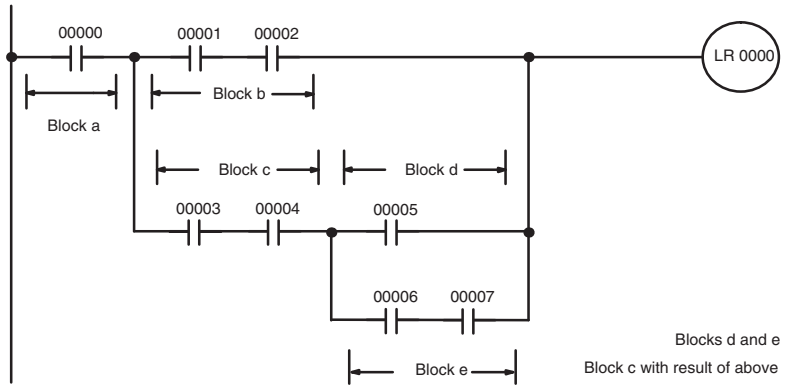
Although the program will execute as written, this diagram could be drawn as shown below to eliminate the need for the first OR LOAD and the AND LOAD, simplifying the program and saving memory space.



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

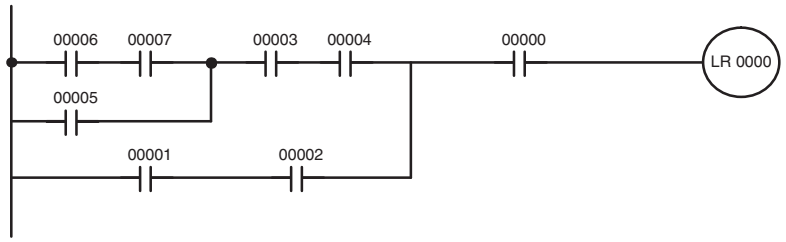
The following diagram requires five blocks, which here are coded in order before using OR LOAD and AND LOAD to combine them starting from the last two blocks and working backward. The OR LOAD at program address

00008 combines blocks blocks d and e, the following AND LOAD combines the resulting execution condition with that of block c, etc.



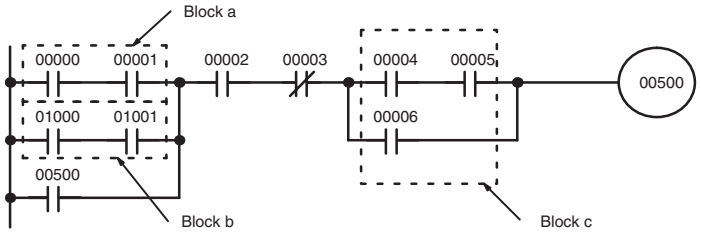
Address	Instruction	Operands
00000	LD	00000
00001	LD	00001
00002	AND	00002
00003	LD	00003
00004	AND	00004
00005	LD	00005
00006	LD	00006
00007	AND	00007
00008	OR LD	--
00009	AND LD	--
00010	OR LD	--
00011	AND LD	--
00012	OUT	LR 0000

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



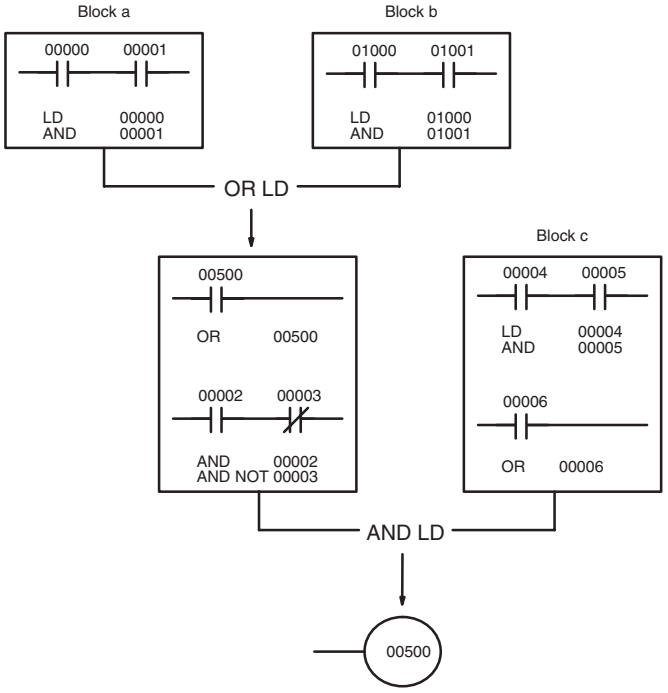
Address	Instruction	Operands
00000	LD	00006
00001	AND	00007
00002	OR	00005
00003	AND	00003
00004	AND	00004
00005	LD	00001
00006	AND	00002
00007	OR LD	--
00008	AND	00000
00009	OUT	LR 0000

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



The first logic block instruction is used to combine the execution conditions resulting from blocks a and b, and the second one is to combine the execu-

tion condition of block c with the execution condition resulting from the normally closed condition assigned IR 00003. The rest of the diagram can be coded with OR, AND, and AND NOT instructions. The logical flow for this and the resulting code are shown below.

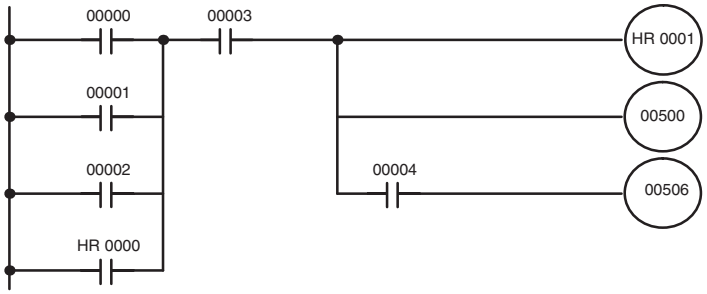


Address	Instruction	Operands
00000	LD	00000
00001	AND	00001
00002	LD	01000
00003	AND	01001
00004	OR LD	--
00005	OR	00500
00006	AND	00002
00007	AND NOT	00003
00008	LD	00004
00009	AND	00005
00010	OR	00006
00011	AND LD	--
00012	OUT	00500

4-3-7 Coding Multiple Right-hand Instructions

If there is more than one right-hand instruction executed with the same execution condition, they are coded consecutively following the last condition on

the instruction line. In the following example, the last instruction line contains one more condition that corresponds to an AND with IR 00004.



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

4-4 The Programming Console

Once a program has been written, it must be input into the PC. This can be done in graphic (ladder diagram) form using a GPC,a FIT, or LSS. The most common way of inputting a program, however, is through a Programming Console using mnemonic code. This and the next section describe the Programming Console and the operation necessary to prepare for program input. *4-6 Inputting, Modifying, and Checking the Program* describes 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. If you are connecting to a C2000H Duplex System, you can connect to either the active or passive CPU, but writing operations to memory areas (including program writing to Program Memory) and changes to data in memory areas will not be possible unless connected to the active CPU.

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.

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

4-4-2 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 mode it will be in is affected by any peripheral device connected or mounted to the CPU, as follows:

1, 2, 3...

1. No Peripheral Device Connected
   When power is applied to the PC without a Peripheral Device connected, the PC is automatically set to RUN mode. Program execution is then controlled through the CPU Power Supply Unit's START terminal.
2. Programming Console Connected
   If the Programming Console is connected to the PC when PC power is applied, the PC is set to the mode set on the Programming Console's mode switch.
3. Other Peripheral Connected
   If a Peripheral Interface Unit, a PROM Writer, a Printer Interface Unit, or a Floppy Disk Interface Unit is attached to the PC when PC power is turned on, the PC is automatically set to PROGRAM mode.

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. If it is necessary to have the PC in PROGRAM mode, (for the PROM Writer, Floppy Disk Interface Unit, etc.), be sure to select this mode before connecting the peripheral device; or, alternatively, apply power to the PC after the peripheral device is connected.

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

⚠ Caution 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. If the START input on the CPU Power Supply Unit is ON and there is no device connected to the CPU, ensure that commencing operation is safe and appropriate before turning on the PC.

4-4-3 The Display Message Switch

Next to the external connector for peripheral devices on the PC there is a small switch for selecting either Japanese or English language messages for display on the Programming Console. It is factory set to OFF, which causes English language messages to be displayed.

4-5 Preparation for Operation

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

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

1, 2, 3...

1. Confirm that all wiring for the PC has been installed and checked properly.
2. Confirm that a RAM Unit is mounted as the Memory Unit and that the write-protect switch is OFF.
3. Connect the Programming Console to the PC. Make sure that the Programming Console is securely connected or mounted to the CPU; improper connection may inhibit operation.
4. Set the mode switch to PROGRAM mode.
5. Turn on PC power.
6. Enter the password.
7. Clear memory.
8. Register the I/O table.
9. Check the I/O table until the I/O table and system configuration are correct and in agreement.

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.

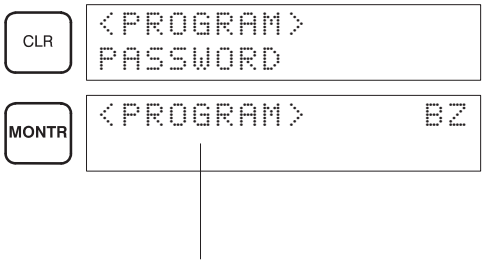
4-5-1 Entering the Password

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 en-

tered, 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

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

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

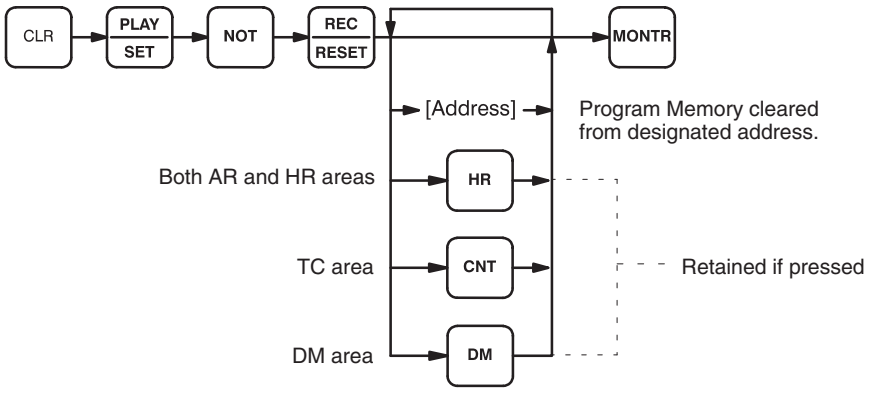
4-5-2 Clearing Memory

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

Before beginning to programming for the first time or when installing a new program, all areas should normally be cleared. Before clearing memory, check to see if a program is already loaded that you need. If you need the program, clear only the memory areas that you do not need, and be sure to check the existing program with the program check key sequence before using it. The check sequence is provided later in this section. Further debugging methods are provided in *Section 7 Program 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.

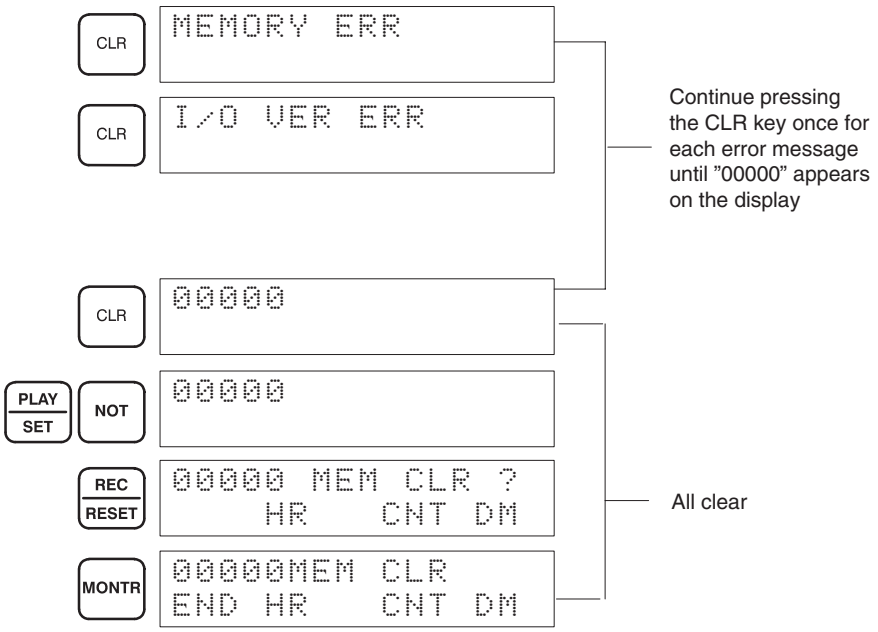
Memory can be cleared in PROGRAM mode only.

Key Sequence



All Clear

The following procedure is used to clear memory completely.

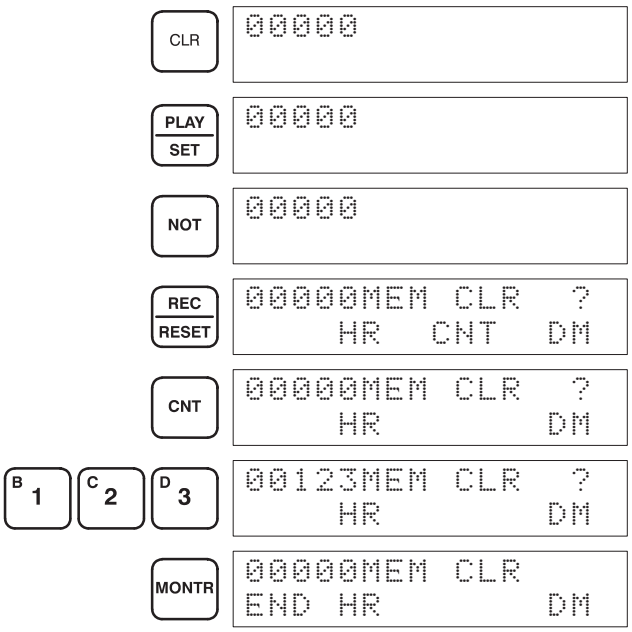


Partial Clear

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:



4-5-3 Registering the I/O Table

The I/O Table Registration operation writes the types of I/O Units controlled by the PC and the Rack locations of the I/O Units into the I/O table memory area of the CPU (see *Section 3-2 I/R Area*). It also clears all I/O bits. The I/O table must be registered before programming operations are begun. As the I/O table remains in memory, a new I/O table must also be registered whenever I/O Units are changed.

Note If "I/OTBL WRIT DISABLED" is displayed, the I/O table cannot be written. Check the number of Remote I/O Units, duplicate word settings, for Optical I/O Units, terminators for Remote I/O Systems, or an excess of I/O units. Check all I/O Units.

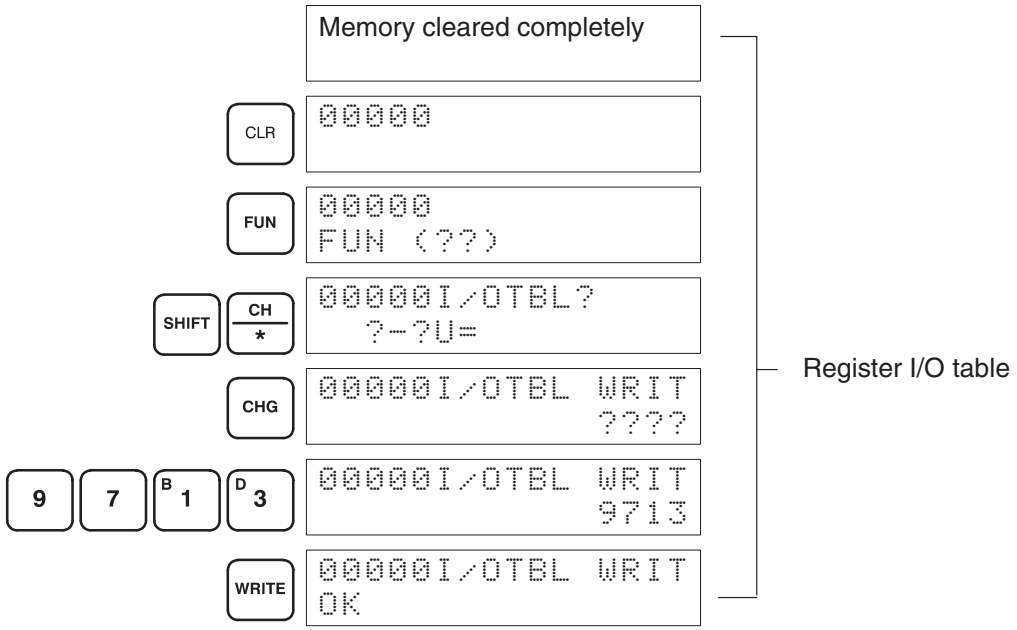
When Remote I/O Master Units connected to I/O Link Units, Optical I/O Units, Remote Terminals, or I/O Terminals are included in the System, word multipliers (see below) must be registered for the Masters to enable word allocation.

I/O Table Registration can be performed only in PROGRAM mode.

The I/O verification error message, "I/O VER ERR", will appear when starting programming operations or after I/O Units have been changed. This error is cleared by registering a new I/O table.

Key Sequence

Initial I/O Table Registration



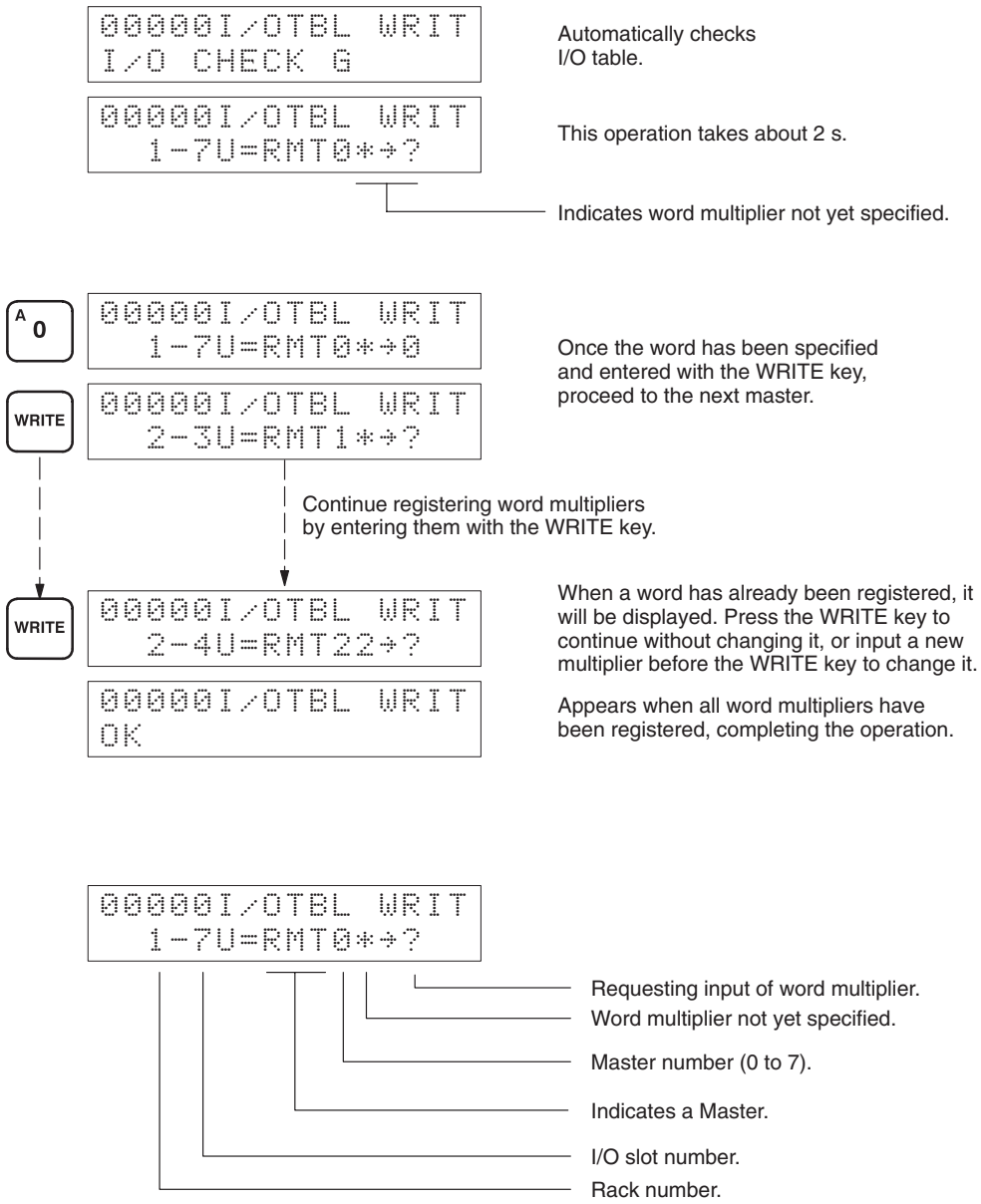
Registering Word Multipliers for Masters

When Remote I/O Master Units in the system are connected to I/O Link Units, Optical I/O Units, Remote Terminals, or I/O Terminals a word multiplier between 0 and 3 must be assigned to each one of the Masters after registering the I/O table. The same word multiplier can be assigned to more than one Master in the same system as long as the same word is not allocated to more than one unit. Word allocations to I/O Link Units, Optical I/O Units, and Remote terminals, and I/O Terminals are computed from the words set on the Units as follows:

(32 x word multiplier) + word setting on the Unit

Make sure that the lowest words allocated to I/O Link Units, Optical I/O Units, Remote Terminals, or I/O Terminals connected to the Master with the lowest

word multiplier, do not overlap with the highest I/O words on the last Expansion I/O Rack.



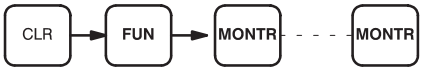
4-5-4 Clearing Error Messages

After the I/O table has been registered, any error messages recorded in memory should be cleared. It is assumed here that the causes of any of the errors for which error messages appear have already been taken care of. If the beeper sounds when an attempt is made to clear an error message, eliminate the cause of the error, and then clear the error message (refer to *Section 8 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



4-5-5 Transferring the I/O Table

The I/O Table Transfer operation transfers a copy of the I/O table to RAM program memory.This allows the user program and I/O table to be written into EPROM together.

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

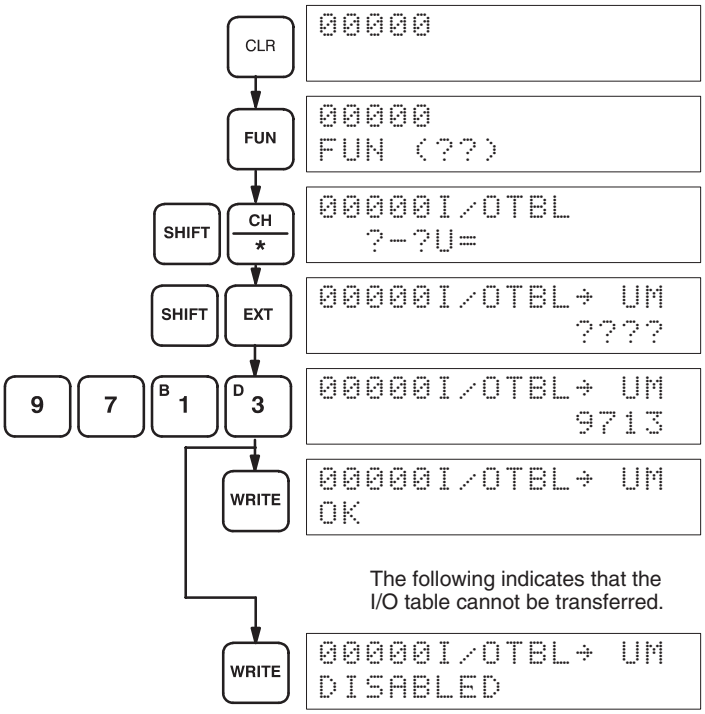
The I/O Table Transfer operation will not work if:

1, 2, 3... 1. When the memory unit is not RAM, or
2. When the contents of program memory exceeds 31.7K words. (To find out the capacity of program memory, do an instruction search for END(01).)

I/O table transfer can only be done in PROGRAM mode.

Key Sequence

Example



4-5-6 Changing the I/O Table

Use of this operation allows I/O Units to be added or removed without the need to re-register the I/O table or amend the user program. The I/O Table

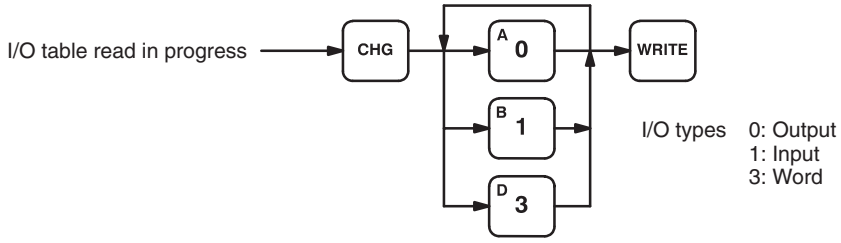
Change operation allows you to register dummy I/O Units in the I/O table. By reserving an entry in the I/O table with this operation, you can prevent word number discrepancies when an I/O Unit is to be added to the System in the future. A dummy I/O Unit can also be registered to prevent discrepancies after an I/O Unit is removed.

When this operation is performed for the first time, the I/O verification error message is displayed because the registered I/O table does not agree with the actually mounted units. Disregard this error message. This message will not be displayed for word reservations (**3**, see below).

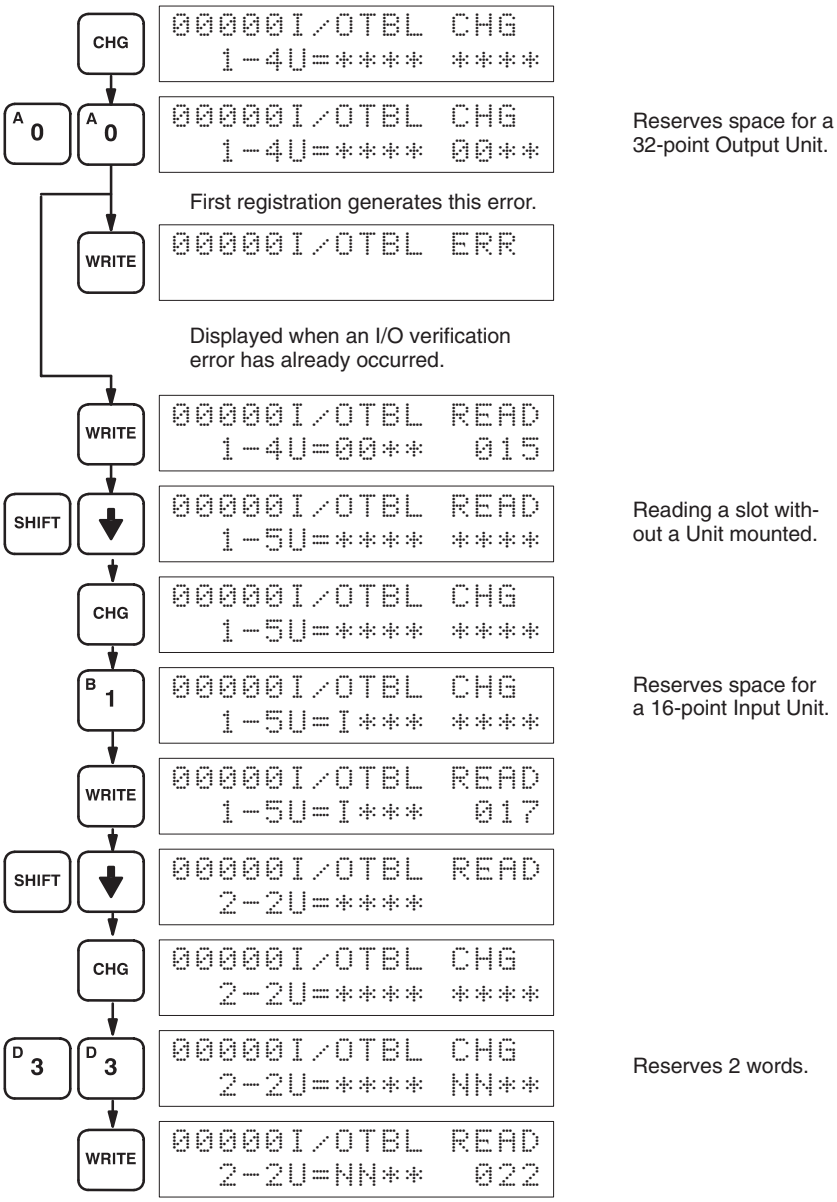
Dummy I/O table entries can be made for Input Units (use the **1** key), Output Units (use the **0** key), and for words (use the **3** key). Press the key once for each Unit or word, e.g., pressing the **3** key twice before pressing **WRITE** will reserve two words. Up to four words may be reserved for each slot.

Note An Input Unit reservation cannot be used for an Output Unit and vice versa. Also, dummy I/O Units cannot be registered for Remote I/O Units, Optical I/O Units, or Interrupt Units.

Key Sequence



Example



4-5-7 Changing I/O Units On-line (C2000H Only)

The On-line Unit Change operation allows mounting and/or removal of I/O Units while the CPU is operating. Only 16-, 32-, and 64-point I/O Units can be changed on-line. On-line changes are not possible on Racks containing Interrupt Input Units, Special I/O Units, or Remote I/O Units. Regardless of whether or not the On-line Unit change operation is used, do not mount a Remote I/O Unit or a Special I/O Unit to a C2000-BI083 Backplane. Doing so will result in an I/O bus error.

Removal or mounting can be done in RUN, MONITOR, or PROGRAM mode. The entire operation must be repeated for each Unit individually, using the following procedure to attach Units.

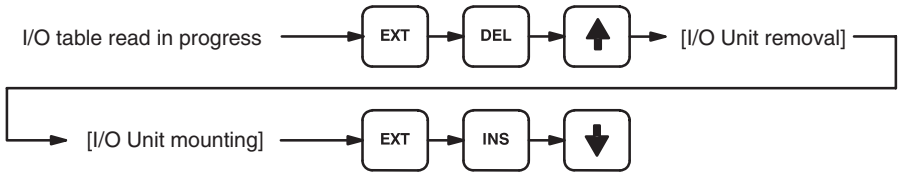
1, 2, 3... 1. Align the bottom of the I/O Unit on the hook on the Rack.
2. Pivoting the Unit on the hook, press in to insert the connector.
3. Tighten the screws at the bottom and top of the I/O Unit.

AR 22 is used to monitor the operation as follows:

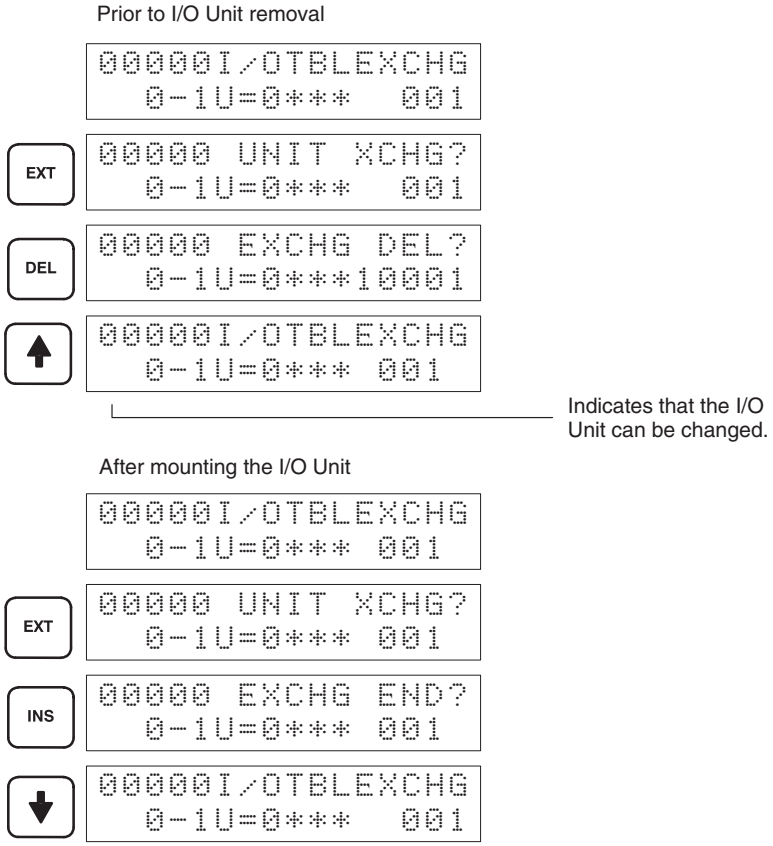
Monitoring via AR Bits

Bit	Function
AR 2200 to AR 2211	A 3-digit BCD number indicating the first word of an I/O Unit that is being changed on-line.
AR 2212 to AR 2214	A 3-bit number indicating the number of words occupied by the I/O Unit that is being changed on-line.
AR 2215	On-line I/O Unit Change Flag.

Key Sequence



Example

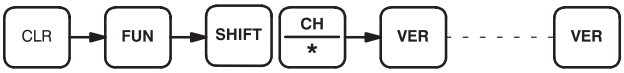


- The outputs of the I/O Unit will turn ON for an instant (causing the I/O Unit's LEDs to blink) when replacing an I/O Unit on-line. Be sure to disconnect the terminal block before removing the Unit. After mounting the new I/O Unit, be sure to reconnect the terminal block.

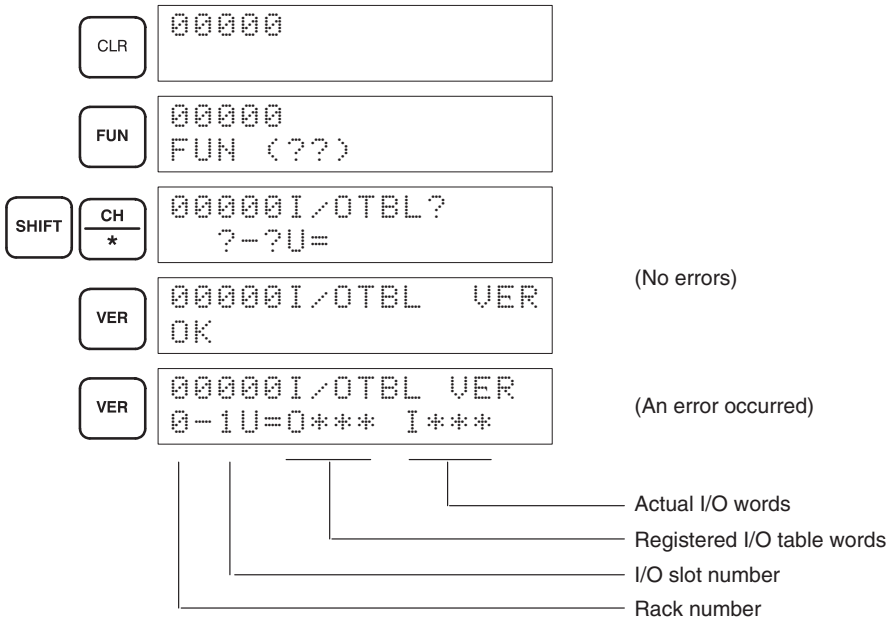
4-5-8 Verifying the I/O Table

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

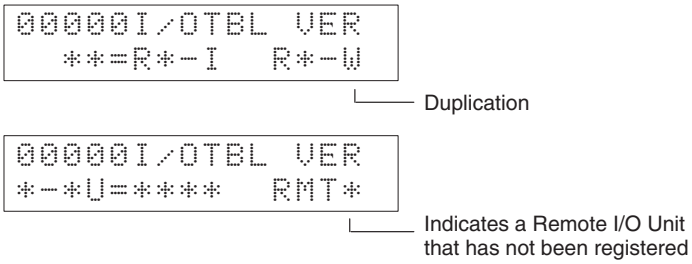
Key Sequence



Example



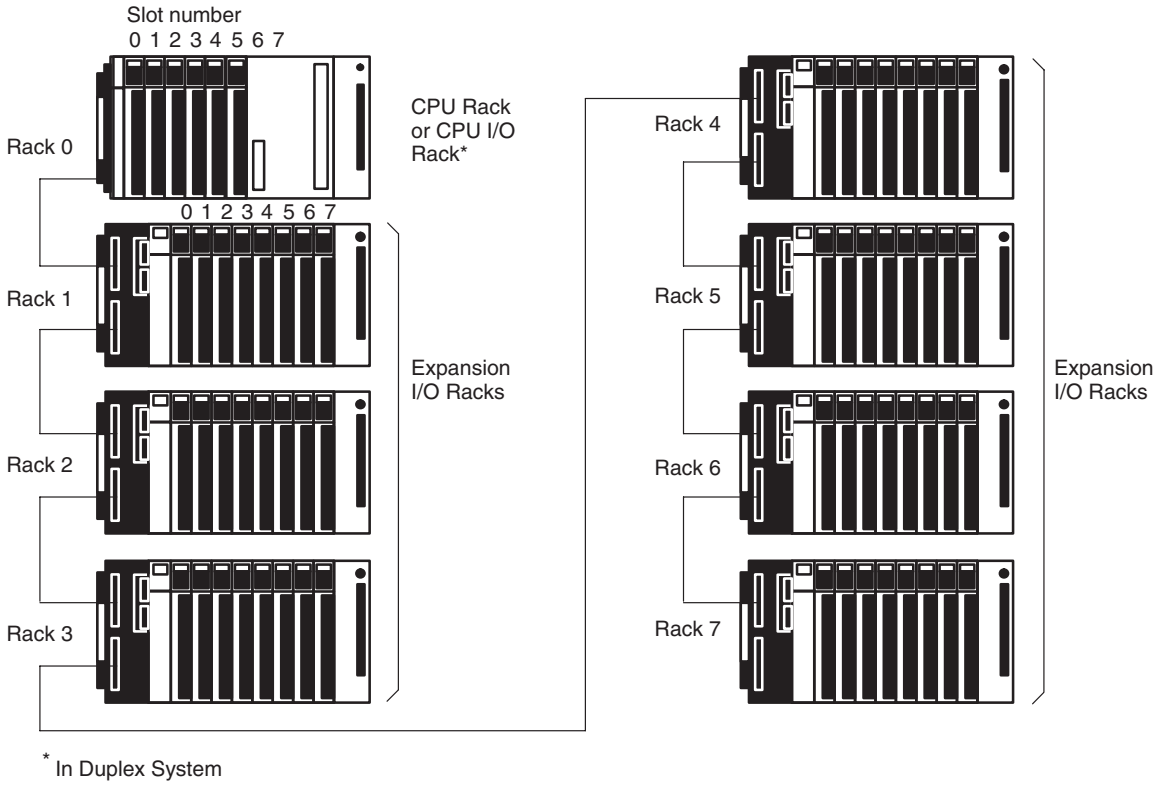
Meaning of Displays



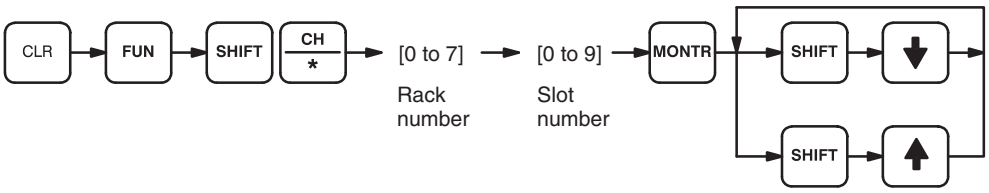
4-5-9 Reading the I/O Table

The I/O Table Read operation is used to access the I/O table that is currently registered in the CPU memory.

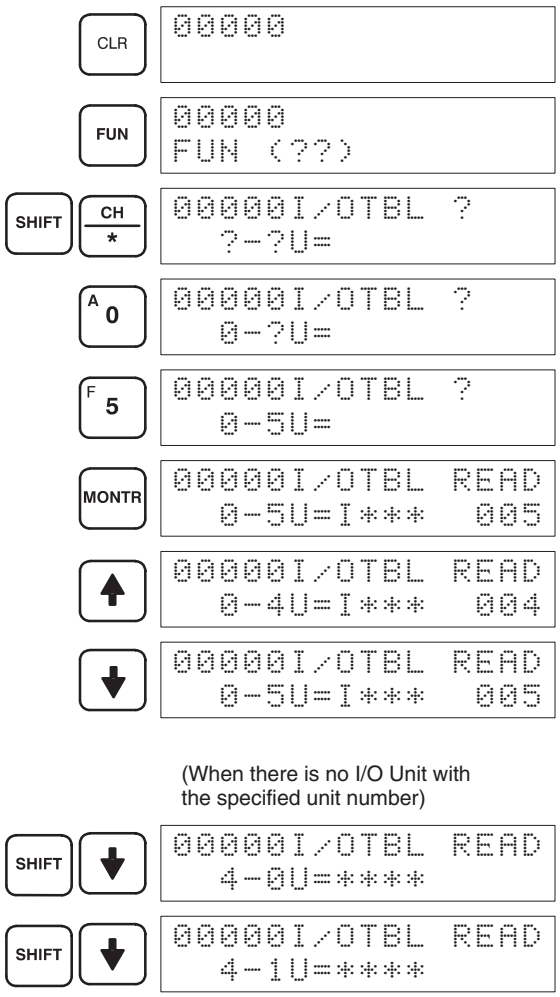
Example of I/O Unit Mounting



Key Sequence



Example

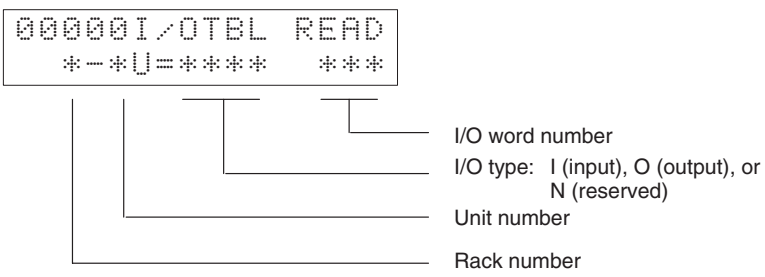


Meaning of Displays

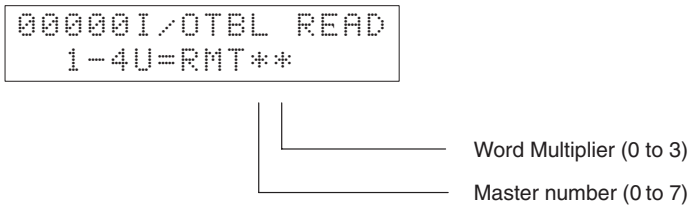
I/O Unit Designations for Displays

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

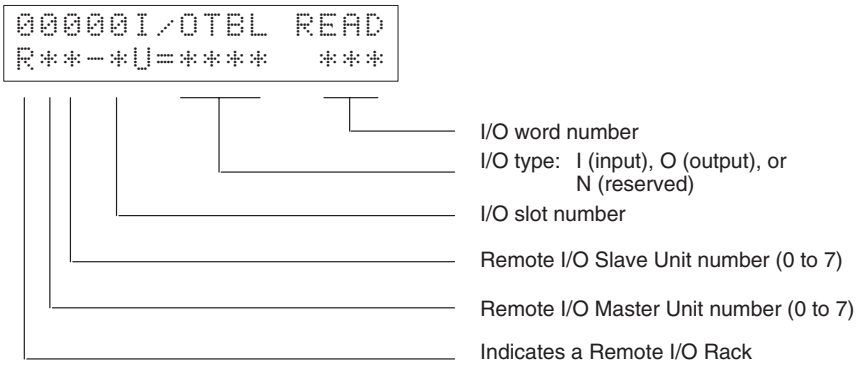
I/O Units, Special I/O Units, I/O Link Units



Remote I/O Master Unit

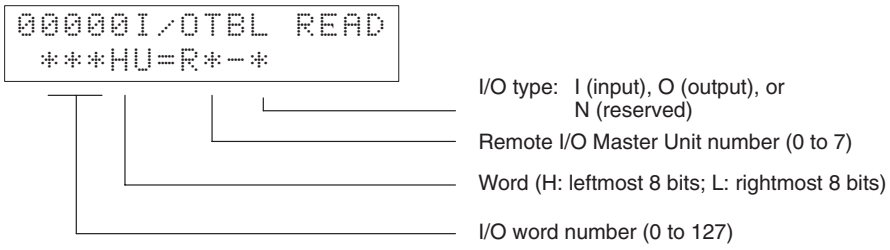


Remote I/O Slave Racks



Interrupt Units

Optical I/O Units, I/O Link Units, Remote Terminals, and I/O Terminals



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

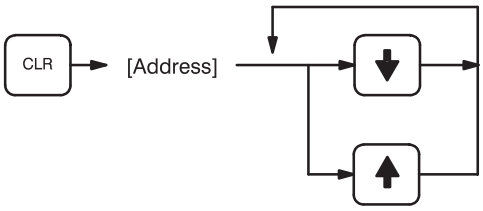
When inputting a program for the first time, it is generally written to Program Memory starting from address 00000. Because this address appears when the display is cleared, it is not necessary to specify it.

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.

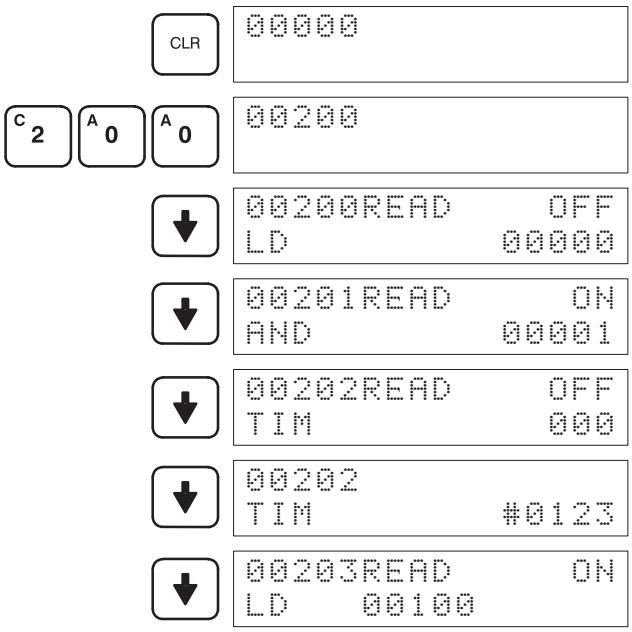
If Program Memory is read in RUN or MONITOR mode, the ON/OFF status of any displayed bit will also be shown.

Key Sequence



Example

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



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 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.

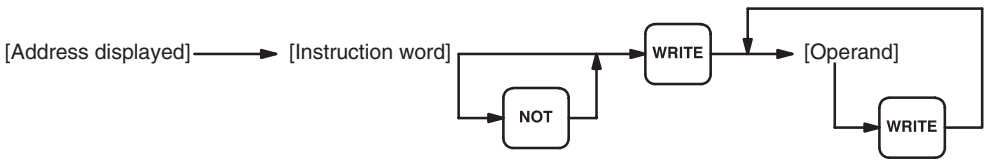
There are two types of function code: those for normal instructions and those for block instructions. Function codes for block instructions are always written between pointed parentheses <like this>. Both types of function code are used in basically the same way, but SHIFT must be pressed before inputting a block instruction function code.

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, press SHIFT if a block instruction is being entered, 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.

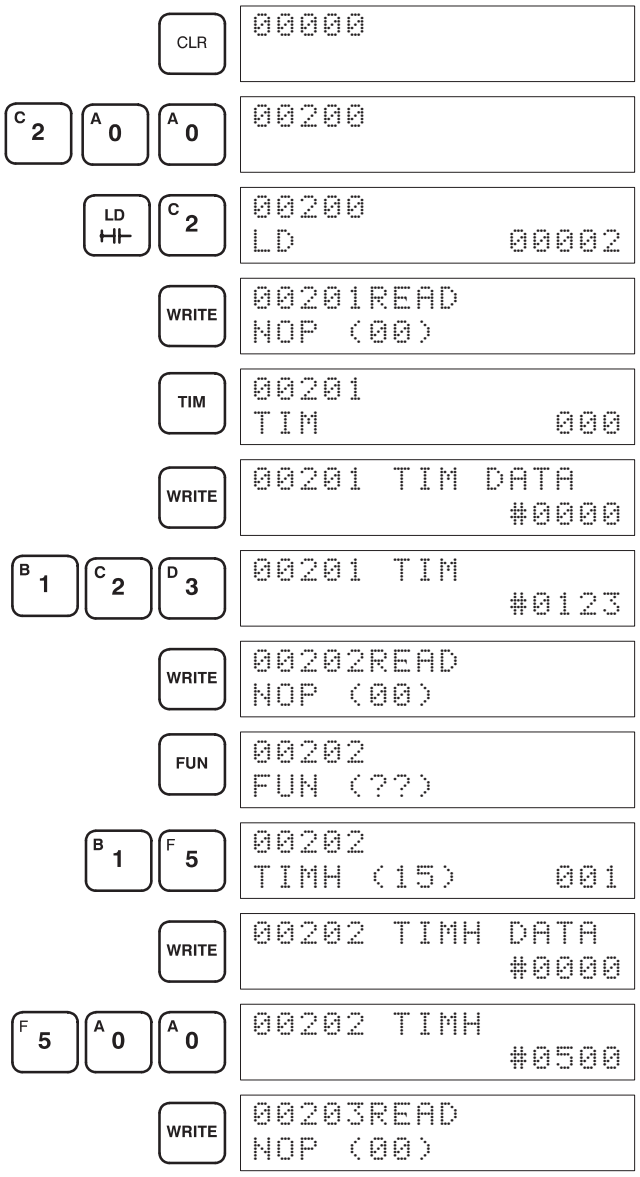
⚠ **Caution** Enter function codes with care and be sure to press SHIFT when required.

Key Sequence



Example

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



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

Error Messages

The following error messages may appear when inputting a program. Correct the error as indicated and continue with the input operation. The asterisks in

the displays shown below will be replaced with numeric data, normally an address, in the actual display.

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

4-6-3 Checking the Program

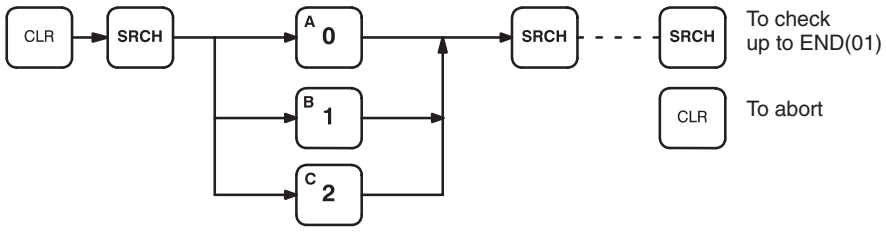
Once a program has been entered, it should be checked for syntax to be sure that no programming rules have been violated. This check should also be performed if the program has been changed in any way that might create a syntax error.

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

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

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

Key Sequence



(0, 1, 2, Check levels)

Check Levels and Error Messages

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

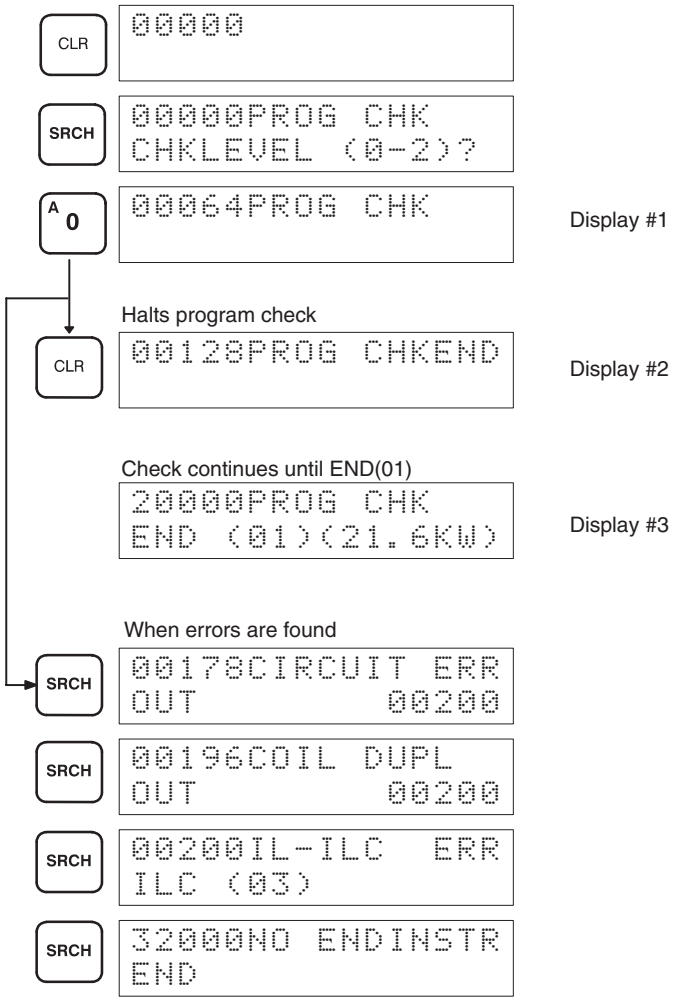
The address where the error was generated will also be displayed.

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

Type	Message	Meaning and appropriate response
Type A	?????	The program has been lost. Re-enter the program.
	NO END INSTR	There is no END(01) in the program. Write END(01) at the final address in the program.
	CIRCUIT ERR	The number of logic blocks and logic block instructions does not agree, i.e., either LD or LD NOT has been used to start a logic block whose execution condition has not been used by another instruction, or a logic block instruction has been used that does not have the required number of logic blocks. Check your program.
	LOCN ERR	An instruction is in the wrong place in the program. Check instruction requirements and correct the program.
	DUPL	The same jump number, block number, or subroutine number has been used twice. Correct the program so that the same number is only used once for each. (Jump number 00 may be used as often as required.)
	SBN UNDEFD	SBS(91) has been programmed for a subroutine number that does not exist. Correct the subroutine number or program the required subroutine.
	JME UNDEFD	A JME(04) is missing for a JMP(05). Correct the jump number or insert the proper JME(04).
	OPERAND ERR	A constant entered for the instruction is not within defined values. Change the constant so that it lies within the proper range.
	STEP ERR	STEP(08) with a section number and STEP(08) without a section number have been used correctly. Check STEP(08) programming requirements and correct the program.
	BPRG-BEND ERR	BPRG(96) and BEND<01> have not been used in pairs. Correct the program so that each BPRG(96) has one and only one BEND<01>.
	IF-IEND ERR	IF<02>, IF<02> NOT, ELSE<03>, and IEND<04> have not been used properly. Correct the program so that each IF<02> and IF<02> NOT has a corresponding IEND<04>, and so that ELSE<03>, when used, is located between them.
	LOOP-LEND ERR	LOOP<09>, LEND<10>, and LOOP<10> NOT have not been used properly. Correct the program so that loops are not nested, do not interrupt any IF<02>-IEND<04> program sections, and are used in the proper order.
Type B	IL-ILC ERR	IL(02) and ILC(03) are not used in pairs. Correct the program so that each IL(02) has a unique ILC(03). Although this error message will appear if more than one IL(02) is used with the same ILC(03), the program will executed as written. Make sure your program is written as desired before proceeding.
	JMP-JME ERR	JMP(04) 00 and JME(05) 00 are not used in pairs. Although this error message will appear if more than one JMP(04) 00 is used with the same JME(05) 00, the program will be executed as written. Make sure your program is written as desired before proceeding.
	SBN-RET ERR	If the displayed address is that of SBN(92), two different subroutines have been defined with the same subroutine number. Change one of the subroutine numbers or delete one of the subroutines. If the displayed address is that of RET(93), RET(93) has not been used properly. Check requirements for RET(93) and correct the program.
Type C	JMP UNDEFD	JME(05) has been used with no JMP(04) with the same jump number. Add a JMP(04) with the same number or delete the JME(05) that is not being used.
	SBS UNDEFD	A subroutine exists that is not called by SBS(91). Program a subroutine call in the proper place, or delete the subroutine if it is not required.
	COIL DUPL	The same bit is being controlled (i.e., turned ON and/or OFF) by more than one instruction (e.g., OUT, OUT NOT, DIFU(13), DIFD(14), KEEP(11), SFT(10), SET<07>). Although this is allowed for certain instructions, check instruction requirements to confirm that the program is correct or rewrite the program so that each bit is controlled by only one instruction.

Example

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

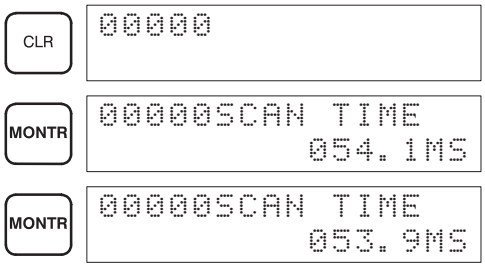


4-6-4 Displaying the Cycle Time

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.

Example



4-6-5 Program Searches

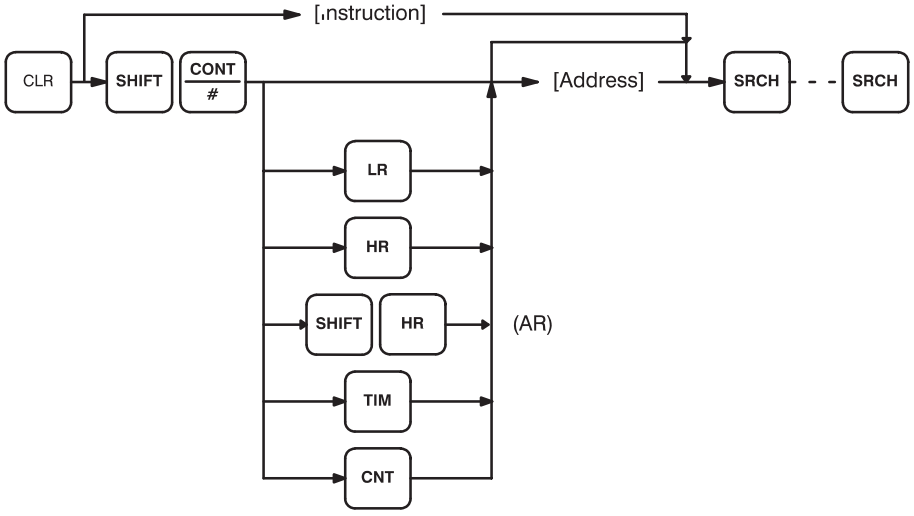
The program can be searched for occurrences of any designated instruction or data area address used in an instruction. Searches can be performed from any currently displayed address or from a cleared display.

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.

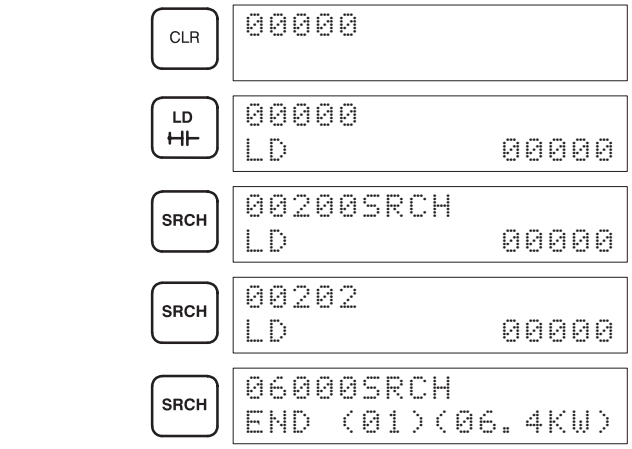
When the first word of a multiword instruction is displayed for a search operation, the other words of the instruction can be displayed by pressing the down key before continuing the search.

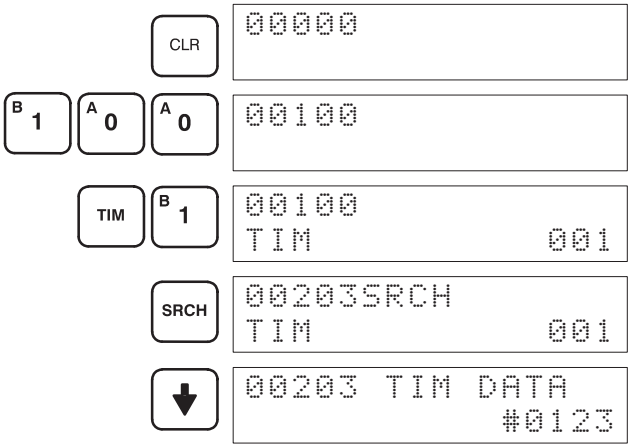
If Program Memory is read in RUN or MONITOR mode, the ON/OFF status of any bit displayed will also be shown.

Key Sequence

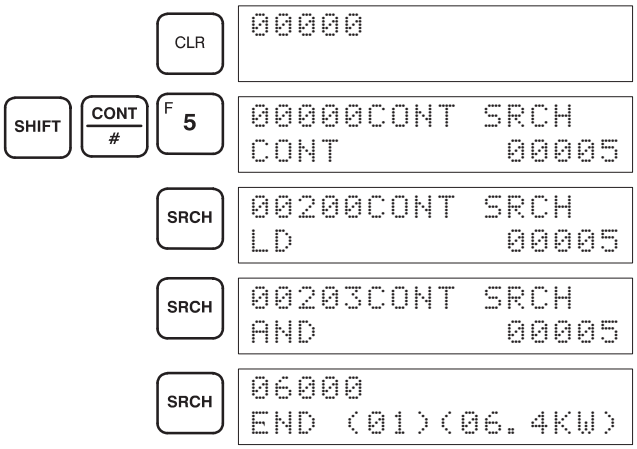


Example:
Instruction Search





Example:
Bit Search



4-6-6 Inserting and Deleting Instructions

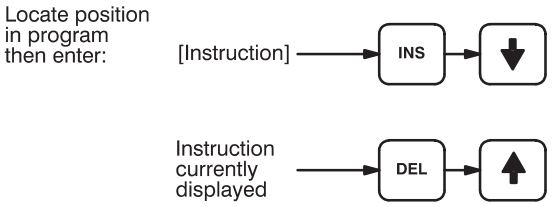
In PROGRAM mode, any instruction that is currently displayed can be deleted or another instruction can be inserted before it. These are not possible in RUN or MONITOR modes.

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

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

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

Key Sequences



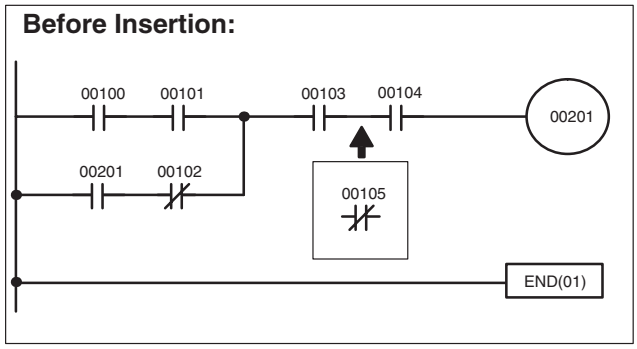
When an instruction is inserted or deleted, all addresses in Program Memory following the operation are adjusted automatically so that there are no blank addresses or no unaddressed instructions.

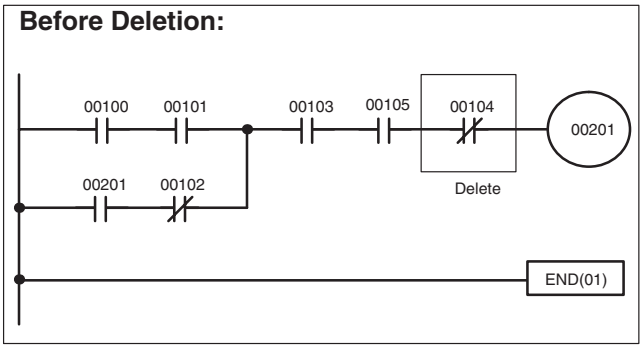
Example

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

Original Program

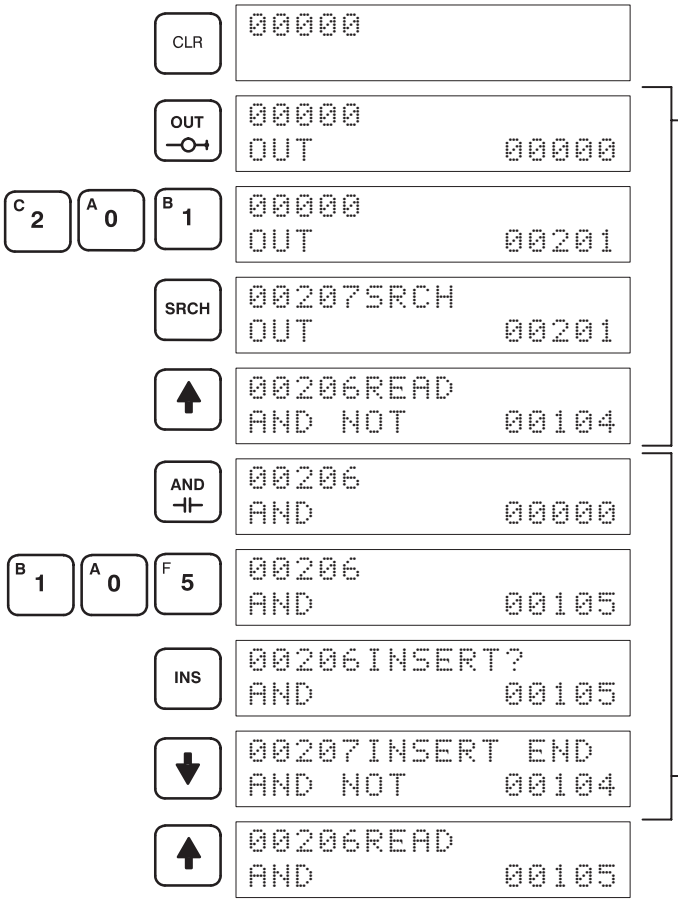
Address	Instruction	Operands
00000	LD	00100
00001	AND	00101
00002	LD	00201
00003	AND NOT	00102
00004	OR LD	-
00005	AND	00103
00006	AND NOT	00104
00007	OUT	00201
00008	END(01)	-





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

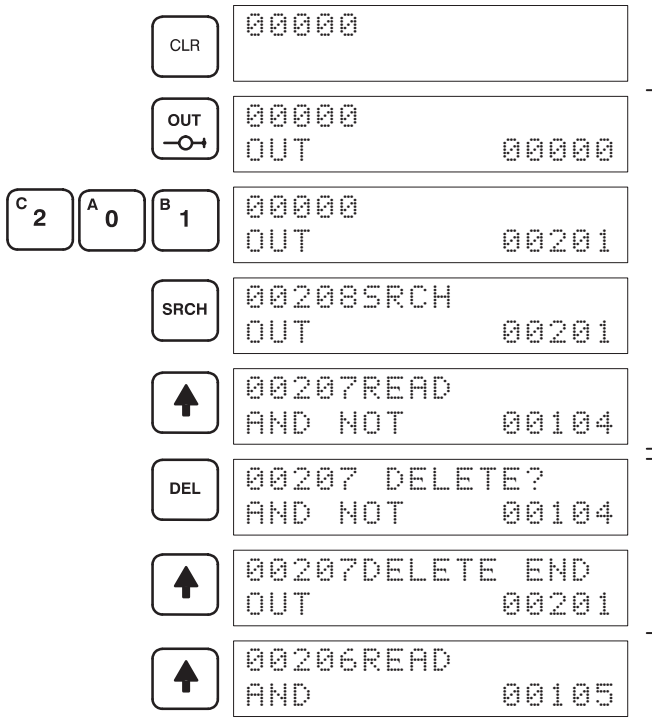
Inserting an Instruction



Program After Insertion

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

Deleting an Instruction



Program After Deletion

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

4-6-7 Branching Instruction Lines

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

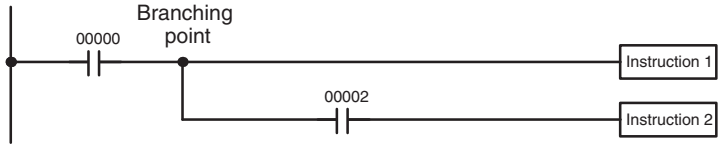


Diagram A: Correct Operation

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

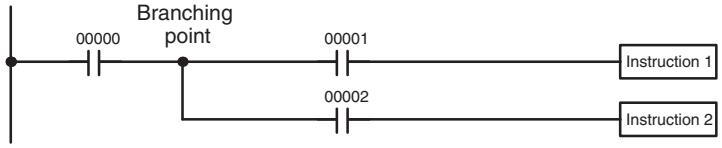


Diagram B: Incorrect Operation

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

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

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

There are two means of programming branching programs to preserve the execution condition. One is to use TR bits; the other, to use interlocks (IL(02)/IL(03)).

TR Bits

The TR area provides eight bits, TR 0 through TR 7, that can be used to temporarily preserve execution conditions. If a TR bit is placed at a branching point, the current execution condition will be stored at the designated TR bit. When returning to the branching point, the TR bit restores the execution status that was saved when the branching point was first reached in program execution.

The previous diagram B can be written as shown below to ensure correct execution. In mnemonic code, the execution condition is stored at the branching point using the TR bit as the operand of the OUTPUT instruction.

This execution condition is then restored after executing the right-hand instruction by using the same TR bit as the operand of a LOAD instruction

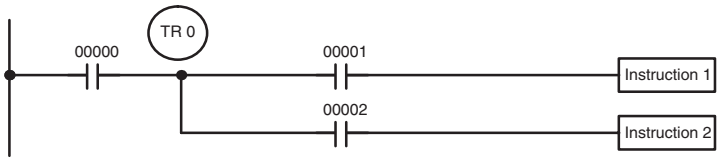
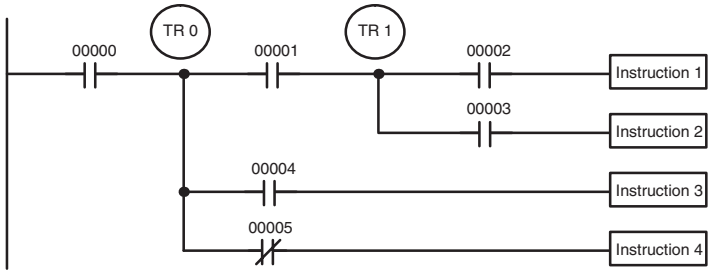


Diagram B: Corrected Using a TR bit

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

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

The following example shows an application using two TR bits.



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

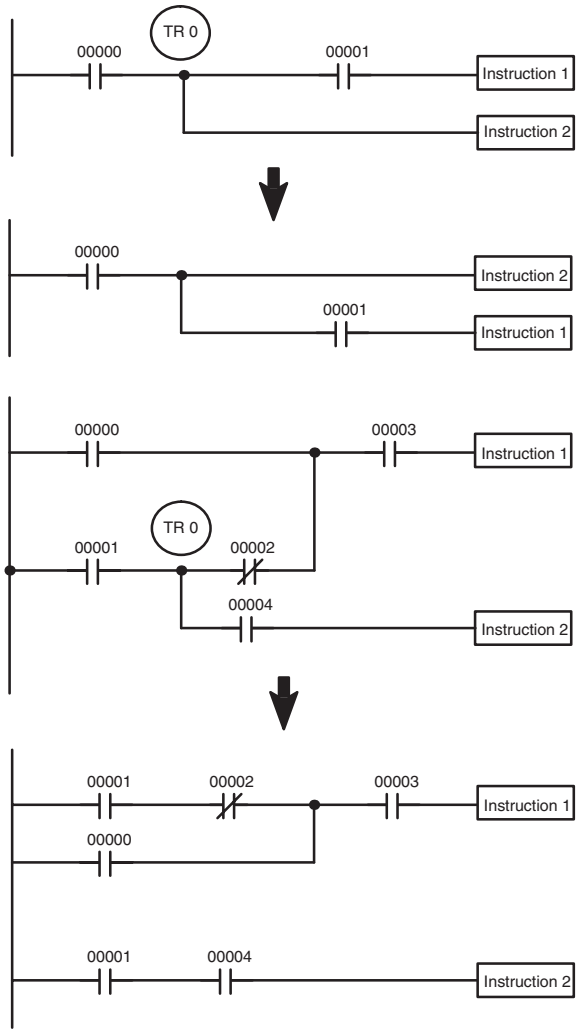
In this example, TR 0 and TR 1 are used to store the execution conditions at the branching points. After executing instruction 1, the execution condition stored in TR 1 is loaded for an AND with the status IR 00003. The execution condition stored in TR 0 is loaded twice, the first time for an AND with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

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

When drawing a ladder diagram, be careful not to use TR bits unless necessary. Often the number of instructions required for a program can be reduced and ease of understanding a program increased by redrawing a diagram that would otherwise required TR bits. In both of the following pairs of diagrams,

the bottom versions require fewer instructions and do not require TR bits. In the first example, this is achieved by reorganizing the parts of the instruction block: the bottom one, by separating the second OUTPUT instruction and using another LOAD instruction to create the proper execution condition for it.

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



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

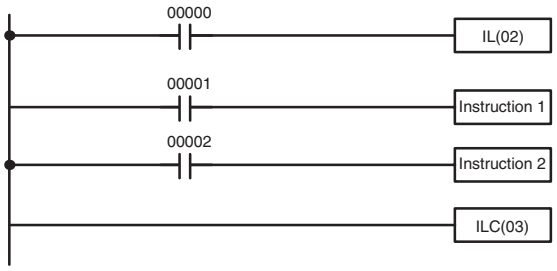
Interlocks

The problem of storing execution conditions at branching points can also be handled by using the INTERLOCK (IL(02)) and INTERLOCK CLEAR (ILC(03)) instructions to eliminate the branching point completely while allowing a specific execution condition to control a group of instructions. The INTERLOCK and INTERLOCK CLEAR instructions are always used together.

When an INTERLOCK instruction is placed before a section of a ladder program, the execution condition for the INTERLOCK instruction will control the

execution of all instruction up to the next INTERLOCK CLEAR instruction. If the execution condition for the INTERLOCK instruction is OFF, all right-hand instructions through the next INTERLOCK CLEAR instruction will be executed with OFF execution conditions to reset the entire section of the ladder diagram. The effect that this has on particular instructions is described in *5-11 INTERLOCK and INTERLOCK CLEAR – IL(02) and ILC(03) .*

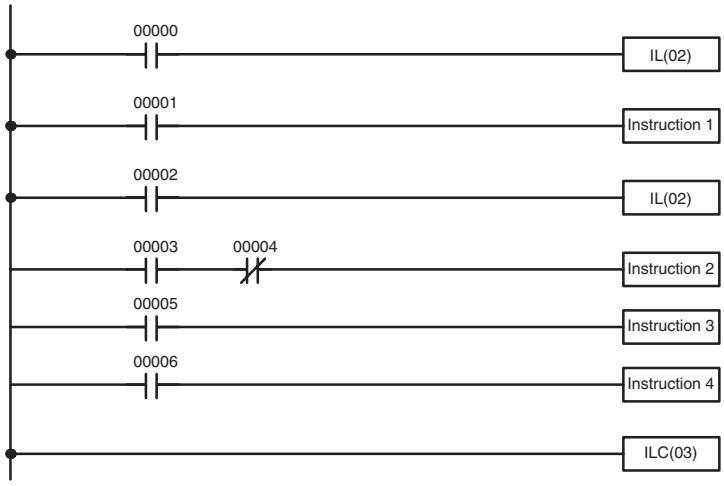
Diagram B can also be corrected with an interlock. Here, the conditions leading up to the branching point are placed on an instruction line for the INTERLOCK instruction, all of lines leading from the branching point are written as separate instruction lines, and another instruction line is added for the INTERLOCK CLEAR instruction. No conditions are allowed on the instruction line for INTERLOCK CLEAR. Note that neither INTERLOCK nor INTERLOCK CLEAR requires an operand.



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

If IR 00000 is ON in the revised version of diagram B, above, the status of IR 00001 and that of IR 00002 would determine the execution conditions for instructions 1 and 2, respectively. Because IR 00000 is ON, this would produce the same results as ANDing the status of each of these bits. If IR 00000 is OFF, the INTERLOCK instruction would produce an OFF execution condition for instructions 1 and 2 and then execution would continue with the instruction line following the INTERLOCK CLEAR instruction.

As shown in the following diagram, more than one INTERLOCK instruction can be used within one instruction block; each is effective through the next INTERLOCK CLEAR instruction.



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

If IR 00000 in the above diagram is OFF (i.e., if the execution condition for the first INTERLOCK instruction is OFF), instructions 1 through 4 would be executed with OFF execution conditions and execution would move to the instruction following the INTERLOCK CLEAR instruction. If IR 00000 is ON, the status of IR 00001 would be loaded as the execution condition for instruc-

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

4-6-8 Jumps

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

Jumps are created using the JUMP (JMP(04)) and JUMP END (JME(05)) instructions. If the execution condition for a JUMP instruction is ON, the program is executed normally as if the jump did not exist. If the execution condition for the JUMP instruction is OFF, program execution moves immediately to a JUMP END instruction without changing the status of anything between the JUMP and JUMP END instruction.

All JUMP and JUMP END instructions are assigned jump numbers ranging between 00 and 99. There are two types of jumps. The jump number used determines the type of jump.

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

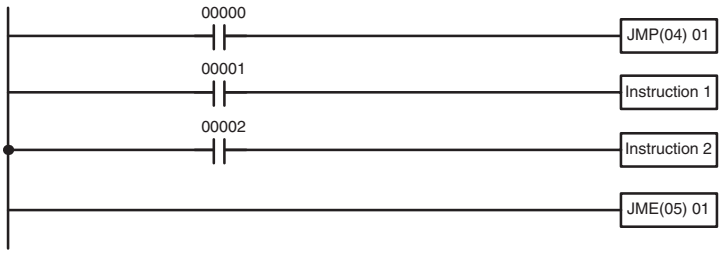


Diagram B: Corrected with a Jump

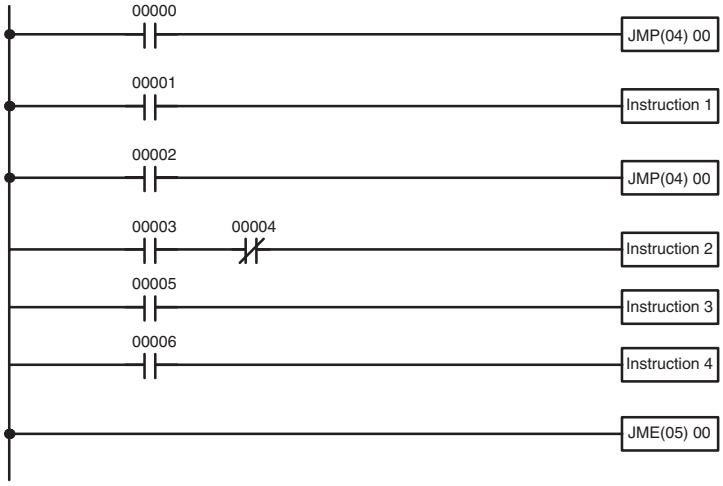
Address	Instruction	Operands
00000	LD	00000
00001	JMP(04)	01
00002	LD	00001
00003	Instruction 1	
00004	LD	00002
00005	Instruction 2	
00006	JME(05)	01

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 num-

ber of 00. Although, as in all jumps, no status is changed and no instructions are executed between the JUMP 00 and JUMP END 00 instructions, the program must search for the next JUMP END 00 instruction, producing a slightly longer execution time.

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



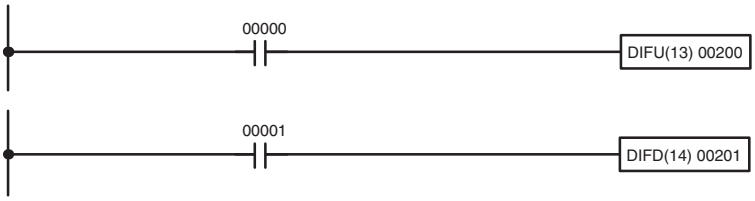
Address	Instruction	Operands
00000	LD	00000
00001	JMP(04)	00
00002	LD	00001
00003	Instruction 1	
00004	LD	00002
00005	JMP(04)	00
00006	LD	00003
00007	AND NOT	00004
00008	Instruction 2	
00009	LD	00005
00010	Instruction 3	
00011	LD	00006
00012	Instruction 4	
00013	JME(05)	00

4-7 Controlling Bit Status

There are five instructions that can be used generally to control individual bit status. These are the OUTPUT, OUTPUT NOT, DIFFERENTIATE UP, DIFFERENTIATE DOWN, and KEEP instructions. All of these instructions appear as the last instruction in an instruction line and take a bit address for an operand. Although details are provided in *5-6 Bit Control Instructions*, these instructions (except for OUTPUT and OUTPUT NOT, which have already been introduced) are described here because of their importance in most programs. Although these instructions are used to turn ON and OFF output bits in the IR area (i.e., to send or stop output signals to external devices), they are also used to control the status of other bits in the IR area or in other data areas.

4-7-1 DIFFERENTIATE UP and DIFFERENTIATE DOWN

DIFFERENTIATE UP and DIFFERENTIATE DOWN instructions are used to turn the operand bit ON for one 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

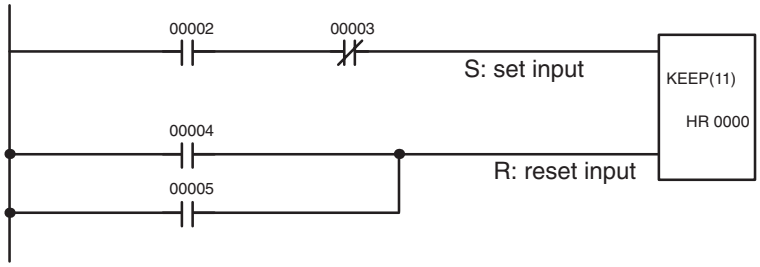
Here, IR 00200 will be turned ON for one cycle after IR 00000 goes ON. The next time DIFU(13) 00200 is executed, IR 00200 will be turned OFF, regardless of the status of IR 00000. With the DIFFERENTIATE DOWN instruction, IR 00201 will be turned ON for one cycle after IR 00001 goes OFF (IR 00201 will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



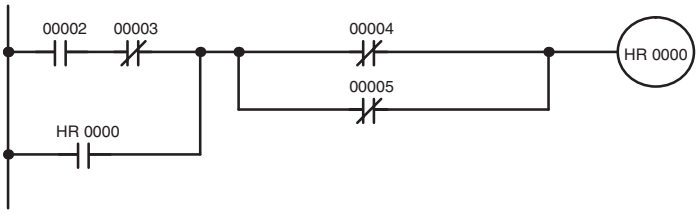
Address	Instruction	Operands
00000	LD	00002
00001	AND NOT	00003
00002	LD	00004
00003	OR	00005
00004	KEEP(11)	HR 0000

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 works bits. All bits in the IR area that are not allocated as I/O bits, and certain unused bits in the AR area, are available for use as work bits. Be careful to keep an accurate record of how and where you use work bits. This helps in program planning and writing, and also aids in debugging operations.

Work Bit Applications

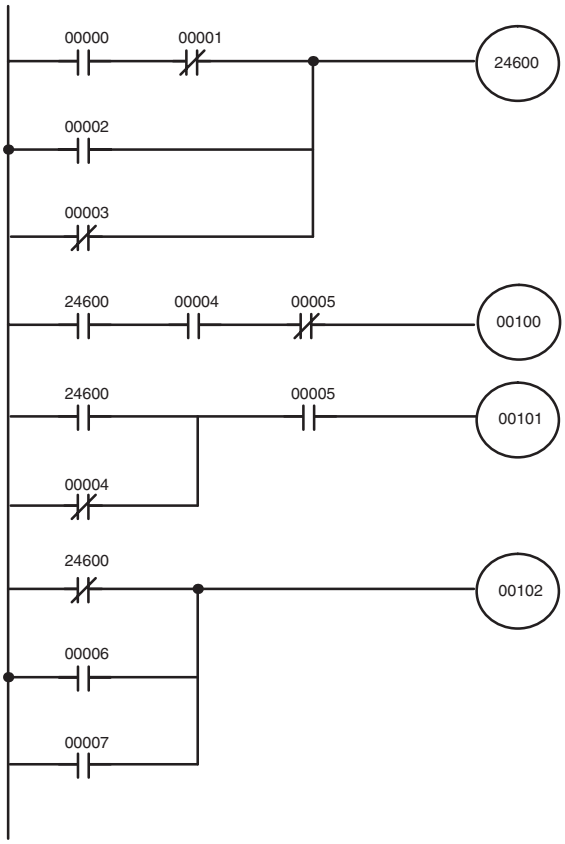
Examples given later in this subsection show two of the most common ways to employ work bits. These should act as a guide to the almost limitless number of ways in which the work bits can be used. Whenever difficulties arise in programming a control action, consideration should be given to work bits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

Work bits are often used with the OUTPUT, OUTPUT NOT, DIFFERENTIATE UP, DIFFERENTIATE DOWN, and KEEP instructions. The work bit is used first as the operand for one of these instructions so that later it can be used as a condition that will determine how other instructions will be executed. Work bits can also be used with other instructions, e.g., with the SHIFT REGISTER instruction (SFT(10)). An example of the use of work words and bits with the SHIFT REGISTER instruction is provided in *5-13-1 SHIFT REGISTER – SFT(10)*.

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

Reducing Complex Conditions

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



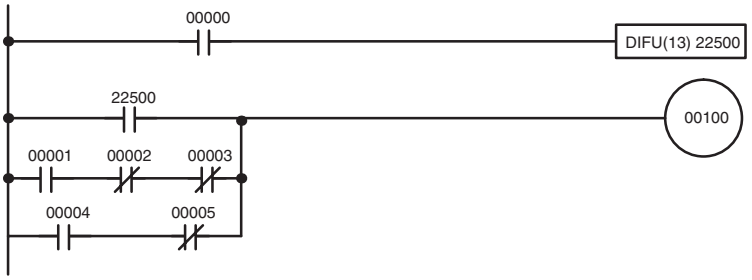
Address	Instruction	Operands
00000	LD	00000
00001	AND NOT	00001
00002	OR	00002
00003	OR NOT	00003
00004	OUT	24600
00005	LD	24600
00006	AND	00004
00007	AND NOT	00005
00008	OUT	00100
00009	LD	24600
00010	OR NOT	00004
00011	AND	00005
00012	OUT	00101
00013	LD NOT	24600
00014	OR	00006
00015	OR	00007
00016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

Differentiated Conditions

Work bits can also be used if differential treatment is necessary for some, but not all, of the conditions required for execution of an instruction. In this exam-

ple, 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.

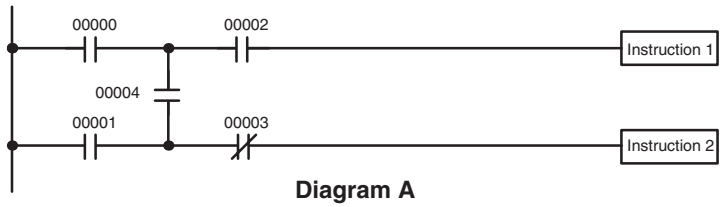


Diagram A

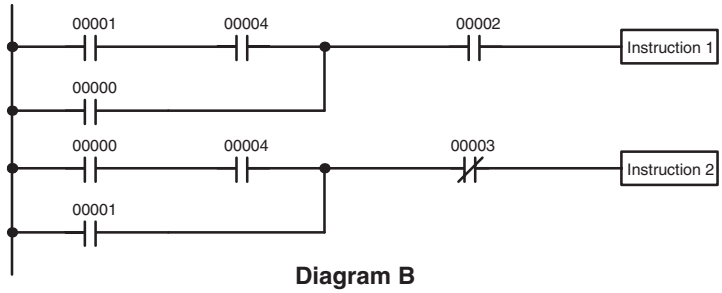


Diagram B

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.

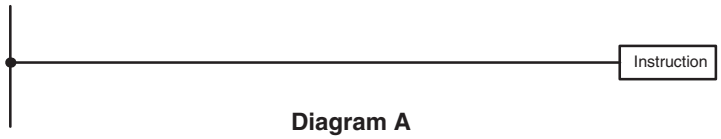


Diagram A

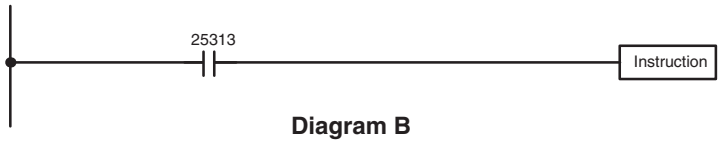


Diagram B

Address	Instruction	Operands
00000	LD	25313
00001	Instruction	

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

When drawing ladder diagrams, it is important to keep in mind the number of instructions that will be required to input it. In diagram A, below, an OR LOAD instruction will be required to combine the top and bottom instruction lines. This can be avoided by redrawing as shown in diagram B so that no AND LOAD or OR LOAD instructions are required. Refer to *5-5-2 AND LOAD and OR LOAD* for more details and *7-5 Inputting, Modifying and Checking the Program* for further examples.

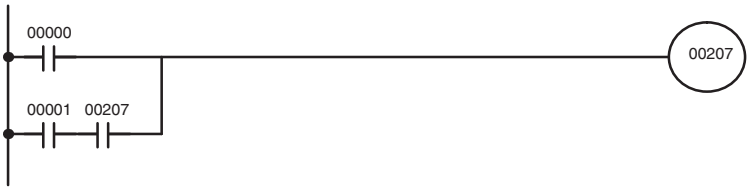


Diagram A

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

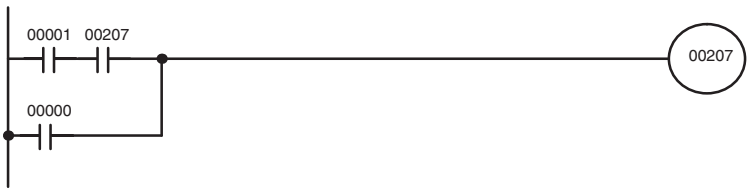


Diagram B

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

4-10 Program Execution

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

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

SECTION 5
Instruction Set

The C1000H and C2000H PC have large programming instruction sets that allow for easy programming of complicated control processes. This section explains each instruction individually and provides the ladder diagram symbol, data areas, and flags used with each.

The many instructions provided by the C1000H and C2000H are described in following subsections by instruction group. These groups include Ladder Diagram Instructions, Bit Control Instructions, Timer and Counter Instructions, Data Shifting Instructions, Data Movement Instructions, Data Comparison Instructions, Data Conversion Instructions, Binary Calculation Instructions, BCD Calculation Instructions, Logic Instructions, Subroutines, Block Programming Instructions, Special Instructions, Intellegent I/O Instructions, and SYSMAC NET Link/SYSMAC LINK System Instructions.

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

5-1 Notation . . . 102
5-2 Instruction Format . . . 102
5-3 Data Areas, Definer Values, and Flags . . . 102
5-4 Differentiated Instructions . . . 104
5-5 Coding Right-hand Instructions . . . 104
5-6 Ladder Diagram Instructions . . . 107
5-6-1 LOAD, LOAD NOT, AND, AND NOT, OR, and OR NOT . . . 108
5-6-2 AND LOAD and OR LOAD . . . 109
5-7 Bit Control Instructions . . . 109
5-7-1 OUTPUT and OUTPUT NOT – OUT and OUT NOT . . . 109
5-7-2 DIFFERENTIATE UP and DIFFERENTIATE DOWN – DIFU(13) and DIFD(14) . . . 110
5-7-3 KEEP – KEEP(11) . . . 112
5-8 INTERLOCK and INTERLOCK CLEAR – IL(02) and ILC(03) . . . 113
5-9 JUMP and JUMP END – JMP(04) and JME(05) . . . 115
5-10 END – END(01) . . . 116
5-11 NO OPERATION – NOP(00) . . . 116
5-12 Timer and Counter Instructions . . . 117
5-12-1 TIMER – TIM . . . 118
5-12-2 HIGH-SPEED TIMER – TIMH(15) . . . 121
5-12-3 COUNTER – CNT . . . 122
5-12-4 REVERSIBLE COUNTER – CNTR(12) . . . 125
5-13 Data Shifting . . . 127
5-13-1 SHIFT REGISTER – SFT(10) . . . 127
5-13-2 REVERSIBLE SHIFT REGISTER – SFTR(84) . . . 129
5-13-3 ARITHMETIC SHIFT LEFT – ASL(25) . . . 131
5-13-4 ARITHMETIC SHIFT RIGHT – ASR(26) . . . 132
5-13-5 ROTATE LEFT – ROL(27) . . . 132
5-13-6 ROTATE RIGHT – ROR(28) . . . 133
5-13-7 ONE DIGIT SHIFT LEFT – SLD(74) . . . 133
5-13-8 ONE DIGIT SHIFT RIGHT – SRD(75) . . . 134
5-13-9 WORD SHIFT – WSFT(16) . . . 134
5-14 Data Movement . . . 135
5-14-1 MOVE – MOV(21) . . . 135
5-14-2 MOVE NOT – MVN(22) . . . 136

5-14-3 BLOCK SET – BSET(71) 136
5-14-4 BLOCK TRANSFER – XFER(70) 138
5-14-5 DATA EXCHANGE – XCHG(73) 138
5-14-6 SINGLE WORD DISTRIBUTE – DIST(80) 139
5-14-7 DATA COLLECT – COLL(81) 139
5-14-8 MOVE BIT – MOVB(82) 140
5-14-9 MOVE DIGIT – MOVD(83) 141
5-15 Data Comparison 142
5-15-1 COMPARE – CMP(20) 142
5-15-2 BLOCK COMPARE – BCMP(68) 145
5-15-3 TABLE COMPARE – TCMP(85) 146
5-16 Data Conversion 148
5-16-1 BCD-TO-BINARY – BIN(23) 148
5-16-2 DOUBLE BCD-TO-DOUBLE BINARY – BINL(58) 148
5-16-3 BINARY-TO-BCD – BCD(24) 149
5-16-4 DOUBLE BINARY-TO-DOUBLE BCD – BCDL(59) 150
5-16-5 4-TO-16 DECODER – MLPX(76) 150
5-16-6 16-TO-4 ENCODER – DMPX(77) 152
5-16-7 7-SEGMENT DECODER – SDEC(78) 154
5-16-8 ASCII CONVERT – ASC(86) 157
5-17 BCD Calculations 158
5-17-1 INCREMENT – INC(38) 159
5-17-2 DECREMENT – DEC(39) 159
5-17-3 SET CARRY – STC(40) 159
5-17-4 CLEAR CARRY – CLC(41) 159
5-17-5 BCD ADD – ADD(30) 160
5-17-6 DOUBLE BCD ADD – ADDL(54) 161
5-17-7 BCD SUBTRACT – SUB(31) 162
5-17-8 DOUBLE BCD SUBTRACT – SUBL(55) 164
5-17-9 BCD MULTIPLY – MUL(32) 165
5-17-10 DOUBLE BCD MULTIPLY – MULL(56) 166
5-17-11 BCD DIVIDE – DIV(33) 167
5-17-12 DOUBLE BCD DIVIDE – DIVL(57) 168
5-17-13 FLOATING POINT DIVIDE – FDIV(79) 169
5-17-14 SQUARE ROOT – ROOT(72) 172
5-18 Binary Calculations 174
5-18-1 BINARY ADD – ADB(50) 174
5-18-2 BINARY SUBTRACT – SBB(51) 176
5-18-3 BINARY MULTIPLY – MLB(52) 178
5-18-4 BINARY DIVIDE – DVB(53) 179
5-19 Logic Instructions 179
5-19-1 COMPLEMENT – COM(29) 179
5-19-2 AND WORD – ANDW(34) 180
5-19-3 OR WORD – ORW(35) 180
5-19-4 EXCLUSIVE OR – XORW(36) 181
5-19-5 EXCLUSIVE NOR – XNRW(37) 182
5-20 Subroutines and Interrupt Control 182
5-20-1 Overview 182
5-20-2 SUBROUTINE START and RETURN – SBN(92)/RET(93) 183
5-20-3 SUBROUTINE ENTER – SBS(91) 183
5-20-4 INTERRUPT CONTROL – INT(89) 185
5-21 Block Programming Instructions 190
5-21-1 Overview 190
5-21-2 BLOCK PROGRAM BEGIN – BPRG(96) and BLOCK PROGRAM END – BEND<01> 190

5-21-3 SET – SET<07> and RESET – RSET<08> 191
5-21-4 Block Branching–IF<02>, IF<02>NOT, ELSE<03>, and IEND<04> 191
5-21-5 ONE CYCLE AND WAIT – WAIT<05> 193
5-21-6 TIMER WAIT – TIMW<13> and HIGH-SPEED TIMER WAIT – TMHW<15> 195
5-21-7 COUNTER WAIT – CNTW<14> 196
5-21-8 CONDITIONAL BLOCK EXIT – EXIT<06> and EXIT<06> NOT 197
5-21-9 Block Loop Control–LOOP<09>, LEND<10>, and LEND<10> NOT 197
5-21-10 BLOCK PROGRAM PAUSE – BPPS<11> and BLOCK PROGRAM RESTART – BPRS<12> 198
5-22 Step Instructions 199
5-22-1 STEP DEFINE and STEP START–STEP(08)/SNXT(09) 199
5-23 Special Instructions 208
5-23-1 FAILURE ALARM – FAL(06) and SEVERE FAILURE ALARM – FALS(07) 208
5-23-2 DISPLAY MESSAGE – MSG(46) 209
5-23-3 BIT COUNTER – BCNT(67) 210
5-23-4 WATCHDOG TIMER REFRESH– WDT(94) 211
5-23-5 I/O REFRESH – IORF(97) 211
5-24 Data Tracing (TRACE MEMORY SAMPLING – TRSM(45)) 211
5-25 File Memory Instructions 214
5-25-1 FILE MEMORY READ – FILR(42) 215
5-25-2 FILE MEMORY WRITE – FILW(43) 216
5-25-3 EXTERNAL PROGRAM READ – FILP(44) 216
5-26 Intelligent I/O Instructions 217
5-26-1 I/O WRITE – WRIT(87) 218
5-26-2 I/O READ – READ(88) 218
5-27 Network Instructions 219
5-27-1 NETWORK SEND – SEND(90) 219
5-27-2 NETWORK RECEIVE – RECV(98) 221
5-27-3 About SYSMAC NET Link/SYSMAC LINK Operations 222

5-1 Notation

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

If an instruction is assigned a function code, it will be given in parentheses after the mnemonic. These function codes, which are 2-digit decimal numbers, are used to input most instructions into the CPU and are described briefly below and in more detail in *7-4 Inputting, Modifying, and Checking the Program*. A table of instructions listed in order of function codes, is also provided in *Appendix B*.

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

5-2 Instruction Format

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

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

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

5-3 Data Areas, Definer Values, and Flags

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

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

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

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

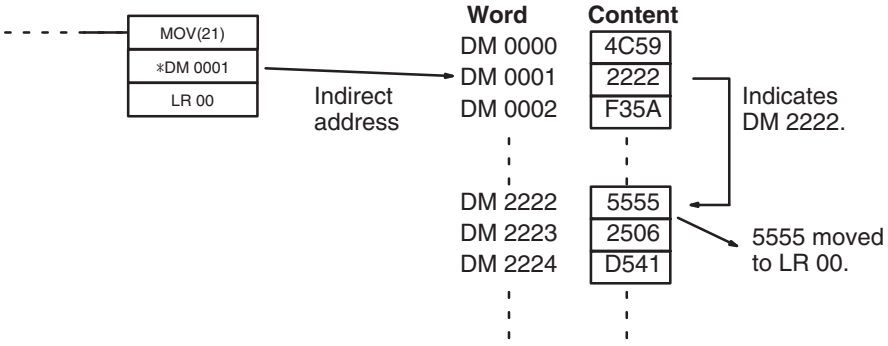
Abbreviation	Name	Bit
ER	Instruction Execution Error Flag	25503
CY	Carry Flag	25504
GR	Greater Than Flag	25505
EQ	Equals Flag	25506
LE	Less Than Flag	25507

ER is the flag most commonly used for monitoring an instruction's execution. When ER goes ON, it indicates that an error has occurred in attempting to execute the current instruction. The *Flags* subsection of each instruction lists possible reasons for ER being ON. ER will turn ON if operands are not entered correctly. Instructions are not executed when ER is ON. A table of instructions and the flags they affect is provided in *Appendix D Error and Arithmetic Flag Operation*.

Indirect Addressing

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

When an indirect DM address is specified, the designated DM word will contain the address of the DM word that contains the data that will be used as the operand of the instruction. If, for example, *DM 0001 was designated as the first operand and LR 00 as the second operand of MOV(21), the contents of DM 0001 was 2222, and DM 2222 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 4095 for the C1000H, and between 0000 and 6655 for the C2000H).

Designating Constants

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

5-4 Differentiated Instructions

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

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

Diagram A

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

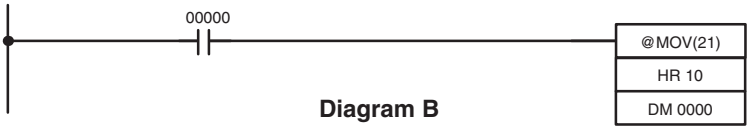


Diagram B

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

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

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

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

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

The C1000H and C2000H also provide differentiation instructions: DIFU(13) and DIFD(14). DIFU(13) operates the same as a differentiated instruction, but is used to turn ON a bit for one cycle. DIFD(14) also turns ON a bit for one cycle, but does it when the execution condition has changed from ON to OFF. Refer to *5-6-2 DIFFERENTIATE UP and DIFFERENTIATE DOWN – DIFU(13) and DIFD(14)* for details.

5-5 Coding Right-hand Instructions

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

gram symbol for all other instructions follows the same pattern, as described below, and is not specified for each instruction individually.

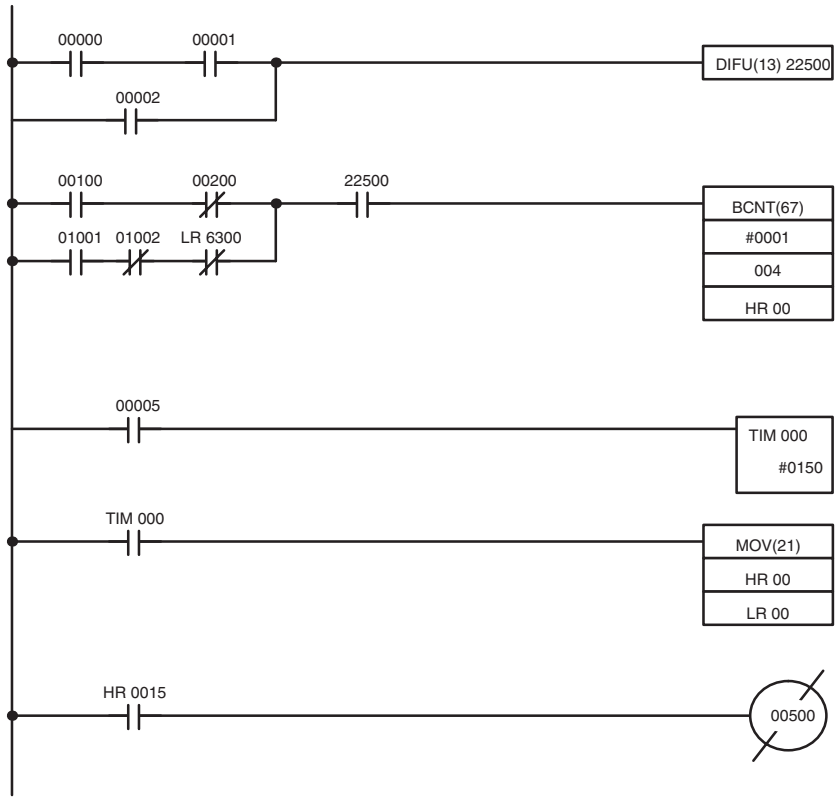
The first word of any instruction defines the instruction and provides any definers. If the instruction requires only a signal bit operand with no definer, the bit operand is also placed on the same line as the mnemonic. All other operands are placed on lines after the instruction line, one operand per line and in the same order as they appear in the ladder symbol for the instruction.

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

If an IR or SR address is used in the data column, the left side of the column is left blank. If any other data area is used, the data area abbreviation is placed on the left side and the address is place on the right side. If a constant to be input, the number symbol (#) is placed on the left side of the data column and the number to be input is placed on the right side. Any numbers input as definers in the instruction word do not require the number symbol on the right side. TC bits, once defined as a timer or counter, take a TIM (timer) or CNT (counter) prefix.

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

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

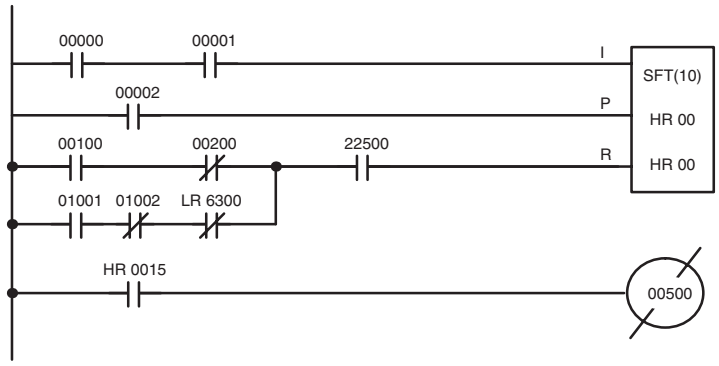


Address	Instruction	Data	
00000	LD		00000
00001	AND		00001
00002	OR		00002
00003	DIFU(13)		22500
00004	LD		00100
00005	AND NOT		00200
00006	LD		01001
00007	AND NOT		01002
00008	AND NOT	LR	6300
00009	OR LD		--
00010	AND		22500
00011	BCNT(67)		--
		#	0001
			004
		HR	00
00012	LD		00005
00013	TIM		000
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

Multiple Instruction Lines

If a right-hand instruction requires multiple instruction lines (such as KEEP(11)), all of the lines for the instruction are entered before the right-hand instruction. Each of the lines for the instruction is coded, starting with

LD or LD NOT, to form 'logic blocks' that are combined by the right-hand instruction. An example of this for SFT(10) is shown below.



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

Block Instructions

Block instructions are coded directly after BPRG(96) in the same order as written. Each address takes one instruction, and each of the block instruction lines requires one word (i.e., one line) in the mnemonic code table. Operand bits, operand words, and definers for block instruction are also coded in the same way as any other instruction. Refer to *5-21 Block Programming Instructions* for details.

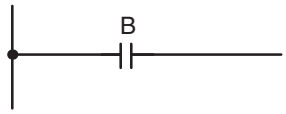
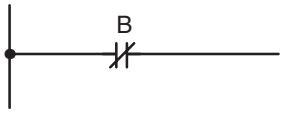
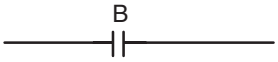
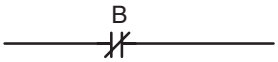
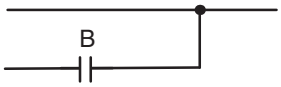
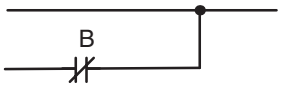
END(01)

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

5-6 Ladder Diagram Instructions

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

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

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

Limitations

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

Description

These six basic instructions correspond to the conditions on a ladder diagram. As described in *Section 4 Programming*, the status of the bits assigned to each instruction determines the execution conditions for all other instructions. Each of these instructions and each bit address can be used as many times as required. Each can be used in as many of these instructions as required.

The status of the bit operand (B) assigned to LD or LD NOT determines the first execution condition. AND takes the logical AND between the execution condition and the status of its bit operand; AND NOT, the logical AND between the execution condition and the inverse of the status of its bit operand. OR takes the logical OR between the execution condition and the status of its bit operand; OR NOT, the logical OR between the execution condition and the inverse of the status of its bit operand. The ladder symbol for loading TR bits is different from that shown above. Refer to *4-2-2 Ladder Instructions* for details.

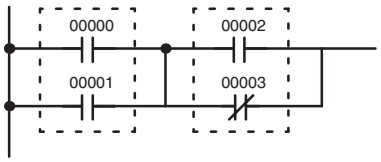
Flags

There are no flags affected by these instructions.

5-6-2 AND LOAD and OR LOAD

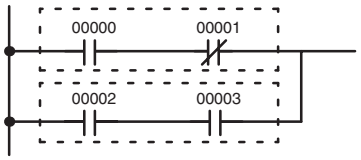
AND LOAD - AND LD

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 the GPC. They are required, however, to convert the program to and input it in mnemonic form.

In order to reduce the number of programming instructions required, a basic understanding of logic block instructions is required. For an introduction to logic blocks, refer to *4-2-3 Logic Block Instructions*. For details and examples, refer to *7-1-3 Logic Block Instructions*.

Flags

There are no flags affected by these instructions.

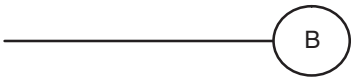
5-7 Bit Control Instructions

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

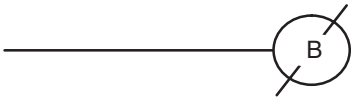


Operand Data Areas

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. Refer to *3-2 IR Area* for details and to *5-20 Block Instructions* for information on using output bits in SET<07> and RSET<08>.

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 *5-11-1 TIMER – TIM* for details.

Flags

There are no flags affected by these instructions.

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

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

Limitations

Any output bit can generally be used in only one instruction that controls its status. Refer to *3-2 IR Area* for details and to *5-20 Block Instructions* for information on using output bits in SET<07> and RSET<08>.

Description

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

Whenever executed, DIFU(13) compares its current execution with the previous execution condition. If the previous execution condition was OFF and the current one is ON, DIFU(13) will turn ON the designated bit. If the previous execution condition was ON and the current execution condition is either ON or OFF, DIFU(13) will either turn the designated bit OFF or leave it OFF (i.e., if the designated bit is already OFF). The designated bit will thus never be ON for longer than one cycle, assuming it is executed each cycle (see *Precautions*, below).

Whenever executed, DIFD(14) compares its current execution with the previous execution condition. If the previous execution condition is ON and the current one is OFF, DIFD(14) will turn ON the designated bit. If the previous execution condition was OFF and the current execution condition is either ON or OFF, DIFD(14) will either turn the designated bit OFF or leave it OFF. The designated bit will thus never be ON for longer than one cycle, assuming it is executed each cycle (see *Precautions*, below).

These instructions are used when differentiated instructions (i.e., those prefixed with an @) are not available and single-cycle execution of a particular instruction is desired. They can also be used with non-differentiated forms of instructions that have differentiated forms when their use will simplify programming. Examples of these are shown below.

Flags

There are no flags affected by these instructions.

Precautions

DIFU(13) and DIFD(14) operation can be uncertain when the instructions are programmed between IL and ILC, between JMP and JME, or in subroutines. Refer to *5-7 INTERLOCK and INTERLOCK CLEAR – IL(02) and ILC(03)*, *5-8 JUMP and JUMP END – JMP(04) and JME(05)*, and *5-19 Subroutines and Interrupt Control* for details.

Example 1: Use when There's No Differentiated Instruction

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

Diagram A

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

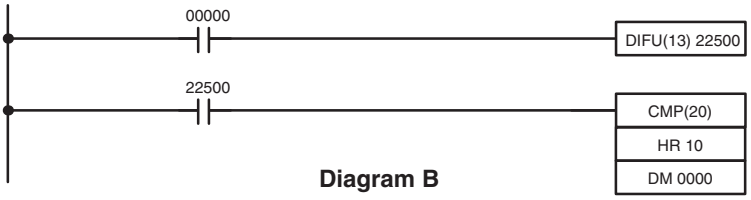
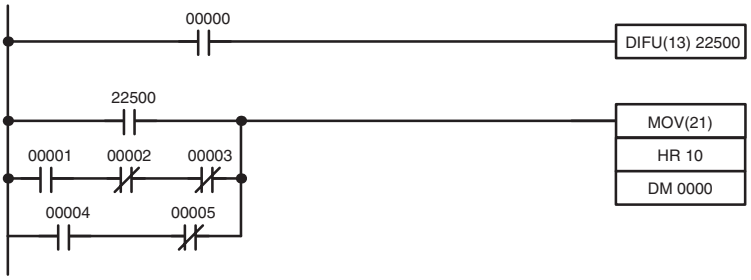


Diagram B

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

Example 2: Use to Simplify Programming

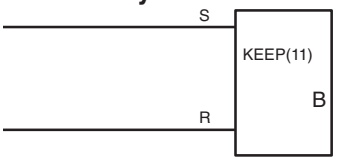
Although a differentiated form of MOV(21) is available, the following diagram would be very complicated to draw using it because only one of the conditions determining the execution condition for MOV(21) requires differentiated treatment.



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

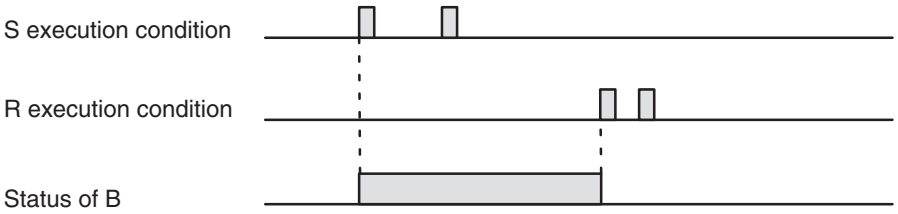
Limitations

Any output bit can generally be used in only one instruction that controls its status. Refer to *3-2 IR Area* for details and to *5-20 Block Instructions* for information on using output bits in SET<07> and RSET<08>.

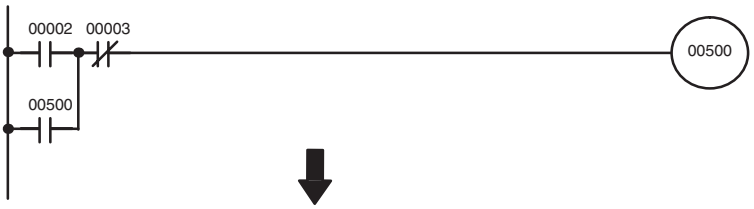
Description

KEEP(11) is used to maintain the status of the designated bit based on two execution conditions. These execution conditions are labeled S and R. S is the set input; R, the reset input. KEEP(11) operates like a latching relay that is set by S and reset by R.

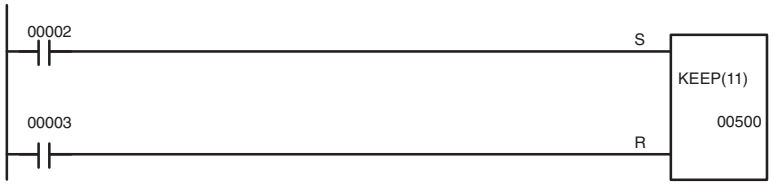
When S turns ON, the designated bit will go ON and stay ON until reset, regardless of whether S stays ON or goes OFF. When R turns ON, the designated bit will go OFF and stay OFF until reset, regardless of whether R stays ON or goes OFF. The relationship between execution conditions and KEEP(11) bit status is shown below.



KEEP(11) operates like the self-maintaining bit described in *4-7-3 Self-maintaining Bits (Seal)*. 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



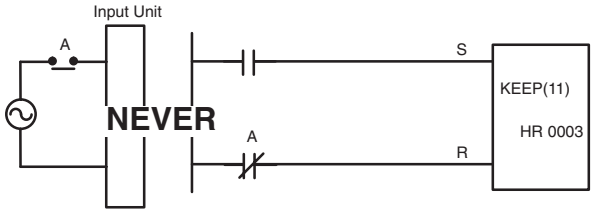
Address	Instruction	Operands
00000	LD	00002
00001	LD	00003
00002	KEEP(11)	00500

Flags

There are no flags affected by this instruction.

Precautions

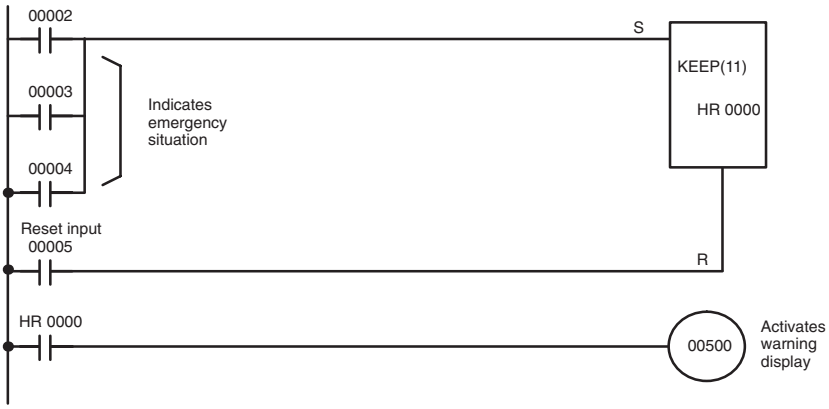
Never use an input bit in a normally closed condition on the reset (R) for KEEP(11) when the input device uses an AC power supply. The delay in shutting down the PC's DC power supply (relative to the AC power supply to the input device) can cause the designated bit of KEEP(11) to be reset. This situation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



Bits used in KEEP are not reset in interlocks. Refer to the *5-7 INTERLOCK and INTERLOCK CLEAR – IL(02) and ILC(03)* for details.

Example

If a HR bit or an AR bit is used, bit status will be retained even during a power interruption. KEEP(11) can thus be used to program bits that will maintain status after restarting the PC following a power interruption. An example of this that can be used to produce a warning display following a system shutdown for an emergency situation is shown below. Bits 00002, 00003, and 00004 would be turned ON to indicate some type of error. Bit 00005 would be turned ON to reset the warning display. HR 0000, which is turned ON when any one of the three bits indicates an emergency situation, is used to turn ON the warning indicator through 00500.

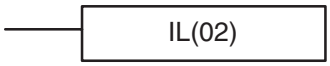


Address	Instruction	Operands
00000	LD	00002
00001	OR	00003
00002	OR	00004
00003	LD	00005
00004	KEEP(11)	HR 0000
00005	LD	HR 0000
00006	OUT	00500

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

5-8 INTERLOCK and INTERLOCK CLEAR – IL(02) and ILC(03)

Ladder Symbol



Ladder Symbol

Description

IL(02) is always used in conjunction with ILC(03) to create interlocks. Interlocks are used to enable branching in the same way as can be achieved with TR bits, but treatment of instructions between IL(02) and ILC(03) differs from that with TR bits when the execution condition for IL(02) is OFF. If the execution condition of IL(02) is ON, the program will be executed as written, with an ON execution condition used to start each instruction line from the point where IL(02) is located through the next ILC(03). Refer to *4-6-7 Branching Instruction Lines* for basic descriptions of both methods.

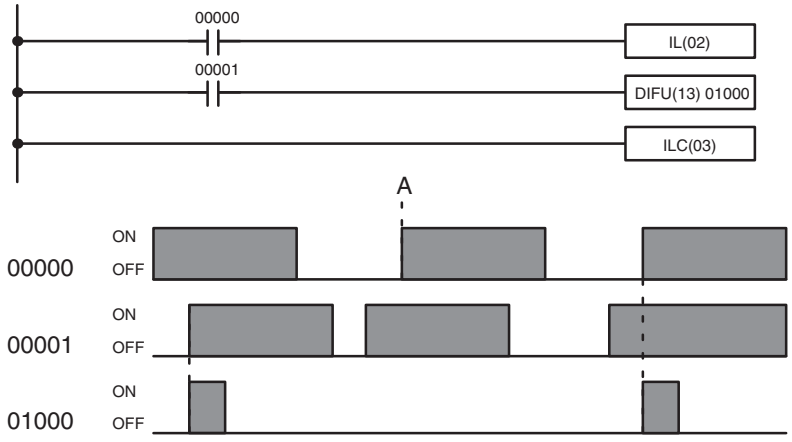
If the execution condition for IL(02) is OFF, the interlocked section between IL(02) and ILC(03) will be treated as shown in the following table:

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

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

DIFU(13) and DIFD(14) in Interlocks

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



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

Precautions

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

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

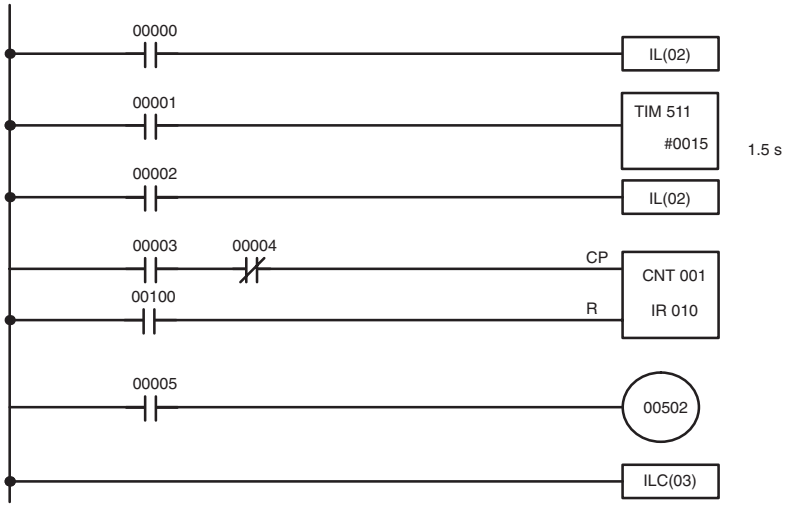
When more than one IL(02) is used with a single ILC(03), an error message will appear when the program check is performed, but execution will proceed normally.

Flags

There are no flags affected by these instructions.

Example

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



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

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

5-9 JUMP and JUMP END – JMP(04) and JME(05)

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

Limitations

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

Description

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

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

tween JMP(04) and JME(05) are skipped, jump numbers 01 through 99 can be used to reduce cycle time.

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

DIFU(13) and DIFD(14) in Jumps

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

Precautions

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

Flags

There are no flags affected by these instructions.

Examples

Examples of jump programs are provided in *4-6-8 Jumps*.

5-10 END – END(01)

Ladder Symbol

Description

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

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

Flags

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

5-11 NO OPERATION – NOP(00)

Description

NOP(00) is not generally required in programming and there is no ladder symbol for it. When NOP(00) is found in a program, nothing is executed and

the program execution moves to the next instruction. When memory is cleared prior to programming, NOP(00) is written at all addresses. NOP(00) can be input through the 00 function code.

Flags

There are no flags affected by NOP(00).

5-12 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 (including timer and counter block instructions discussed in *5-20 Block 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 counter wired in this way can only be set externally during RUN or MONITOR mode. All SVs, including those set externally, must be in BCD.

5-12-1 TIMER – TIM

Ladder Symbol

Definer Values

N: TC number
(000 through 511)

Operand Data Areas

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

Limitations

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

Each TC number can be used as the definer in only one timer or counter instruction.

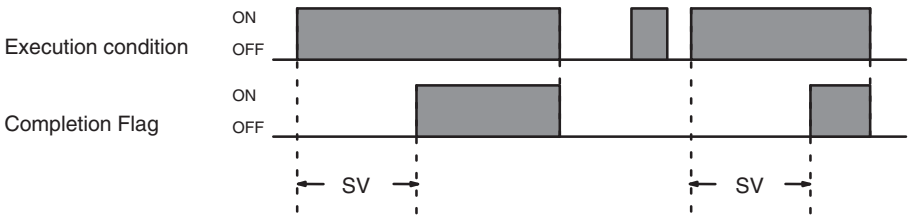
TC 000 through TC 047 should not be used in TIM if they are required for TIMH(15). Refer to *5-11-2 HIGH-SPEED TIMER – TIMH(15)* for details.

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 *5-11-3 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.)

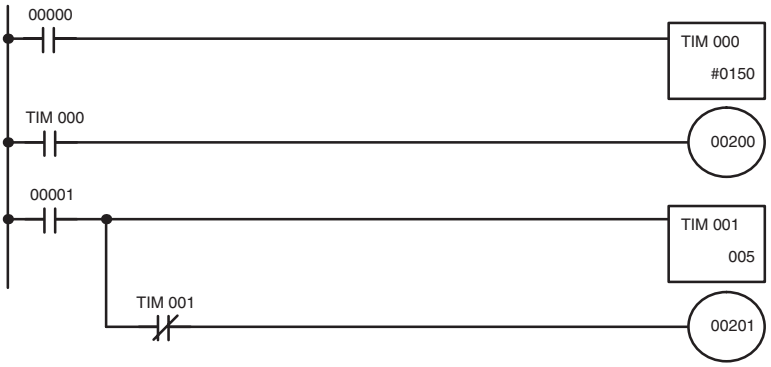
Examples

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

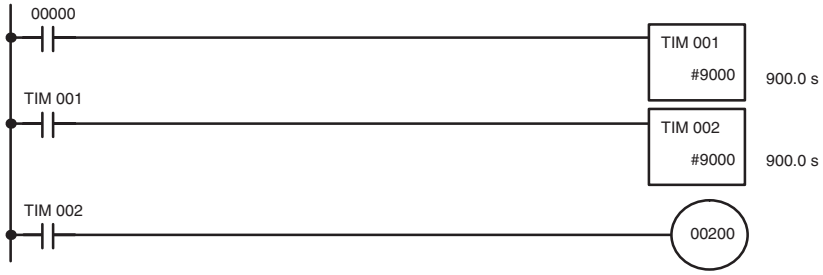
stays ON for at least 15 seconds. When 00000 goes OFF, the timer will be reset and 00200 will be turned OFF. When 00001 goes ON, TIM 001 is started from the SV provided through IR word 005. Bit 00201 is also turned ON when 00001 goes ON. When the SV in 005 has expired, 00201 is turned OFF. This bit will 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 *5-11-3 COUNTER – CNT*.

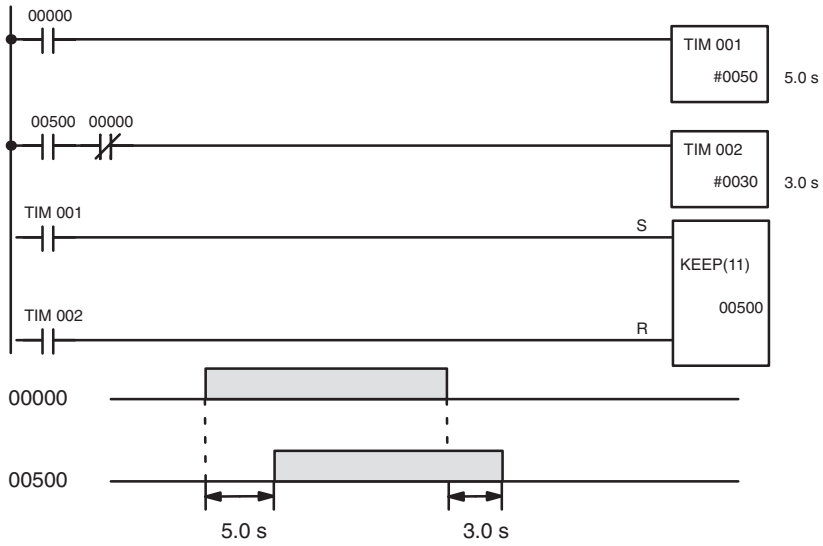
Example 3: ON/OFF Delays

TIM can be combined with KEEP(11) to delay turning a bit ON and OFF in reference to a desired execution condition. KEEP(11) is described in *5-6-3 KEEP – KEEP(11)*.

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

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

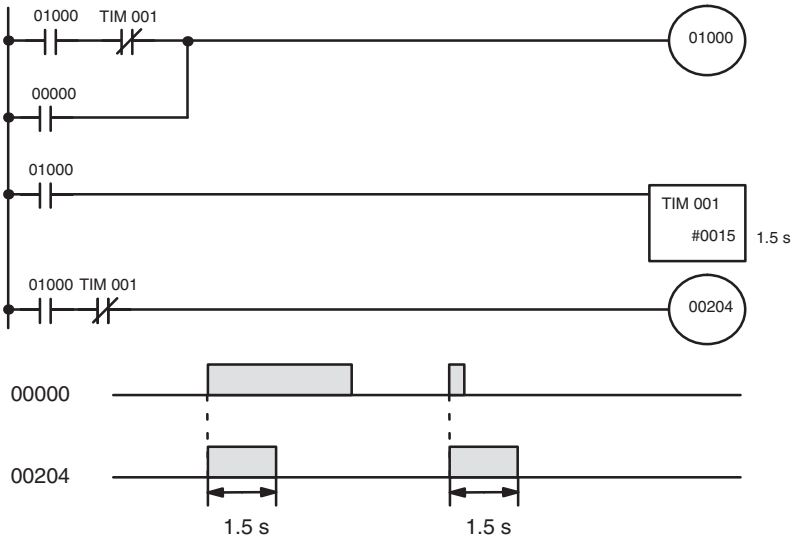
TIM 002 when 00000 goes ON and 00500 is necessary to activate TIM 002 (when 00000 is OFF).



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 a normally closed condition, creates an OFF execution condition for OUT 00204).

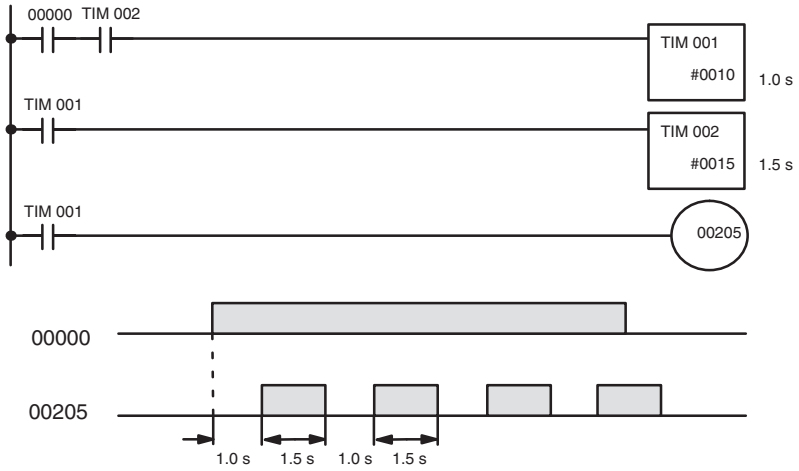


Address	Instruction	Operands
00000	LD	01000
00001	AND NOT	TIM 001
00002	OR	00000
00003	OUT	01000
00004	LD	01000
00005	TIM	001
		# 0015
00006	LD	01000
00007	AND NOT	TIM 001
00008	OUT	00204

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

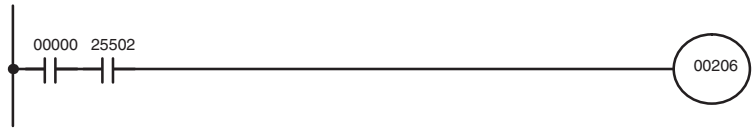
second TIM is started and when the second TIM's Completion Flag goes ON, the first TIM is started.



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	LD	25502
00002	OUT	00206

5-12-2 HIGH-SPEED TIMER – TIMH(15)

Ladder Symbol

Definer Values

N: TC number
(000 through 047 preferred)

Operand Data Areas

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

Limitations

SV is between 00.02 and 99.99. (Although 00.00 and 00.01 may be set, 00.00 will disable the timer, i.e., turn ON the Completion Flag immediately, and 00.01 is not reliably scanned.) The decimal point is not entered.

Each TC number can be used as the definer in only one timer or counter instruction.

Although TC 048 through TC 511 can be programmed, TC 000 through TC 047 must be used to ensure accuracy if the cycle time is greater than 10 ms.

With a C2000H Duplex System, TC 048 through TC 511 cannot be used for TIMH(15).

Description

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

The cycle time affects TIMH(15) accuracy if TC 048 through TC 511 are used. If the cycle time is greater than 10 ms, use TC 000 through TC 047.

Refer to *5-11-1 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 *5-11-3 COUNTER – CNT* for details.

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

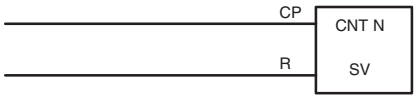
Flags

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.)

5-12-3 COUNTER – CNT

Ladder Symbol



Definer Values

N: TC number
(000 through 511)

Operand Data Areas

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

Limitations

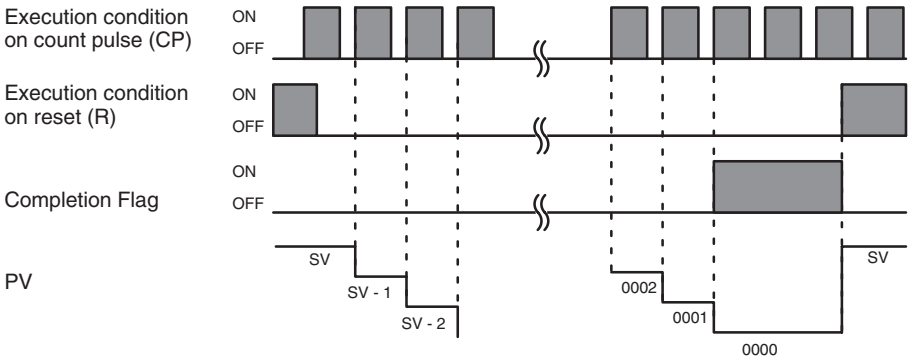
Each TC number can be used as the definer in only one timer or counter instruction.

Description

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

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

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



Precautions

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

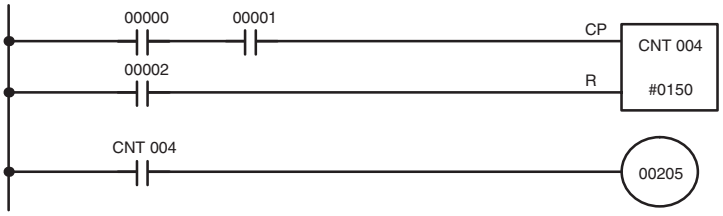
Flags

ER: SV is not in BCD.

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

Example 1: Basic Application

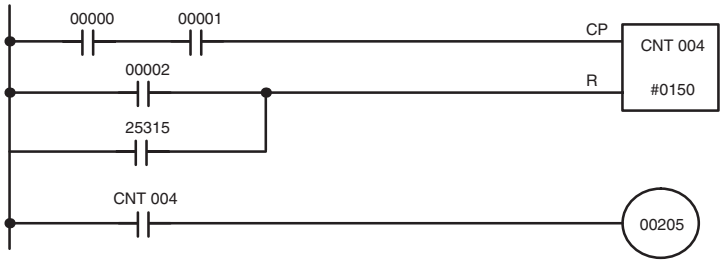
In the following example, the PV will be decremented whenever both 00000 and 00001 are ON provided that 00002 is OFF and either 00000 or 00001 was OFF the last time CNT 004 was executed. When 150 pulses have been counted down (i.e., when PV reaches zero), 00205 will be turned ON.



Address	Instruction	Operands
00000	LD	00000
00001	AND	00001
00002	LD	00002
00003	CNT	004
		# 0150
00004	LD	CNT 004
00005	OUT	00205

Here, 00000 can be used to control when CNT is operative and 00001 can be used as the bit whose OFF to ON changes are being counted.

The above CNT can be modified to restart from SV each time power is turned ON to the PC. This is done by using the First Cycle Flag in the SR area (25315) to reset CNT as shown below.



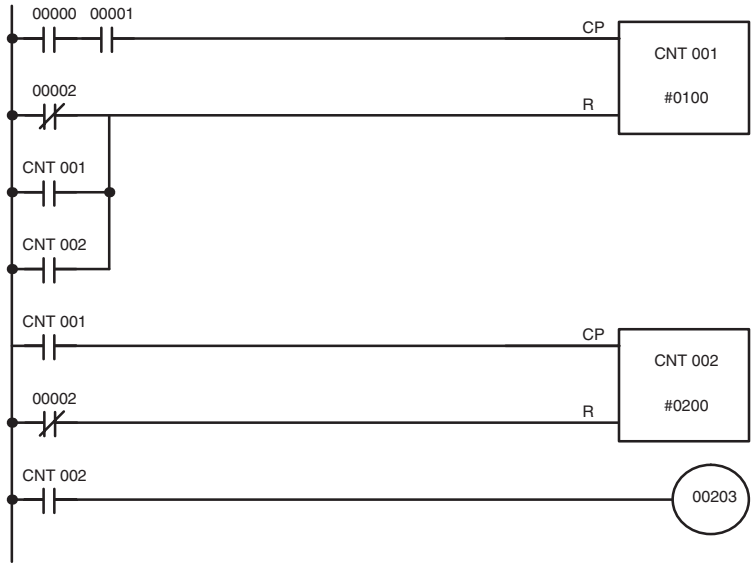
Address	Instruction	Operands
00000	LD	00000
00001	AND	00001
00002	LD	00002
00003	OR	25315
00004	CNT	004
		# 0150
00005	LD	CNT 004
00006	OUT	00205

Example 2: Extended Counter

Counters that can count past 9,999 can be programmed by using one CNT to count the number of times another CNT has counted to zero from SV.

In the following example, 00000 is used to control when CNT 001 operates. CNT 001, when 00000 is ON, counts down the number of OFF to ON changes in 00001. CNT 001 is reset by its Completion Flag, i.e., it starts counting again as soon as its PV reaches zero. CNT 002 counts the number of times the Completion Flag for CNT 001 goes ON. Bit 00002 serves as a reset for the entire extended counter, resetting both CNT 001 and CNT 002 when it is OFF. The Completion Flag for CNT 002 is also used to reset CNT 001 to inhibit CNT 001 operation, once SV for CNT 002 has been reached, until the entire extended counter is reset via 00002.

Because in this example the SV for CNT 001 is 100 and the SV for CNT 002 is 200, the Completion Flag for CNT 002 turns ON when 100 x 200 or 20,000 OFF to ON changes have been counted in 00001. This would result in 00203 being turned ON.



Address	Instruction	Operands	
00000	LD		00000
00001	AND		00001
00002	LD NOT		00002
00003	OR	CNT	001
00004	OR	CNT	002
00005	CNT		001
		#	0100
00006	LD	CNT	001
00007	LD NOT		00002
00008	CNT		002
		#	0200
00009	LD	CNT	002
00010	OUT		00203

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

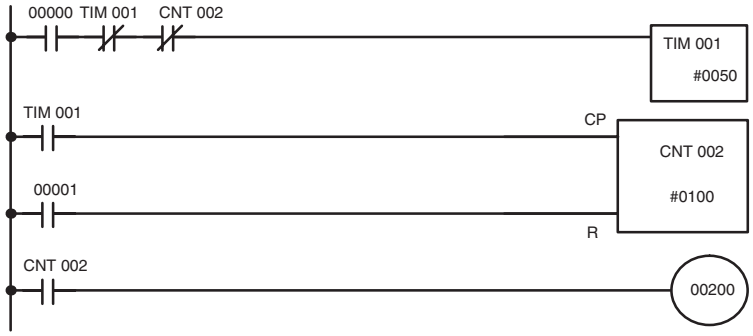
Example 3: Extended Timers

CNT can be used to create extended timers in two ways: by combining TIM with CNT and by counting SR area clock pulse bits.

In the following example, CNT 002 counts the number of times TIM 001 reaches zero from its SV. The Completion Flag for TIM 001 is used to reset TIM 001 so that is runs continuously and CNT 002 counts the number of times the Completion Flag for TIM 001 goes ON (CNT 002 would be executed once each time between when the Completion Flag for TIM 001 goes ON and TIM 001 is reset by its Completion Flag). TIM 001 is also reset by the Completion Flag for CNT 002 so that the extended timer would not start again until CNT 002 was reset by 00001, which serves as the reset for the entire extended timer.

Because in this example the SV for TIM 001 is 5.0 seconds and the SV for CNT 002 is 100, the Completion Flag for CNT 002 turns ON when 5 seconds

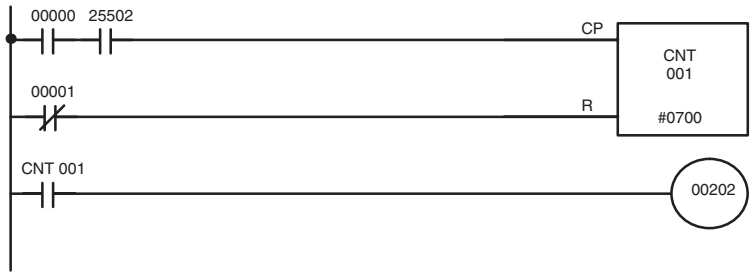
x 100 times, i.e., 500 seconds (or 8 minutes and 20 seconds) have expired. This would result in 00201 being turned ON.



Address	Instruction	Operands	
00000	LD		00000
00001	AND NOT	TIM	001
00002	AND NOT	CNT	002
00003	TIM		001
		#	0050
00004	LD	TIM	001
00005	LD		00001
00006	CNT		002
		#	0100
00007	LD	CNT	002
00008	OUT		00200

In the following example, CNT 001 counts the number of times the 1-second clock pulse bit (25502) goes from OFF to ON. Here again, 00000 is used to control the times when CNT is operating.

Because in this example the SV for CNT 001 is 700, the Completion Flag for CNT 002 turns ON when 1 second x 700 times, or 11 minutes and 40 seconds have expired. This would result in 00202 being turned ON.

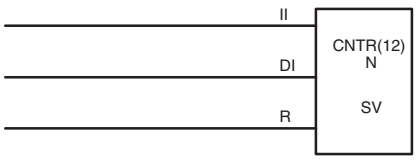


Address	Instruction	Operands	
00000	LD		00000
00001	AND		25502
00002	LD NOT		00001
00003	CNT		001
		#	0700
00004	LD	CNT	001
00005	OUT		00202

⚠ Caution The shorter clock pulses will not necessarily produce accurate timers because their short ON times might not be read accurately during longer cycles. In particular, the 0.02-second and 0.1-second clock pulses should not be used to create timers with CNT instructions.

5-12-4 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).

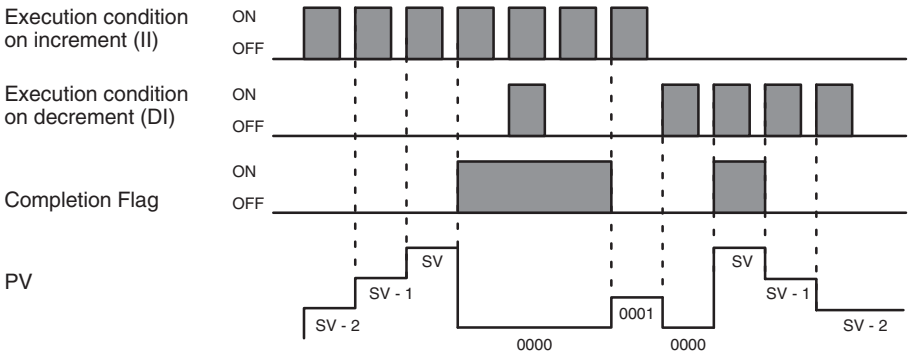
The present value (PV) will be incremented by one whenever CNTR(12) is executed with an ON execution condition for II and the last execution condition for II was OFF. The present value (PV) will be decremented by one whenever CNTR(12) is executed with an ON execution condition for DI and the last execution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

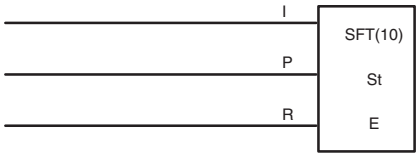
Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

5-13 Data Shifting

All of the instructions described in this section are used to shift data, but in differing amounts and directions. The first shift instruction, SFT(10), shifts an execution condition into a shift register; the rest of the instructions shift data that is already in memory.

5-13-1 SHIFT REGISTER – SFT(10)

Ladder Symbol



Operand Data Areas

St: Starting word
IR, AR, HR, LR

E: End word
IR, AR, HR, LR

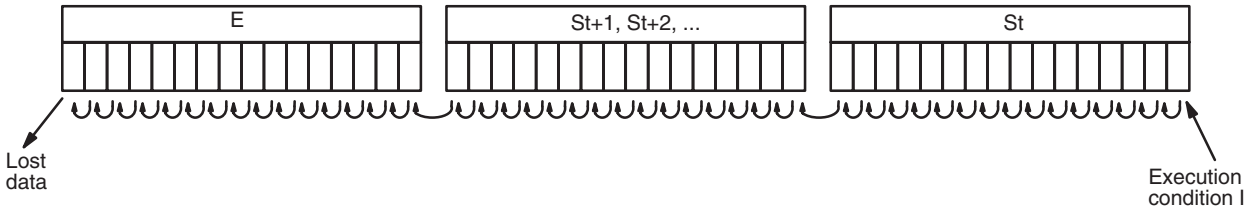
Limitations

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

If a bit address in one of the words used in a shift register is also used in an instruction that controls individual bit status (e.g., OUT, KEEP(11), SET<07>), an error ("COIL DUPL") will be generated when program syntax is checked on the Programming Console or another Programming Device. The program, however, will be executed as written. See *Example 2: Controlling Bits in Shift Registers* for a programming example that does this.

Description

SFT(10) is controlled by three execution conditions, I, P, and R. If SFT(10) is executed and 1) execution condition P is ON and was OFF the last execution and 2) R is OFF, then execution condition I is shifted into the rightmost bit of a shift register defined between St and E, i.e., if I is ON, a 1 is shifted into the register; if I is OFF, a 0 is shifted in. When I is shifted into the register, all bits previously in the register are shifted to the left and the leftmost bit of the register is lost.



The execution condition on P functions like a differentiated instruction, i.e., I will be shifted into the register only when P is ON and was OFF the last time SFT(10) was executed. If execution condition P has not changed or has gone from ON to OFF, the shift register will remain unaffected.

St designates the rightmost word of the shift register; E designates the leftmost. The shift register includes both of these words and all words between them. The same word may be designated for St and E to create a 16-bit (i.e., 1-word) shift register.

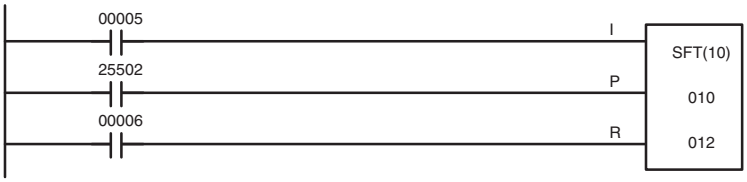
When execution condition R goes ON, all bits in the shift register will be turned OFF (i.e., set to 0) and the shift register will not operate until R goes OFF again.

Flags

There are no flags affected by SFT(10).

Example 1: Basic Application

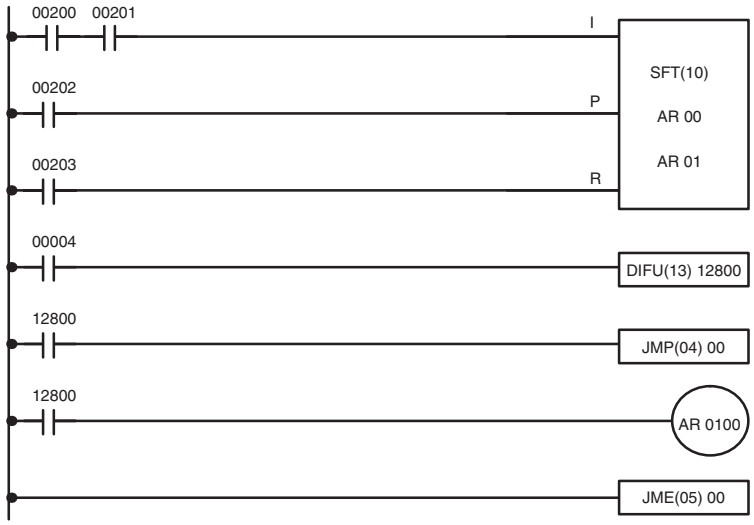
The following example uses the 1-second clock pulse bit (25502) so that the execution condition produced by 00005 is shifted into a 3-word register between IR 010 and IR 012 every second.



Address	Instruction	Operands
00000	LD	00005
00001	LD	25502
00002	LD	00006
00003	SFT(10)	
		010
		012

Example 2: Controlling Bits in Shift Registers

The following program is used to control the status of the 17th bit of a shift register running from AR 00 through AR 01. When the 17th bit is to be set, 00004 is turned ON. This causes the jump for JMP(04) 00 not to be made for that one cycle, and AR 0100 (the 17th bit) will be turned ON. When 12800 is OFF (i.e., at all times except during the first cycle after 00004 has changed from OFF to ON), the jump is executed and the status of AR 0100 will not be changed.



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

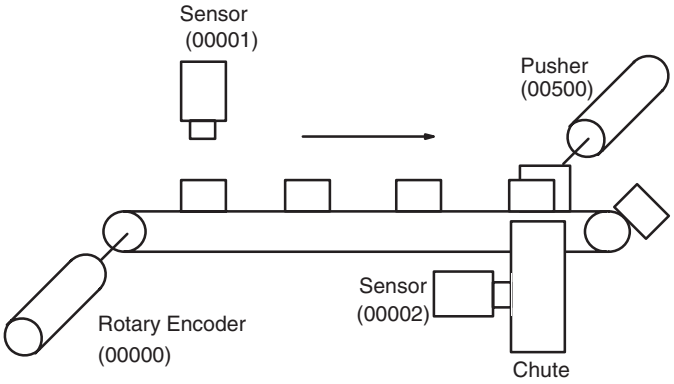
When a bit that is part of a shift register is used in OUT (or any other instruction that controls bit status), a syntax error will be generated during the program check, but the program will executed properly (i.e., as written).

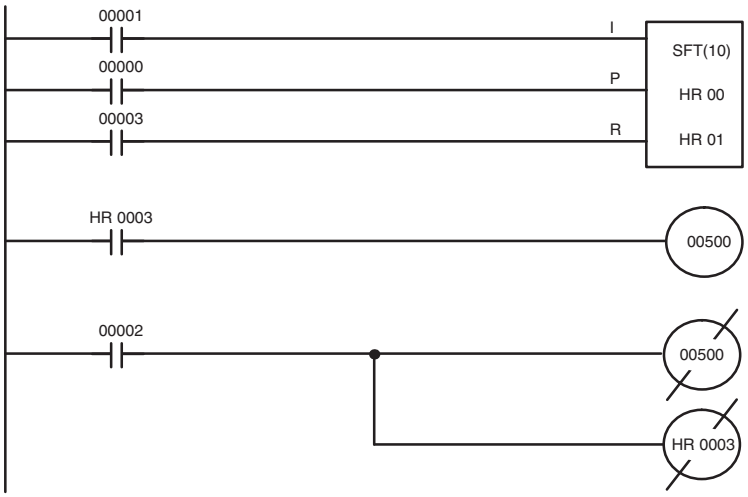
Example 3: Control Action

The following program controls the conveyor line shown below so that faulty products detected at the sensor are pushed down a chute. To do this, the execution condition determined by inputs from the first sensor (00001) are stored in a shift register: ON for good products; OFF for faulty ones. Conveyor speed has been adjusted so that HR 0003 of the shift register can be used to activate a pusher (00500) when a faulty product reaches it, i.e., when HR 0003 turns ON, 00500 is turned ON to activate the pusher.

The program is set up so that a rotary encoder (00000) controls execution of SFT(10) through a DIFU(13), the rotary encoder is set up to turn ON and OFF each time a product passes the first sensor. Another sensor (00002) is

used to detect faulty products in the chute so that the pusher output and HR 0003 of the shift register can be reset as required.





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

5-13-2 REVERSIBLE SHIFT REGISTER – SFTR(84)

Ladder Symbols

SFTR(84)
C
St
E

@SFTR(84)
C
St
E

Operand Data Areas

C: Control word
IR, AR, DM, HR, LR

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

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

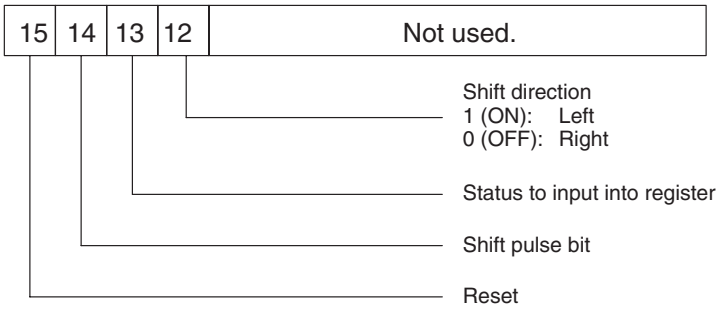
Limitations

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

Description

SFTR(84) is used to create a single- or multiple-word shift register that can shift data to either the right or the left. To create a single-word register, designate the same word for St and E. The control word provides the shift direc-

tion, the status to be put into the register, the shift pulse, and the reset input. The control word is allocated as follows:



The data in the shift register will be shifted one bit in the direction indicated by bit 12, shifting one bit out to CY and the status of bit 13 into the other end whenever SFTR(84) is executed with an ON execution condition as long as the reset bit is OFF and as long as bit 14 is ON. If SFTR(84) is executed with an OFF execution condition or if SFTR(84) is executed with bit 14 OFF, the shift register will remain unchanged. If SFTR(84) is executed with an ON execution condition and the reset bit (bit 15) is OFF, the entire shift register and CY will be set to zero.

Flags

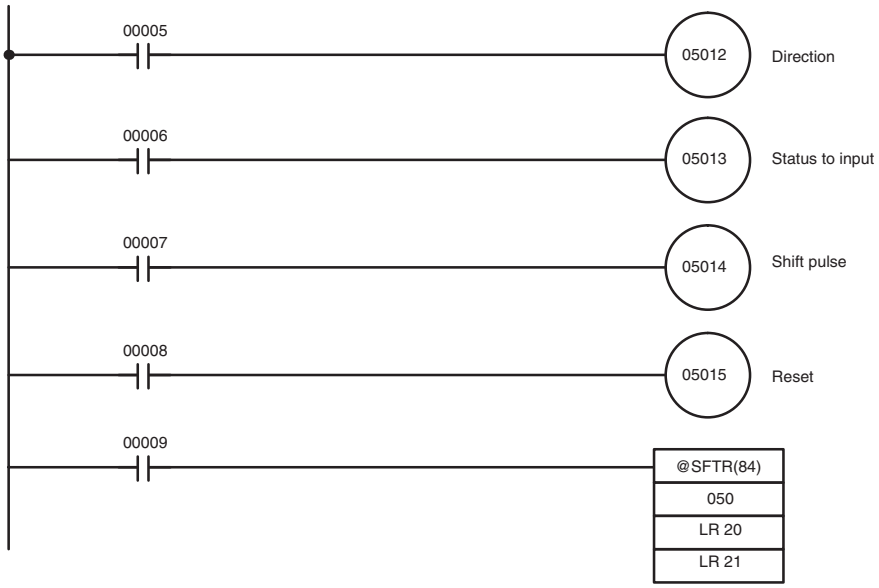
ER: Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

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

CY: Receives the status of bit 00 of St or bit 15 of E, depending on the shift direction.

Example

In the following example, IR 00005, IR 00006, IR 00007, and IR 00008 are used to control the bits of C used in @SFTR(84). The shift register is between LR 20 and LR 21, and it is controlled through IR 00009.

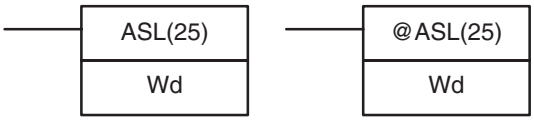


Address	Instruction	Operands
00000	LD	00005
00001	OUT	05012
00002	LD	00006
00003	OUT	05013
00004	LD	00007
00005	OUT	00514
00006	LD	00008

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

5-13-3 ARITHMETIC SHIFT LEFT – ASL(25)

Ladder Symbols

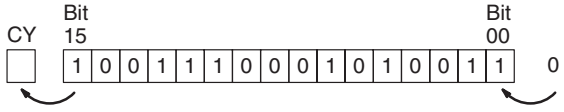


Operand Data Areas

Wd: Shift word
IR, AR, DM, HR, LR

Description

When the execution condition is OFF, ASL(25) is not executed. When the execution condition is ON, ASL(25) shifts a 0 into bit 00 of Wd, shifts the bits of Wd one bit to the left, and shifts the status of bit 15 into CY.



Flags

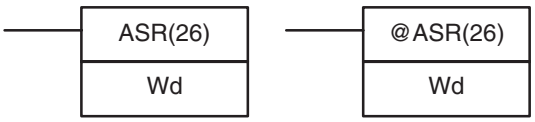
ER: Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

CY: Receives the status of bit 15.

EQ: ON when the content of Wd is 0; otherwise OFF.

5-13-4 ARITHMETIC SHIFT RIGHT – ASR(26)

Ladder Symbols

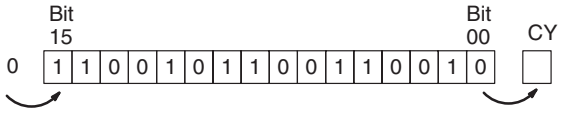


Operand Data Areas

Wd: Shift word
IR, AR, DM, HR, LR

Description

When the execution condition is OFF, ASR(25) is not executed. When the execution condition is ON, ASR(25) shifts a 0 into bit 15 of Wd, shifts the bits of Wd one bit to the right, and shifts the status of bit 00 into CY.



Flags

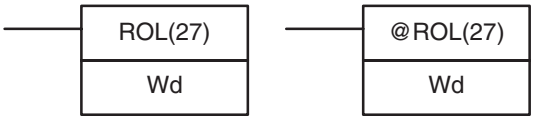
ER: Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

CY: Receives the data of bit 00.

EQ: ON when the content of Wd is 0; otherwise OFF.

5-13-5 ROTATE LEFT – ROL(27)

Ladder Symbols

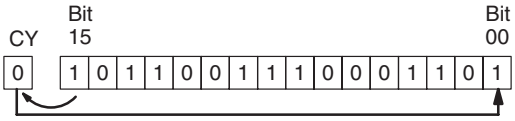


Operand Data Areas

Wd: Rotate word
IR, AR, DM, HR, LR

Description

When the execution condition is OFF, ROL(27) is not executed. When the execution condition is ON, ROL(27) shifts all Wd bits one bit to the left, shifting CY into bit 00 of Wd and shifting bit 15 of Wd into CY.



Precautions

Use STC(41) to set the status of CY or CLC(41) to clear the status of CY before doing a rotate operation to ensure that CY contains the proper status before execution ROL(27).

Flags

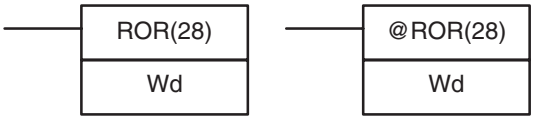
ER: Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

CY: Receives the data of bit 15.

EQ: ON when the content of Wd is 0; otherwise OFF.

5-13-6 ROTATE RIGHT – ROR(28)

Ladder Symbols

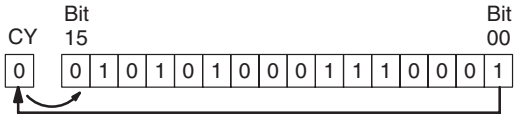


Operand Data Areas

Wd: Rotate word
IR, AR, DM, HR, LR

Description

When the execution condition is OFF, ROR(28) is not executed. When the execution condition is ON, ROR(28) shifts all Wd bits one bit to the right, shifting CY into bit 15 of Wd and shifting bit 00 of Wd into CY.



Precautions

Use STC(41) to set the status of CY or CLC(41) to clear the status of CY before doing a rotate operation to ensure that CY contains the proper status before execution ROR(28).

Flags

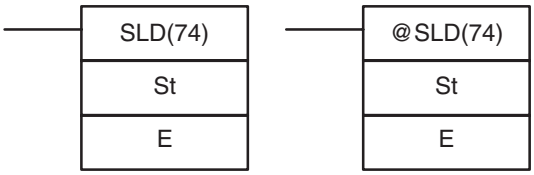
ER: Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

CY: Receives the data of bit 15.

EQ: ON when the content of Wd is 0; otherwise OFF.

5-13-7 ONE DIGIT SHIFT LEFT – SLD(74)

Ladder Symbols



Operand Data Areas

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

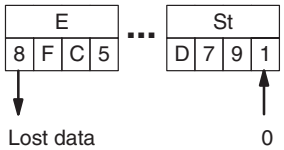
E: End word
IR, AR, DM, HR, LR

Limitations

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

Description

When the execution condition is OFF, SLD(74) is not executed. When the execution condition is ON, SLD(74) shifts data between St and E (inclusive) by one digit (four bits) to the left. 0 is written into the rightmost digit of the St, and the content of the leftmost digit of E is lost.

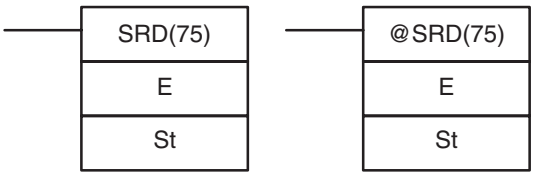


Precautions If a power failure occurs during a shift operation across more than 50 words, the shift operation might not be completed.

Flags **ER:** The St and E words are in different areas, or St is greater than E.
Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

5-13-8 ONE DIGIT SHIFT RIGHT – SRD(75)

Ladder Symbols



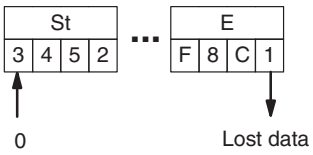
Operand Data Areas

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

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

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

Description When the execution condition is OFF, SRD(75) is not executed. When the execution condition is ON, SRD(75) shifts data between St and E (inclusive) by one digit (four bits) to the right. 0 is written into the leftmost digit of St and the rightmost digit of E is lost.

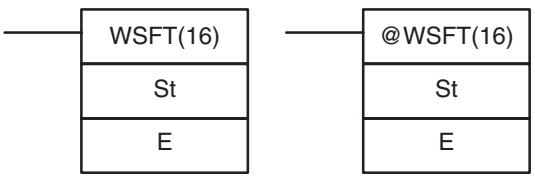


Precautions If a power failure occurs during a shift operation across more than 50 words, the shift operation might not be completed.

Flags **ER:** The St and E words are in different areas, or St is greater than E.
Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

5-13-9 WORD SHIFT – WSFT(16)

Ladder Symbols



Operand Data Areas

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

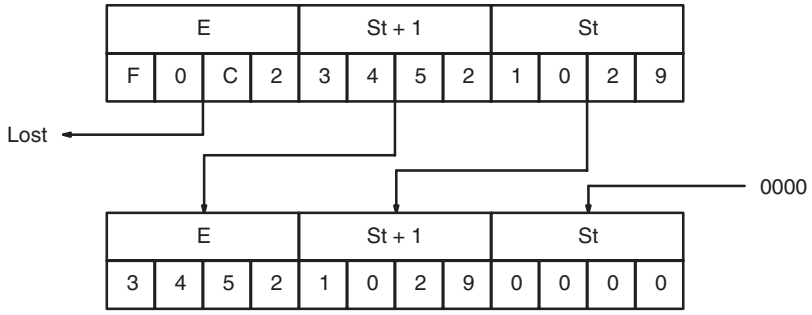
E: End word
IR, AR, DM, HR, LR

Limitations

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

Description

When the execution condition is OFF, WSFT(16) is not executed. When the execution condition is ON, WSFT(16) shifts data between St and E in word units. Zeros are written into St and the content of E is lost.



Flags

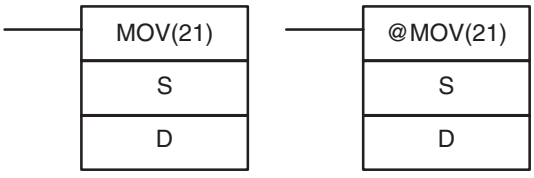
ER: The St and E words are in different areas, or St is greater than E.
Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

5-14 Data Movement

This section describes the instructions used for moving data between different addresses in data areas. These movements can be programmed to be within the same data area or between different data areas. Data movement is essential for utilizing all of the data areas of the PC. Effective communications in Link Systems also requires data movement. All of these instructions change only the content of the words to which data is being moved, i.e., the content of source words is the same before and after execution of any of the data movement instructions.

5-14-1 MOVE – MOV(21)

Ladder Symbols



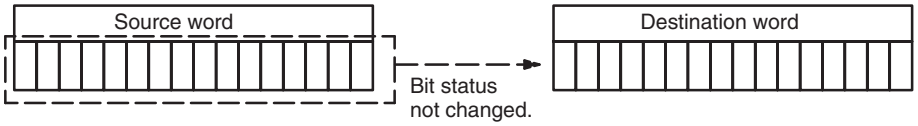
Operand Data Areas

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

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

Description

When the execution condition is OFF, MOV(21) is not executed. When the execution condition is ON, MOV(21) copies the content of S to D.



Precautions

TC numbers cannot be designated as D to change the PV of the timer or counter. However, these can be easily changed using BSET(71).

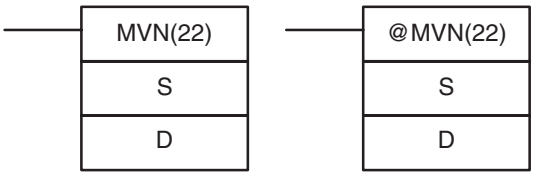
Flags

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

EQ: ON when all zeros are transferred to D.

5-14-2 MOVE NOT – MVN(22)

Ladder Symbols



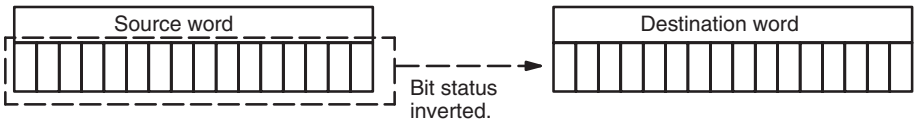
Operand Data Areas

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

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

Description

When the execution condition is OFF, MVN(22) is not executed. When the execution condition is ON, MVN(22) transfers the complement of the content of S (specified word or four-digit hexadecimal constant) to D, i.e., for each ON bit in S, the corresponding bit in D is turned OFF, and for each OFF bit in S, the corresponding bit in D is turned ON.



Precautions

TC numbers cannot be designated as D to change the PV of the timer or counter. However, these can be easily changed using BSET(71).

Flags

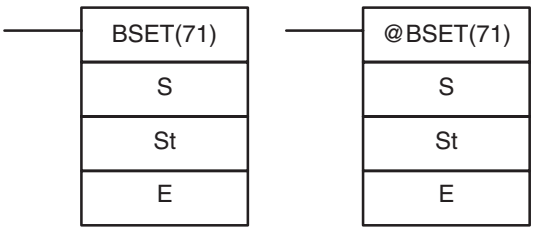
ER: Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON when all zeros are transferred to D.

5-14-3 BLOCK SET – BSET(71)

Operand Data Areas

Ladder Symbols



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

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

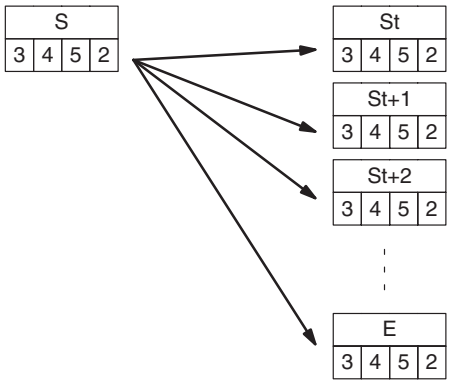
E: End Word
IR, AR, DM, HR, TC, LR

Limitations

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

Description

When the execution condition is OFF, BSET(71) is not executed. When the execution condition is ON, BSET(71) copies the content of S to all words from St through E.



BSET(71) can be used to change timer/counter PV. (This cannot be done with MOV(21) or MVN(22).) BSET(71) can also be used to clear sections of a data area, i.e., the DM area, to prepare for executing other instructions.

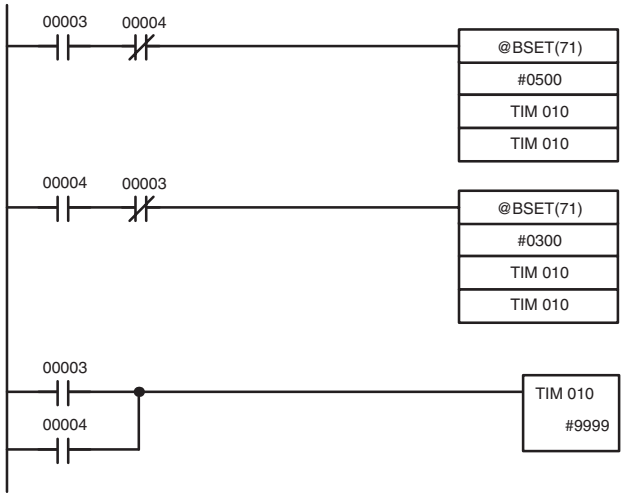
Flags

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

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

Example

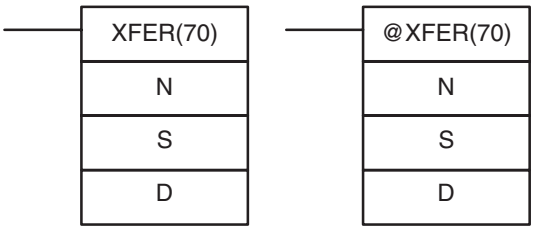
The following example shows how to use BSET(71) to change the PV of a timer depending on the status of IR 00003 and IR 00004. When IR 00003 is ON, TIM 010 will operate as a 50-second timer; when IR 00004 is ON, TIM 010 will operate as a 30-second timer.



Address	Instruction	Operands	
00000	LD		00003
00001	AND NOT		00004
00002	@BSET(71)		
		#	0500
		TIM	010
		TIM	010
00003	LD		00004
00004	AND NOT		00003
00005	@BSET(71)		
		#	0300
		TIM	010
		TIM	010
00006	LD		00003
00007	OR		00004
00008	TIM		010
		#	9999

5-14-4 BLOCK TRANSFER – XFER(70)

Ladder Symbols



Operand Data Areas

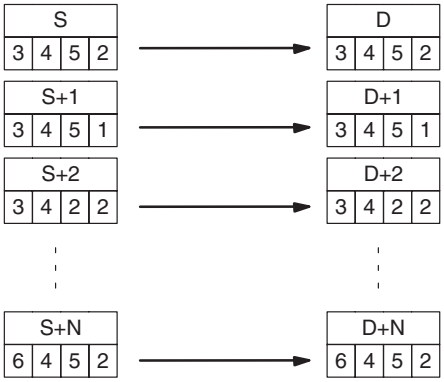
N: Number of words (BCD)
IR, SR, AR, DM, HR, TC, LR, #
S: Starting source word
IR, SR, AR, DM, HR, TC, LR
D: Starting destination word
IR, AR, DM, HR, TC, LR

Limitations

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

Description

When the execution condition is OFF, XFER(70) is not executed. When the execution condition is ON, XFER(70) copies the contents of S, S+1, ..., S+N to D, D+1, ..., D+N.



Flags

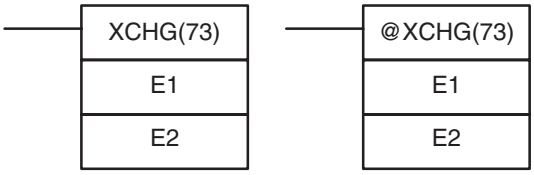
ER: N is not BCD

S and S+N or D and D+N are not in the same data area.

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

5-14-5 DATA EXCHANGE – XCHG(73)

Ladder Symbols

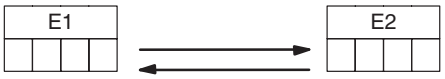


Operand Data Areas

E1: Exchange word 1
IR, AR, DM, HR, TC, LR
E2: Exchange word 2
IR, AR, DM, HR, TC, LR

Description

When the execution condition is OFF, XCHG(73) is not executed. When the execution condition is ON, XCHG(73) exchanges the content of E1 and E2.



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

Flags

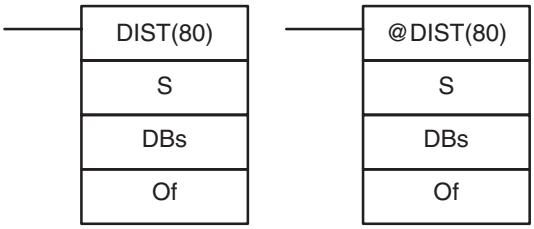
ER: Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

5-14-6 SINGLE WORD DISTRIBUTE – DIST(80)

Operand Data Areas

S: Source data
IR, SR, AR, DM, HR, TC, LR, #
DBs: Destination base word
IR, AR, DM, HR, TC, LR
Of: Offset data (BCD)
IR, AR, DM, HR, TC, LR, #

Ladder Symbols

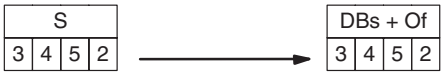


Limitations

Of must be BCD. DBs must be in the same data area as DBs+Of.

Description

When the execution condition is OFF, DIST(80) is not executed. When the execution condition is ON, DIST(80) copies the content of S to DBs+Of, i.e.,Of is added to DBs to determine the destination word.



Flags

ER: The specified offset data is not BCD, or when added to the DBs, the resulting address lies outside the data area of the DBs.

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

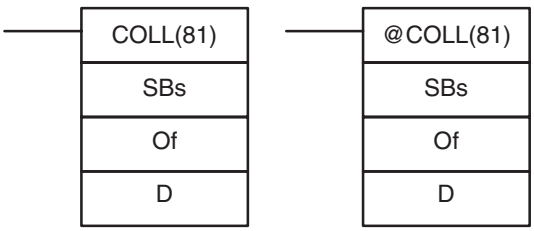
EQ: ON when the content of S is 0; otherwise OFF.

5-14-7 DATA COLLECT – COLL(81)

Operand Data Areas

SBs: Source base word
IR, SR, AR, DM, HR, TC, LR
Of: Offset data (BCD)
IR, AR, DM, HR, TC, LR, #
D: Destination word
IR, AR, DM, HR, TC, LR

Ladder Symbols



Limitations Of must be a BCD. SBs must be in the same data area as SBs+Of.

Description When the execution condition is OFF, COLL(81) is not executed. When the execution condition is ON, COLL(81) copies the content of SBs + Of to D, i.e., Of is added to SBs to determine the source word.

Flags

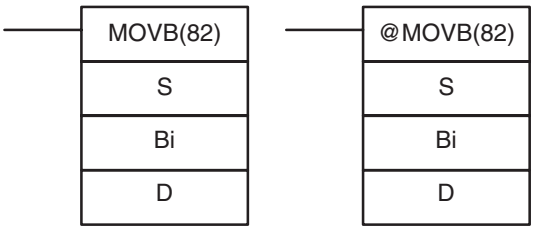
ER: Of is not BCD, or when added to the SBs, or when added to the SBs, the resulting address lies outside the data area of the SBs.

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

EQ: ON when the content of S is 0; otherwise OFF.

5-14-8 MOVE BIT – MOVB(82)

Ladder Symbols

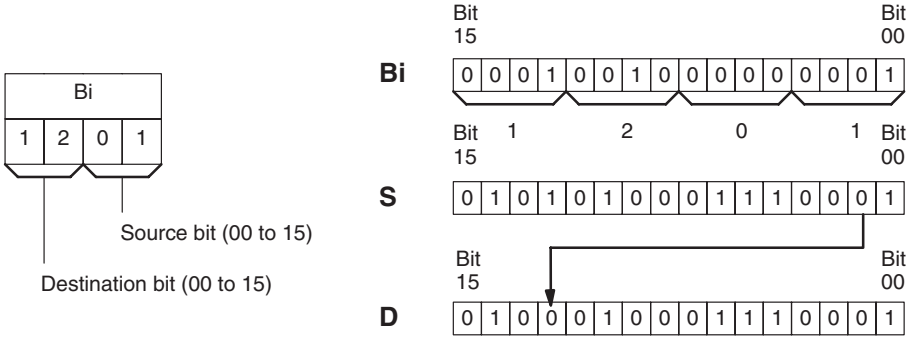


Operand Data Areas

S: Source word
IR, SR, AR, DM, HR, LR, #
Bi: Bit designator (BCD)
IR, AR, DM, HR, TC, LR, #
D: Destination word
IR, AR, DM, HR, LR

Limitations The rightmost two digits and the leftmost two digits of Bi must each be between 00 and 15.

Description When the execution condition is OFF, MOVB(82) is not executed. When the execution condition is ON, MOVB(82) copies the specified bit of S to the specified bit in D. The bits in S and D are specified by Bi. The rightmost two digits of Bi designate the source bit; the leftmost two bits designate the destination bit.



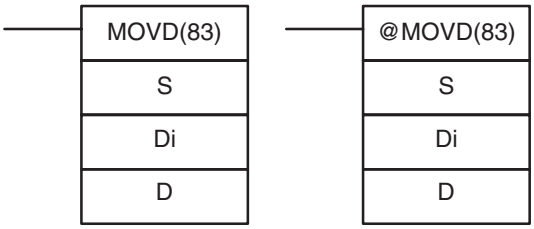
Flags

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

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

5-14-9 MOVE DIGIT – MOVD(83)

Ladder Symbols



Operand Data Areas

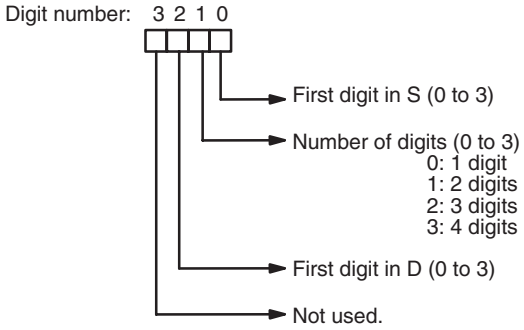
S: Source word	IR, SR, AR, DM, HR, TC, LR, #
Di: Digit designator (BCD)	IR, AR, DM, HR, TC, LR, #
D: Destination word	IR, AR, DM, HR, TC, LR

Limitations

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

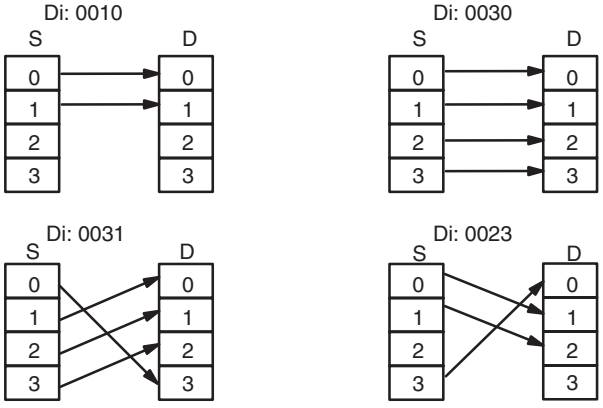
Description

When the execution condition is OFF, MOVD(83) is not executed. When the execution condition is ON, MOVD(83) copies the content of the specified digit(s) in S to the specified digit(s) in D. Up to four digits can be transferred at one time. The first digit to be copied, the number of digits to be copied, and the first digit to receive the copy are designated in Di as shown below. Digits from S will be copied to consecutive digits in D starting from the designated first digit and continued for the designated number of digits. If the last digit is reached in either S or D, further digits are used starting back at digit 0.



Digit Designator

The following show examples of the data movements for various values of Di.



Flags

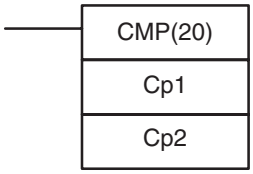
ER: At least one of the rightmost three digits of Di is not between 0 and 3.
Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

5-15 Data Comparison

This section describes the instructions used for comparing data. CMP(20) is used to compare the contents of two words; BCMP(68) is used to determine within which of several preset ranges the content of one word lies; and TCMP(85) is used to determine which of several preset values the content of one word equals.

5-15-1 COMPARE – CMP(20)

Ladder Symbols



Operand Data Areas

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

Cp2: Second compare word
IR, SR, AR, DM, HR, TC, LR, #

Limitations

When comparing a value to the PV of a timer or counter, the value must be in BCD.

Description

When the execution condition is OFF, CMP(20) is not executed. When the execution condition is ON, CMP(20) compares Cp1 and Cp2 and outputs the result to the GR, EQ, and LE flags in the SR area.

Precautions

Placing other instructions between CMP(20) and the operation which accesses the EQ, LE, and GR flags may change the status of these flags. Be sure to access them before the desired status is changed.

Flags

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

EQ: ON if Cp1 equals Cp2.

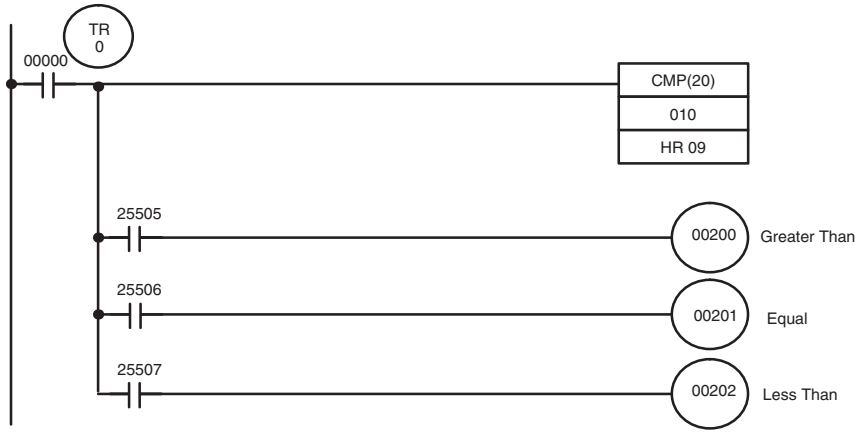
LE: ON if Cp1 is less than Cp2.

GR: ON if Cp1 is greater than Cp2.

Example 1: Saving CMP(20) Results

The following example shows how to save the comparison result immediately. If the content of 010 is greater than that of HR 09, 00200 is turned ON; if the two contents are equal, 00201 is turned ON; if content of 010 is less than that of HR 09, 00202 is turned ON. In some applications, only one of the three OUTs would be necessary, making the use of TR 0 unnecessary. With

this type of programming, 00200, 00201, and 00202 are changed only when CMP(20) is executed.



Address	Instruction	Operands	
00000	LD		00000
00001	OUT	TR	0
00002	CMP(20)		
			010
		HR	09
00003	LD	TR	0
00004	AND		25505

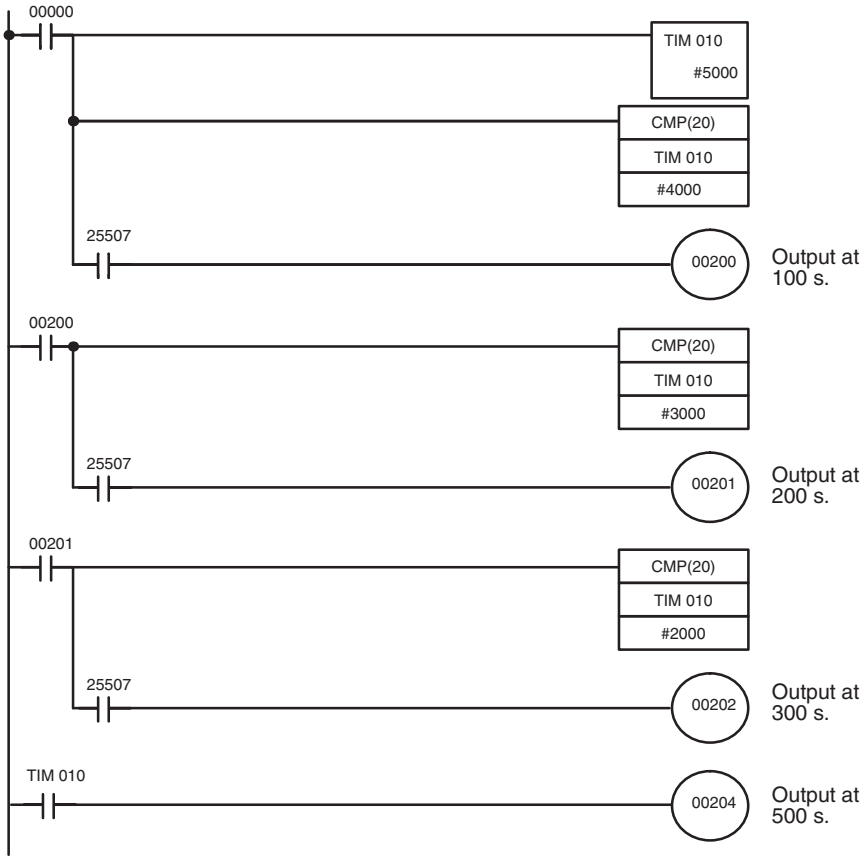
Address	Instruction	Operands	
00005	OUT		00200
00006	LD	TR	0
00007	AND		25506
00008	OUT		00201
00009	LD	TR	0
00010	AND		25507
00011	OUT		00202

Example 2: Obtaining Indications during Timer Operation

The following example uses TIM, CMP(20), and the LE flag (25507) to produce outputs at particular times in the timer's countdown. The timer is started by turning ON 00000. When 00000 is OFF, TIM 010 is reset and the second two CMP(20)s are not executed (i.e., executed with OFF execution conditions). Output 00200 is produced after 100 seconds; output 00201, after 200 seconds; output 00202, after 300 seconds; and output 00204, after 500 seconds.

The branching structure of this diagram is important in order to ensure that 00200, 00201, and 00202 are controlled properly as the timer counts down.

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

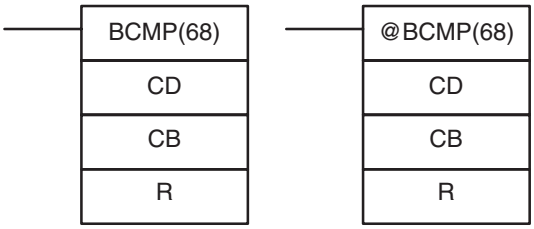


Address	Instruction	Operands
00000	LD	00000
00001	TIM	010
		# 5000
00002	CMP(20)	
		TIM 010
		# 4000
00003	AND	25507
00004	OUT	00200
00005	LD	00200
00006	CMP(20)	
		TIM 010
		# 3000

Address	Instruction	Operands
00007	AND	25507
00008	OUT	00201
00009	LD	00201
00010	CMP(20)	
		TIM 010
		# 2000
00011	AND	25507
00012	OUT	00202
00013	LD	TIM 010
00014	OUT	00204

5-15-2 BLOCK COMPARE – BCMP(68)

Ladder Symbols



Operand Data Areas

CD: Compare data
IR, SR, DM, HR, TC, LR, #
CB: First comparison block word
IR, SR, DM, HR, TC, LR
R: Result word
IR, AR, DM, HR, TC, LR

Limitations

Each lower limit word in the comparison block must be less than or equal to the upper limit.

Description

When the execution condition is OFF, BCMP(68) is not executed. When the execution condition is ON, BCMP(68) compares CD to the ranges defined by a block consisting of of CB, CB+1, CB+2, ..., CB+32. Each range is defined by two words, the first one providing the lower limit and the second word providing the upper limit. If CD is found to be within any of these ranges (inclusive of the upper and lower limits), the corresponding bit in R is set. The comparisons that are made and the corresponding bit in R that is set for each true comparison are shown below. The rest of the bits in R will be turned OFF.

CB ≤ CD ≤ CB+1	Bit 00
CB+2 ≤ CD ≤ CB+3	Bit 01
CB+4 ≤ CD ≤ CB+5	Bit 02
CB+6 ≤ CD ≤ CB+7	Bit 03
CB+8 ≤ CD ≤ CB+9	Bit 04
CB+10 ≤ CD ≤ CB+11	Bit 05
CB+12 ≤ CD ≤ CB+13	Bit 06
CB+14 ≤ CD ≤ CB+15	Bit 07
CB+16 ≤ CD ≤ CB+17	Bit 08
CB+18 ≤ CD ≤ CB+19	Bit 09
CB+20 ≤ CD ≤ CB+21	Bit 10
CB+22 ≤ CD ≤ CB+23	Bit 12
CB+24 ≤ CD ≤ CB+25	Bit 13
CB+26 ≤ CD ≤ CB+27	Bit 14
CB+28 ≤ CD ≤ CB+29	Bit 15
CB+30 ≤ CD ≤ CB+31	Bit 16

Flags

ER: The comparison block (i.e., CB through CB+31) exceeds the data area.

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

Example

The following example shows the comparisons made and the results provided for BCMP(68). Here, the comparison is made during each cycle when 00000 is ON.

Address	Instruction	Operands	
00000	LD		00000
00001	BCMP(68)		
			001
		HR	10
		HR	05

CD: 001
001 0210

Compare data in IR 001 (which contains 0210) with the given ranges.

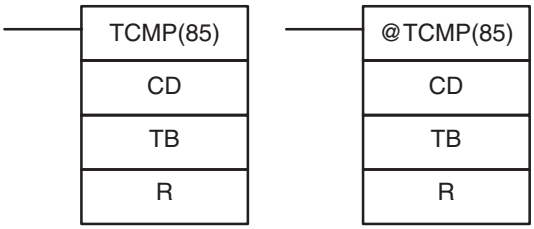
Lower limits	
HR 10	0000
HR 12	0101
HR 14	0201
HR 16	0301
HR 18	0401
HR 20	0501
HR 22	0601
HR 24	0701
HR 26	0801
HR 28	0901
HR 30	1001
HR 32	1101
HR 34	1201
HR 36	1301
HR 38	1401
HR 40	1501

Upper limits	
HR 11	0100
HR 13	0200
HR 15	0300
HR 17	0400
HR 19	0500
HR 21	0600
HR 23	0700
HR 25	0800
HR 27	0900
HR 29	1000
HR 31	1100
HR 33	1200
HR 35	1300
HR 37	1400
HR 39	1500
HR 41	1600

R: HR 05	
HR 0500	0
HR 0501	0
HR 0502	1
HR 0503	0
HR 0504	0
HR 0505	0
HR 0506	0
HR 0507	0
HR 0508	0
HR 0509	0
HR 0510	0
HR 0511	0
HR 0512	0
HR 0513	0
HR 0514	0
HR 0515	0

5-15-3 TABLE COMPARE – TCMP(85)

Ladder Symbols



Operand Data Areas

CD: Compare data
IR, SR, AR, DM, HR, TC, LR, #
TB: First comparison table word
IR, SR, DM, HR, TC, LR
R: Result word
IR, AR, DM, HR, TC, LR

Description

When the execution condition is OFF, TCMP(85) is not executed. When the execution condition is ON, TCMP(85) compares CD to the content of TB, TB+1, TB+2, ..., and TB+15. If CD is equal to the content of any of these words, the corresponding bit in R is set, e.g., if the CD equals the content of TB, bit 00 is turned ON, if it equals that of TB+1, bit 01 is turned ON, etc. The rest of the bits in R will be turned OFF.

Flags

ER: The comparison table (i.e., TB through TB+15) exceeds the data area.

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

Example

The following example shows the comparisons made and the results provided for TCMP(85). Here, the comparison is made during each cycle when 00000 is ON.

Address	Instruction	Operands
00000	LD	00000
00001	TCMP(85)	
		001
		HR 10
		HR 05

CD: 001	
001	0210

Compare the data in IR 001 with the given ranges.

Upper limits	
HR 10	0100
HR 11	0200
HR 12	0210
HR 13	0400
HR 14	0500
HR 15	0600
HR 16	0210
HR 17	0800
HR 18	0900
HR 19	1000
HR 20	0210
HR 21	1200
HR 22	1300
HR 23	1400
HR 24	0210
HR 25	1600

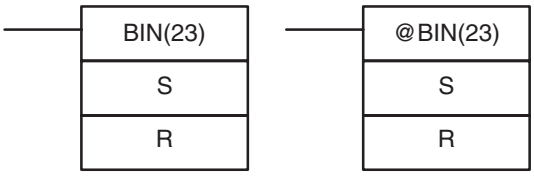
R: HR 05	
HR 0500	0
HR 0501	0
HR 0502	1
HR 0503	0
HR 0504	0
HR 0505	0
HR 0506	1
HR 0507	0
HR 0508	0
HR 0509	0
HR 0510	1
HR 0511	0
HR 0512	0
HR 0513	0
HR 0514	1
HR 0515	0

5-16 Data Conversion

The conversion instructions convert word data that is in one format into another format and output the converted data to specified result word(s). Conversions are available to convert between binary (hexadecimal) and BCD, to 7-segment display data, to ASCII, and between multiplexed and non-multiplexed data. All of these instructions change only the content of the words to which converted data is being moved, i.e., the content of source words is the same before and after execution of any of the conversion instructions.

5-16-1 BCD-TO-BINARY – BIN(23)

Ladder Symbols

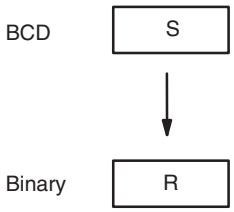


Operand Data Areas

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

Description

When the execution condition is OFF, BIN(23) is not executed . When the execution condition is ON, BIN(23) converts the BCD content of S into the numerically equivalent binary bits, and outputs the binary value to R. Only the content of R is changed; the content of S is left unchanged.



BIN(23) can be used to convert BCD to binary so that displays on the Programming Console or any other programming device will appear in hexadecimal rather than decimal. It can also be used to convert to binary to perform binary arithmetic operations rather than BCD arithmetic operations, e.g., when BCD and binary values must be added.

Flags

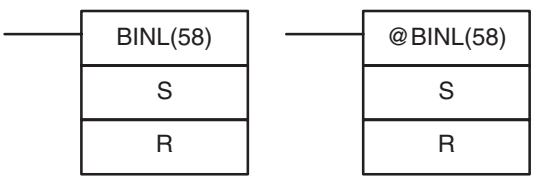
ER: The content of S is not BCD.

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

EQ: ON when 0 is placed in R.

5-16-2 DOUBLE BCD-TO-DOUBLE BINARY – BINL(58)

Ladder Symbols

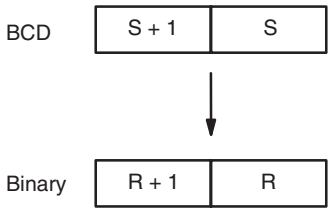


Operand Data Areas

S: First source word (BCD)	IR, SR, AR, DM, HR, TC, LR
R: First result word	IR, AR, DM, HR, LR

Description

When the execution condition is OFF, BINL(58) is not executed. When the execution condition is ON, BINL(58) converts an eight-digit number in S and S+1 into 32-bit binary data, and outputs the converted data to R and R+1.



Flags

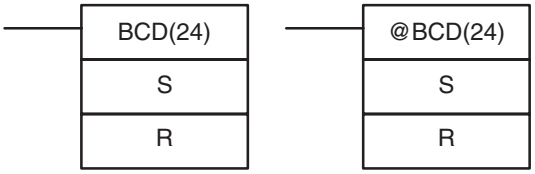
ER: The contents of S and/or S+1 words are not BCD.

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

EQ: ON when 00000000 is placed in R.

5-16-3 BINARY-TO-BCD – BCD(24)

Ladder Symbols



Operand Data Areas

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

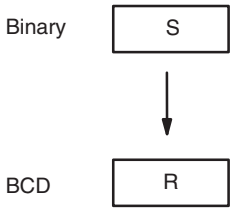
R: Result word
IR, AR, DM, HR, LR

Limitations

If the content of S exceeds 270F, the converted result would exceed 9999 and BCD(24) will not be executed. When the instruction is not executed, the content of R remains unchanged.

Description

BCD(24) converts the binary (hexadecimal) content of S into the numerically equivalent BCD bits, and outputs the BCD bits to R. Only the content of R is changed; the content of S is left unchanged.



BCD(24) can be used to convert binary to BCD so that displays on the Programming Console or any other programming device will appear in decimal rather than hexadecimal. It can also be used to convert to BCD to perform BCD arithmetic operations rather than binary arithmetic operations, e.g., when BCD and binary values must be added.

Flags

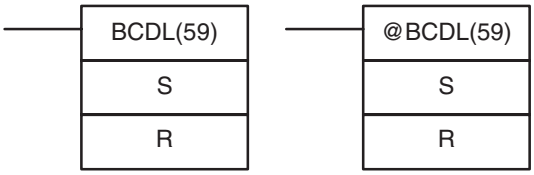
ER: S is greater than 270F.

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

EQ: ON when 0 is placed in R.

5-16-4 DOUBLE BINARY-TO-DOUBLE BCD – BCDL(59)

Ladder Symbols



Operand Data Areas

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

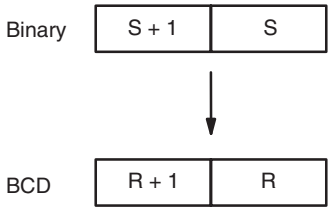
R: First result word
IR, AR, DM, HR, LR

Limitations

If the content of S exceeds 05F5E0FF, the converted result would exceed 99999999 and BCDL(59) will not be executed. When the instruction is not executed, the content of R and R+1 remain unchanged.

Description

BCDL(59) converts the 32-bit binary content of S and S+1 into eight digits of BCD data, and outputs the converted data to R and R+1.



Flags

ER: Content of R and R+1 exceeds 99999999.

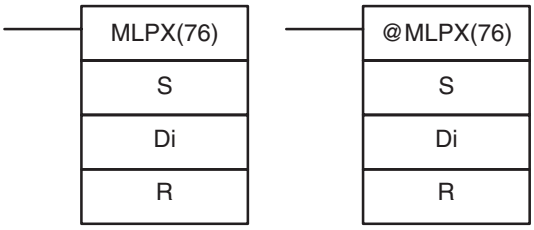
Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON when 0 is placed in R.

5-16-5 4-TO-16 DECODER – MLPX(76)

Operand Data Areas

Ladder Symbols



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

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

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

Limitations

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

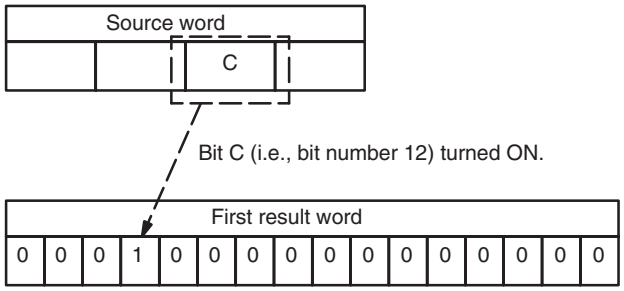
All result words must be in the same data area.

Description

When the execution condition is OFF, MLPX(76) is not executed. When the execution condition is ON, MLPX(76) converts up to four, four-bit hexadecimal digits from S into decimal values from 0 to 15, each of which is used to indicate a bit position. The bit whose number corresponds to each converted

value is then turned ON in a result word. If more than one digit is specified, then one bit will be turned ON in each of consecutive words beginning with R. (See examples, below.)

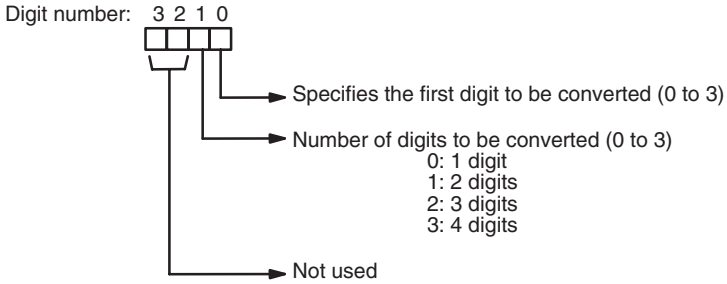
The following is an example of a one-digit decode operation from digit number 1 of S, i.e., here Di would be 0001.



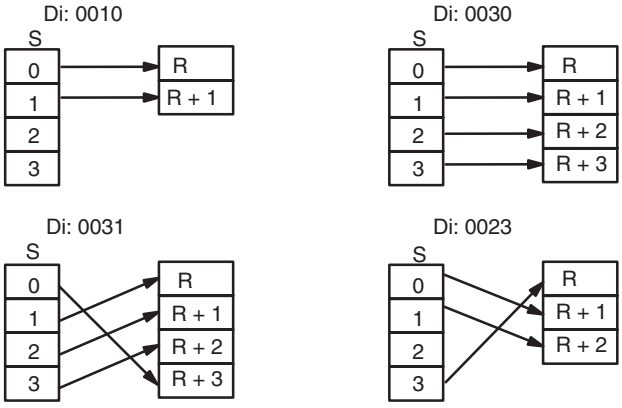
The first digit and the number of digits to be converted are designated in Di. If more digits are designated than remain in S (counting from the designated first digit), the remaining digits will be taken starting back at the beginning of S. The final word required to store the converted result (R plus the number of digits to be converted) must be in the same data area as R, e.g., if two digits are converted, the last word address in a data area cannot be designated; if three digits are converted, the last two words in a data area cannot be designated.

Digit Designator

The digits of Di are set as shown below.



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



Flags

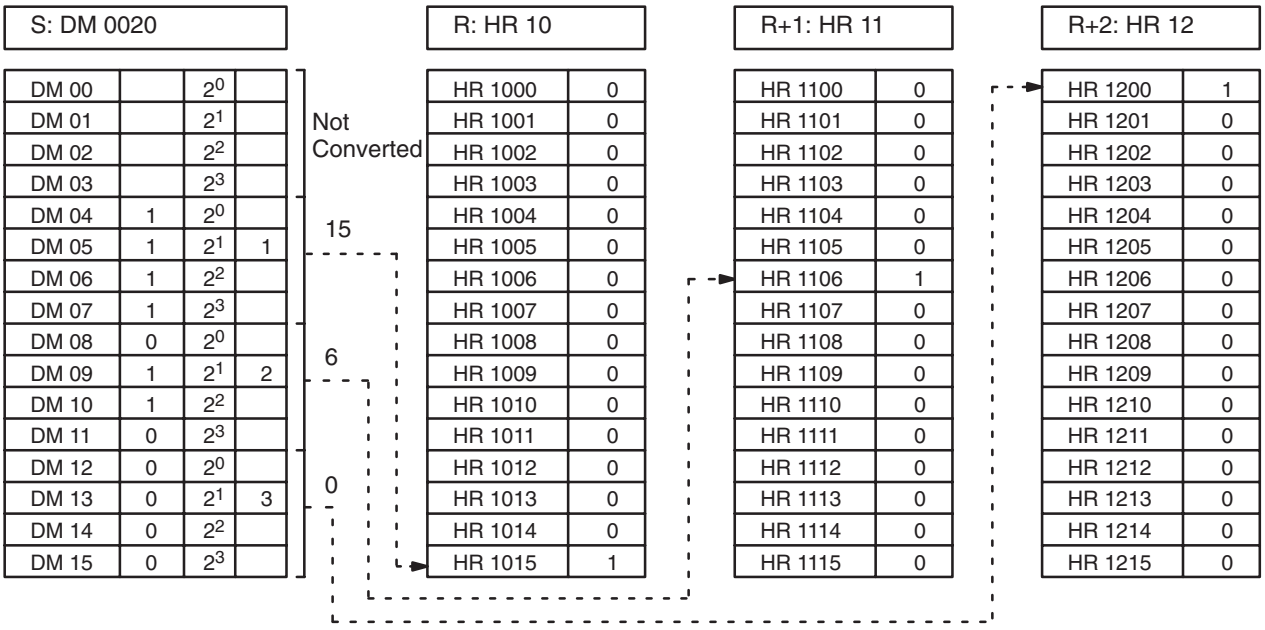
ER: Undefined digit designator, or R plus number of digits exceeds a data area.

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

Example

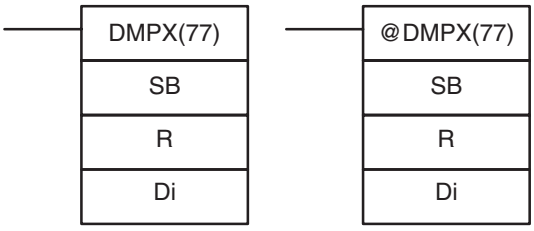
The following program converts three digits of data from DM 0020 to bit positions and turns ON the corresponding bits in three consecutive words starting with HR 10.

Address	Instruction	Operands	
00000	LD		00000
00001	MLPX(76)		
		DM	00200
		#	0021
		HR	10



5-16-6 16-TO-4 ENCODER – DMPX(77)

Ladder Symbols



Operand Data Areas

SB: First source word
IR, SR, AR, DM, HR, TC, LR

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

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

Limitations

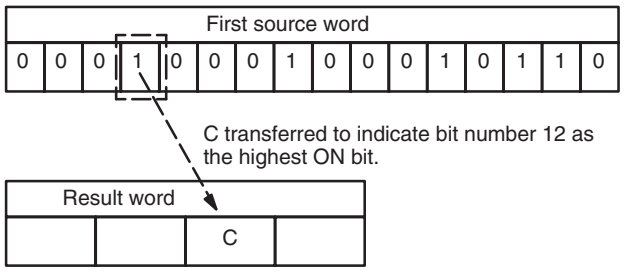
The rightmost two digits of Di must each be between 0 and 3.

All source words must be in the same data area.

Description

When the execution condition is OFF, DMPX(77) is not executed. When the execution condition is ON, DMPX(77) determines the position of the highest ON bit in S, encodes it into single-digit hexadecimal value corresponding to the bit number of the highest ON bit number, then transfers the hexadecimal value to the specified digit in R. The digits to receive the results are specified in Di, which also specifies the number of digits to be encoded.

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

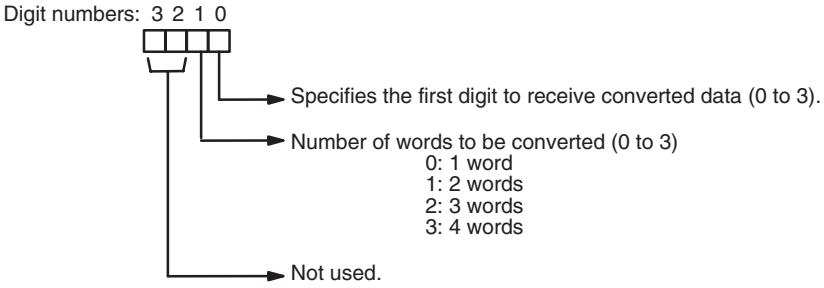


Up to four digits from four consecutive source words starting with S may be encoded and the digits written to R in order from the designated first digit. If more digits are designated than remain in R (counting from the designated first digit), the remaining digits will be placed at digits starting back at the beginning of R.

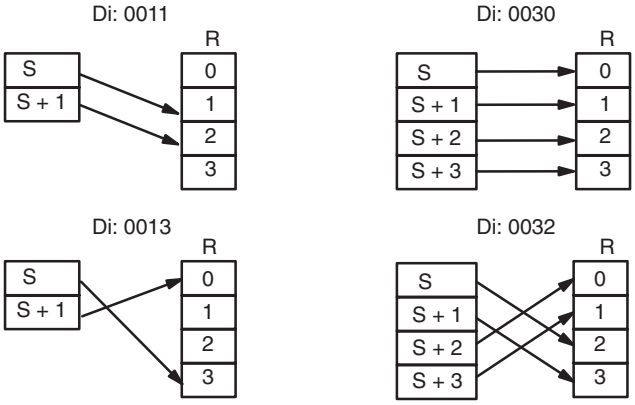
The final word to be converted (S plus the number of digits to be converted) must be in the same data area as SB.

Digit Designator

The digits of Di are set as shown below.



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



Flags

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

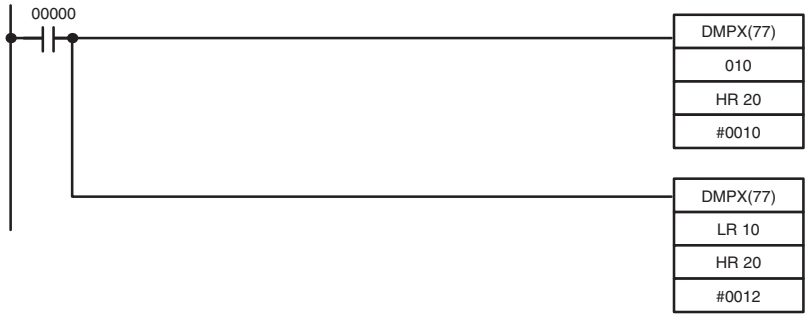
Content of a source word is 0.

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

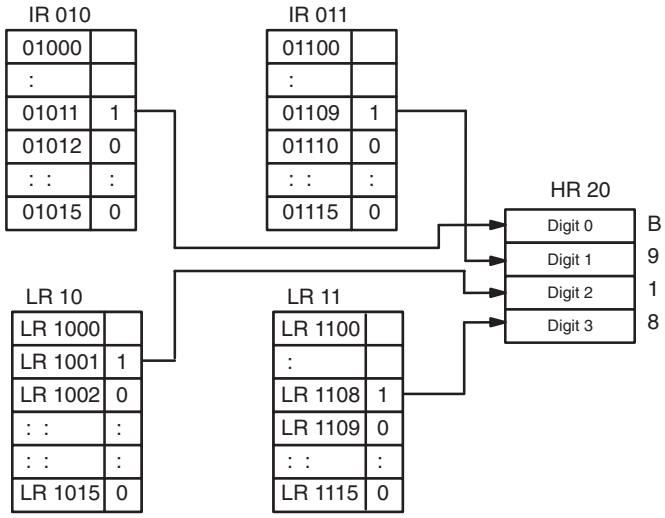
Example

When 00000 is ON, the following diagram encodes IR words 010 and 011 to the first two digits of HR 20 and then encodes LR 10 and 11 to the last two

digits of HR 20. Although the status of each source word bit is not shown, it is assumed that the bit with status 1 (ON) shown is the highest bit that is ON in the word.

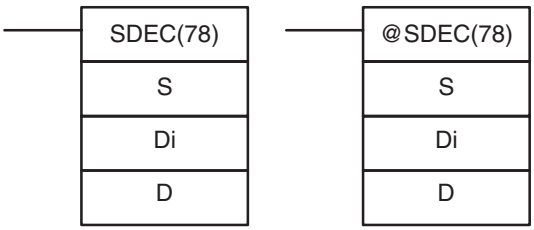


Address	Instruction	Operands	
00000	LD		00000
00001	DMPX(77)		
			010
		HR	20
		#	0010
00002	DMPX(77)		
		LR	10
		HR	20
		#	0012



5-16-7 7-SEGMENT DECODER – SDEC(78)

Ladder Symbols



Operand Data Areas

S: Source word (binary)
IR, SR, AR, DM, HR, TC, LR
Di: Digit designator
IR, AR, DM, HR, TC, LR, #
D: First destination word
IR, AR, DM, HR, LR

Limitations

Di must be within the values given below

All destination words must be in the same data area.

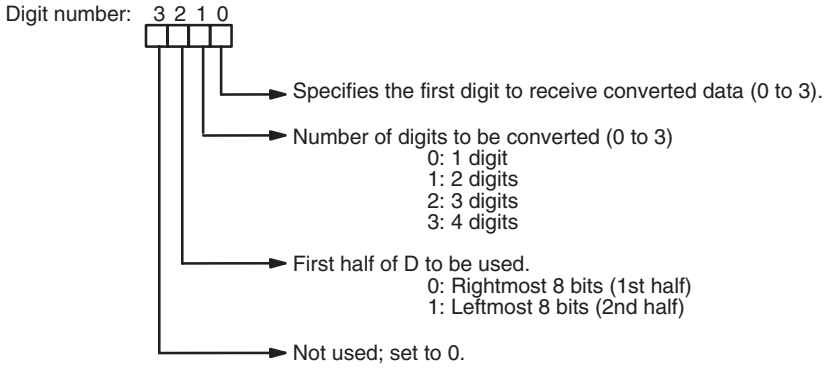
Description

When the execution condition is OFF, SDEC(78) is not executed. When the execution condition is ON, SDEC(78) converts the designated digit(s) of S into the equivalent 8-bit, 7-segment display code and places it into the destination word(s) beginning with D.

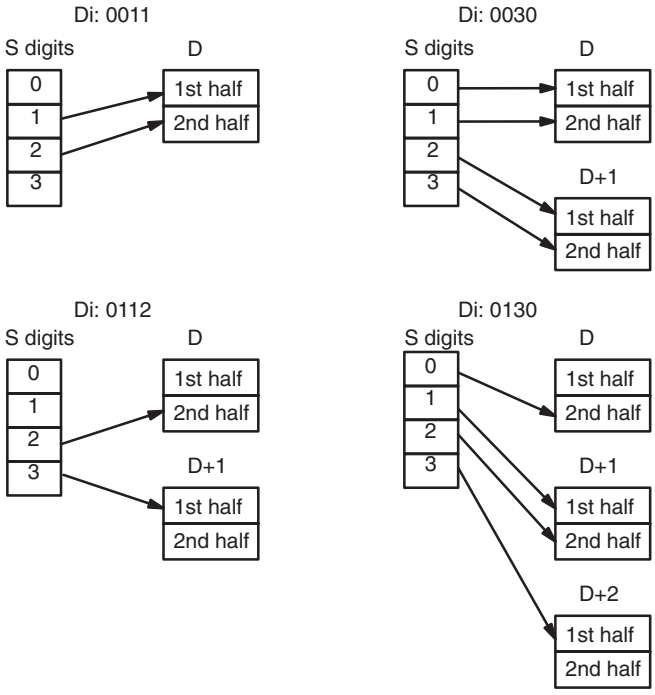
Any or all of the digits in S may be converted in sequence from the designated first digit. The first digit, the number of digits to be converted, and the half of D to receive the first 7-segment display code (rightmost or leftmost 8 bits) are designated in Di. If multiple digits are designated, they will be placed in order starting from the designated half of D, each requiring two digits. If more digits are designated than remain in S (counting from the designated first digit), further digits will be used starting back at the beginning of S.

Digit Designator

The digits of Di are set as shown below.



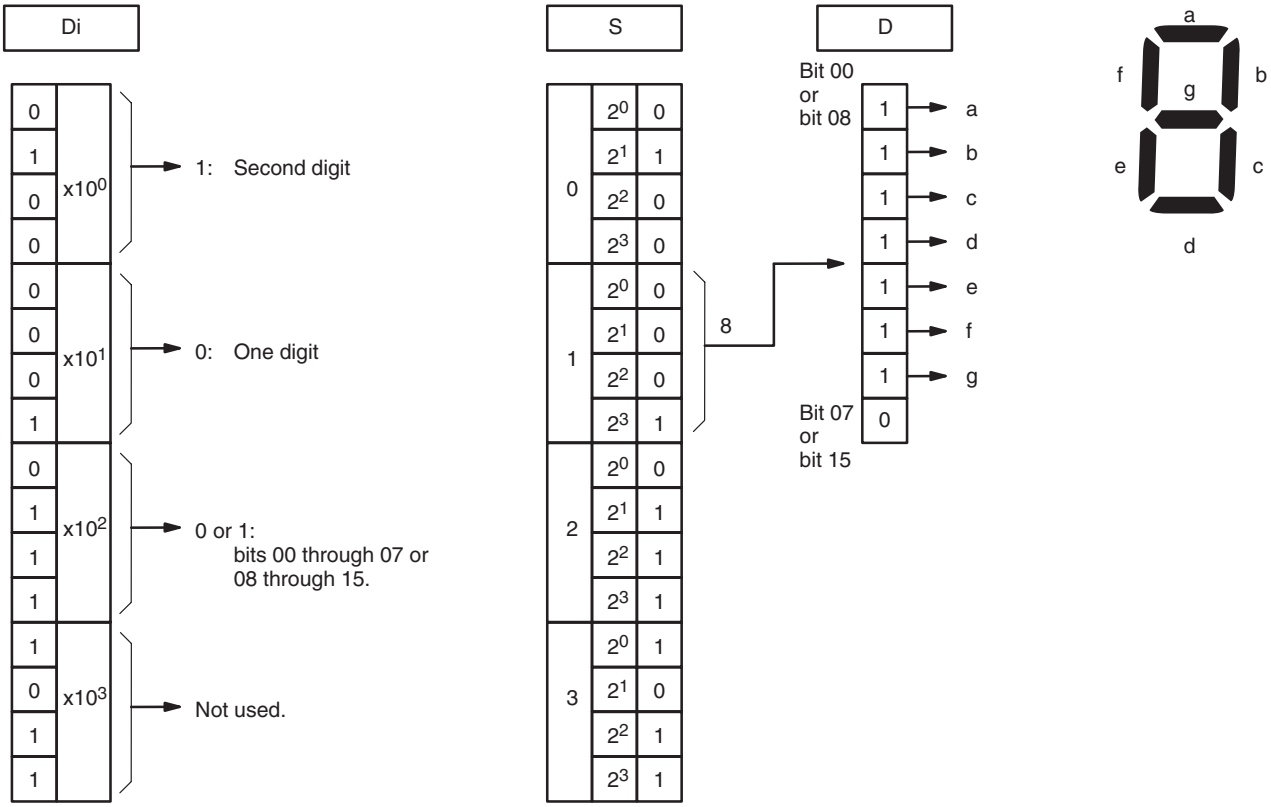
Some example Di values and the 4-bit binary to 7-segment display conversions that they produce are shown below.



Example

The following example shows the data to produce an 8. The lower case letters show which bits correspond to which segments of the 7-segment display.

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



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

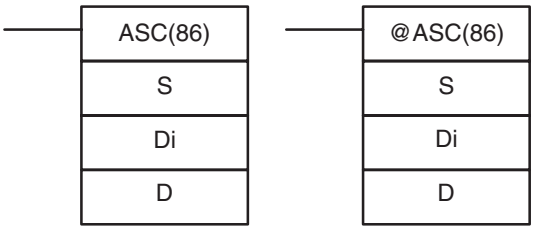
Flags

ER: Incorrect digit designator, or data area for destination exceeded

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

5-16-8 ASCII CONVERT – ASC(86)

Ladder Symbols



Operand Data Areas

S: Source word	IR, SR, AR, DM, HR, TC, LR
Di: Digit designator	IR, AR, DM, HR, TC, LR, #
D: First destination word	IR, AR, DM, HR, LR

Limitations

Di must be within the values given below

All destination words must be in the same data area.

Description

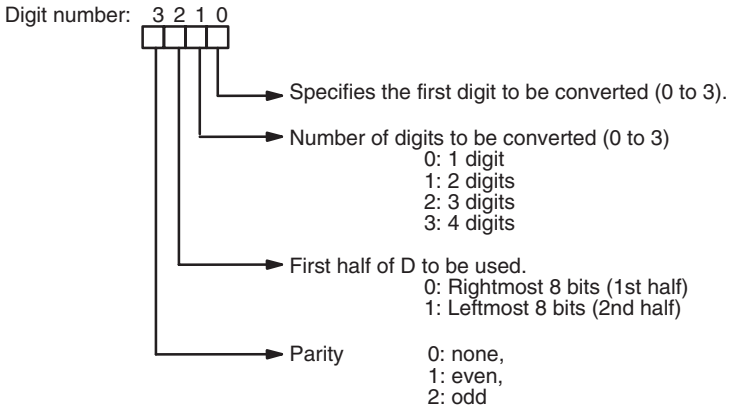
When the execution condition is OFF, ASC(86) is not executed. When the execution condition is ON, ASC(86) converts the designated digit(s) of S into the equivalent 8-bit ASCII code and places it into the destination word(s) beginning with D.

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

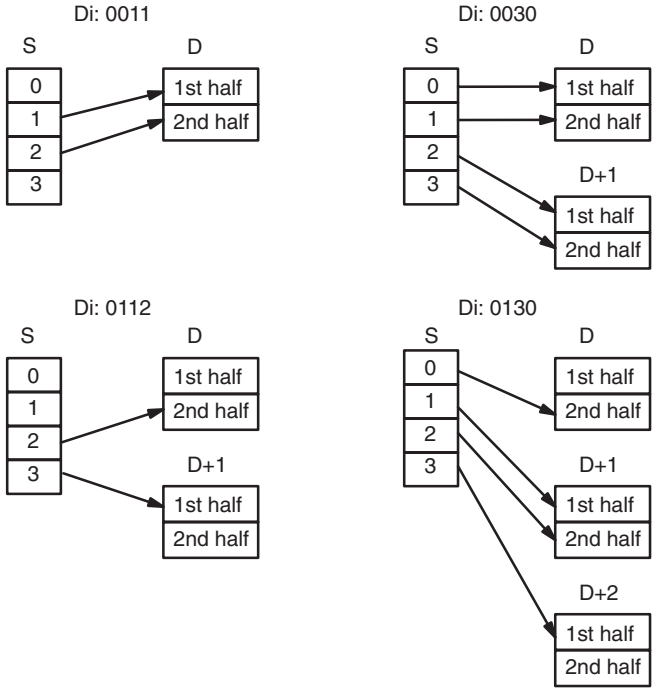
Refer to *Appendix F* for a table of extended ASCII characters.

Digit Designator

The digits of Di are set as shown below.



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



Parity

The leftmost bit of each ASCII character (2 digits) can be automatically adjusted for either even or odd parity. If no parity is designated, the leftmost bit will always be zero.

When even parity is designated, the leftmost bit will be adjusted so that the total number of ON bits is even, e.g., when adjusted for even parity, ASCII "31" (00110001) will be "B1" (10110001: parity bit turned ON to create an even number of ON bits); ASCII "36" (00110110) will be "36" (00110110: parity bit turned OFF because the number of ON bits is already even). The status of the parity bit does not affect the meaning of the ASCII code.

When odd parity is designated, the leftmost bit of each ASCII character will be adjusted so that there is an odd number of ON bits.

Flags

ER: Incorrect digit designator, or data area for destination exceeded..

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

5-17 BCD Calculations

The BCD calculation instructions – INC(38), DEC(39), ADD(30), ADDL(54), SUB(31), SUBL(55), MUL(32), MULL(56), DIV(33), DIVL(57), FDIV(79), and ROOT(72) – all perform arithmetic operations on BCD data.

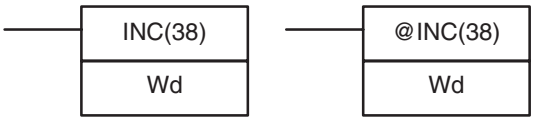
For INC(38) and DEC(39) the source and result words are the same. That is, the content of the source word is overwritten with the instruction result. All other instructions change only the content of the words in which results are placed, i.e., the contents of source words are the same before and after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

STC(40) and CLC(41), which set and clear the carry flag, are included in this group because most of the BCD operations make use of the carry flag (CY) in their results. Binary calculations and shift operations also use CY.

The addition and subtraction instructions include CY in the calculation as well as in the result. Be sure to clear CY if its previous status is not required in the calculation, and to use the result placed in CY, if required, before it is changed by execution of any other instruction.

5-17-1 INCREMENT – INC(38)

Ladder Symbols



Operand Data Areas

Wd: Increment word (BCD)
IR, AR, DM, HR, LR

Description When the execution condition is OFF, INC(38) is not executed. When the execution condition is ON, INC(38) increments Wd, without affecting carry (CY).

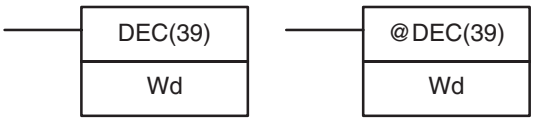
Flags

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

EQ: ON when the incremented result is 0.

5-17-2 DECREMENT – DEC(39)

Ladder Symbols



Operand Data Areas

Wd: Decrement word (BCD)
IR, AR, DM, HR, LR

Description When the execution condition is OFF, DEC(39) is not executed. When the execution condition is ON, DEC(39) decrements Wd, without affecting CY. DEC(39) works the same way as INC(38) except that it decrements the value instead of incrementing it.

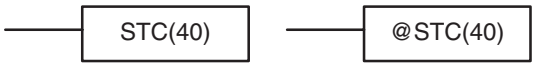
Flags

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

EQ: ON when the decremented result is 0.

5-17-3 SET CARRY – STC(40)

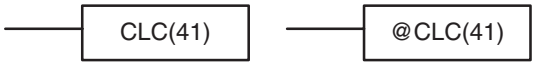
Ladder Symbols



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

5-17-4 CLEAR CARRY – CLC(41)

Ladder Symbols



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

5-17-5 BCD ADD – ADD(30)

Ladder Symbols

ADD(30)
Au
Ad
R

@ADD(30)
Au
Ad
R

Operand Data Areas

Au: Augend word (BCD)
IR, SR, AR, DM, HR, TC, LR, #

Ad: Addend word (BCD)
IR, SR, AR, DM, HR, TC, LR, #

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

Description

When the execution condition is OFF, ADD(30) is not executed. When the execution condition is ON, ADD(30) adds the contents of Au, Ad, and CY, and places the result in R. CY will be set if the result is greater than 9999.

Au + Ad + CY → CY R

Flags

ER: Au and/or Ad is not BCD.

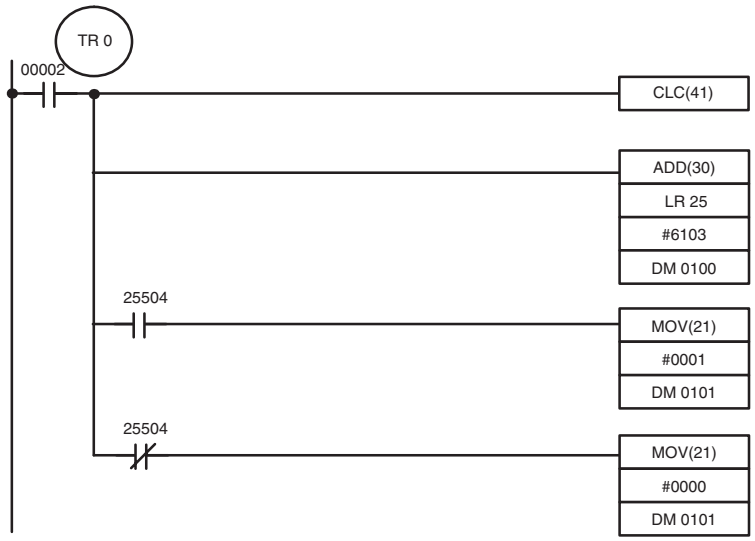
Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

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

EQ: ON when the result is 0.

Example

If 00002 is ON, the program represented by the following diagram clears CY with CLC(41), adds the content of LR 25 to a constant (6103), places the result in DM 0100, and then moves either all zeros or 0001 into DM 0101 depending on the status of CY (25504). This ensures that any carry from the last digit is preserved in R+1 so that the entire result can be later handled as eight-digit data.

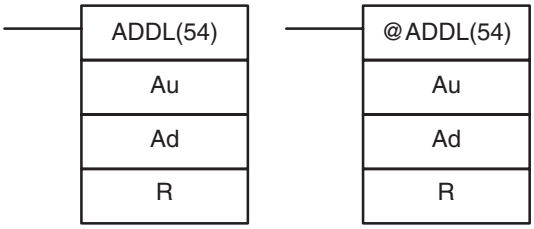


Address	Instruction	Operands	
00000	LR		00002
00001	OUT	TR	0
00002	CLC(41)		
00003	AND(30)		
		LR	25
		#	6103
		DM	0100
00004	AND		25504
00005	MOV(21)		
		#	0001
		DM	0101
00006	LD	TR	0
00007	AND NOT		25504
00008	MOV(21)		
		#	0000
		DM	0101

Although two ADD(30) can be used together to perform eight-digit BCD addition, ADDL(54) is designed specifically for this purpose.

5-17-6 DOUBLE BCD ADD – ADDL(54)

Ladder Symbols

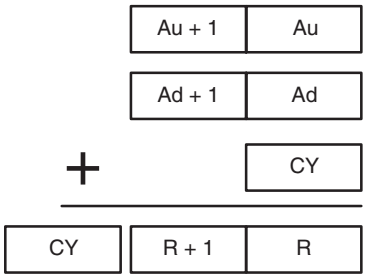


Operand Data Areas

Au: First augend word (BCD)	IR, SR, AR, DM, HR, TC, LR
Ad: First addend word (BCD)	IR, SR, AR, DM, HR, TC, LR
R: First result word	IR, AR, DM, HR, LR

Description

When the execution condition is OFF, ADDL(54) is not executed. When the execution condition is ON, ADDL(54) adds the contents of CY to the 8-digit value in Au and Au+1 to the 8-digit value in Ad and Ad+1, and places the result in R and R+1. CY will be set if the result is greater than 99999999.



Flags

ER: Au and/or Ad is not BCD.

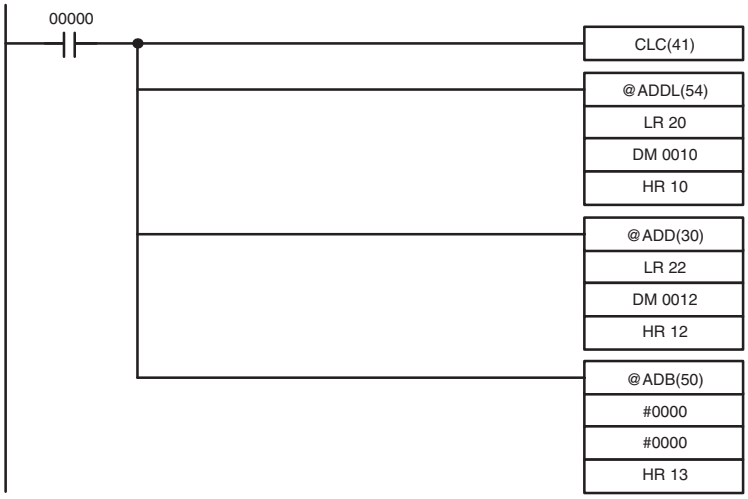
Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

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

EQ: ON when the result is 0.

Example

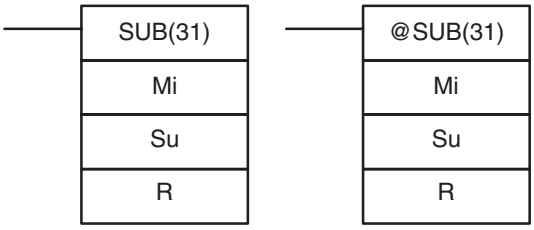
When 00000 is ON, the following program adds two 12-digit numbers, the first contained in LR 20 through LR 22 and the second in DM 0012. The result is placed in LR 10 through HR 13. In the second addition (using ADD(30)), any carry from the first addition is included. The carry from the second addition is placed in HR 13 by using @ADB(50) (see *Section 5-17-1*) with two all-zero constants to indirectly place the content of CY into HR 13.



Address	Instruction	Operands	
00000	LD		00000
00001	CLC(41)		
00002	@ADDL(54)		
		LR	20
		DM	0010
		HR	10
00003	@ADD(30)		
		LR	22
		DM	0012
		HR	12
00004	@ADB(50)		
		#	0000
		#	0000
		HR	13

5-17-7 BCD SUBTRACT – SUB(31)

Ladder Symbols



Operand Data Areas

Mi: Minuend word (BCD)
IR, SR, AR, DM, HR, TC, LR, #
Su: Subtrahend word (BCD)
IR, SR, AR, DM, HR, TC, LR, #
R: Result word
IR, AR, DM, HR, LR

Description

When the execution condition is OFF, SUB(31) is not executed. When the execution condition is ON, SUB(31) subtracts the contents of Su and CY from Mi, and places the result in R. If the result is negative, CY is set and the 10's complement of the actual result is placed in R. To convert the 10's complement to the true result, subtract the content of R from zero (see example below).

Flags

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

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

EQ: ON when the result is 0.

⚠ Caution Be sure to clear the carry flag with CLC(41) before executing SUB(31) if its previous status is not required, and check the status of CY after doing a subtraction with SUB(31). If CY is ON as a result of executing SUB(31) (i.e., if the result is negative), the result is output as the 10's complement of the true answer. To convert the output result to the true value, subtract the value in R from 0.

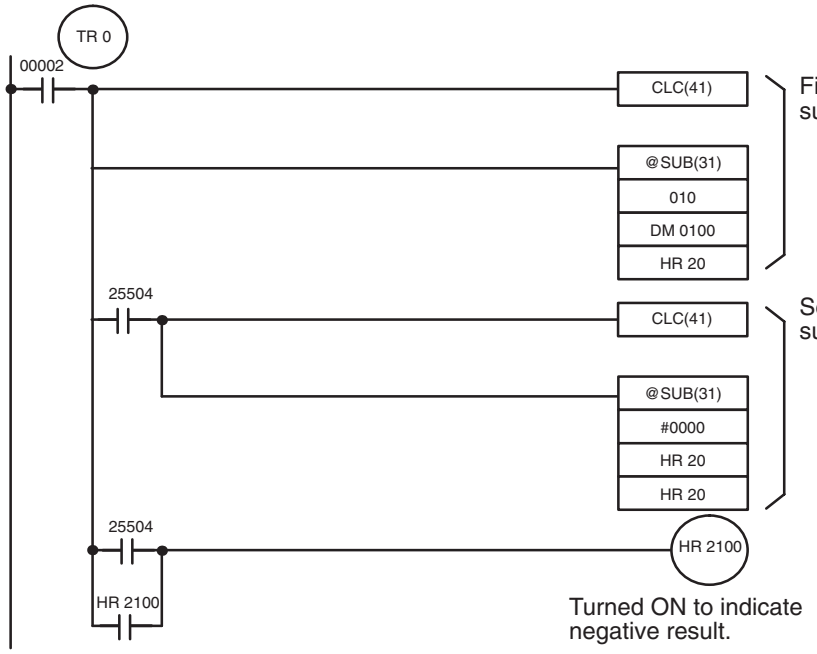
Example

When 00002 is ON, the following ladder program clears CY, subtracts the contents of DM 0100 and CY from the content of 010 and places the result in HR 20.

If CY is set by executing SUB(31), the result in HR 20 is subtracted from zero (note that CLC(41) is again required to obtain an accurate result), the result is placed back in HR 20, and HR 2100 is turned ON to indicate a negative result.

If CY is not set by executing SUB(31), the result is positive, the second subtraction is not performed, and HR 2100 is not turned ON. HR 2100 is programmed as a self-maintaining bit so that a change in the status of CY will not turn it OFF when the program is recycled.

In this example, differentiated forms of SUB(31) are used so that the subtraction operation is performed only once each time 00002 is turned ON. When another subtraction operation is to be performed, 00002 will need to be turned OFF for at least one cycle (resetting HR 2100) and then turned back ON.



Address	Instruction	Operands	
00000	LD		00002
00001	OUT	TR	0
00002	CLC(41)		
00003	@SUB(31)		
			010
		DM	0100
		HR	20
00004	AND		25504
00005	CLC(41)		
00006	@SUB(31)		
		#	0000
		HR	20
		HR	20
00007	LD	TR	0
00008	AND		25504
00009	OR	HR	2100
00010	OUT	HR	2100

The first and second subtractions for this diagram are shown below using example data for 010 and DM 0100.

Note The actual SUB(31) operation involves subtracting Su and CY from 10,000 plus Mi. For positive results the leftmost digit is truncated. For negative results the 10s complement is obtained. The procedure for establishing the correct answer is given below.

First Subtraction

IR 010	1029	
DM 0100	– 3452	
CY	– 0	
HR 20	7577	(1029 + (10000 – 3452))
CY	1	(negative result)

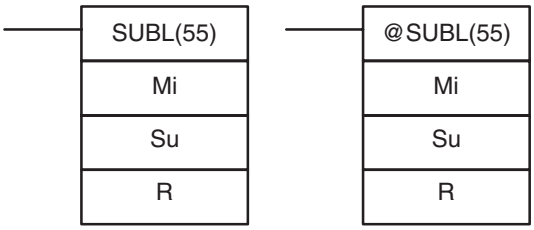
Second Subtraction

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

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

5-17-8 DOUBLE BCD SUBTRACT – SUBL(55)

Ladder Symbols

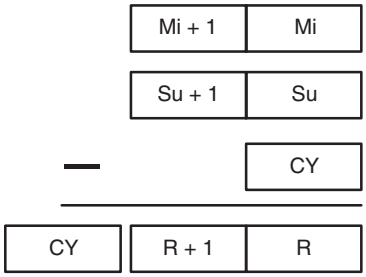


Operand Data Areas

Mi: First minuend word (BCD)
IR, SR, AR, DM, HR, TC, LR, #
Su: First subtrahend word (BCD)
IR, SR, AR, DM, HR, TC, LR, #
R: First result word
IR, AR, DM, HR, LR

Description

When the execution condition is OFF, SUBL(55) is not executed. When the execution condition is ON, SUBL(55) subtracts CY and the 8-digit contents of Su and Su+1 from the 8-digit value in Mi and Mi+1, and places the result in R and R+1. If the result is negative, CY is set and the 10's complement of the actual result is placed in R. To convert the 10's complement to the true result, subtract the content of R from zero. Since an 8-digit constant cannot be directly entered, use the BSET(71) instruction (see *Section 5-13-3*) to create an 8-digit constant.



Flags

ER: Mi, M+1,Su, or Su+1 are not BCD.

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

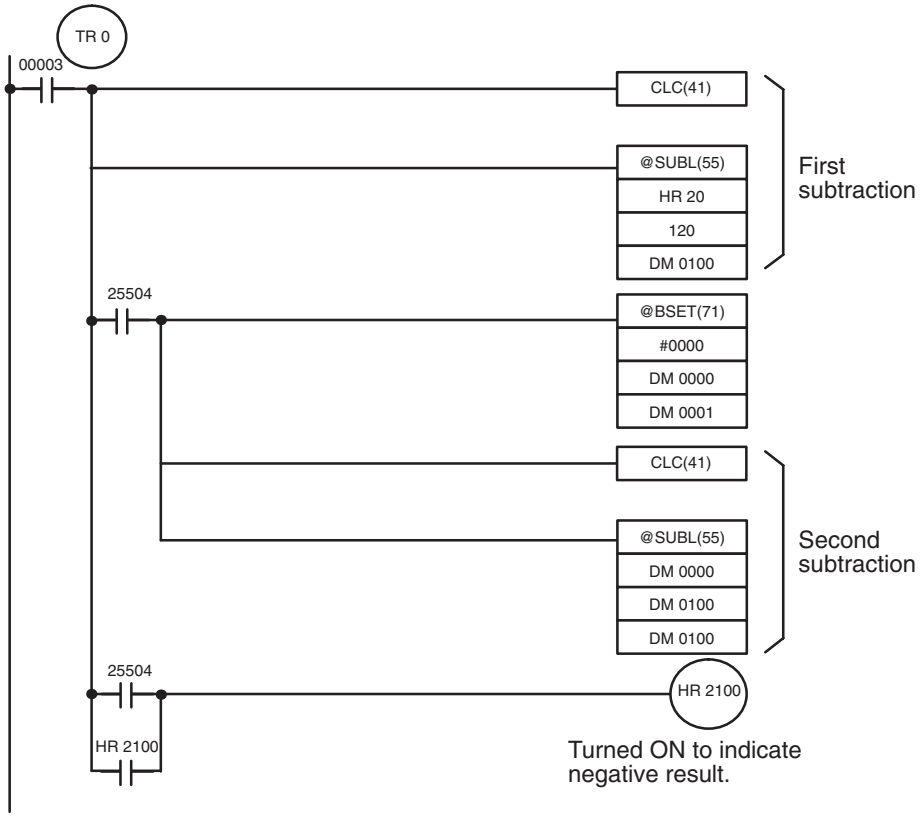
CY: ON when the result is negative, i.e., when Mi is less than Su.

EQ: ON when the result is 0.

Example

The following example works much like that for single-word subtraction. In this example, however, BSET(71) is required to clear the content of DM 0000

and DM 0001 so that a negative result can be subtracted from 0 (inputting an 8-digit constant is not possible).



Address	Instruction	Operands	
00000	LD		00003
00001	OUT	TR	0
00002	CLC(41)		
00003	@SUBL(55)		
		HR	20
			120
		DM	0100
00004	AND		25504
00005	@BSET(71)		
		#	0000
		DM	0000
		DM	0001

Address	Instruction	Operands	
00006	CLC(41)		
00007	@SUBL(55)		
		DM	0000
		DM	0100
		DM	0100
00008	LD	TR	0
00009	AND		25504
00010	OR	HR	2100
00011	OUT	HR	2100

5-17-9 BCD MULTIPLY – MUL(32)

Ladder Symbols

MUL(32)
Md
Mr
R

@MUL(32)
Md
Mr
R

Operand Data Areas

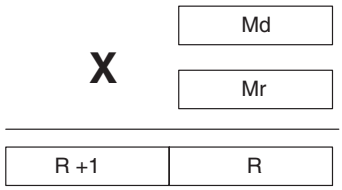
Md: Multiplicand (BCD)
IR, SR, AR, DM, HR, TC, LR, #

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

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

Description

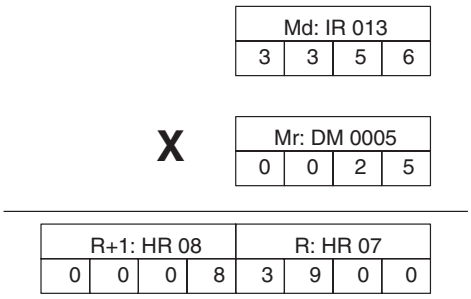
When the execution condition is OFF, MUL(32) is not executed. When the execution condition is ON, MUL(32) multiplies Md by the content of Mr, and places the result In R and R+1.



Example

When IR 00000 is ON with the following program, the contents of IR 013 and DM 0005 are multiplied and the result is placed in HR 07 and HR 08. Example data and calculations are shown below the program.

Address	Instruction	Operands	
00000	LD		00000
00001	MUL(32)		
			013
		DM	00005
		HR	07



Flags

ER: Md and/or Mr is not BCD.

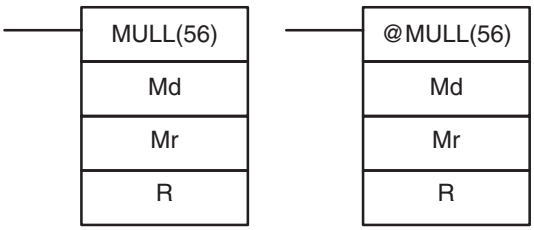
Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

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

EQ: ON when the result is 0.

5-17-10 DOUBLE BCD MULTIPLY – MULL(56)

Ladder Symbols

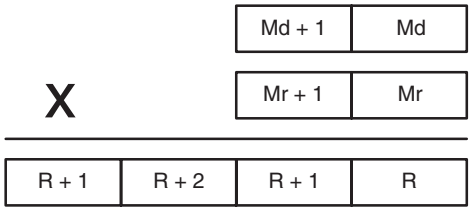


Operand Data Areas

Md: First multiplicand word (BCD)
IR, SR, AR, DM, HR, TC, LR, #
Mr: First multiplier word (BCD)
IR, SR, AR, DM, HR, TC, LR, #
R: First result word
IR, AR, DM, HR LR

Description

When the execution condition is OFF, MULL(56) is not executed. When the execution condition is ON, MULL(56) multiplies the eight-digit content of Md and Md+1 by the content of Mr and Mr+1, and places the result in R to R+3.



Flags

ER: Md, Md+1,Mr, or Mr+1 is not BCD.

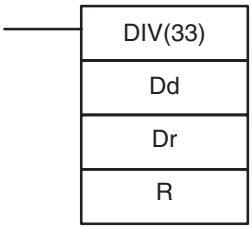
Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

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

EQ: ON when the result is 0.

5-17-11 BCD DIVIDE – DIV(33)

Ladder Symbol



Operand Data Areas

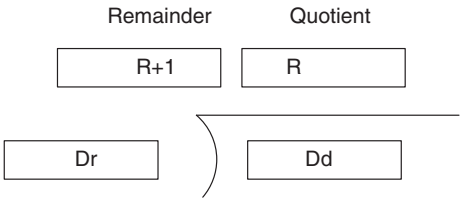
Dd: Dividend word (BCD)
IR, SR, AR, DM, HR, TC, LR, #
Dr: Divisor word (BCD)
IR, SR, AR, DM, HR, TC, LR, #
R: First result word (BCD)
IR, AR, DM, HR, LR

Limitations

R and R+1 must be in the same data area.

Description

When the execution condition is OFF, DIV(33) is not executed and the program moves to the next instruction. When the execution condition is ON, Dd is divided by Dr and the result is placed in R and R + 1: the quotient in R and the remainder in R + 1.



Flags

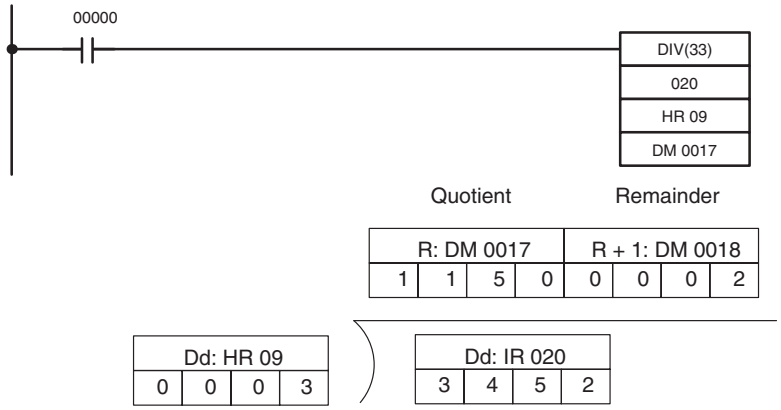
ER: Dd or Dr is not in BCD.

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

EQ: ON when the result is 0.

Example

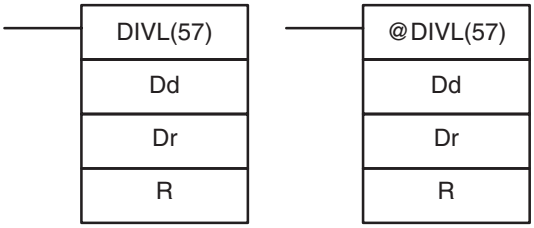
When IR 00000 is ON with the following program, the content of IR 020 is divided by the content of HR 09 and the result is placed in DM 0017 and DM 0018. Example data and calculations are shown below the program.



Address	Instruction	Operands	
00000	LD		00000
00001	DIV(33)		
			020
		HR	09
		DM	0017

5-17-12 DOUBLE BCD DIVIDE – DIVL(57)

Ladder Symbols

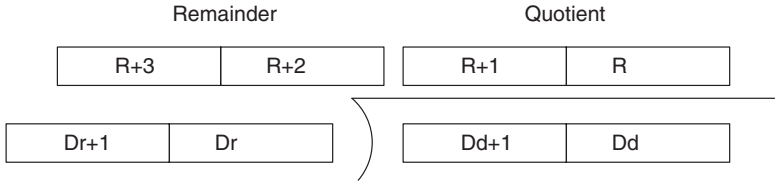


Operand Data Areas

Dd: First dividend word (BCD)	IR, SR, AR, DM, HR, TC, LR
Dr: First divisor word (BCD)	IR, SR, AR, DM, HR, TC, LR
R: First result word	IR, AR, DM, HR LR

Description

When the execution condition is OFF, DIVL(57) is not executed. When the execution condition is ON, DIVL(57) the eight-digit content of Dd and D+1 is divided by the content of Dr and Dr+1 and the result is placed in R to R+3: the quotient in R and R+1, the remainder in R+2 and R+3.



Flags

ER: Dr and Dr+1 contain 0.

Dd, Dd+1, Dr, or Dr+1 is not BCD.

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

EQ: ON when the result is 0.

5-17-13 FLOATING POINT DIVIDE – FDIV(79)

Ladder Symbols

FDIV(79)
Dd
Dr
R

@FDIV(79)
Dd
Dr
R

Operand Data Areas

Dd: First dividend word (BCD)
IR, SR, AR, DM, HR, TC, LR

Dr: First divisor word (BCD)
IR, SR, AR, DM, HR, TC, LR

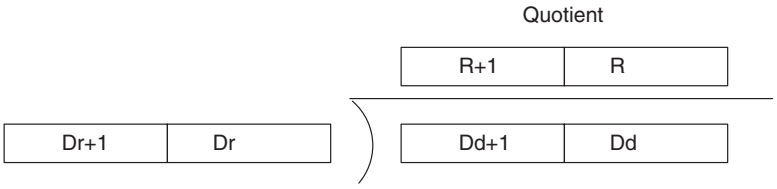
R: First result word
IR, AR, DM, HR LR

Limitations

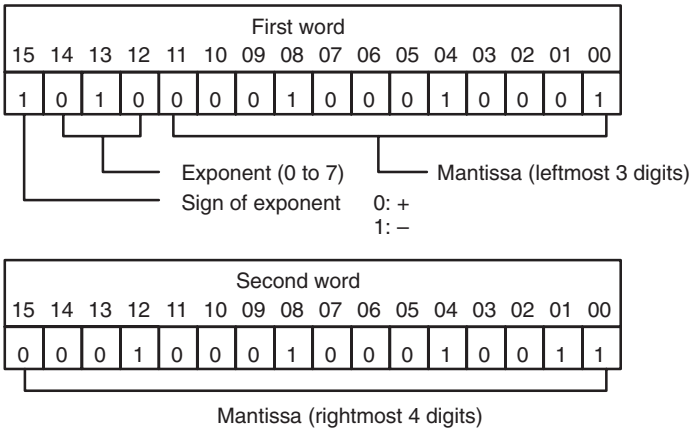
Dr and Dr+1 cannot contain zero. Dr and Dr+1 must be in the same data area, as must Dd and Dd+1; R and R+1.

Description

When the execution condition is OFF, FDIV(79) is not executed. When the execution condition is ON, FDIV(79) divides the floating-point value in Dd and Dd+1 by that in Dr and Dr+1 and places the result in R and R+1.



To represent the floating point values, the rightmost seven digits are used for the mantissa and the leftmost digit is used for the exponent, as shown in the diagram below. The mantissa is expressed as a value less than one, i.e., to seven decimal places.



$= 0.1111113 \times 10^{-2}$

Flags

ER: Dr and Dr+1 contain 0.

Dd, Dd+1, Dr, or Dr+1 is not BCD.

The result is not between 0.0000001×10^{-7} and $0.999999 \times 10^{+7}$.

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

EQ: ON when the result is 0.

Valid Ranges for Division Data and Quotient

Maximum value for division data and quotient (same)
0.9999999×10^{7}

Contents of words Dd+1, Dr+1, or R+1

7				9				9				9			
0	1	1	1	1	0	0	1	1	0	0	1	1	0	0	1

MSB LSB

Contents of words Dd, Dr, or R

9				9				9				9			
1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1

MSB LSB

Minimum value for division data
0.0000001×10^{-7}

Contents of words Dd +1 or Dr+1

F				0				0				0			
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

MSB LSB

Contents of words Dd or Dr

0				0				0				1			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

MSB LSB

Minimum value for quotient
0.100000×10^{-7}

Contents of word R+1

F				1				0				0			
1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0

MSB LSB

Contents of word R

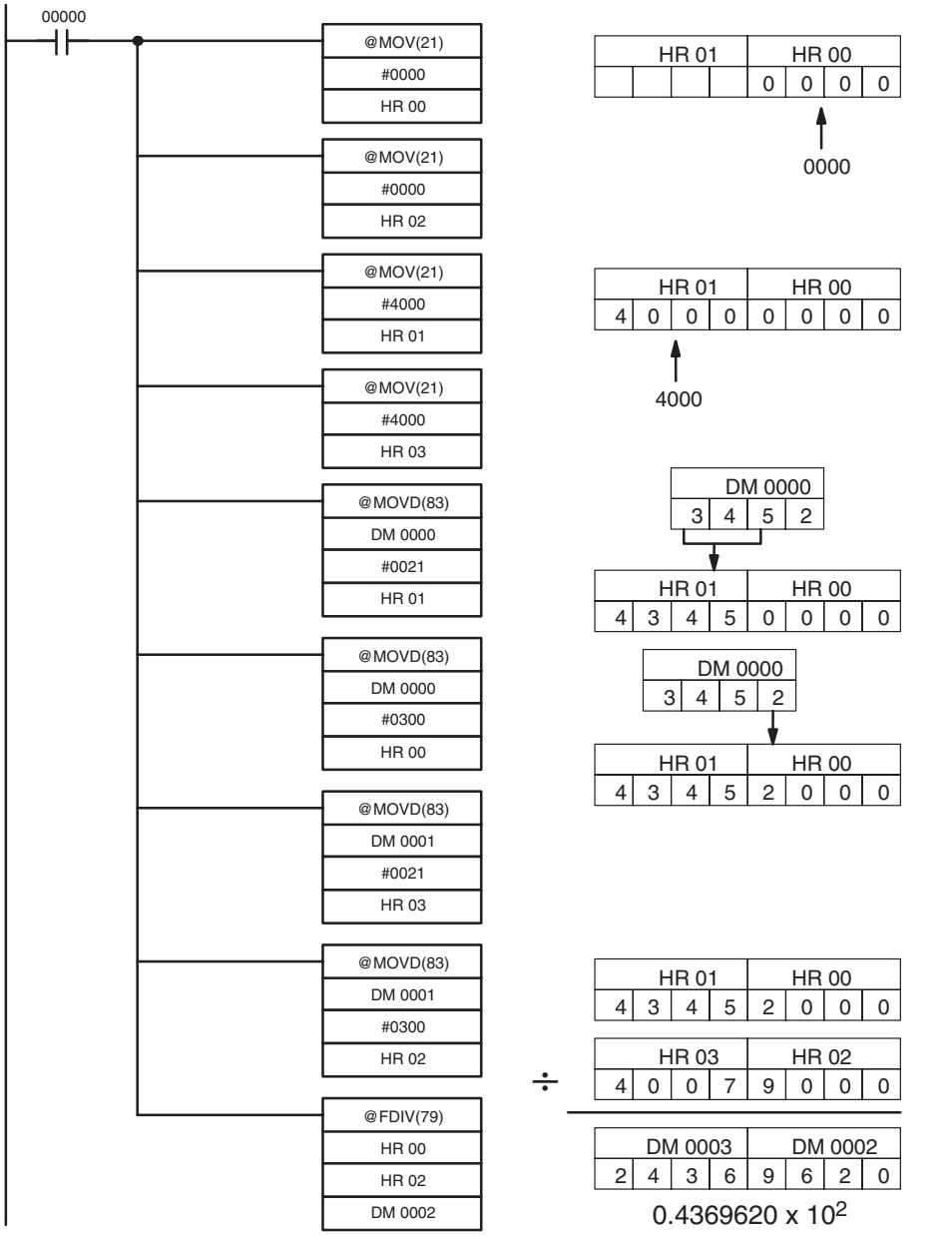
0				0				0				0			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MSB LSB

Example

The following example shows how to divide two whole four-digit numbers (i.e., numbers without fractions) so that a floating-point value can be obtained.

First the original numbers must be placed in floating-point form. Because the numbers are originally without decimal points, the exponent will be 4 (e.g., 3452 would equal 0.3452×10^{4}). All of the moves are to place the proper data into consecutive words for the final division, including the exponent and zeros. Data movements for Dd and Dd+1 are shown at the right below. Movements for Dr and Dr+1 are basically the same.The original values to be divided are in DM 0000 and DM 0001. The final division is also shown.

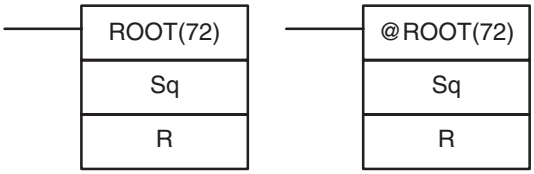


Address	Instruction	Operands
00000	LD	00000
00001	@MOV(21)	
		# 0000
		HR 00
00002	@MOV(21)	
		# 0000
		HR 02
00003	@MOV(21)	
		# 4000
		HR 01
00004	@MOV(21)	
		# 4000
		HR 03
00005	@MOVD(83)	
		DM 0000
		# 0021
		HR 01

Address	Instruction	Operands
00006	@MOVD(83)	
		DM 0000
		# 0300
		HR 00
00007	@MOVD(83)	
		DM 0001
		# 0021
		HR 03
00008	@MOVD(83)	
		DM 0001
		# 0300
		HR 02
00009	@FDIV(79)	
		HR 00
		HR 02
		DM 0002

5-17-14 SQUARE ROOT – ROOT(72)

Ladder Symbols



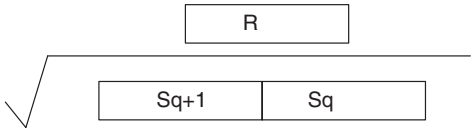
Operand Data Areas

Sq: First source word (BCD)
IR, SR, AR, DM, HR, TC, LR

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

Description

When the execution condition is OFF, ROOT(72) is not executed. When the execution condition is ON, ROOT(72) computes the square root of the eight-digit content of Sq and Sq+1 and places the result in R. The fractional portion is truncated.



Flags

ER: Sq is not BCD.

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

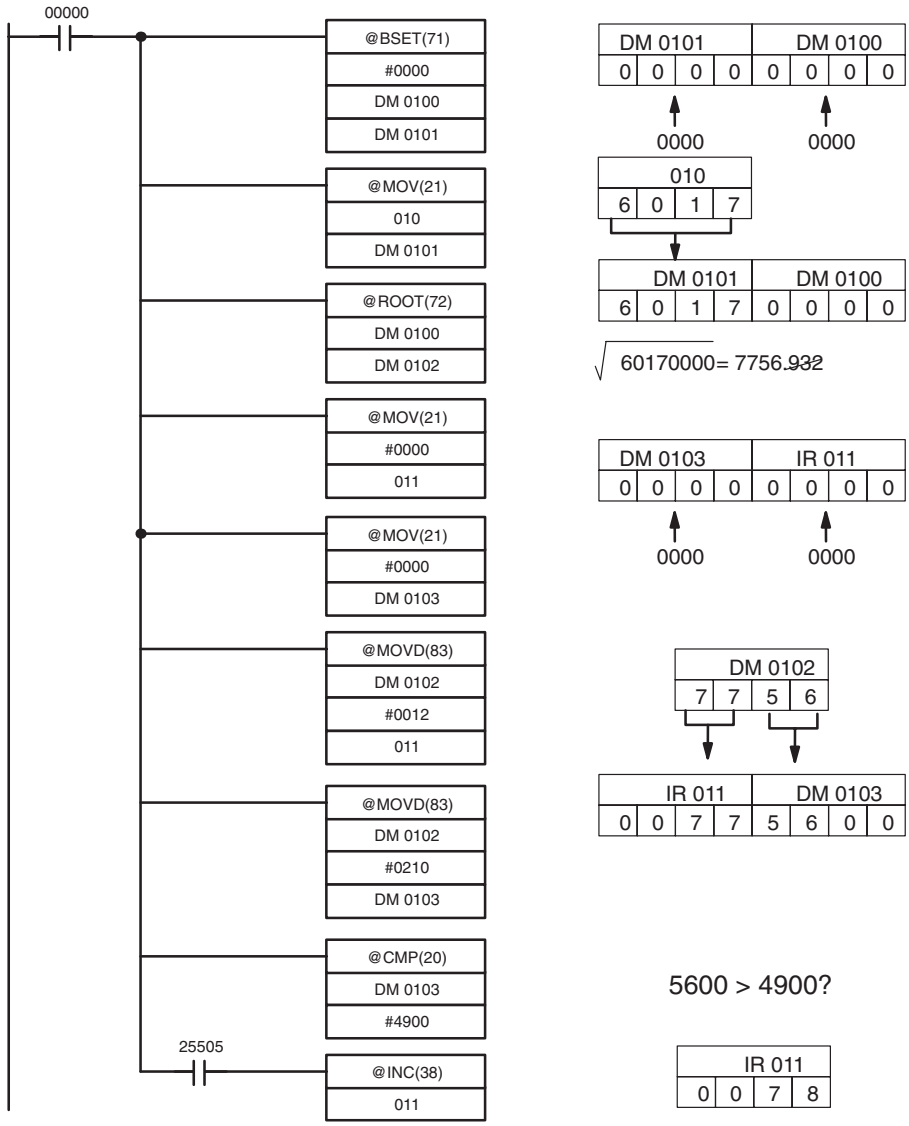
EQ: ON when the result is 0.

Example

The following example shows how to take the square root of a four-digit number and then round the result.

First the words to be used are cleared to all zeros and then the value whose square root is to be taken is moved to Sq+1. The result, which has twice the number of digits required for the answer (because the number of digits in the original value was doubled), is placed in DM 0102, and the digits are split into two different words, the leftmost two digits to IR 011 for the answer and the rightmost two digits to DM 0103 so that the answer in IR 011 can be rounded up if required. The last step is to compare the value in DM 0103 so that IR 011 can be incremented using the Greater Than flag.

In this example, $\sqrt{6017} = 77.56$. The result is rounded off to an integer, according to the digit in the tenths place. A remainder less than 0.5 is rounded to 0, and a remainder of 0.5 or greater is rounded to 1. In this case, 77.56 is rounded off to 78.



Address	Instruction	Operands
00000	LD	00000
00001	@BSET(71)	
		# 0000
		DM 0100
		DM 0101
00002	@MOV(21)	
		010
		DM 0101
00003	@ROOT(72)	
		DM 0100
		DM 0102
00004	@MOV(21)	
		# 0000
		011
00005	@MOV(21)	
		# 0000
		DM 0103

Address	Instruction	Operands
00006	@MOVD(83)	
		DM 0102
		# 0012
		011
00007	@MOVD(83)	
		DM 0102
		# 0210
		DM 0103
00008	@CMP(20)	
		DM 0103
		# 4900
00009	LD	25505
00010	@INC(38)	
		011

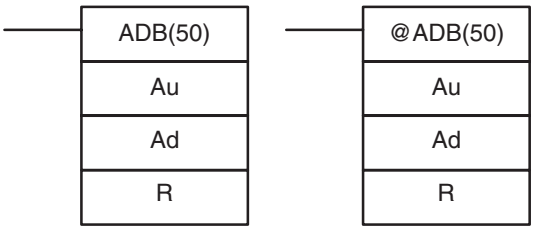
5-18 Binary Calculations

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

The addition and subtraction instructions include CY in the calculation as well as in the result. Be sure to clear CY if its previous status is not required in the calculation, and to use the result placed in CY, if required, before it is changed by the execution of any other instruction. STC(40) and CLC(41) can be used to control CY. Refer to *5-16 BCD Calculations*.

5-18-1 BINARY ADD – ADB(50)

Ladder Symbols



Operand Data Areas

Au: Augend word (binary)
IR, SR, AR, DM, HR, TC, LR, #
Ad: Addend word (binary)
IR, SR, AR, DM, HR, TC, LR, #
R: Result word
IR, AR, DM, HR, LR

Description

When the execution condition is OFF, ADB(50) is not executed. When the execution condition is ON, ADB(50) adds the contents of Au, Ad, and CY, and places the result in R. CY will be set if the result is greater than FFFF.

Au + Ad + CY → CY R

Flags

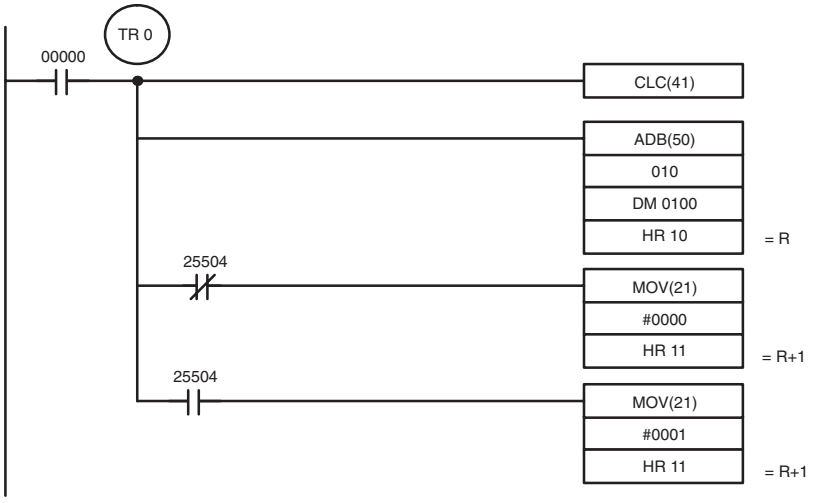
ER: Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

CY: ON when the result is greater than FFFF.

EQ: ON when the result is 0.

Examples

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



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

In the case below, A6E2 + 80C5 = 127A7. The result is a 5-digit number, so CY (SR 25504) = 1, and the content of R + 1 becomes #0001.

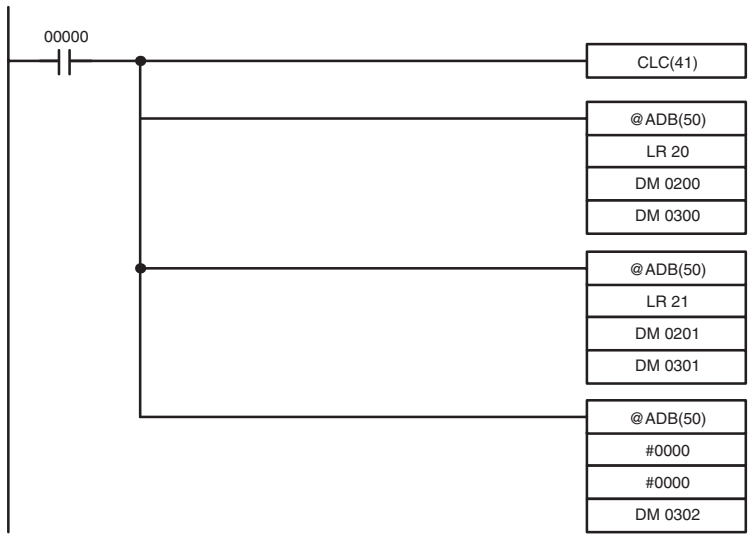
Au: IR 010			
A	6	E	2

\+

Ad: DM 0100			
8	0	C	5

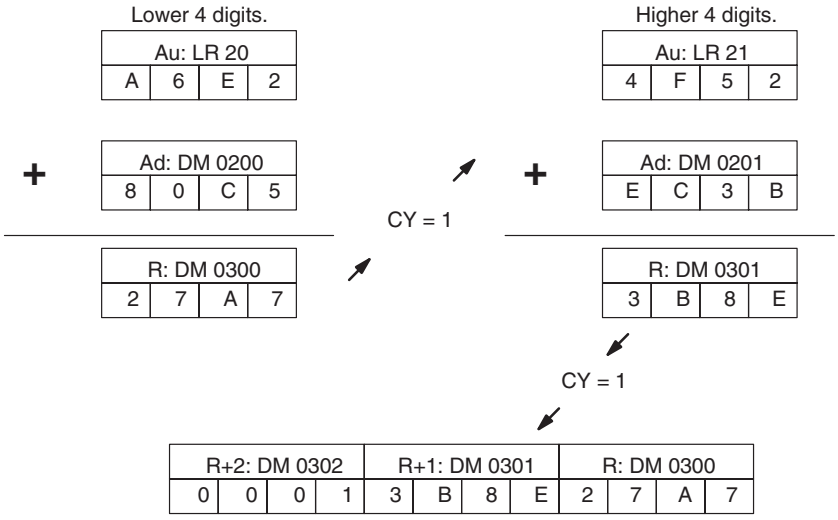
R+1: HR 11				R: HR 10			
0	0	0	1	2	7	A	7

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



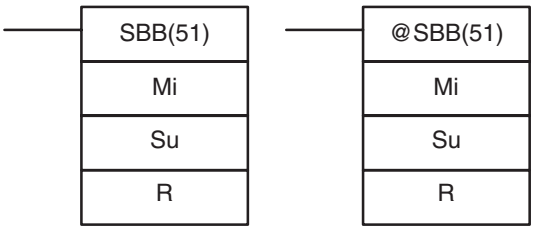
Address	Instruction	Operands	
00000	LD		00000
00001	CLC(41)		
00002	@ADB(50)		
		LR	20
		DM	0200
		DM	0300
00003	@ADB(50)		
		LR	21
		DM	0201
		DM	0301
00004	@ADB(50)		
		#	0000
		#	0000
		DM	0302

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



5-18-2 BINARY SUBTRACT – SBB(51)

Ladder Symbols



Operand Data Areas

Mi: Minuend word (binary)
IR, SR, AR, DM, HR, TC, LR, #
Su: Subtrahend word (binary)
IR, SR, AR, DM, HR, TC, LR, #
R: Result word
IR, AR, DM, HR, LR

Description

When the execution condition is OFF, SBB(51) is not executed. When the execution condition is ON, SBB(51) subtracts the contents of Su and CY from Mi and places the result in R. If the result is negative, CY is set and the 2's complement of the actual result is placed in R.

Flags

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

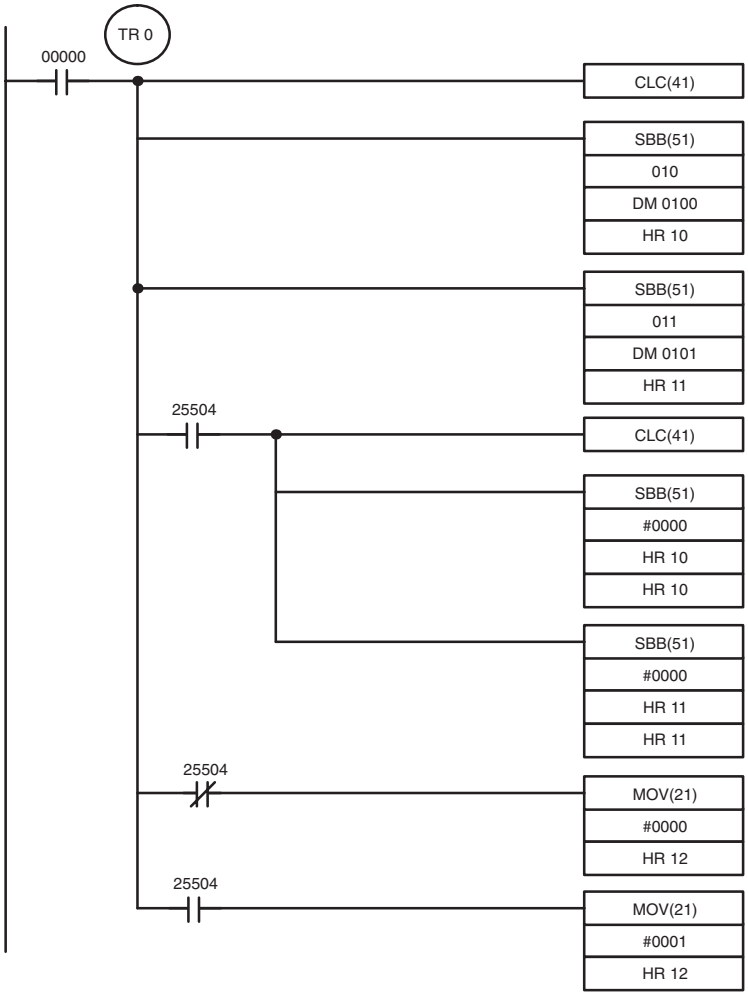
CY: ON when the result is negative, i.e., when Mi is less than Su plus CY.

EQ: ON when the result is 0.

Example

The following example shows eight-digit subtraction. CY is tested following the first two subtractions to see if the result is negative. If it is, the first result is subtracted from zero to obtain the true result, which is placed in HR 10 and

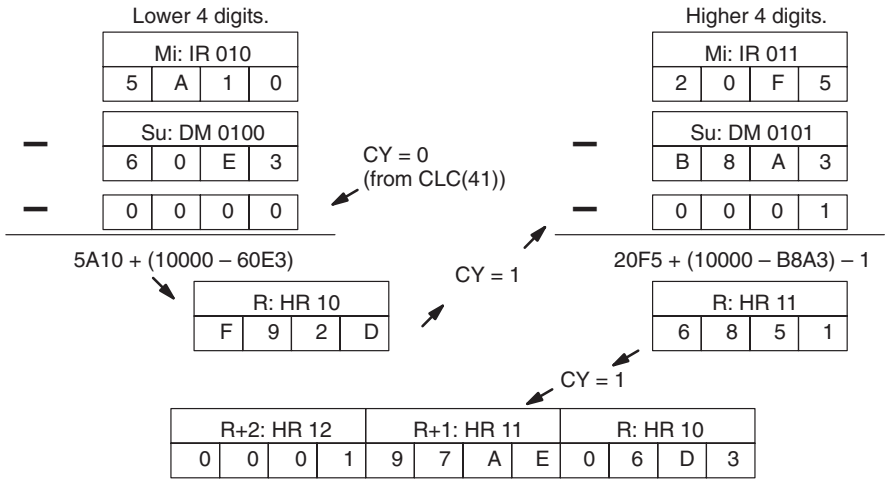
HR 11, and either 0000 or 0001 is placed in HR 12 (0001 indicates a negative answer).



Address	Instruction	Operands	
00000	LD		00000
00001	OUT	TR	0
00002	CLC(41)		
00003	SBB(51)		
			010
		DM	0100
		HR	10
00004	SBB(51)		
			011
		DM	0101
		HR	11
00005	AND		25505
00006	CLC(41)		
00007	SBB(51)		
		#	0000
		HR	10
		HR	10
00008	SBB(51)		
		#	0000
		HR	11
		HR	11
00009	LD	TR	0
00010	AND NOT		25504
00011	MOV(21)		
		#	0000
		HR	12
00012	LD	TR	0
00013	AND		25504
00014	MOV(21)		
		#	0000
		HR	12

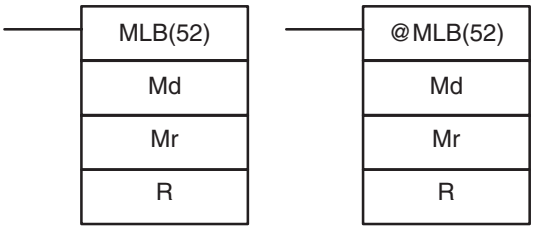
In the case below, 20F55A10 – B8A360E3 = 97AE06D3. In the the lower 4-digit subtraction, Su > Mi, so CY(SR 25504) becomes 1, and the result of the higher 4-digit subtraction is decremented by 1. In the final calculations, #0000 – F9D2 = 0000 + (10000 – F9D2) = 06D3.

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



5-18-3 BINARY MULTIPLY – MLB(52)

Ladder Symbols

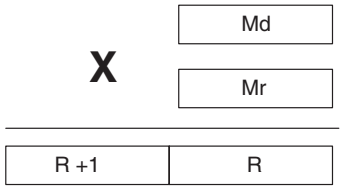


Operand Data Areas

Md: Multiplicand word (binary)
IR, SR, AR, DM, HR, TC, LR, #
Mr: Multiplier word (binary)
IR, SR, AR, DM, HR, TC, LR, #
R: First result word
IR, AR, DM, HR LR

Description

When the execution condition is OFF, MLB(52) is not executed. When the execution condition is ON, MLB(52) multiplies the content of Md by the contents of Mr, places the rightmost four digits of the result in R, and places the leftmost four digits in R+1.



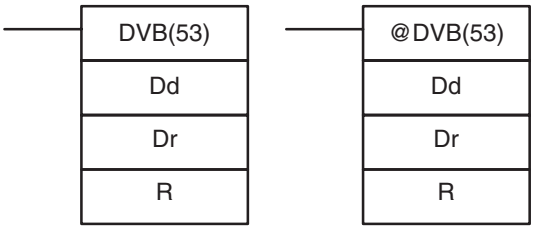
Flags

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

EQ: ON when the result is 0.

5-18-4 BINARY DIVIDE – DVB(53)

Ladder Symbols

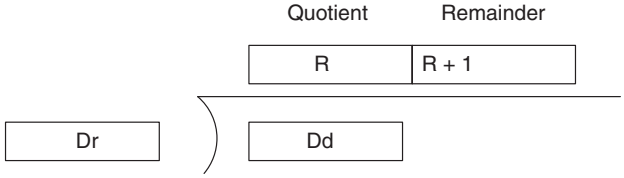


Operand Data Areas

Dd: Dividend word (binary)	IR, SR, AR, DM, HR, TC, LR, #
Dr: Divisor word (binary)	IR, SR, AR, DM, HR, TC, LR, #
R: First result word	IR, AR, DM, HR LR

Description

When the execution condition is OFF, DVB(53) is not executed. When the execution condition is ON, DVB(53) divides the content of Dd by the content of Dr and the result is placed in R and R+1: the quotient in R, the remainder in R+1.



Flags

ER: Dr contains 0.

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

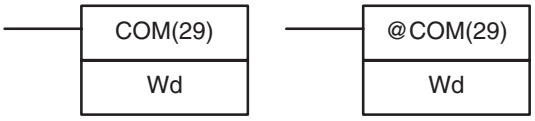
EQ: ON when the result is 0.

5-19 Logic Instructions

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

5-19-1 COMPLEMENT – COM(29)

Ladder Symbols



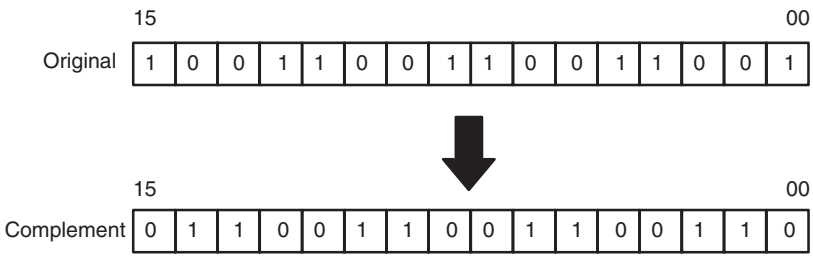
Operand Data Areas

Wd: Complement word
IR, AR, DM, HR, LR

Description

When the execution condition is OFF, COM(29) is not executed. When the execution condition is ON, COM(29) clears all ON bits and sets all OFF bits in Wd.

Example



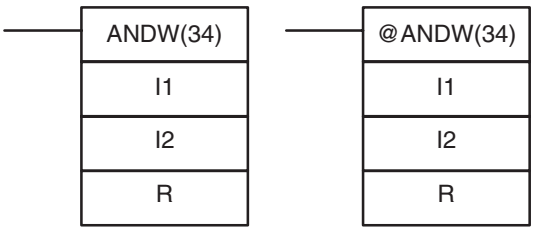
Flags

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

EQ: ON when the result is 0.

5-19-2 AND WORD – ANDW(34)

Ladder Symbols



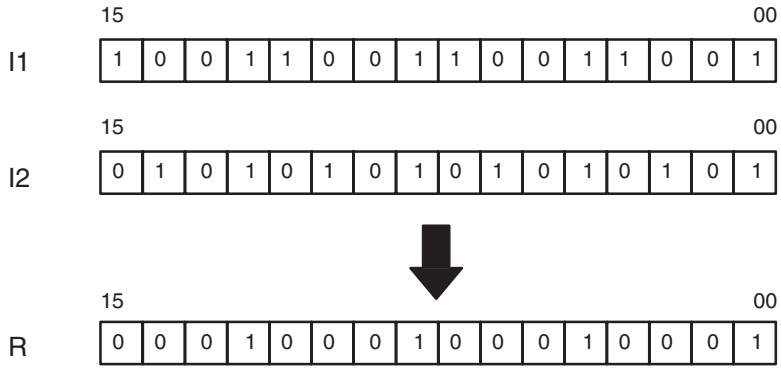
Operand Data Areas

I1: Input 1
IR, SR, AR, DM, HR, TC, LR, #
I2: Input 2
IR, SR, AR, DM, HR, TC, LR, #
R: Result word
IR, AR, DM, HR, LR

Description

When the execution condition is OFF, ANDW(34) is not executed. When the execution condition is ON, ANDW(34) logically AND's the contents of I1 and I2 bit-by-bit and places the result in R.

Example



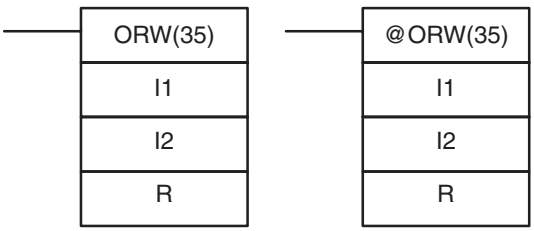
Flags

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

EQ: ON when the result is 0.

5-19-3 OR WORD – ORW(35)

Ladder Symbols



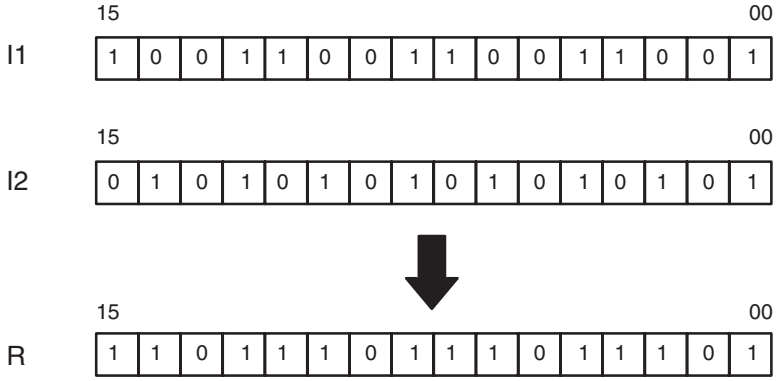
Operand Data Areas

I1: Input 1
IR, SR, AR, DM, HR, TC, LR, #
I2: Input 2
IR, SR, AR, DM, HR, TC, LR, #
R: Result word
IR, AR, DM, HR, LR

Description

When the execution condition is OFF, ORW(35) is not executed. When the execution condition is ON, ORW(35) logically OR's the contents of I1 and I2 bit-by-bit and places the result in R.

Example



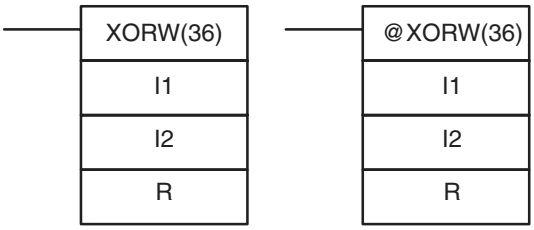
Flags

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

EQ: ON when the result is 0.

5-19-4 EXCLUSIVE OR – XORW(36)

Ladder Symbols



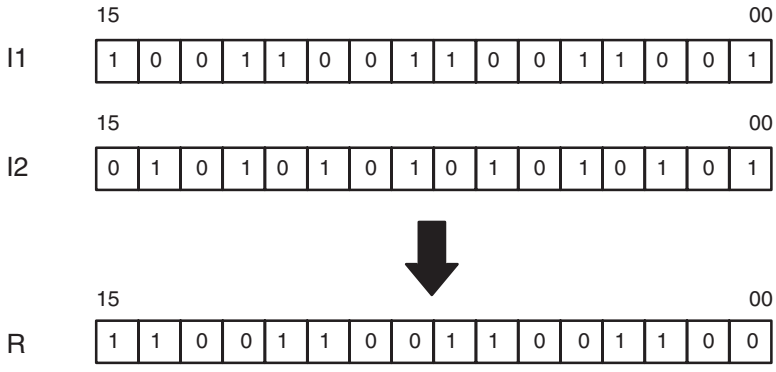
Operand Data Areas

I1: Input 1
IR, SR, AR, DM, HR, TC, LR, #
I2: Input 2
IR, SR, AR, DM, HR, TC, LR, #
R: Result word
IR, AR, DM, HR, LR

Description

When the execution condition is OFF, XORW(36) is not executed. When the execution condition is ON, XORW(36) exclusively OR's the contents of I1 and I2 bit-by-bit and places the result in R.

Example



Flags

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

EQ: ON when the result is 0.

5-19-5 EXCLUSIVE NOR – XNRW(37)

Ladder Symbols

XNRW(37)
I1
I2
R

@XNRW(37)
I1
I2
R

Operand Data Areas

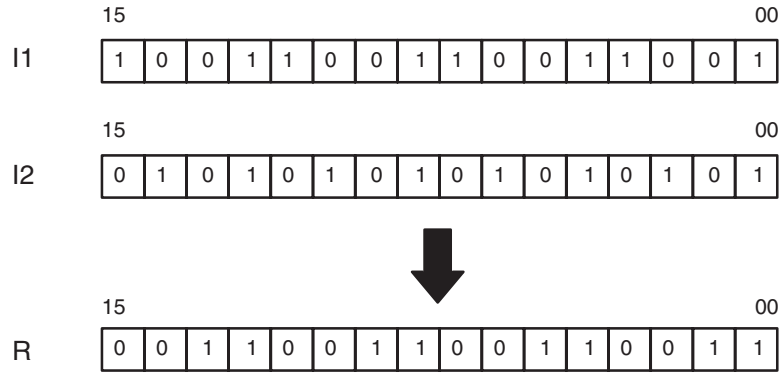
I1: Input 1
IR, SR, AR, DM, HR, TC, LR, #

I2: Input 2
IR, SR, AR, DM, HR, TC, LR, #

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

Description

When the execution condition is OFF, XNRW(37) is not executed. When the execution condition is ON, XNRW(37) exclusively NOR's the contents of I1 and I2 bit-by-bit and places the result in R.



Flags

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

EQ: ON when the result is 0.

5-20 Subroutines and Interrupt Control

5-20-1 Overview

Subroutines break large control tasks into smaller ones and enable you to reuse a given set of instructions. When the main program calls a subroutine, control is transferred to the subroutine and the subroutine instructions are executed. The instructions within a subroutine are written in the same way as main program code. When all the subroutine instructions have been executed, control returns to the main program to the point just after the point from which the subroutine was entered (unless otherwise specified in the subroutine).

Subroutines may also be activated by interrupts. Like subroutine calls, interrupts cause a break in the flow of the main program execution such that the flow can be resumed from that point after completion of the subroutine. An interrupt is caused either by an external source, such as an input signal from an Interrupt Input Unit, or a scheduled interrupt. In the case of the scheduled interrupt, the interrupt signal is repeated at regular intervals.

Whereas subroutine calls are controlled from within the main program, subroutines activated by interrupts are triggered when the interrupt signal is received. Also, multiple interrupts from different Interrupt Input Units can occur

at the same time. To effectively deal with this, the PC employs a priority system for handling interrupts.

In the case of the scheduled interrupt, the time interval between interrupts is set by the user and is unrelated to the cycle timing of the PC. This capability is useful for periodic supervisory or executive program execution.

INT(89) is used to control the interrupt signals received from the Interrupt Input Units, and also to control the scheduling of the scheduled interrupt. INT(89) provides such functions as masking of interrupts (so that they are recorded but ignored) and clearing of interrupts.

5-20-2 SUBROUTINE START and RETURN – SBN(92)/RET(93)

Ladder Symbols **Operand Data Areas**

N: Subroutine number
(00 to 99)

Limitations

Each subroutine number can be used in SBN(92) once only, i.e., up to 100 subroutines may be programmed. Subroutine numbers 00 through 31 are used by Interrupt Input Units and subroutine number 99 is used for the scheduled interrupt. Refer to *5-19-4 Interrupt Control – INT(89)* for details.

Description

SBN(92) is used to mark the beginning of a subroutine program; RET(93) is used to mark the end. Each subroutine is identified with a subroutine number, N, that is programmed as a definer for SBN(92). This same subroutine number is used in any SBS(91) that calls the subroutine (see next subsection). No subroutine number is required with RET(93).

All subroutines must be programmed at the end of the main program. When one or more subroutines have been programmed, the main program will be executed up to the first SBN(92) before returning to address 00000 for the next cycle. Subroutines will not be executed unless called by SBS(91) or activated by an interrupt.

END(01) must be placed at the end of the last subroutine program, i.e., after the last RET(93). It is not required at any other point in the program. (Refer to the next subsection for further details.)

Precautions

If SBN(92) is mistakenly placed in the main program, it will inhibit program execution past that point, i.e., program execution will return to the beginning when SBN(92) is encountered.

If either DIFU(13) or DIFU(14) is placed within a subroutine, the operand bit will not be turned OFF until the next time the subroutine is executed, i.e., the operand bit may stay ON longer than one cycle.

Flags

There are no flags directly affected by these instructions.

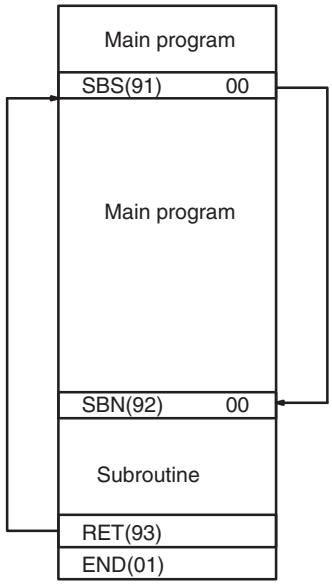
5-20-3 SUBROUTINE ENTER – SBS(91)

Ladder Symbol **Operand Data Areas**

N: Subroutine number
(00 to 99)

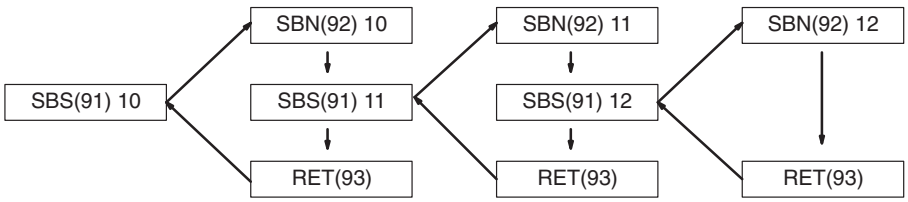
Description

A subroutine can be executed by placing SBS(91) in the main program at the point where the subroutine is desired. The subroutine number used in SBS(91) indicates the desired subroutine. When SBS(91) is executed (i.e., when the execution condition for it is ON), the instructions between the SBN(92) with the same subroutine number and the first RET(93) after it are executed before execution returns to the instruction following the SBS(91) that made the call.



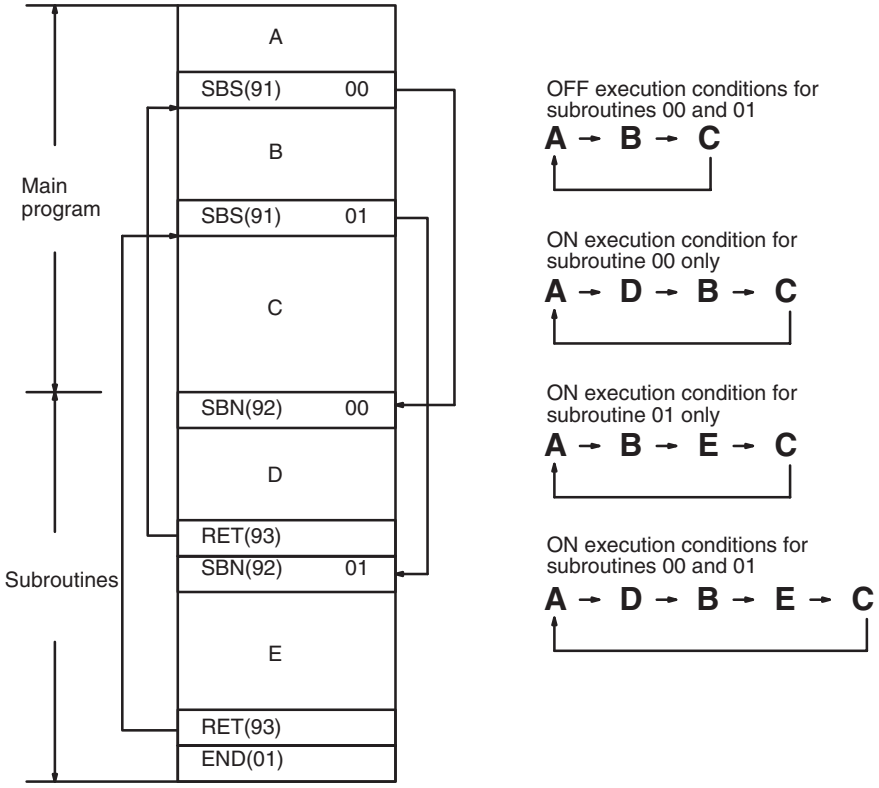
SBS(91) may be used as many times as desired in the program, i.e., the same subroutine may be called from different places in the program).

SBS(91) may also be placed into a subroutine to shift program execution from one subroutine to another, i.e., subroutines may be nested. When the second subroutine has been completed (i.e., RET(93) has been reached), program execution returns to the original subroutine which is then completed before returning to the main program. Nesting is possible to up to sixteen levels. A subroutine cannot call itself, (e.g., SBS(91) 00 cannot be programmed within the subroutine defined with SBN(92) 00). The following diagram illustrates two levels of nesting.



Although subroutines 00 through 31 can be called by using SBS(91), they are also activated by interrupt signals from Interrupt Input Units. Subroutine 99, which can also be called using SBS(91), is used for the scheduled interrupt. (Refer to the next subsection for details.)

The following diagram illustrates program execution flow for various execution conditions for two SBS(91).



Flags

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

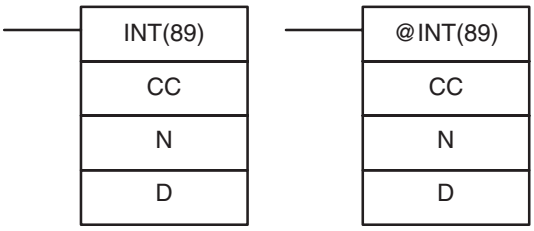
A subroutine has called itself.

Subroutines have been nested to more than sixteen levels.

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

5-20-4 INTERRUPT CONTROL – INT(89)

Ladder Symbols



Operand Data Areas

CC: Control code
(000 to 002)

N: Interrupt designator
(000 to 004)

D: Control data
IR, AR, DM, HR, TC, LR, TR, #

Limitations

D may be a constant only when CC is 000 or 001. D must be a word address when CC is 002. See below for details.

⚠ Caution INT(89) cannot be used during execution of step programs or in C2000H Duplex CPUs. Refer to *5-21 Set Instructions* for details on step programs.

Description

INT(89) is used both to control interrupts from Interrupt Input Units and to control the scheduled interrupt. If N is 000, 001, 002, or 003, it indicates an Interrupt Input Unit number and INT(89) is used to control interrupts from the designated Unit. If N is 004, INT(89) is used to control the scheduled interrupt. Interrupts from Interrupt Input Units and scheduled interrupts are covered separately.

Interrupts from Interrupt Input Units (N = 000, 001, 002, or 003)

Description

Up to four (0 to 3) Interrupt Input Units can be used for one PC. Interrupt Input Unit numbers are assigned sequentially starting from 0 and following the same order as the numbers of the I/O channels to which the Units are mounted.

For each Interrupt Input Unit, bits 00 through 07 may be used for interrupt signals. Bits 08 through 15 are not used. When one of the bits assigned to an Interrupt Input Unit turns ON, the subroutine associated with it is called and executed. A unique interrupt routine number is associated with each bit according to the following table.

Interrupt Input Unit		Subroutine	Interrupt Input Unit		Subroutine
Unit no.	Bit no.		Unit no.	Bit no.	
0	0	00	2	0	16
	1	01		1	17
	2	02		2	18
	3	03		3	19
	4	04		4	20
	5	05		5	21
	6	06		6	22
	7	07		7	23
1	0	08	3	0	24
	1	09		1	25
	2	10		2	26
	3	11		3	27
	4	12		4	28
	5	13		5	29
	6	14		6	30
	7	15		7	31

The subroutines used by these interrupts may also be called from the program using SBS(91). Calls to interrupt routines will generate the error message “SBS UNDEFD” during program check, but it will not inhibit program execution.

CC is used to specify the desired operation as shown below. The function of D will vary with the value of CC (see below).

CC = 000: Masking/unmasking interrupts
001: Clearing interrupts
002: Accessing the current mask status

CC = 000 (Mask/Unmask)

A control code of CC = 000 causes those bits of the designated Interrupt Input Unit corresponding to ON bits in D to be masked, and those corresponding to OFF bits in D to be unmasked. All masked interrupts will still be recorded. When a masked bit has been recorded as being ON, the subroutine

for it will be run as soon as the bit is unmasked (unless it is cleared first – see below). All interrupts are initially masked.

CC = 001 (Clear)

A control code of CC = 001 causes the masks on those bits of the designated Interrupt Input Unit, corresponding to ON bits in D to be cleared so that the subroutine will not be executed even if the interrupt is unmasked. Because interrupt inputs are stored, masked interrupts will be serviced after the mask is removed, unless they are cleared first.

CC = 002 (Read Mask)

A control code of CC = 002 writes the current mask status of the designated Interrupt Input Unit into D.

Scheduled Interrupts (N = 004)

Description

Subroutine 99 can be established so that it will be executed repeatedly at a fixed interval through scheduled interrupts.The actual time at which it is executed is independent of the cycle time. INT(89) is used to control the scheduled interrupt. If N is 004, CC is used to designate the desired function as follows:

CC = 000: Setting time interval
001: Setting the time to first scheduled interrupt
002: Reading the current time interval

Scheduling the Interrupt

Even when a subroutine 99 has been written, it will not be executed according to scheduled interrupts unless INT(89) is used to set the proper times. INT(89) should be used to set both the time interval (CC = 000) for the scheduled interrupt and the time to the first scheduled interrupt (CC = 001. Unstable operation may result is the time to the first interrupt is not set.

CC = 000 (Interval)

To set the time interval for the scheduled interrupt, set CC to 000 and set D to any value between 00.01 and 99.99 seconds. The decimal point is not input. The time interval can be changed at any time.

To cancel the scheduled interrupt, set the time interval to 00.00 seconds.

⚠ Caution

If the scheduled execution time of the subroutine becomes too large, it will have a serious effect on the overall execution time of the main program. Therefore, you should take extra care to write a subroutine that is fast and efficient. INT(89), with a CC of 000, is used to change the scheduled interrupt time interval, the new time interval is not effective until after the next scheduled interrupt. (c.f. CC = 001 below.)

CC = 001 (Time to First Interrupt)

To set the time to the first interrupt, set CC to 001 and set D to any value between 00.01 and 99.99 seconds. The decimal point is not entered. If D is set to 00.00, the interrupt will not occur.

⚠ Caution

INT(89), with a CC code of 001, can be used to change the scheduled interrupt time interval for one cycle. The new time interval is effective immediately. The scheduled interrupt may never actually occur if the time to the first interrupt is changed repeatedly, i.e., before the interrupt has time to occur.

CC = 002 (Read Interval)

To access the current time interval for the scheduled interrupt, set CC 002. The current time interval will be placed in D.

Flags

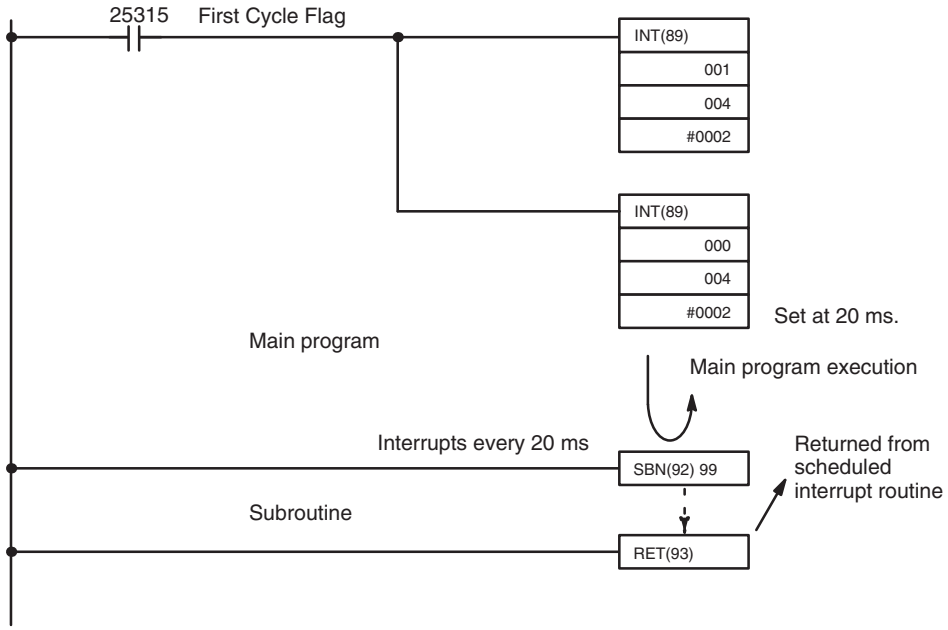
ER: CC, D, or N is not within specified values.

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

Examples

The following program shows the overall structure and operation of the scheduled interrupt.

Here, the scheduled subroutine is started and will be repeated every 20 ms. The control flow logic of the main program is unaffected by execution of the scheduled subroutine, i.e., immediately after the sub
routine has finished execution, control returns to the point in the main program where it was suspended.

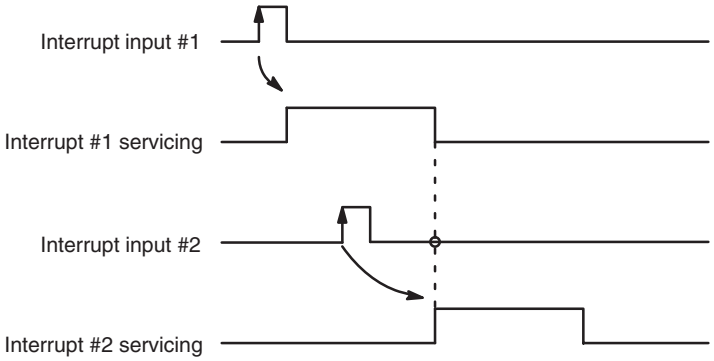


Address	Instruction	Operands
00000	LD	25315
00001	INT(89)	
		001
		004
		# 0002
00002	INT(89)	
		000
		004
		# 0002

Address	Instruction	Operands
	Main program.	
00500	SBN(92)	99
	Subroutine.	
00600	RET(93)	

Interrupt Priority Levels

When an interrupt is being serviced (i.e. the subroutine is executing), another incoming interrupt must wait until the first is finished before it will be serviced.



If two or more interrupt inputs are turned ON simultaneously, the subroutine with the lowest subroutine number takes precedence. For example, interrupts from interrupt Unit 0, bit 0 will be serviced before interrupts from Unit 2, bit 3. Subroutines for interrupts from Interrupt Input Units will be interrupted for the scheduled interrupt.

5-21 Block Programming Instructions

Block programming facilitates coding operations which are difficult to write with normal ladder diagrams, i.e. ladder diagrams which include sequential arithmetic operations or conditional branching. It also reduces the overall cycle time. Up to 100 block programs may be defined in a given main program.

5-21-1 Overview

Block programming instructions are not part of the 'ladder' section of a ladder-diagram program. Rather, they are written in sequence down the right side of the ladder diagram. Block programming instructions still require execution conditions, and most of them require operands.

Instructions dedicated for use within block programs are distinguished by pointed parentheses <like these>. For example, FUN<02> is IF, whereas IL(02) is IL. BPRG(96), which is used to start a section of block programming instructions, is a normal ladder-diagram program instructions. Block programming instructions are treated as NOP(00) if programmed outside a block program.

Instructions Not Available for Use Within Block Programs

None of the following instructions can be used within a block program:

OUT	OUT NOT	TIM	CNT
END(01)	IL(02)	ILC(03)	JMP(04)
JME(05)	STEP(08)	SNXT(09)	SFT(10)
KEEP(11)	CNTR(12)	DIFU(13)	DIFD(14)
TIMH(15)	SBN(92)	RET(93)	

All Differentiated Instructions (i.e., those preceded by @)

If these instructions are included with in block programs, they will be treated as NOP(00). Other instructions may be used within a block program. Instructions such as LD, AND, and OR are still used to create execution conditions. In a block program, however, these are written in their mnemonic form, rather than in the normal ladder-diagram form.

5-21-2 BLOCK PROGRAM BEGIN – BPRG(96) and BLOCK PROGRAM END – BEND<01>

Ladder Symbols

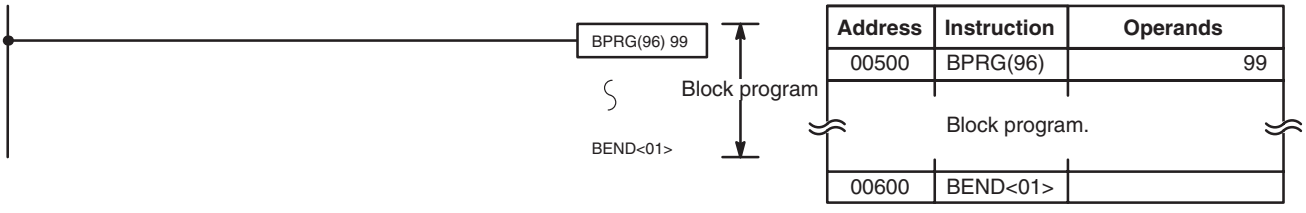
BEND<01>

Definer Data Areas

N: Block program number
(00 to 99)

Description

BPRG(96) is used to switch to block programming and BEND<01> is used to switch back to ladder-diagram programming. For every BPRG(96) there must be a corresponding BEND<01>.



5-21-3 SET – SET<07> and RESET – RSET<08>

Instruction Formats		Operand Data Areas
SET<07>	B	**B**: Bit
RSET<08>	B	IR, AR, HR, LR

Description SET<07> turns B ON, and RSET<08> turns B OFF.

Flags No flags are affected by this instruction.

5-21-4 Block Branching–IF<02>, IF<02>NOT, ELSE<03>, and IEND<04>

Instruction Formats		Operand Data Areas
IF<02>	B	**B**: Bit
IF<02>		IR, SR, AR, HR, TC, LR
IF<02> NOT	B	
ELSE<03>		
IEND<04>		

Description These instructions are used to branch according to either the current execution condition or the status of a designated bit. IF<02> and IF<02> NOT must be used in combination with IEND<04). ELSE<03> may be used in between them, but is optional.

Branching is initiated with any of the following: IF<02> with a bit operand, IF<02> without a bit operand, or IF<02> NOT with a bit operand.

If the IF condition is YES, the instructions immediately following the IF<02> or IF<02> NOT will be executed. A YES execution condition is produced by an ON bit or ON execution condition for IF<02> or an OFF bit for IF<02>NOT.

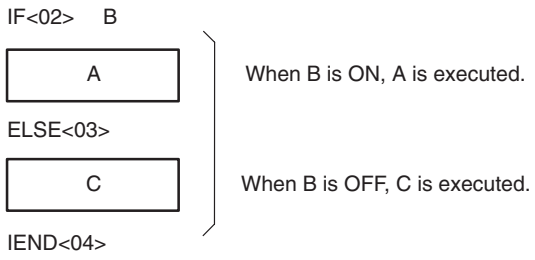
If ELSE<03> is encountered following IF<02> or IF<02>NOT, execution will jump to IEND<03> without executing any instruction in between. If ELSE<03> is not encountered, execution will continue as normal.

If the IF condition is NO, execution will jump to ELSE<03> or to IEND<04>, whichever appears first after the IF<02> or IF<02> NOT.

LD, possible in combination with AND or OR, must be used to establish the execution condition for IF<02> or IF<02> NOT without an operand.

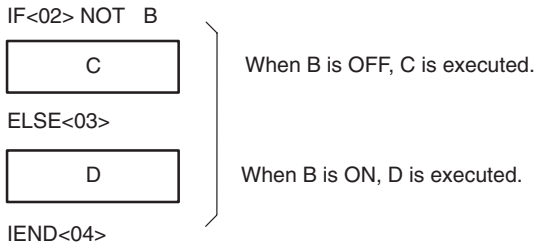
Execution Flow Examples

IF<02> with an Operand IF<02> to ELSE to IEND



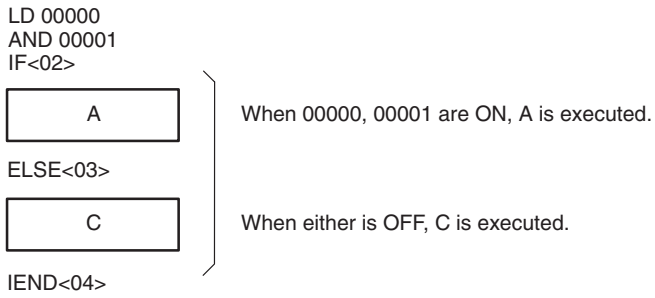
IF<02> NOT with an Operand

IF<02> NOT to ELSE to IEND



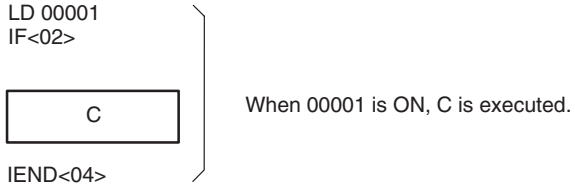
IF<02> without an Operand

IF<02> to ELSE to IEND



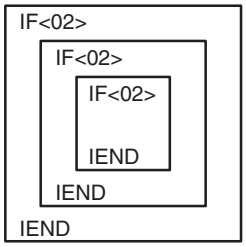
IF<02> without ELSE

IF<02> to IEND



Nesting

IF<02> blocks can be nested up to a maximum of 253 levels. Each IF<02> or IF<02> NOT will be effective through the next ELSE<03> and/or IEND<04>.

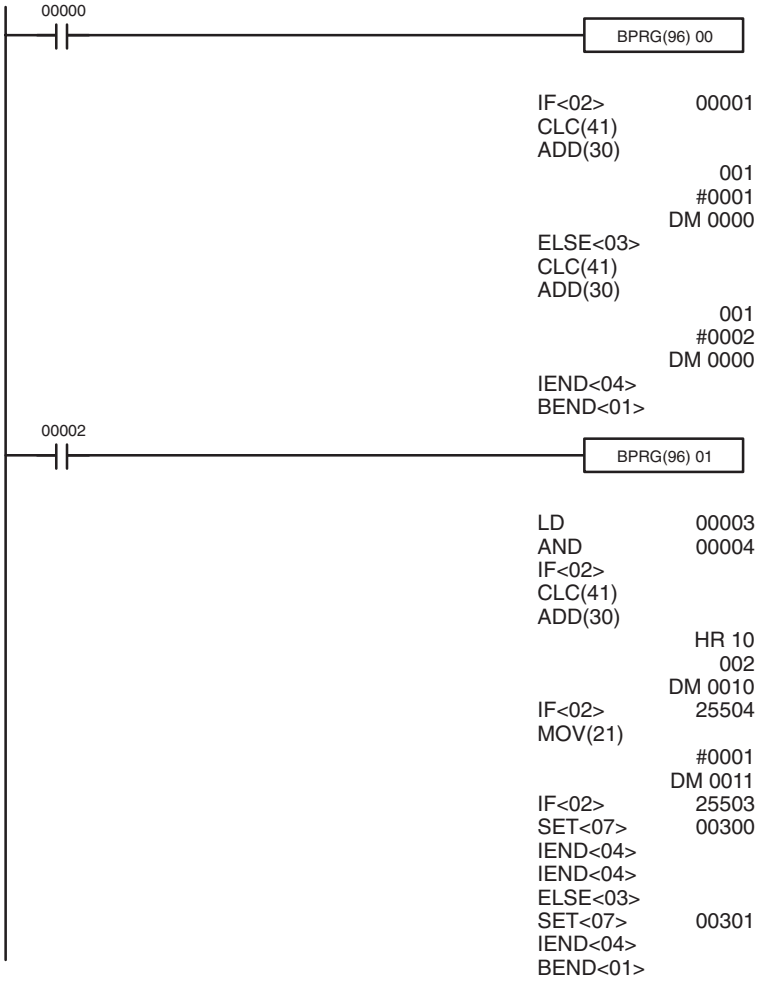


Flags

No flags are affected by this instruction.

Example

The following example shows two different block programs controlled by 00000 and 00002. The first block executes one of two additions depending on the status of 00001. The second block shows nesting to two levels.



Address	Instruction	Operands
00000	LD	00000
00001	BPRG(96)	00
00002	IF<02>	00001
00003	CLC(41)	
00004	ADD(30)	
		001
		# 0001
		DM 0000
00005	ELSE<03>	
00006	CLC(41)	
00007	ADD(30)	
		001
		# 0002
		DM 0000
00008	IEND<04>	
00009	BEND<01>	
00010	LD	00002
00011	BPRG(96)	01
00012	LD	00003
00013	AND	00004
00014	IF<02>	
00015	CLC(41)	
00016	ADD(30)	
		HR 10
		002
		DM 0010
00017	IF<02>	25504
00018	MOV(21)	
		# 0001
		DM 0011
00019	IF<02>	25503
00020	SET<07>	00300
00021	IEND<04>	
00022	IEND<04>	
00023	ELSE<03>	
00024	SET<07>	00301
00025	IEND<04>	
00026	BEND<01>	

5-21-5 ONE CYCLE AND WAIT – WAIT<05>

Instruction Formats

WAIT<05>	
WAIT<05>	B
WAIT<05> NOT	B

Operand Data Areas

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

Description

WAIT<05> and WAIT<05> NOT allow you to inhibit execution of the portion of block program from WAIT<05> to BEND<01> until B turns ON.

As long of the execution condition or operand bit of WAIT<05> is ON, or the operand bit of WAIT<05> NOT is OFF, the block program will be executed as normal. If the execution condition or operand bit of WAIT<05> is OFF or the operand bit of WAIT<05> NOT is ON, only the part of the block program up to the WAIT<05> or WAIT<05> NOT instruction will be executed during the

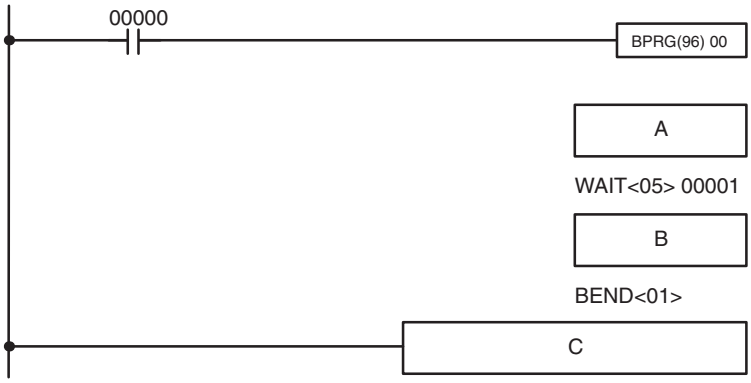
first cycle. During following cycles, none of the block program will be executed until the operand bit or execution condition changes, at which point the remainder of the block program will be executed. Once the entire block program has been executed, the process is repeated.

Precautions

WAIT<05> NOT cannot be used without an operand bit.

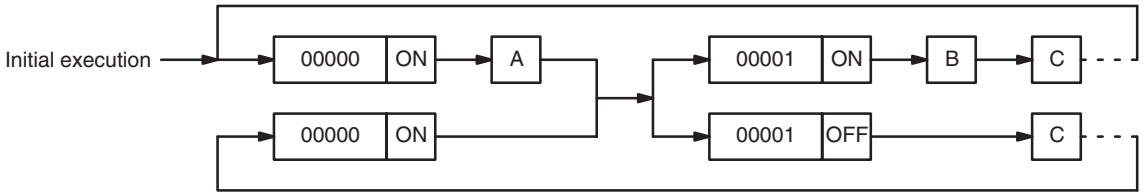
Execution Flow Examples

When 00000 is ON, the block program is executed as normal. If 00001 is OFF, however, A is executed and then B is skipped and program control jumps to BEND<01>. During the following cycles, until 00001 turns ON, no instructions within block 00 will be executed (except WAIT <05>).

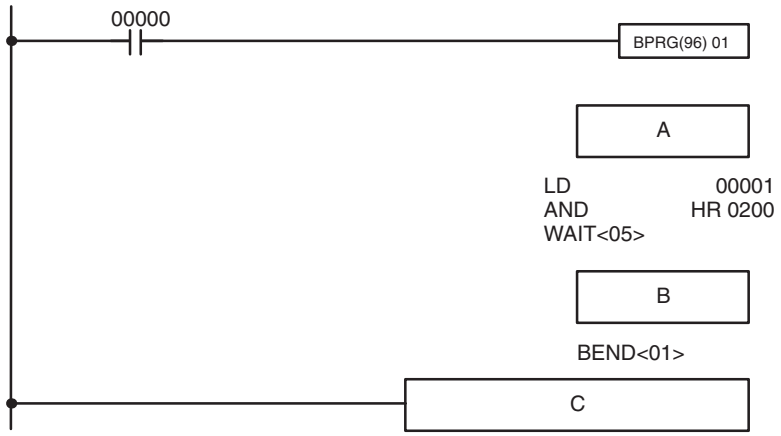


Address	Instruction	Operands
00000	LD	00000
00001	BPRG(96)	00
	A	
00100	WAIT<05>	00001
	B	
00200	BEND<01>	
	C	

The execution flow for this example would be as shown below:



The following example would work similarly, except that execution of WAIT<05> would be based on an AND between the status of 00001 and HR 0200.



Address	Instruction	Operands
00000	LD	00000
00001	BPRG(96)	01
	A	
00200	LD	00001
00201	AND	HR 0200
00202	WAIT<05>	
	B	
00300	BEND<01>	
	C	

5-21-6 TIMER WAIT – TIMW<13> and HIGH-SPEED TIMER WAIT – TMHW<15>

Instruction Formats

TIMW<13>	N SV
TMHW<15>	N SV

Definer Data Areas

N: Timer number
TC

Operand Data Areas

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

Limitations

SV is between 000.0 and 999.9 for TIMW<13>, and between 00.00 and 99.99 for TMHW<15>. The decimal point is not entered.

Each TC number can be used as the definer in only one timer or counter instruction, including those used in normal ladder-diagram timers and counters.

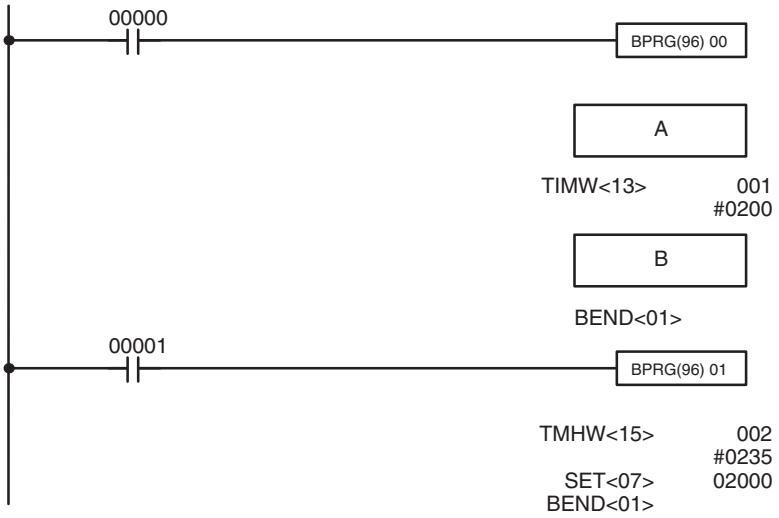
If cycle time is greater than 10 ms, TC 000 through TC 047 must be used for TMHW<15> to ensure accuracy.

Description

TIMW<13> and TMHW<15> allow you to create a specified time lag (SV) between execution of the program part preceding it and the part following. The first part will be executed the first time the block program is entered. When the block timer instruction is reached, execution of the block program will halt until SV has expired, at which time the second part of the block program will be executed. Once the entire block program has been executed, the process is repeated.

Example

In the following example, B will be executed 20 seconds after A whenever 00000 is ON, and 02000 will be set 2.35 seconds after 00001 goes ON.



Address	Instruction	Operands
00000	LD	00000
00001	BPRG(96)	00
	A	
00200	TIMW<13>	001
		# 0200
	B	
00300	BEND<01>	
00301	LD	00001
00302	BPRG(96)	01
00303	TMHW<15>	002
		# 0235
00304	SET<07>	02000
00305	BEND<01>	

Flags

ER: SV data is not BCD.

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

5-21-7 COUNTER WAIT – CNTW<14>

Instruction Format

CNTW<14>	N
	SV
	I

Definer Data Areas

N: Counter number
TC

Operand Data Areas

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

I: Count input
IR, SR, AR, HR, TC, LR

Limitations

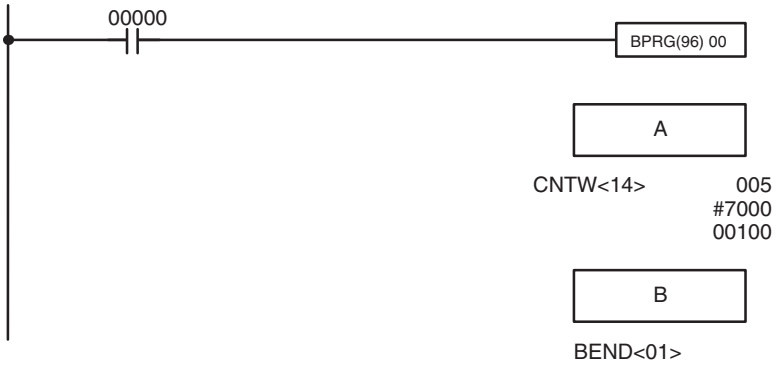
Each TC number can be used as the definer in only one timer or counter instruction, including the normal ladder-diagram timers and counters.

Description

CNTW<14> allows you to create a 'count' lag (SV) between execution of the program part preceding the CNTW<14> (i.e., between BPRG(96) and CNTW<14>) and the part following it (i.e., between CNTW<14> and BEND<01>). The first part will be executed the first time the block program is entered. When CNTW<14> is reached, the execution of the block program will stop until SV has been reached, at which time the second part of the block program will be executed. Once the entire block program has been executed, the process is repeated.

Example

In the following example, B will be executed after the execution of A and after 7,000 counts of 00100 while 00000 is ON.



Address	Instruction	Operands
00000	LD	00000
00001	BPRG(96)	00
	A	
00200	CNTW<14>	005
		# 7000
		00100
	B	
00300	BEND<01>	

Flags

ER: SV data is not BCD.

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

5-21-8 CONDITIONAL BLOCK EXIT – EXIT<06> and EXIT<06> NOT

Instruction Formats

EXIT<06>	
EXIT<06>	B
EXIT<06> NOT	B

Operand Data Areas

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

Description

EXIT<06> and EXIT<06> NOT conditionally end execution of the block program in which they occur, based on either the execution condition or the operand bit.

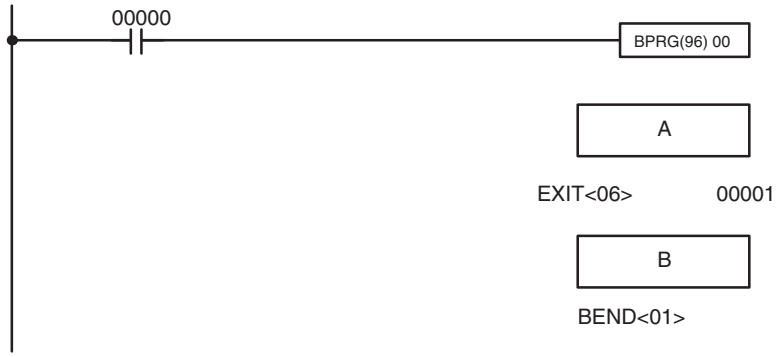
If the EXIT condition is YES when EXIT<06> or EXIT<05> NOT is reached, execution moves directly to BEND<01> without executing any more instructions in the block. If the EXIT condition is NO, the block program is executed normally.

A YES EXIT condition is produced by an ON execution condition for EXIT<06> without an operand, by an ON bit for EXIT<06> with an operand bit, or by an OFF bit for EXIT<06> NOT with an operand bit.

LD, possibly in combination with AND or OR, must be used to create an execution condition for EXIT<06> when used without an operand bit.

Example

In the following example, B will be executed only when 00001 is OFF.



Address	Instruction	Operands
00000	LD	00000
00001	BPRG(96)	00
	A	
00200	EXIT<06>	00001
	B	
00300	BEND<01>	

5-21-9 Block Loop Control–LOOP<09>, LEND<10>, and LEND<10> NOT

Instruction Formats

LOOP<09>	
LEND<10>	
LEND<10>	B
LEND<10> NOT	B

Operand Data Areas

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

Description

LOOP<09> and LEND<10> are used to create a loop that is repeatedly executed until the LOOP END condition becomes YES. LOOP<09> designates the beginning of the loop program, and a LEND<10> or LEND<10> NOT instruction specifies the end of the loop. When LEND<10> or LEND<10> NOT

is reached, program execution will loop back to the next previous LOOP<09> an exit condition is attained.

A YES LOOP END condition is produced by an ON execution condition for LEND<10> without an operand, by an ON bit for LEND<10> with an operand bit, or by an OFF bit for LEND<10> NOT with an operand bit.

LD, possibly in combination with AND or OR, must be used to create an execution condition for LEND<10> when used without an operand bit.

Note Execution inside a loop does not refresh I/O data. If I/O data must be refreshed during the loop, use IORF(97).

Precautions

- Conditional block branching can be used within a loop, but the entire branch operation must be within the loop.

Correct:

```
LOOP<09>
  IF<02>
      IF<02>
      IEND<04>
  IEND<04>
LEND<10>
```

Incorrect:

```
LOOP<09>
  IF<02>
IF<02>
      IEND<04>
LEND<10>
```

- Loops cannot be nested within loops.

Incorrect:

```
LOOP<09>
  LOOP<09>
  LEND<10>
LEND<10>
```

- Do not reverse the order of LOOP and LEND.

Incorrect:

```
LEND<10>
:
:
LOOP<09>
```

5-21-10 BLOCK PROGRAM PAUSE – BPPS<11> and BLOCK PROGRAM RESTART – BPRS<12>

Instruction Formats		Definer Data Areas
BPPS<11>	N	**N**: Block program number
BPRS<12>	N	# (00 to 99)

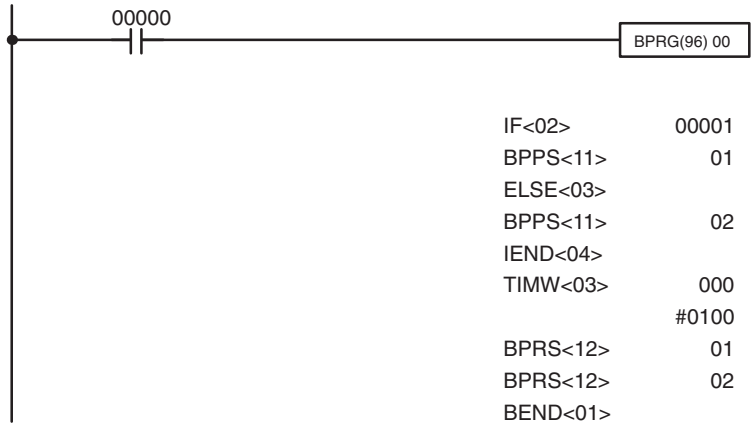
Description

BPPS<11> is used inside one block program to suspend the execution of another block program. BPRS<12> restarts the specified block program. These instructions are effective whenever executed, i.e., they do not rely on operand bit status or execution condition.

Example

If 00000 is ON, the following program suspends execution of either block program 01 or block program 02 depending on the status of 00001. The block

program that was suspended is then restarted after 10 seconds. Note that the ladder diagram simply has the mnemonic code typed after the BPRG(96) instruction.

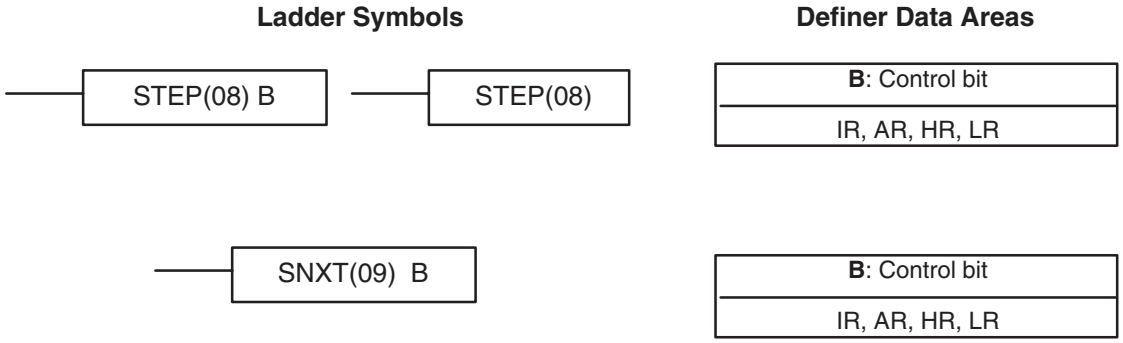


Address	Instruction	Operands
00000	LD	00000
00001	BPRG(96)	00
00002	IF<02>	00001
00003	BPPS<11>	01
00004	ELSE<03>	
00005	BPPS<11>	02
00006	IEND<04>	
00007	TIMW<03>	000
		# 0100
00008	BPRS<12>	01
00009	BPRS<12>	02
00010	BEND<01>	

5-22 Step Instructions

The step instructions STEP(08) and SNXT(09) are used in conjunction to set up breakpoints between sections in a large program so that the sections can be executed as units and reset upon completion. A section of program will usually be defined to correspond to an actual process in the application. (Refer to the application examples later in this section.) A step is like a normal programming code, except that certain instructions (e.g. IL(02)/ILC(03), JMP(04)/JME(05)) may not be included.

5-22-1 STEP DEFINE and STEP START–STEP(08)/SNXT(09)



Limitations

Control bits within one section of step programming must be sequential and from the same word.

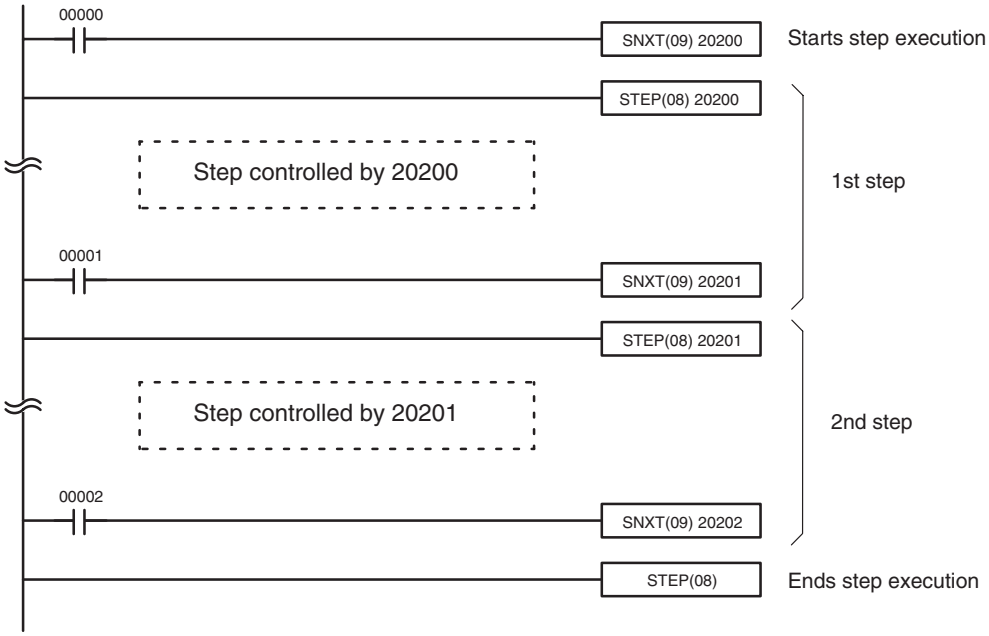
Description

STEP(08) uses a control bit in the IR or HR areas to define the beginning of a section of the program called a step. STEP(08) does not require an execution condition, i.e., its execution is controlled through the control bit. To start execution of the step, SNXT(09) is used with the same control bit as used for STEP(08). If SNXT(09) is executed with an ON execution condition, the step with the same control bit is executed. If the execution condition is OFF, the step is not executed. The SNXT(09) instruction must be written into the pro-

gram so that it is executed before the program reaches the step it starts. It can be used at different locations before the step to control the step according to two different execution conditions (see example 2, below). Any step in the program that has not been started with SNXT(09) will not be executed.

Once SNXT(09) is used in the program, step execution will continue until STEP(08) is executed without a control bit. STEP(08) without a control bit must be preceded by SNXT(09) with a dummy control bit. The dummy control bit may be any unused IR or HR bit. It cannot be a control bit used in a STEP(08).

Execution of a step is completed either by execution of the next SNXT(09) or by turning OFF the control bit for the step (see example 3 below). When the step is completed, all of the IR and HR bits in the step are turned OFF and all timers in the step are reset to their SVs. Counters, shift registers, and bits used in KEEP(11) maintain status. Two simple steps are shown below.



Address	Instruction	Operands
00000	LD	00000
00001	SNXT(09)	20200
00002	STEP(08)	20200
Step controlled by 20200.		
00100	LD	00001
00101	SNXT(09)	20201

Address	Instruction	Operands
00102	STEP(08)	20201
Step controlled by 20201.		
00200	LD	00002
00201	SNXT(09)	20202
00202	STEP(08)	---

Steps can be programmed in consecutively. Each step must start with STEP(08) and generally ends with SNXT(09) (see example 3, below, for an exception). When steps are programmed in series, three types of execution are possible: sequential, branching, or parallel. The execution conditions for, and the positioning of, SNXT(09) determine how the steps are executed. The three examples given below demonstrate these three types of step execution.

Precautions

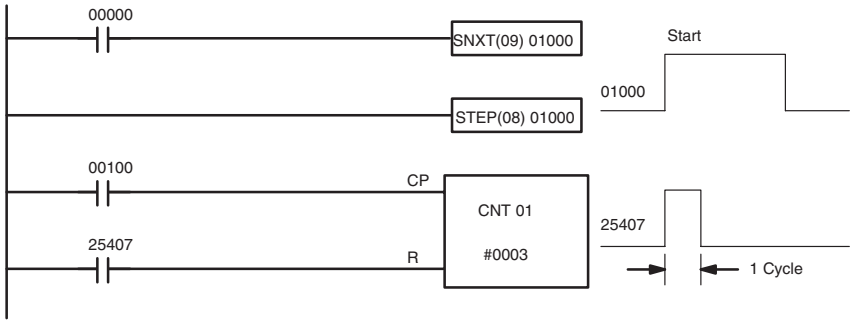
Interlocks, jumps, SBN(92), and END(01) cannot be used within step programs.

Bits used as control bits must not be used anywhere else in the program unless they are being used to control the operation of the step (see example 3, below).

If IR or LR bits are used for control bits, their status will be lost during any power interruption. If it is necessary to maintain status to resume execution at the same step, HR bits must be used.

Flags

25407: Step Start Flag; turns ON for one cycle when STEP(08) is executed and can be used to reset counters in steps as shown below if necessary.



Address	Instruction	Operands
00000	LD	00000
00001	SNXT(09)	01000
00002	STEP(08)	01000
00003	LD	00100

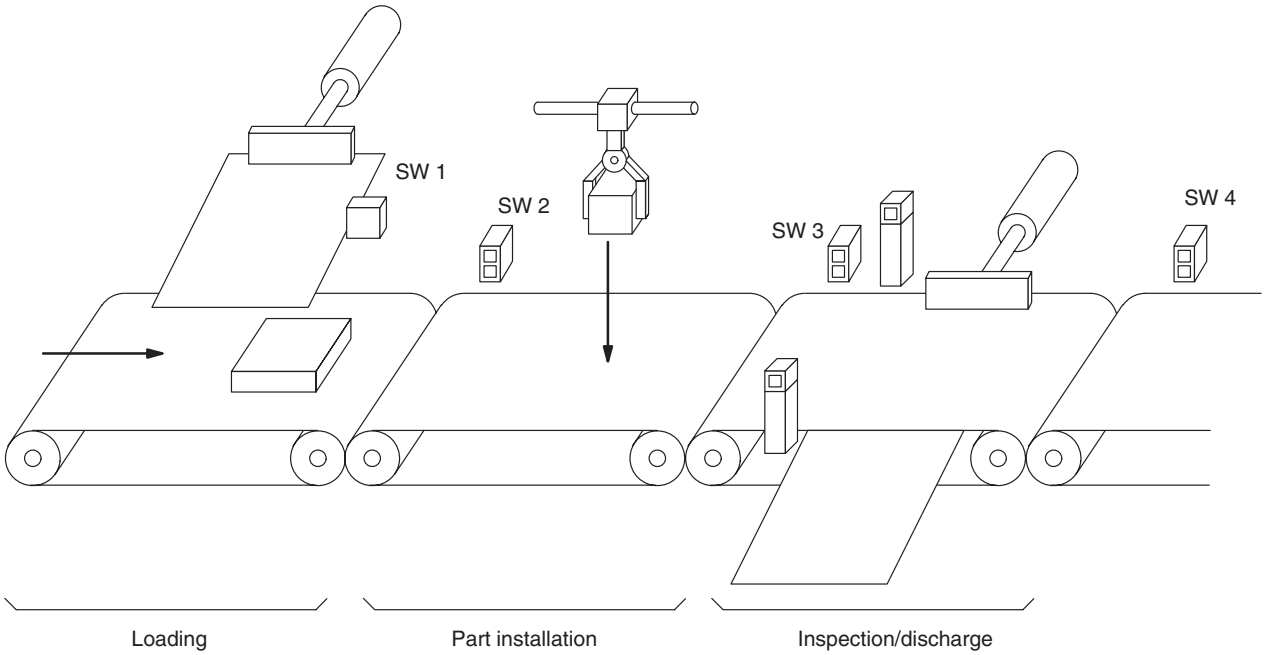
Address	Instruction	Operands
00004	LD	25407
00005	CNT	01
		# 0003

Examples

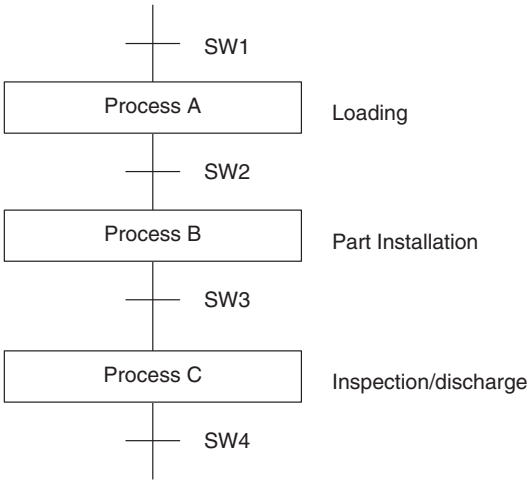
The following three examples demonstrate the three types of execution control possible with step programming. *Example 1* demonstrates sequential execution; *Example 2*, branching execution; and *Example 3*, parallel execution.

Example 1: Sequential Execution

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

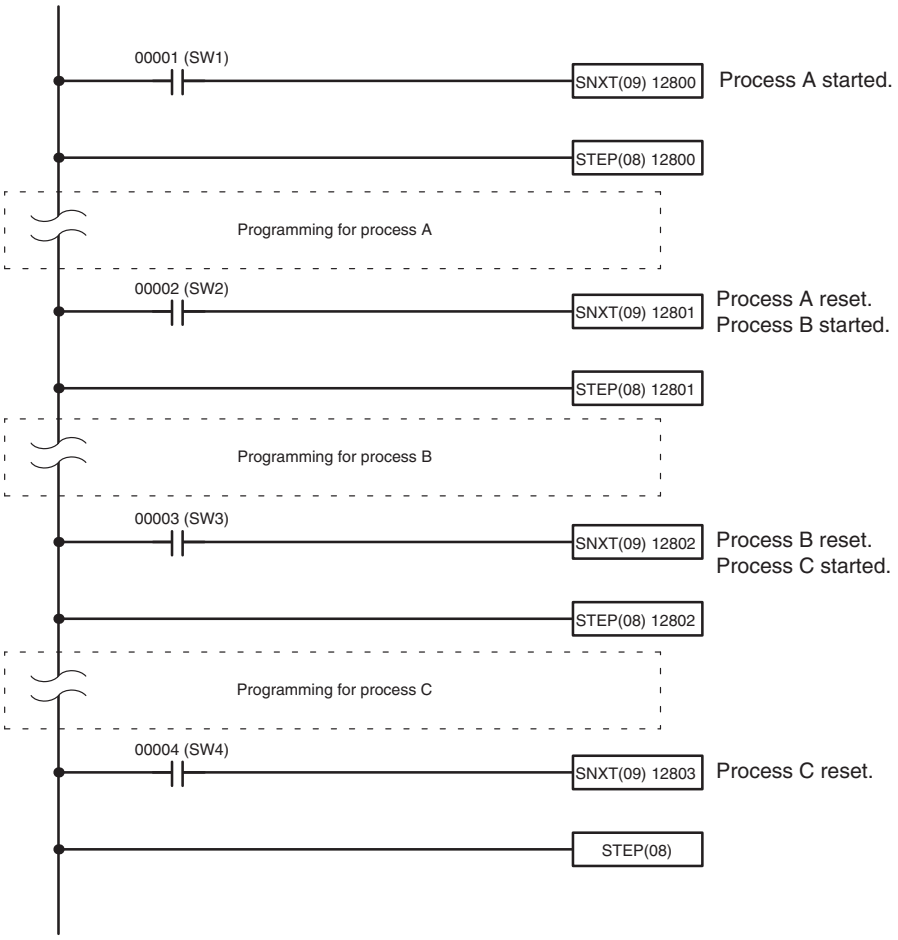


The following diagram demonstrates the flow of processing and the switches that are used for execution control.



The program for this process, shown below, utilizes the most basic type of step programming: each step is completed by a unique SNXT(09) that starts

the next step. Each step starts when the switch that indicates the previous step has been completed turns ON.

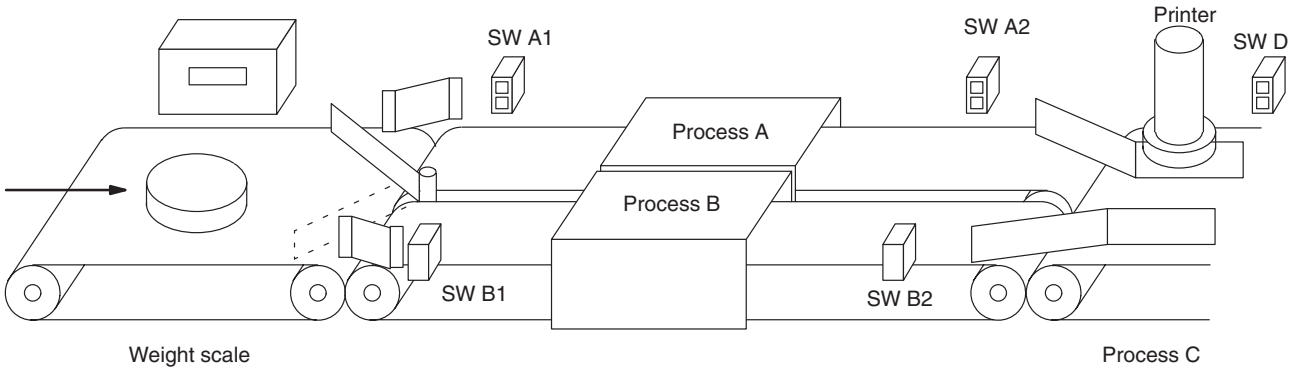


Address	Instruction	Operands
00000	LD	00001
00001	SNXT(09)	12800
00002	STEP(08)	12800
	Process A	
00100	LD	00002
00101	SNXT(09)	12801
00102	STEP(08)	12801

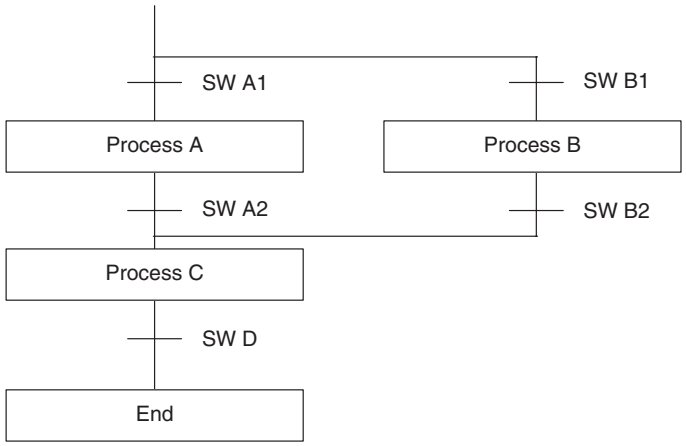
Address	Instruction	Operands
	Process B	
00100	LD	00003
00101	SNXT(09)	12802
00102	STEP(08)	12802
	Process C	
00200	LD	00004
00201	SNXT(09)	12803
00202	STEP(08)	---

Example 2: Branching Execution

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

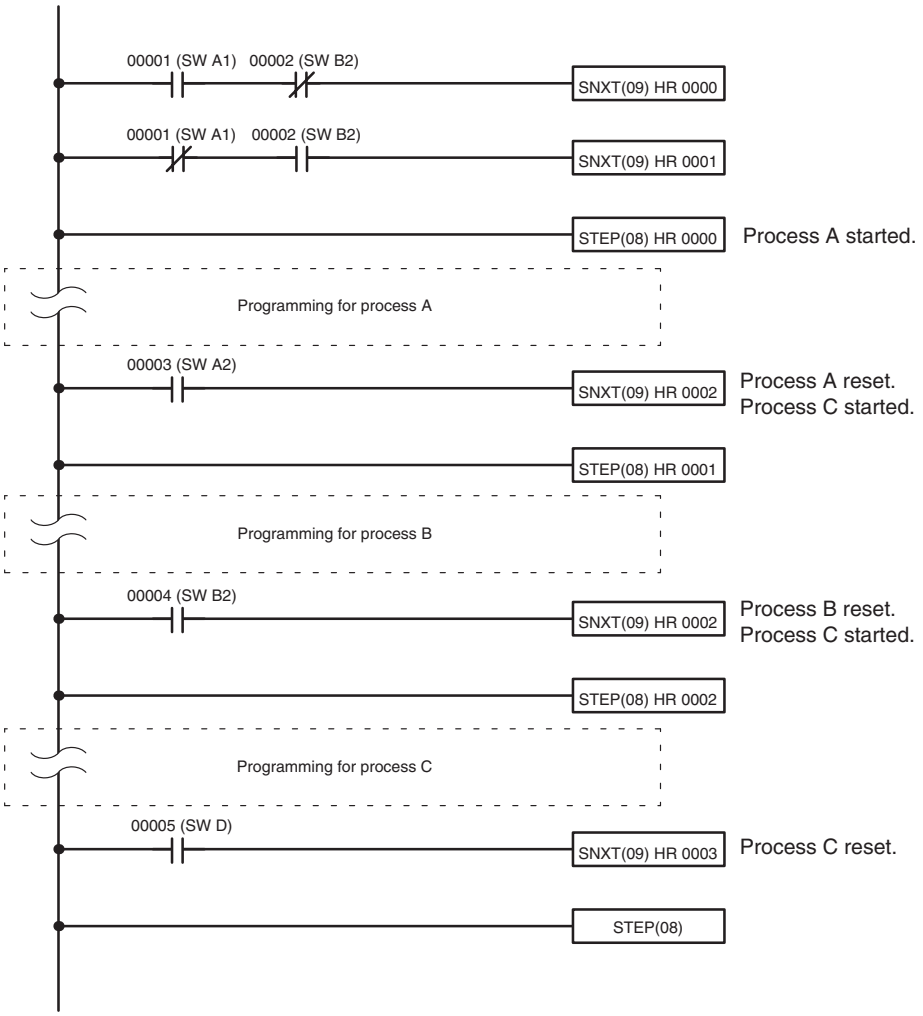


The following diagram demonstrates the flow of processing and the switches that are used for execution control. Here, either process A or process B is used depending on the status of SW A1 and SW B1.



The program for this process, shown below, starts with two SNXT(09) instructions that start processes A and B. Because of the way 00001 (SW A1) and 00002 (SB B1) are programmed, only one of these will be executed to

start either process A or process B. Both of the steps for these processes end with a SNXT(09) that starts the step for process C.



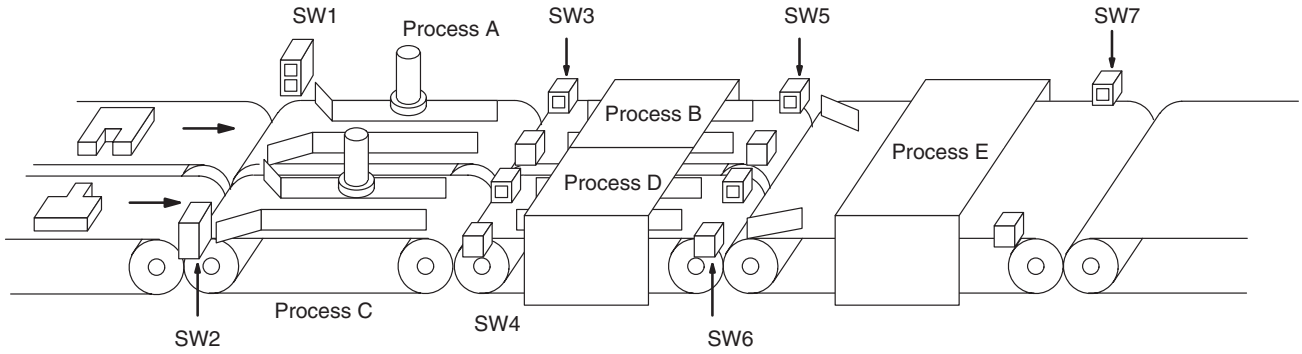
Address	Instruction	Operands	
00000	LD		00001
00001	AND NOT		00002
00002	SNXT(09)	HR	0000
00003	LD NOT		00001
00004	AND		00002
00005	SNXT(09)	HR	0001
00006	STEP(08)	HR	0000
	Process A		
00100	LD		00003
00101	SNXT(09)	HR	0002
00102	STEP(08)	HR	0001

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

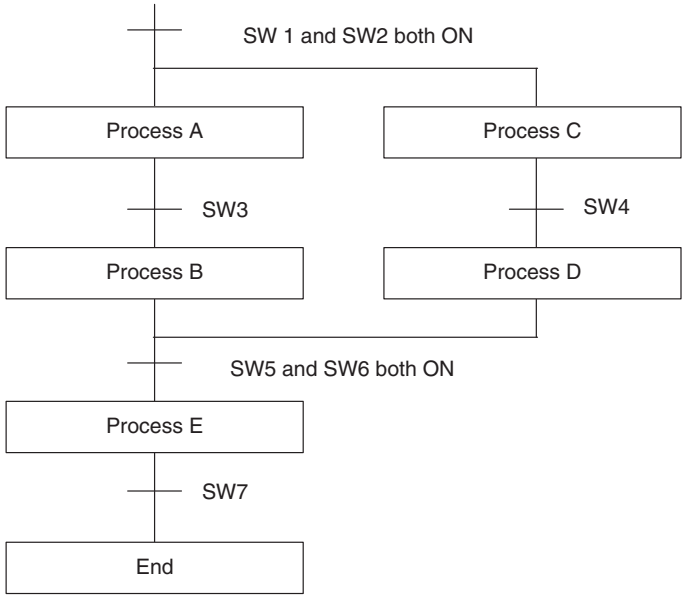
Note Using HR 0002 as the operand in two SNXT(09) instructions is not considered duplicate use of the bit and an error will not occur in the program check.

Example 3: Parallel Execution

The following process requires that two parts of a product pass simultaneously through two processes each before they are joined together in a fifth process. Various sensors are positioned to signal when processes are to start and end.



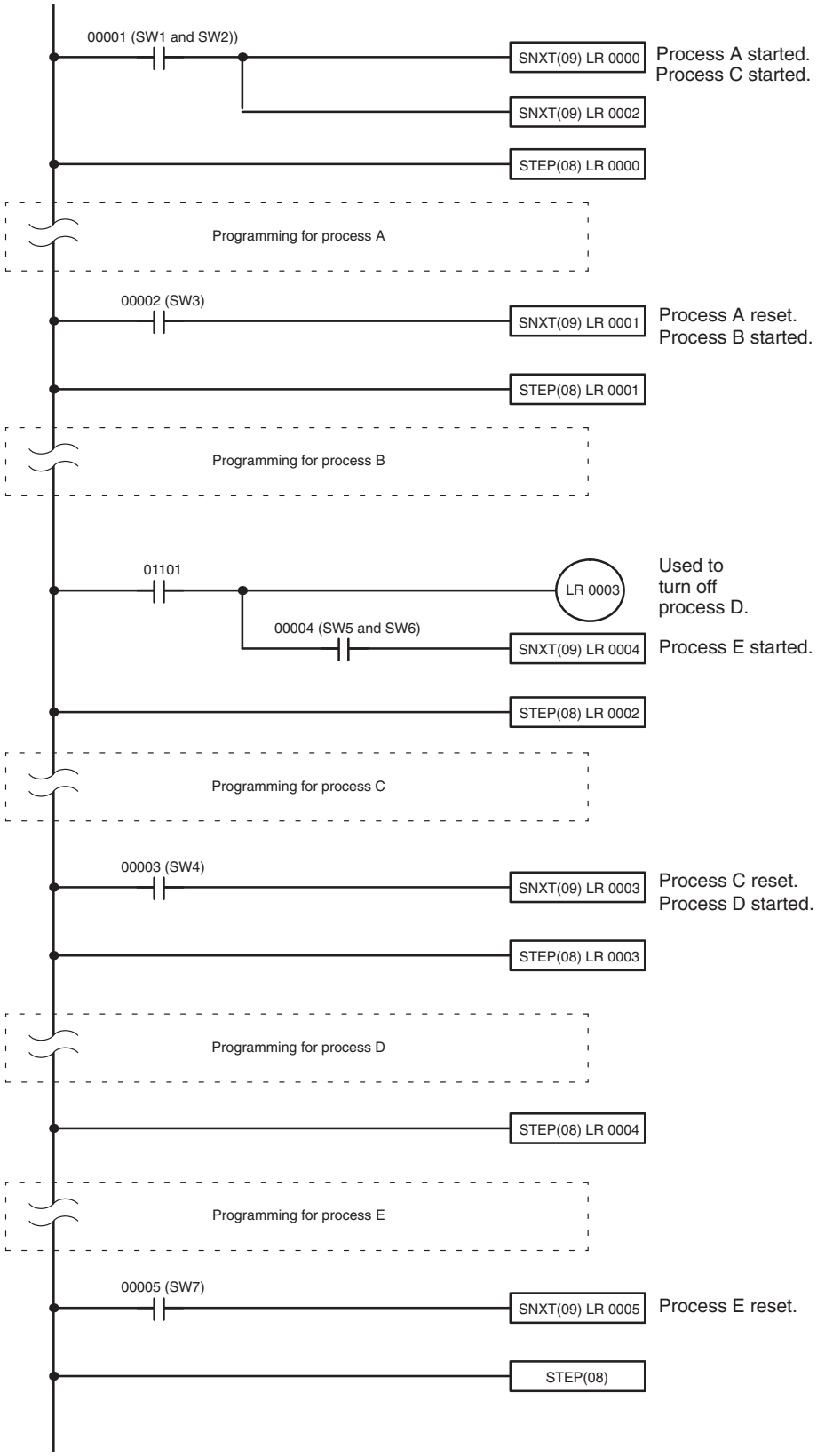
The following diagram demonstrates the flow of processing and the switches that are used for execution control. Here, process A and process C are started together. When process A finishes, process B starts; when process C finishes, process D starts. When both processes B and D have finished, process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



The program for this operation, shown below, starts with two SNXT(09) instructions that start processes A and C. These instructions branch from the same instruction line and are always executed together, starting steps for both A and C. When the steps for both A and C have finished, the steps for process B and D begin immediately.

When both process B and process D have finished (i.e., when SW5 and SW6 turn ON), processes B and D are reset together by the SNXT(09) at the end of the programming for process B. Although there is no SNXT(09) at the end of process D, the control bit for it is turned OFF by executing SNXT(09) LR 0004. This is because the OUT for LR 0003 is in the step reset by SNXT(09) LR 0004, i.e., LR 003 is turned OFF when SNXT(09) LR 0004 is executed

Process B is thus reset directly and process D is reset indirectly before executing the step for process E.



Address	Instruction	Operands	
00000	LD		00001
00001	SNXT(09)	LR	0000
00002	SNXT(09)	LR	0002
00003	STEP(08)	LR	0000
	Process A		
00100	LD		00002
00101	SNXT(09)	LR	0001
00102	STEP(08)	LR	0001
	Process B		
00100	LD		01101
00101	OUT	LR	0003
00101	AND		00004
00101	SNXT(09)	LR	0004

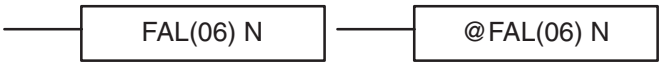
Address	Instruction	Operands	
00102	STEP(08)	LR	0002
	Process C		
00200	LD		00003
00201	SNXT(09)	LR	0003
00202	STEP(08)	LR	0003
	Process D		
00300	STEP(08)	LR	0004
	Process E		
00400	LD		00005
00401	SNXT(09)	LR	0005
00402	STEP(08)		---

5-23 Special Instructions

The instructions in this section are used for various operations, including programming user error codes and messages, counting ON bits, setting the watchdog timer, and refreshing I/O during program execution.

5-23-1 FAILURE ALARM – FAL(06) and SEVERE FAILURE ALARM – FALS(07)

Ladder Symbols



Definer Data Areas

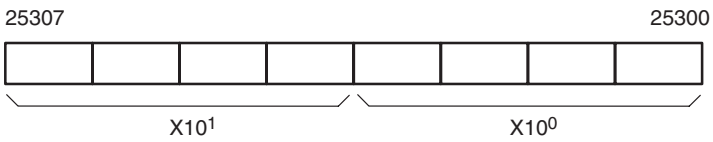
N: FAL number
(00 to 99)

N: FAL number
(01 to 99)

Description

FAL(06) and FALS(07) are provided so that the programmer can output error numbers for use in operation, maintenance, and debugging. When executed with an ON execution condition, either of these instruction will output a FAL number to bits 00 to 07 of SR 253. The FAL number that is output can be between 01 and 99 and is input as the definer for FAL(06) or FALS(07). FAL(06) with a definer of 00 is used to reset this area (see below).

FAL Area



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

The system also generates error codes to the FAL area.

Resetting Errors

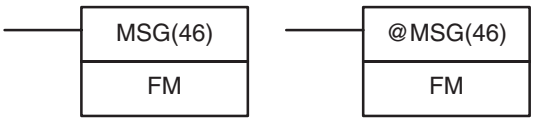
A maximum of three FAL error codes will be retained in memory, although only one of these is available in the FAL area. To access the other FAL codes, reset the FAL area by executing FAL(06) 00. Each time FAL(06) 00 is executed, another FAL error will be moved to the FAL area, clearing the one that is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

FAL(06) 00 is also used to clear message programmed with the next instruction MSG(46).

If the FAL area cannot be cleared, as is generally the case when FALS(07) is executed, first remove the cause of the error and then clear the FAL area through the Programming Console (see *4-5-4 Clearing Error Messages*).

5-23-2 DISPLAY MESSAGE – MSG(46)

Ladder Symbols



Operand Data Areas

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

Description

When executed with an ON execution condition, MSG(46) reads eight words of extended ASCII code from FM to FM+7 and displays the message on the Programming Console, GPC, or FIT. The displayed message can be up to 16 characters long, i.e., each ASCII character code requires eight bits (two digits). Refer to *Appendix F* for the extended ASCII codes. Japanese katakana characters are included in this code.

If not all eight words are required for the message, it can be stopped at any point by inputting "OD." When OD is encountered in a message, no more words will be read and the words that normally would be used for the message can be used for other purposes.

Message Buffering and Priority

Up to three messages can be buffered in memory. Once stored in the buffer, they are displayed on a first in, first out basis. Since it is possible that more than three MSG(46)s may be executed within a single cycle, there is a priority scheme, based on the area where the messages are stored, for the selection of those messages to be buffered.

The priority of the data areas is as follows for message display:

- LR > IR (I/O) > IR (not I/O) > HR > AR > DM
- In handling messages from the same area, those with the lowest address values have higher priority.
- In handling indirectly addressed messages (i.e. *DM), those with the lowest DM address values have higher priority.

Clearing Messages

To clear a message, execute FAL(06) 00 or clear it via a Programming Console using the procedure in *4-5-4 Clearing Error Messages*.

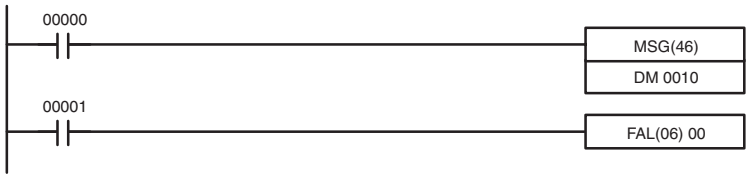
If the message data changes while the message is being displayed, the display will also change.

Flags

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

Example

The following example shows the display that would be produced for the instruction and data given when 00000 was ON. If 00001 goes ON, a message will be cleared.

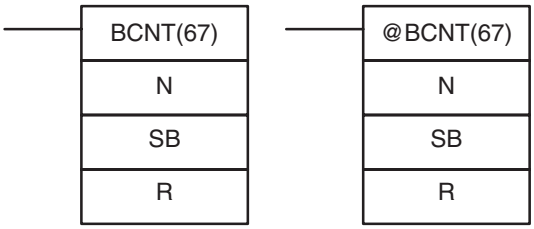


Address	Instruction	Operands
00000	LD	00000
00001	MSG(46)	
		DM 0010
00002	LD	00001
00003	FAL(06)	00

DM contents					ASCII equivalent	
DM 0010	4	1	4	2	A	B
DM 0011	4	3	4	4	C	D
DM 0012	4	5	4	6	E	F
DM 0013	4	7	4	8	G	H
DM 0014	4	9	4	A	I	J
DM 0015	4	B	4	C	K	L
DM 0016	4	D	4	E	M	N
DM 0017	4	F	5	0	O	P

5-23-3 BIT COUNTER – BCNT(67)

Ladder Symbols



Operand Data Areas

N: Number of words (BCD)
IR, AR, DM, HR, TC, LR, #

SB: Source beginning word
IR, SR, AR, DM, HR, TC, LR

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

Limitations

N cannot be 0.

Description

When the execution condition is OFF, BCNT(67) is not executed. When the execution condition is ON, FUN(67) counts the total number of bits that are ON in all words between SB and SB+(N–1) and places the result in D.

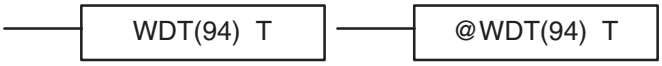
Flags

ER: N is not BCD, or N is 0; SB and SB+(N–1) are not in the same area.
The resulting count value exceeds 9999.
Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

EQ: ON when the result is 0.

5-23-4 WATCHDOG TIMER REFRESH– WDT(94)

Ladder Symbols



Definer Data Areas

T: Watchdog timer value
(00 to 63)

Description

When the execution condition is OFF, WDT(94) is not executed. When the execution condition is ON, WDT(94) extends the setting of the watchdog timer (normally set by the system to 130 ms) by 100 ms times T.

Timer extension = 100 ms x T.

Precautions

If the cycle time is longer than the time set for the watchdog timer, 9F will be output to the FAL area and the CPU will stop.

If the cycle time exceeds 6,500 ms, a FALS 9F will be generated and the system will stop.

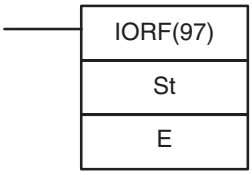
Timers might not function properly when the cycle time exceeds 100 ms. When using WDT(94), the same timer should be repeated in the program at intervals that are less than 100 ms apart. TIMH(15) should be used only in a scheduled interrupt routine executed at intervals of 10 ms or less.

Flags

ER: There are no flags affected by this instruction.

5-23-5 I/O REFRESH – IORF(97)

Ladder Symbol



Operand Data Areas

St: Starting word
IR (I/O word only)

E: End word
IR (I/O word only)

Limitations

IORF(97) can be used to refresh I/O words allocated to the CPU or an Expansion I/O Rack only. It cannot be used for other I/O words.

St must be less than or equal to E.

Description

When the execution condition is OFF, IORF(97) is not executed. When the execution condition is ON, all words between St and E will be refreshed. This will be in addition to the normal I/O refresh performed during the CPU's cycle.

Execution Time

The execution time for IORF(97), T_{IORF}, is computed as follows:

T_{IORF} = 1 ms + (130 μs x number of words refreshed)

Flags

There are no flags affected by this instruction.

5-24 Data Tracing (TRACE MEMORY SAMPLING – TRSM(45))

Data tracing can be used to facilitate debugging programs. To set up and use data tracing it is necessary to have a GPC or FIT; no data tracing is possible

from a Programming Console. Data tracing is described in detail in the GPC, FIT, and LSS *Operation Manuals*. This section shows the ladder symbol for TRSM(45) and gives an example program.

Address tracing also aids debugging and is possible from the Programming Console. It does not, however, require any direct programming considerations, including use of TRSM(45). Refer to *7-1-5 Address Tracing* for details.

Ladder Symbol

Description

TRSM(45) is used in the program to mark locations where specified data is to be stored in Trace Memory. Up to 12 bits and up to 3 words may be designated for tracing (refer to the GPC, FIT or LSS *Operation Manual*).

TRSM(45) is not controlled by an execution condition, but rather by two bits in the AR area: AR 1815 and AR 1814. AR 1815 is the Sampling Start bit. This bit is turned ON to start the sampling processes for tracing. The Sampling Start bit must not be turned ON from the program, i.e., it must be turned ON only from the peripheral device. AR 1814 is the Trace Start bit. When it is set, the specified data is recorded in Trace Memory. The Trace Start bit can be set either from the program or from the Programming Device. A positive or negative delay can also be set to alter the actual point from which tracing will begin.

Data can be recorded in any of three ways. TRSM(45) can be placed at one or more locations in the program to indicate where the specified data is to be traced. If TRSM(45) is not used, the specified data will be traced when END(01) is executed. The third method involves setting a timer interval from the peripheral devices so that the specified data will be tracing at a regular interval independent of the cycle time (refer to the GPC, FIT or LSS *Operation Manual*).

TRSM(45) can be incorporated anywhere in a program, any number of times. The data in the trace memory can then be monitored via a Programming Console, GPC, etc.

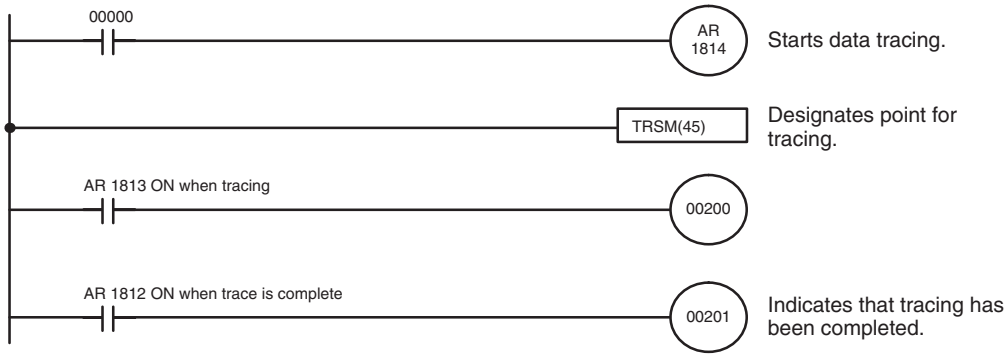
AR Control Bits and Flags

The following control bits and flags are used during data tracing. The Tracing Flag will be ON during tracing operations. The Trace Completed Flag will turn ON when enough data has been traced to fill Trace Memory.

Flag	Function
AR 1815	Sampling Start Bit
AR 1814	Trace Start Bit
AR 1813	Tracing Flag
AR 1812	Trace Completed Flag

Precautions

If TRSM(45) occurs TRSM(45) will not be executed within a JMP(08) – JME(09) block when the jump condition is OFF .

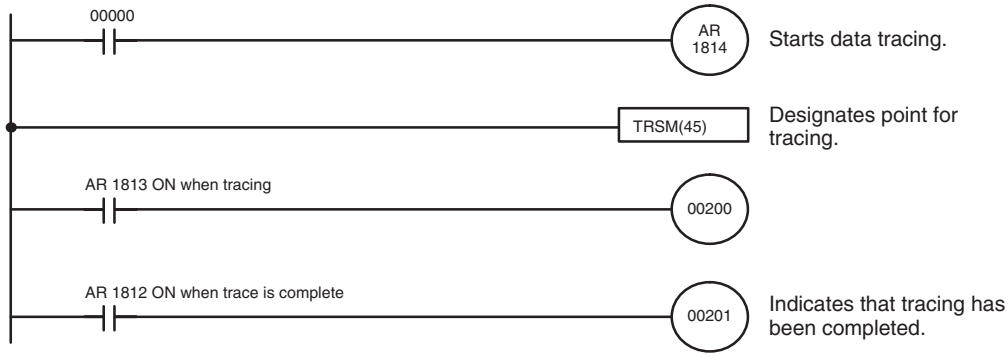


Address	Instruction	Operands
00000	LD	0000
00001	OUT	AR 1814
00002	TRSM(45)	
00003	LD	AR 1813

Address	Instruction	Operands
00004	OUT	00200
00005	LD	AR 1812
00006	OUT	00201

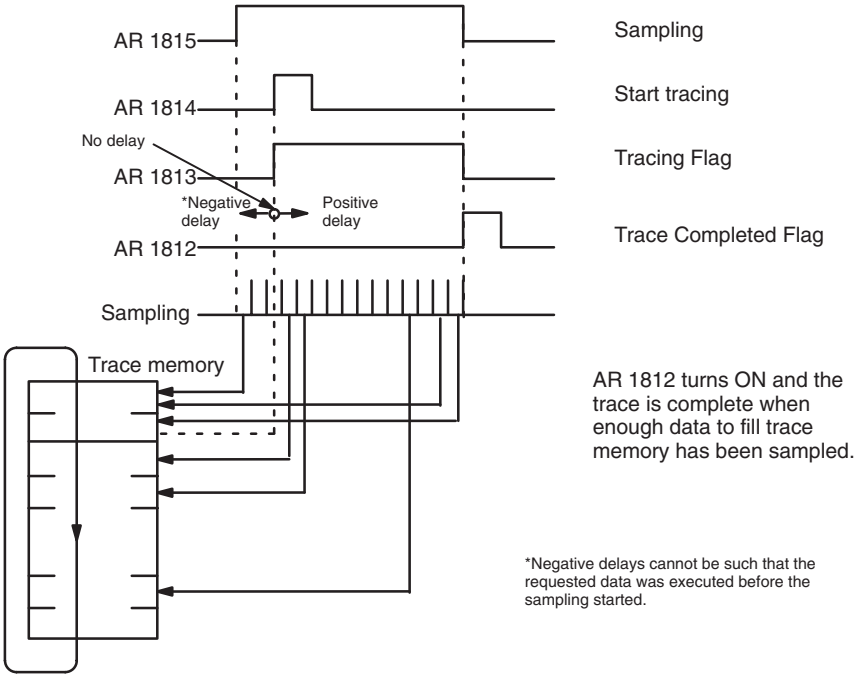
Example

The following shows the basic program and operation for data tracing. Refer to the GPC, FIT, or LSS *Operation Manual* for details. The Sampling Start Bit starts the sampling. The data is read and stored into trace memory. When the Trace Start Bit is received, the CPU looks at the delay and marks the trace memory accordingly. This can mean that some of the samples already made will be recorded as the trace memory (negative delay), or that more samples will be made before they are recorded (positive delay). The sampled data is written to trace memory, jumping to the beginning of the memory area once the end has been reached and continuing up to the start marker. This might mean that previously recorded data (i.e., data from this sample that falls before the start marker) is overwritten (this is especially true if the delay is positive). The negative delay cannot be such that the required data was executed before sampling was started.



Address	Instruction	Operands
00000	LD	0000
00001	OUT	AR 1814
00002	TRSM(45)	
00003	LD	AR 1813

Address	Instruction	Operands
00004	OUT	00200
00005	LD	AR 1812
00006	OUT	00201



5-25 File Memory Instructions

File memory instructions all involve the transfer of data to and from the File Memory area and the PC memory areas.The File Memory area is contained in a File Memory Unit. The instructions described in this section can thus only be used if there is a File Memory Unit mounted.

File memory transfers are done in blocks of 128 words. File Memory Units are available with a capacity of either 1000 or 2000 blocks. The blocks are numbered from zero.

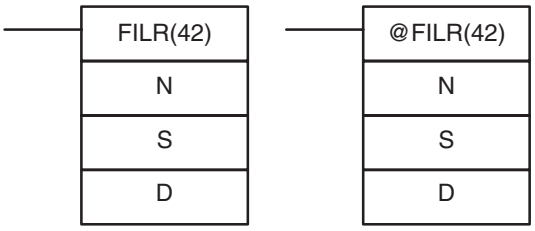
Exercise care when transferring a very large number of blocks or words since this can greatly increase the overall cycle time.

The following flags are used by all of the File Memory instructions. Refer to *3-4-3 File Memory Flags and Control Bits* for details.

AR 1900	Turned ON and OFF to reset AR 1903 through AR 1907.
AR 1901	Transfer in progress (ON)
AR 1902	Transfer direction (ON for transfer away from File Memory; OFF for transfer to File memory)
AR 1903 to1906	Error flags
AR 20	Total number of transferred blocks (4-digit BCD)
AR 21	Remaining number of blocks to be transferred (4-digit BCD)

5-25-1 FILE MEMORY READ – FILR(42)

Ladder Symbols



Operand Data Areas

N: Number of transfer blocks (BCD)
IR, AR, DM, HR, TC, LR, #
S: Source beginning block (BCD)
IR, AR, DM, HR, TC, LR, #
D: Destination beginning word
IR, AR, DM, HR, TC, LR

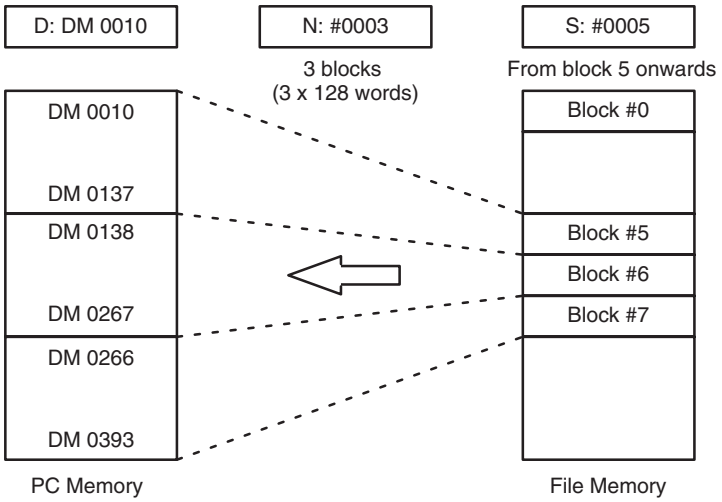
Limitations

S+(N–1) must be less than the largest block number provided by the File Memory Unit (0999 or 1999).

Description

When the execution condition is OFF, FILR(42) is not executed. When the execution condition is ON, FILR(42) reads blocks of data from the File Memory (128-word blocks), and outputs the data to the designated PC memory area beginning at D.

If the destination memory area is too small to accommodate all of the transfer data, only the portion that fits will be transferred.



Precautions

If a failure occurs, the transfer of the current word will be completed before operation is stopped.

Flags

ER: File Memory Unit is not mounted.

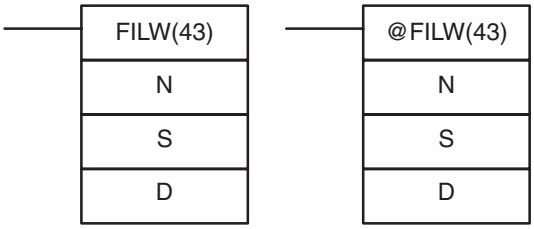
N or S is not BCD.

S+(N–1) is greater than 999 or 1999.

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

5-25-2 FILE MEMORY WRITE – FILW(43)

Ladder Symbols



Operand Data Areas

N: Number of transfer blocks (BCD)
IR, SR, AR, DM, HR, TC, LR, #
S: Source beginning word
IR, SR, AR, DM, HR, TC, LR
D: First destination block (BCD)
IR, AR, DM, HR, TC, LR, #

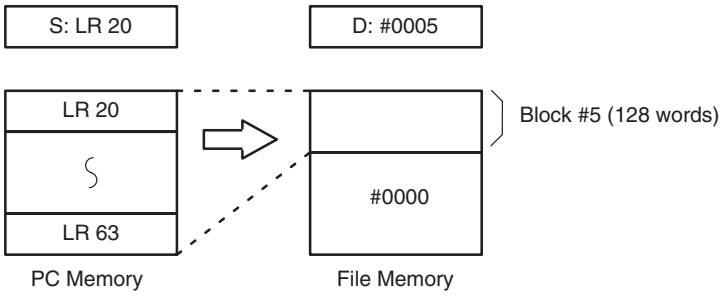
Limitations

D+(N–1) must be less than the largest block number provided by the File Memory Unit (0999 or 1999).

Description

When the execution condition is OFF, FILW(43) is not executed. When the execution condition is ON, FILW(43) transfers the designated number of blocks (N) from the designated PC memory area starting at S, to the File Memory beginning at D. The data is transferred in 128-word blocks.

If the last block of the source transfer area does not have a full 128 words, the unused words of the File Memory block will be empty.



Precautions

If a power failure occurs, transfer of the current block will be completed before operation is stopped.

Flags

ER: File Memory Unit is not mounted.

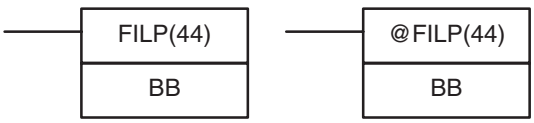
N or D is not BCD.

D+(N–1) is greater than 999 or 1999.

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

5-25-3 EXTERNAL PROGRAM READ – FILP(44)

Ladder Symbols



Operand Data Areas

BB: Beginning data block
IR, AR, DM, HR, TC, LR, #

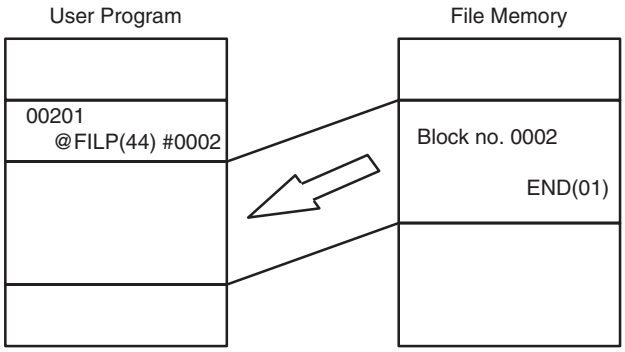
Limitations

BB must be less than the largest block number provided by the File Memory Unit (0999 or 1999). The Memory Unit must be RAM, and the blocks being transferred from the File Memory must contain program data.

Description

When the execution condition is OFF, FILP(44) is not executed. When the execution condition is ON, FILP(44) reads the data stored in the File Memory beginning at BB, transfers it to the program memory location immediately following FILP(44), and then executes the transferred program. The transferred program data replaces existing program data in the program memory area. The transfer is completed when the first END(01) is encountered or when the next non-program block is encountered. If END(01) is encountered within the beginning block, only that block is transferred.

Because the transfer is done in block units, any other data between the END(01) instruction and the block boundary is also transferred.



Precautions

Execution of loaded program data is inhibited when the FILP(44) execution condition is ON; use @FILP(44) when required to ensure that program execution can continue.

Exercise care when overwriting program data that contains an interrupt routine. Ensure that the program data to be loaded defines an interrupt routine with the same subroutine number.

Flags

ER: ROM memory unit is being used for user memory.

File memory unit is not mounted.

BB is not BCD.

BB is greater than 999 or 1999.

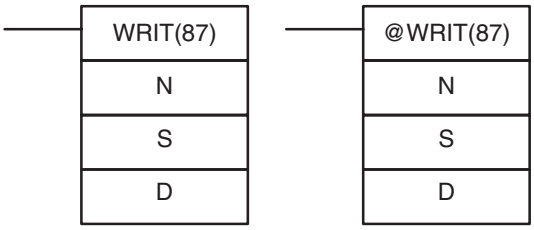
The blocks will not fit in User Memory.

5-26 Intelligent I/O Instructions

The intelligent I/O instructions are used for input/output operations with Special I/O Units such as an ASCII Unit.

5-26-1 I/O WRITE – WRIT(87)

Ladder Symbols



Operand Data Areas

N: Number of transfer words (BCD)
IR, AR, DM, HR, TC, LR, #
S: Source beginning word
IR, SR, AR, DM, HR, TC, LR
D: Destination word
IR (I/O word only)

Limitations

S and S+(N–1) must be in the same data area.

Description

When the execution condition is OFF, WRIT(87) is not executed. When the execution condition is ON, WRIT(87) transfers the contents of S through S+(N–1) to the Intelligent I/O Unit allocated D.

Precautions

If an Intelligent I/O Unit is busy and unable to receive data, the data will be written during the next cycle. To make sure that WRIT(87) execution has been completed, check EQ .

Flags

ER: N is not BCD.

D is not allocated to an Intelligent I/O Unit.

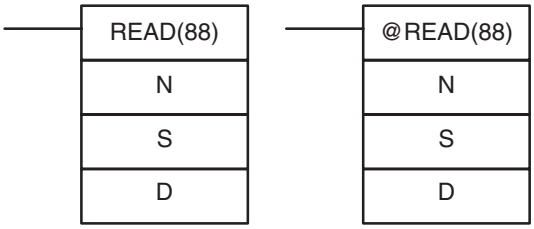
S+(N–1) exceeds the data area boundary.

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

EQ: OFF while data is being written; ON when writing has been completed.

5-26-2 I/O READ – READ(88)

Ladder Symbols



Operand Data Areas

N: Number of transfer words (BCD)
IR, AR, DM, HR, TC, LR, #
S: Source word
IR (I/O word only)
D: Destination word
IR, AR, DM, HR, TC, LR

Description

When the execution condition is OFF, READ(88) is not executed. When the execution condition is ON, READ(88) reads data from the memory area of the Intelligent I/O Unit allocated S and transfers it to D through D+(N–1).

Precautions

If the data cannot be sent or the Intelligent I/O Unit is busy, the data will be transferred during the next cycle. To make sure that READ(88) execution has been completed, check EQ.

Flags

ER: S is not allocated to an intelligent I/O Unit.
D+(N–1) exceeds a data area boundary.
N is not BCD.
Indirectly addressed DM word is non-existent. (Content of ∗DM word is not BCD, or the DM area boundary has been exceeded.)

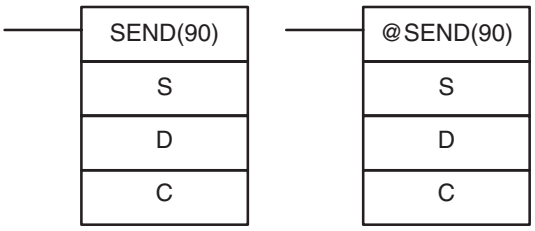
EQ: OFF while data is being read; ON when reading has been completed.

5-27 Network Instructions

The Network instructions are used for communicating with other PCs linked through the SYSMAC NET Link System or SYSMAC LINK System.

5-27-1 NETWORK SEND – SEND(90)

Ladder Symbols



Operand Data Areas

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

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

C: First control data word
IR, AR, DM, HR, TC, LR

Limitations

C through C+2 must be within the same data area and must be within the values specified below. To be able to use SEND(90), the system must have a SYSMAC NET Link or SYSMAC LINK Unit mounted.

Description

When the execution condition is OFF, SEND(90) is not executed. When the execution condition is ON, SEND(90) transfers data beginning at word S, to addresses specified by D in the designated node on the SYSMAC NET Link/ SYSMAC LINK System. The control words, beginning with C, specify the number of words to be sent, the destination node, and other parameters. The contents of the control data depends on whether a transmission is being sent in a SYSMAC NET Link System or a SYSMAC LINK System.

The status of bit 15 of C+1 determines whether the instruction is for a SYSMAC NET Link System or a SYSMAC LINK System.

Control Data

SYSMAC NET Link Systems

The destination port number is always set to 0. Set the destination node number to 0 to send the data to all nodes. Set the network number to 0 to send data to a node on the same Subsystem (i.e., network). Refer to the *SYSMAC NET Link System Manual* for details.

Word	Bits 00 to 07	Bits 08 to 15
C	Number of words (0 to 1000 in 4-digit hexadecimal, i.e., 0000_{hex} to $03E8_{hex}$)	
C+1	Network number (0 to 127 in 2-digit hexadecimal, i.e., 00_{hex} to $7F_{hex}$)	Bit 14 ON: Operating level 0 OFF: Operating level 1 Bits 08 to 13 and 15: Set to 0.
C+2	Destination node (0 to 126 in 2-digit hexadecimal, i.e., 00_{hex} to $7E_{hex}$)*	Destination port NSB: 00 NSU: 01/02

*The node number of the PC executing the send may be set.

SYSMAC LINK Systems

Set the destination node number to 0 to send the data to all nodes. Refer to the *SYSMAC LINK System Manual* for details.

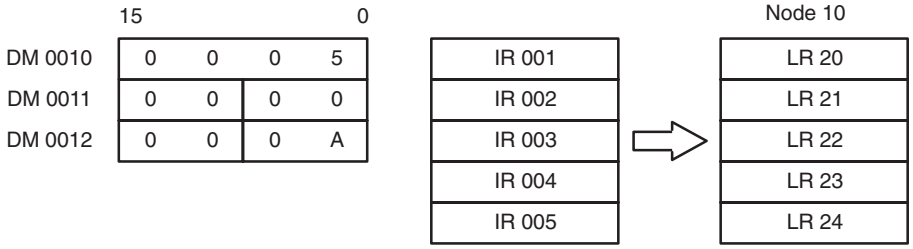
Word	Bits 00 to 07	Bits 08 to 15
C	Number of words (0 to 256 in 4-digit hexadecimal, i.e., 0000_{hex} to 0100_{hex})	
C+1	Response time limit (0.1 and 25.4 seconds in 2-digit hexadecimal without decimal point, i.e., 00_{hex} to FF_{hex}) **Note:** The response time will be 2 seconds if the limit is set to 0_{hex}. There will be no time limit if the time limit is set to FF_{hex}.	Bits 08 to 11: No. of retries (0 to 15 in hexadecimal, i.e., 0_{hex} to F_{hex}) Bit 12: Set to 0. Bit 13 ON: Response not returned. OFF: Response returned. Bit 14 ON: Operating level 0 OFF: Operating level 1 Bit 15: Set to 1.
C+2	Destination node (0 to 62 in 2-digit hexadecimal, i.e., 00_{hex} to $3E_{hex}$)*	Set to 0.

*The node number of the PC executing the send cannot be set.

Examples

This example is for a SYSMAC NET Link System. When 00000 is ON, the following program transfers the content of IR 001 through IR 005 to LR 20 through LR 24 on node 10.

Address	Instruction	Operands
00000	LD	00000
00001	SEND(90)	
		001
		LR 20
		DM 0010



Flags

ER: The specified node number is greater than 126 in a SYSMAC NET Link System or greater than 62 in a SYSMAC LINK System.

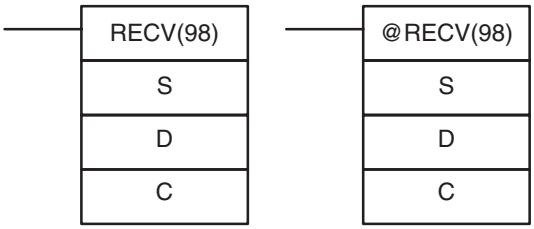
The sent data overruns the data area boundaries.

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

There is no SYSMAC NET Link/SYSMAC LINK Unit.

5-27-2 NETWORK RECEIVE – RECV(98)

Ladder Symbols



Operand Data Areas

S: Source beginning word
IR, SR, AR, DM, HR, TC, LR
D: Destination beginning word
IR, AR, DM, HR, TC, LR
C: First control data word
IR, AR, DM, HR, TC, LR

Limitations

C through C+2 must be within the same data area and must be within the values specified below. To be able to use RECV(98), the system must have a SYSMAC NET Link or SYSMAC LINK Unit mounted.

Description

When the execution condition is OFF, RECV(98) is not executed. When the execution condition is ON, RECV(98) transfers data beginning at S from a node on the SYSMAC NET Link/SYSMAC LINK System to words beginning at D. The control words, beginning with C, provide the number of words to be received, the source node, and other transfer parameters.

The status of bit 15 of C+1 determines whether the instruction is for a SYSMAC NET Link System or a SYSMAC LINK System.

Control Data

SYSMAC NET Link Systems

The source port number is always set to 0. Set the network number to 0 to receive data to a node on the same Subsystem (i.e., network). Refer to the *SYSMAC NET Link System Manual* for details.

Word	Bits 00 to 07	Bits 08 to 15
C	Number of words (0 to 1000 in 4-digit hexadecimal, i.e., 0000_{hex} to $03E8_{hex}$)	
C+1	Network number (0 to 127 in 2-digit hexadecimal, i.e., 00_{hex} to $7F_{hex}$)	Bit 14 ON: Operating level 0 OFF: Operating level 1 Bits 08 to 13 and 15: Set to 0.
C+2	Source node (1 to 126 in 2-digit hexadecimal, i.e., 01_{hex} to $7E_{hex}$)	Source port NSB: 00 NSU: 01/02

SYSMAC LINK Systems

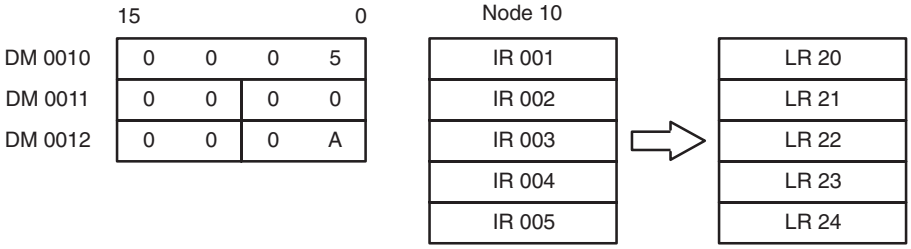
Refer to the *SYSMAC LINK System Manual* for details.

Word	Bits 00 to 07	Bits 08 to 15
C	Number of words (0 to 256 in 4-digit hexadecimal, i.e., 0000_{hex} to 0100_{hex})	
C+1	Response time limit (0.1 and 25.4 seconds in 2-digit hexadecimal without decimal point, i.e., 00_{hex} to FF_{hex}) **Note:** The response time will be 2 seconds if the limit is set to 0_{hex}. There will be no time limit if the time limit is set to FF_{hex}.	Bits 08 to 11: No. of retries (0 to 15 in hexadecimal, i.e., 0_{hex} to F_{hex}) Bit 12: Set to 0. Bit 13: Set to 0. Bit 14 ON: Operating level 0 OFF: Operating level 1 Bit 15: Set to 1.
C+2	Source node (0 to 62 in 2-digit hexadecimal, i.e., 00_{hex} to $3E_{hex}$)	Set to 0.

Examples

This example is for a SYSMAC NET Link System. When 00000 is ON, the following program transfers the content of IR 001 through IR 005 to LR 20 through LR 24 on node 10.

Address	Instruction	Operands
00000	LD	00000
00001	RECV(98)	
		001
		LR 20
		DM 0010



Flags

ER: The specified node number is greater than 126 in a SYSMAC NET Link System or greater than 62 in a SYSMAC LINK System.

The received data overflows the data area boundaries.

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

There is no SYSMAC NET Link/SYSMAC LINK Unit.

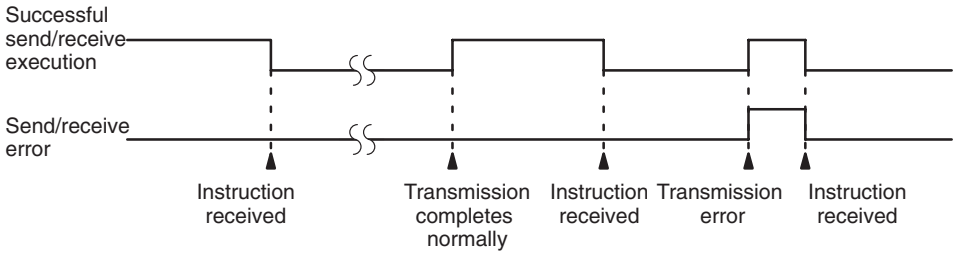
5-27-3 About SYSMAC NET Link/SYSMAC LINK Operations

SEND(90) and RECV(98) are based on command/response processing. That is, the transmission is not complete until the sending node receives and acknowledges a response from the destination node. Note that the SEND(90)/RECV(98) Enable Flag is not turned ON until the first END(01) after the transmission is completed. Refer to the *SYSMAC NET Link System Manual* or *SYSMAC LINK System Manual* for details about command/response operations.

If multiple SEND(90)/RECV(98) operations are used, the following flags must be used to ensure that any previous operation has completed before attempting further send/receive SEND(90)/RECV(98) operations

SR Flag	Functions
SEND(90)/RECV(98) Enable Flag (SR 25204)	OFF during SEND(90)/RECV(98) execution (including command response processing). Do not start a SEND(90)/RECV(98) operation unless this flag is ON.
SEND(90)/RECV(98) Error Flag (SR 25203)	OFF following normal completion of SEND/RECV (i.e., after reception of response signal) ON after an unsuccessful SEND(90)/RECV(98) attempt. Error status is maintained until the next SEND(90)/RECV(98) operation. Error types: Time-out error (command/response time greater than 1 second) Transmission data errors

Timing

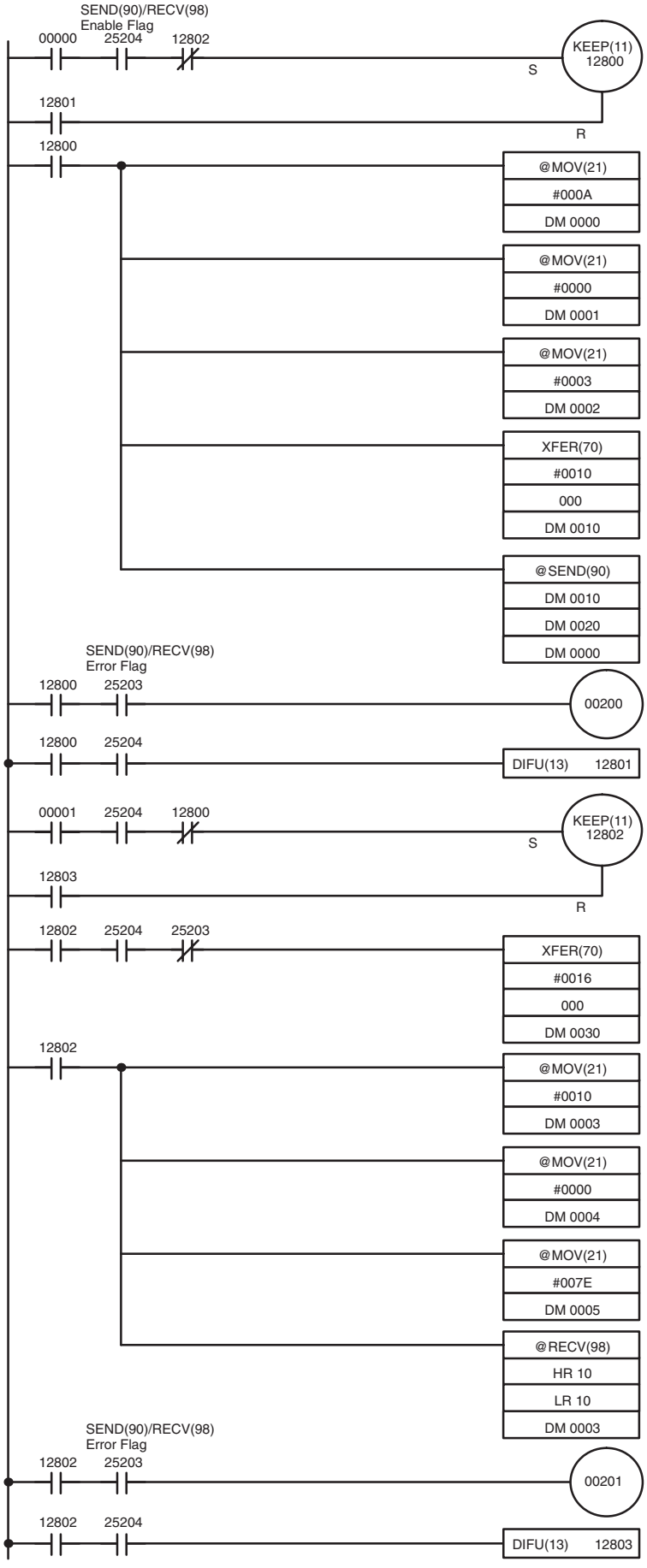


Data Processing for SEND(90)/RECV(98)

Data is transmitted for SEND(90) and RECV(98) for all PCs when SEND(90)/RECV(98) is executed. Final processing for transmissions/receptions is performed when END(01) is executed in C2000H Duplex CPUs, and during servicing of peripheral devices and Link Units for all other CPUs.

Programming Example: Multiple SEND(90)/RECV(98)

To ensure successful SEND(90)/RECV(98) operations, your program must use the SEND(90)/RECV(98) Enable Flag and SEND(90)/RECV(98) Error Flags to confirm that execution is possible. The following program shows one example of how to do this for a SYSMAC NET Link System.



12800 prevents execution of SEND(90) until RECV(98) (below) has completed. IR 00000 is turned ON to start transmission.

Data is placed into control data words to specify the 10 words to be transmitted to node 3 in operating level 1 of network 00 (NSB).

Turns ON to indicate transmission error.

Resets 12800, above.

12802 prevents execution of RECV(98) when SEND(90) above has not completed. IR 00001 is turned ON to start transmission.

Transmitted data moved into words beginning at DM 0030 for storage.

Data moved into control data words to specify the 16 words to be transmitted from node 126 in operating level 1 of network 00 (NSB).

Turns ON to indicate reception error.

Resets 12802, above.

Address	Instruction	Operands	
00000	LD		00000
00001	AND		25204
00002	AND NOT		12802
00003	LD		12801
00004	KEEP(11)		12800
00005	LD		12800
00006	@MOV(21)		
		#	000A
		DM	0000
00007	@MOV(21)		
		#	0000
		DM	0001
00008	@MOV(21)		
		#	0003
		DM	00002
00009	@XFER(70)		
		#	0010
			000
		DM	0002
00010	@SEND(90)		
		DM	0010
		DM	0020
		DM	0000
00011	LD		12800
00012	AND		25203
00013	OUT		00200
00014	LD		12800
00015	AND		25204
00016	DIFU(13)		12801
00017	LD		00001
00018	AND		25204

Address	Instruction	Operands	
00019	AND NOT		12800
00020	LD		12803
00021	KEEP(11)		12802
00022	LD		12802
00023	AND		25204
00024	AND NOT		25203
00025	XFER(70)		
		#	0016
			000
		DM	0030
00026	LD		12802
00027	@MOV(21)		
		#	0010
		DM	0003
00028	@MOV(21)		
		#	0000
		DM	0004
00029	@MOV(21)		
		#	007E
		DM	0005
00030	@RECV(98)		
		HR	10
		LR	10
		DM	0003
00031	LD		12802
00032	AND		25203
00033	OUT		00201
00034	LD		12802
00035	AND		25204
00036	DIFU(13)		12803

SECTION 6
Program Execution Timing

The timing of various operations must be considered both when writing and debugging a program. The time required to execute the program and perform other CPU operations is important, as is the timing of each signal coming into and leaving the PC in order to achieve the desired control action at the right time. This section explains the cycle and shows how to calculate the cycle time and I/O response times.

I/O response times in Link Systems are described in the individual System Manuals. These are listed at the end of *Section 1 Introduction*.

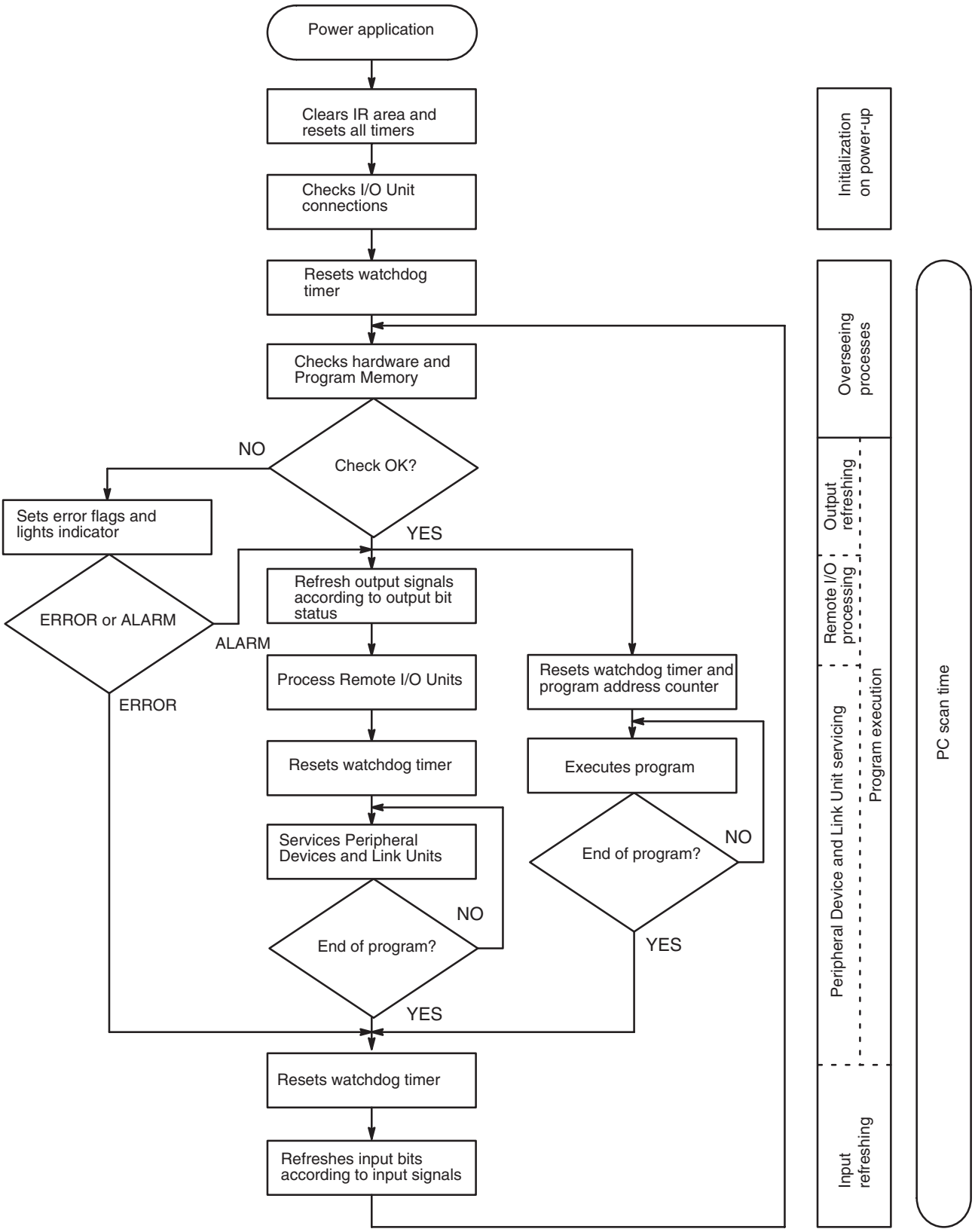
6-1 Cycle Time 228
6-2 Calculating Cycle Time 232
6-2-1 I/O Units Only 232
6-2-2 PC with Link Units 233
6-3 Instruction Execution Times 235
6-4 I/O Response Time 241

6-1 Cycle Time

To aid in PC operation, the average, maximum, and minimum cycle times can be displayed on the Programming Console or any other Programming Device and the maximum cycle time and current cycle time values are held in AR 26 and AR 27. Understanding the operations that occur during the cycle and the elements that affect cycle time is, however, essential to effective programming and PC operations.

The major factors in determining program timing are the cycle time and the I/O response time. One 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 four types of operation.

1. Overseeing
2. Output refreshing and Unit servicing
3. Input refreshing
4. Program execution

The cycle time is the total time required for the PC to perform all of the above operations. Of these, number two, output refreshing and Unit servicing, and number four, program execution, are executed in parallel. Because Unit servicing is repeated until program execution is completed (see below), output refreshing and Unit servicing time will be the same or longer than the program execution time. The cycle time will thus be the total of the time required for overseeing operations, the time required for output refreshing and Unit servicing, and the time required for input refreshing.

Cycle time = overseeing time + output refreshing and Unit servicing time + input refreshing time

The second of the above four operations is composed of up to six separate components. The breakdown of this operation and the function and time required for each operation are shown in the following table.

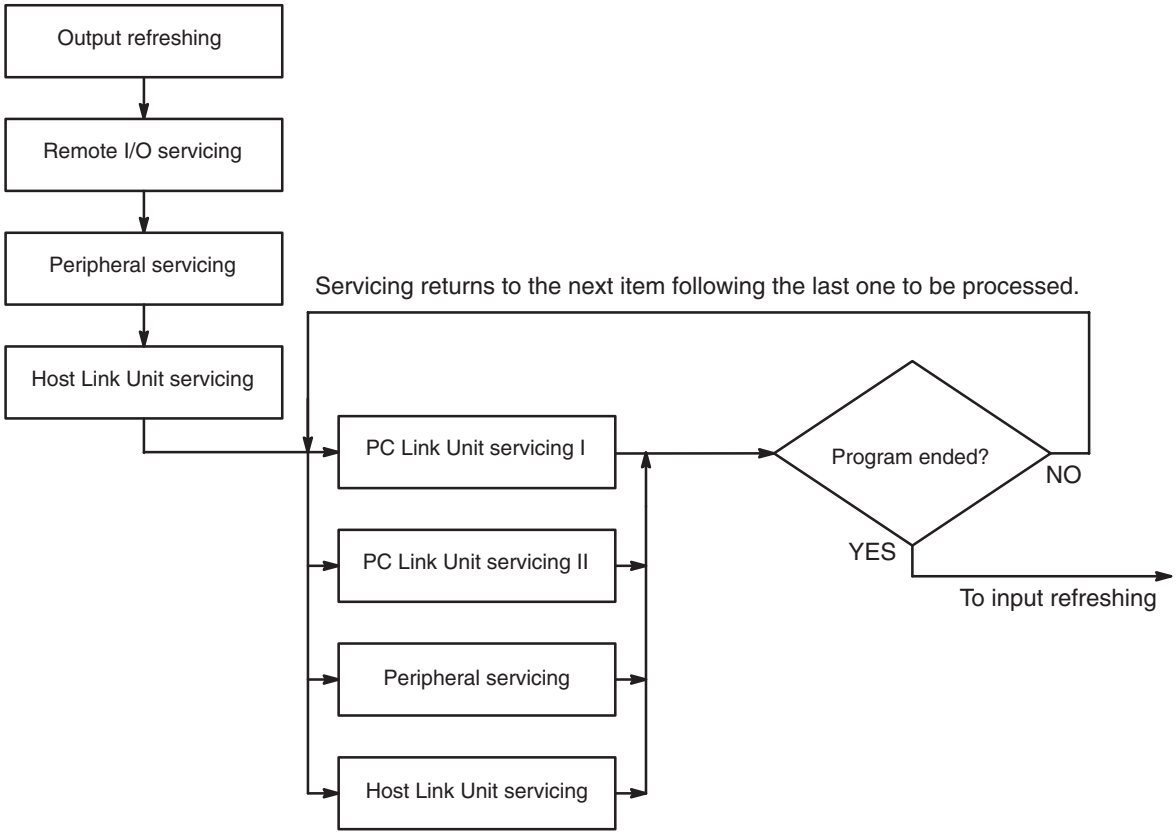
Overseeing	Approx. 3.0 ms + sync time (1.0 ms)	Watchdog timer set and cycle time checked. CPUs synchronized for C2000H Duplex System.
Output refreshing and Unit servicing		
I/O bus check	0.8 ms	I/O connections and bus operation checked.
Output refreshing	18 μs per output word	Output signals sent according to status of output bits in memory.
Remote I/O servicing	1 ms per Master plus 20 μs per word controlled though each Master	Inputs and Outputs in Remote I/O Systems refreshed, i.e., output signals sent according to output bit status and input bits set according to input signal status.
Peripheral device servicing	1.5 ms per service if Unit is mounted; 0.5 ms if Unit is not mounted.	Commands from Programming Devices and Interface Units processed.
Host Link Unit servicing	1.5 ms per service if Unit is mounted; 0.5 ms if Unit is not mounted.	Commands from computers connected through Host Link Units processed.
PC Link Unit servicing I and II	1.5 ms per service if Unit is mounted; 0.5 ms if Unit is not mounted.	Data communications for PC Link System processed.
Program execution	Total execution time for all instructions varies with program size, the instructions used, and execution conditions. Refer to *6-3 Instruction Execution Times* for details.	Program executed.
Input refreshing	25 μs per input word	Input bits set according to status of input signals.

All Link Units and peripheral devices are serviced once each cycle in the order given in the above table. If more time is required for program execution

than is required for output refreshing and Unit servicing, Link Units (except for Remote I/O Units) and all peripheral devices will be serviced again, one type at a time, until program execution is completed.

From the second cycle on, servicing will be in the following cycle (with servicing for the remaining time in any one cycle starting where servicing was left off the previous cycle): PC Link Unit servicing I, PC Link Unit servicing II, peripheral devices, and then Host Link Unit servicing. Program execution completion will be checked between the servicing of each of these, and servicing will be ended as soon as the program has been executed.

The following flowchart illustrates this portion of CPU operation.



Because Link Unit and peripheral device servicing is repeated until program execution is completed, the time required for output refreshing and Unit servicing will be the program execution time plus the time required to finish the Link Unit or Peripheral Device servicing operation that is in process when program execution has been completed. This is explained in more detail in *6-2 Calculating Cycle Time*.

Watchdog Timer and Long Cycle Times

Within the PC, the watchdog timer measures the cycle time and compares it to a set value. If the cycle time exceeds the set value of the watchdog timer, a FALS 9F error is generated and the CPU stops. WDT(94) can be used to extend the set value for the watchdog timer.

Even if the cycle time does not exceed the set value of the watchdog timer, a long cycle time can adversely affect the accuracy of system operations as shown in the following table.

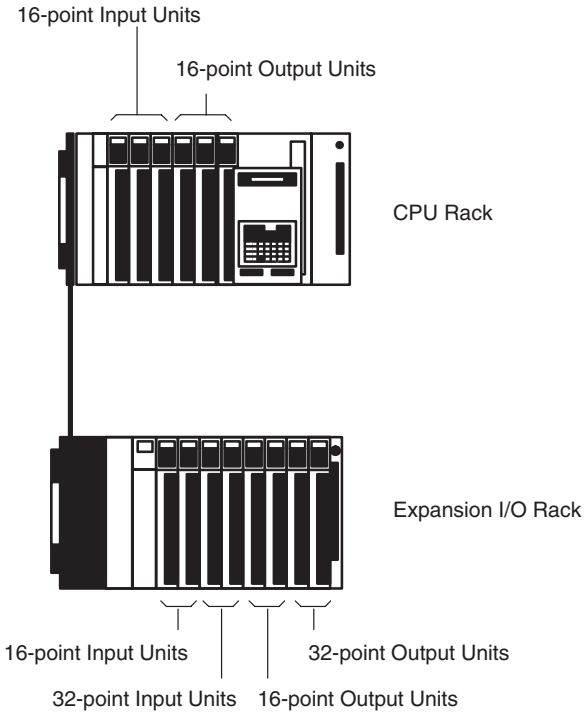
Cycle time (ms)	Possible adverse affects
10 or greater	TIMH(15) inaccurate when TC 048 through TC 511 are used.
20 or greater	0.02-second clock pulse not accurately readable.
100 or greater	0.1-second clock pulse not accurately readable and Cycle Timer Error Flag (25309) turns ON.
200 or greater	0.2-second clock pulse not accurately readable.
6,500 or greater	FALS code 9F generated regardless of watchdog timer setting and the system stops.

6-2 Calculating Cycle Time

The PC configuration, the program, and program execution conditions must be taken into consideration when calculating the cycle time. This means taking into account such things as the number of I/O points, the programming instructions used, and whether or not peripheral devices are employed. This section shows some basic cycle time calculation examples. To simplify the examples, the instructions used in the programs have been assumed to be all either LD or OUT. The average execution time for the instructions is thus 0.6 μs. (Operating times are given in the table in *Section 6-1*.)

6-2-1 I/O Units Only

Here, we'll compute the cycle time for a PC that controls only I/O Units, six on the CPU Rack and eight on an Expansion I/O Rack. In this PC configuration, there is also a Programming Console mounted to the CPU that needs to be taken into consideration. The PC configuration for this would be as shown below. It is assumed that the program contains 20,000 instructions requiring an average of 0.6 μs each to execute.



Calculations

The equation for the cycle time from above is as follows:

Cycle time = overseeing time
+ output refreshing and Unit servicing time
+ input refreshing time

The overseeing time is fixed at 3.0 ms. The input refresh time would be as follows for the five 16-point Input Units and two 32-point Input Units controlled by the PC:

$$\frac{(16 \text{ points} \times 5) + (32 \text{ points} \times 2)}{16 \text{ points}} \times 25\ \mu s = 0.23 \text{ ms}$$

The output refreshing and Unit servicing time equals the I/O bus check time plus the output refresh time plus the peripheral device and Link Unit servicing time. The I/O bus check time is fixed at 0.8 ms. The output refresh time would be as follows for the five 16-point Output Units and two 32-point Output Units controlled by the PC:

$$\frac{(16 \text{ points} \times 5) + (32 \text{ points} \times 2)}{16 \text{ points}} \times 18\ \mu s = 0.16 \text{ ms}$$

The basic peripheral device and Link Unit servicing time would be 1.5 ms for the Programming Console and 0.5 ms each for a Host Link Unit and two PC Link Units. This time, a total of 3 ms, is required even though not all of these Units are mounted to the PC. The total output refreshing time, including Unit servicing time computed above, is thus 0.8 ms + 0.16 ms + 3 ms = 3.96 ms.

Because this is considerably less than the program execution time, which is 12 ms (0.6 μs/instruction times 20,000 instructions) it is necessary to compute the number of times that peripheral device and Link Unit servicing would be repeated to arrive at the actual output refreshing and Unit servicing time. Subtracting 3.96 ms from 12 ms gives us 8.04 ms remaining before program execution is completed. This would be enough time to complete two more cycles of Peripheral Device and Link Unit servicing and still leave 2.04 ms to service the Programming Console (1.5 ms) and two more Link Units (0.5 ms each). The total output refreshing and Unit servicing time would thus be as follows:

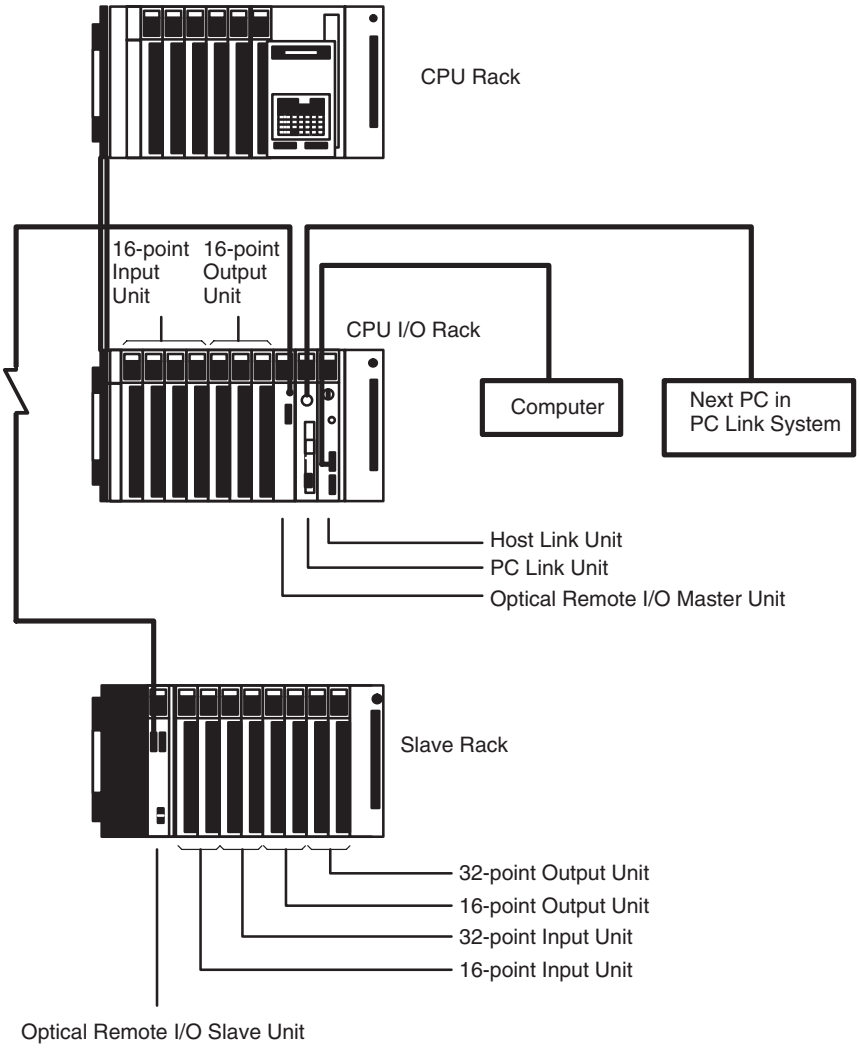
$$3.96 \text{ ms} + (2 \times 3 \text{ ms}) + 1.5 \text{ ms} + 0.5 \text{ ms} + 0.5 \text{ ms} = 12.46 \text{ ms}$$

The cycle time would thus be: 3.0 ms + 12.46 ms + 0.23 ms = 15.69 ms

6-2-2 PC with Link Units

Here, the cycle time is computed for a C2000H Duplex System with three 16-point Input Units, two 16-point Output Units, a Remote I/O Master Unit, a Host Link Unit, and a PC Link Unit on the CPU I/O Rack and two 16-point Input Units, two 32-point Input Units, two 16-point Output Units, and two 32-point Output Units on an Optical Slave Rack. In this PC configuration, there is also a Programming Console mounted to the CPU that needs to be taken into consideration. The PC configuration for this could be as shown

below. It is assumed that the program contains 20,000 instructions requiring an average of 0.6 μs each to execute.



Calculations

The equation for the cycle time is as follows:

Cycle time = overseeing time
+ output refreshing and Unit servicing time
+ input refreshing time

The overseeing time is fixed at 3.0 ms plus 1.0 ms (for synchronization of Duplex CPUs), or 4.0 ms. The input refresh time is 3 Units x 25 μs per 16-point Input Unit, or 0.08 ms

The output refreshing and Unit servicing time equals the I/O bus check time plus the output refresh time plus the Remote I/O servicing time plus the peripheral device and Link Unit servicing time. The I/O bus check time is fixed at 0.8 ms. The output refresh time would 2 Units x 18 μs per 16-point Output Unit, or 0.16 ms. The Remote I/O servicing time would be as as follows for the four 16-point I/O Units and four 32-point I/O Units controlled through the Master:

$$1\ \text{ms} + \frac{(16\ \text{points} \times 4) + (32\ \text{points} \times 4)}{16\ \text{points}} \times 20\ \mu\text{s} = 1.24\ \text{ms}$$

The basic peripheral device and Link Unit servicing time would be 1.5 ms x 4, or 6 ms, for the Programming Console, Host Link Unit, and two PC Link Units. The total output refreshing and Unit servicing time computed above is thus 0.8 ms + 0.04 ms + 1.24 ms + 6 ms = 8.08 ms.

Because this is less than the program execution time, which is 12 ms (0.6 μs/instruction times 20,000 instructions) it is necessary to compute the number of times that peripheral device and Link Unit servicing would be repeated to arrive at the actual output refreshing and Unit servicing time. Subtracting 8.08 ms from 12 ms gives us 3.92 ms remaining before program execution is completed. This would be enough time to repeat Programming Console servicing, the Host Link Unit servicing, and PC Link Unit servicing (1.5 ms each). The total output refreshing and Unit servicing time would thus be as follows:

8.08 ms + (3 x 1.5 ms) = 12.58 ms

The cycle time would thus be 4.0 ms + 12.58 ms + 0.08 ms = 16.66 ms

6-3 Instruction Execution Times

This following table lists the execution times for all instructions that are available for the C1000H and C2000H. The maximum and minimum execution times and the conditions which cause them are given where relevant. When "word" is referred to in the Conditions column, it implies the content of any word except for indirectly addressed DM words. Indirectly addressed DM words, which create longer execution times when used, are indicated by "*DM."

Execution times for most instructions depend on whether they are executed with an ON or an OFF execution condition. Exceptions are the ladder diagram instructions OUT and OUT NOT, which require the same time regardless of the execution condition. The OFF execution time for an instruction can also vary depending on the circumstances, i.e., whether it is in an interlocked program section and the execution condition for IL is OFF, whether it is between JMP(04) 00 and JME(05) 00 and the execution condition for JMP(04) 00 is OFF, or whether it is reset by an OFF execution condition. "R," "IL," and "JMP" are used to indicate these three times.

Table: Instruction Execution Times

Instruction	Conditions	ON execution time (μs)[1,2]		OFF execution time (μs)[1,2]	
		C1000H	C2000H	C1000H	C2000H
LD	---	0.4	0.4	0.4	0.4
LD NOT	---	0.4	0.4	0.4	0.4
AND	---	0.4	0.4	0.4	0.4
AND NOT	---	0.4	0.4	0.4	0.4
OR	---	0.4	0.4	0.4	0.4
OR NOT	---	0.4	0.4	0.4	0.4
AND LD	---	0.4	0.4	0.4	0.4
OR LD	---	0.4	0.4	0.4	0.4
OUT	---	0.8	0.8	0.8	0.8
OUT NOT	---	0.8	0.8	0.8	0.8

Notes
1. The execution time is given in microseconds unless otherwise stated.
2. Times for non-differentiated forms are given to the left of the slash, and those for differentiated forms given to the right.

Instruction	Conditions	ON execution time (μs)[1,2]		OFF execution time (μs)[1,2]	
		C1000H	C2000H	C1000H	C2000H
TIM	Constant for SV	2.4	2.4	R: 2.4 IL: 2.4 JMP: 2.4	
	∗DM for SV	20	13	R: 29 IL: 29 JMP: 14	R: 19 IL: 19 JMP: 10
CNT	Constant for SV	2.4	2.4	R: 2.4 IL: 2.4 JMP: 2.4	
	∗DM for SV	16	11	R: 28 IL: 11 JMP: 11	R: 18 IL: 7 JMP: 7
NOP(00)	---	0.4	0.4	---	---
END(01)	---	8	5	---	---
IL(02)	---	9	6	8	6
ILC(03)	---	9	6	7	5
JMP(04)	---	10	7	9	6
JME(05)	---	10	7	9	6
FAL(06)	---	16/17	11/12	7/8	5/6
FAL(06) 00	---	11/12	7/8	7/8	5/6
FALS(07)	---	11/12	7/8	7	5
STEP(08)	---	24	16	15	10
SNXT(09)	---	10	6	7	5
SFT(10)	With 1-word shift register	40	26	R: 35 IL: 7 JMP: 7	R: 25 IL: 5 JMP: 5
	With 252-word shift register	444	296	R: 20 0 IL: 7 JMP: 7	R: 13 3 IL: 5 JMP: 5
KEEP(11)	---	0.8	0.8	---	---
CNTR(12)	Constant for SV	21	14	R: 15 IL: 10 JMP: 10	R: 10 IL: 7 JMP: 7
	∗DM for SV	29	19		
DIFU(13)	---	16	10	Normal: 15 IL: 16 JMP: 8	Normal: 10 IL: 10 JMP: 5
DIFD(14)	---	16	11	Normal: 16 IL: 16 JMP: 9	Normal: 11 IL: 11 JMP: 6

Notes
1. The execution time is given in microseconds unless otherwise stated.
2. Times for non-differentiated forms are given to the left of the slash, and those for differentiated forms given to the right.

Instruction	Conditions	ON execution time (μs)[1,2]		OFF execution time (μs)[1,2]	
		C1000H	C2000H	C1000H	C2000H
TIMH(15)	Interrupt Constant for SV	20	13	R: 20	
	Normal cycle	22	15	IL: 21	
	Interrupt *DM for SV	20	13	JMP: 15	
	Normal cycle	18	12	R: 29 IL: 30 JMP: 16	R: 19 IL: 20 JMP: 10
WSFT(16)	When shifting 1 word	36/38	24/26	7/8	5/6
	When shifting 4,096 words using *DM	5.59 ms	3.72 ms		
CMP(20)	When comparing a constant to a word	14	9	7	5
	When comparing two *DM	29	20		
MOV(21)	When transferring a constant to a word	15/17	10/11	7/8	5/6
	When transferring *DM to *DM	30/31	20/21		
MVN(22)	When transferring a constant to a word	16/17	10/11	7/8	5/6
	When transferring *DM to *DM	30/31	20/21		
BIN (23)	When converting a word to a word	21/23	14/15	7/8	5/6
	When converting *DM to *DM	34/35	22/23		
BCD(24)	When converting a word to a word	21/22	14/15	7/8	5/6
	When converting *DM to *DM	33/34	22/23		
ASL(25)	When shifting a word	18/19	12/13	7/8	5/6
	When shifting *DM	24/25	16/17		
ASR(26)	When shifting a word	18/19	12/13	7/8	5/6
	When shifting *DM	24/25	16/17		
ROL(27)	When rotating a word	18/19	12/13	7/8	5/6
	When rotating *DM	24/25	16/17		
ROR(28)	When rotating a word	18/19	12/13	7/8	5/6
	When rotating *DM	24/25	16/17		
COM(29)	When inverting a word	15/17	10/11	7/8	5/6
	When inverting *DM	21/23	14/15		
ADD(30)	Constant + word → word	33/35	22/23	7/8	5/6
	*DM + *DM → *DM	53/55	35/36		
SUB(31)	Constant + word → word	33/34	22/23	7/8	5/6
	*DM – *DM → *DM	53/55	35/36		
MUL(32)	Constant x word → word	48/50	32/33	7/8	5/6
	*DM x *DM → word	68/70	55/56		
DIV(33)	Word ÷ constant → word	63/65	42/43	7/8	5/6
	*DM ÷ *DM → *DM	84/86	56/57		

Notes
1. The execution time is given in microseconds unless otherwise stated.
2. Times for non-differentiated forms are given to the left of the slash, and those for differentiated forms given to the right.

Instruction	Conditions	ON execution time (μs)[1,2]		OFF execution time (μs)[1,2]	
		C1000H	C2000H	C1000H	C2000H
ANDW(34)	Constant AND word → word	19/21	13/14	7/8	5/6
	*DM AND *DM → *DM	40/41	27/28		
ORW(35)	Constant OR word → word	19/20	13/14	7/8	5/6
	*DM OR *DM → *DM	40/41	27/28		
XORW(36)	Constant XOR word → word	19/21	13/14	7/8	5/6
	*DM XOR *DM → *DM	40/41	27/28		
XNRW(37)	Constant XNOR word → word	20/21	13/14	7/8	5/6
	*DM XNOR *DM → *DM	40/41	27/28		
INC(38)	When incrementing a word	23/25	15/16	7/8	5/6
	When incrementing *DM	29/31	20/21		
DEC(39)	When decrementing a word	22/24	15/16	7/8	5/6
	When decrementing *DM	28/30	19/20		
STC(40)	---	9/10	6/7	6/8	4/6
CLC(41)	---	9/10	6/7	6/8	4/6
FILR(42)	When reading 1 block	4.89 ms	3.26 ms	6/8	4/6
	When reading 20 blocks	81.9 ms	54.5 ms		
FILW(43)	When writing 1 block	7.21 ms	4.8 ms	6/8	4/6
	When writing 20 blocks	131 ms	87 ms		
FILP(44)	When reading 600 addresses	38 ms	25 ms	9	6
	When reading 30 addresses	1.66 s	1.11 s		
TRSM(45)	When tracing 1 point + 1 word	56	38	8	5
	When tracing 12 points + 3 words	93	62		
MSG(46)	---	16/17	11/12	7/8	5/6
ADB(50)	Constant + word → word	22/23	15/16	7/8	5/6
	*DM + *DM → *DM	42/44	28/29		
SBB(51)	Constant – word → word	22/24	15/16	7/8	5/6
	*DM – *DM → *DM	43/44	29/30		
MLB(52)	Constant x word → word	26/27	17/18	7/8	5/6
	*DM x *DM → *DM	6/48	31/32		
DVB(53)	Word ÷ constant → word	51/52	34/35	7/8	5/6
	*DM ÷ *DM → *DM	71/73	47/48		
ADDL(54)	Word + word → word	74/76	49/50	6/8	4/6
	*DM + *DM → *DM	92/93	61/62		
SUBL(55)	Word – word → word	73/75	49/50	6/8	4/6
	*DM – *DM → *DM	93/95	62/63		
MULL(56)	Word x word → word	187/188	125/126	6/8	4/6
	*DM x *DM → *DM	205/206	137/138		

Notes 1. The execution time is given in microseconds unless otherwise stated.
2. Times for non-differentiated forms are given to the left of the slash, and those for differentiated forms given to the right.

Instruction	Conditions	ON execution time (μs)[1,2]		OFF execution time (μs)[1,2]	
		C1000H	C2000H	C1000H	C2000H
DIVL(57)	Word ÷ word → word	192/194	128/129	6/8	4/6
	*DM ÷ *DM → *DM	210/212	140/141		
BINL(58)	When converting words to words	35/37	27/28	7/8	5/6
	When converting *DM to *DM	47/49	31/32		
BCDL(59)	When converting words to words	39/40	26/27	7/8	5/6
	When converting *DM to *DM	50/52	33/34		
BCNT(67)	When counting 1 word	51/53	34/35	7/8	5/6
(Bit Count)	When counting 4,096 words using *DM	8.2 ms	5.5 ms		
BCMP(68)	Comparing constant to word-designated table	66/68	44/45	7/8	5/6
	Comparing *DM → *DM-designated table	95/96	63/64		
XFER(70)	When transferring 1 word	46/48	31/32	7/8	5/6
	When transferring 4,096 words using *DM	5.49 ms	3.66 ms		
BSET(71)	When setting a constant to 1 word	35/37	23/24	7/8	5/6
	When setting *DM ms to 4,096 words using *DM	4.14 ms	2.76 ms		
ROOT(72)	When taking root of word and placing in a word	99/100	66/67	7/8	5/6
	When taking root of 99,999,999 in *DM and placing in *DM	109/110	73/74		
XCHG(73)	Between words	19/20	13/14	7/8	5/6
	Between *DM	31/33	21/22		
SLD(74)	When shifting 1 word	35/37	23/24	7/8	5/6
	When shifting 4,096 DM words using *DM	6.31 ms	4.20 ms	7/8	5/6
SRD(75)	When shifting 1 word	35/37	23/24	7/8	5/6
	When shifting 4,006 DM words using *DM	6.27 ms	4.18 ms		
MLPX(76)	When decoding word to word	26/28	17/18	7/8	5/6
	When decoding *DM to *DM	56/58	37/38		
DMPX(77)	When encoding a word to a word	35/36	23/24	7/8	5/6
	When encoding *DM to *DM	65/67	43/44		
SDEC(78)	When decoding a word to a word	30/31	20/21	7/8	5/6
	When decoding *DM to *DM	67/68	45/46		
FDIV(79)	Word ÷ word → word (equals 0)	52/54	35/36	7/8	5/6
	Word ÷ word → word (doesn't equal 0)	174/175	116/117		
	*DM ÷ *DM → *DM	192/194	128/129		
DIST(80)	Constant → word + (word)	25/27	17/18	7/8	5/6
	*DM → (*DM + (*DM))	47/49	31/32		
COLL(81)	(Word + (word)) → word	28/30	19/20	7/8	5/6
	(*DM + (*DM)) → *DM	48/50	32/33		

Notes
1. The execution time is given in microseconds unless otherwise stated.
2. Times for non-differentiated forms are given to the left of the slash, and those for differentiated forms given to the right.

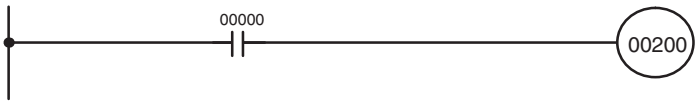
Instruction	Conditions	ON execution time (μs)[1,2]		OFF execution time (μs)[1,2]	
		C1000H	C2000H	C1000H	C2000H
MOVB (82)	When transferring word to a word	31/32	21/22	7/8	5/6
	When transferring *DM to *DM	51/52	34/35		
MOVD(83)	When transferring word to a word	27/28	18/19	7/8	5/6
	When transferring *DM to *DM	47/49	31/32		
SFTR(84)	When shifting 1 word	42/44	28/29	7/8	5/6
	When shifting 4,096 DM words using *DM	7.35 ms	4.90 ms		
TCMP(85)	Comparing constant to word-designated table	51/53	34/35	7/8	5/6
	Comparing *DM → *DM-designated table	71/73	47/48		
ASC(86)	Word → word	32/34	21/22	7/8	5/6
	*DM → *DM	74/76	49/50		
WRIT(87)	When writing 1 word	1.24 ms	0.83 ms	6/8	4/6
	When writing 255 words	6.32 ms	4.21 ms		
READ(88)	When reading 1 word	1.24 ms	0.83 ms	6/8	4/6

Notes
1. The execution time is given in microseconds unless otherwise stated.
2. Times for non-differentiated forms are given to the left of the slash, and those for differentiated forms given to the right.

6-4 I/O Response Time

The I/O response time is the time it takes for the PC to output a control signal after it has received an input signal. The time it takes to respond depends on the cycle time and when the CPU receives the input signal relative to the input refresh period. The I/O response times for a PC not in a Link System are discussed below. For response times for PCs with Link Systems, refer to the relevant *System Manual.*

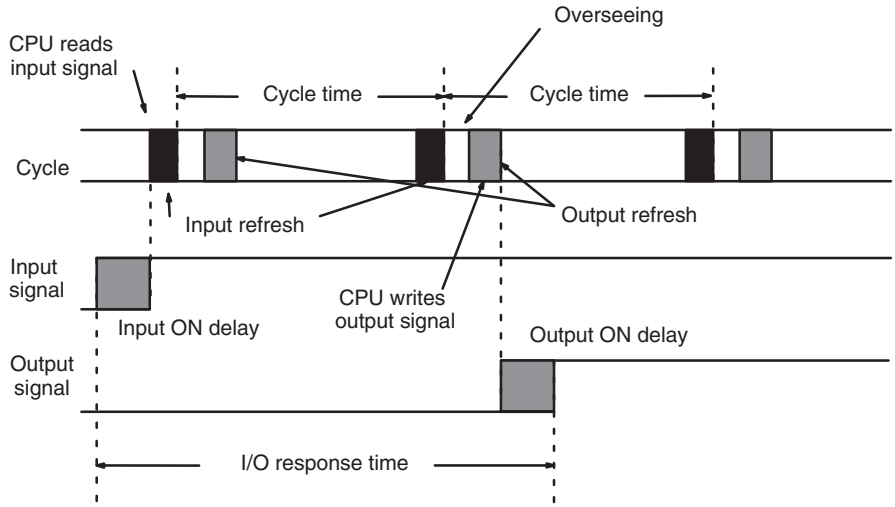
The minimum and maximum I/O response time calculations described below are for where 00000 is the input bit that receives the signal and 00200 is the output bit corresponding to the desired output point.



Address	Instruction	Operands
00000	LD	00000
00001	OUT	00200

Minimum I/O Response Time

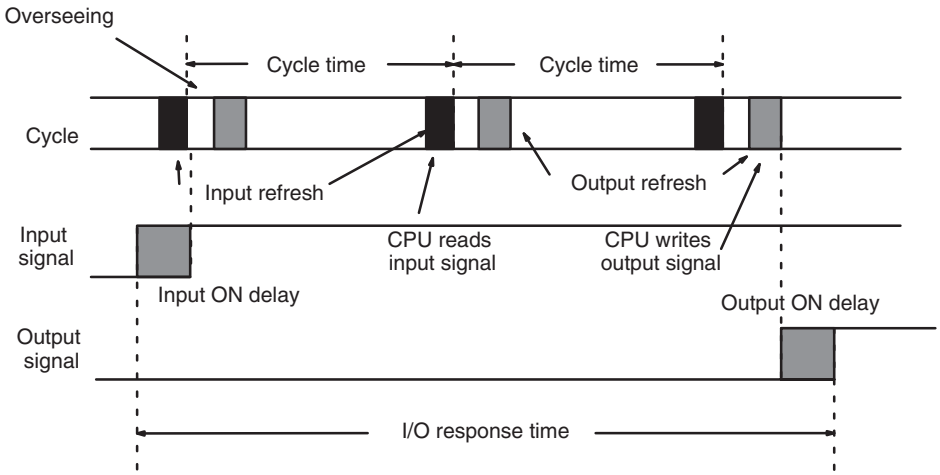
The PC responds most quickly when it receives an input signal just prior to the input refresh period in the cycle. Once the input bit corresponding to the signal has been turned ON, the program will have to be executed once to turn ON the output bit for the desired output signal and then the input refresh and overseeing operations would have to be repeated before the output refresh operation refreshes the output bit. The I/O response time in this case is thus found by adding the input ON-delay time, the cycle time (including the I/O refresh times and the overseeing time), and the output ON-delay time. This situation is illustrated below.



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

Maximum I/O Response Time

The PC takes longest to respond when it receives the input signal just after the input refresh phase of the cycle. In this case the CPU does not recognize the input signal until the end of the next cycle. The maximum response time is thus one cycle longer than the minimum I/O response time, except that the input refresh time would not need to be added in because the input comes just after it rather than before it.



Maximum I/O response time = input ON delay + (cycle time x 2) + overseeing time + output ON delay

Calculation Example

The data in the following table would produce the minimum and maximum cycle times shown calculated below.

Input ON-delay	1.5 ms
Cycle time	20 ms
Input refresh time	0.23 ms
Overseeing time	3.0 ms
Output ON-delay	15 ms

Minimum I/O response time = 1.5 + 20 + 0.23 + 3.0 +15 = 39.73 ms

Maximum I/O response time = 1.5 + (20 x 2) + 3.0 +15 = 59.5 ms

SECTION 7
Program Debugging and Execution

This section provides the procedures for debugging a program and monitoring and controlling the PC through a Programming Console.

If you are using a GPC, a FIT, or a computer running LSS, refer to the *Operation Manual* for procedures on these.

7-1 Debugging 244
7-1-1 Displaying and Clearing Error Messages 244
7-1-2 Entering Debug Mode 245
7-1-3 Address Execution 246
7-1-4 Debug Execution 248
7-1-5 Address Tracing 249
7-1-6 Address Trace Read 250
7-2 Monitoring Operation and Modifying Data 252
7-2-1 Bit/Word Monitor 252
7-2-2 Force Set/Reset 255
7-2-3 Hexadecimal/BCD Data Modification 257
7-2-4 Hex/ASCII Display Change 258
7-2-5 Three-word Monitor 259
7-2-6 Three-word Data Modification 260
7-2-7 Binary Monitor 261
7-2-8 Binary Data Modification 262
7-2-9 Changing Timer/Counter SV 263
7-3 File Memory Operations 266
7-3-1 File Memory Clear 266
7-3-2 File Memory Write 267
7-3-3 File Memory Verify 270
7-3-4 File Memory Read 272
7-3-5 File Memory Edit 275
7-4 Program Backup and Restore Operations 276
7-4-1 Saving Program Memory Data 277
7-4-2 Restoring or Comparing Program Memory Data 278
7-4-3 Saving, Restoring, and Comparing DM Data 280

7-1 Debugging

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.

The procedures in this section are designed to aid in debugging the program and quickly achieving an operative Control System.These procedures can be used in combination with the monitoring and data modification procedures provided in the next section.

All but the first of these procedures can be performed only after entering the Debug operation from PROGRAM mode.

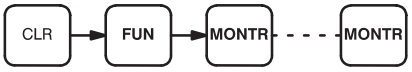
7-1-1 Displaying and Clearing Error Messages

When an error occurs during program execution, it can be displayed for identification by pressing CLR, FUN, and then MONTR. If an error message is displayed, MONTR can be pressed to access any other error messages that are stored by the system in memory. If MONTR is pressed in PROGRAM mode, the error message will be cleared from memory; be sure to write down the error message when required before pressing MONTR. OK will be displayed when the last message has been cleared. This procedure can also be used to clear messages produced by the program through MSG(46).

If a beeper sounds and the error cannot be cleared by pressing MONTR, the cause of the error still exists and must be eliminated before the error message can be cleared. If this happens, take the appropriate corrective action to eliminate the error. Refer to *Section 8 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.

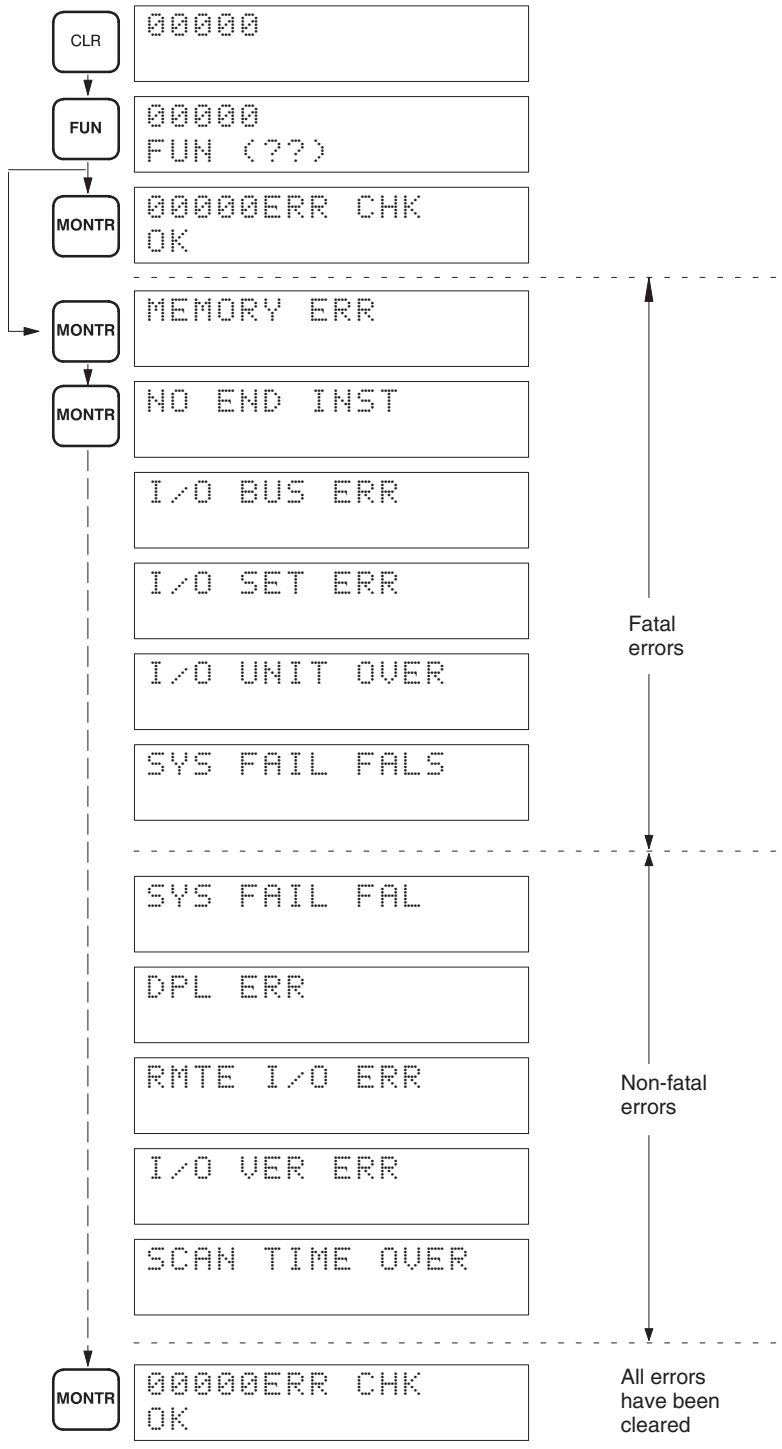
Although error messages can be displayed in any mode, they can be cleared only in PROGRAM mode. There is no way to restart the PC following a fatal error without first clearing the error message in PROGRAM mode.

Key Sequence



Example

The following displays show some of the messages that may appear. Refer to *Section 8 Troubleshooting* for an extensive list of error messages, their meanings, and the appropriate responses.

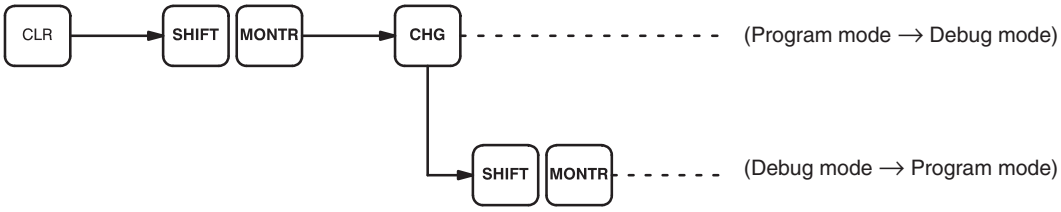


7-1-2 Entering Debug Mode

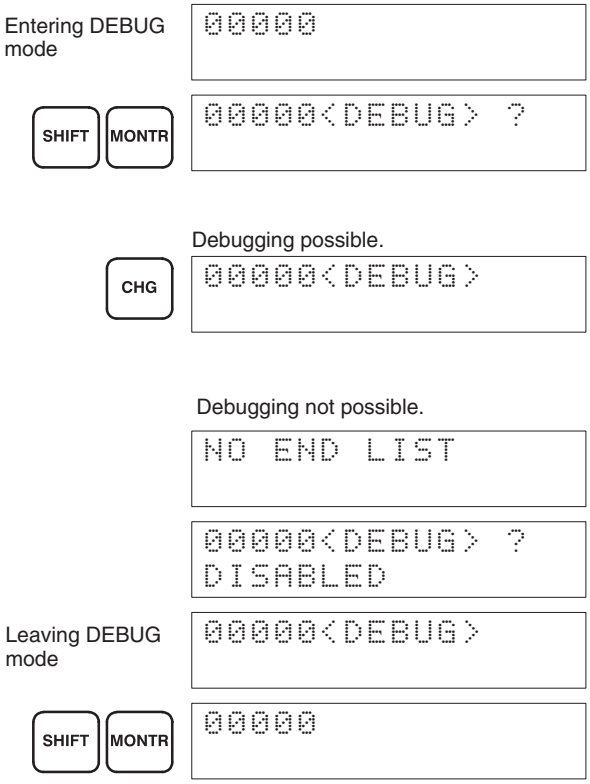
Debug mode cannot be entered if there are no instructions in the program or if there is no END(01). Although the Program Clear, Program Write, and Instruction Insert/Delete operations are not possible during debugging, all other PROGRAM mode functions are available.

When Debug mode is entered or left, data in the IR, AR, and LR areas is cleared unless the Data Retention control bit is ON (see 3-3-2 *Data Retention Control Bit)*

Key Sequence



Example



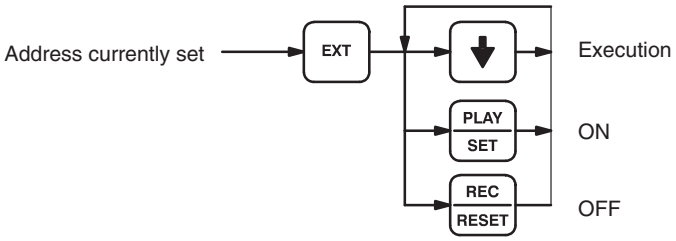
7-1-3 Address Execution

This operation is used to execute a program on an instruction-by-instruction basis while examining the execution status at each step.

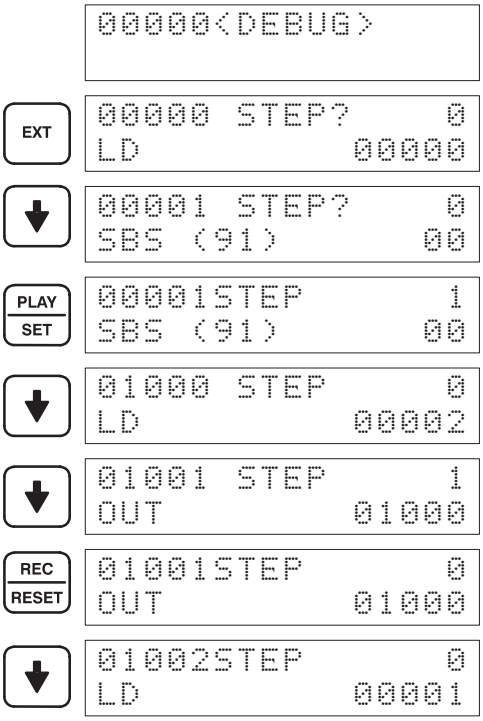
After Debug mode is entered, the address from which execution is to begin is set. EXT is then used to start execution; the down key, to go to the next address. If the address holds an instruction that ends a cycle (such as END(01) and SBN(92)) or an instruction that stops program execution (FALS(07)), address execution will be ended.

Since the results of the executed instruction are transferred to the I/O memory, word data can be monitored through the Data Monitor operation.

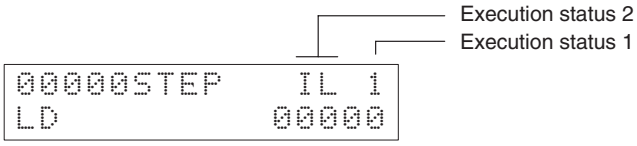
Key Sequence



Example



Meaning of Displays



Execution status 1 shows the execution condition of the current instruction or indicates that a block program is being executed. This status can be changed (set or reset) by using PLAY/SET or REC/RESET as long as B (see below) is not also being displayed. Execution status 1 values indicate the following:

0: OFF

1: ON

B: Block program execution

Execution status 2 values indicate the following:

IL: The displayed instruction is between IL(02) and ILC(03) and the IL condition is not met.

JP: The displayed instruction is between JMP(04) and JME(05) and the JMP condition is not met.

NP: Block program is not executed.

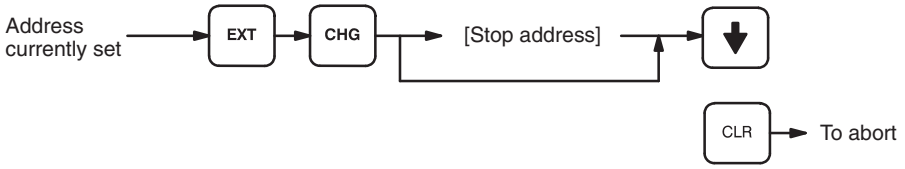
7-1-4 Debug Execution

The Debug Execution operation is used to execute the program from the currently displayed address to the address before the specified stop address.

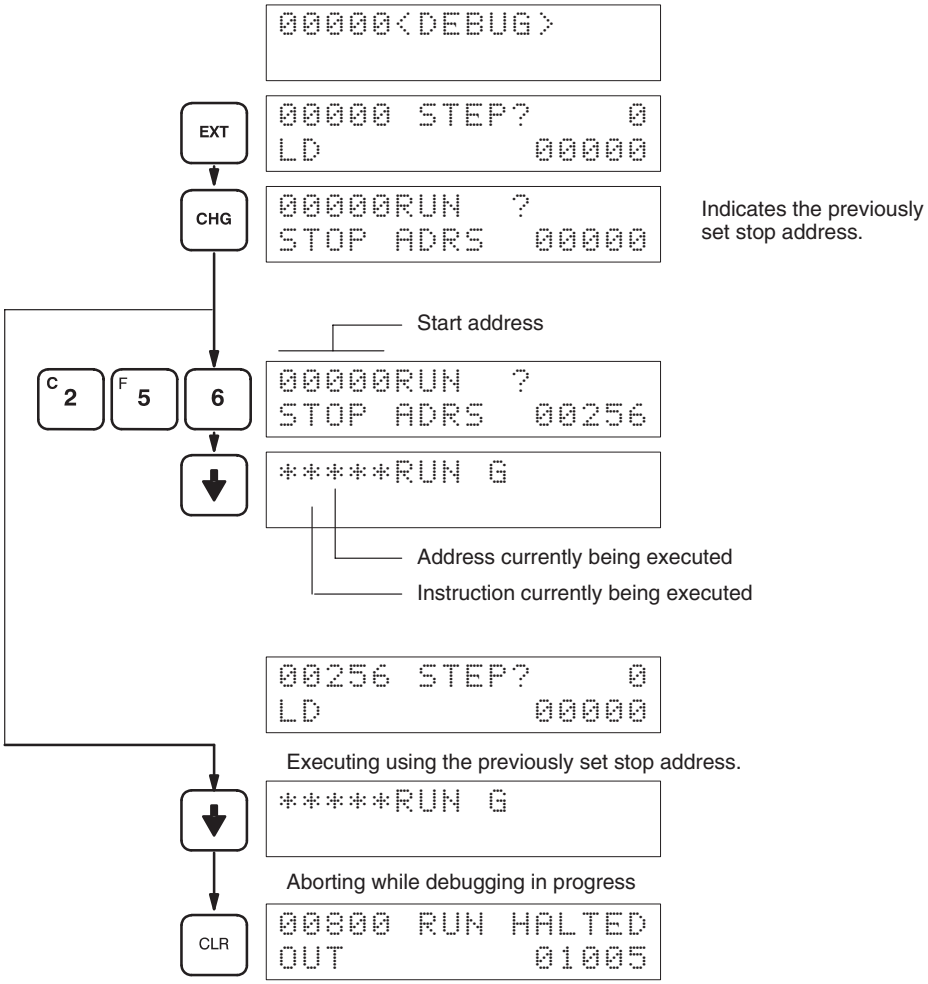
After switching from PROGRAM to Debug mode, read the program and set the address from which to start debugging. Then, press EXT and CHG and enter the stop address. Press the down key to start debugging.

If END(01) is encountered before the stop address, execution will halt. Also, program execution can be aborted by pressing CLR.

Key Sequence



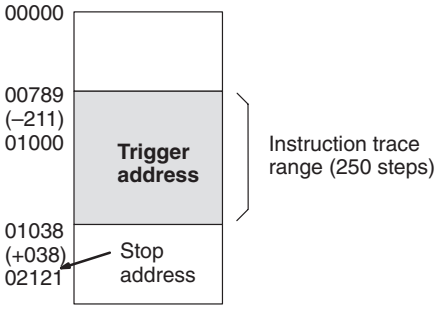
Example



7-1-5 Address Tracing

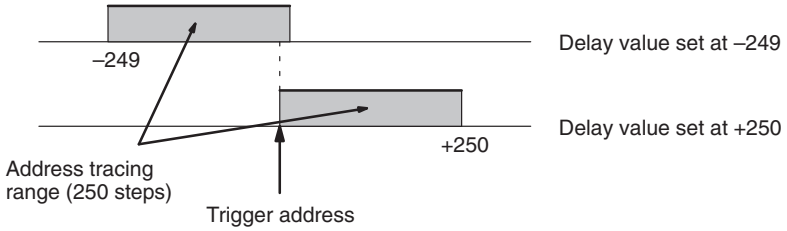
The Address Tracing operation is used to debug a section of 250 instructions and store the results in the Trace Memory. To set up the Address Tracing operation, press EXT and CHG, then specify a stop address, a trigger address, and a delay value (the address where tracing is to begin relative to the trigger address). The delay value can be any integer from –249 to +250. Use NOT to change between positive and negative delays. If the trigger address is 01000 and the delay value is –211, the trace area will range from 00789 (01000 – 211) to 01038 (00789 + 250). Even if the stop address lies outside this trace area, only those 250 instruction steps between 00789 and 01038 will be traced.

If the stop address or END(01) lies within the trace area, trace execution will go only as far as the stop address or the END(01) and then will loop back over the instructions at the start of the trace area. Address tracing will continue to loop until 250 instructions have been traced. Results for the instruction that have been recorded in Trace Memory can later be read (see the next operation) to check the status of program execution instruction-by-instruction.

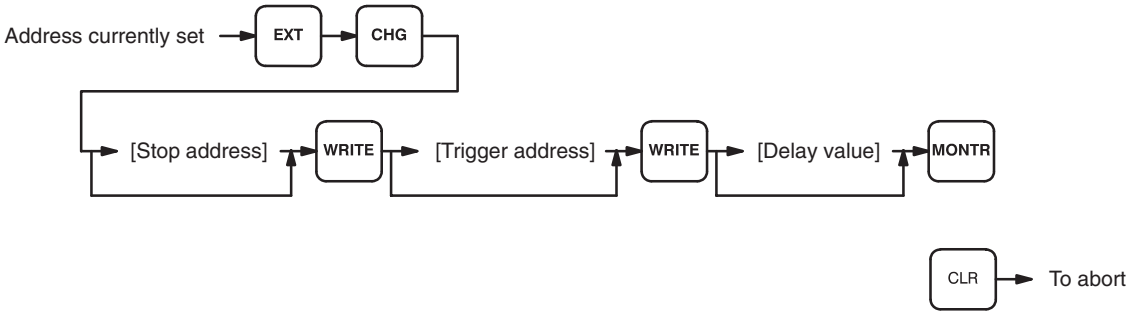


Delay Values

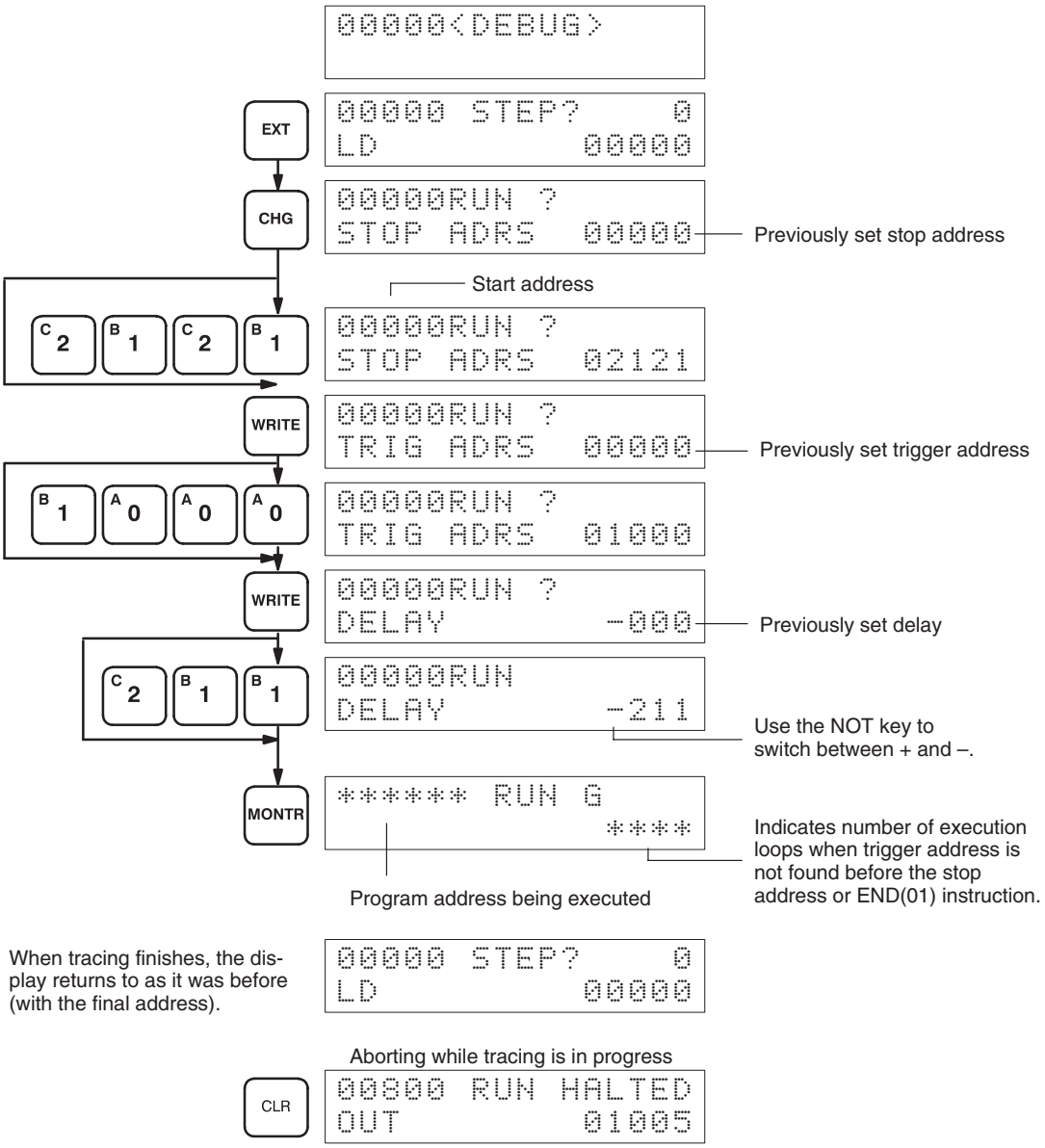
The following diagram shows the maximum and the minimum ranges from the trigger address.



Key Sequence



Example

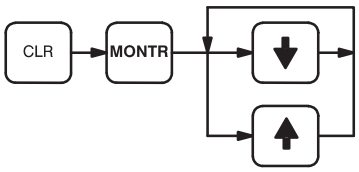


If no trigger address is found between address 00000 and either the stop address or END(01), no tracing will occur and execution will loop for 250 addresses.

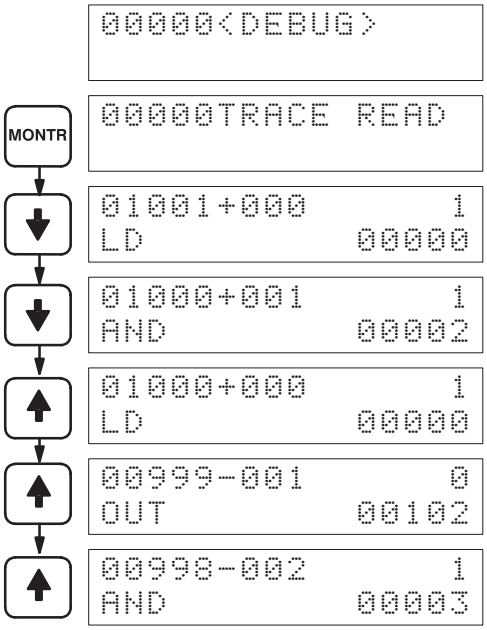
7-1-6 Address Trace Read

The Address Trace Read operation is used to read the contents of Trace Memory starting with the trigger address. Pressing the up and down keys increments or decrements the Trace Memory area address being read.

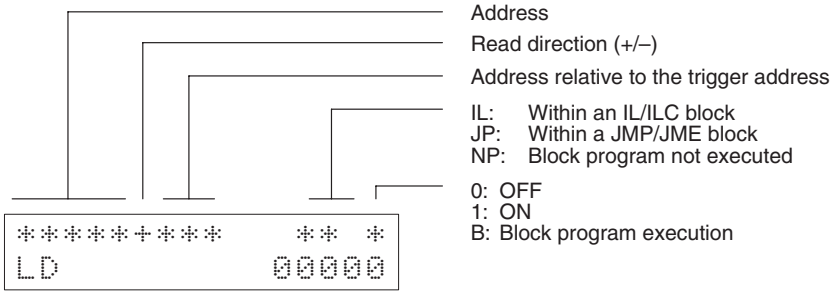
Key Sequence



Example



Meaning of Displays



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.

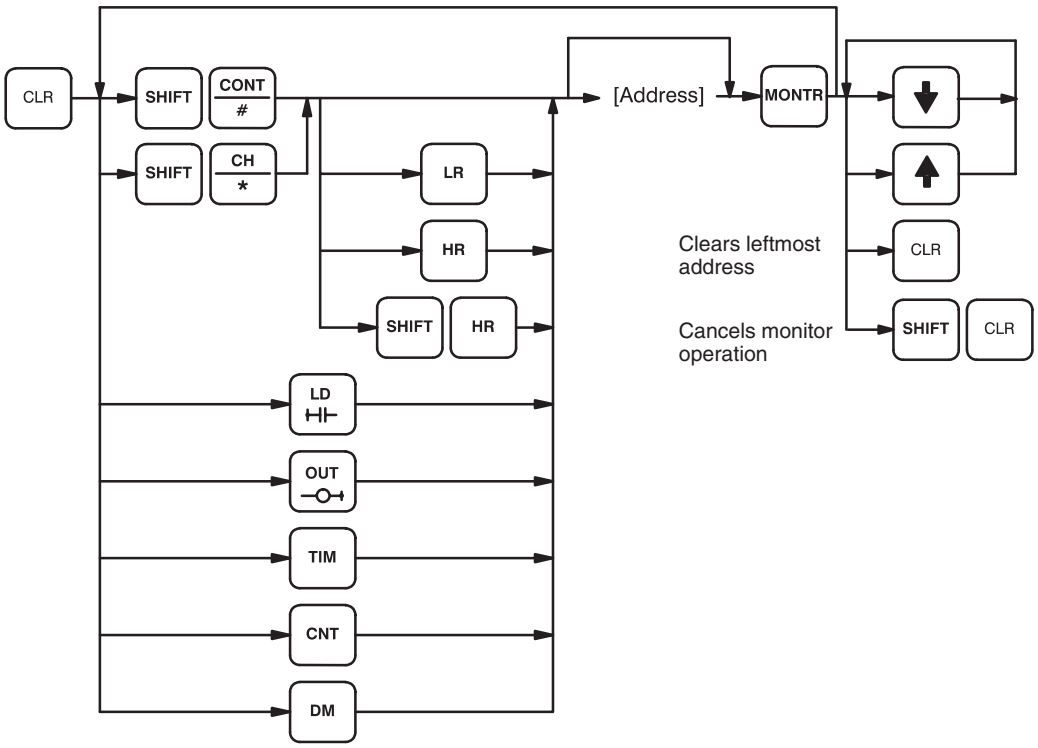
Up to six memory addresses, either bits, words, or a combination of both, can be monitored at once, although only three of these are displayed at any one time. To monitor more than one address, return to the start of the procedure and continue designating addresses. Monitoring of all designated addresses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

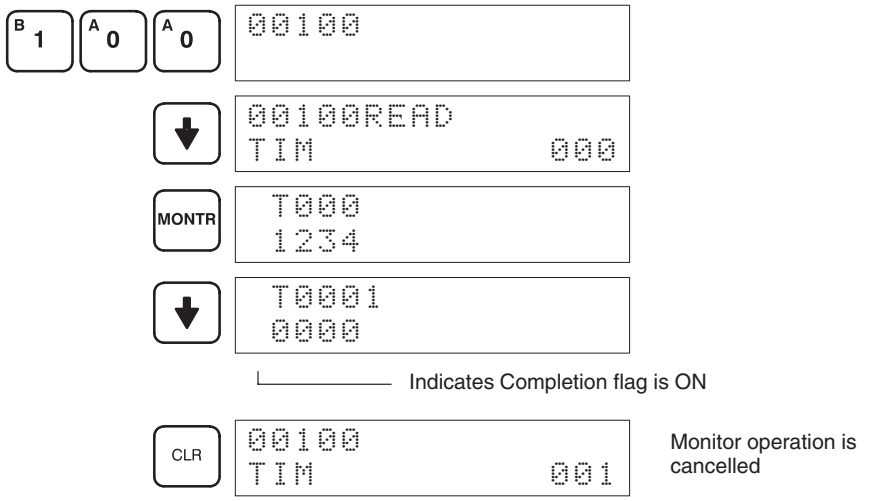
Key Sequence



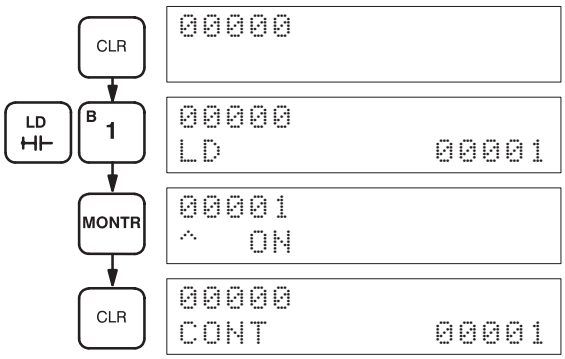
Examples

The following examples show various applications of this monitor operation.

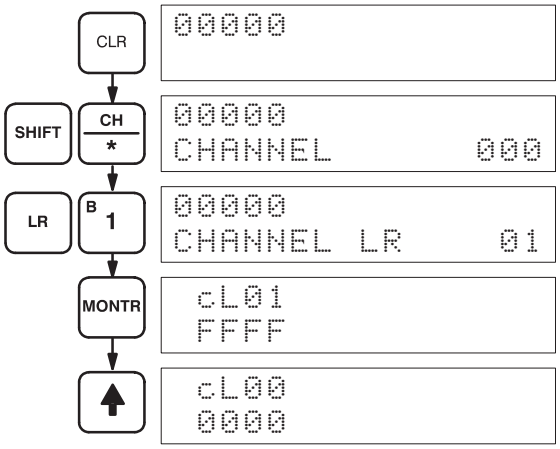
Program Read then Monitor



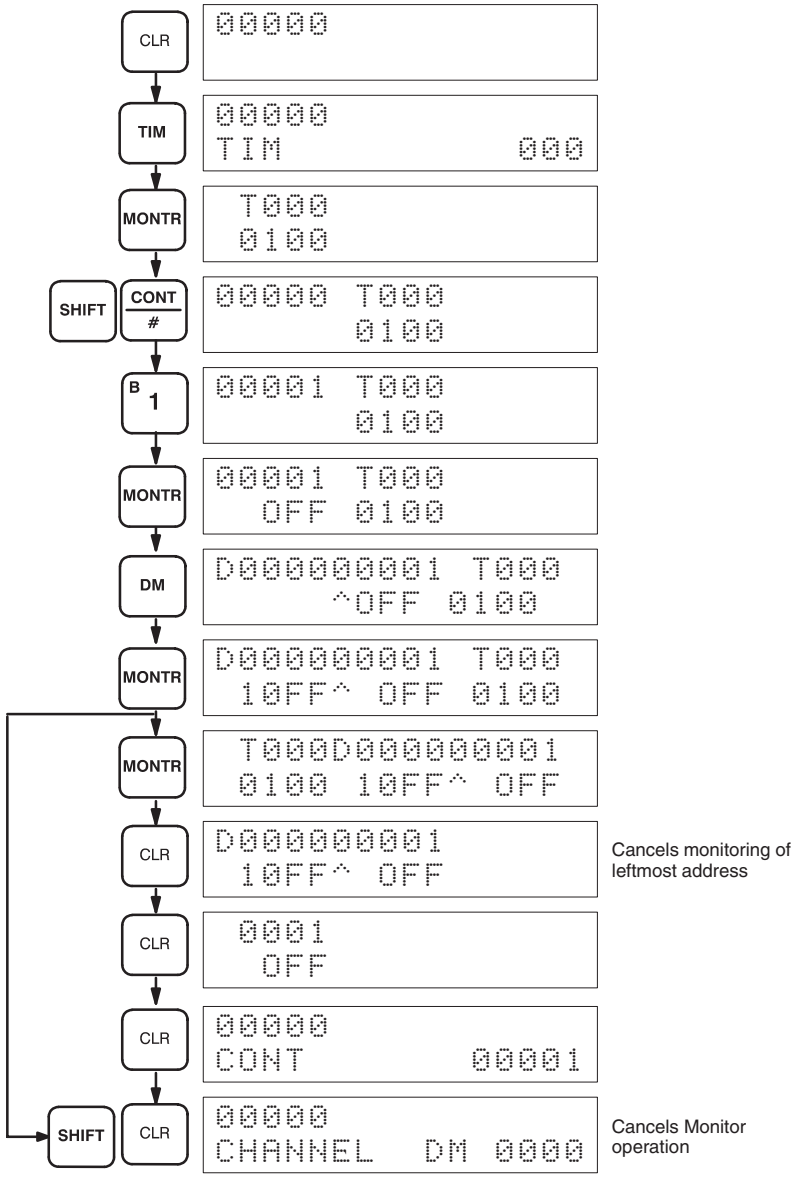
Bit Monitor



Word Monitor



Multiple Address Monitoring



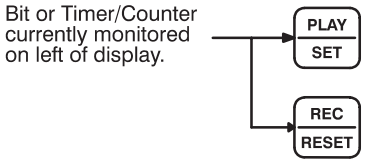
7-2-2 Force Set/Reset

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.

Bit status will remain ON or OFF only as long as the key is held down; the original status will return as soon as the key is released. If a timer is started, the completion flag for it will be turned ON when SV has been reached.

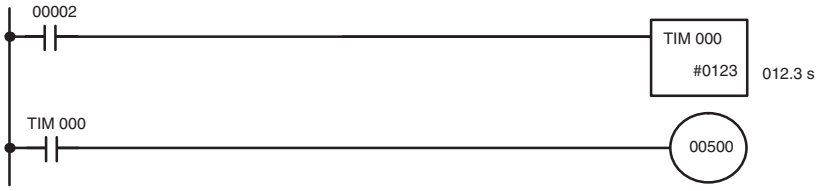
This operation can be used in MONITOR mode to check wiring of outputs from the PC prior to actual program execution. This operation cannot be used in RUN mode.

Key Sequence



Example

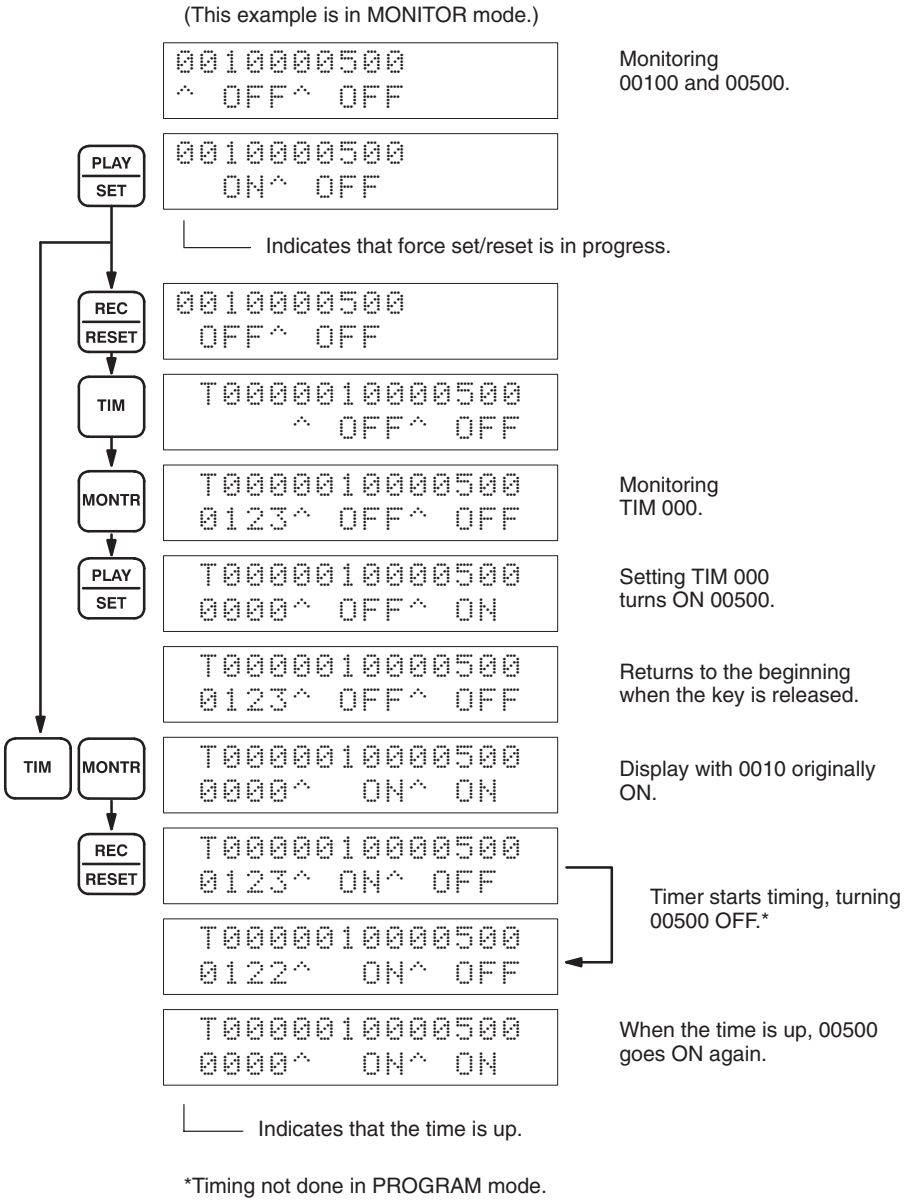
The following example shows how either bits or timers can be controlled with the Force Set/Reset operation. The displays shown below are for the following program section.



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).



*Timing not done in PROGRAM mode.

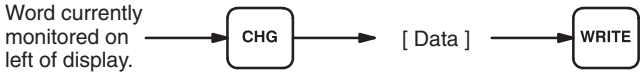
7-2-3 Hexadecimal/BCD Data Modification

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.

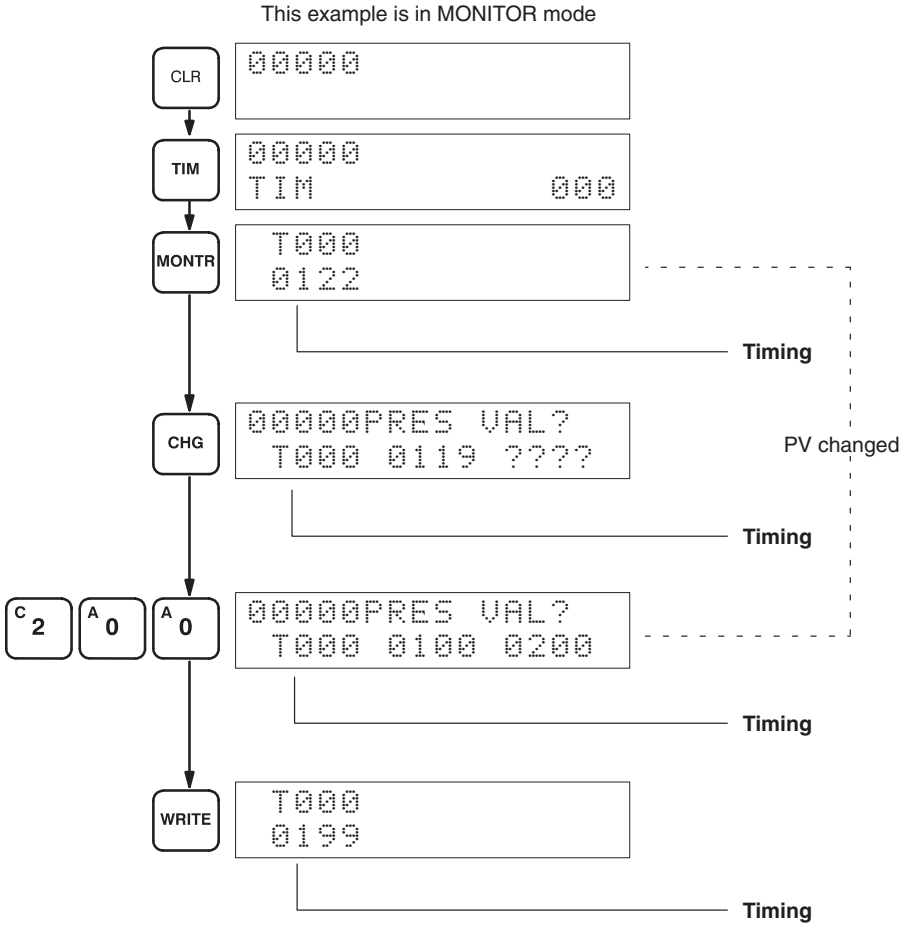
If a timer or counter is leftmost on the display, the PV will be displayed and will be the value changed. See *7-6-7 Changing Timer/Counter SV* for the procedure to change SV. PV can be changed in MONITOR mode only when the timer or counter is operating.

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

Key Sequence



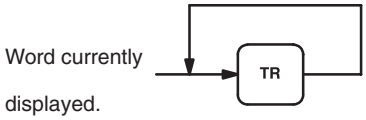
Example The following example shows the effects of changing the PV of a timer.



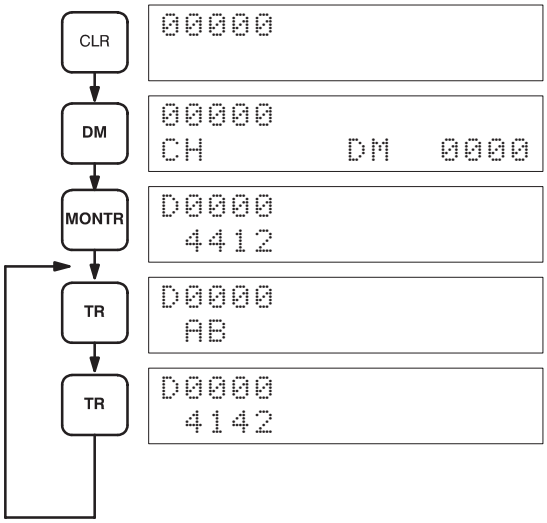
7-2-4 Hex/ASCII Display Change

This operation converts DM data displays back and forth between 4-digit hexadecimal data and ASCII.

Key Sequence



Example

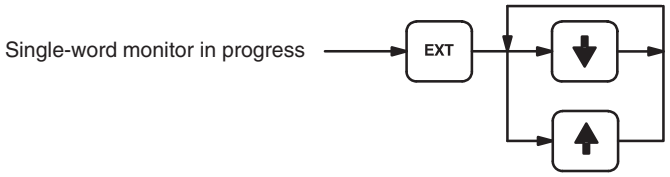


7-2-5 Three-word Monitor

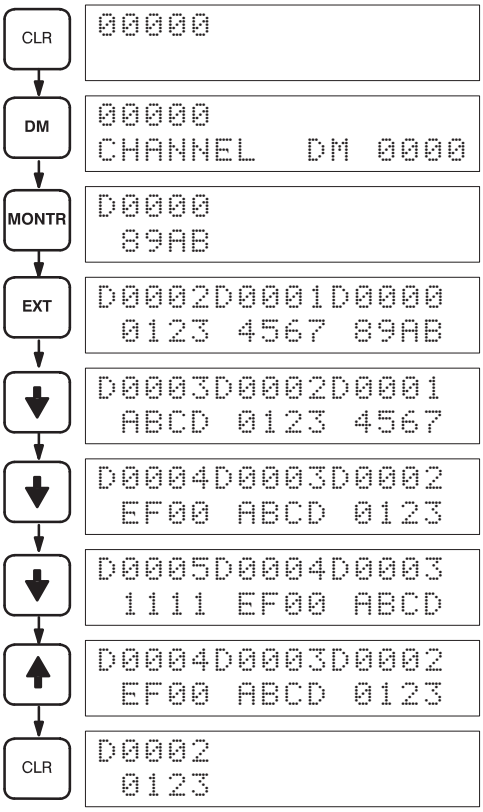
To monitor three consecutive words together, specify the lowest numbered word, press MONTR, and then press EXT to display the data contents of the specified word and the two words that follow it.

A CLR entry changes the three-word monitor operation to a single-word display.

Key Sequence



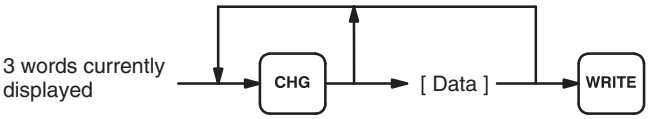
Example



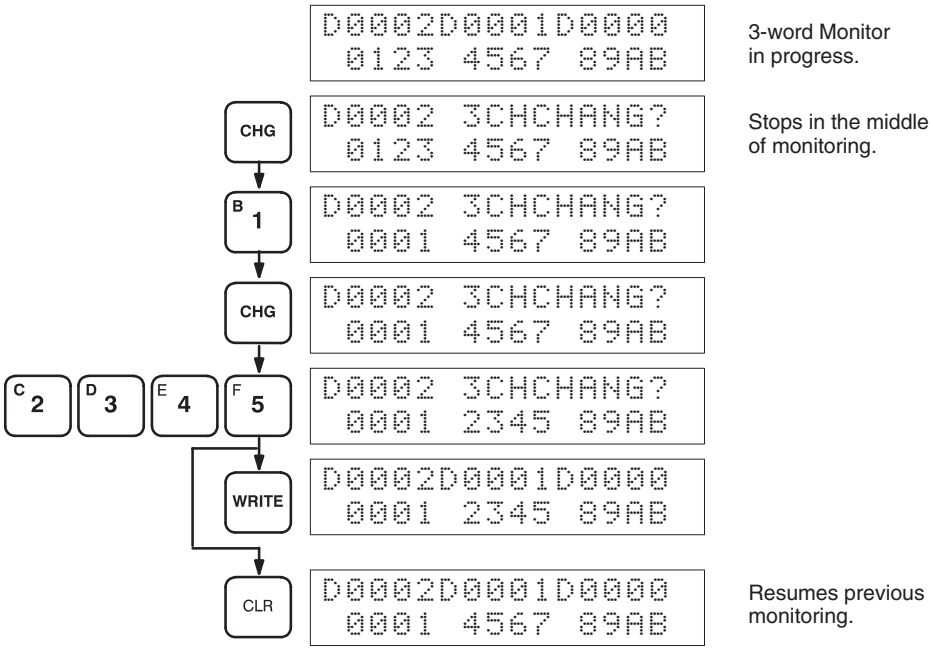
7-2-6 Three-word Data Modification

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



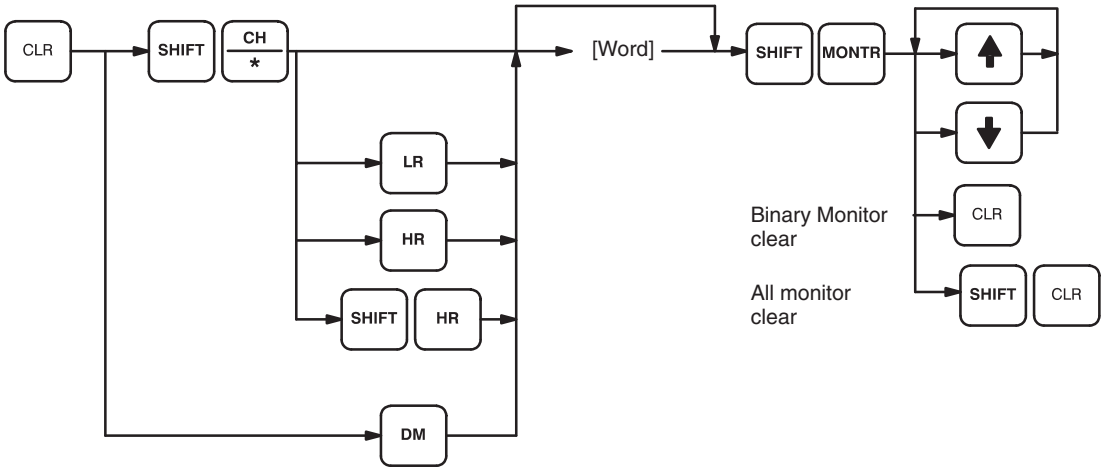
Example



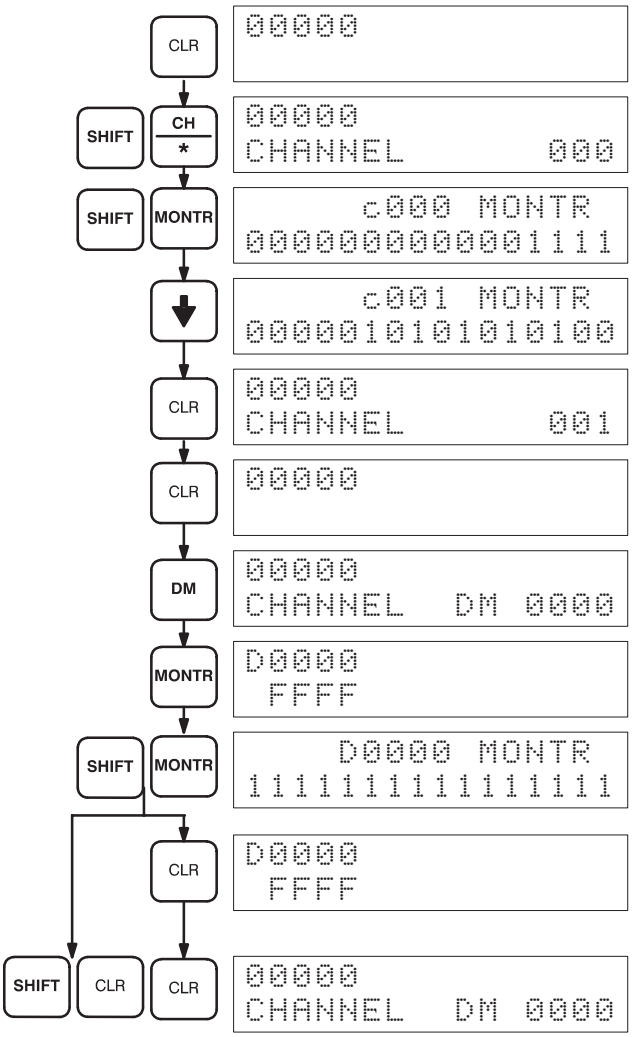
7-2-7 Binary Monitor

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

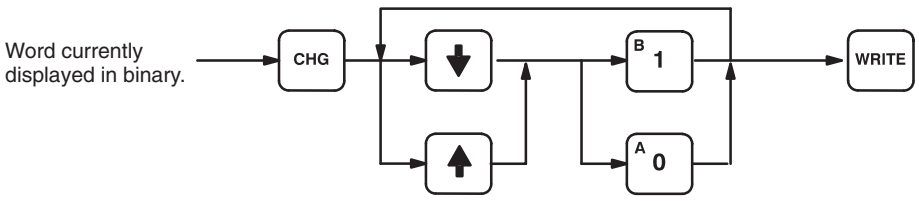


7-2-8 Binary Data Modification

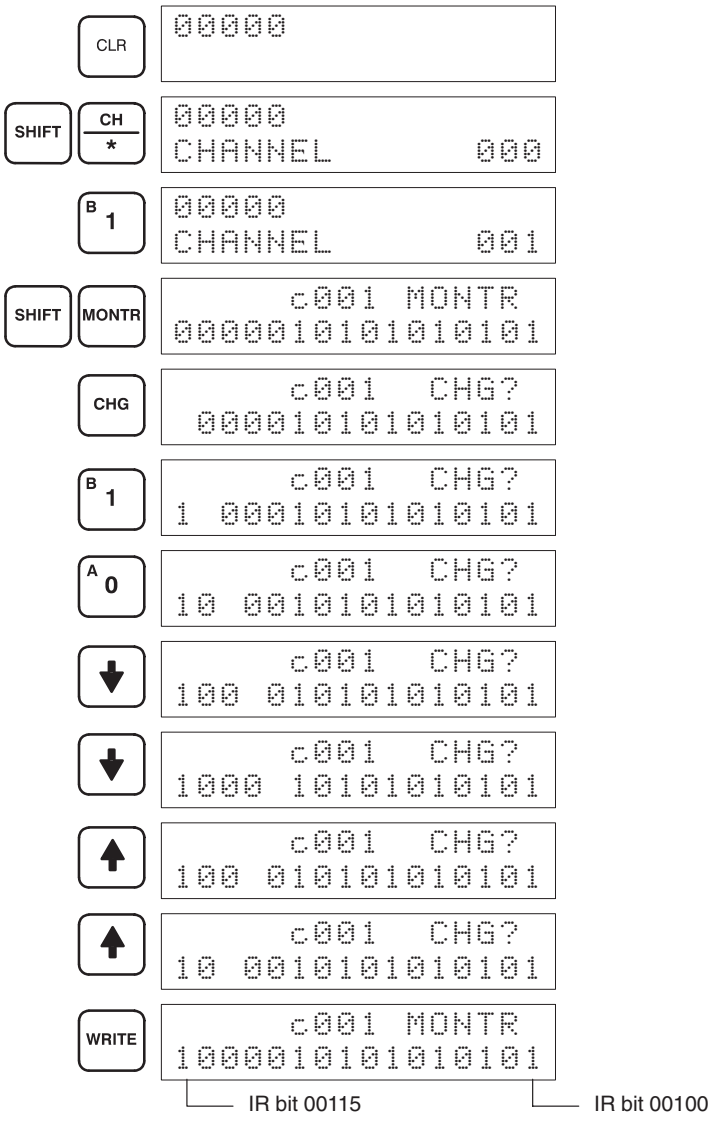
This operation assigns a new 16-digit binary value to an IR, HR, AR, LR, or DM word.

The cursor, which can be shifted to the left with the up key and to the right with the down key, indicates the position of the bit that can be changed. After positioning to the desired bit, a 0 or a 1 can then be entered as the new bit value. 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



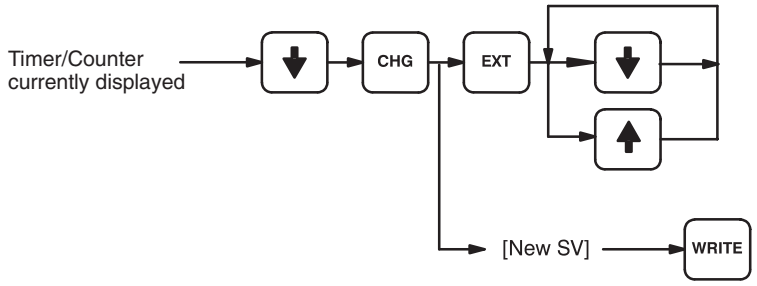
7-2-9 Changing Timer/Counter SV

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 vice versa.

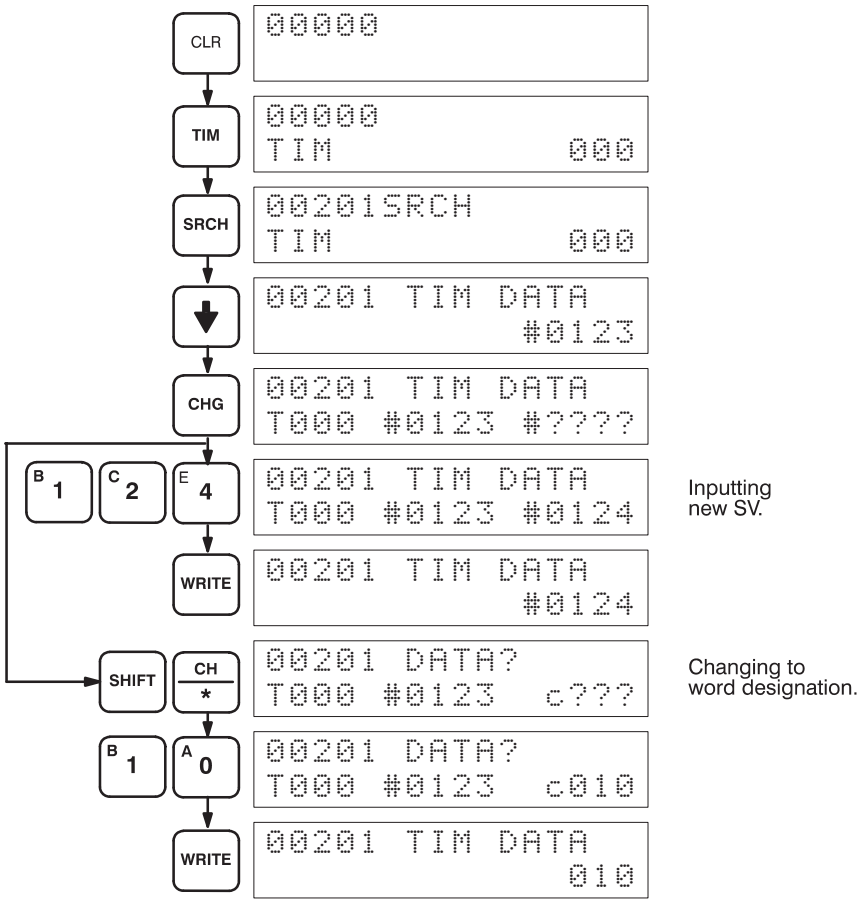
Key Sequence



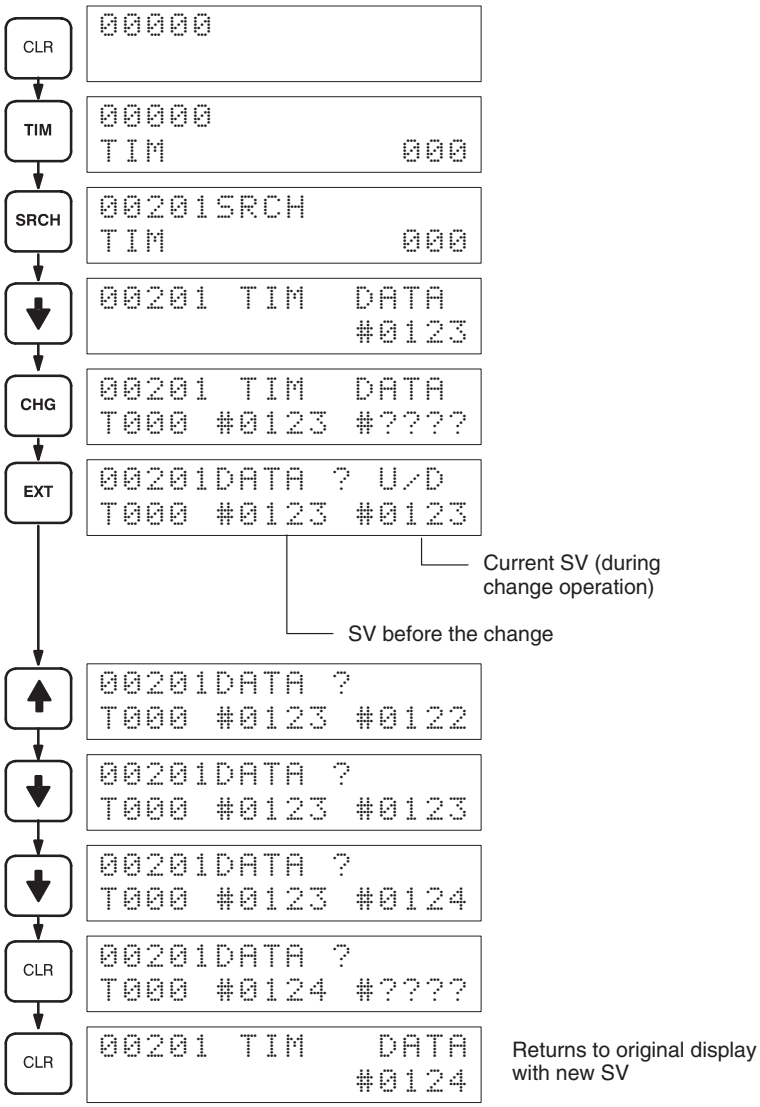
Example

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



7-3 File Memory Operations

When a File Memory Unit is connected to the PC, contents of the Program Memory and data areas can be transferred to and from the File Memory (FM) area.

The FM area can thus be used for auxiliary program memory (UM) and data memory (IOM) storage, as well as for comment memory storage (CM) for ladder logic program comments.

Except where specifically noted, these operations are available only in the MONITOR and PROGRAM modes.

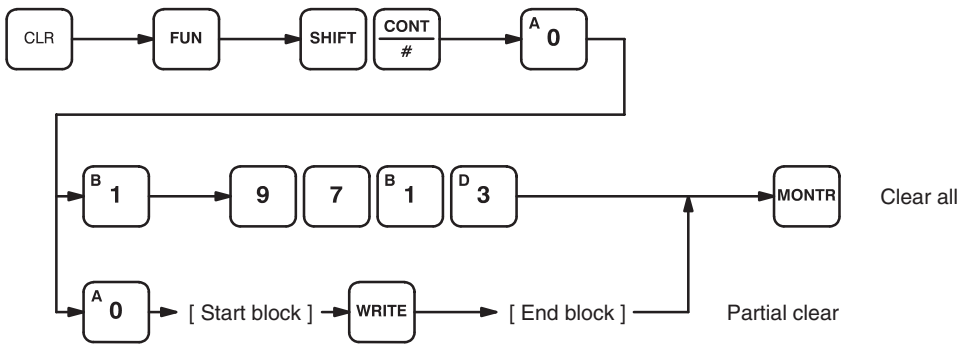
The Programming Console operations described in this section can be cancelled by pressing CLR during or after the normal key sequence.

7-3-1 File Memory Clear

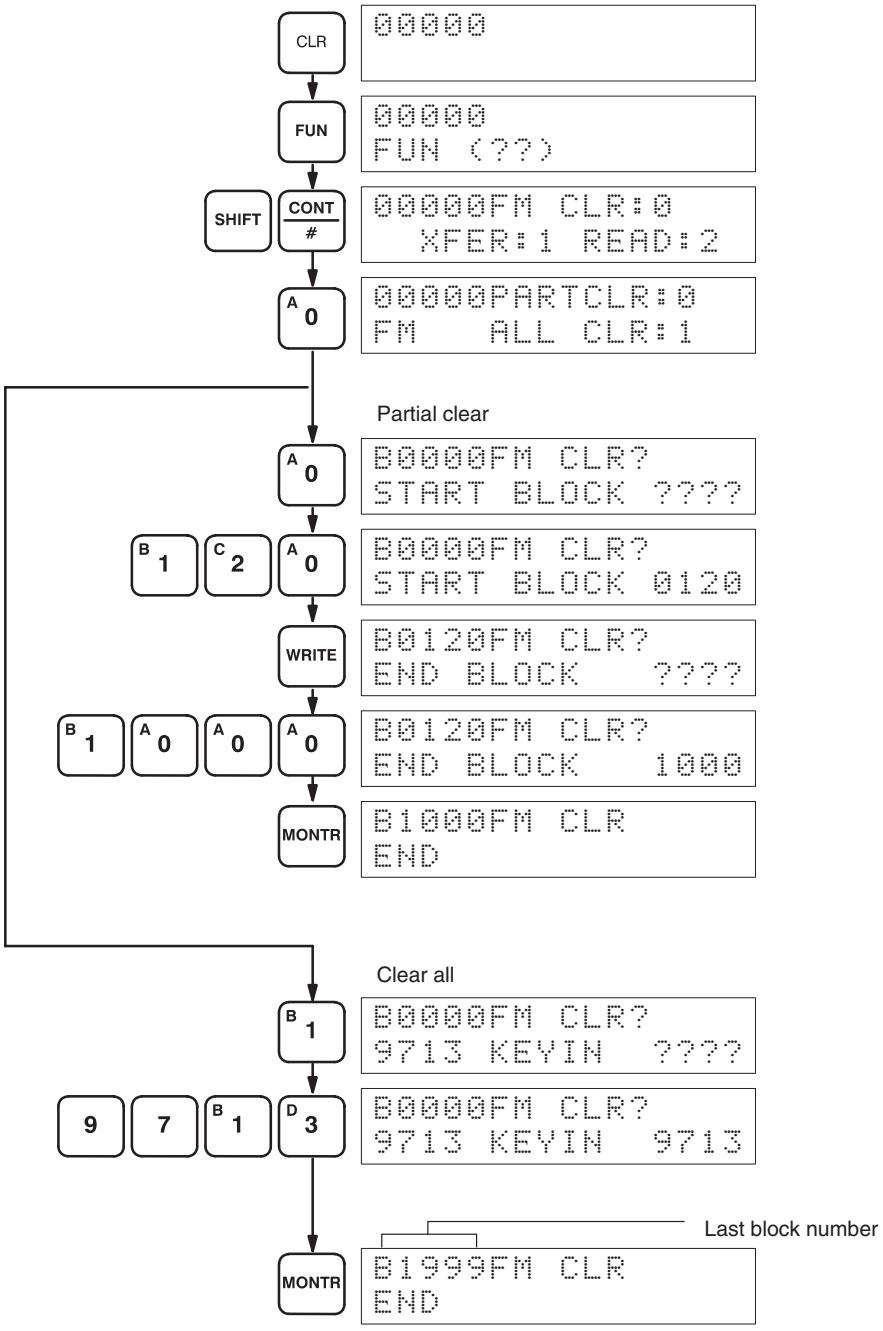
This operation clears the FM area. Clearing the entire FM area initializes it and prepares it for future data storage. This must be done when using an FM Unit for the first time or when an FM error occurs because of memory damage.

To clear a portion of the FM, you must specify a start block number and an end block number. FM is not initialized for partial clears.

Key Sequence



Example



7-3-2 File Memory Write

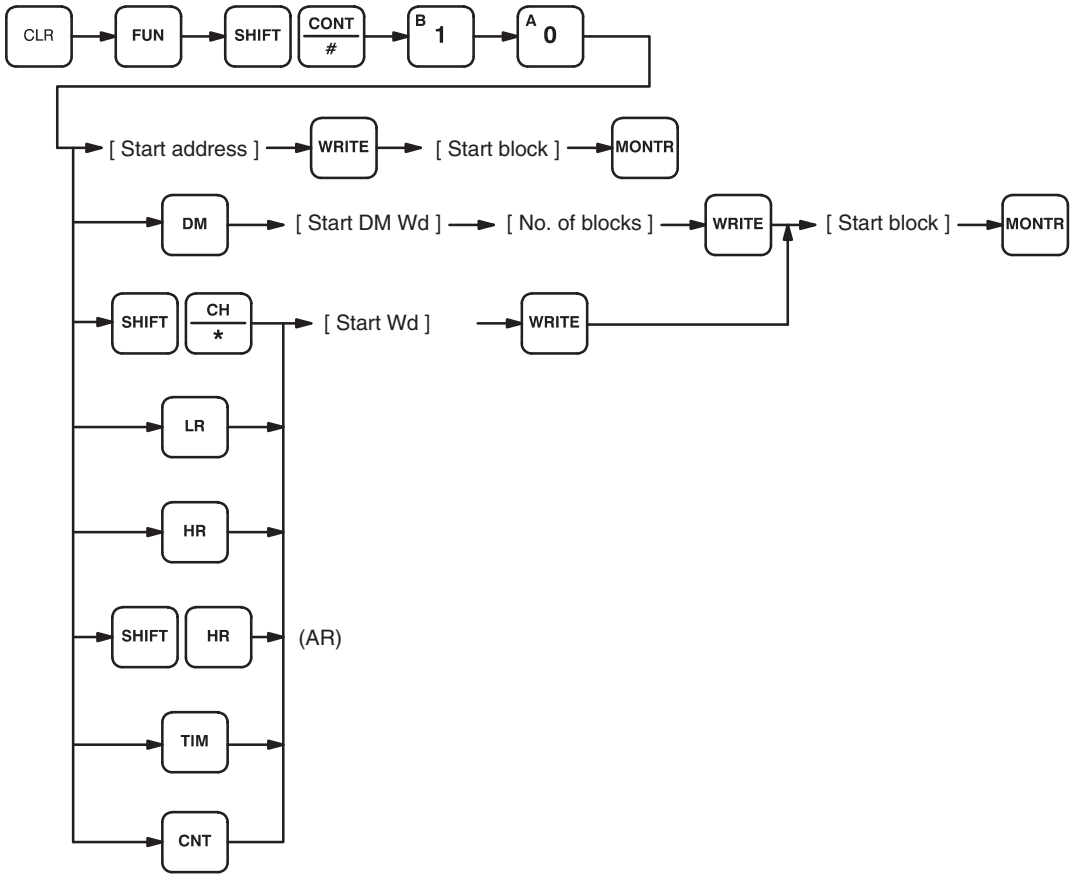
The File Memory Write operation writes data to the FM area. Data either from program memory or a data area can be transferred to the FM area with this operation. Data is written in blocks of 128 words.

When transferring from program memory, data between the specified starting address and the next END(01) in the program is moved into the FM area starting at the FM start block. If the size of the program data exceeds the space between the FM start block and the end of the FM area, only that part of the program data which will fit into the FM area will be transferred.

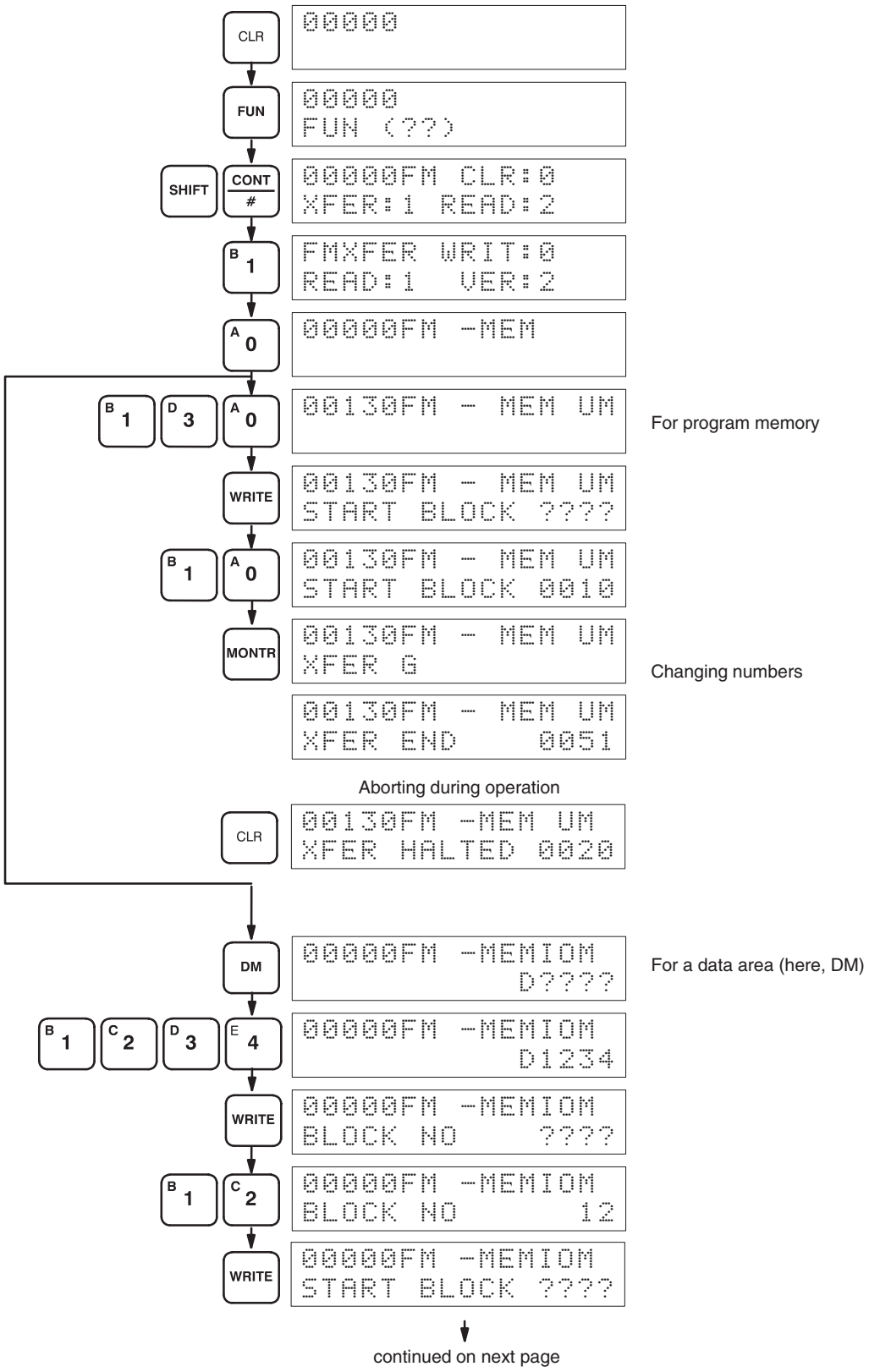
When transferring from a data area, data between the specified start word and the last word in the particular data area will be transferred. For data

transfer from the DM area, you must specify the number of blocks to be transferred.

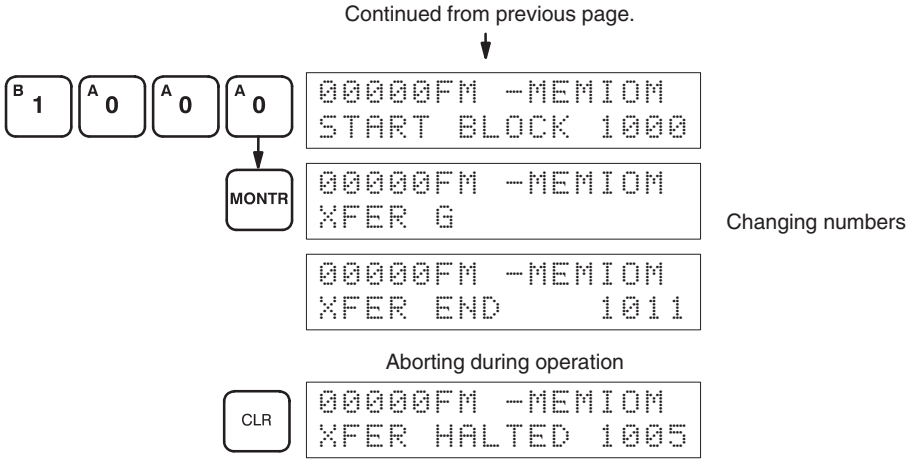
Key Sequence



Example



continued on next page



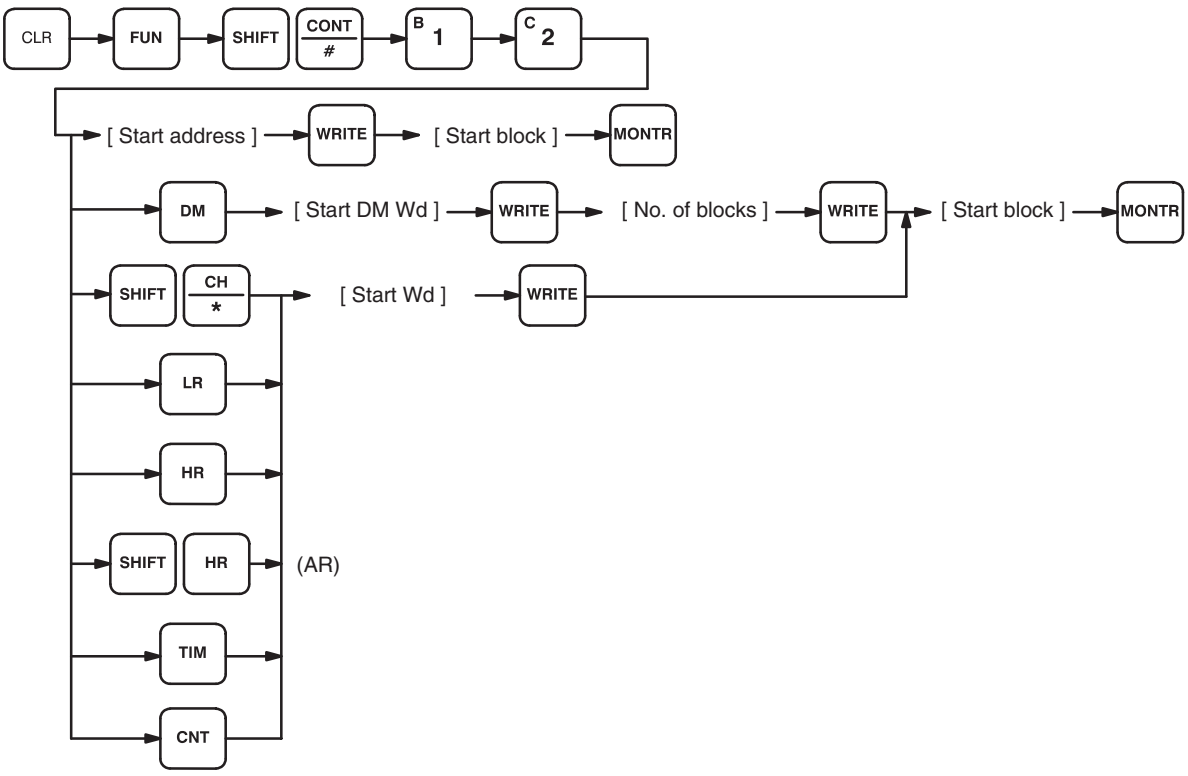
7-3-3 File Memory Verify

The File Memory Verify operation compares data either in program memory or in a data area with data stored in the FM area. If the two sets of data differ, a ”VER ERR” message will be displayed on the Programming Console.

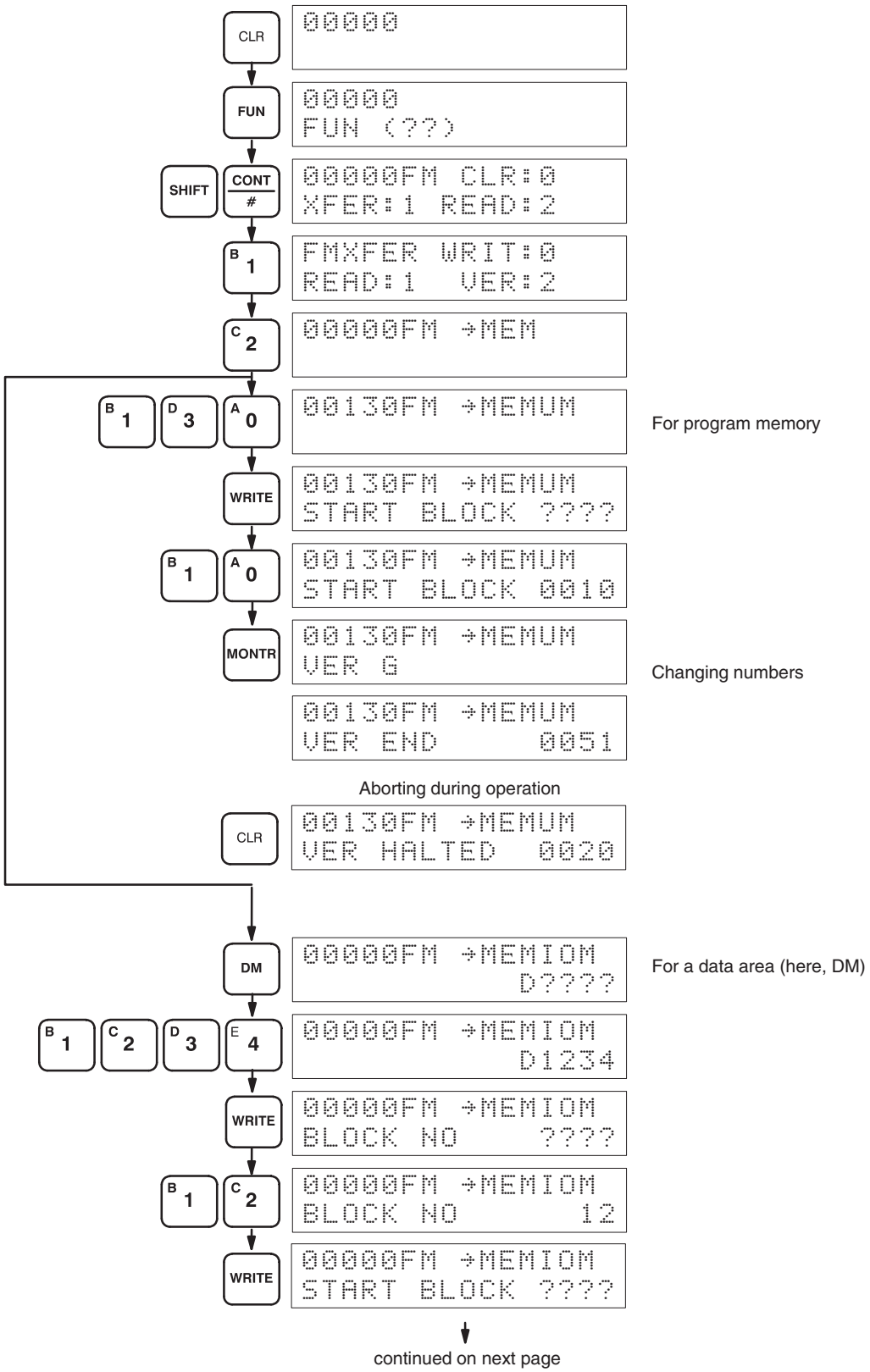
When data in program memory is being verified, this operation compares program memory data starting at the specified address and ranging to END(01), to FM data starting at a specified block (the start block).

When data in a data area is being verified, this operation compares data between the specified start word and the last word in the particular data area, to data in an FM area. For DM area verification, you must specify the number of blocks to be verified (one block consists of 128 words).

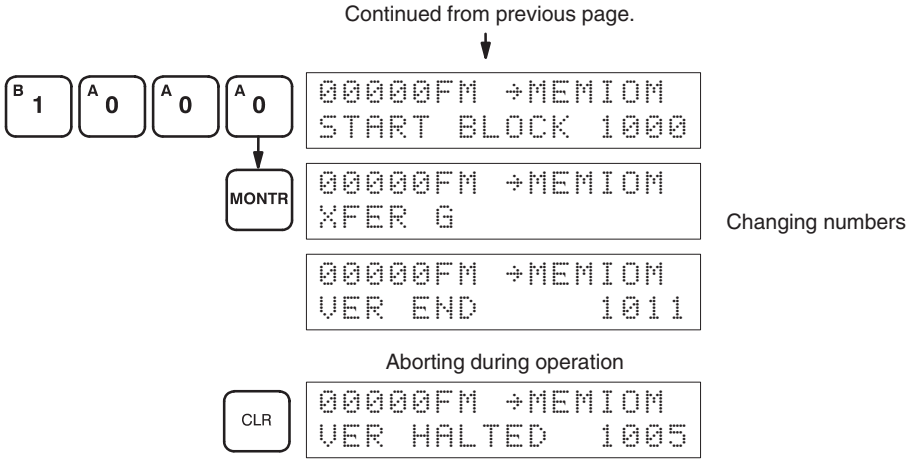
Key Sequence



Example



continued on next page



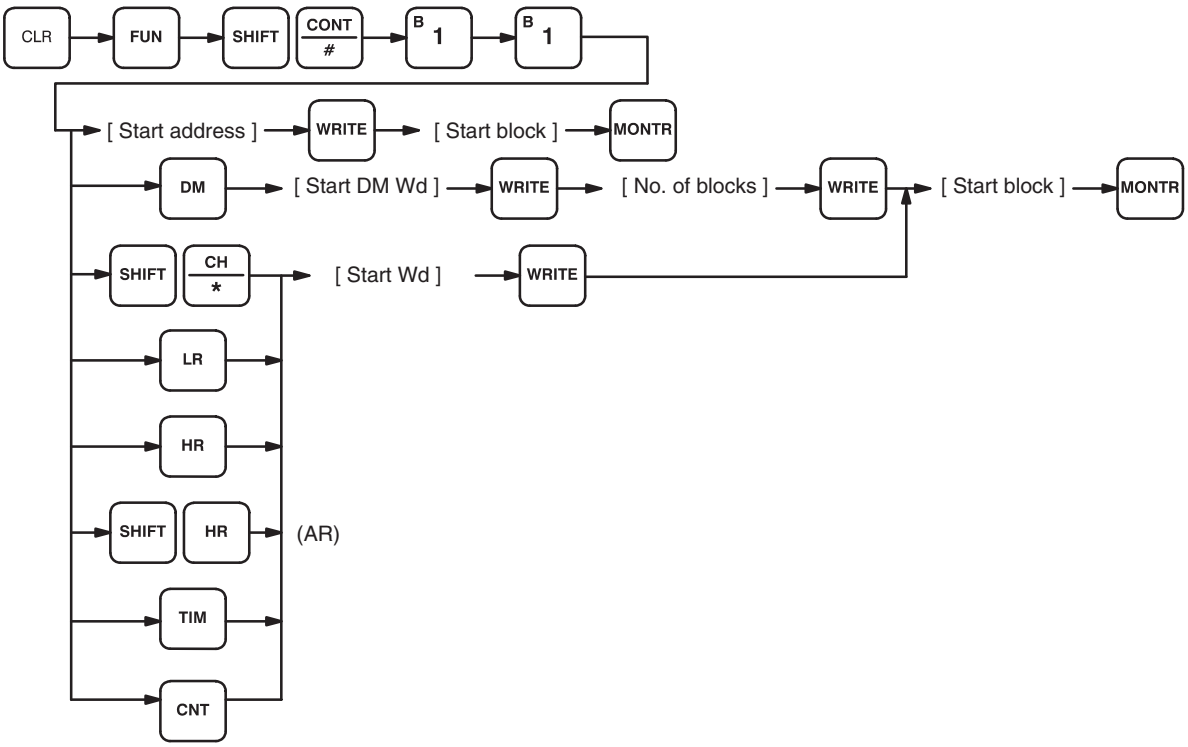
7-3-4 File Memory Read

The File Memory Read operation is used to read user program data (UM) stored in the FM area and transfer it to a specified area in RAM Program Memory, or to read user data (IOM) in the FM area and transfer it to one of the CPU data areas. The data is read and transferred in blocks of 128 words.

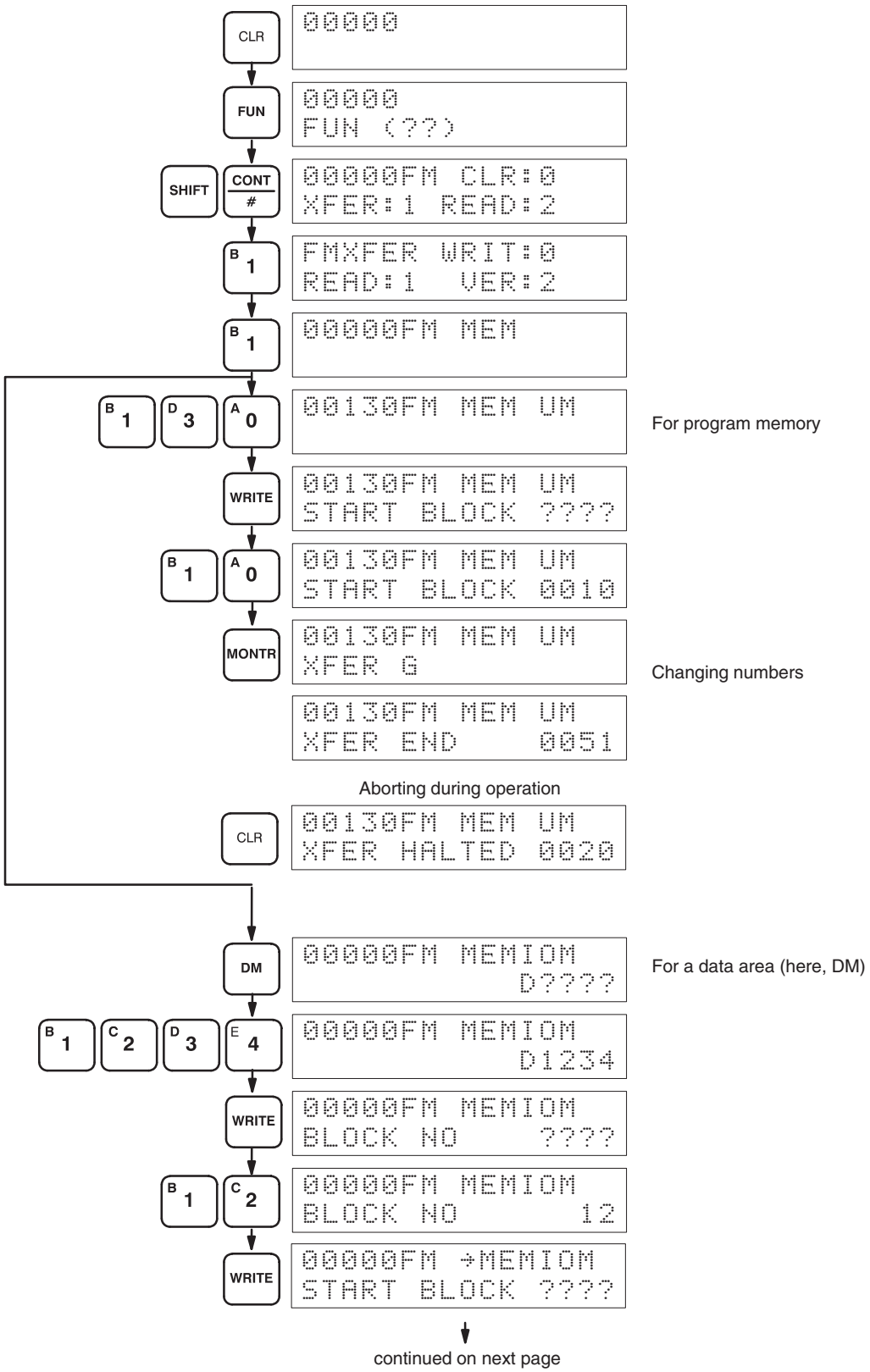
When reading data to be transferred to program memory, reading begins at the specified FM start block. Reading and block transfer end when either the first END(01) or non-UM (i.e., CM or IOM) FM block is encountered, or when the RAM user program memory area to which the data is being transferred overflows.

When reading data for a data area, reading begins at the specified FM start block and continues to the end. If a non-IOM block is encountered, an error message will be displayed and transfer will not be possible. When transferring data to the DM area, you must specify the number of blocks to be read and transferred.

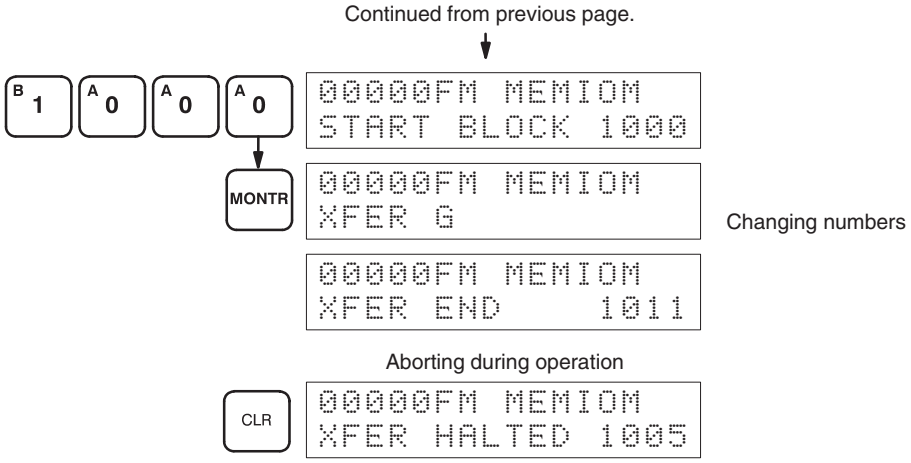
Key Sequence



Example



continued on next page



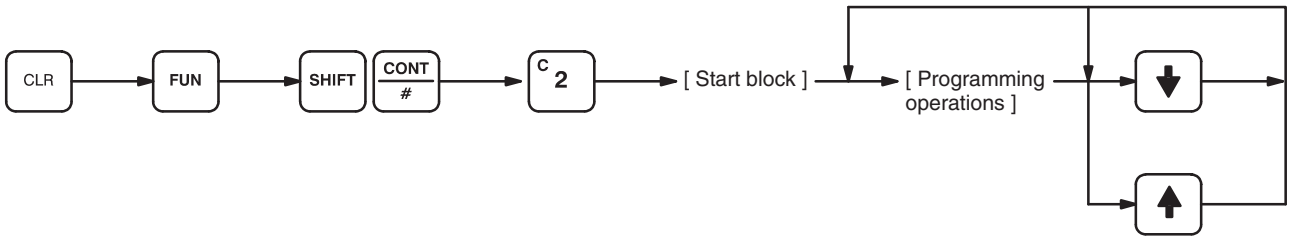
7-3-5 File Memory Edit

The File Memory Edit operation allows you to read and modify data stored in the FM area (IOM). Data can be read in RUN mode with this function, but it can be modified only when in MONITOR or PROGRAM mode.

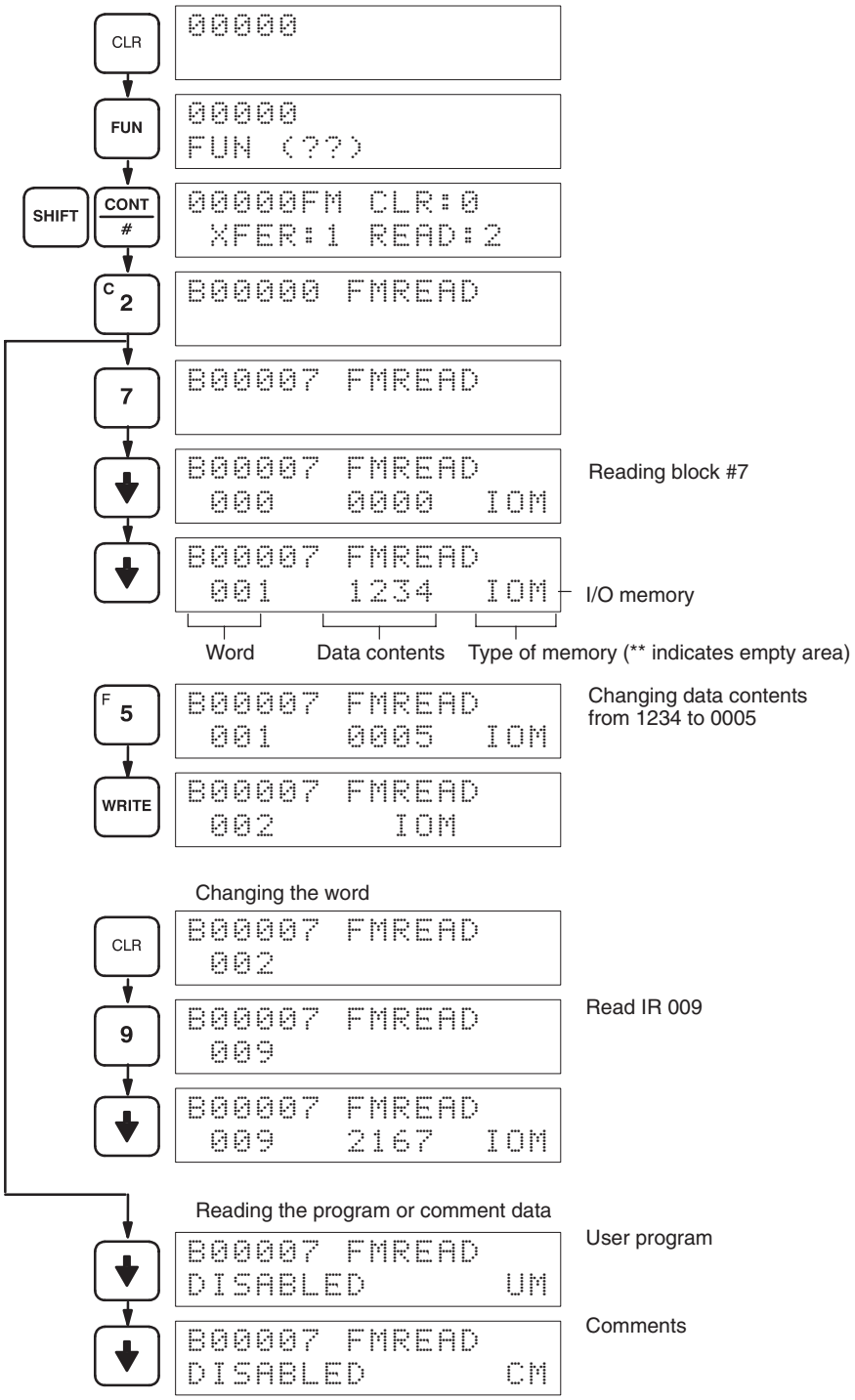
If you do not specify a start block, the reading will begin at the first block of the FM area. To access words within the current block, use the up and down keys to scroll through word-by-word, or to go directly to a specific word within the block by entering CLR, the word number, then the down key.

To change the contents of an FM area word, enter the new data for the current word, followed by WRITE. This will write the data into the FM area word. Program and comment data stored in the FM area cannot be rewritten with this operation.

Key Sequence



Example



7-4 Program Backup and Restore Operations

Both Program Memory (UM) and DM area data can be backed-up on a standard, commercially available cassette tape recorder. Any dependable magnetic cassette tape of adequate length will suffice. To save a 16K-word program, the tape must be 30 minutes long. Always allow for about 5 seconds of blank leader tape before the taped data begins. Store only one program or section of DM area on a single side of a tape; there is no way to identify separate programs or DM areas stored on the same side of the tape. If a program is too long for one side, it can be split onto two sides.

Be sure to clearly label all cassette tapes.

Use patch cords to connect the cassette recorder earphone (or LINE-OUT) jack to the Programming Console EAR jack and the cassette recorder microphone (or LINE-IN) jack to the Programming Console MIC jack. Set the cassette recorder volume and frequency equalizer controls to maximum levels.

The PC must be in PROGRAM mode for all cassette tape operations.

While the operation is in progress, the cursor will blink and the block count will be incremented on the display.

Cassette tape operations may be halted at any time by pressing CLR.

Error Messages

The following error messages may appear during cassette tape operations.

Message	Meaning and appropriate response
0000 ERR *******	File number on cassette and designated file number are not the same. Repeat the operation using the correct file number.
FILE NO.********	
**** MT VER ERR	Cassette tape contents differs from that in the PC. Check content of tape and/or the PC.
**** MT ERR	Cassette tape is faulty. Replace it with another.

7-4-1 Saving Program Memory Data

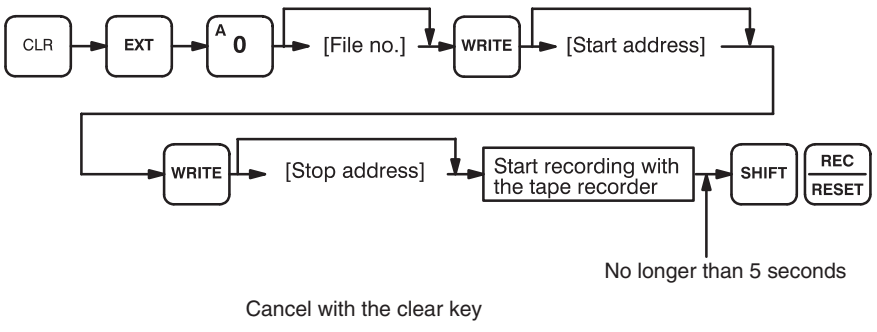
This operation is used to copy the content of Program Memory to a cassette tape. The procedure is as follows:

1, 2, 3...

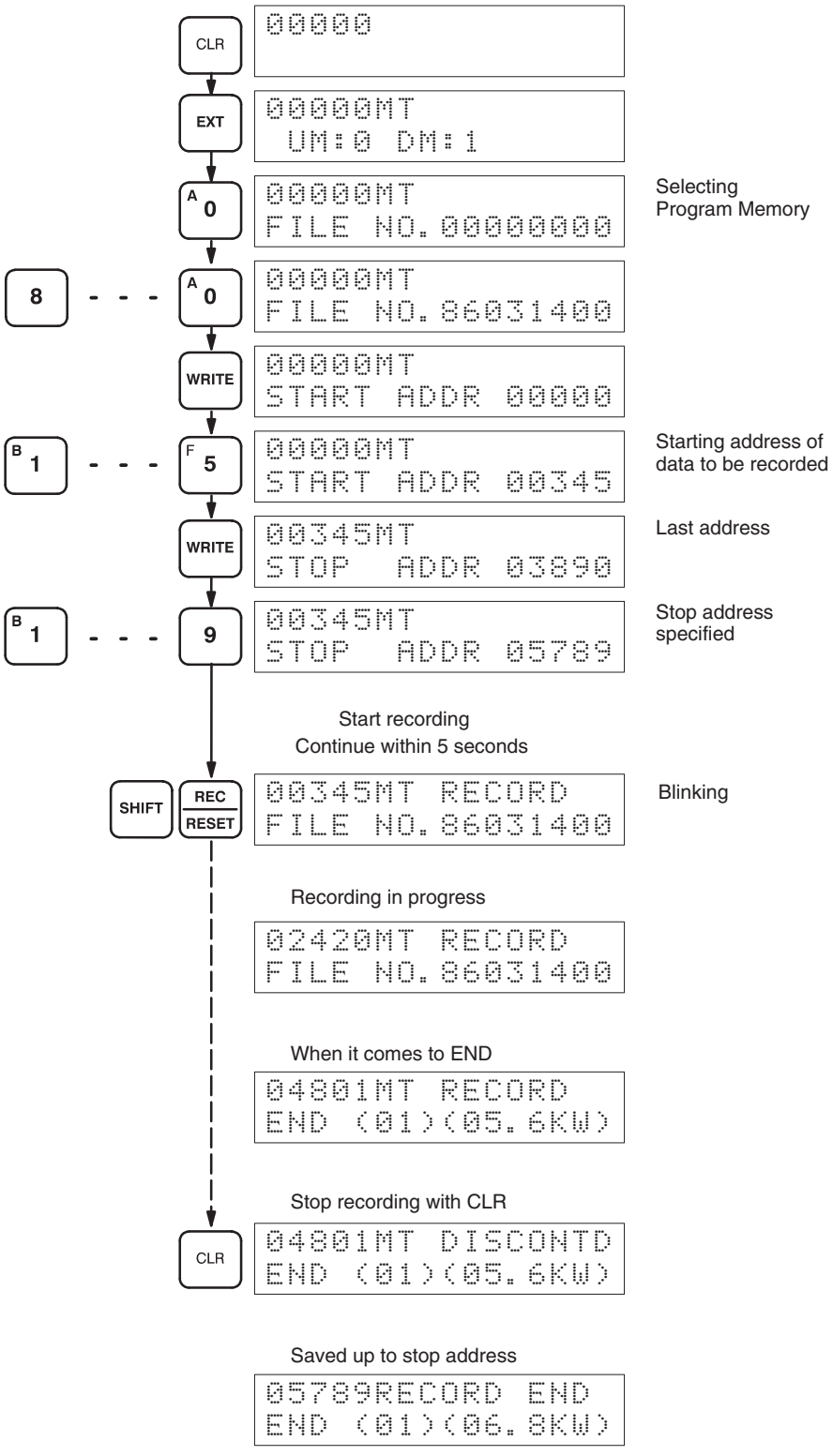
1. Press EXT and the 0 key to specify Program Memory.
2. Input a file number for the data that is to be saved.
3. Specify the start and stop addresses of the section of Program Memory that is to be recorded. When the start address is designated, the default stop address will indicate the last address that can be stored on one side of a cassette tape (i.e., approximately 16K words for one side of a C60 tape). Do not designate a stop address greater than this one.
4. Start cassette tape recording.
5. Within 5 seconds, press SHIFT and REC/RESET.

Program saving continues until END(01) or the stop address is reached. At that time the program size in Kwords is displayed. If the END(01) is reached before the stop address, the recording operation will continue, however, through the designated stop address unless CLR is pressed to cancel.

Key Sequence



Example



7-4-2 Restoring or Comparing Program Memory Data

This operation is used to restore Program Memory data from a cassette tape or to compare Program Memory data with the contents on a cassette tape. The procedure is as follows:

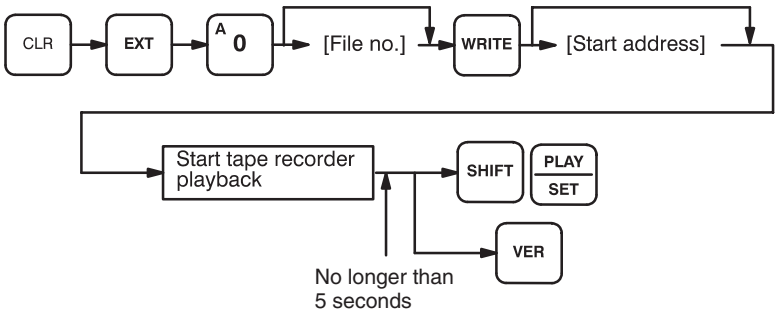
1, 2, 3... 1. Press EXT and the 0 key to specify Program Memory.
2. Specify the number of the file to be restored or compared.

3. Specify the start address for the data that is to be restored or compared.
4. Start playing the cassette tape.
5. Within 5 seconds, press SHIFT and PLAY/SET to restore data or VER to compare data.

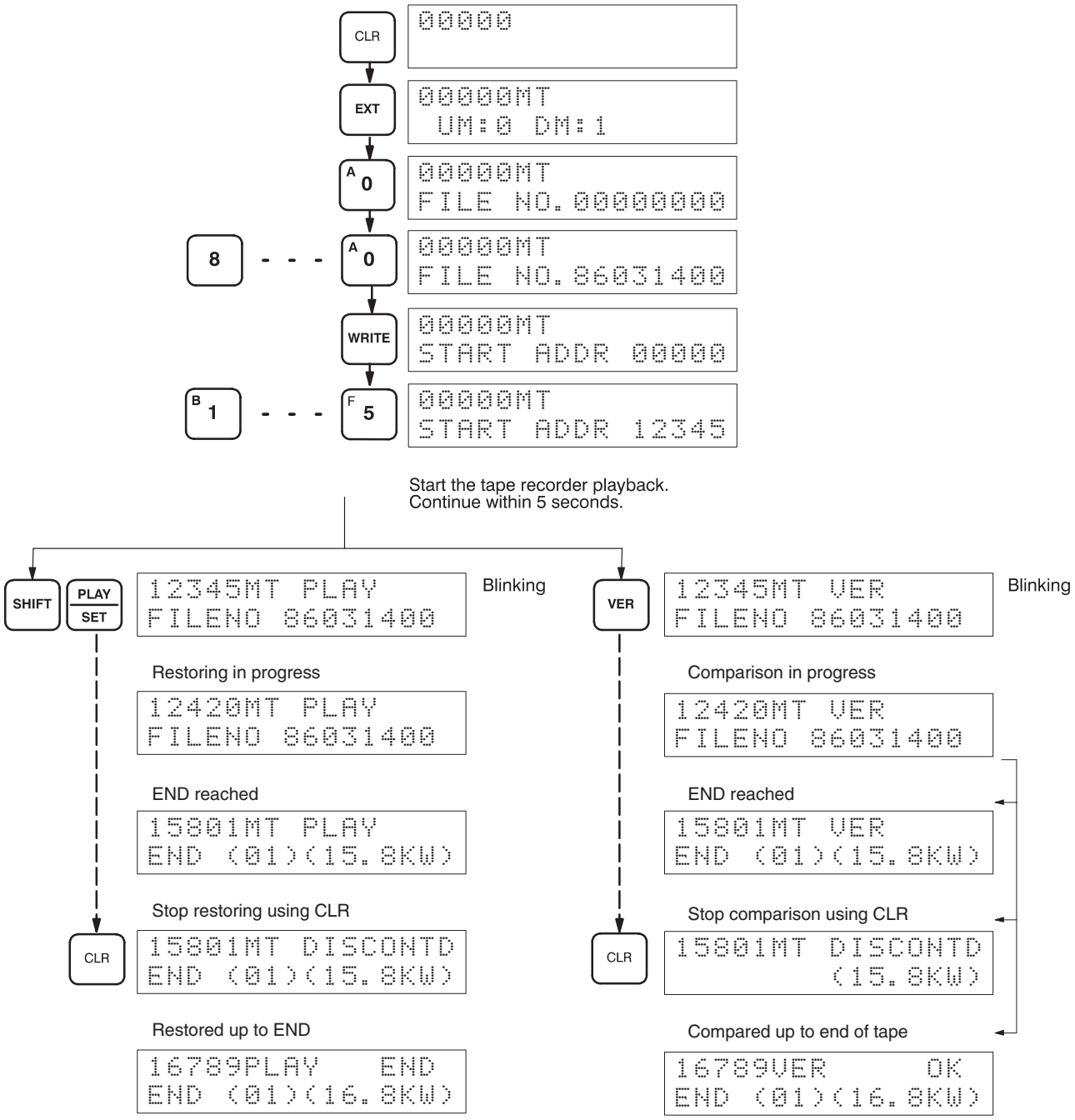
Program restoration or comparison continues until END(01) is reached or until the tape is finished, at which time the program size in Kwords is displayed. At that time the program size in Kwords is displayed. Even if END(01) is reached before the end of the tape, the restoring or comparison operation will continue through the end of the tape unless CLR is pressed to cancel.

To restore or compare program data recorded on two sides of a tape or on two or more tapes, begin restoring or comparing from the lowest address.

Key Sequence



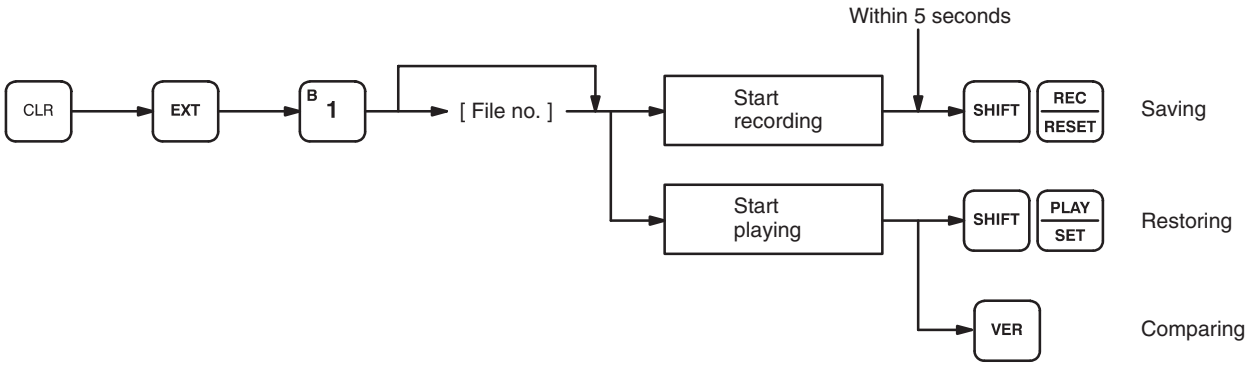
Example



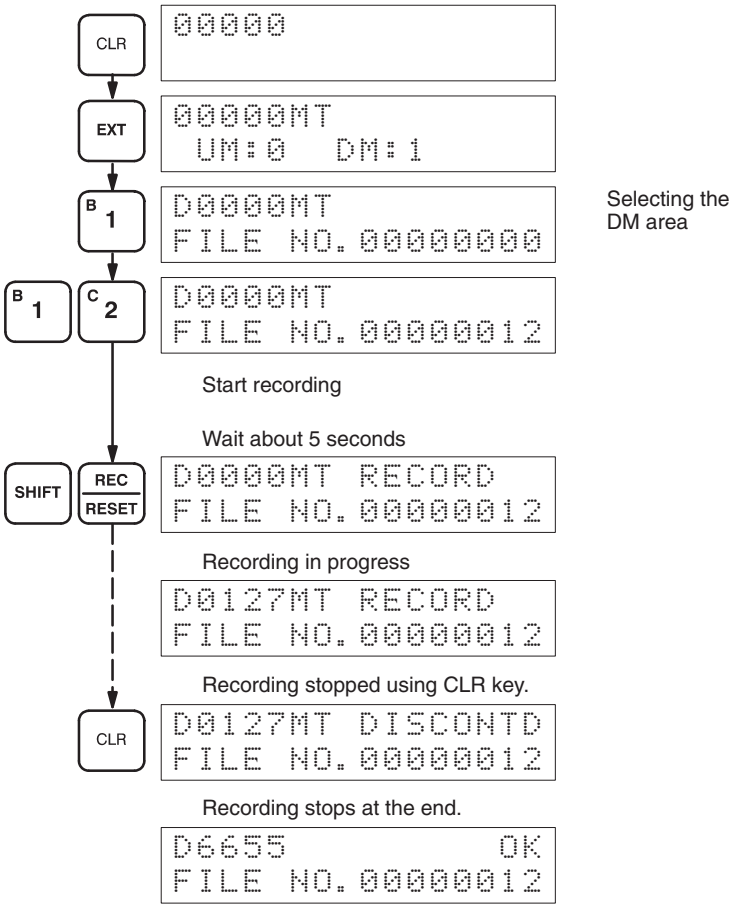
7-4-3 Saving, Restoring, and Comparing DM Data

The procedures for saving, restoring and comparing DM area data are identical to those for Program Memory except that the DM area is specified and start and stop addresses are not required. Cassette tape operations for DM area data will be continued to the end of the DM area or the end of the cassette tape unless CLR is pressed to cancel. Refer to the relevant operation in the preceding sections for details. An example for each operation is given below.

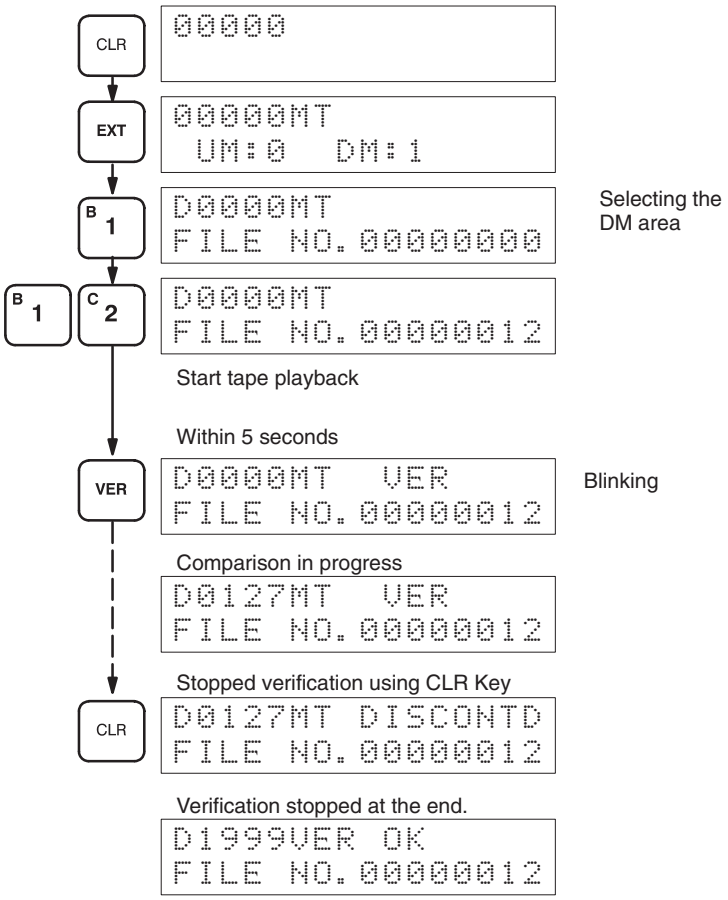
Key Sequence



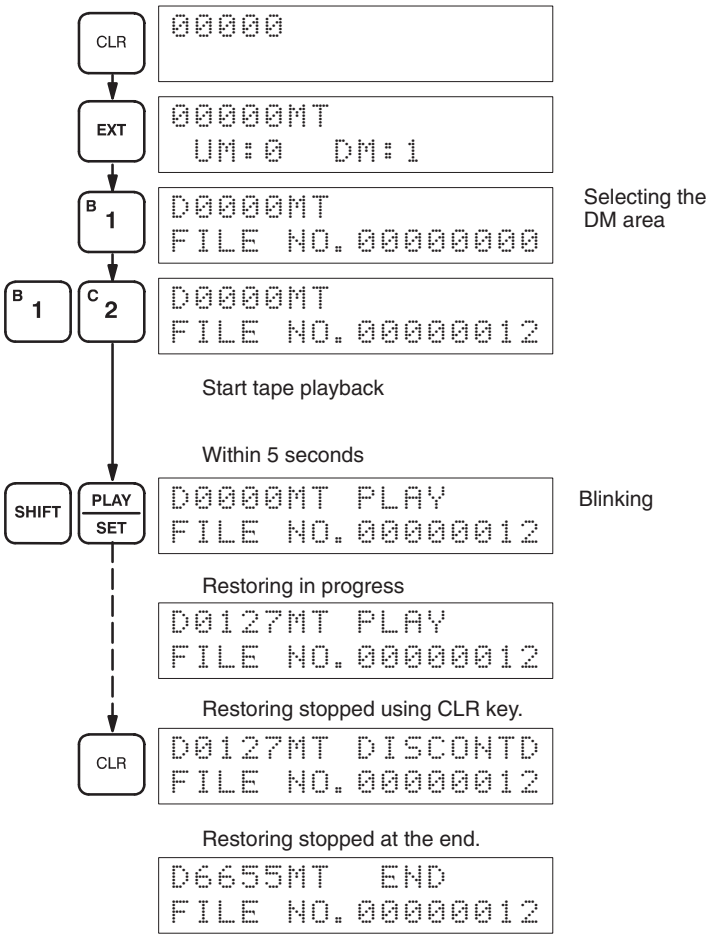
Example: Saving DM Data



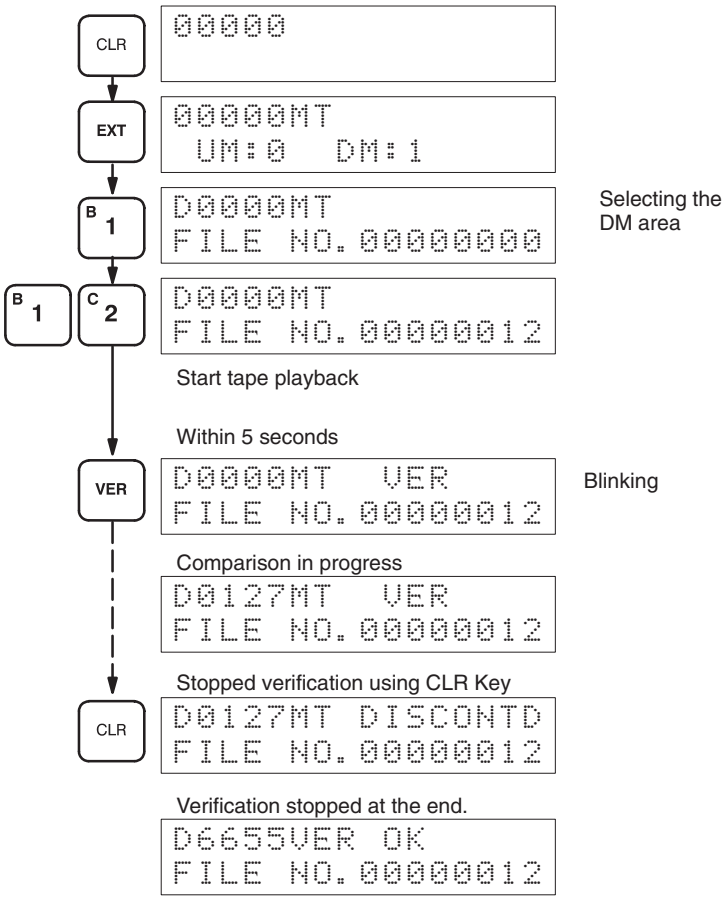
Example: Comparing DM Data



Example: Restoring DM Data



Example: Comparing DM Data



SECTION 8
Error Processing

The C1000H and C2000H provide self-diagnostic functions to identify many types of abnormal system conditions. These functions minimize downtime and enable quick, smooth error correction.

This section provides information on hardware and software errors that occur during PC operation. Program input and program syntax errors are described in *Section 4 Writing and Inputting the Program*. Although described in *Section 3 Memory Areas*, flags and other error information provided in SR and AR areas are listed in *8-5 Error Flags*.

8-1 Alarm Indicators 286
8-2 Programmed Alarms and Error Messages 286
8-3 Reading and Clearing Errors and Messages 286
8-4 Error Messages 286
8-5 Error Flags 289
8-6 Troubleshooting 290

8-1 Alarm Indicators

There are two indicators on the front of the CPU that provide visual indication of an abnormality in the PC. The error indicator (ERR) indicates fatal errors (i.e., ones that will stop PC operation); the alarm indicator (ALARM) indicates nonfatal ones. These indicators are shown in *2-1 Indicators*.

⚠ Caution The PC will turn ON the error indicator (ERR), stop program execution, and turn OFF all outputs from the PC for most hardware errors, for certain fatal software errors, or when FALS(07) is executed in the program (see tables on following pages). PC operation will continue for all other errors. It is the user's responsibility to take adequate measures to ensure that a hazardous situation will not result from automatic system shutdown for fatal errors and to ensure that proper actions are taken for errors for which the system is not automatically shut down. System flags and other system and/or user-programmed error indications can be used to program proper actions.

8-2 Programmed Alarms and Error Messages

FAL(06), FALS(07), and MSG(46) can be used in the program to provide user-programmed information on error conditions. With these three instructions, the user can tailor error diagnosis to aid in troubleshooting.

FAL(06) is used with a FAL number other than 00, which is output to the SR area when FAL(06) is executed. Executing FAL(06) will not stop PC operation or directly affect any outputs from the PC.

FALS(07) is also used with a FAL number, which is output to the same location in the SR area when FALS(07) is executed. Executing FALS(07) will stop PC operation and will cause all outputs from the PC to be turned OFF.

When FAL(06) is executed with a function number of 00, the current FAL number contained in the SR area is cleared and replaced by another, if more have been stored in memory by the system.

When MSG(46) is used, a message containing specified data area words is displayed onto the Programming Console or another Programming Device.

The use of these instructions is described in detail in *Section 5 Instruction Set*.

8-3 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 *7-3 Monitoring Operation and Modifying Data*. Procedures for the GPC, LSS, and FIT are provided in the relevant *Operation Manuals*.

8-4 Error Messages

There are basically three types of errors for which messages are displayed: initialization errors, non-fatal operating errors, and fatal operating errors.

Most of these are also indicated by FAL number being transferred to the FAL area of the SR area.

The type of error can be quickly determined from the indicators on the CPU, as described below for the three types of errors. If the status of an indicator is not mentioned in the description, it makes no difference whether it is lit or not.

After eliminating the cause of an error, clear the error message from memory before resuming operation.

Asterisks in the error messages in the following tables indicate variable numeric data. An actual number would appear on the display.

Initialization Errors

The following error messages appear before program execution has been started. The POWER indicator will be lit and the RUN indicator will not be lit for either of these. The RUN output will be OFF for each of these errors.

Error and message	FAL no.	Probable cause	Possible remedy
Waiting for start input CPUWAIT G	None	Start input on CPU Power Unit is OFF.	Short start input terminals on CPU Power Unit.
Waiting for Remote I/O CPUWAIT G	None	Power to Remote I/O Unit is off or terminator cannot be found or more than one terminator has been set.	Check power supply to Remote I/O Units, connections between Remote I/O Units, and terminator setting.

Non-fatal Operating Errors

The following error messages appear for errors that occur after program execution has been started. PC operation and program execution will continue after one or more of these errors have occurred. For each of these errors, the POWER, RUN, and ALARM indicators will be lit and the ERR indicator will not be lit. The RUN output will be ON.

Error and message	FAL no.	Probable cause	Possible correction
FAL error SYS FAIL FAL**	01 to 99	FAL(06) has been executed in program. Check the FAL number to determine conditions that would cause execution (set by user).	Correct according to cause indicated by FAL number (set by user).
Cycle time overrun SCAN TIME OVER	F8	Watchdog timer has exceeded 100 ms.	Program cycle time is longer than recommended. Reduce cycle time if possible.
I/O table verification error I/O VER ERR	E7	Unit has been removed making I/O table incorrect.	Use I/O Table Verify Operation to check I/O table and either connect dummy Units or register the I/O table again.
Remote I/O error RMTE I/O ERR Remote I/O Master Unit	B0 to B7	Error occurred in transmissions between Remote I/O Units.	Check transmission line between PC and Master and between Remote I/O Units.

Error and message	FAL no.	Probable cause	Possible correction
Duplex system error DPL ERR	D0	Error has occurred in Duplex System operation.	Check Duplex System settings.
Battery error BATT LOW	F7	Backup battery is missing or its voltage has dropped.	Check battery and replace if necessary.

Fatal Operating Errors

The following error messages appear for errors that occur after program execution has been started. PC operation and program execution will stop and all outputs from the PC will be turned OFF when any of the following errors occur. All CPU indicators will not be lit for the power interruption error. For all other fatal operating errors, the POWER, and ERR indicators will be lit. The RUN output will be OFF.

Error and message	FAL no.	Probable cause	Possible correction
Power interruption No message	None	Power has been interrupted for at least 10 ms.	Check power supply voltage and power lines. Try to power-up again.
Power interruption/fault No message	None	Power has been interrupted at an Expansion Rack. A fault has occurred in a Power Supply Unit.	Check power supply voltage and power lines. Replace the Power Supply Unit.
CPU error No message	None	Watchdog timer has exceeded maximum setting (default setting: 130 ms).	Restart system in PROGRAM mode and check program. Reduce cycle time or reset watchdog timer if longer time required. (Consider effects of longer cycle time before resetting.)
Memory error MEMORY ERR	F1	Memory Unit is incorrectly mounted or missing, or parity error has occurred.	Check Memory Unit to make sure it is mounted and backed up properly. Perform a Program Check Operation to locate cause of error. If error not correctable, try inputting program again.
No END(01) instruction NO END INST	F0	END(01) is not written anywhere in program.	Write END(01) at the final address of the program.
I/O bus error I/O BUS ERR Rack no.	C0 to C7	Error has occurred in the bus line between the CPU and I/O Units.	The rightmost digit of the FAL number will indicate the number of the Rack where the error was detected. Check cable connections between the I/O Units and Racks.
Too many Units I/O UNIT OVER	E1	Maximum number of I/O points exceeded when I/O Table Registration operation was performed.	Check the number of points with I/O Table Read. If necessary, reduce number of Units in the system to keep within maximum number of I/O points and register the I/O table again.

Error and message	FAL no.	Probable cause	Possible correction
Input-output I/O table error I/O SET ERROR	E0	Input and output word designations registered in I/O table do no agree with input/output words required by Units actually mounted.	Check the I/O table with I/O Table Verification operation and check all Units to see that they are in correct configuration. When the system has been confirmed, register the I/O table again.
FALS error SYS FAIL FAL**	0 to 99 or 9F	FALS has been executed by the program. Check the FAL number to determine conditions that would cause execution (Set by user or by system).	Correct according to cause indicated by FAL number. If FAL number is 9F, check watchdog timer and cycle time, which may be to long. 9F will be output when FALS(07) is executed and the cycle time is between 120 and 130 ms.

Other Error Messages

A number of other error messages are detailed within this manual. Errors in program input and debugging can be examined in *Sections 4-6-2 and 4-6-3* and errors in cassette tape operation are detailed in *Section 7-4.*

8-5 Error Flags

The following table lists the flags and other information provided in the SR and AR areas that can be used in troubleshooting. Details are provided in *3-3 SR Area* and *3-4 AR Area.*

SR Area

Address(es)	Function
24700 to 25015	PC Link Unit Run and Error Flags
25100 to 25115	Remote I/O Error Flags
25203	SYSMAC NET Link System SEND(90)/RECV(98) Error Flag
25206	Rack-mounting Host Link Unit Level 1 Error Flag
25208	CPU-mounting Host Link Unit Error Flag
25300 to 25307	FAL number output area.
25308	Low Battery Flag (CPU or File Memory)
25309	Cycle Time Error Flag
25310	I/O Verification Error Flag
25311	Rack-mounting Host Link Unit Level 0 Error Flag
25312	Remote I/O Error Flag
25408	Duplex System Bus Error Flag
25410	Duplex System CPU Error Flag
25411	Duplex System Memory Error Flag
25503	Instruction Execution Error (ER) Flag

AR Area

Address(es)	Function
1901	FM Data Transfer Flag
1903	FM Blocks Different Error Flag
1904	FM Write-protected Error Flag
1905	Unsuccessful FM Write Flag
1906	FM Checksum Error Flag
1907	File Memory Unit Low Battery Flag
2400 to 2403	Leftmost digit of FALS-generating address (AR 25 contains the other four digits)
2500 to 2515	Rightmost four digits of FALS-generating address (AR 2400 to AR 2403 contain the fifth digit)

8-6 Troubleshooting

The following tables provide basic hardware and software troubleshooting information.

CPU

Symptom	Possible cause	Correction
Power Supply does not turn on.	Voltage selector terminal setting error	Connect the voltage selector terminal correctly.
	Fuse is blown.	Replace fuse.
Fuse blows repeatedly.	Voltage selector terminal setting error	Connect the voltage selector terminal correctly.
	Circuit board is short-circuited, or burnt.	Replace CPU Rack Power Supply Unit or Backplane.
RUN indicator does not light.	Start input is OFF.	Turn the start input ON.
	Programming error	Correct the program.
	Power line is defective.	Replace CPU Power Supply Unit.
RUN output does not turn ON.	Power circuit is defective.	Replace CPU Power Supply Unit.
I/Os following a particular I/O number do not operate.	I/O bus is defective.	Replace Backplane.
Abnormal I/Os on Expansion I/O Rack are in groups of 8.	I/O Connecting Cable is defective. (Cable wiring is broken.)	Replace the I/O Connecting Cable.
	I/O bus is defective.	Replace Backplane.
One I/O turns ON erroneously.	I/O bus is defective.	Replace Backplane.
All I/Os of a particular I/O Unit do not operate.	I/O bus is defective.	Replace Backplane.

Input Units

Symptom	Possible cause	Correction
All inputs do not turn ON, and indicators do not light.	External input voltage is not supplied.	Supply power.
	External input voltage is low.	Raise supply voltage.
	Terminal screws are loose.	Tighten terminal screws.
	Faulty contact of terminal block connector	Replace terminal block connector.
All inputs do not turn ON, but indicators are lit.	Input circuit is defective.	Replace defective Input Unit.
All inputs do not turn OFF.	Input circuit is defective.	Replace defective Input Unit.
One input point does not turn ON.	Input device is defective.	Replace input device.
	Input wiring is broken.	Check and replace input wiring.
	Terminal screws are loose.	Tighten terminal screws.
	Faulty contact of terminal block connector	Replace terminal block connector.
	Input ON-time is too short.	Adjust field input device.
	Input circuit is defective.	Replace defective Input Unit.
	Input bit is incorrectly programmed in OUTPUT instruction.	Correct the program.
One input point does not turn OFF.	Input circuit is defective.	Replace defective Input Unit.
	Input bit is incorrectly programmed in OUTPUT instruction.	Correct the program.
Inputs turn ON and OFF irregularly.	External input voltage is low.	Raise external voltage.
	Malfunction due to noise	Countermeasures against noise: Install surge suppressor. Install insulating transformer. Wire with shielded cable.
	Terminal screws are loose.	Tighten terminal screws.
Inputs turn ON and OFF irregularly.	Faulty contact of terminal block connector	Replace terminal block connector.
Abnormal input numbers are in groups of 8 bits.	Common terminal screws are loose.	Tighten common terminal screws.
	Faulty contact of terminal block connector	Replace terminal block connector.
	Data bus is faulty.	Replace defective Unit.
	CPU is defective.	Replace CPU.
Input operation indicator does not light.	Indicator is defective.	Replace defective Unit.

Output Units

Symptom	Possible cause	Correction
All Outputs do not turn ON.	Power is not supplied to loads.	Supply power.
		Raise supply voltage.
	Terminal screws are loose.	Tighten terminal screws.
	Faulty contact of terminal block connector	Replace terminal block connector.
	Fuse is blown.	Replace fuse.
	Faulty contact of I/O bus connector	Replace defective Unit.
	Output circuit is defective.	Replace defective Unit.
All Outputs do not turn OFF.	Output circuit is defective.	Replace defective Unit.
One output does not turn ON, and indicator does not light.	Output ON-time is too short.	Correct the program.
	The same output bit is control by two different instructions (duplication).	Correct the program.
	Output circuit is defective.	Replace defective Unit.
One output does not turn ON, but indicator is lit.	Output device is defective.	Replace output device.
	Output wiring is broken.	Check output wiring.
	Terminal screws are loose.	Tighten terminal screws.
	Faulty contact of terminal block connector	Replace terminal block connector.
	Output relay is defective.	Replace defective relay.
	Output circuit is defective.	Replace defective Unit.
One output does not turn OFF, but indicator does not light.	Output relay is defective.	Replace defective relay.
	Leakage current, or residual voltage	Replace external load or add dummy resistor.
One output does not turn OFF, and indicator is lit.	The same output bit is control by two different instructions (duplication).	Correct the program.
	Output circuit is defective.	Replace defective Unit.
Outputs turn ON and OFF irregularly.	Supply voltage for external load is low.	Raise external supply voltage.
	The same output bit is control by two different instructions (duplication).	Correct the program.
	Malfunction due to noise	Countermeasures against noise; Install surge suppressor. Install insulating transformer. Wire with shielded cable.
	Terminal screws are loose.	Tighten terminal screws.
	Faulty contact of terminal block connector	Replace terminal block connector.
Abnormal output points are in groups of 8.	Common terminal screws are loose.	Tighten common terminal screws.
	Faulty contact of terminal block connector.	Replace terminal block connector.
	Fuse is blown.	Replace fuse.
	Data bus is faulty.	Replace CPU.
	CPU is defective.	
Output indicator does not light.	Indicator is defective.	Replace defective Unit.

Appendix A
Standard Models

CPU Backplane

Name	Remarks			Model
Backplane	C1000H	9 I/O slots (see note)	6 Link slots	C500-BC091
		8 I/O slots	3 Link slots	3G2A5-BC081
			5 Link slots	C500-BC082
		6 I/O slots	5 Link slots	C500-BC061
		5 I/O slots	3 Link slots	3G2A5-BC051
			5 Link slots	C500-BC052
		3 I/O slots	3 Link slots	C500-BC031
	C2000H Simplex	6 I/O slots		3G2C5-BC061
	C2000H Duplex			3G2C5-BC001
CPU	C1000H			C1000H-CPU01-EV1
	C2000H			C2000H-CPU01-EV1
RAM Unit	8K words			C2000-MR831-V2
	16K words			C2000-MR141-V2
	24K words			C2000-MR241-V2
ROM Unit	32K words			C2000-MP341-V1
EPROM Chip	27128 150 ns, Write voltage 12.5 V			ROM-ID-B
Duplex Unit	For C2000H duplex system			3G2C5-DPL01-E
CPU Power Supply	100 to 120/200 to	Output: 7 A 5 VDC		3G2A5-PS221-E
	240 VAC (selectable)	Output: 12 A 5 VDC		3G2A5-PS223-E
	24 VDC	Output: 7 A max. 5 VDC		3G2A5-PS211-E
		Output: 9 A max. 5 VDC		C500-PS213-E
I/O Control Unit	Required to connect Expansion I/O Racks			3G2A5-II101
File Memory Unit	RAM, 1K blocks			C1000H-FMR11
	RAM, 2K blocks			C1000H-FMR21

Note *The rightmost slot is only for Link Units.

I/O Backplane (for C2000H Duplex System)

Name	Remarks		Model
I/O Backplane	---		3G2C5-BI082
I/O Backplane Power Supply	100 to 120/200 to 240 VAC (selectable)	Output: 7 A 5 VDC	3G2A5-PS222-E
	24 VDC	Output: 7 A 5 VDC	3G2A5-PS212-E
I/O Control Unit	---		3G2A5-II101
File Memory Unit	RAM type, 1K blocks		C1000H-FMR11
	RAM type, 2K blocks		C1000H-FMR21

Expansion I/O Backplane

Name	Remarks		Model
Expansion I/O Backplane	For C2000H, 8 slots, w/I/O on-line exchange function		3G2C5-BI083
	8 slots		3G2A5-BI081
	5 slots		3G2A5-BI051
Power Supply	100 to 120/200 to 240 VAC (selectable)	Output: 7 A 5 VDC	3G2A5-PS222-E
	24 VDC	Output: 7 A 5 VDC	3G2A5-PS212-E
I/O Interface Unit	---		3G2A5-II002
I/O Connecting Cable	Vertical type	30 cm	C500-CN312N
		50 cm	C500-CN512N
		80 cm	C500-CN812N
		1 m	C500-CN122N
		2 m	C500-CN222N

For I/O Unit On-line Exchange (C2000H)

Name	Remarks	Model
I/O Remove Auxiliary Unit	For CPU Rack in Simplex System and I/O Rack in Duplex System	3G2C5-IOD01
	For Expansion I/O Rack	3G2C5-IOD02
Connecting Cable	35 cm	C2000-CN313
	55 cm	C2000-CN513
	85 cm	C2000-CN813
	105 cm	C2000-CN123
	205 cm	C2000-CN223

I/O Units

Name		Remarks			Model
Input Unit	DC	16 mA 5 to 12 VDC, 8 points/common, 2 circuits		16 pts	3G2A5-ID112
		10 mA 12 to 24 VDC, 8 points/common, 2 circuits		16 pts	3G2A5-ID213
		10 mA 12 to 24 VDC, 8 points/ common, 4 circuits	ON response time: 15 ms max.	32 pts	3G2A5-ID215
			ON response time: 1.5 ms	32 pts	3G2A5-ID218
		10 mA 12 to 24 VDC, 8 points/common, 4 circuits		32 pts	3G2A5-ID218CN
		7 mA 12 VDC, static, 8 points/common, 8 circuits		64 pts	3G2A5-ID114
		10 mA 12 to 24 VDC, dynamic		64 pts	3G2A5-ID212
		7 mA 24 VDC, static, 8 points/common, 8 circuits		64 pts	3G2A5-ID219
	Interrupt Input Unit	13 mA 12 to 24 VDC (sep. commons)		8 pts	3G2A5-ID216
	AC	10 mA 100 to 120 VAC, 8 points/common, 2 circuits		16 pts	3G2A5-IA121
		10 mA 200 to 240 VAC, 8 points/common, 2 circuits		16 pts	3G2A5-IA222
		10 mA 100 to 120 VAC, 8 points/common, 4 circuits		32 pts	3G2A5-IA122
		10 mA 200 to 240 VAC, 8 points/common, 4 circuits		32 pts	3G2A5-IA223
	AC/DC	10 mA 12 to 24 VAC/DC, 8 points/common, 2 circuits		16 pts	3G2A5-IM211
		10 mA 12 to 24 VAC/DC, 8 points/common, 4 circuits		32 pts	3G2A5-IM212
	TTL	3.5 mA 5 VDC, 8 points/common, 4 circuits		32 pts	3G2A5-ID501CN
Output Unit	Contact	2 A 250 VAC/24 VDC, 8 points/common, 2 circuits		16 pts	3G2A5-OC221
		2 A 250 VAC/24 VDC (sep. commons)		16 pts	3G2A5-OC223
		2 A 250 VAC/24 VDC, 8 points/common, 4 circuits		32 pts	3G2A5-OC224-E
	Transistor	1 A 12 to 24 VDC, 8 points/common, 2 circuits		16 pts	3G2A5-OD217
		1 A 12 to 48 VDC, 16 points/common, 1 circuit		16 pts	3G2A5-OD411
		50 mA 24 VDC (sep. commons)		16 pts	3G2A5-OD215
		0.3 A 12 to 24 VDC, 16 points/common, 2 circuits		32 pts	3G2A5-OD218
		2.1 A 12 to 24 VDC, 8 points/common, 2 circuits		16 pts	C500-OD219
		0.3 A 12 to 48 VDC, 16 points/common, 2 circuits		32 pts	3G2A5-OD414
		0.3 A 12 to 48 VDC, 32 points/common, 1 circuit		32 pts	3G2A5-OD412
		0.3 A 12 to 24 VDC, PNP output, 16 points/common, 2 circuits		32 pts	3G2A5-OD212
		0.3 A 12 to 48 VDC I/O relay terminal can be connected. 16 points/common, 2 circuits		32 pts	3G2A5-OD415CN
		0.1 A 24 VDC, dynamic		64 pts	3G2A5-OD211
		0.1 A 24 VDC, static, 8 points/common, 8 circuits		64 pts	3G2A5-OD213
	Triac	1 A 132 VAC max., 8 points/common, 2 circuits		16 pts	3G2A5-OA121
		1 A 250 VAC max., 8 points/common, 2 circuits		16 pts	3G2A5-OA222
		1 A 250 VAC max., 8 points/common, 3 circuits		24 pts	3G2A5-OA223
		1 A 250 VAC max., 8 points/common, 4 circuits		32 pts	3G2A5-OA225
	TTL	3.5 mA 5 VDC, 8 points/common, 4 circuits		32 pts	3G2A5-OD501CN
DC Input/Transistor Output Unit		12 to 24 VDC	Input: 10 mA	16 pts	3G2A5-MD211CN
			Output: 0.3 A	each	
Dummy I/O Unit		No. of I/O points is selectable		---	3G2A5-DUM01

Special I/O Units

Name	Remarks			Model
A/D Conversion Input	4 to 20 mA 1 to 5 V		2 pts	3G2A5-AD001
	0 to 10 V		2 pts	3G2A5-AD002
	0 to 5 V		2 pts	3G2A5-AD003
	–10 to 10 V		2 pts	3G2A5-AD004
	–5 to 5 V		2 pts	3G2A5-AD005
	4 to 20 mA 1 to 5 V		4 pts	3G2A5-AD006
	0 to 10 V		4 pts	3G2A5-AD007
	0 to 10 V, 0 to 20 mA		8 pts	C500-AD101
D/A Conversion Output	4 to 20 mA 1 to 5 V		2 pts	3G2A5-DA001
	0 to 10 V		2 pts	3G2A5-DA002
	0 to 5 V		2 pts	3G2A5-DA003
	–10 to 10 V		2 pts	3G2A5-DA004
	–5 to 5 V		2 pts	3G2A5-DA005
	4 to 20 mA, 1 to 5 V, 0 to 10 V		4 pts	C500-DA101
High-speed counter	6 BCD digits, 50 K cps 1 Set Value		1 pt	3G2A5-CT001
	6 BCD digits, 50 K cps 8 Set Value		1 pt	3G2A5-CT012
	4 BIN digits, 20 K cps 1 Set Value		4 pt	C500-CT041
Magnetic Card Reader	---			3G2A5-MGC01
Connecting Cable	---			3G2A9-CN521
Card Reader	---			3S4YR-MAW2C-04
Card	---			3G2A9-MCD01
PID	---			3G2A5-PID01-E
Position Control	1-axis, for stepping/servo motor			3G2A5-NC103-E
	1-axis, for servo motor			3G2A5-NC111-EV1
	2-axis, for servo motor			C500-NC222-E
	Stepping Motor Driver	Phase current: 0.5 to 2 A		3G2A5-SMD21
		Phase current: 0.6 to 2 A		3G2A5-SMD41
	Encoder Adapter			3G2A5-AE001
	Teaching Box	---		3G2A5-TU001-E
				3G2A5-TU002-E
	Connecting Cable for TU002	For NC211-E	2 m	C200H-CN222
			4 m	C200H-CN422
		For NC103-E/111-EV1/121	4 m	C500-CN422
	External Display			C500-ND201
	Adapter Box			3G2A5-IF101
	Power Supply			3G2A5-PS103
ASCII Unit	RAM + EEPROM			C500-ASC04
Ladder Program I/O	---			C500-LDP01-V1
Cam Positioner	---			C500-CP131

Name	Remarks	Model
ID Sensor	---	C500-IDS01-V2
	For long distance (3G2A5-ID02-E is required)	C500-IDS02-V1
	ID Adapter	C500-IDA02
	R/W Head	V600-H06
	Data Carrier	V600-D2KR01

Link Units and Remote I/O Units

Name		Remarks		Model
Host Link	Rack-mounting	APF/PCF		3G2A5-LK101-PEV1
		PCF		3G2A5-LK101-EV1
		RS-232C/RS-422		3G2A5-LK201-EV1
		APF/PCF		C500-LK103-P
		PCF		C500-LK103
		RS-232C/RS-422		C500-LK203
	CPU-mounting	APF/PCF		3G2A6-LK101-PEV1
		PCF		3G2A6-LK101-EV1
		RS-232C		3G2A6-LK201-EV1
		RS-422		3G2A6-LK202-EV1
PC Link		Links up to 32 PCs		C500-LK009-V1
SYSMAC Net		General-purpose		C500-SNT31-V4
SYSMAC Link		Use optical fiber cable		C1000H-SLK11
		Only for C1000H and C2000H Simplex		C1000H-SLK21-V1
Optical Remote I/O Master		APF/PCF		3G2A5-RM001-PEV1
		PCF		3G2A5-RM001-EV1
Optical Remote I/O Slave		APF/PCF	w/1 optical connector	3G2A5-RT001-PEV1
			w/2 optical connectors	3G2A5-RT002-PEV1
		PCF	w/1 optical connector	3G2A5-RT001-EV1
			w/2 optical connectors	3G2A5-RT002-EV1
Optical I/O Link		APF/PCF		3G2A5-LK010-PE
		PCF		3G2A5-LK010-E
Wired Remote I/O Master		---		C500-RM201
Wired Remote I/O Slave		---		C500-RT201
Remote Terminal		Input	Specify 12 VDC or 24 VDC	G71-IC16
		Output		G71-OD16
Input Block	AC Input	Specify 100 VAC or 200 VAC		G7TC-IA16
	DC Input	Specify 12 VDC or 24 VDC		G7TC-ID16
Output Block	Output	Specify 12 VDC or 24 VDC		G7TC-OC16

Link Units and Remote I/O Units (Continued)

Name	Remarks				Model
Optical Transmitting I/O	DC Input	No-voltage	8 pts	APF/PCF	3G5A2-ID001-PE
		contact, 100 VAC		PCF	3G5A2-ID001-E
	AC/DC Input	12 to 24 VAC/DC	8 pts	APF/PCF	3G5A2-IM211-PE
		100 VAC		PCF	3G5A2-IM211-E
	AC Input	100 VAC	8 pts	APF/PCF	3G5A2-IA121-PE
		100 VAC		PCF	3G5A2-IA121-E
		200 VAC	8 pts	APF/PCF	3G5A2-IA221-PE
		100 VAC		PCF	3G5A2-IA221-E
	Contact output	2 A 250 VAC/24 VDC	8 pts	APF/PCF	3G5A2-OC221-PE
		100/200 VAC		PCF	3G5A2-OC221-E
	Triac Output	100/200 VAC	8 pts	APF/PCF	3G5A2-OA222-PE
		100/200 VAC		PCF	3G5A2-OA222-E
	Transistor	0.3 A 12 to 48 VDC	8 pts	APF/PCF	3G5A2-OD411-PE
	output	100/200 VAC		PCF	3G5A2-OD411-E

SYSBUS

Name	Remarks	Model
Link Adapter	RS-422, 3 pcs	3G2A9-AL001
	Optical (APF/PCF), 3pcs	3G2A9-AL002-PE
	Optical (PCF), 3pcs	3G2A9-AL002-E
	Optical (APF/PCF), RS-422, RS-232C, 1 pc each	3G2A9-AL004-PE
	Optical (PCF), RS-422, RS-232C, 1 pc each	3G2A9-AL004-E
	Optical (APF/PCF), optical (AGF), 1 pc each	3G2A9-AL005-PE
	Optical (PCF), optical (AGF), 1 pc each	3G2A9-AL005-E
	Optical (APF/PCF), optical (AGF), 2 pcs each	3G2A9-AL006-PE
	Optical (PCF), optical (AGF), 2 pcs each	3G2A9-AL006-E
	Optical (APF/PCF), 1 pc, RS-485 1 pc for Wired Remote I/O system only	B500-AL007-P
Repeater	APF/PCF	3G5A2-RPT01-PE
	PCF	3G5A2-RPT01-E

All Plastic Optical Fiber Cable (APF)

Name	Remarks	Model
Plastic Optical Fiber Cable	Cable only, 5 to 100 m in multiples of 5 meters or multiples of 200 or 500m	3G5A2-PF002
Optical Connector A	2 pcs (brown), for plastic optical fiber 10 m long max.	3G5A2-CO001
Optical Connector B	2 pcs (black) for plastic optical fiber 8 to 20 m long	3G5A2-CO002
Plastic Optical Fiber Cable	1 m, w/optical connector A provided at both ends	3G5A2-PF101

Plastic-Clad Optical Fiber Cable (PCF)

Name	Remarks		Model
Optical Fiber Cable (indoor)	0.1 m, w/connector	Ambient temperature: –10°C to 70°C	3G5A2-OF011
	1 m, w/connector		3G5A2-OF101
	2 m, w/connector		3G5A2-OF201
	3 m, w/connector		3G5A2-OF301
	5 m, w/connector		3G5A2-OF501
	10 m, w/connector		3G5A2-OF111
	20 m, w/connector		3G5A2-OF211
	30 m, w/connector		3G5A2-OF311
	40 m, w/connector		3G5A2-OF411
	50 m, w/connector		3G5A2-OF511
Optical Fiber Cable (indoor/outdoor)	1 to 500 m (Order in Units of 1 m)	Ambient temperature: –10°C to 70°C	3G5A2-OF002
	501 to 800 m (Order in Units of 1 m)	Ambient temperature: 0°C to 55°C (Must not be subjected to direct sunlight)	

H-PCF Optical Fiber Cords and Cables with Connectors

The following diagram illustrates the model number for cables with connectors. tension members and power lines are provided in the cable. Half-lock connectors use the S3200-COCF2511 and are compatible with C200H SYSMAC LINK or SYSMAC NET Link Unit connectors. Full-lock connectors use the S3200-COCF2011 and are compatible with CV-series SYSMAC LINK or SYSMAC NET and C1000H SYSMAC LINK Link Unit connectors. Full-lock connectors cannot be used with C200H connectors.

The above connectors cannot be used with C500 SYSMAC NET Link Unit connectors, cable relays, or NSB. Refer to the *SYSMAC NET Link System Manual* for appropriate connectors for these applications.

S3200-CN□□□-□□-□□

Cable Length

201	2 m
501	5 m
102	10 m
152	15 m
202	20 m
Blank	Over 20 m*

Connector Type

20-20	Full-lock connecter on each end
20-25	One full-lock and one half-lock connector
25-25	Full lock connector on each end

*Specify lengths over 20 m separately when ordering.

Optical Connectors

Name	Model
SYSMAC NET: CV500-SNT31 SYSMAC LINK: CV500-SLK11, C1000H-SLK11 SYSMAC BUS/2: CV500-RM211/RT211	S3200-COCF2011
SYSMAC NET: C200H-SNT31 SYSMAC LINK: C200H-SLK11	S3200-COCF2511
SYSMAC NET: C500-SNT31-V4 S3200-LSU03-01E/NSB11-E S3200-NSUA1-00E/NSUG4-00E FIT10-IF401	S3200-COCH62M
SYSMAC BUS: 3G2A5-RM001-(P)EV1 3G2A5-RT001/RT002-(P)EV1 3G2A9-AL□□-(P)E	S3200-COCH82
SYSMAC NET Relay (M) Connector	S3200-COCF62M
SYSMAC NET Relay (F) Connector	S3200-COCF62F

Cable Assembly Tool and Cutter

Name	Model
Cable Assembly Tool	S3200-CAK1062

Optical Power Tester

Name	Model
SYSMAC NET: CV500-SNT31	S3200-CAT2000
SYSMAC LINK: CV500-SLK11 SYSMAC BUS/2: CV500-RM211/RT211	S3200-CAT2700
SYSMAC BUS: 3G2A5-RM001-(P)EV1 3G2A5-RT001/RT002-(P)EV1	S3200-CAT2820
SYSMAC NET: S3200-LSU03-01E FIT10-IF401	S3200-CAT3200

Optical Power Tester Head Unit

Name	Model
SYSMAC NET: CV500-SNT31	S3200-CAT2002
SYSMAC LINK: CV500-SLK11 SYSMAC BUS/2: CV500-RM211/RT211	S3200-CAT2702
SYSMAC BUS: 3G2A5-RM001-(P)EV1 3G2A5-RT001/RT002-(P)EV1	S3200-CAT2822
SYSMAC NET: S3200-LSU03-01E FIT10-IF401	S3200-CAT3202

Peripheral Devices

Name	Remarks		Model
Programming Console	Vertical, w/backlight		3G2A5-PRO13-E
	Horizontal, w/backlight		3G2A6-PRO15-E
Programming Console Connecting Cable	For connecting Programming Console, GPC, or FIT. (Only use CN221 [2 m] for Programming Console.)	2 m	3G2A2-CN221
		5 m	C500-CN523
		10 m	C500-CN131
		20 m	C500-CN231
		30 m	C500-CN331
		40 m	C500-CN431
		50 m	C500-CN531
Programming Console Adapter	For extending Programming Console. Connecting cable is separate.		3G2A5-AP001
Programming Console Base			3G2A5-BP001
Data Access Console	---		C200H-DAC01
Handheld Programming Console	---		C200H-PR027-E
Programming Console Adapter	Required for each Handheld Programming Console	---	C500-AP003
Connecting Cable		2 m	C200H-CN222
		4 m	C200H-CN422
PROM Writer	Write voltage 12.5/21 V applicable		3G2A5-PRW06
Printer Interface Unit	Memory Pack is separate.		3G2A5-PRT01-E
Memory Pack (for Printer Interface)	---		3G2C5-MP102-EV3
Printer Connecting Cable	2 m, for connecting printer		SCY-CN201
Floppy Disk Interface Unit			3G2C5-FDI03-E
Peripheral Interface Unit	Connecting cable is separate.		3G2A5-IP004-E
FIT	CPU and System Disk Set		FIT10-SET11-E
Graphic Programming Console	100 to 120 VAC, 32 K, w/comments		3G2C5-GPC03-E
	200 to 240 VAC, 32 K, w/comments		3G2C5-GPC04-E
GPC Memory Pack	w/comments for C20, P-type, C120, C500		C500-MP303-EV2
	w/comments for K-type, C200H, C1000H, C2000H		3G2C5-MP304-EV3
CRT Interface Unit	For connecting GPC to CRT		3G2A5-GD101-E
Cassette Recorder Connecting Cable	1 m		SCYPOR-PLG01
LSS	Ladder diagram programming software for PC/AT	5.25" 2D	C500-SF711-EV3
		3.5" 2DD	C500-SF312-EV3

Optional Products

Name	Remarks	Model
Battery	---	3G2A9-BAT08
Relay	24 VDC	G6B-1174P-FD-US-M
I/O Terminal Cover	For 38-pin block, special type	3G2A5-COV11
	For 38-pin block, standard	C500-COV12
	For 20-pin block, standard	C500-COV13
Connector Cover	For I/O connector	3G2A5-COV01
	For Link connector	3G2A5-COV02
	For I/O Control Unit / I/O Interface Unit connector	3G2A5-COV03
	For C2000H system simplex, CPU Connector	C2000-COV04
Space Unit	For I/O Control Unit	3G2A5-SP001
	For I/O Unit	3G2A5-SP002

Appendix B
Programming Instructions

A PC instruction is input either by inputting the corresponding Programming Console key(s) (e.g., LD, AND, OR, NOT) or by using function codes. To input an instruction via its function code, press FUN, the function code, and then WRITE. If the function code is in pointed parentheses <like this>, then SHIFT must be pressed before the above sequence. Codes requiring SHIFT are for block programming instructions.

Function Code	Name	Mnemonic	Page
--	AND	AND	46, 108
--	AND LOAD	AND LD	49, 109
--	AND NOT	AND NOT	46, 108
--	COUNTER	CNT	122
--	LOAD	LD	45, 108
--	LOAD NOT	LD NOT	45, 108
--	OR	OR	46, 108
--	OR NOT	OR NOT	46, 108
--	OR LOAD	OR LD	50, 109
--	OUTPUT	OUT	48, 109
--	OUTPUT NOT	OUT NOT	48, 109
--	TIMER	TIM	118
00	NO OPERATION	NOP	116
01	END	END	48, 107, 116
02	INTERLOCK	IL	88, 113
03	INTERLOCK CLEAR	ILC	88, 113
04	JUMP	JMP	90, 115
05	JUMP END	JME	90, 115
06	FAILURE ALARM	FAL	208
07	SEVERE FAILURE ALARM	FALS	208
08	STEP DEFINE	STEP	199
09	STEP START	SNXT	199
10	SHIFT REGISTER	SFT	127
11	KEEP	KEEP	112
12	REVERSIBLE COUNTER	CNTR	125
13	DIFFERENTIATE UP	DIFU	92, 110
14	DIFFERENTIATE DOWN	DIFD	92, 110
15	HIGH-SPEED TIMER	TIMH	121
16	WORD SHIFT	WSFT	134
20	COMPARE	CMP	142
21	MOVE	MOV	135
22	MOVE NOT	MVN	136
23	BCD-TO-BINARY	BIN	148
24	BINARY-TO-BCD	BCD	149
25	ARITHMETIC SHIFT LEFT	ASL	131
26	ARITHMETIC SHIFT RIGHT	ASR	132
27	ROTATE LEFT	ROL	132

Function Code	Name	Mnemonic	Page
28	ROTATE RIGHT	ROR	133
29	COMPLEMENT	COM	179
30	BCD ADD	ADD	160
31	BCD SUBTRACT	SUB	162
32	BCD MULTIPLY	MUL	165
33	BCD DIVIDE	DIV	167
34	AND WORD	ANDW	180
35	OR WORD	ORW	180
36	EXCLUSIVE OR	XORW	181
37	EXCLUSIVE NOR	XNRW	182
38	INCREMENT	INC	159
39	DECREMENT	DEC	159
40	SET CARRY	STC	159
41	CLEAR CARRY	CLC	159
42	FILE MEMORY READ	FILR	215
43	FILE MEMORY WRITE	FILW	216
44	EXTERNAL PROGRAM READ	FILP	216
45	TRACE MEMORY SAMPLE	TRSM	211
46	MESSAGE	MSG	209
50	BINARY ADD	ADB	174
51	BINARY SUBTRACT	SBB	176
52	BINARY MULTIPLY	MLB	178
53	BINARY DIVIDE	DVB	179
54	DOUBLE BCD ADD	ADDL	161
55	DOUBLE BCD SUBTRACT	SUBL	164
56	DOUBLE BCD MULTIPLY	MULL	166
57	DOUBLE BCD DIVIDE	DIVL	168
58	DOUBLE BCD-TO-DOUBLE BINARY	BINL	148
59	DOUBLE BINARY-TO-DOUBLE BCD	BCDL	150
67	BIT COUNTER	BCNT	210
68	BLOCK COMPARE	BCMP	145
70	BLOCK TRANSFER	XFER	138
71	BLOCK SET	BSET	136
72	SQUARE ROOT	ROOT	172
73	DATA EXCHANGE	XCHG	138
74	ONE DIGIT SHIFT LEFT	SLD	133
75	ONE DIGIT SHIFT RIGHT	SRD	134
76	4-TO-16 DECODER	MLPX	150
77	16-TO-4 ENCODER	DMPX	152
78	7-SEGMENT DECODER	SDEC	154
79	FLOATING POINT DIVIDE	FDIV	169
80	SINGLE WORD DISTRIBUTE	DIST	139
81	DATA COLLECT	COLL	139
82	MOVE BIT	MOVB	140
83	MOVE DIGIT	MOVD	141

Function Code	Name	Mnemonic	Page
84	REVERSIBLE SHIFT REGISTER	SFTR	129
85	TABLE COMPARE	TCMP	146
86	ASCII CONVERT	ASC	157
87	I/O WRITE	WRIT	218
88	I/O READ	READ	218
89	INTERRUPT CONTROL	INT	185
90	NETWORK SEND	SEND	219
91	SUBROUTINE ENTRY	SBS	183
92	SUBROUTINE DEFINE	SBN	183
93	RETURN	RET	183
94	WATCHDOG TIMER REFRESH	WDT	211
96	BLOCK PROGRAM BEGIN	BPRG	190
97	I/O REFRESH	IORF	211
98	NETWORK RECEIVE	RECV	221
<01>	BLOCK PROGRAM END	BEND	190
<02>	CONDITIONAL BRANCH	IF	191
<03>	NO BRANCH	ELSE	191
<04>	BRANCH END	IEND	191
<05>	ONE CYCLE AND WAIT	WAIT	193
<06>	CONDITIONAL BLOCK EXIT	EXIT	197
<07>	SET	SET	191
<08>	RESET	RSET	191
<09>	LOOP	LOOP	197
<10>	LOOP END	LEND	197
<11>	BLOCK PROGRAM PAUSE	BPPS	198
<12>	BLOCK PROGRAM RESTART	BPRS	198
<13>	TIMER WAIT	TIMW	195
<14>	COUNTER WAIT	CNTW	196
<15>	HIGH-SPEED TIMER WAIT	TMHW	195

Table: Instruction Execution Times

Instruction	ON execution time (μs)[1,2]		Conditions	OFF execution time (μs)[1,2]	
	C1000H	C2000H		C1000H	C2000H
LD	0.4	0.4	---	0.4	0.4
LD NOT	0.4	0.4	---	0.4	0.4
AND	0.4	0.4	---	0.4	0.4
AND NOT	0.4	0.4	---	0.4	0.4
OR	0.4	0.4	---	0.4	0.4
OR NOT	0.4	0.4	---	0.4	0.4
AND LD	0.4	0.4	---	0.4	0.4
OR LD	0.4	0.4	---	0.4	0.4
OUT	0.8	0.8	---	0.8	0.8
OUT NOT	0.8	0.8	---	0.8	0.8
TIM	2.4	2.4	Constant for SV	R 2.4 IL 2.4 JMP 2.4	
	20	13	∗DM for SV	R: 29 IL: 29 JMP: 14	R: 19 IL: 19 JMP: 10
CNT	2.4	2.4	Constant for SV	R 2.4 IL 2.4 JMP 2.4	
	16	11	∗DM for SV	R: 28 IL: 11 JMP: 11	R: 18 IL: 7 JMP: 7
NOP(00)	0.4	0.4	---	---	---
END(01)	8	5	---	---	---
IL(02)	9	6	---	8	6
ILC(03)	9	6	---	7	5
JMP(04)	10	7	---	9	6
JME(05)	10	7	---	9	6
FAL(06)	16/17	11/12	---	7/8	5/6
FAL(06) 00	11/12	7/8	---	7/8	5/6
FALS(07)	11/12	7/8	---	7	5
STEP(08)	24	16	---	15	10
SNXT(09)	10	6	---	7	5
SFT(10)	40	26	With 1-word shift register	R: 35 IL: 7 JMP: 7	R: 25 IL: 5 JMP: 5
	444	296	With 252-word shift register	R: 200 IL: 7 JMP: 7	R: 133 IL: 5 JMP: 5
KEEP(11)	0.8	0.8	---	---	---

Notes
1. The execution time is given in microseconds unless otherwise stated.
2. Times for non-differentiated forms are given to the left of the slash, and those for differentiated forms given to the right.

Instruction	ON execution time (μs)[1,2]		Conditions	OFF execution time (μs)[1,2]	
	C1000H	C2000H		C1000H	C2000H
CNTR(12)	21	14	Constant for SV	R: 15 IL: 10	R: 10 IL: 7
	29	19	*DM for SV	JMP: 10	JMP: 7
DIFU(13)	16	10	---	Normal: 15 IL: 16 JMP: 8	Normal: 10 IL: 10 JMP: 5
DIFD(14)	16	11	---	Normal: 16 IL: 16 JMP: 9	Normal: 11 IL: 11 JMP: 6
TIMH(15)	20	13	Interrupt Constant for SV	R 20 IL 21 JMP 15	
	22	15	Normal cycle		
	20	13	Interrupt *DM for SV		
	18	12	Normal cycle	R: 29 IL: 30 JMP: 16	R: 19 IL: 20 JMP: 10
WSFT(16)	36/38	24/26	When shifting 1 word	7/8	5/6
	5.59 ms	3.72 ms	When shifting 4,096 words using *DM		
CMP(20)	14	9	When comparing a constant to a word	7	5
	29	20	When comparing two *DM		
MOV(21)	15/17	10/11	When transferring a constant to a word	7/8	5/6
	30/31	20/21	When transferring *DM to *DM		
MVN(22)	16/17	10/11	When transferring a constant to a word	7/8	5/6
	30/31	20/21	When transferring *DM to *DM		
BIN (23)	21/23	14/15	When converting a word to a word	7/8	5/6
	34/35	22/23	When converting *DM to *DM		
BCD(24)	21/22	14/15	When converting a word to a word	7/8	5/6
	33/34	22/23	When converting *DM to *DM		
ASL(25)	18/19	12/13	When shifting a word	7/8	5/6
	24/25	16/17	When shifting *DM		
ASR(26)	18/19	12/13	When shifting a word	7/8	5/6
	24/25	16/17	When shifting *DM		
ROL(27)	18/19	12/13	When rotating a word	7/8	5/6
	24/25	16/17	When rotating *DM		
ROR(28)	18/19	12/13	When rotating a word	7/8	5/6
	24/25	16/17	When rotating *DM		
COM(29)	15/17	10/11	When inverting a word	7/8	5/6
	21/23	14/15	When inverting *DM		

Notes
1. The execution time is given in microseconds unless otherwise stated.
2. Times for non-differentiated forms are given to the left of the slash, and those for differentiated forms given to the right.

Instruction	ON execution time (μs)[1,2]		Conditions	OFF execution time (μs)[1,2]	
	C1000H	C2000H		C1000H	C2000H
ADD(30)	33/35	22/23	Constant + word → word	7/8	5/6
	53/55	35/36	*DM + *DM → *DM		
SUB(31)	33/34	22/23	Constant + word → word	7/8	5/6
	53/55	35/36	*DM – *DM → *DM		
MUL(32)	48/50	32/33	Constant x word → word	7/8	5/6
	68/70	55/56	*DM x *DM → word		
DIV(33)	63/65	42/43	Word ÷ constant → word	7/8	5/6
	84/86	56/57	*DM ÷ *DM → *DM		
ANDW(34)	19/21	13/14	Constant AND word → word	7/8	5/6
	40/41	27/28	*DM AND *DM → *DM		
ORW(35)	19/20	13/14	Constant OR word → word	7/8	5/6
	40/41	27/28	*DM OR *DM → *DM		
XORW(36)	19/21	13/14	Constant XOR word → word	7/8	5/6
	40/41	27/28	*DM XOR *DM → *DM		
XNRW(37)	20/21	13/14	Constant XNOR word → word	7/8	5/6
	40/41	27/28	*DM XNOR *DM → *DM		
INC(38)	23/25	15/16	When incrementing a word	7/8	5/6
	29/31	20/21	When incrementing *DM		
DEC(39)	22/24	15/16	When decrementing a word	7/8	5/6
	28/30	19/20	When decrementing *DM		
STC(40)	9/10	6/7	---	6/8	4/6
CLC(41)	9/10	6/7	---	6/8	4/6
FILR(42)	4.89 ms	3.26 ms	When reading 1 block	6/8	4/6
	81.9 ms	54.5 ms	When reading 20 blocks		
FILW(43)	7.21 ms	4.8 ms	When writing 1 block	6/8	4/6
	131 ms	87 ms	When writing 20 blocks		
FILP(44)	38 ms	25 ms	When reading 600 addresses	9	6
	1.66 s	1.11 s	When reading 30 addresses		
TRSM(45)	56	38	When tracing 1 point + 1 word	8	5
	93	62	When tracing 12 points + 3 words		
MSG(46)	16/17	11/12	---	7/8	5/6
ADB(50)	22/23	15/16	Constant + word → word	7/8	5/6
	42/44	28/29	*DM + *DM → *DM		
SBB(51)	22/24	15/16	Constant – word → word	7/8	5/6
	43/44	29/30	*DM – *DM → *DM		
MLB(52)	26/27	17/18	Constant x word → word	7/8	5/6
	6/48	31/32	*DM x *DM → *DM		

Notes 1. The execution time is given in microseconds unless otherwise stated.
2. Times for non-differentiated forms are given to the left of the slash, and those for differentiated forms given to the right.

Instruction	ON execution time (μs)[1,2]		Conditions	OFF execution time (μs)[1,2]	
	C1000H	C2000H		C1000H	C2000H
DVB(53)	51/52	34/35	Word ÷ constant → word	7/8	5/6
	71/73	47/48	*DM ÷ *DM → *DM		
ADDL(54)	74/76	49/50	Word + word → word	6/8	4/6
	92/93	61/62	*DM + *DM → *DM		
SUBL(55)	73/75	49/50	Word – word → word	6/8	4/6
	93/95	62/63	*DM – *DM → *DM		
MULL(56)	187/188	125/126	Word x word → word	6/8	4/6
	205/206	137/138	*DM x *DM → *DM		
DIVL(57)	192/194	128/129	Word ÷ word → word	6/8	4/6
	210/212	140/141	*DM ÷ *DM → *DM		
BINL(58)	35/37	27/28	When converting words to words	7/8	5/6
	47/49	31/32	When converting *DM to *DM		
BCDL(59)	39/40	26/27	When converting words to words	7/8	5/6
	50/52	33/34	When converting *DM to *DM		
BCNT(67)	51/53	34/35	When counting 1 word	7/8	5/6
(Bit Count)	8.2 ms	5.5 ms	When counting 4,096 words using *DM		
BCMP(68)	66/68	44/45	Comparing constant to word-designated table	7/8	5/6
	95/96	63/64	Comparing *DM → *DM-designated table		
XFER(70)	46/48	31/32	When transferring 1 word	7/8	5/6
	5.49 ms	3.66 ms	When transferring 4,096 words using *DM		
BSET(71)	35/37	23/24	When setting a constant to 1 word	7/8	5/6
	4.14 ms	2.76 ms	When setting *DM ms to 4,096 words using *DM		
ROOT(72)	99/100	66/67	When taking root of word and placing in a word	7/8	5/6
	109/110	73/74	When taking root of 99,999,999 in *DM and placing in *DM		
XCHG(73)	19/20	13/14	Between words	7/8	5/6
	31/33	21/22	Between *DM		
SLD(74)	35/37	23/24	When shifting 1 word	7/8	5/6
	6.31 ms	4.20 ms	When shifting 4,096 DM words using *DM	7/8	5/6
SRD(75)	35/37	23/24	When shifting 1 word	7/8	5/6
	6.27 ms	4.18 ms	When shifting 4,006 DM words using *DM		
MLPX(76)	26/28	17/18	When decoding word to word	7/8	5/6
	56/58	37/38	When decoding *DM to *DM		
DMPX(77)	35/36	23/24	When encoding a word to a word	7/8	5/6
	65/67	43/44	When encoding *DM to *DM		
SDEC(78)	30/31	20/21	When decoding a word to a word	7/8	5/6
	67/68	45/46	When decoding *DM to *DM		

Notes 1. The execution time is given in microseconds unless otherwise stated.
2. Times for non-differentiated forms are given to the left of the slash, and those for differentiated forms given to the right.

Instruction	ON execution time (μs)[1,2]		Conditions	OFF execution time (μs)[1,2]	
	C1000H	C2000H		C1000H	C2000H
FDIV(79)	52/54	35/36	Word ÷ word → word (equals 0)	7/8	5/6
	174/175	116/117	Word ÷ word → word (doesn't equal 0)		
	192/194	128/129	*DM ÷ *DM → *DM		
DIST(80)	25/27	17/18	Constant → word + (word)	7/8	5/6
	47/49	31/32	*DM → (*DM + (*DM))		
COLL(81)	28/30	19/20	(Word + (word)) → word	7/8	5/6
	48/50	32/33	(*DM + (*DM)) → *DM		
MOVB (82)	31/32	21/22	When transferring word to a word	7/8	5/6
	51/52	34/35	When transferring *DM to *DM		
MOVD(83)	27/28	18/19	When transferring word to a word	7/8	5/6
	47/49	31/32	When transferring *DM to *DM		
SFTR(84)	42/44	28/29	When shifting 1 word	7/8	5/6
	7.35 ms	4.90 ms	When shifting 4,096 DM words using *DM		
TCMP(85)	51/53	34/35	Comparing constant to word-designated table	7/8	5/6
	71/73	47/48	Comparing *DM → *DM-designated table		
ASC(86)	32/34	21/22	Word → word	7/8	5/6
	74/76	49/50	*DM → *DM		
WRIT(87)	1.24 ms	0.83 ms	When writing 1 word	6/8	4/6
	6.32 ms	4.21 ms	When writing 255 words		
READ(88)	1.24 ms	0.83 ms	When reading 1 word	6/8	4/6

Notes
1. The execution time is given in microseconds unless otherwise stated.
2. Times for non-differentiated forms are given to the left of the slash, and those for differentiated forms given to the right.

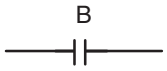
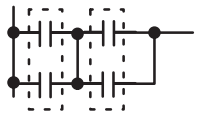
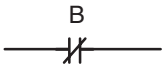
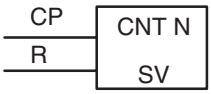
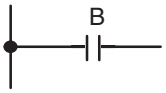
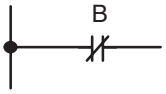
Programming Instructions

The following tables detail all of the ladder diagram programming instructions for the C1000H/C2000H PCs, and the applicable data areas for each. Bit and word addresses for each area are given in the footnotes at the bottom of the page.

Differentiated instructions (indicated with @) are entered by pressing the NOT key on the Programming Console following the function code.

The DM area can be indirectly addressed by specifying the data area as *DM, and then entering the address of the DM word that contains the actual data.

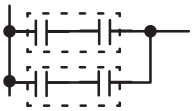
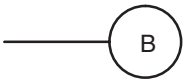
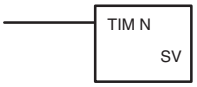
Basic Instructions

Name and Mnemonic	Symbol	Function	Operand Data Areas	Page
AND AND		Logically ANDs the status of the designated bit with the current execution condition.	**B:** IR SR HR AR LR TC	108
AND LOAD AND LD		Logically ANDs the resultant execution conditions of the preceding logic blocks.	None	109
AND NOT AND NOT		Logically ANDs the inverse of the designated bit with the current execution condition.	**B:** IR SR HR AR LR TC	108
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 #	122
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	108
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	108

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 23615	23700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	C1000H: DM 0000 to DM 4095 C2000H: DM 0000 to DM 6655	0000 to 9999 or 0000 to FFFF

Name and Mnemonic	Symbol	Function	Operand Data Areas	Page
OR OR		Logically ORs the status of the designated bit with the current execution condition.	**B**: IR SR HR AR LR TC	108
OR LOAD OR LD		Logically ORs the resultant execution conditions of the preceding logic blocks.	None	109
OR NOT OR NOT		Logically ORs the inverse of the designated bit with the execution condition.	**B:** IR SR HR AR LR TC	108
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	109
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	109
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 #	118

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 23615	23700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	C1000H: DM 0000 to DM 4095 C2000H: DM 0000 to DM 6655	0000 to 9999 or 0000 to FFFF

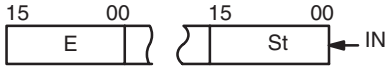
Special Instructions

Name Mnemonic	Symbol	Function	Operand Data Areas	Page
NO OPERATION NOP(00)	None	Nothing is executed and program operation moves to the next instruction.	None	116
END END(01)	END(01)	Required at the end of each program. Instructions located after END(01) will not be executed.	None	116
INTERLOCK IL(02) **INTERLOCK CLEAR** ILC(03)	IL(02) ILC(03)	If an interlock condition is OFF, all outputs and all timer PVs between the current IL(02) and the next ILC(03) are turned OFF or reset, respectively. Other instructions are treated as NOP. Counter PVs are maintained. If the execution condition is ON, execution continues normally.	None	113
JUMP JMP(04) **JUMP END** JME(05)	JMP(04) N JME(05) N	When the execution condition for the JMP(04) instruction is ON, all instructions between JMP(04) and the corresponding JME(05) are to be ignored or treated as NOP(00). For direct jumps, the corresponding JMP(04) and JME(05) instructions have the same N value in the range 01 through 99. Direct jumps are usable only once each per program (i.e., N is 01 through 99 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 99	115
FAILURE ALARM (@)FAL(06)	FAL(06) N	Assigns a failure alarm code to the given execution condition.When N can be given a value between 01 and 99 to indicate that a non-fatal error (i.e., one that will not stop the CPU) has occurred. This is indicated by the PC 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	208
SEVERE FAILURE ALARM FALS(07)	FALS(07) N	A fatal error is indicated by outputting N to the FAL output area and the CPU is stopped. The same FAL numbers are used for both FAL(06) and FALS(07).	**N:** 01 to 99	208
STEP DEFINE STEP(08)	STEP(08) B	When used with a control bit (B), defines the start of a new step and resets the previous step. When used without B, it defines the end of step execution.	**B:** IR HR AR LR	199

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

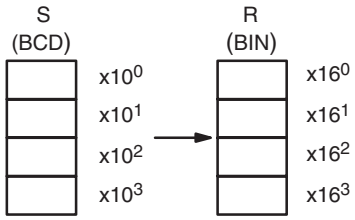
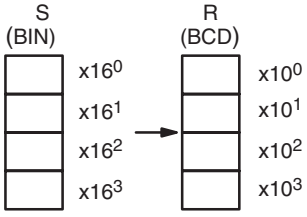
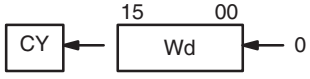
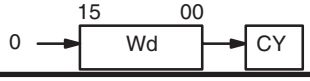
IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 23615	23700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	C1000H: DM 0000 to DM 4095 C2000H: DM 0000 to DM 6655	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas	Page
STEP START SNXT(09)	SNXT(09) B	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	199
SHIFT REGISTER SFT(10)	I P SFT(10) St R E	Creates a bit shift register for data from the starting word (St) through to the ending word (E). I: input bit; P: shift pulse; R: reset input. St must be less than or equal to E. St and E must be in the same data area. 	**St/E:** IR HR AR LR	127
KEEP KEEP(11)	S KEEP(11) R B	Defines a bit (B) as a latch, controlled by the set (S) and reset (R) inputs.	**B:** IR HR AR LR	112
REVERSIBLE COUNTER CNTR (12)	II DI CNTR(12) N R SV	Increases or decreases the PV by one whenever the increment input (II) or decrement input (DI) signals, respectively, go from OFF to ON. SV: 0 to 9999; R: reset input. Each TC bit can be used for one timer/counter only. The TC bit is entered as a constant.	**N:** TC **SV:** IR SR HR AR LR DM #	125
DIFFERENTIATE UP DIFU(13) **DIFFERENTIATE DOWN** DIFD(14)	DIFU(13) B DIFD(14) B	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	110
HIGH-SPEED TIMER TIMH(15)	TIMH(15) N SV	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 #	121
WORD SHIFT (@)WSFT(16)	WSFT(16) St E	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	134
COMPARE (@)CMP(20)	CMP(20) Cp1 Cp2	Compares the data in two 4-digit hexadecimal words (Cp1 and Cp2) and outputs result to the GR, EQ, or LE Flags.	**Cp1/Cp2:** IR SR HR AR LR TC DM #	142

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

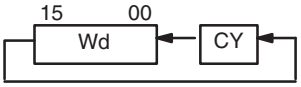
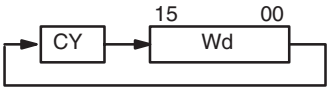
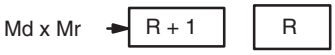
IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 23615	23700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	C1000H: DM 0000 to DM 4095 C2000H: DM 0000 to DM 6655	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas		Page
MOVE (@)MOV(21)	MOV(21) S D	Transfers data from source word, (S) to destination word (D).	**S:** IR SR HR AR LR TC DM #	**D:** IR HR AR LR DM	135
MOVE NOT (@)MVN(22)	MVN(22) S D	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	136
BCD TO BINARY (@)BIN(23)	BIN(23) S R	Converts 4-digit, BCD data in source word (S) into 16-bit binary data, and outputs converted data to result word (R). 	**S:** IR SR HR AR LR TC DM	**R:** IR HR AR LR DM	148
BINARY TO BCD (@)BCD(24)	BCD(24) S R	Converts binary data in source word (S) into BCD, and outputs converted data to result word (R). 	**S:** IR SR HR AR LR DM	**R:** IR HR AR LR DM	149
ARITHMETIC SHIFT LEFT (@)ASL(25)	ASL(25) Wd	Each bit within a single word of data (Wd) is shifted one bit to the left, with zero written to bit 00 and bit 15 moving to CY. 	**Wd:** IR HR AR LR DM		131
ARITHMETIC SHIFT RIGHT (@)ASR(26)	ASR(26) Wd	Each bit within a single word of data (Wd) is shifted one bit to the right, with zero written to bit 15 and bit 00 moving to CY. 	**Wd:** IR HR AR LR DM		132

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

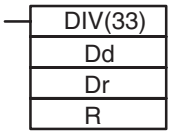
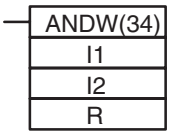
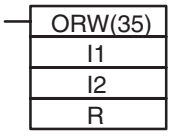
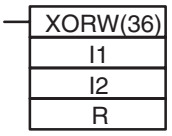
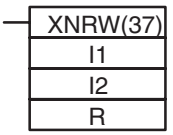
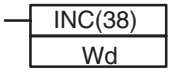
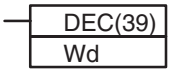
IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 23615	23700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	C1000H: DM 0000 to DM 4095 C2000H: DM 0000 to DM 6655	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas	Page
ROTATE LEFT (@)ROL(27)	ROL(27) Wd	Each bit within a single word of data (Wd) is moved one bit to the left, with bit 15 moving to carry (CY), and CY moving to bit 00. 	**Wd:** IR HR AR LR DM	132
ROTATE RIGHT (@)ROR(28)	ROR(28) Wd	Each bit within a single word of data (Wd) is moved one bit to the right, with bit 00 moving to carry (CY), and CY moving to bit 15. 	**Wd:** IR HR AR LR DM	133
COMPLEMENT (@)COM(29)	COM(29) Wd	Inverts bit status of one word (Wd) of data, changing 0s to 1s, and vice versa. 	**Wd:** IR HR AR LR DM	179
BCD ADD (@)ADD(30)	ADD(30) Au Ad R	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:** IR SR HR AR LR TC DM # **R:** IR HR AR LR DM	160
BCD SUBTRACT (@)SUB(31)	SUB(31) Mi Su R	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:** IR SR HR AR LR TC DM # **R:** IR HR AR LR DM	162
BCD MULTIPLY (@)MUL(32)	MUL(32) Md Mr R	Multiplies the 4-digit BCD multiplicand (Md) and 4-digit BCD multiplier (Mr), and outputs the result to the specified result words (R and R + 1). R and R + 1 must be in the same data area. 	**Md/ Mr:** IR SR HR AR LR TC DM # **R:** IR HR AR LR DM	165

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

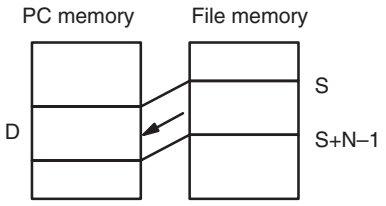
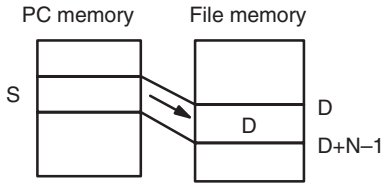
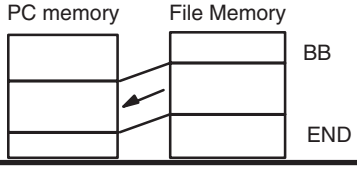
IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 23615	23700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	C1000H: DM 0000 to DM 4095 C2000H: DM 0000 to DM 6655	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas	Page
BCD DIVIDE (@)DIV(33)		Divides the 4-digit BCD dividend (Dd) by the 4-digit BCD divisor (Dr), and outputs the result to the specified result words. R receives the quotient; R + 1 receives the remainder. R and R + 1 must be in the same data area. 	**Dd/ Dr:** IR SR HR AR LR TC DM # **R:** IR HR AR LR DM	167
AND WORD (@)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	180
OR WORD (@)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	180
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	181
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	182
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	159
DECREMENT (@)DEC(39)		Decrements the value of a 4-digit BCD word by 1, without affecting carry (CY).	**Wd:** IR HR AR LR DM	159

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

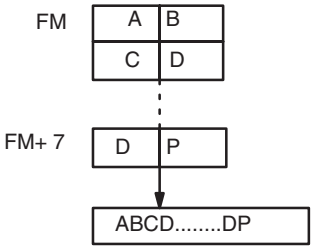
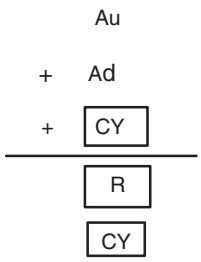
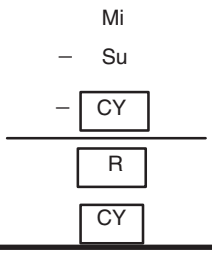
IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 23615	23700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	C1000H: DM 0000 to DM 4095 C2000H: DM 0000 to DM 6655	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas	Page
SET CARRY (@)STC(40)	STC(40)	Sets the Carry Flag (i.e., turns CY ON).	None	159
CLEAR CARRY (@)CLC(41)	CLC(41)	Clears the Carry Flag (i.e, turns CY OFF).	None	159
FILE MEMORY READ (@)FILR(42)	FILR(42) N S D	Reads data from the File Memory area in 128-word block units, and outputs data to the specified PC destination words. N gives the number of blocks to be transferred. S specifies the starting source block. D specifies the address of the starting destination word. 	**N/S:** IR, SR, HR, AR, LR, TC, DM, # **D:** IR, HR, AR, LR, TC, DM	215
FILE MEMORY WRITE (@)FILW(43)	FILW(43) N S D	Transfers data from the PC memory area to the File Memory area in 128-word (block) units. N gives the number of blocks to be transferred. S specifies the address of the starting source word. D gives the address of the starting destination block 	**N:** IR, SR, HR, AR, LR, TC, DM, # **S:** IR, SR, HR, AR, LR, TC, DM **D:** IR, HR, AR, LR, TC, DM	216
EXTERNAL PROGRAM READ (@)FILP(44)	FILP(44) BB	Copies program data from the File Memory blocks between the beginning block number (BB) and the first END(01), and transfers it to Program Memory area at the addresses immediately following FLIP(44). The transferred program is then executed. 	**BB:** IR, SR, HR, AR, LR, TC, DM, #	216

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 23615	23700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	C1000H: DM 0000 to DM 4095 C2000H: DM 0000 to DM 6655	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas	Page
TRACE MEMORY SAMPLE TRSM(45)	TRSM(45)	Initiates data tracing. Used in conjunction with flags in the AR area to simplify debugging. Parameters are set using the Address Trace operation on a Programming Console or other System Support Tool (e.g., FIT, GPC, or LSS). AR 1815 starts sampling, AR 1814 starts the recording of the samples which are written to the Trace Memory of the GPC, FIT, or LSS. A positive or negative delay can be set for the recording of the samples. AR 1813 and 1812 indicate tracing in progress and tracing complete, respectively.	None	211
DISPLAY MESSAGE (@)MSG(46)	MSG(46) FM	Displays eight words of ASCII code, starting from FM, on the Programming Console or GPC. All eight words must be in the same data area. 	**FM:** IR HR AR LR TC DM #	209
BINARY ADD (@)ADB(50)	ADB(50) Au Ad R	Adds the 4-digit augend (Au), 4-digit addend (Ad), and content of CY and outputs the result to the specified result word (R). 	**Au/ Ad:** IR SR HR AR LR TC DM # **R:** IR HR AR LR DM	174
BINARY SUBTRACT (@)SBB(51)	SBB(51) Mi Su R	Subtracts the 4-digit hexadecimal subtrahend (Su) and content of carry, from the 4-digit hexadecimal minuend (Mi), and outputs the result to the specified result word (R). 	**Mi/ Su:** IR SR HR AR LR TC DM # **R:** IR HR AR LR DM	176

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

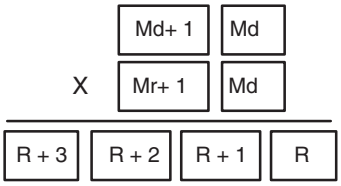
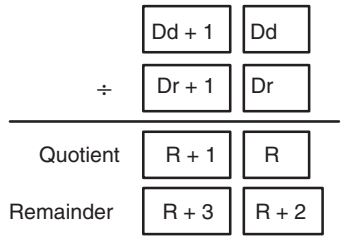
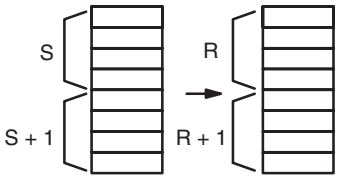
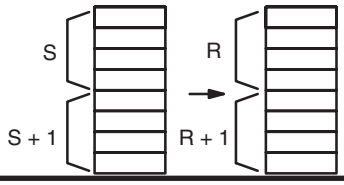
IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 23615	23700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	C1000H: DM 0000 to DM 4095 C2000H: DM 0000 to DM 6655	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas	Page
BINARY MULTIPLY (@)MLB(52)	MLB(52) Md Mr R	Multiplies the 4-digit hexadecimal multiplicand (Md) and 4-digit multiplier (Mr), and outputs the 8-digit hexadecimal result to the specified result words (R and R+1). R and R+1 must be in the same data area. Md **X** Mr Quotient R Remainder R+1	**Md/ Mr:** IR, SR, HR, AR, LR, TC, DM, # **R:** IR, HR, AR, LR, DM	178
BINARY DIVIDE (@)DVB(53)	DVB(53) Dd Dr R	Divides the 4-digit hexadecimal dividend (Dd) by the 4-digit divisor (Dr), and outputs result to the designated result words (R and R + 1). R and R + 1 must be in the same data area. D d ÷ Dr Quotient R Remainder R+ 1	**Dd /Dr:** IR, SR, HR, AR, LR, TC, DM, # **R:** IR, HR, AR, LR	179
DOUBLE BCD ADD (@)ADDL(54)	ADDL(54) Au Ad R	Adds two 8-digit values (2 words each) and the content of CY, and outputs the result to the specified result words. All words for any one operand must be in the same data area. Au+ 1 Au + Ad+ 1 Ad + CY CY R+ 1 R	**Au/ Ad:** IR, SR, HR, AR, LR, TC, DM **R:** IR, HR, AR, LR, DM	161
DOUBLE BCD SUBTRACT (@)SUBL(55)	SUBL(55) Mi Su R	Subtracts both the 8-digit BCD subtrahend and the content of CY from an 8-digit BCD minuend, and outputs the result to the specified result words. All words for any one operand must be in the same data area. Mi + 1 Mi – Su + 1 Su – CY CY R + 1 R	**Mi/ Su:** IR, SR, HR, AR, LR, TC, DM **R:** IR, HR, AR, LR, DM	164

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

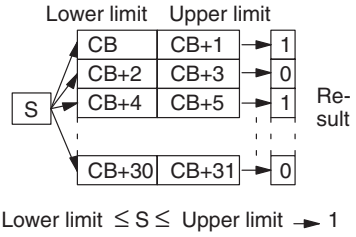
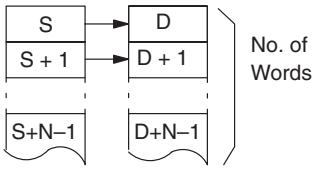
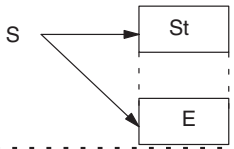
IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 23615	23700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	C1000H: DM 0000 to DM 4095 C2000H: DM 0000 to DM 6655	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas	Page
DOUBLE BCD MULTIPLY (@)MULL(56)	MULL(56) Md Mr R	Multiplies the 8-digit BCD multiplicand and 8-digit BCD multiplier, and outputs the result to the specified result words. All words for any one operand must be in the same data area. 	**Md/ Mr:** IR SR HR AR LR TC DM **R:** IR HR AR LR DM	166
DOUBLE BCD DIVIDE (@)DIVL(57)	DIVL(57) Dd Dr R	Divides the 8-digit BCD dividend by an 8-digit BCD divisor, and outputs the result to the specified result words. All words for any one operand must be in the same data area. 	**Dd /Dr:** IR SR HR AR LR TC DM **R:** IR HR AR LR DM	168
DOUBLE BCD TO DOUBLE BINARY (@)BINL(58)	BINL(58) S R	Converts the BCD value of the two source words (S: starting word) into binary and outputs the converted data to the two result words (R: starting word). All words for any one operand must be in the same data area. 	**S:** IR SR HR AR LR TC DM **R:** IR HR AR LR DM	148
DOUBLE BINARY TO DOUBLE BCD (@)BCDL(59)	BCDL(59) S R	Converts the binary value of the two source words (S: starting word) into eight digits of BCD data, and outputs the converted data to the two result words (R: starting result word). Both words for any one operand must be in the same data area. 	**S:** IR SR HR AR LR DM **R:** IR HR AR LR DM	150

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

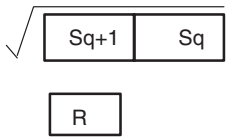
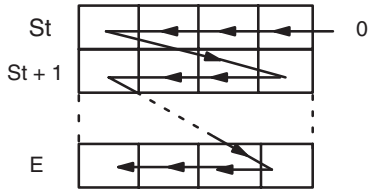
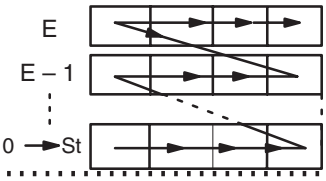
IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 23615	23700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	C1000H: DM 0000 to DM 4095 C2000H: DM 0000 to DM 6655	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas			Page
BIT COUNTER (@)BCNT(67)	BCNT(67) N SB R	Counts the number of ON bits in one or more words (SB is the beginning source word) and outputs the result to the specified result word (R). N gives the number of words to be counted. All words in which bit are to be counted must be in the same data area.	**N:** IR SR HR AR LR TC DM	**R:** IR HR AR LR TC DM	**SB:** IR SR HR AR LR TC DM	210
BLOCK COMPARE (@)BCMP(68)	BCMP(68) S CB R	Compares a 1-word binary value (S) with the 16 ranges given in the comparison table (CB is the starting word of the comparison block). If the value falls within any of the ranges, the corresponding bits in the result word (R) will be set. The comparison block must be within one data area.  Lower limit ≤ S ≤ Upper limit → 1	**S:** IR SR HR AR LR TC DM #	**CB:** IR SR HR AR LR TC DM	**R:** IR HR AR LR TC DM	145
BLOCK TRANSFER (@)XFER(70)	XFER(70) N S D	Moves the content of several consecutive source words (S gives the address of the starting source word) to consecutive destination words (D is the starting destination word). All source words must be in the same data area, as must all destination words. Transfers can be within one data area or between two data areas, but the source and destination words must not overlap.  	**N:** IR SR HR AR LR TC DM #	**S :** IR HR AR LR TC DM	**D:** IR SR HR AR LR TC DM #	138
BLOCK SET (@)BSET(71)	BSET(71) S St E	Copies the content of one word or constant (S) to several consecutive words (from the starting word, St, through to the ending word, E). St and E must be in the same data area.  	**St/E:** IR HR AR LR TC DM	**S:** IR SR HR AR LR TC DM #		136

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

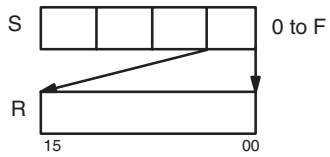
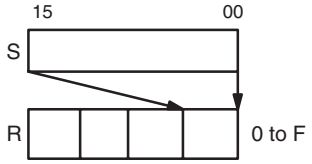
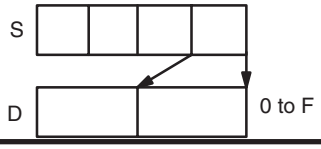
IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 23615	23700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	C1000H: DM 0000 to DM 4095 C2000H: DM 0000 to DM 6655	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas	Page
SQUARE ROOT (@)ROOT(72)	ROOT(72) Sq R	Computes the square root of an 8-digit BCD value (Sq and Sq + 1) and outputs the truncated 4-digit, integer result to the specified result word (R). Sq and Sq + 1 must be in the same data area. 	**Sq:** IR SR HR AR LR TC DM **R:** IR HR AR LR DM	172
DATA EXCHANGE (@)XCHG(73)	XCHG(73) E1 E2	Exchanges the contents of two words (E1 and E2). 	**E1/E2:** IR HR AR LR TC DM	138
ONE DIGIT SHIFT LEFT (@)SLD(74)	SLD(74) St E	Shifts all data, between the starting word (St) and ending word (E), one digit (four bits) to the left, writing zero into the rightmost digit of the starting word. St and E must be in the same data area. 	**St/E:** IR HR AR LR DM	133
ONE DIGIT SHIFT RIGHT (@)SRD(75)	SRD(75) E St	Shifts all data, between starting word (St) and ending word (E), one digit (four bits) to the right, writing zero into the leftmost digit of the ending word. St and E must be in the same data area. 	**St/E:** IR HR AR LR DM	134

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 23615	23700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	C1000H: DM 0000 to DM 4095 C2000H: DM 0000 to DM 6655	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas	Page
4-TO-16 DECODER (@)MLPX(76)	MLPX(76) S Di R	Converts up to four hexadecimal digits in the source word (S), into decimal values from 0 to 15, and turns ON the corresponding bit(s) in the result word(s) (R). There is one result word for each converted digit. Digits to be converted are designated by Di. (The rightmost digit specifies the first digit. The next digit to the left gives the number of digits to be converted minus 1. The two leftmost digits are not used.) 	**S:** IR, SR, HR, AR, LR, TC, DM **Di:** IR, HR, AR, LR, TC, DM, # **R:** IR, HR, AR, LR, DM	150
16-TO-4 ENCODER (@)DMPX(77)	DMPX(77) S R Di	Determines the position of the leftmost ON bit in the source word(s) (starting word: S) and turns ON the corresponding bit(s) in the specified digit of the result word (R). One digit is used for each source word. Digits to receive the converted values are designated by Di. (The rightmost digit specifies the first digit. The next digit to left gives the number of words to be converted minus 1. The two leftmost digits are not used.) 	**S:** IR, SR, HR, AR, LR, TC, DM **R:** IR, HR, AR, LR, DM **Di:** IR, HR, AR, LR, TC, DM, #	152
7-SEGMENT DECODER (@)SDEC(78)	SDEC(78) S Di D	Converts hexadecimal values from the source word (S) into 7-segment display data. Results are placed in consecutive half-words, starting at the first destination word (D). Di gives digit and destination details. (The rightmost digit gives the first digit to be converted. The next digit to the left gives the number of digits to be converted minus 1. If the next digit is 1, the first converted data is transferred to left half of the first destination word. If it is 0, the transfer is to the right half). 	**S:** IR, SR, HR, AR, LR, TC, DM **Di:** IR, HR, AR, LR, TC, DM, # **D:** IR, HR, AR, LR, DM	154

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

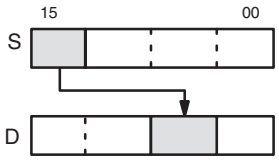
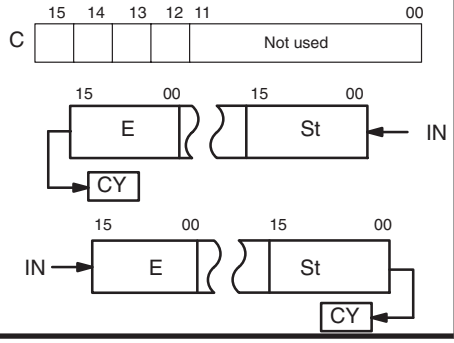
IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 23615	23700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	C1000H: DM 0000 to DM 4095 C2000H: DM 0000 to DM 6655	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas	Page
FLOATING POINT DIVIDE (@)FDIV(79)	FDIV(79) Dd Dr R	Divides one floating point value by another and outputs a floating point result. The rightmost seven digits of each set of two words (eight digits) are used for mantissa, and the leftmost digit is used for the exponent and its sign (Bits 12 to 14 give the exponent value, 0 to 7. If bit 15 is 0, the exponent is positive; if it's 1, the exponent is negative). Dd+ 1 | Dd ÷ Dr+ 1 | Dr R + 1 | R	**Dd/ Dr:** IR SR HR AR LR TC DM **R:** IR HR AR LR DM	169
SINGLE WORD DISTRIBUTE (@)DIST(80)	DIST(80) S DBs Of	Moves one word of source data (S) to the destination word whose address is given by the destination base word (DBs) plus offset (Of). S → Base (DBs) + Offset (OF) (S) → (DBs + Of)	**S:** IR SR HR AR LR TC DM # **DBs:** IR HR AR LR TC DM **Of:** IR HR AR LR TC DM #	139
DATA COLLECT (@)COLL(81)	COLL(81) SBs Of D	Extracts data from the source word and writes it to the destination word (D). The source word is determined by adding the offset (Of) to the address of the source base word (SBs). The offset cannot be entered as a constant when using C120 or C500 PCs. Base (DBs) + Offset (OF) (SBs+Of) → (D)	**SBs:** IR SR HR AR LR TC DM **Of:** IR HR AR LR TC DM # **D:** IR HR AR LR TC DM	139
MOVE BIT (@)MOVB(82)	MOVB(82) S Bi D	Transfers the designated bit of the source word or constant (S) to the designated bit of the destination word (D). The rightmost two digits of the bit designator (Bi) specify the source bit. The two leftmost digits specify the destination bit. S D	**S:** IR SR HR AR LR DM # **Bi:** IR HR AR LR TC DM # **D:** IR HR AR LR DM	140

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

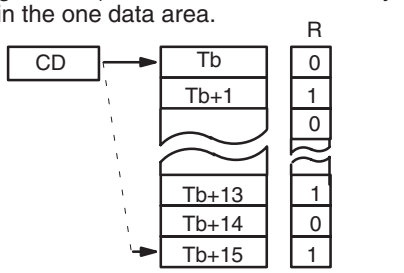
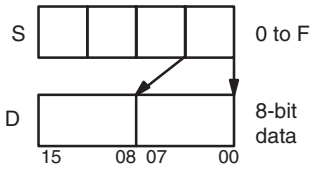
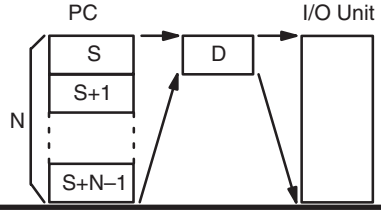
IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 23615	23700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	C1000H: DM 0000 to DM 4095 C2000H: DM 0000 to DM 6655	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas			Page
MOVE DIGIT (@)MOVD(83)	MOVD(83) S Di D	Moves hexadecimal content of up to four specified 4-bit source digit(s) from the source word to the specified destination digit(s) (S gives the source word address. D specifies the destination word). Specific digits within the source and destination words are defined by the Digit Designator (Di) digits. (The rightmost digit gives the first source digit. The next digit to the left gives the number of digits to be moved. The next digit specifies the first digit in the destination word.) 	**S:** IR SR HR AR LR TC DM #	**Di:** IR HR AR LR TC DM #	**D:** IR SR HR AR LR TC DM	141
REVERSIBLE SHIFT REGISTER (@)SFTR(84)	SFTR(84) C St E	Shifts bits in the specified word or series of words either left or right. Starting (St) and ending words (E) must be specified. Control word (C) contains shift direction, reset input, and data input. (Bit 12: 0 = shift right, 1 = shift left. Bit 13 is the value shifted into the source data, with the bit at the opposite end being moved to CY. Bit 14: 1 = shift enabled, 0 = shift disabled. If bit 15 is ON when SFTR(89) is executed with an ON condition, the entire shift register and CY will be set to zero.) St and E must be in the same data area and St must be less than or equal to E. 	**St/E/C:** IR HR AR TC LR DM			129

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

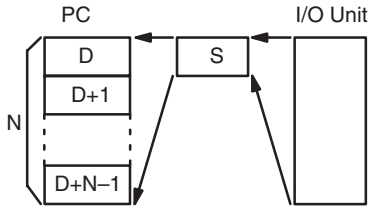
IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 23615	23700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	C1000H: DM 0000 to DM 4095 C2000H: DM 0000 to DM 6655	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas	Page
TABLE COMPARE (@)TCMP(85)	TCMP(85) CD TB R	Compares a 4-digit hexadecimal value (CD) with values in table consisting of 16 words (TB: is the first word of the comparison table). If the value of CD falls within any of the comparison ranges, corresponding bits in result word (R) are set (1 for agreement, and 0 for disagreement). The table must be entirely within the one data area.  1: agreement 0: disagreement	**CD:** IR SR HR AR LR TC DM # **TB/R:** IR HR AR LR TC DM	146
ASCII CONVERT (@)ASC(86)	ASC(86) S Di D	Converts hexadecimal digits from the source word (S) into 8-bit ASCII values, starting at leftmost or rightmost half of the starting destination word (D). The rightmost digit of Di designates the first source digit. The next digit to the left gives the number of digits to be converted. The next digit specifies the whether the data is to be transferred to the rightmost (0) or leftmost (1) half of the first destination word. The leftmost digit specifies parity: 0: none, 1: even, or 2: odd. 	**S:** IR SR HR AR LR TC DM **Di:** IR HR LR TC DM # **D:** IR HR LR DM	157
I/O WRITE (@)WRIT(87)	WRIT(87) N S D	Transfers word data through I/O word (D) allocated to an Special I/O Unit and sequentially writes data to the memory area of the Special I/O Unit. N is the number of words to be transferred, and S is the address of the first PC source word to be transferred. The EQ Flag is set when the transfer is completed. 	**N:** IR SR HR AR LR TC DM # **S:** IR SR HR AR LR TC DM **D:** I	218

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 23615	23700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	C1000H: DM 0000 to DM 4095 C2000H: DM 0000 to DM 6655	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas			Page
I/O READ (@)READ(88)	READ(88) N S D	READ(88) reads data from memory area of an Special I/O Unit and transfers it through word (S) allocated to the Special I/O Unit to destination words (D gives the address of the first destination word). N is the number of words to be transferred. The EQ Flag is set when the transfer is completed. 	**N**: IR SR HR AR LR TC DM #	**S:** IR D: IR HR AR LR TC DM	**D:** IR HR AR LR TC DM	218
INTERRUPT CONTROL (@)INT(89)	INT(89) CC N D	Controls programmed (scheduled) interrupts and interrupts from Interrupt Input Units. Each PC can have up to 4 IIUs. N defines the source of the interrupt: 000 to 003 designate the no. of the IIU; 004 designates a scheduled interrupt. In IIUs, bits 00 to 07 identify the interrupting subroutine, higher bits are not used. Bit 00 of Unit 0 corresponds to interrupt subroutine 00, through to bit 07 of Unit 3 which corresponds to subroutine 31. CC is the control code, the meaning of which depends on the value of N, as follows:	**CC:** 000 to 002	**N:** 000 to 004	**D:** IR HR AR LR TC DM #	185

CC	N = 000 to 003	N = 004
000	Masks and unmasks interrupt bits for the designated Unit (N) according to the data in D. Bits corresponding to ON bits in D are masked, those corresponding to OFF bits are unmasked. Masked bits are recorded and will be executed when they are unmasked (unless previously cleared).	The interrupt time interval is set according to the data in D (00.01 to 99.99 s) The decimal point is not entered. The interrupt is cancelled if D is 00.00.
001	Clears the masked interrupt bits of the designated Unit (N) according to the corresponding ON bits in D. The subroutines corresponding to bits cleared in this manner will not be executed when the bit is unmasked.	The time to the first interrupt is set according to the data in D (00.01 to 99.99 s) The decimal point is not entered. The interrupt is cancelled if D is 00.00.
002	Copies the mask status of the designated IIU to D.	Copies the time interval data to D.

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 23615	23700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	C1000H: DM 0000 to DM 4095 C2000H: DM 0000 to DM 6655	0000 to 9999 or 0000 to FFFF

<table>
<tr><th>Name Mnemonic</th><th>Symbol</th><th>Function</th><th>Operand Data Areas</th><th>Page</th></tr>
<tr><td>NETWORK SEND
(@)SEND(90)</td><td>SEND(90)
S
D
C</td><td>Transfers data from n source words (S is the starting word) to the destination words (D is the first address) in node N of the specified network (in a SYSMAC LINK or NET Link System). The format of the control words varies depending on the type of system. In both types of systems, the first control word (C) gives the number of words to be transferred.
For NET Link Systems, in word C+1, bit 14 specifies the system (0 for system 1, and 1 for system 0), and the rightmost 7 bits define the network number. The left half of word C+2 specifies the destination port (00: NSB, 01/02: NSU), and the right half specifies the destination node number. If the destination node number is set to 0, data is transmitted to all nodes.
For SYSMAC LINK Systems, the right half of C+1 specifies the response monitoring time (default 00: 2 s, FF: monitoring disabled), the next digit to the left gives the maximum number of re-transmissions (0 to 15) that the PC will attempt if no response signal is received. Bit 13 specifies whether a response is needed (0) or not (1), and bit 14 specifies the system number (0 for system 1, and 1 for system 0). The right half of C+2 gives the destination node number. If this is set to 0, the data will be sent to all nodes.
NET Link<table><tr><td>C</td><td colspan="3">n: no. of words to be transmitted (0 to 1000)</td></tr><tr><td>C+1</td><td>0X00</td><td>0000</td><td>Network no. (0 to 127)</td></tr><tr><td>C+2</td><td colspan="2">Destination port no.</td><td>Destination node no. (0 to 126)</td></tr></table>SYSMAC LINK<table><tr><td>C</td><td colspan="3">n: no. of words to be transmitted, 0 to 1000</td></tr><tr><td>C+1</td><td>0XX0</td><td>Re-trans-missions</td><td>Response monitor time (0.1 to 25.4 s)</td></tr><tr><td>C+2</td><td>0000</td><td>0000</td><td>Destination node no. (0 to 62)</td></tr></table>Source N → Destination node N
S → D
S+1 → D+1
S+n–1 → D+n–1</td><td>S:
IR
SR
HR
AR
LR
TC
DM
D/C:
IR
HR
AR
LR
TC
DM</td><td>219</td></tr>
<tr><td>SUBROUTINE ENTER
(@)SBS(91)</td><td>SBS(91) N</td><td>Calls subroutine N. Moves program operation to the specified subroutine.</td><td>N:
00 to 99</td><td>183</td></tr>
<tr><td>SUBROUTINE START
SBN(92)</td><td>SBN(92) N</td><td>Marks the start of subroutine N.</td><td>N:
00 to 99</td><td>183</td></tr>
<tr><td>RETURN
RET(93)</td><td>RET(93)</td><td>Marks the end of a subroutine and returns control to the main program.</td><td>None</td><td>183</td></tr>
</table>

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 23615	23700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	C1000H: DM 0000 to DM 4095 C2000H: DM 0000 to DM 6655	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas	Page
WATCHDOG TIMER REFRESH (@)WDT(94)	WDT(94) T	Sets the maximum and minimum limits for the watchdog timer (normally 0 to 130 ms). New limits: Maximum time = 130 + (100 x T) Minimum time = 130 + (100 x (T–1))	**T:** 0 to 63	211
BLOCK PROGRAM START BPRG(96)	BPRG(96) N	Indicates the beginning of a block program. Block programs allow flowchart-style programming within ladder diagram programs.	**N:** 00 to 99	190
I/O REFRESH (@)IORF(97)	IORF(97) St E	Refreshes all I/O words between the start (St) and end (E) words. Only I/O words may be designated. Normally these words are refreshed only once per cycle, but refreshing words before use in an instruction can increase execution speed. St must be less than or equal to E.	**St/E:** IR	211

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 23615	23700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	C1000H: DM 0000 to DM 4095 C2000H: DM 0000 to DM 6655	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas	Page
NETWORK RECEIVE (@)RECV(98)	RECV(98) S D C	Transfers data from the source words (S is the first word) from node N of the specified network (in a SYSMAC LINK or NET Link System) to the destination words starting at D. The format of the control words varies depending on the type of system. In both types of systems, the first control word (C) gives the number of words to be transferred. For NET Link Systems, in the second word (C+1), bit 14 specifies the system (0 for system 1, and 1 for system 0), and the rightmost 7 bits define the network number. The left half of word C+2 specifies the source port (00: NSB, 01/02: NSU), and the right half specifies the source node number. For SYSMAC LINK Systems, the right half of C+1 specifies the response monitoring time (default 00: 2 s, FF: monitoring disabled), the next digit to the left gives the maximum number of re-transmissions (0 to 15) that the PC will attempt if no response signal is received. Bit 13 specifies whether a response is needed (0) or not (1), and bit 14 specifies the system number (0 for system 1, and 1 for system 0). The right half of C+2 gives the source node number. **NET Link** C: n: no. of words to be transmitted (0 to 1000) C+1: 0X00 | 0000 | Network no. (0 to 127) C+2: Source port no. (NSB: 00, NSU: 01/02) | Source node no. (0 to 126) **SYSMAC LINK** C: n: no. of words to be transmitted, 0 to 1000 C+1: 0XX0 | Re-transmissions | Response monitor time (0.1 to 25.4 s) C+2: 0000 | 0000 | Source node no. (0 to 62) Source node N → Destination node S → D S+1 → D+1 S+n–1 → D+n–1	**S:** IR, SR, HR, AR, LR, TC, DM **C/D:** IR, HR, AR, LR, TC, DM	221

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 23615	23700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	C1000H: DM 0000 to DM 4095 C2000H: DM 0000 to DM 6655	0000 to 9999 or 0000 to FFFF

Block Program Instructions

Name Mnemonic	Symbol	Function	Operand Data Areas	Page
BLOCK PROGRAM END BEND<01>	BEND<01>	Indicates the end of a block program.	None	190
CONDITIONAL BRANCH IF<02> IF<02> B IF<02> NOT B	IF<02> IF<02> NOT	Indicates the part of the program that is to be executed when a given condition is satisfied.	**B**: IR SR HR AR LR TC	191
NO BRANCH ELSE<03>	ELSE<03>	Specifies the part of the program that is to be executed when the IF condition is not satisfied.	None	191
BRANCH END IEND<04>	IEND<04>	Defines the end of the program portion that has started with IF<02>.	None	191
ONE CYCLE AND WAIT WAIT<05> WAIT<05> B WAIT<05> NOT B	WAIT<05> WAIT<05> NOT	Halts execution of a block program until a specified condition is satisfied.	**B:** IR SR HR AR LR TC	193
CONDITIONAL BLOCK EXIT EXIT<06> EXIT<06 B EXIT<06> NOT B	EXIT<06> EXIT<06> NOT	Exits a block program if a given condition is satisfied.	**B**: IR SR HR AR LR TC	197
SET SET<07> **B**	SET<07>	Sets (turns ON) the specified bit.	**B**: IR HR AR LR	191
RESET RSET<08> B	RSET<08>	Resets (turns OFF) the specified bit.	**B**: IR HR AR LR	191
LOOP LOOP<09>	LOOP<09>	Defines the beginning of section to be repeated until a specified terminal condition is satisfied.	None	197
LOOP END LEND<10> LEND<10> B LEND<10> NOT B	LEND<10> LEND<10> NOT	Defines the end of the section to be repeated. Execution of the specified section continues until the terminal condition is satisfied.	**B**: IR SR HR AR LR TC	197
BLOCK PROGRAM PAUSE BPPS<11> N	BPPS<11>	Causes the execution of designated block program to pause until a specified condition is satisfied (often used in conjunction with a timer or counter).	**N**: 0 to 99	198

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 23615	23700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	C1000H: DM 0000 to DM 4095 C2000H: DM 0000 to DM 6655	0000 to 9999 or 0000 to FFFF

Name Mnemonic	Symbol	Function	Operand Data Areas	Page
BLOCK PROGRAM RESTART BPRS<12> N	BPRS<12>	Restarts execution of the designated block program.	**N**: 0 to 99	198
TIMER WAIT TIMW<13> N SV	TIMW<13>	The execution of the block program between the TIMW<13> instruction and BEND<04> is not executed until the set value of the specified timer has been reached. SV: 000.0 to 999.9.	**SV**: IR AR DM HR LR # **N**: TC	195
COUNTER WAIT CNTW<14> N SV I	CNTW<14>	The portion of block program between the CNTW<14> instruction and BEND<04> is not executed until the set value of the specified counter has been reached.	**SV**: IR AR DM HR LR # **N**: TC	196
HIGH-SPEED TIMER WAIT TMHW<15> N SV	TMHW<15>	The portion of program between the TIMH<15> instruction and BEND<04> is not executed until the set value of the high-speed timer has been reached. SV: 00.02 to 99.99	**SV**: IR AR DM HR LR # **N**: TC	195

Data Areas

These footnote tables show the actual ranges of all data areas. Bit numbers are provided (except for DM and TC areas); remove the rightmost two digits for word numbers.

IR	SR	HR	TR	AR	LR	TC	DM	#
00000 to 23615	23700 to 25515	HR 0000 to 9915	TR 0 to 7	AR 0000 to 2715	LR 0000 to 6315	TC 000 to 511	C1000H: DM 0000 to DM 4095 C2000H: DM 0000 to DM 6655	0000 to 9999 or 0000 to FFFF

Appendix C
Programming Console Operations

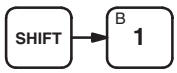
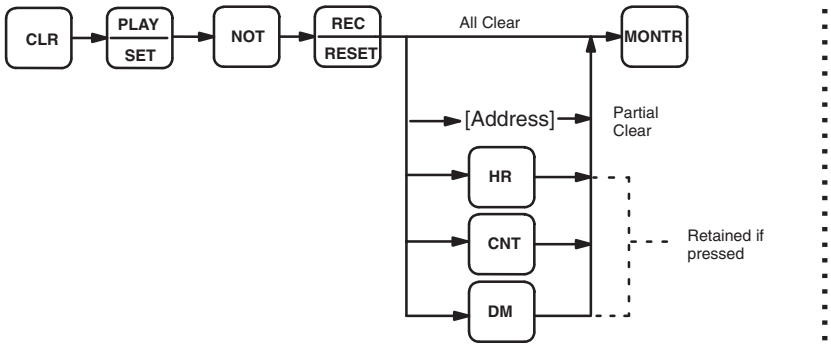
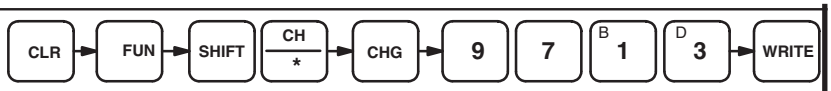
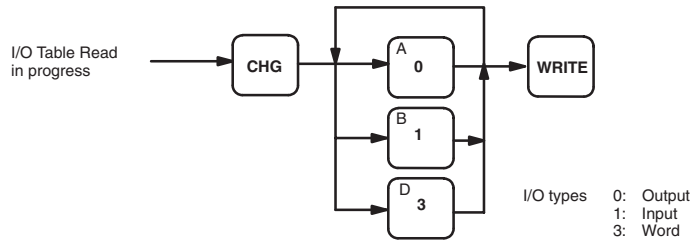
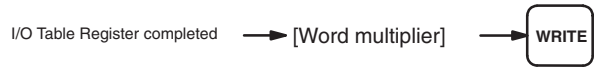
This Appendix provides a table which sums up the Programming Console operations. The table gives the operation name and the function it performs.

Name	Function	Page
Password Input	Prompts the user for the access password.	61
Buzzer ON/OFF	Controls whether the buzzer will sound for keystroke inputs.	339
Data Clear	Used to erase data, either selectively or totally, from the Program Memory and the IR, AR, HR, DM, and TC areas.	62
I/O Table Register	Registers the I/O after initial entry or subsequent amendments.	64
I/O Table Change	Allows Dummy Units to be registered in the table so that the table does not need to be altered when Units are added.	67
Word Multiplier Enter	Used to assign word multipliers to Remote I/O Master Units as required.	65
I/O Table Verify	Checks the I/O Table against the actual arrangement of I/O Units.	70
I/O Table Read	Displays the Unit type, location, I/O word number, and word multiplier (where applicable).	71
I/O Table Transfer	Copies the I/O table to RAM. This enables simultaneous writing of the table and user program to EPROM.	67
Error Message Read	Displays error messages in sequence, starting with the most severe messages.	66, 244
On-line I/O Unit Change	Permissible Units can be mounted or removed while the CPU is operating.	69
Address Designation	Displays the specified address.	75
Program Input	Used to edit or input program instructions.	76
Program Read	Allows the user to scroll through the program address-by-address. In RUN and MONITOR modes, status of bits is also given.	75
Program Search	Searches a program for the specified data address or instruction.	82
Instruction Insert Instruction Delete	Allows a new instruction to be inserted before the displayed instruction, or deletes the displayed instruction (respectively).	83
Program Check	Checks the completed program for three levels of syntax errors.	79
Debug Operation Enter	Places the Unit into debug mode.	244
Address Execution	Executes a program instruction-by-instruction starting from the current address.	246
Debug Execution	Debugs the program from the current address to the first END(01).	248
Address Trace	Debugs a section of 250 instructions and stores the results in the Trace Memory.	249
Address Trace Read	Allows the user to scroll address-by-address through the contents of Trace Memory.	250
Bit/Word Monitor	Displays the specified address whose operand is to be monitored. In RUN or MONTR mode it will show the status of the operand for any bit or word in any data area.	252
Three-word Monitor	Simultaneously monitors three consecutive words.	259
Temporary Forced Set/Reset	Set: Used to turn on bits or timers, or to increment counters currently displayed on the left of the screen. Reset: Used to turn off bits, or to reset timers or counters.	255
Hex/BCD Data Change	Used to change the value of the leftmost BCD or hexadecimal word displayed during a Bit/Word Monitor operation.	257
Binary Data Change	Changes the value of 16-bit words bit-by-bit. Bits can be changed temporarily or permanently to the desired status.	262
SV Change SV Reset	Alters the SV of a timer or counter either by incrementing or decrementing the value, or by overwriting the original value with a new one.	263
Three-word Change	Used to change the value of a word displayed during a 3-word Monitor operation.	260

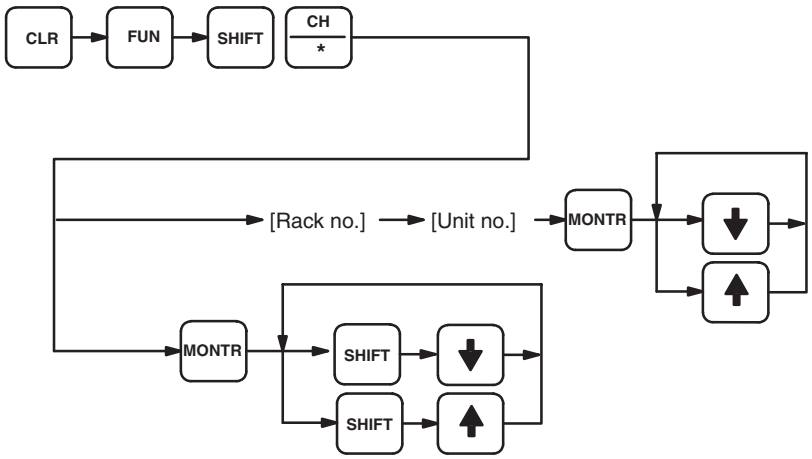
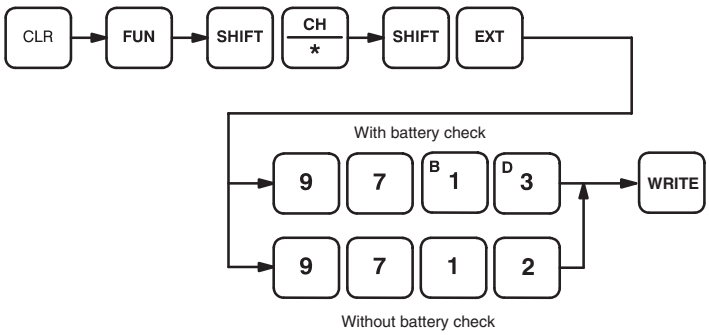
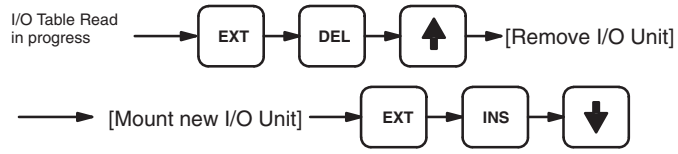
Name	Function	Page
Cycle Time Display	Measures the duration of the current cycle. Cycle times will vary according to the execution conditions which exist in each cycle.	81
Hex-ASCII Display Change	Converts 4-digit hexadecimal data in the DM area to ASCII and vice-versa.	258
Binary Monitor	Displays the monitored area in binary format.	261
Program Memory Save	Saves Program Memory to tape.	277
Program Memory Restore	Reads Program Memory from tape.	278
Program Memory Compare	Compares Program Memory data on tape with that in the Program Memory area.	278
DM Data Save, Restore, Compare	Save, restore, and compare tape operations for DM area data.	280
File Memory Clear	Clears memory, either selectively or totally, from the FM area.	266
File Memory Edit	Allows data in the FM area to be read and modified.	275
File Memory Read	Copies UM data from the FM area to a specified part of the Program Memory RAM, or IOM data in the FM area to one of the CPU data areas.	272
File Memory Write	Writes data from the Program Memory or data areas to the FM area.	267
File Memory Verify	Compares data in the FM area with that in the specified Program Memory or data areas.	270

System Operations

The following table lists the Programming Console operations according to their function. A brief description of each operation is given, along with the allowable modes in which it can be implemented, and the keystroke sequence used to enter it.

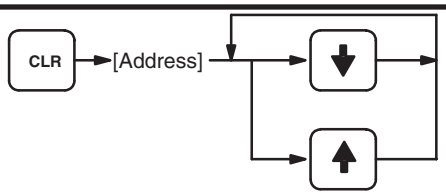
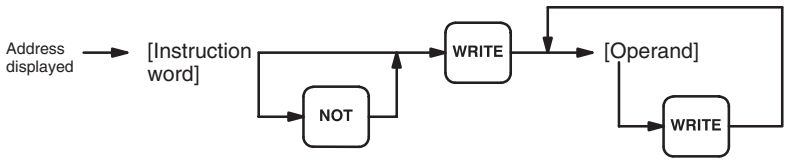
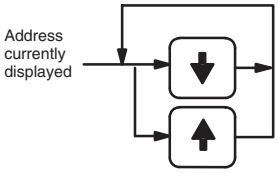
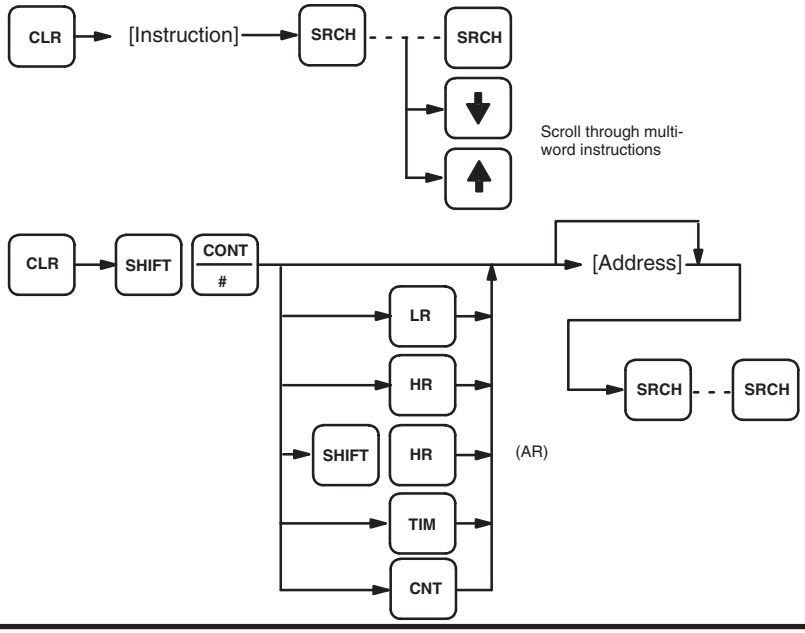
Operation/Description	Modes*	Key sequence
Password Input Controls access to the PC's programming functions. To gain access to the system once "PASSWORD" has been displayed, press CLR, MONTR, and then CLR.	R M P	
Buzzer ON/OFF The buzzer can be switched to operate whenever Programming Console keys are pressed (as well as for the normal error indication). BZ is displayed in the upper right corner when the buzzer is operative. The buzzer can be enabled by pressing SHIFT and then 1 immediately after entering the password, or after changing the mode.	R M P	
Data Clear Unless otherwise specified, this operation will clear all erasable memory in Program Memory and IR, HR, AR, DM, and TC areas. To clear EPROM memory the write enable switch must be ON (i.e., enabled). The branch lines shown are used only when performing a partial memory clear, with each of the memory areas entered being retained. Specifying an address will result in the Program Memory after and including that address being deleted. All memory up to that address will be retained.	P	
I/O Table Register Whenever I/O Unit changes are made that affect the operation of the system, the I/O table needs to be corrected to reflect the changes. This includes the initial registration once the system has been established.	P	
I/O Table Change Allows I/O Units to be removed or added without the need to re-register the I/O Table or to amend the user program. By creating dummy entries, word address discrepancies can be avoided when the Units are added later.	P	
Word Multiplier Enter If the system has Remote I/O Masters connected to I/O Link Units, Optical I/O Units, or Remote Terminals, a word multiplier must be assigned to each Master after the I/O table is registered. Word multipliers can take values from 0 to 3.	P	

*Modes in which the given instruction is applicable: R = RUN, M = MONITOR, P = PROGRAM

Operation/Description	Modes*	Key sequence
I/O Table Verify Used to check that the registered I/O Table matches the actual arrangement of I/O Units. Pressing VER displays the next inconsistency.	R P M	
I/O Table Read Used to read the I/O Table. The display gives the Unit type, location, I/O word allocation, and word multiplier. Rack and unit numbers will vary according to the system in use. By pressing Shift before the arrow key, the Rack and Unit numbers need not be specified.	R P M	
I/O Table Transfer Transfers a copy of the I/O Table to RAM so that the table and and the user program can be written to EPROM memory at the same time. The operation will not work if the memory is not RAM, or when the contents exceed 31.7K words.	P	
On-line I/O Unit Change Allows mounting or removal of permissible Units while the CPU is operating. Be sure to disconnect the terminal block before removal and to reconnect it after mounting.	R P M	

*Modes in which the given instruction is applicable: R = RUN, M = MONITOR, P = PROGRAM

Programming Operations

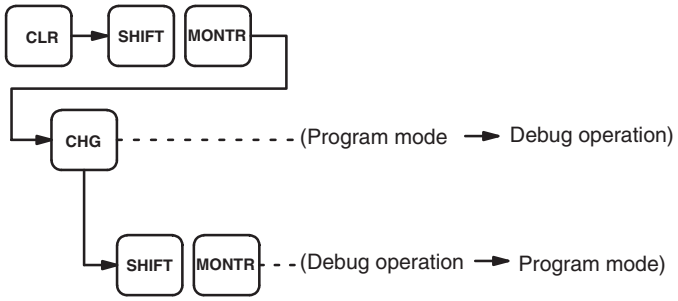
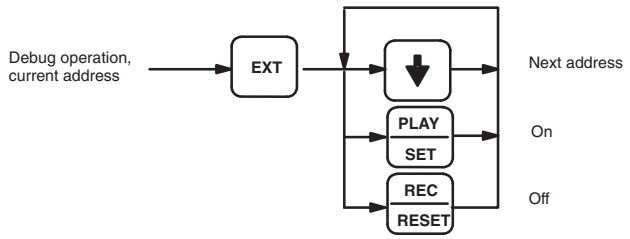
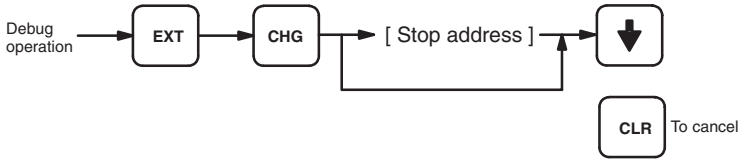
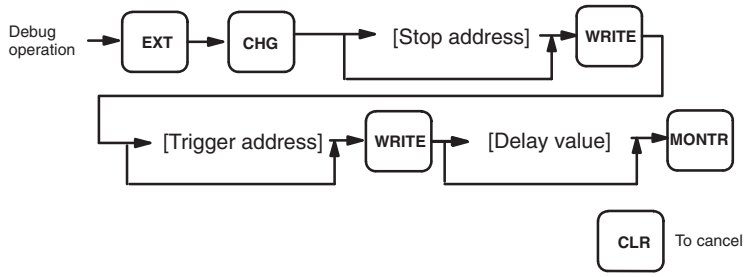
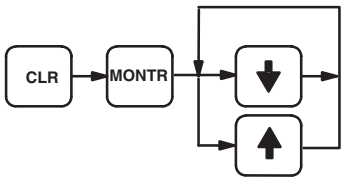
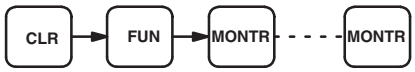
Operation/Description	Modes*	Key sequence
Address Designation Displays the specified address. Can be used to start programming from a non-zero address or to access an address for editing. Leading zeros need not be entered. The contents of the address will not be displayed until the down key is pressed. The up and down keys can then be used to scroll through the Program Memory.	R P M	
Program Input Used to enter or edit program instructions. This operation over-writes the contents of the memory at the displayed address. Once at the desired address, enter the new instruction word and then press WRITE (preceded by NOT for differentiated instructions). Input the required operands, and press WRITE after each.	P	
Program Read Allows the user to scroll through the program address-by-address. If the Program Memory is read in RUN or MONITOR mode, the ON/OFF status of each displayed bit is also shown.	R P M	
Program Search Allows the program to be searched for occurrences of any designated instruction or data area address. To designate a bit address, press SHIFT, CONT/#, and then input the address. Then press SRCH. Pressing SRCH again will find the next occurrence. For multi-word instructions, the up and down keys can be used to scroll through the words before continuing the search. In RUN or MONITOR mode, the ON/OFF status of each monitored bit will also be displayed.	R P M	

*Modes in which the given instruction is applicable: R = RUN, M = MONITOR, P = PROGRAM

Operation/Description	Modes*	Key sequence
Instruction Insert and Instruction Delete The displayed instruction can be deleted, or another instruction can be inserted before it. Care should be taken to avoid inadvertent deletions as there is no way of recovering the instructions other than to re-enter them. When an instruction is deleted all subsequent instruction addresses are automatically adjusted so that there are no empty addresses, or instructions without addresses.	P	At the desired position in program: **Insert**: [Enter new instruction] → INS → ↓ **Delete**: Instruction currently displayed → DEL → ↑
Program Check Once a program has been entered, it should be checked for errors. This program check can be used to search for three levels of syntax errors. Details of the errors covered by each level are given in the relevant manuals. The address where the error was generated will also be displayed.	P	CLR → SRCH → A 0 / B 1 / C 2 (0, 1, 2: check levels) → SRCH → SRCH: Press SRCH to find next error. CLR: Cancel

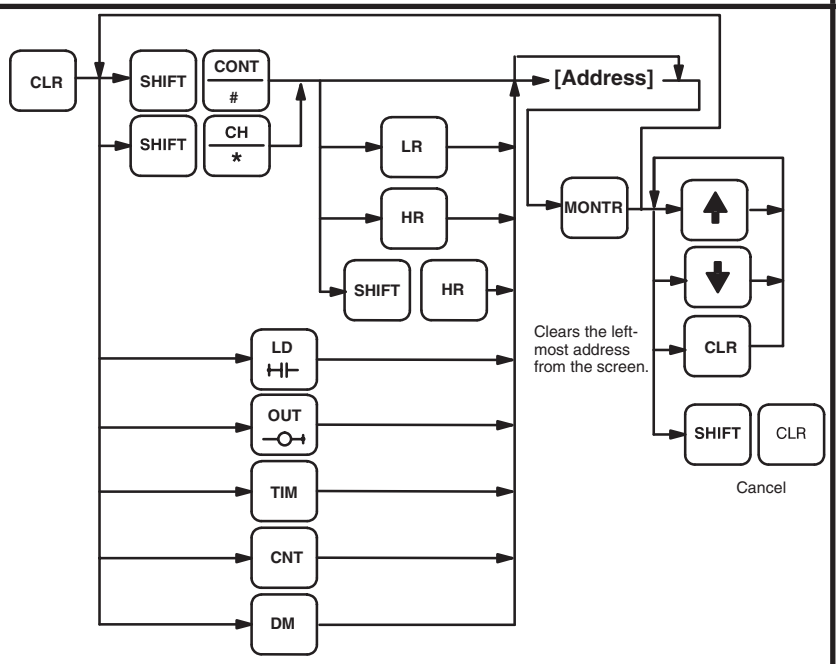
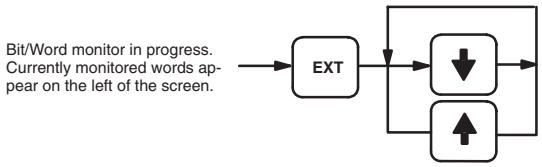
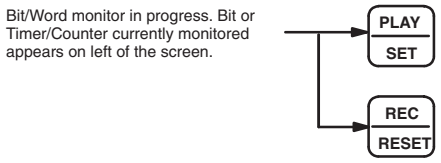
*Modes in which the given instruction is applicable: R = RUN, M = MONITOR, P = PROGRAM

Debugging Operations

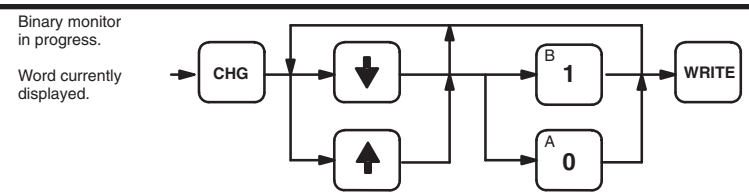
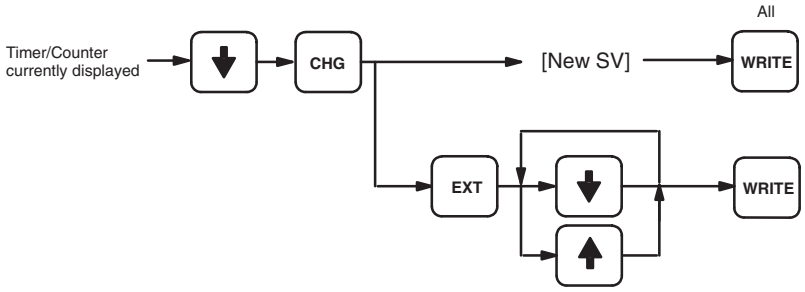
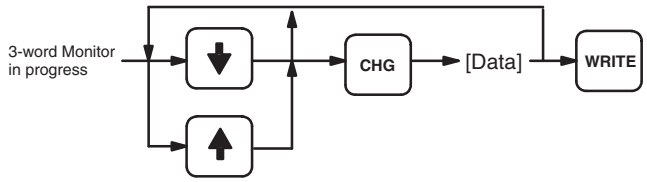
Operation/Description	Modes*	Key sequence
Debug Operation Enter Used to debug the program while in Program Mode. Program Input, Program Clear, Instruction Insert, and Instruction Delete operations are not available. Unless the Data Retention control bit is ON, all data in the IR, AR, and LR areas will be cleared when debug operation is entered or exited.	P	
Address Execution Executes the program instruction-by-instruction starting from the current address. Press the down key to go to the next address. Word data can be monitored via the Data Monitor operation.	P	
Debug Execution Executes the program from the current address to the specified stop address. Debugging will stop if END(01) is encountered, or if CLR is pressed.	P	
Address Trace Traces up to 250 instructions from the program. Tracing begins at the address indicated by the trigger address and the delay. The delay can be in the range –249 to +250 (press NOT to toggle between – and +). Tracing continues up to the stop address, or until a maximum of 250 instruction words have been recorded. If END(01) is encountered, the tracing will loop back to the beginning address and continues until 250 words have been traced and recorded.	P	
Address Trace Read Reads the Address Trace starting at the trigger address. The up and down arrow keys can be used to scroll through the instructions word-by-word.	P	
Error Message Read Displays error messages in sequence with most severe messages displayed first. Press monitor to access remaining messages. In PROGRAM mode, pressing MONTR clears the displayed message from memory and the next message is displayed.	R P M	

*Modes in which the given instruction is applicable: R = RUN, M = MONITOR, P = PROGRAM

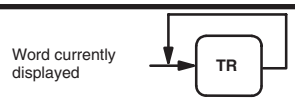
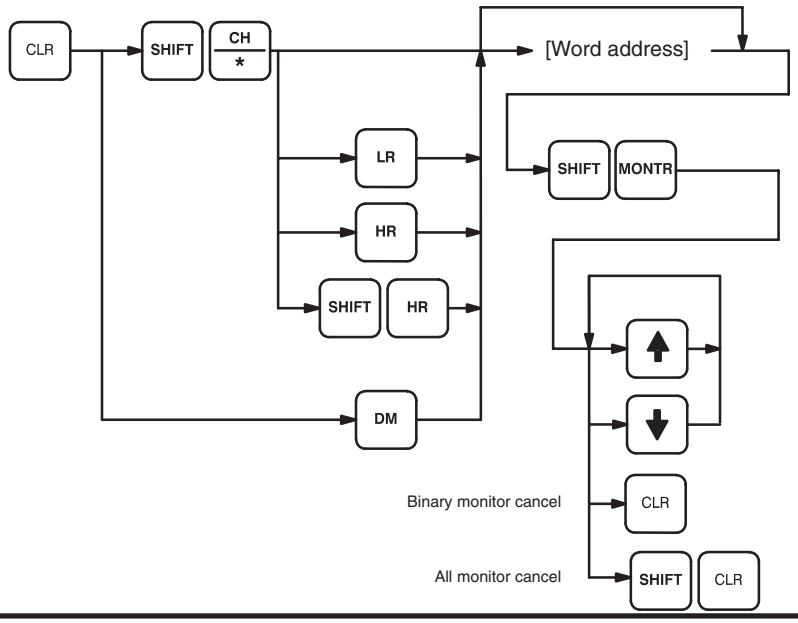
Monitoring and Data Changing Operations

Operation/Description	Modes*	Key sequence
Bit/Word Monitor Up to six memory addresses, containing either words or bits, or a combination of the two, can be monitored at once. Only three can be displayed at any one time. If operated in RUN or MONITOR mode, the status of monitored bits will also be displayed. The operation can be started from a cleared display by entering the address of the first word or bit to be monitored and pressing MONTR, or from any address in the program by displaying the address of the bit or word to be monitored and pressing MONTR. When a timer or counter is monitored, its PV will be displayed and a box is displayed in the bottom left hand corner if the Completion Flag is ON.	R P M	
3-word Monitor Monitors three consecutive words simultaneously. Specify the lowest valued address of the three words, press MONTR, and then press EXT to display the data contents of the specified word and the two words that follow. Pressing CLR will change the three-word monitor operation into a single-word display.	R P M	
Temporary Forced Set/Reset If a bit, timer, or counter address is leftmost on the screen during a Bit/Word Monitor operation, pressing PLAY/SET will turn ON the bit, start the timer, or increment the counter. Pressing REC/RESET will turn OFF the bit, or reset the timer or counter. These force-sets and force-resets are effective while the key is held down. Timers will not operate in PROGRAM mode. SR bits are not affected by this operation.	P M	
Hex/BCD Data Change Used to edit the leftmost BCD or hexadecimal value displayed during a Bit/Word Monitor operation. If a timer or counter is leftmost on the display, the PV will be the value displayed and affected by this operation. It can only be changed in MONITOR mode and only while the timer or counter is operating. SR words cannot be changed using this operation.	P M	

*Modes in which the given instruction is applicable: R = RUN, M = MONITOR, P = PROGRAM

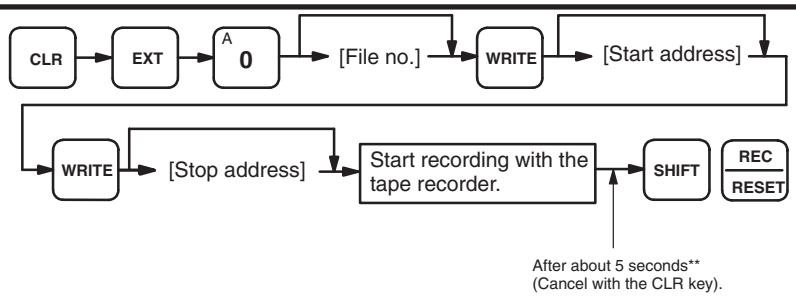
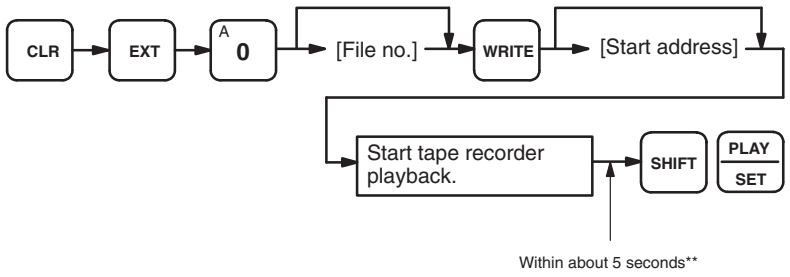
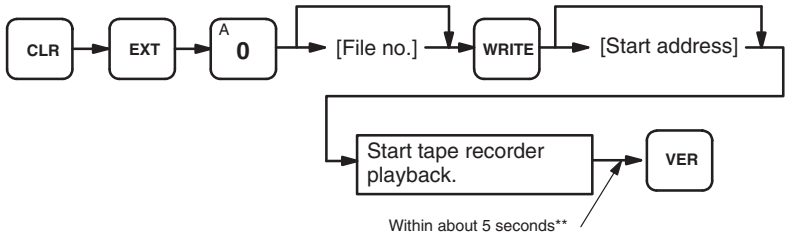
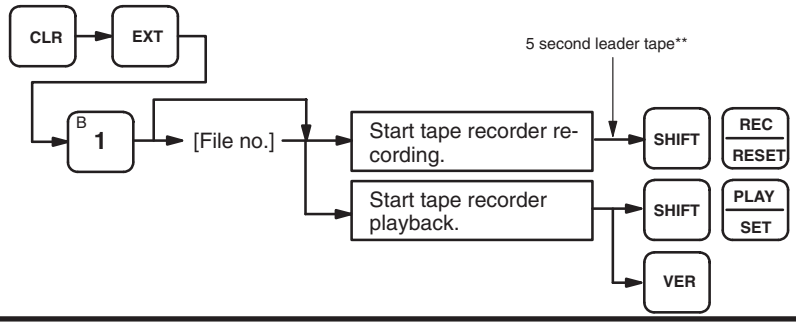
Operation/Description	Modes*	Key sequence
Binary Data Change This operation is used to change the value of IR, HR, AR, LR, or DM words bit-by-bit. The cursor can be moved left by using the up key, and right by using the down key. The position of the cursor is the bit that will be overwritten.	P M	
SV Change, **SV Reset** There are two ways of modifying the SVs for timers and counters. One method is to enter a new value. The second is to increment or decrement the existing SV. In MONITOR mode the SV can be changed while the program is being executed. Incrementing and decrementing can only be carried out if the SV has been entered as a constant.	P M M	
3-word Change This operation changes the value of a word displayed during a 3-word Monitor operation. The blinking cursor indicates the word that will be affected by the operation. The cursor can be moved by using the up and down keys. When the cursor is at the desired location, press CHG. After entering the new data, pressing WRITE causes the original data to be overwritten.	P M	
Cycle Time Display This operation should be performed after all syntax errors have been corrected. The cycle time can only be checked in RUN or MONITOR mode and while the program is being executed. The cycle time displayed after pressing CLR and MONTR is that for the current cycle. Pressing MONTR again will display a new cycle time. Any difference between successive cycle times is due to the different execution conditions that exist during each cycle.	R M	

*Modes in which the given instruction is applicable: R = RUN, M = MONITOR, P = PROGRAM

Operation/Description	Modes*	Key sequence
Hex-ASCII Display Change Converts 4-digit hexadecimal DM data to ASCII and vice-versa.	R P M	
Binary Monitor The contents of a monitored word can be specified to be displayed in binary by pressing SHIFT and MONTR after entering the word address. Words can be scrolled by pressing the up and down keys to increment and decrement the displayed address. To terminate the binary display, press CLR.	R P M	

*Modes in which the given instruction is applicable: R = RUN, M = MONITOR, P = PROGRAM

Cassette Tape Operations

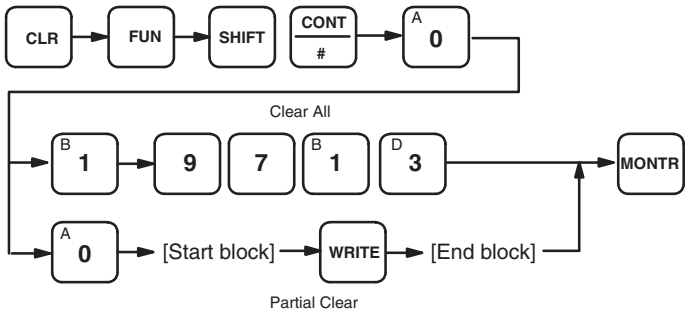
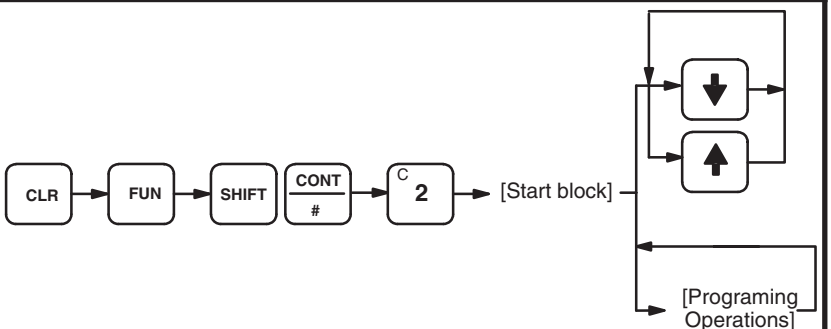
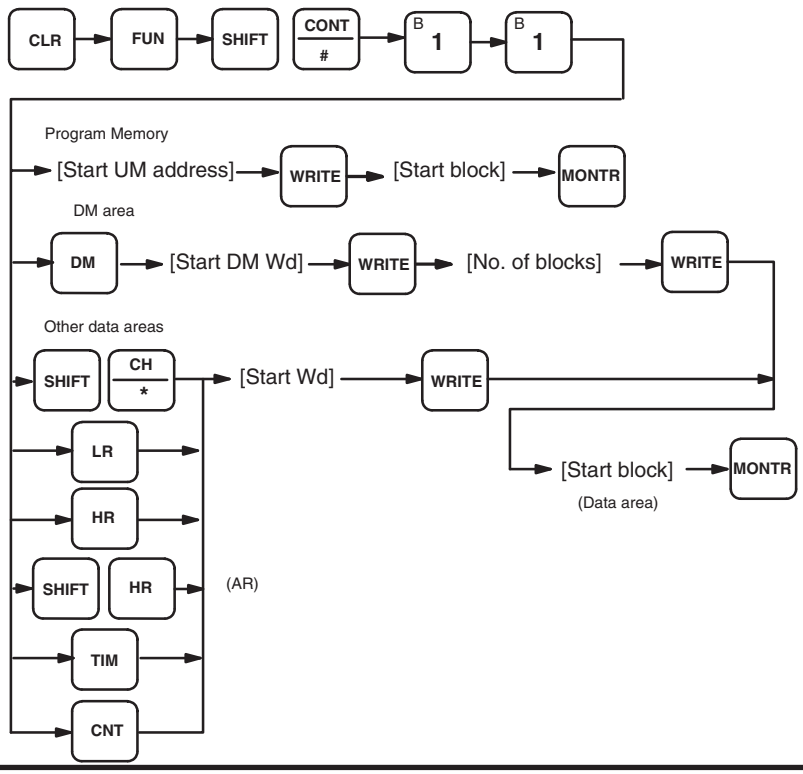
Operation/Description	Modes*	Key sequence
Program Memory Save Copies data from the Program Memory to tape. The file no. specified in the instructions provides an identifying address for the information within the tape. Each file number should be used only once per tape. If only a part of the Program Memory is to be stored, the appropriate start and stop addresses must be entered. Each C60 tape can store approximately 16K words on each side of the tape. When the start address is entered, the maximum stop address is set as the default. Do not set a stop address greater than this one. If you wish to record past this address the additional information will need to be recorded either on the flip side of the tape or on a separate tape. After starting the tape recorder, wait about 5 seconds before pressing SHIFT REC/RESET. This is to allow the leader tape to pass before the data transmission starts.	P	
Program Memory Restore To read Program Memory data which has been recorded on a cassette tape, the keystrokes are as given here. The file number must be the same as the one entered when the data was recorded. The read operation will proceed from the specified start address up to the end of the tape, unless halted by a CLR command. The instruction must be completed before the required data is reached on the tape, i.e., usually before the leader tape finishes.	P	
Program Memory Compare The procedure to compare Program Memory data stored on a tape with that in the PC's Program Memory area is the same as that for reading it (see above), except that after starting the tape playback, VER should be pressed instead of SHIFT and PLAY/SET.	P	
DM Data Save, Restore, Compare The procedures for transferring DM area data to and from tape, and for comparing it, are basically the same as for the Program Memory, given above. The exceptions are that start and stop addresses are not required, and the DM area is specified instead of the Program Memory. Each operation will continue through to the end of the tape unless cancelled by pressing CLR.	P	

*Modes in which the given instruction is applicable: R = RUN, M = MONITOR, P = PROGRAM

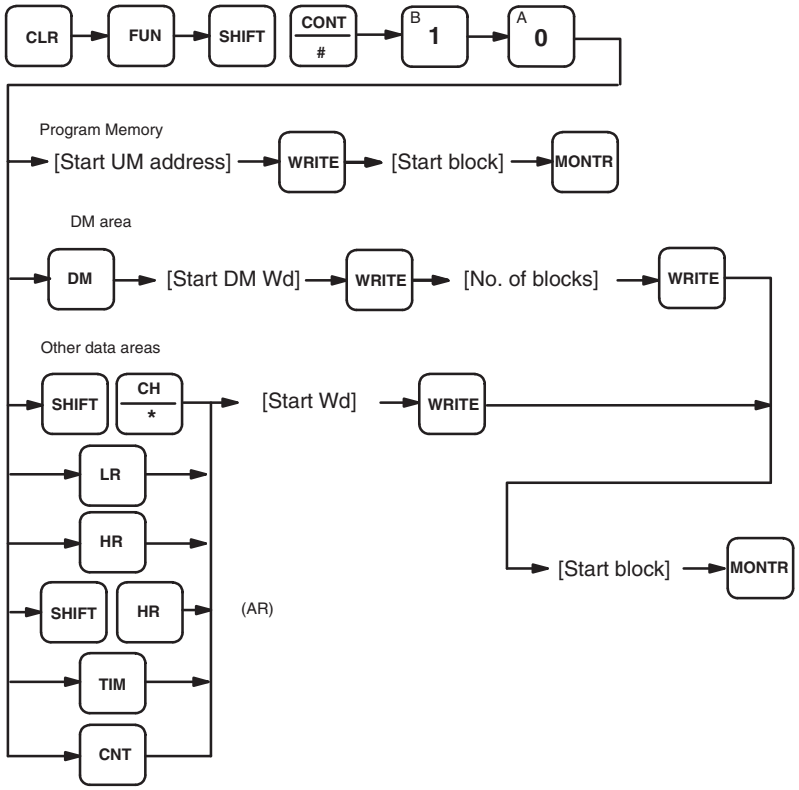
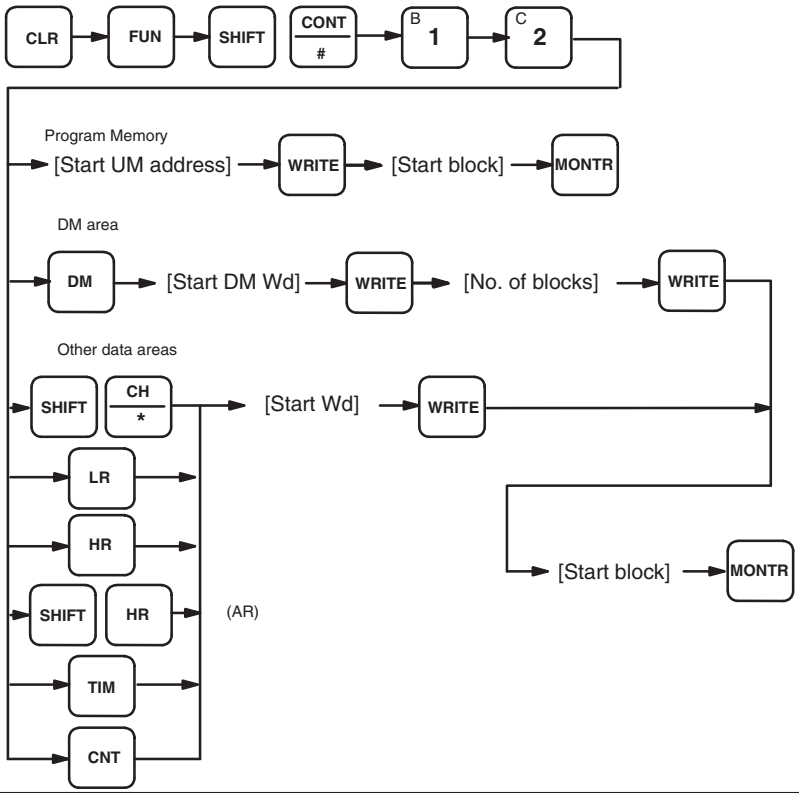
**These times take the cassette leader tape into consideration according to the following:
a) When recording to tape, the leader tape needs to be allowed to pass before the data transmission to the tape player starts.

b) When restoring from tape or comparing data, the Programming Console needs to be ready to receive data before the data is transfered from the tape.

File Memory Operations

Operation/Description	Modes*	Key sequence
File Memory Clear Clears the FM area. For partial clears, the start block number and end block number must be specified. The FM area should be totally cleared when being used for the first time or if an FM error occurs due to corrupted memory data. Clearing the entire FM area initializes it in preparation for future data storage.	P	
File Memory Edit Allows the data stored in the FM area to be read and modified. The data can be viewed in all modes, but editing is only possible in MONITOR or PROGRAM modes. If a start block is not specified, the operation will begin at the first block. The up and down keys can be used to scroll through each block word-by-word. Specific words can be displayed by entering CLR, the word address, and then pressing the down key. To change the content of the displayed word, enter the new data and press WRITE. Program and comment data stored in the FM area cannot be modified using this operation.	PM	
File Memory Read Used to read user program data (UM) stored in the FM area and transfer it to a specified part of the Program Memory RAM, or to read user data in the FM area and transfer it to one of the CPU data areas. Data is read and transferred in blocks of 128 words. UM data is read from the specified FM start block and continues until the first END(01) instruction, or until the first non-UM FM block is encountered, or until the destination memory area overflows. User data is read starting from the specified start block and continues until the end. If a non-data block is encountered, an error message is displayed and transfer is aborted. When transferring to the DM area, the number of blocks must be specified.	P	

*Modes in which the given instruction is applicable: R = RUN, M = MONITOR, P = PROGRAM

Operation/Description	Modes*	Key sequence
File Memory Write Writes data from the Program Memory or specified data areas to the FM area. Data is written in blocks of 128 words. When transferring data from the Program Memory, the data between the specified start address and the next END(01) instruction is moved to the FM area. If the amount of data exceeds the available memory in the FM area, only the data that will fit into the memory is transferred. When transferring data from the DM area, the number of blocks must be specified.	P M	
File Memory Verify Compares the data stored in the FM area with data in the Program Memory or specified data areas. When comparing with data in the Program Memory, reading starts at the specified address and continues to the first END(01). This is compared with the data starting at the specified start block in the FM area. For comparison with the DM area, the number of blocks to be compared must be specified. Blocks consist of 128 words. When comparing with data in one of the data areas, reading begins at the start word in the data area and continues until the end of that particular data area.	P M	

*Modes in which the given instruction is applicable: R = RUN, M = MONITOR, P = PROGRAM

Appendix D
Error and Arithmetic Flag Operation

The following table shows the instructions that affect the ER, CY, GT, LT and EQ flags. In general, ER indicates that operand data is not within requirements. CY indicates arithmetic or data shift results. GT indicates that a compared value is larger than some standard, LT that it is smaller, and EQ, that it is the same. EQ also indicates a result of zero for arithmetic operations. Refer to subsections of *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, TIMW<13>, CNT, CNT<14>, TMHW<15>, and CNTR<12> 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.

SR bits (25503 to 25507) change according to the result of the previously executed instruction. If the previous instruction is a differentiated instruction, it will be executed only for the first scan in which the input conditions are satisfied and so the SR bits will remain in the status determined by the result of the instruction executed before the differentiated one.

	Flags				
Instructions	**25503(ER)**	**25504(CY)**	**25505(GR)**	**25506(EQ)**	**25507(LE)**
TIM	↕				
CNT					
END(01)	OFF	OFF	OFF	OFF	OFF
STEP(08)					
SNXT(09)					
CNTR(12)					
TIMH(15)	↕				
WSFT(16)					
CMP(20)	↕		↕	↕	↕
MOV(21)					
MVN(22)	↕			↕	
BIN(23)					
BCD(24)					
ASL(25)					
ASR(26)	↕	↕		↕	
ROL(27)					
ROR(28)					
COM(29)	↕			↕	
ADD(30)	↕	↕		↕	
SUB(31)					
MUL(32)					
DIV(33)					
ANDW(34)					
ORW(35)	↕			↕	
XORW(36)					
XNRW(37)					
INC(38)					
DEC(39)					
STC(40)		ON			
CLC(41)		OFF			
FILR(42)	↕				
FILW(43)					

Note: ↕ means that the flag is affected by the result of instruction execution.

	Flags				
Instructions	**25503(ER)**	**25504(CY)**	**25505(GR)**	**25506(EQ)**	**25507(LE)**
FILP(44)	↕				
TRSM(45)					
MSG(46)	↕				
ADB(50)	↕	↕		↕	
SBB(51)					
MLB(52)	↕			↕	
DVB(53)					
ADDL(54)	↕	↕		↕	
SUBL(55)					
MULL(56)					
DIVL(57)					
BINL(58)	↕			↕	
BCDL(59)					
FUN67					
BCMP(68)					
XFER(70)	↕				
BSET(71)					
ROOT(72)	↕			↕	
XCHG(73)					
SLD(74)					
SRD(75)	↕				
MLPX(76)					
DMPX(77)					
SDEC(78)					
FDIV(79)					
DIST(80)	↕			↕	
COLL(81)					
MOVB(82)	↕				
MOVD(83)					
SFTR(84)	↕	↕			
TCMP(85)	↕			↕	
ASC(86)	↕				

Note: ↕ means that the flag is affected by the result of instruction execution.

	Flags				
Instructions	**25503(ER)**	**25504(CY)**	**25505(GR)**	**25506(EQ)**	**25507(LE)**
WRIT(87)	↕			↕	
READ(88)					
FUN89	↕				
SEND(90)	↕				
SBS(91)					
SBN(92)					
RET(93)					
WDT(94)					
BPRG(96)					
IORF(97)					
RECV(98)	↕				
BEND<01>					
IF<02>					
ELSE<03>					
IEND<04>					
WAIT<05>					
EXIT<06>					
SET<07>					
RSET<08>					
LOOP<09>					
LEND<10>					
BPPS<11>					
BPRS<12>					
TIMW<13>	↕				
CNTW<14>					
TMHW<15>					

Note: ↕ means that the flag is affected by the result of instruction execution.

Appendix E
Data Areas

The data areas in the C1000H and C2000H are summarized below. These are the same for both PCs unless specified. Only dedicated bits are shown specifically. The use of all other bits is determined either by the System the PC is involved in, e.g., PC Link or SYSMAC LINK Systems use the LR area, or by the programmer, e.g., storage of data in the DM area.

In the following table, prefixes are included with bit and word addresses when inputting them is required to designate the area, i.e., bits input without a prefix are considered to be IR or SR bits.

Area	Bits	Words	Notes
IR	00000 to 23600	000 to 236	Possible for I/O bits C1000H without Remote I/O . . . . . 00000 to 06315 Remote I/O with C1000H . . . . . 06400 to 12700 C2000H (with or without Remote I/O) . . 00000 to 12700 IR area work bits 12700 to 23600 plus any I/O bits not used for I/O.
SR	24700 to 25515	247 to 255	SR bits are dedicated for specific purposes. Unused bits are not available for programmer use. In designating operands, the SR area is considered as a continuation of the IR area. See tables of dedicated bits following this table.
HR	HR 0000 to HR 9915	HR 00 to HR 99	HR bits are available for general data storage and manipulation. The HR area maintains data when PC power is turned off.
AR	AR 0000 to AR 2715	AR 00 to AR 27	AR bits are mostly dedicated for specific purposes. Unused AR bits may be used as works bits. See tables of dedicated bits following this table.
LR	LR 0000 to LR 6315	LR 00 to LR 63	LR bits are used for data transmission in PC Link and SYSMAC LINK Systems. When the PC does not contain either of these Systems, LR bits may be used as work bits.
DM	Not accessible as bits.	C1000H: DM 0000 to DM 4095 C2000H: DM 0000 to DM 6655	DM words are basically used for data storage.
TC	(TC 000 to TC 511)	(TC 000 to TC 511)	The TC area consists of TC numbers used to manipulate and access timers and counters. 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.
TR	(TC 0 to TR 7)	Not accessible as words.	TR bits can only be used in the Load and Output instructions to store and retrieve execution conditions. Storing and retrieving execution conditions is necessary when programming certain types of branching ladder diagrams.

Dedicated Bits

Most of the bits in the SR and AR area are dedicated for specific purposes. These are summarized in the following tables. Refer to *3-4 SR Area* and *3-5 AR Area* for details.

SR Area

As a rule, SR area bits can be used only for the purposes for which they are dedicated. SR 237 through SR 251 can be used as work bits if the Systems for which they are intended are not used in the PC System.

Word(s)	Bit(s)	Function
237	00 to 07	Completion code output area following execution of SEND(90)/RECV(98) for SYSMAC LINK System
	08 to 15	Not used
238 to 241	00 to 15	Data link status output area for operating level 0 of SYSMAC LINK or SYSMAC NET Link System
242 to 245	00 to 15	Data link status output area for operating level 1 of SYSMAC LINK or SYSMAC NET Link System
246	00 to 15	Not used
247 to 250	00 to 07	PC Link Unit Run Flags or data link status for operating level 1
	08 to 15	PC Link Unit Run Flags or data link status for operating level 1
251	00 to 15	Remote I/O Error Flags
252	00 and 01	Not used
	02	Operating Level 0 Data Link Operating Flag
	03	SEND(90)/RECV(98) Error Flag
	04	SEND(90)/RECV(98) Enable Flag
	05	Operating Level 1 Data Link Operating Flag
	06	Rack-mounting Host Link Unit Level 1 Error Flag
	08	CPU-mounting Host Link Unit Error Flag
	09	CPU-mounting Host Link Unit Restart Bit
	10	Leave set to 0
	12	Data Retention Control Bit
	13	Rack-mounting Host Link Unit Restart Bit
	14	Leave set to 0
	15	Output OFF Bit
253	00 to 07	FAL number output area.
	08	Low Battery Flag
	09	Cycle Time Error Flag
	10	I/O Verification Error Flag
	11	Rack-mounting Host Link Unit Level 0 Error Flag
	12	Remote I/O Error Flag
	13	Normally ON Flag
	14	Normally OFF Flag
	15	First Cycle

Word(s)	Bit(s)	Function
254	00	1-minute clock pulse bit
	01	0.02-second clock pulse bit
	07	Step Flag
	08 to 12	Duplex System flags
255	00	0.1-second clock pulse bit
	01	0.2-second clock pulse bit
	02	1.0-second clock pulse bit
	03	Instruction Execution Error (ER) Flag
	04	Carry (CY) Flag
	05	Greater Than (GR) Flag
	06	Equals (EQ) Flag
	07	Less Than (LE) Flag

AR Area

Word(s)	Bit(s)	Function
07	00 to 03	Data Link setting for operating level 0 of SYSMAC LINK System
	05 to 07	Data Link setting for operating level 1 of SYSMAC LINK System
	08 to 15	Not used. May be used as work bits.
08 to 10	00 to 15	Active Node Flags for SYSMAC LINK System nodes of operating level 0
11	00 to 13	
11	14	Communications Controller Error Flag for operating level 0
	15	EEPROM Error Flag for operating level 0
12 to 14	00 to 15	Node Active Flags for SYSMAC LINK System nodes of operating level 1
15	00 to 13	
15	14	Communications Controller Error Flag for operating level 1
	15	EEPROM Error Flag for operating level 1
16	00 to 15	SYSMAC LINK/SYSMAC NET Link System operating level 0 service time per cycle
17	00 to 15	SYSMAC LINK/SYSMAC NET Link System operating level 1 service time per cycle
18	12	Trace Complete Flag
	13	Tracing Flag
	14	Trace Start Bit
	15	Sampling Start Bit

Word(s)	Bit(s)	Function
19	00	File Memory Unit Error Reset Bit
	01	FM Data Transfer Flag
	02	FM Write/Read Flag
	03	FM Blocks Different Error Flag
	04	FM Write-protected Error Flag
	05	Unsuccessful FM Write Flag
	06	FM Checksum Error Flag
	07	File Memory Unit Low Battery Flag
	08	FM Blocks 0 to 249 Write-protect Bit
	09	FM Blocks 250 to 499 Write-protect Bit
	10	FM Blocks 500 to 749 Write-protect Bit
	11	FM Blocks 750 to 999 Write-protect Bit
	12	FM Blocks 1,000 to 1,249 Write-protect Bit
	13	FM Blocks 1,250 to 1,499 Write-protect Bit
	14	FM Blocks 1,500 to 1,749 Write-protect Bit
	15	FM Blocks 1,750 to 1,999 Write-protect Bit
20	00 to 15	FM Blocks Counter
21	00 to 15	Remaining FM Blocks Counter
22	00 to 11	On-line Removal First Word Indicator
	12 to 14	Number of Words Indicator for On-line Removal
	15	On-line Removal Flag
23	00 to 15	Power-Off Counter
24	00 to 03	Leftmost digit of FALS-generating address (AR 25 contains the other four digits)
	04 and 05	Not used and not accessible by user.
	06	Level 1 Network Parameter Flag
	07	Level 0 Network Parameter Flag
	08 to 10	Not used and not accessible by user.
	11	PC Link Unit Level 1 Mounted Flag
	12	PC Link Unit Level 0 or Single-level PC Link Unit Mounted Flag
	13	SYSMAC NET Link Unit Mounted Flag
	14	Rack-mounting Host Link Unit Mounted Flag
	15	CPU-mounting Device Flag
25	00 to 15	Rightmost four digits of FALS-generating address (AR 2400 to AR 2403 contain the fifth digit)
26	00 to 15	Maximum cycle time
27	00 to 15	Present cycle time

Appendix F
I/O Assignment Records Sheets

This appendix contains sheets that can be copied by the programmer to record I/O bit allocations and terminal assignments on the Racks, as well as details of work bits, data storage areas, timers, and counters.

Programmer: **Program:** **Date:** **Page:**

Word:	Unit:	
Bit	**Field device**	**Notes**
00		
01		
02		
03		
04		
05		
06		
07		
08		
09		
10		
11		
12		
13		
14		
15		

Word:	Unit:	
Bit	**Field device**	**Notes**
00		
01		
02		
03		
04		
05		
06		
07		
08		
09		
10		
11		
12		
13		
14		
15		

Word:	Unit:	
Bit	**Field device**	**Notes**
00		
01		
02		
03		
04		
05		
06		
07		
08		
09		
10		
11		
12		
13		
14		
15		

Word:	Unit:	
Bit	**Field device**	**Notes**
00		
01		
02		
03		
04		
05		
06		
07		
08		
09		
10		
11		
12		
13		
14		
15		

Programmer: **Program:** **Date:** **Page:**

Area:	Word:	
Bit	**Usage**	**Notes**
00		
01		
02		
03		
04		
05		
06		
07		
08		
09		
10		
11		
12		
13		
14		
15		

Area:	Word:	
Bit	**Usage**	**Notes**
00		
01		
02		
03		
04		
05		
06		
07		
08		
09		
10		
11		
12		
13		
14		
15		

Area:	Word:	
Bit	**Usage**	**Notes**
00		
01		
02		
03		
04		
05		
06		
07		
08		
09		
10		
11		
12		
13		
14		
15		

Area:	Word:	
Bit	**Usage**	**Notes**
00		
01		
02		
03		
04		
05		
06		
07		
08		
09		
10		
11		
12		
13		
14		
15		

Programmer: **Program:** **Date:** **Page:**

Word	Contents	Notes

Programmer: **Program:** **Date:** **Page:**

TC address	T or C	Set value	Notes

Appendix G
Program Coding Sheet

The following page can be copied for use in coding ladder diagram programs. It is designed for flexibility, allowing the user to input all required addresses and instructions.

When coding programs, be sure to specify all function codes for instructions and data areas (or # for constant) for operands. These will be necessary when inputting programs though a Programming Console or other Peripheral Device.

Programmer: **Program:** **Date:** **Page:**

Address	Instruction	Operand(s)

Programmer: **Program:** **Date:** **Page:**

Address	Instruction	Operand(s)

Appendix H
Data Conversion Table

Decimal	BCD	Hex	Binary
00	00000000	00	00000000
01	00000001	01	00000001
02	00000010	02	00000010
03	00000011	03	00000011
04	00000100	04	00000100
05	00000101	05	00000101
06	00000110	06	00000110
07	00000111	07	00000111
08	00001000	08	00001000
09	00001001	09	00001001
10	00010000	0A	00001010
11	00010001	0B	00001011
12	00010010	0C	00001100
13	00010011	0D	00001101
14	00010100	0E	00001110
15	00010101	0F	00001111
16	00010110	10	00010000
17	00010111	11	00010001
18	00011000	12	00010010
19	00011001	13	00010011
20	00100000	14	00010100
21	00100001	15	00010101
22	00100010	16	00010110
23	00100011	17	00010111
24	00100100	18	00011000
25	00100101	19	00011001
26	00100110	1A	00011010
27	00100111	1B	00011011
28	00101000	1C	00011100
29	00101001	1D	00011101
30	00110000	1E	00011110
31	00110001	1F	00011111
32	00110010	20	00100000

Appendix I
Extended ASCII
ASCII Codes

Bits 0 to 3		Bits 4 to 7													
BIN		0000	0001	0010	0011	0100	0101	0110	0111	1010	1011	1100	1101	1110	1111
	HEX	0	1	2	3	4	5	6	7	A	B	C	D	E	F
0000	0	NUL	DLE	Space	0	@	P	`	p		0	@	P	`	p
0001	1	SOH	DC_1	!	1	A	Q	a	q	!	1	A	Q	a	q
0010	2	STX	DC_2	"	2	B	R	b	r	"	2	B	R	b	r
0011	3	ETX	DC_3	#	3	C	S	c	s	#	3	C	S	c	s
0100	4	EOT	DC_4	$	4	D	T	d	t	$	4	D	T	d	t
0101	5	ENQ	NAK	%	5	E	U	e	u	%	5	E	U	e	u
0110	6	ACK	SYN	&	6	F	V	f	v	&	6	F	V	f	v
0111	7	BEL	ETB	'	7	G	W	g	w	'	7	G	W	g	w
1000	8	BS	CAN	(	8	H	X	h	x	(	8	H	X	h	x
1001	9	HT	EM	)	9	I	Y	i	y	)	9	I	Y	i	y
1010	A	LF	SUB	*	:	J	Z	j	z	*	:	J	Z	j	z
1011	B	VT	ESC	+	;	K	[	k	{	+	;	K	[	k	{
1100	C	FF	FS	,	<	L	¥	l	|	,	<	L	¥	l	|
1101	D	CR	GS	-	=	M	]	m	}	-	=	M	]	m	}
1110	E	S0	RS	.	>	N	^	n	←	.	>	N	^	n	
1111	F	S1	US	/	?	O	_	o	→	/	?	O	_	o	→

Glossary

address The location in memory where data is stored. For data areas, an address consists of a two-letter data area designation and a number that designates the word and/or bit location. For the UM area, an address designates the instruction location (UM area). In the FM area, the address designates the block location, etc.

allocation The process by which the PC assigns certain bits or words in memory for various functions. This includes pairing I/O bits to I/O points on Units.

AND A logic operation whereby the result is true if and only if both premises are true. In ladder-diagram programming the premises are usually ON/OFF states of bits or the logical combination of such states called execution conditions.

APF Acronym for all plastic fiber-optic cable.

AR area A PC data area allocated to flags, control bits, and work bits.

arithmetic shift A shift operation wherein the carry flag is included in the shift.

ASCII Short for American Standard Code for Information Interchange. ASCII is used to code characters for output to printers and other external devices.

ASCII Unit A Special I/O Unit used to program in BASIC. When connected to an NSU on a SYSMAC NET Link System, commands can be sent to other nodes.

Backplane A base onto which Units are mounted to form a Rack. Backplanes provide a series of connectors for these Units along with wiring to connect them to the CPU. Backplanes also provide connectors used to connect them to other Backplanes. In some Systems, different Backplanes are used for different Racks; in other Systems, Racks differ only according to the Units mounted to them.

BCD Short for binary-coded decimal.

BCD calculation An arithmetic calculation that uses numbers expressed in binary-coded decimal.

binary A number system where all numbers are expressed to the base 2, i.e., any number can be written using only 1's or 2's. Each group of four binary bits is equivalent to one hexadecimal digit.

binary calculation An arithmetic calculation that uses numbers expressed in binary.

binary-coded decimal A system used to represent numbers so that each group of four binary bits is numerically equivalent to one decimal digit.

bit A binary digit; hence a unit of data in binary notation. The smallest unit of information that can be electronically stored in a PC. The status of a bit is either ON or OFF. Different bits at particular addresses are allocated to spe-

	cial purposes, such as holding the status input from external devices, while other bits are available for general use in programming.
bit address	The location in memory where a bit of data is stored. A bit address must specify (sometimes by default) the data area and word that is being addressed, as well as the number of the bit.
bit designator	An operand that is used to designate the bit or bits of a word to be used by an instruction.
bit number	A number that indicates the location of a bit within a word. Bit 00 is the rightmost (least-significant) bit; bit 15 is the leftmost (most-significant) bit.
block	Block can refer to one of three aspects of PC operation: a block in the FM area, a block instruction (program), or a logic block. A block in the FM is the unit used to transfer data to and from the File Memory Unit and equals 128 words. Refer to *block instruction, block program,* and *logic block* for definitions of these.
block instruction	A special class of instruction used within ladder-diagram programming to allow flowchart-like coding, which is often difficult to write with ladder diagrams. Function codes for block instructions are indicated between pointed parentheses <like this>.
block program	A section of program written within a ladder diagram but based on block instructions. Block programs can also contain some, but not all, of the ladder-diagram instructions.
buffer	A temporary storage space for data in a computerized device.
building-block PC	A PC that is constructed from individual components, or “building blocks.” With building-block PCs, there is no one Unit that is independently identifiable as a PC. The PC is rather a functional assembly of components.
bus bar	The line leading down the left and sometimes right side of a ladder diagram. Instruction execution proceeds down the bus bar, which is the starting point for all instruction lines.
call	A process by which instruction execution shifts from the main program to a subroutine. The subroutine may be called by an instruction or by an interrupt.
carry flag	A flag that is used with arithmetic operations to hold a carry from an addition or multiplication operation, or to indicate that the result is negative in a subtraction operation. The carry flag is also used with certain types of shift operations.
clock pulse	A pulse available at a certain bit in memory for use in timing operations. Various clock pulses are available with different pulse widths.
clock pulse bit	A bit in memory that supplies a pulse that can be used to time operations. Various clock pulse bits are available with different pulse widths, and therefore different frequencies.
common data	Data that is stored in the LR Area of a PC and which is shared by other PCs in the same the same system. Each PC has a specified section of the LR Area allocated to it. This allocation is the same in each LR Area of each PC.

condition An message placed in an instruction line to direct the way in which the terminal instructions, on the right side, are to be executed. Each condition is assigned to a bit in memory that determines its status. The status of the bit assigned to each condition determines, in turn, the execution condition for each instruction up to a terminal instruction on the right side of the ladder diagram.

constant An operand for which the actual numeric value is specified by the user, and which is then stored in a particular address in the data memory.

control bit A bit in a memory area that is set either through the program or via a Programming Device to achieve a specific purpose, e.g., a Restart bit is turned ON and OFF to restart a Unit.

Control System All of the hardware and software components used to control other devices. A Control System includes the PC System, the PC programs, and all I/O devices that are used to control or obtain feedback from the controlled system.

controlled system The devices that are being controlled by a PC System.

control signal A signal sent from the PC to effect the operation of the controlled system.

counter A dedicated group of digits or words in memory used to count the number of times a specific process has occurred, or a location in memory accessed through a TC bit and used to count the number of times the status of a bit or an execution condition has changed from OFF to ON.

CPU An acronym for central processing unit. In a PC System, the CPU executes the program, processes I/O signals, communicates with external devices, etc.

CPU Backplane A Backplane which is used to create a CPU Rack.

CPU Rack Part of a building-block PC, the CPU Rack contains the CPU, a power supply, and other Units. With most PCs, the CPU Rack is the only Rack that provides linkable slots.

CTS An acronym for clear-to-send, a signal used in communications between electronic devices to indicate that the receiver is ready to accept incoming data.

cycle The process used to execute a ladder-diagram program. The program is examined sequentially from start to finish and each instruction is executed in turn based on execution conditions.

cycle time The time required for a single cycle of the ladder-diagram program.

data area An area in the PC's memory that is designed to hold a specific type of data, e.g., the LR area is designed to hold common data in a PC Link System. Memory areas that hold programs are not considered data areas.

data area boundary The highest address available within a data area. When designating an operand that requires multiple words, it is necessary to ensure that the highest address in the data area is not exceeded.

data sharing An aspect of PC Link Systems and of Data Links in SYSMAC NET Link Systems in which common data areas or common data words are created between two or more PCs.

debug A process by which a draft program is corrected until it operates as intended. Debugging includes both the removal of syntax errors, as well as the fine-tuning of timing and coordination of control operations.

decimal A number system where all numbers are expressed to the base 10. In a PC all data is ultimately stored in binary form, four binary bits are often used to represent one decimal digit, via a system called binary-coded decimal.

decrement Decreasing a numeric value.

default A value automatically set by the PC when the user omits to set a specific value. Many devices will assume such default conditions upon the application of power.

definer A number used as an operand for an instruction but that serves to define the instruction itself, rather that the data on which the instruction is to operate. Definers include jump numbers, subroutine numbers, etc.

delay In tracing, a value that specifies where tracing is to begin in relationship to the trigger. A delay can be either positive or negative, i.e., can designate an offset on either side of the trigger.

destination The location where an instruction is to place the data on which it is operating, as opposed to the location from which data is taken for use in the instruction. The location from which data is taken is called the source.

differentiated instruction An instruction that is executed only once each time its execution condition goes from OFF to ON. Nondifferentiated instructions are executed each cycle as long as the execution condition stays ON.

differentiation instruction An instruction used to ensure that the operand bit is never turned ON for more than one cycle after the execution condition goes either from OFF to ON for a Differentiate Up instruction or from ON to OFF for a Differentiate Down instruction.

digit A unit of storage in memory that consists of four bits.

digit designator An operand that is used to designate the digit or digits of a word to be used by an instruction.

distributed control An automation concept in which control of each portion of an automated system is located near the devices actually being controlled, i.e., control is decentralized and 'distributed' over the system. Distributed control is one of the fundamental concepts of PC Systems.

DM area A data area used to hold only word data. Words in the DM area cannot be accessed bit by bit.

download The process of transferring a program or data from a higher-level computer to a lower-level computer or PC.

Duplex CPU A CPU arrangement available for a C2000H PC in which there are actually two CPUs. Each of the CPUs holds the same program and the same data. One of the CPUs is active and controls current PC operation; the other CPU serves as a backup and takes over PC operation if the active CPU fails.

Duplex System A C2000H PC System that uses a Duplex CPU.

Duplex Unit The Unit that coordinates the CPU activities of a Duplex System.

electrical noise Random variations of one or more electrical characteristics such as voltage, current, and data, which might interfere with the normal operation of a device.

error code A numeric code generated to indicate that an error exists, and something about the nature of the error. Some error codes are generated by the system; others are defined in the program by the operator.

exclusive OR A logic operation whereby the result is true if one, and only one, of the premises is true. In ladder-diagram programming the premises are usually the ON/OFF states of bits, or the logical combination of such states, called execution conditions.

exclusive NOR A logic operation whereby the result is true if both of the premises are true or both of the premises are false. In ladder-diagram programming the premises are usually the ON/OFF states of bits, or the logical combination of such states, called execution conditions.

exection condition The ON or OFF status under which an instruction is executed. The execution condition is determined by the logical combination of conditions on the same instruction line and up to the instruction currently being executed.

execution time The time required for the CPU to execute either an individual instruction or an entire program.

Expansion I/O Backplane A Backplane which is used to create an Expansion I/O Rack.

Expansion I/O Rack Part of a building-block PC, an Expansion I/O Rack is connected to either a CPU Rack or another Expansion I/O Rack to increase the number of slots available for mounting Units.

extended counter A counter created in a program by using two or more count instructions in succession. Such a counter is capable of counting higher than any of the standard counters provided by the individual instructions.

extended timer A timer created in a program by using two or more timers in succession. Such a timer is capable of timing longer than any of the standard timers provided by the individual instructions.

Factory Intelligent Terminal A programming device provided with advanced programming and debugging capabilities to facilitate PC operation. The Factory Intelligent Terminal also provides various interfaces for external devices, such as floppy disk drives.

fatal error An error that stops PC operation and requires correction before operation can continue.

FIT Abbreviation for Factory Intelligent Terminal.

flag A dedicated bit in memory that is set by the system to indicate some type of operating status. Some flags, such as the carry flag, can also be set by the operator or via the program.

flicker bit A bit that is programmed to turn ON and OFF at a specific frequency.

floating point decimal A decimal number expressed as a number between 0 and 1 (the mantissa) multiplied by a power of 10, e.g., 0.538×10^{-5}.

Floppy Disk Interface Unit A Unit used to interface a floppy disk drive to a PC so that programs and/or data can be stored on floppy disks.

FM area A memory area located in a File Memory Unit used to store or backup programs and/or data.

force reset The process of forcibly turning OFF a bit via a programming device. Bits are usually turned OFF as a result of program execution.

force set The process of forcibly turning ON a bit via a programming device. Bits are usually turned ON as a result of program execution.

function code A two-digit number used to input an instruction into the PC.

GPC Acronym for Graphic Programming Console.

Graphic Programming Console A programming device with advanced programming and debugging capabilities to facilitate PC operation. A Graphic Programming Console is provided with a large display onto which ladder-diagram programs can be written directly in ladder-diagram symbols for input into the PC without conversion to mnemonic form.

hardware error An error originating in the hardware structure (electronic components) of the PC, as opposed to a software error, which originates in software (i.e., programs).

hexadecimal A number system where all numbers are expressed to the base 16. In a PC all data is ultimately stored in binary form, however, displays and inputs on Programming Devices are often expressed in hexadecimal to simplify operation. Each group of four binary bits is numerically equivalent to one hexadecimal digit.

Host Link System A system with one or more host computers connected to one or more PCs via Host Link Units so that the host computer can be used to transfer data to and from the PC(s). Host Link Systems enable centralized management and control of PC Systems.

Host Link Unit An interface used to connect a PC to a host computer in a Host Link System.

host computer A computer that is used to transfer data or programs to from a PC in a Host Link System. The host computer is used for data management and overall system control. Host computers are generally personal or business computers.

HR area A data area used to store and manipulate data, and to preserve data when power to the PC is turned OFF.

increment Increasing a numeric value.

indirect address An address whose contents indicates another address. The contents of the second address will be used as the operand. Indirect addressing is possible in the DM area only .

initialization error An error that occurs either in hardware or software during the PC System startup, i.e., during initialization.

initialize Part of the startup process whereby some memory areas are cleared, system setup is checked, and default values are set.

input The signal coming from an external device into the PC. The term input is often used abstractly or collectively to refer to incoming signals.

input bit A bit in the IR area that is allocated to hold the status of an input.

input device An external device that sends signals into the PC System.

input point The point at which an input enters the PC System. Input points correspond physically to terminals or connector pins.

input signal A change in the status of a connection entering the PC. Generally an input signal is said to exist when, for example, a connection point goes from low to high voltage or from a nonconductive to a conductive state.

instruction A direction given in the program that tells the PC of an action to be carried out, and which data is to be used in carrying out the action. Instructions can be used to simply turn a bit ON or OFF, or they can perform much more complex actions, such as converting and/or transferring large blocks of data.

instruction block A group of instructions that is logically related in a ladder-diagram program. Although any logically related group of instructions could be called an instruction block, the term is generally used to refer to blocks of instructions called logic blocks that require logic block instructions to relate them to other instructions or logic blocks.

instruction execution time The time required to execute an instruction. The execution time for any one instruction can vary with the execution conditions for the instruction and the operands used within it.

instruction line A group of conditions that lie together on the same horizontal line of a ladder diagram. Instruction lines can branch apart or join together to form instruction blocks.

interface An interface is the conceptual boundary between systems or devices and usually involves changes in the way the communicated data is represented. Interface devices such as NSBs perform operations like changing the coding, format, or speed of the data.

interlock A programming method used to treat a number of instructions as a group so that the entire group can be reset together when individual execution is not required. An interlocked program section is executed normally for an ON execution condition and partially reset for an OFF execution condition.

interrupt (signal) A signal that stops normal program execution and causes a subroutine to be run.

Interrupt Input Unit A Rack-mounting Unit used to input external interrupts into a PC System.

I/O capacity The number of inputs and outputs that a PC is able to handle. This number ranges from around one hundred for smaller PCs to two thousand for the largest ones.

I/O Control Unit A Unit mounted to the CPU Rack in certain PCs to monitor and control I/O points on Expansion I/O Units.

I/O devices The devices to which terminals on I/O Units or Special I/O Units, or other Units are connected. I/O devices may be either part of the Control System, if they function to help control other devices, or they may be part of the controlled system.

I/O Interface Unit A Unit mounted to an Expansion I/O Rack in certain PCs to interface the Expansion I/O Rack to the CPU Rack.

I/O Link Created in an Optical Remote I/O System to enable input/output of one or two IR words directly between PCs. The words are input/output between the PC controlling the Master and a PC connected to the Remote I/O System through an I/O Link Unit or an I/O Link Rack.

I/O Link Unit A Unit used with certain PCs to create an I/O Link in an Optical Remote I/O System.

I/O point The place at which an input signal enters the PC System, or at which an output signal leaves the PC System. In physical terms, I/O points correspond to terminals or connector pins on a Unit; in terms of programming, an I/O points correspond to I/O bits in the IR area.

I/O response time The time required for an output signal to be sent from the PC in response to an input signal received from an external device.

I/O table A table created within the memory of the PC that lists the IR area words allocated to each Unit in the PC System. The I/O table can be created by, or modified from, a Programming Device.

I/O Unit The most basic type of Unit mounted to a backplane to create a Rack. I/O Units include Input Units and Output Units, each of which is available in a range of specifications. I/O Units do not include Special I/O Units, Link Units, etc.

I/O word A word in the IR area that is allocated to a Unit in the PC System.

IR area A data area whose principal function is to hold the status of inputs coming into the system and that of outputs that are to be set out of the system. Bits and words in the IR that are used this way are called I/O bits and I/O words. The remaining bits in the IR area are work bits.

JIS Acronym for Japanese Industrial Standards.

jump A type of programming where execution moves directly from one point in a program to another, without sequentially executing any instructions inbetween. Jumps are usually conditional on an execution condition.

jump number A definer used with a jump that defines the points from and to which a jump is to be made.

ladder diagram (program) A form of program arising out of relay-based control systems that uses circuit-type diagrams to represent the logic flow of programming instructions. The appearance of the program is similar to a ladder, and thus the name.

ladder diagram symbol A symbol used in a ladder-diagram program.

ladder instruction An instruction that represents the 'rung' portion of a ladder-diagram program. The other instructions in a ladder diagram fall along the right side of the diagram and are called terminal instructions.

Ladder Support Software A software package that provides most of the functions of the Factory Intelligent Terminal on an IBM AT, IBM XT, or compatible computer.

LAN An acronym for local area network.

leftmost (bit/word) The highest numbered bits of a group of bits, generally of an entire word, or the highest numbered words of a group of words. These bits/words are often called most-significant bits/words.

Link Adapter A Unit used to connect communications lines, either to branch the lines or to convert between different types of cable. There are two types of Link Adapter: Branching Link Adapters and Converting Link Adapters.

link A hardware or software connection formed between two Units. "Link" can refer either to a part of the physical connection between two Units (e.g., optical links in Wired Remote I/O Systems) or a software connection created to data existing at another location (Network Data Links).

linkable slot A slot on either a CPU or Expansion I/O Backplane to which a Link Unit can be mounted. Backplanes differ in the slots to which Link Units can be mounted.

Link System A system that includes one or more of the following systems: Remote I/O System, PC Link System, Host Link System, or SYSMAC NET Link System.

Link Unit Any of the Units used to connect a PC to a Link System. These are Remote I/O Units, I/O Link Units, PC Link Units, Host Link Units, and SYSMAC NET Link Units.

load The processes of copying data either from an external device or from a storage area to an active portion of the system such as a display buffer. Also, an output device connected to the PC is called a load.

local area network A network consisting of nodes or positions in a loop arrangement. Each node can be any one of a number of devices, which can transfer data to and from each other.

logic block A group of instructions that is logically related in a ladder-diagram program and that requires logic block instructions to relate it to other instructions or logic blocks.

logic block instruction An instruction used to locally combine the execution condition resulting from a logic block with a current execution condition. The current execution condition could be the result of a single condition, or of another logic block. AND Load and OR Load are the two logic block instructions.

logic instruction Instructions used to logically combine the content of two words and output the logical results to a specified result word. The logic instructions combine all the same-numbered bits in the two words and output the result to the bit of the same number in the specified result word.

loop A group of instructions that can be executed more than once in succession (i.e., repeated) depending on an execution condition or bit status.

LR area A data area that is used in a PC Link System so that data can be transferred between two or more PCs. If a PC Link System is not used, the LR area is available for use as work bits.

LSS Abbreviation for Ladder Support Software.

main program All of a program except for the subroutines.

masking 'Covering' an interrupt signal so that the interrupt is not effective until the mask is removed.

Master Short for Remote I/O Master Unit.

memory area Any of the areas in the PC used to hold data or programs.

mnemonic code A form of a ladder-diagram program that consists of a sequential list of the instructions without using a ladder diagram. Mnemonic code is required to input a program into a PC when using a Programming Console.

MONITOR mode A mode of PC operation in which normal program execution is possible, and which allows modification of data held in memory. Used for monitoring or debugging the PC.

most-significant (bit/word) See *leftmost (bit/word).*

NC input An input that is normally closed, i.e., the input signal is considered to be present when the circuit connected to the input opens.

nest Programming one loop within another loop, programming a call to a subroutine within another subroutine, or programming an IF-ELSE programming section within another IF-ELSE section.

Network Service Board A device with an interface to connect devices other than PCs to a SYSMAC NET Link System.

Network Service Unit A Unit that provides two interfaces to connect peripheral devices to a SYSMAC NET Link System.

node One of the positions in a LAN. Each node incorporates a device that can communicate with the devices at all of the other nodes. The device at a node is identified by the node number. One loop of a SYSMAC NET Link System (OMRON's LAN) can consist of up to 126 nodes. Each node is occupied by a SYSMAC NET Link Unit mounted to a PC or a device providing an interface to a computer or other peripheral device.

NO input An input that is normally open, i.e., the input signal is considered to be present when the circuit connected to the input closes.

noise interference Disturbances in signals caused by electrical noise.

nonfatal error A hardware or software error that produces a warning but does not stop the PC from operating.

normally closed condition A condition that produces an ON execution condition when the bit assigned to it is OFF, and an OFF execution condition when the bit assigned to it is ON.

normally closed condition A condition that produces an ON execution condition when the bit assigned to it is ON, and an OFF execution condition when the bit assigned to it is OFF.

NOT A logic operation which inverts the status of the operand. For example, AND NOT indicates an AND operation with the opposite of the actual status of the operand bit.

NSB An acronym for Network Service Board.

NSU An acronym for Network Service Unit.

OFF The status of an input or output when a signal is said not to be present. The OFF state is generally represented by a low voltage or by non-conductivity, but can be defined as the opposite of either.

OFF delay The delay between the time when a signal is switched OFF (e.g., by an input device or PC) and the time when the signal reaches a state readable as an OFF signal (i.e., as no signal) by a receiving party (e.g., output device or PC).

ON The status of an input or output when a signal is said to be present. The ON state is generally represented by a high voltage or by conductivity, but can be defined as the opposite of either.

ON delay The delay between the time when an ON signal is initiated (e.g., by an input device or PC) and the time when the signal reaches a state readable as an ON signal by a receiving party (e.g., output device or PC).

one-shot bit A bit that is turned ON or OFF for a specified interval of time which is longer than one cycle.

on-line removal Removing a Rack-mounted Unit for replacement or maintenance during PC operation.

operand Bit(s) or word(s) designated as the data to be used for an instruction. An operand can be input as a constant expressing the actual numeric value to be used or as an address to express the location in memory of the data to be used.

operand bit A bit designated as an operand for an instruction.

operand word A word designated as an operand for an instruction.

operating error An error that occurs during actual PC operation as opposed to an initialization error, which occurs before actual operations can begin.

Optical I/O Unit A Unit that is connected in an Optical Remote I/O System to provide 8 I/O points. Optical I/O Units are not mounted to a Rack.

Optical Slave Rack A Slave Rack connected through an Optical Remote I/O Slave Unit.

OR A logic operation whereby the result is true if either of two premises is true, or if both are true. In ladder-diagram programming the premises are usually ON/OFF states of bits or the logical combination of such states called execution conditions.

output The signal sent from the PC to an external device. the term output is often used abstractly or collectively to refer to outgoing signals.

output bit A bit in the IR area that is allocated to hold the status to be sent to an output device.

output device An external device that receives signals from the PC System.

output point The point at which an output leaves the PC System. Output points correspond physically to terminals or connector pins.

output signal A signal being sent to an external device. Generally an output signal is said to exist when, for example, a connection point goes from low to high voltage or from a nonconductive to a conductive state.

overseeing Part of the processing performed by the CPU that includes general tasks required to operate the PC.

overwrite Changing the content of a memory location so that the previous content is lost.

parity Adjustment of the number of ON bits in a word or other unit of data so that the total is always an even number or always an odd number. Parity is generally used to check the accuracy of data after being transmitted by confirming that the number of ON bits is still even or still odd.

PC An acronym for Programmable Controller.

PCB An acronym for printed circuit board.

PC configuration The arrangement and interconnections of the Units that are put together to form a functional PC.

PCF Acronym for plastic-clad optical fiber cable.

PC Link System A system in which PCs are connected through PC Link Units to enable them to share common data areas, i.e., each of the PCs writes to certain words in the LR area and receives the data of the words written by all other PC Link Units connected in series with it.

PC Link Unit The Unit used to connect PCs in a PC Link System.

PC System With building-block PCs, all of the Racks and independent Units connected directly to them up to, but not including the I/O devices. The boundaries of a PC System are the PC and the program in its CPU at the upper end; and the I/O Units, Special I/O Units, Optical I/O Units, Remote Terminals, etc., at the lower end.

peripheral device Devices connected to a PC System to aid in system operation. Peripheral devices include printers, programming devices, external storage media, etc.

port A connector on a PC or computer that serves as a connection to an external device.

present value The current value registered in a device at any instant during its operation. Present value is abbreviated as PV.

printed circuit board A board onto which electrical circuits are printed for mounting into a computer or electrical device.

Printer Interface Unit A Unit used to interface a printer so that ladder diagrams and other data can be printed out.

program The list of instructions that tells the PC the sequence of control actions to be carried out.

Programmable Controller A computerized device that can accept inputs from external devices and generate outputs to external devices according to a program held in memory. Programmable Controllers are used to automate control of external devices. Although single-component Programmable Controllers are available, building-block Programmable Controllers are constructed from separate components. Such building-block Programmable Controllers are formed only when enough of these separate components are assembled to form a functional assembly, i.e., no one individual Unit is called a PC.

programmed alarm An alarm given as a result of execution of an instruction designed to generate the alarm in the program, as opposed to one generated by the system.

programmed error An error arising as a result of the execution of an instruction designed to generate the error in the program, as opposed to one generated by the system.

programmed message A message generated as a result of execution of an instruction designed to generate the message in the program, as opposed to one generated by the system.

Programming Console The simplest form or programming device available for a PC. Programming Consoles are available both as hand-held models and as CPU-mounting models.

Programming Device A peripheral device used to input a program into a PC or to alter or monitor a program already held in the PC. There are dedicated programming devices, such as Programming Consoles, and there are non-dedicated devices, such as a host computer.

PROGRAM mode A mode of operation that allows inputting and debugging of programs to be carried out, but that does not permit normal execution of the program.

PROM Writer A peripheral device used to write programs and other data into a ROM for permanent storage and application.

prompt A message or symbol that appears on a display to request input from the operator.

PV Acronym for present value.

Rack An assembly of various Units on a Backplane that forms a functional unit in a building-block PC System. Racks include CPU Racks, Expansion I/O Racks, I/O Racks, and Slave Racks.

refresh The process of updating output status sent to external devices so that it agrees with the status of output bits held in memory and of updating input bits in memory so that they agree with the status of inputs from external devices.

relay-based control The forerunner of PCs. In relay-based control, groups of relays are interconnected to form control circuits. In a PC, these are replaced by programmable circuits.

Remote I/O Master Unit The Unit in a Remote I/O System through which signals are sent to all other Remote I/O Units. The Remote I/O Master Unit is mounted either to a CPU Rack or an Expansion I/O Rack connected to the CPU Rack. Remote I/O Master Unit is generally abbreviated to Master.

Remote I/O Slave Unit A Unit mounted to a Backplane to form a Slave Rack. Remote I/O Slave Unit is generally abbreviated to Slave.

Remote I/O System A system in which remote I/O points are controlled through a Master mounted to a CPU Rack or an Expansion I/O Rack connected to the CPU Rack.

Remote I/O Unit Any of the Units in a Remote I/O System. Remote I/O Units include Masters, Slaves, Optical I/O Units, I/O Link Units, and Remote Terminals.

remote I/O word An I/O word allocated to a Unit in a Remote I/O System.

reset The process of turning a bit or signal OFF or of changing the present value of a timer or counter to its set value or to zero.

return The process by which instruction execution shifts from a subroutine back to the main program (usually the point from which the subroutine was called).

reversible counter A counter that can be both incremented and decremented depending on the specified conditions.

reversible shift register A shift register that can shift data in either direction depending on the specified conditions.

right-hand instruction Another term for terminal instruction.

rightmost (bit/word) The lowest numbered bits of a group of bits, generally of an entire word, or the lowest numbered words of a group of words. These bits/words are often called least-significant bits/words.

rotate register A shift register in which the data moved out from one end is placed back into the shift register at the other end.

RUN mode The operating mode used by the PC for normal control operations.

scheduled interrupt An interrupt that is automatically generated by the system at a specific time or program location specified by the operator. Scheduled interrupts result in the execution of specific subroutines that can be used for instructions that must be executed repeatedly for a specified period of time.

self diagnosis A process whereby the system checks its own operation and generates a warning or error if an abnormality is discovered.

self-maintaining bit A bit that is programmed to maintain either an OFF or ON status until set or reset by specified conditions.

servicing The process whereby the PC provides data to or receives data from external devices or remote I/O Units, or otherwise handles data transactions for Link Systems.

set The process of turning a bit or signal ON.

set value The value from which a decrementing counter starts counting down or to which an incrementing counter counts up (i.e., the maximum count), or the time from which or for which a timer starts timing. Set value is abbreviated SV.

shift register One or more words in which data is shifted a specified number of units to the right or left in bit, digit, or word units. In a rotate register, data shifted out one end is shifted back into the other end. In other shift registers, new data (either specified data, zero(s) or one(s)) is shifted into one end and the data shifted out at the other end is lost.

Simplex CPU A C2000H PC that uses only a single CPU as opposed to a Duplex CPU. This term is meaningless with other PCs, which are only available with simplex operation.

Simplex System A C2000H PC System that uses a Simplex CPU.

Slave Short for Remote I/O Slave Unit.

Slave Rack A Rack containing a Remote I/O Slave Unit and controlled through a Remote I/O Master Unit. Slave Racks are generally located away from the CPU Rack.

slot A position on a Rack (Backplane) to which a Unit can be mounted.

software error An error that originates in a software program.

software protect A means of protecting data from being changed that uses software as opposed to a physical switch or other hardware setting.

source The location from which data is taken for use in an instruction, as opposed to the location to which the result of an instruction is to be written. The latter is called the destination.

Special I/O Unit A dedicated Unit that is designed for a specific purpose. Special I/O Units include Position Control Units, High-Speed Counter Units, Analog I/O Units, ASCII Units, Ladder Diagram I/O Units, etc.

SR area A data area in a PC used mainly for flags, control bits, and other information provided about PC operation. The status of only certain SR bits may be controlled by the operator, i.e., most SR bits can only be read.

subroutine A group of instructions placed after the main program and executed only if called from the main program or activated by an interrupt.

subroutine number A definer used to identify the subroutine that a subroutine call or interrupt activates.

SV Abbreviation for set value.

switching capacity The maximum voltage/current that a relay can safely switch on and off.

syntax error An error in the way in which a program is written. Syntax errors can include 'spelling' mistakes (i.e., a function code that does not exist), mistakes in specifying operands within acceptable parameters (e.g., specifying reserved SR bits as a destination), and mistakes in actual application of instructions (e.g., a call to a subroutine that does not exist).

SYSMAC NET Link System An optical LAN formed from PCs and other devices connected through SYSMAC NET Link Units, NSBs, and NSUs. A SYSMAC NET Link System also normally contains nodes interfacing computers and other peripheral devices. PCs in the SYSMAC NET Link System can pass data back and forth, receive commands from any interfaced computer, and share any interfaced peripheral device.

SYSMAC NET Link Unit The Unit used to connect PCs to a SYSMAC NET Link System.

system configuration The arrangement in which Units in a system are connected.

system error An error generated by the system, as opposed to one resulting from execution of an instruction designed to generate an error.

system error message An error message generated by the system, as opposed to one resulting from execution of an instruction designed to generate a message.

TC area A data area that can be used only for timers and counters. Each bit in the TC area serves as the access point for the SV, PV, and Completion flag for the timer or counter defined with that bit.

TC number A definer that corresponds to a bit in the TC area and used to define the bit as either a timer or a counter.

terminal instruction An instruction placed on the right side of a ladder diagram that uses the final execution conditions of an instruction line.

terminator The code comprising an asterisk and a carriage return (* CR) which indicates the end of a block of data, whether it is a single-frame or multi-frame block. Frames within a multi-frame block are separated by delimiters.

timer A location in memory accessed through a TC bit and used to time down from the timer's set value. Timers are turned ON and reset according to their execution conditions.

TM area A memory area used to store the results of a trace.

transmission distance The distance that a signal can be transmitted.

TR area A data area used to store execution conditions so that they can be reloaded later for use with other instructions.

trace An operation whereby the program is executed and the resulting data is stored in TM memory to enable step-by-step analysis and debugging.

transfer The process of moving data from one location to another within the PC, or between the PC and external devices. When data is transferred, generally a copy of the data is sent to the destination, i.e., the content of the source of the transfer is not changed.

trigger address An address in the program that defines the beginning point for tracing. The actual beginning point can be altered from the trigger by defining either a positive or negative delay.

UM area The memory area used to hold the active program, i.e., the program that is being currently executed.

Unit In OMRON PC terminology, the word Unit is capitalized to indicate any product sold for a PC System. Though most of the names of these products end with the word Unit, not all do, e.g., a Remote Terminal is referred to in a collective sense as a Unit. Context generally makes any limitations of this word clear.

unit number A number assigned to some Link Units and Special I/O Units to facilitate identification when assigning words or other operating parameters to it.

watchdog timer A timer within the system that ensures that the cycle time stays within specified limits. When limits are reached, either warnings are given or PC operation is stopped depending on the particular limit that is reached.

Wired Slave Rack A Slave Rack connected through a Wired Remote I/O Slave Unit.

word A unit of data storage in memory that consists of 16 bits. All data areas consists of words. Some data areas can be accessed only by words; others, by either words or bits.

word address The location in memory where a word of data is stored. A word address must specify (sometimes by default) the data area and the number of the word that is being addressed.

word multiplier A value between 0 and 3 that is assigned to a Master in a Remote I/O System so that words can be allocated to non-Rack-mounting Units within the System. The word setting made on the Unit is added to 32 times the word multiplier to arrive at the actual word to be allocated.

work bit A bit in a work word.

work word A word that can be used for data calculation or other manipulation in programming, i.e., a ‘work space’ in memory. A large portion of the IR area is always reserved for work words. Parts of other areas not required for special purposes may also be used as work words, e.g., LR words not used in a PC Link or SYSMAC NET Link System.

Index

A

address, in data area, 17

address tracing. *See* tracing, data tracing.

AR area, 31–36

arithmetic flags, 103

arithmetic operations, flags, 31

ASCII, converting data, 157

assembly tool, 300

B

backup
- DM area data, 280
- program, 277–278

battery, Low Battery Flag, 29

BCD
- calculations, 158–173
- converting, 17
- definition, 17

binary
- calculations, 174
- definition, 17

bits
- controlling, 109
- force set/reset, 255
- monitoring, 252–255

block instructions, converting to mnemonic code, 107

block programs
- instructions, 190–199
- restricted instructions, 190

branching
- block programs, 191
- exiting block programs, 197

C

cassette tape operation, 276–284
- comparing Program Memory data, 278–280
- error messages, 277
- restoring Program Memory data, 278–280
- saving Program Memory data, 277

channel. *See* word

clock pulse bits, 29

comparing Program Memory data, 278–280

constants, operands, 103

control bit
- Data Retention, 28
- data tracing, 34
- definition, 16
- File Memory, 34
- manipulating, 23
- Output OFF, 29

Control System, definition, 3

controlled system, definition, 3

counters
- bits in TC area, 37
- block programs, 196
- changing SV, 263
- conditions when reset, 122, 126
- creating extended timers, 124
- extended, 123
- inputting SV, 76
- Power-off, 35
- reversible, 125

CPU
- Device Mounted Flag, 36
- operational flow, 229

CPU indicators, 10

CPU Rack, definition, 12

cycle, First Cycle Flag, 29

cycle time, 228–232
- calculating, 232–234
- Cycle Time Indicators, 36
- displaying on Programming Console, 81

Cycle Time Error Flag, 29

D

data
- comparison instructions, 142–147
- converting, 18, 148–158
- decrementing, 159
- incrementing, 159
- modifying, 260
- modifying binary, 262
- modifying hex/BCD, 257
- moving, 135–141

data areas
- definition, 15
- structure, 16

data retention
- control bit, 28
- in AR area, 31
- in DM area, 36
- in HR area, 37
- in IR area, 18
- in LR area, 38
- in SR area, 23
- in TC area, 37
- in TR area, 39

data tracing, 211–214
  control bits and flags, 34

debugging, 244–251
  address trace read, 250–251
  address tracing, 249–250
  execution by instruction address, 246
  execution from Programming Console, 248
  Programming Console debug operation, 245

decrementing, 159

definers, definition, 102

delay, address tracing, 249

differentiated instructions, 104
  function codes, 102

digit numbers, 17–18

displays
  converting between hex and ASCII, 258
  I/O Unit designations, 73
  Programming Console, English/Japanese switch, 61

DM area, 36–37
  saving, restoring, and comparing, 280–284

Duplex System, flags, 30

Duplex Unit, indicators, 11

E

ER. *See* flag, Instruction Execution Error

error codes, programming, 208

error messages, programming, 209

errors
  cassette tape operations, 277
  clearing messages, 66
  displaying and clearing, 244
  fatal, 288
  initialization, 287
  Instruction Execution Error Flag, 30
  message tables, 286–289
  messages when inputting programs, 78
  non-fatal, 287
  programming indications, 286
  programming messages, 209
  reading and clearing messages, 286
  resetting, 209
  SR and AR area flags, 289
  troubleshooting, 290

execution condition, definition, 44

execution time
  instructions, 235–240
  program, 230

Expansion I/O Rack, definition, 12

F

Factory Intelligent Terminal. *See* peripheral devices

FAL area, 29, 208

FAL code, FALS-generating Address, 36

fatal operating errors, 288

File Memory, 39
  clearing, 266–267
  edit, 275
  flags and control bits, 34
  instructions, 214–217
  Programming Console operations, 266–275
  read, 272–275
  read/write, 275
  verify, 270
  write, 267

FIT. *See* peripheral devices

flag
  AR and SR area errors, 289
  arithmetic, 31
    programming example, 142
  CPU-mounting Device, 36
  CY
    clearing, 159
    setting, 159
  Cycle Time Error, 29
  Data Tracing, 34
  definition, 16
  Duplex System, 30
  File Memory, 34
  First Cycle, 29
  I/O Verification Error, 29
  Instruction Execution Error, 30
  Link Units, 36
  Low Battery, 29
  on-line removal of Units, 35
  Step, 30

floating-point decimal, division, 169

Floppy Disk Interface Unit. *See* peripheral devices

force setting/resetting, 255

function codes, 102

G

GPC. *See* peripheral devices

Graphic Programming Console. *See* peripheral devices

H

Hard-plastic-clad Quartz Fiber: H-PCF
  cables, 299
  cords, 299

hexadecimal, definition, 17

Host Link Systems, error bits and flags, 25

HR area, 37

I

I/O bit
  definition, 18
  limits, 18

I/O points, refreshing, 211

I/O Rack, definition, 12

I/O refreshing, time required, 230

I/O response times, 241

I/O table
  creating, 19
  reading, 71
  registering I/O words, 67
  registration, 64
  transferring, 67
  verification, 70
  Verification Error Flag, 29

I/O Units
  See also Units
  changes on-line, 69
  numbering for I/O table, 72

I/O words
  allocation, 19
  definition, 18
  limits, 18
  number required by Units, 19
  reserving in I/O table, 22

incrementing, 159

indicators, Duplex Unit, 11

indirect addressing, 37, 103

input bit
  application, 19
  definition, 3

input device, definition, 3

input point, definition, 3

input signal, definition, 3

instruction set
  ADB(50), 174
  ADD(30), 160
  ADDL(54), 161
  AND, 46, 108
    combining with OR, 47
  AND LD, 49, 109
    combining with OR LD, 51
    use in logic blocks, 50
  AND NOT, 46, 108
  ANDW(34), 180
  ASC(86), 157
  ASL(25), 131
  ASR(26), 132
  BCD(24), 149
  BCDL(59), 150
  BCMP(68), 145
  BCNT(67), 210
  BEND<01>, 190
  BIN(23), 148
  BINL(58), 148
  BPPS<11>, 198
  BPRG(96), 190
  BPRS<12>, 198
  BSET(71), 136
  CLC(41), 159
  CMP(20), 142
  CNT, 122
  CNTR(12), 125
  CNTW<14>, 196
  COLL(81), 139
  COM(29), 179
  DEC(39), 159
  DIFD(14), 92, 110–111
    using in interlocks, 114
    using in jumps, 116
  DIFU(13), 92, 110–111
    using in interlocks, 114
    using in jumps, 116
  DIST(80), 139
  DIV(33), 167
  DIVL(57), 168
  DMPX(77), 152
  DVB(53), 179
  ELSE<03>, 191
  END(01), 48, 107, 116
  execution times, 235–240
  EXIT<06>, 197
  EXIT<06> NOT, 197
  FAL(06), 208
  FALS(07), 208
  FDIV(79), 169
  FILP(44), 216
  FILR(42), 215
  FILW(43), 216
  IEND<04>, 191
  IF<02>, 191
  IF<02> NOT, 191
  IL(02), 88, 113–115
  ILC(03), 88, 113–115
  INC(38), 159
  INT(89), 185
  IORF(97), 211
  JME(05), 115
  JMP(04), 115
  JMP(04) and JME(05), 90
  KEEP(11), 112
    in controlling bit status, 92
  ladder instructions, 45
  LD, 45, 108
  LD NOT, 45, 108
  LEND<10>, 197
  LEND<10> NOT, 197
  LOOP<09>, 197
  MLB(52), 178
  MLPX(76), 150
  MOV(21), 135
  MOVB(82), 140
  MOVD(83), 141
  MSG(46), 209
  MUL(32), 165
  MULL(56), 166
  MVN(22), 136
  NOP(00), 116
  NOT, 43
  operands, 42

OR, 46, 108
combining with AND, 47
OR LD, 50, 109
combining with AND LD, 51
use in logic blocks, 51
OR NOT, 46, 108
ORW(35), 180
OUT, 48, 109
OUT NOT, 48, 109
READ(88), 218
RECV(98), 221
RET(93), 183
ROL(27), 132
ROOT(72), 172
ROR(28), 133
RSET<08>, 191
SBB(51), 176
SBN(92), 183
SBS(91), 183
SDEC(78), 154
SEND(90), 219
SET<07>, 191
SFT(10), 127
SFTR(84), 129
SLD(74), 133
SNXT(09), 199
SRD(75), 134
STC(40), 159
STEP(08), 199
SUB(31), 162
SUBL(55), 164
TCMP(85), 146
terminology, 42
TIM, 118
TIMH(15), 121
TIMW<13>, 195
TMHW<15>, 195
TRSM(45), 211
WAIT<05>, 193
WDT(94), 211
WRIT(87), 218
WSFT(16), 134
XCHG(73), 138
XFER(70), 138
XNRW(37), 182
XORW(36), 181

instructions, designations when inputting, 77

intelligent I/O instructions, 217–219

interlocks, 113–115
converting to mnemonic code, 114
using self-maintaining bits in, 93

Interrupt Input Units, 186–187

interrupts, 182
clearing, 186
control, 185
masking/unmasking, 186
priority, 189
scheduled, 187–188
example, 188

IR area, 18–23

J

jump numbers, 115

jumps, 115–116

L

ladder diagram
branching, 86
IL(02) and ILC(03), 88
using TR bits, 86
controlling bit status
using DIFU(13) and DIFD(14), 92, 110–111
using KEEP(11), 112–117
using OUT and OUT NOT, 48
converting to mnemonic code, 44–58
display via GPC, FIT, or LSS, 43
instructions
combining, AND LD and OR LD, 51
controlling bit status
using KEEP(11), 92
using OUT and OUT NOT, 109
format, 102
notation, 102
structure, 43
using logic blocks, 49

ladder diagram instructions, 107–109

Ladder Support Software. *See* peripheral devices

LEDs. *See* CPU indicators

leftmost, definition, 17

Link System, flags and control bits, 24–28

Link Systems, servicing, 230

Link Units
See also Units
flags, 36
PC cycle time, 233

logic block instructions, converting to mnemonic code, 49–57

logic blocks. *See* ladder diagram

logic instructions, 179–182

loops, block programs, 197

LR area, 38–39

LSS. *See* peripheral devices

M

memory areas
clearing, 62
definition, 15

messages, programming, 209

mnemonic code, converting, 44–58

modifying data, hex/binary, 257

monitoring
binary, 261
three words, 259

mounting Units, location, 13

N

nesting, subroutines, 184
non-fatal operating errors, 287
normally closed condition, definition, 43
NOT, definition, 43

O

on-line I/O Unit changes, 69
on-line removal, flags and other information, 35
operand bit, 44
operands, 102
  allowable designations, 102
  requirements, 102
operating modes, 59
operation, preparations, 61–74
optical connectors, 300
Optical Power Tester, 300
Optical Power Tester Head Unit, 300
output bit
  application, 19
  controlling, via Output OFF bit, 29
  controlling ON/OFF time, 110
  controlling status, 92, 93
  definition, 3
output device, definition, 3
output point, definition, 3
output signal, definition, 3

P

password, entering on Programming Console, 61
PC
  configuration, 12
  definition, 3
PC Link Systems
  error bits and flags, 27–28
  LR area application, 38
peripheral devices, 5
  Factory Intelligent Terminal (FIT), 6
  Floppy Disk Interface Unit, 6
  Graphic Programming Console (GPC), 6
  Ladder Support Software (LSS), 6
  Printer Interface Unit, 6
  Programming Console, 6, 58–61
  PROM Writer, 6
  servicing, 230
power supply, Power-off counter, 35
present value. *See* PV
Printer Interface Unit. *See* peripheral devices
program execution, 97
Program Memory, 39
  backup and restore, 278–280
  setting address and reading content, 75–76
  structure, 44
programming
  backup onto cassette tape, 276–284
  checks for syntax, 79–81
  displaying and clearing error messages, 244
  entering and editing, 76
  example, using shift register, 128
  inputting, modifying and checking, 75–91
  inserting and deleting instructions, 83–85
  jumps, 90
  pausing/restarting block programs, 198
  precautions, 95
  preparing data in data areas, 136
  searching, 82–83
  setting and reading from memory address, 75
  simplification with differentiated instructions, 111
  writing, 42
Programming Console, 58–61
  See also peripheral devices
PROM Writer. *See* peripheral devices
PV
  accessing via PC area, 38
  CNTR(12), 126
  timers and counters, 117

R

Racks, types, 12
Remote I/O Systems
  error bits and flags, 24
  I/O word allocation, 21
remote I/O word, 19
response times, I/O, 241–242
rightmost, definition, 17

S

self-maintaining bits, using KEEP(11), 112
set value. *See* SV
seven-segment displays, converting data, 154
shift registers, 127–135
  controlling individual bits, 128
Special I/O Units. *See* Units
special instructions, 208–211
SR area, 23–31
status indicators. *See* CPU indicators
status retention, Data Retention Control bit, 28
step execution, Step Flag, 30
step instructions, 199–208
subroutine number, 183

subroutines, 182–189

SV
  accessing via TC area, 38
  changing, 263
  CNTR(12), 126
  timers and counters, 117

SYSMAC LINK System
  Active Node Flags, 33
  communications completion code, 25
  data link settings, 33
  data link status, 26
  flags, 25
  instructions, 219
  LR area application, 38
  routing table and monitor timer, 37
  service time, 34

SYSMAC NET Link System
  data link status, 26
  flags, 25
  instructions, 219
  LR area application, 38
  service time, 34

T

TC area, 37–38

TC numbers, 37, 117

Three-word Monitor, 259

timers
  bits in TC area, 37
  block programs, 195
  changing SV, 263
  conditions when reset, 118, 122
  example using CMP(20), 143
  extended, 119
  flicker bits, 120
  inputting SV, 76
  ON/OFF delays, 119
  one-shot bits, 120

TR area, 39

TR bits, use in branching, 86

Trace Memory, 39

tracing
  See also See data tracing and address tracing.
  address trace read, 250–251
  address tracing, 249–250

trigger (address), address tracing, 249

troubleshooting, 290

U

Units
  changing configuration, 22
  definition, 4
  I/O Units, definition, 4
  Link Units, definition, 4
  Special I/O Units, definition, 4
  words required by each, 20

W

watchdog timer, 231
  extending, 211

word bit, definition, 16

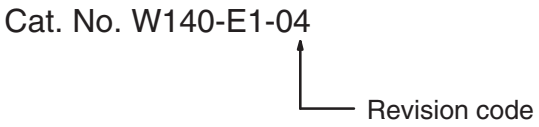
word multiplier, registering in I/O table, 65

words, monitoring, 252

work word, definition, 16

Revision History

A manual revision code appears as a suffix to the catalog number on the front cover of the manual.



The following table outlines the changes made to the manual during each revision. Page numbers refer to the previous version.

Revision code	Date	Revised content
1	——	Original production
2	August 1990	Complete rewrite and reorganization based on resource document 29–93G. Information on SYSMAC LINK Systems added especially in reference to SR and AR bits and SEND(90)/RECV(98).
3	September 1993	The sections have been rearranged for greater clarity and ease of understanding. *Sections 7–1* and *7–2* have been deleted and *Sections 7–3* and *7–4* have been moved to *Sections 4–5* and *4–6*, respectively. Mnemonic codes have been added throughout the manual. Terms have been standardized. The information in *Appendices B* and *C* has been given in greater detail and includes standardized terms. Scan time has been replaced by cycle time throughout the manual. **Page 7:** Catalogue number for the SYSMAC LINK System Manual was corrected. **Page 20:** Model number CT041 has been added to High-speed Counter Units in the table. ID Sensor Unit has been added to the table. **Page 28:** “node” corrected to “data link table entry” in the table. **Page 163:** Step programming diagram clarified. **Page 165:** Step programming diagram clarified and a note has been added. **Page 167:** Step programming diagram clarified. **Pages 299 to 311:** Model numbers have been updated. **Pages 342 to 344 :** Flag bit numbers have been corrected.
3A	October 1994	Caution, Warning, and Danger symbols have been added throughout the manual. **Page 7:** LSS Operation Manual catalog number changed. **Pages 295 to 305:** Model numbers have been updated.
04	May 2003	**Page 69:** Information on changing Units on-line added. **Page 70:** Paragraph at bottom of page removed. **Page 170:** Information on valid ranges added. **Page 209:** “TC” removed from priority list. **Page 288:** One row added to table. **Page 349:** Information on SR bit operation added.

OMRON

OMRON ELECTRONICS LLC

1 Commerce Drive
Schaumburg, IL 60173
847.843.7900
For US technical support or
other inquiries: 800.556.6766

OMRON CANADA, INC.

885 Milner Avenue
Toronto, Ontario M1B 5V8
416.286.6465

OMRON ON-LINE

Global - http://www.omron.com
USA - http://www.omron.com/oei
Canada - http://www.omron.ca

UNITED STATES

To locate a Regional Sales Office, local Distributor or to obtain product information, call: 847.843.7900

CANADA REGIONAL SALES OFFICES

Ontario	Toronto	416.286.6465
	Kitchener	519.896.1144
	Kingston	613.376.3968
Quebec	Montreal	514.636.6676
British Columbia	Vancouver	604.522.8855
Alberta	Edmonton	403.440.0818
	Calgary	403.257.3095

BRAZIL SALES OFFICE

Sao Paulo	55.11.5564.6488

ARGENTINA SALES OFFICE

Cono Sur	54.114.787.1129

MEXICO SALES OFFICES

Florida	954.227.2121	**Ciudad Juarez**	656.623.7083
Mexico, D.F.	555.534.1195	**Monterrey, N.L.**	818.377.4281

W140-E1-4

©2005 OMRON ELECTRONICS LLC
Printed in the U.S.A.

Specifications subject to change without notice.