

**SYSMAC  
CQM1/CPM1/CPM1A/SRM1  
Programmable Controllers**

**PROGRAMMING MANUAL**

**OMRON**

# **CQM1/CPM1/CPM1A/SRM1 Programmable Controllers**


## **Programming Manual**


*Revised May 1999*


## Notice:

OMRON products are manufactured for use according to proper procedures by a qualified operator and only for the purposes described in this manual.

The following conventions are used to indicate and classify precautions in this manual. Always heed the information provided with them. Failure to heed precautions can result in injury to people or damage to property.

 **DANGER** Indicates an imminently hazardous situation which, if not avoided, will result in death or serious injury.

 **WARNING** Indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury.

 **Caution** Indicates a potentially hazardous situation which, if not avoided, may result in minor or moderate injury, or property damage.

## OMRON Product References

All OMRON products are capitalized in this manual. The word “Unit” is also capitalized when it refers to an OMRON product, regardless of whether or not it appears in the proper name of the product.

The abbreviation “Ch,” which appears in some displays and on some OMRON products, often means “word” and is abbreviated “Wd” in documentation in this sense.

The abbreviation “PC” means Programmable Controller and is not used as an abbreviation for anything else.

## Visual Aids

The following headings appear in the left column of the manual to help you locate different types of information.

**Note** Indicates information of particular interest for efficient and convenient operation of the product.

**1, 2, 3...** 1. Indicates lists of one sort or another, such as procedures, checklists, etc.

## © OMRON, 1993

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.

## About this Manual:

This manual describes programming of the CQM1, CPM1, CPM1A, and SRM1 Programmable Controllers, including memory structure, memory contents, ladder-diagram instructions, etc., and includes the sections described below. Refer to the *CQM1 Operation Manual*, *CPM1 Operation Manual*, *CPM1A Operation Manual*, and *SRM1 Master Control Units Operation Manual* for hardware information and Programming Console operating procedures. Refer to the *SSS Operation Manual: C-series PCs* for SSS operating procedures.

**Note** The SRM1 is a specialized programmable controller and is normally called a CompoBus/S Master Control Unit. The SRM1, however, is programmed in the same way as the other Programmable Controllers and it is treated and referred to as a PC in this manual.

Please read this manual carefully and be sure you understand the information provided before attempting to program and operate the CQM1, CPM1, CPM1A or SRM1.

**Section 1** explains the PC Setup and related PC functions, including interrupt processing and communications. The PC Setup can be used to control the operating parameters of the PC.

**Section 2** provides an introduction to new PC features, including the new instructions available through expansion instructions and a new monitoring feature call differential monitoring.

**Section 3** describes the structure of the PC's memory areas, and explains how to use them. It also describes Memory Cassette operations used to transfer data between the CQM1 and a Memory Cassette.

**Section 4** explains the basic steps and concepts involved in writing a basic ladder diagram program. It introduces the instructions that are used to build the basic structure of the ladder diagram and control its execution.

**Section 5** individually describes the ladder-diagram programming instructions that can be used with the PC.

**Section 6** explains the methods and procedures for using host link commands, which can be used for host link communications via the PC ports.

**Section 7** explains the internal processing of the PCs, and the time required for processing and execution. Refer to this section to gain an understanding of the precise timing of PC operation.

**Section 8** describes how to diagnose and correct the hardware and software errors that can occur during PC operation.

The following appendices are also provided: **A Programming Instructions**, **B Error and Arithmetic Flag Operation**, **C Memory Areas**, **D Using the Clock Function**, **E I/O Assignment Sheet**, **F Program Coding Sheet**, **G List of FAL Numbers**, **H Extended ASCII**, and **I CPM1A and CPM1 Memory Area Comparison**.



**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.

# TABLE OF CONTENTS

<b>PRECAUTIONS</b> .....	<b>xiii</b>
1 Intended Audience .....	xiv
2 General Precautions .....	xiv
3 Safety Precautions .....	xiv
4 Application Precautions .....	xiv
<b>SECTION 1</b>	
<b>PC Setup and Other Features</b> .....	<b>1</b>
1-1 PC Setup .....	3
1-2 Basic PC Operation and I/O Processes .....	16
1-3 Pulse Output Function (CQM1 Only) .....	22
1-4 Pulse Output Function (CPM1A Only) .....	35
1-5 CQM1 Interrupt Functions .....	38
1-6 CPM1/CPM1A Interrupt Functions .....	67
1-7 SRM1 Interrupt Functions .....	83
1-8 CompoBus/S Distributed I/O Functions (SRM1 Only) .....	86
1-9 Communications Functions .....	87
1-10 Calculating with Signed Binary Data .....	111
<b>SECTION 2</b>	
<b>Special Features</b> .....	<b>115</b>
2-1 Expansion Instructions (CQM1/SRM1 Only) .....	116
2-2 Advanced I/O Instructions (CQM1 Only) .....	118
2-3 Macro Function .....	128
2-4 Differential Monitor .....	129
2-5 Analog Settings (CQM1-CPU42-EV1/CPM1/CPM1A Only) .....	129
2-6 Quick-response Inputs (CPM1/CPM1A Only) .....	131
<b>SECTION 3</b>	
<b>Memory Areas</b> .....	<b>133</b>
3-1 CQM1 Memory Area Functions .....	134
3-2 CPM1/CPM1A Memory Area Functions .....	139
3-3 SRM1 Memory Area Functions .....	143
3-4 SRM1 Flash Memory .....	145
3-5 Using Memory Cassettes (CQM1 Only) .....	146
<b>SECTION 4</b>	
<b>Ladder-diagram Programming</b> .....	<b>151</b>
4-1 Basic Procedure .....	152
4-2 Instruction Terminology .....	152
4-3 Basic Ladder Diagrams .....	153
4-4 Controlling Bit Status .....	173
4-5 Work Bits (Internal Relays) .....	175
4-6 Programming Precautions .....	177
4-7 Program Execution .....	179

# TABLE OF CONTENTS

<b>SECTION 5</b>	
<b>Instruction Set</b> .....	<b>181</b>
5-1 Notation .....	184
5-2 Instruction Format .....	184
5-3 Data Areas, Definer Values, and Flags .....	184
5-4 Differentiated Instructions .....	186
5-5 Coding Right-hand Instructions .....	186
5-6 Instruction Tables .....	190
5-7 Ladder Diagram Instructions .....	196
5-8 Bit Control Instructions .....	197
5-9 NO OPERATION – NOP(00) .....	201
5-10 END – END(01) .....	201
5-11 INTERLOCK and INTERLOCK CLEAR – IL(02) and ILC(03) .....	201
5-12 JUMP and JUMP END – JMP(04) and JME(05) .....	203
5-13 User Error Instructions: FAILURE ALARM AND RESET – FAL(06) and SEVERE FAILURE ALARM – FALS(07) .....	205
5-14 Step Instructions: STEP DEFINE and STEP START–STEP(08)/SNXT(09) .....	206
5-15 Timer and Counter Instructions .....	208
5-16 Shift Instructions .....	224
5-17 Data Movement Instructions .....	232
5-18 Comparison Instructions .....	242
5-19 Conversion Instructions .....	252
5-20 BCD Calculation Instructions .....	278
5-21 Binary Calculation Instructions .....	289
5-22 Special Math Instructions .....	300
5-23 Logic Instructions .....	308
5-24 Increment/Decrement Instructions .....	311
5-25 Subroutine Instructions .....	313
5-26 Special Instructions .....	315
5-27 Communications Instructions .....	340
5-28 Advanced I/O Instructions .....	345
<b>SECTION 6</b>	
<b>Host Link Commands</b> .....	<b>349</b>
6-1 Communications Procedure .....	350
6-2 Command and Response Formats .....	352
6-3 Host Link Commands .....	356
<b>SECTION 7</b>	
<b>PC Operations and Processing Time</b> .....	<b>381</b>
7-1 CQM1 Cycle Time and I/O Response Time .....	382
7-2 CPM1/CPM1A Cycle Time and I/O Response Time .....	400
7-3 SRM1 Cycle Time and I/O Response Time .....	411
<b>SECTION 8</b>	
<b>Troubleshooting</b> .....	<b>423</b>
8-1 Introduction .....	424
8-2 Programming Console Operation Errors .....	424
8-3 Programming Errors .....	425
8-4 User-defined Errors .....	426
8-5 Operating Errors .....	427
8-6 Error Log .....	430
8-7 Host Link Errors .....	432
8-8 Troubleshooting Flowcharts .....	434

# TABLE OF CONTENTS

## Appendices

A Programming Instructions .....	441
B Error and Arithmetic Flag Operation .....	447
C Memory Areas .....	451
D Using the Clock Function .....	469
E I/O Assignment Sheet .....	471
F Program Coding Sheet .....	473
G List of FAL Numbers .....	477
H Extended ASCII .....	479
I CPM1/CPM1A and CQM1 Memory Area Comparison .....	481
<b>Glossary .....</b>	<b>483</b>
<b>Index .....</b>	<b>499</b>
<b>Revision History .....</b>	<b>507</b>

# PRECAUTIONS

This section provides general precautions for using the Programmable Controller (PC) and related devices.

**The information contained in this section is important for the safe and reliable application of the Programmable Controller. You must read this section and understand the information contained before attempting to set up or operate a PC system.**

1 Intended Audience .....	xiv
2 General Precautions .....	xiv
3 Safety Precautions .....	xiv
4 Application Precautions .....	xiv



## 1 Intended Audience

This manual is intended for the following personnel, who must also have knowledge of electrical systems (an electrical engineer or the equivalent).

- Personnel in charge of installing FA systems.
- Personnel in charge of designing FA systems.
- Personnel in charge of managing FA systems and facilities.


## 2 General Precautions

The user must operate the product according to the performance specifications described in the operation manuals.


Before using the product under conditions which are not described in the manual or applying the product to nuclear control systems, railroad systems, aviation systems, vehicles, combustion systems, medical equipment, amusement machines, safety equipment, and other systems, machines, and equipment that may have a serious influence on lives and property if used improperly, consult your OMRON representative.


Make sure that the ratings and performance characteristics of the product are sufficient for the systems, machines, and equipment, and be sure to provide the systems, machines, and equipment with double safety mechanisms.

This manual provides information for programming and operating the Unit. Be sure to read this manual before attempting to use the Unit and keep this manual close at hand for reference during operation.

 **WARNING** It is extremely important that a PC and all PC Units be used for the specified purpose and under the specified conditions, especially in applications that can directly or indirectly affect human life. You must consult with your OMRON representative before applying a PC System to the above-mentioned applications.


## 3 Safety Precautions

 **Caution** Execute online edit only after confirming that no adverse effects will be caused by extending the cycle time. Otherwise, the input signals may not be readable.

 **Caution** Confirm safety at the destination node before transferring a program to another node or changing the I/O memory area. Doing either of these without confirming safety may result in injury.


## 4 Application Precautions


Observe the following precautions when using the PC System.


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

- Fail-safe measures must be taken by the customer to ensure safety in the event of incorrect, missing, or abnormal signals caused by broken signal lines, momentary power interruptions, or other causes.
- Interlock circuits, limit circuits, and similar safety measures in external circuits (i.e., not in the Programmable Controller) must be provided by the customer.

- Check the user program for proper execution before actually running it on the Unit. Not checking the program may result in an unexpected operation.
- Confirm that no adverse effect will occur in the system before attempting any of the following. Not doing so may result in an unexpected operation.
  - Changing the operating mode of the PC.
  - Force-setting/force-resetting any bit in memory.
  - Changing the present value of any word or any set value in memory.
- Resume operation only after transferring to the new CPU Unit the contents of the DM and HR Areas required for resuming operation. Not doing so may result in an unexpected operation.
- Do not place objects on top of the cables. Doing so may break the cables.
- Before touching the Unit, be sure to first touch a grounded metallic object in order to discharge any static built-up. Not doing so may result in malfunction or damage.
- Do not touch the Expansion I/O Unit Connecting Cable while the power is being supplied in order to prevent any malfunction due to static electricity.

 **Caution** Always clear memory before beginning to program the CPM1, CPM1A or SRM1. Although memory is cleared before the CPU Unit is shipped (except for bits with specific functions), AR 1314, which turns ON when the internal capacitor cannot back up memory, may have turned ON during shipment.

 **Caution** If the CPM1 or CPM1A will be turned OFF for periods exceeding the data backup period of the internal capacitor, design the system so that it will not be influenced if data in the DM, HR, and CNT areas is cleared when power is turned OFF.

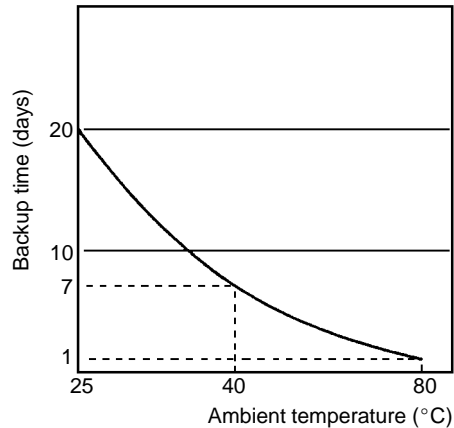
 **Caution** Either switch the CPM1 or CPM1A to RUN or MONITOR mode, or turn OFF and ON power to the CPM1 or CPM1A after changing from a Programming Device any data that is backed up in flash memory. This data includes the user program, read-only DM area (DM 6144 to DM 6599), and the PC Setup (DM 6600 to DM 6655).

- The user program and memory area data in the CPM1 or CPM1A are backed up either by an internal capacitor or in flash memory as shown in the following table.

Backup method	Data
Internal capacitor	Read/write DM area (DM 0000 to DM 0999, DM 1022, and DM 1023) Error log area (DM 1000 to DM 1021) HR area (HR 00 to HR 19) Counter area (CNT 000 to CNT 127)
Flash memory	User program Read-only DM area (DM 6144 to DM 6599) PC Setup (DM 6600 to DM 6655)

- Note**
1. The IR, TR, LR, and timer areas are not normally backed up when power is turned OFF and all contents will be cleared the next time power is turned ON. (The PC Setup setting in DM 6601 can be used to back up this data. Refer to details on the PC Setup later in this manual for details.)
  2. The bits in the AR and SR areas have special functions and are set according to these functions when power is turned ON.

- The capacitor backup time depends on the ambient temperature, as shown in the following graph. The backup time, however, assumes that the capacitor is fully charged, which requires that power be supplied to the CPU Unit continuously for at least 15 minutes.



If the power remains OFF for a period exceeding the data backup period, AR 1314 will turn ON to indicate that the capacitor can no longer back up data and the data backed up by the capacitor will be cleared. AR 1314 will remain ON unless it is turned OFF using I/O monitor operations, using memory clear operations, or from the user program.

If desired, the PC Setup setting in DM 6604 can be set to create a fatal error and thus stop the system when AR 1314 goes ON.

- The data stored in flash memory will not be lost even if power remains OFF for a period exceeding the data backup period, because the data stored in flash memory will be read to the CPU Unit when the CPM1 or CPM1A is turned ON.
- If the power is turned OFF without changing the mode from PROGRAM mode to RUN or MONITOR mode after having made changes in the data that is backed up in flash memory, the changes will not be written to flash memory. If the power is then left OFF for more than 20 days (at 25°C), the changes (i.e., the contents of the RAM) will be erased and the data values will become undefined.



### Caution

Be sure that the SRM1 system is not influenced by any undefined data if the data in the DM, HR, or CNT area is cleared when the SRM1 has been turned OFF for a period exceeding the data backup period of the internal lithium battery. If the AR 1414 flag is ON, the data will be held unless it is turned OFF using the I/O Monitor operation, instructions, etc. The system can be stopped by designating DM 6604 in the PC Setup so that a memory error occurs when the power interruption hold area is not held (with AR 1314 ON)

- A lithium battery in the CPU Unit is used to back up the counter values and the contents of the DM area, and HR area. The deterioration of the lithium battery capacity depends on the ambient temperature. The standard service life is 12 years at an ambient temperature of 40°C when operating 8 hours a day.

If the power remains off for a period exceeding the data backup period, the contents of the Data Memory (DM), Hold Relay (HR), and Counter (CNT) Areas in the CPU Unit may be cleared and the AR 1314 flag (which turns ON when the power interruption hold area is not held) may turn ON.

If the contents of the CPU Unit's program area are lost, the program stored in flash memory will be read to the CPU Unit's program area when the SRM1 is started up because the contents in the read-only area (DM 6144 through DM 6599) and PC Setup (DM 6600 through DM 6655) will be written to flash memory.

- However, if the power is turned OFF without changing the mode even if changes are made in the read-only DM area (DM 6144 through DM 6599), or PC Setup (DM 6600 through DM 6655) using a peripheral device, the contents of changes will not be written to flash memory. Although the data in these areas is backed up by the lithium battery, contents of changes will disappear if the service life of the lithium battery expires. In this case, programs in the flash memory will be automatically read into the user program memory.

The changes can be saved by switching the SRM1 to RUN or MONITOR mode or turning OFF and restarting the SRM1 soon after the changes are made.

# TABLE OF CONTENTS

<b>PRECAUTIONS</b> .....	<b>xiii</b>
1 Intended Audience .....	xiv
2 General Precautions .....	xiv
3 Safety Precautions .....	xiv
4 Application Precautions .....	xiv
<b>SECTION 1</b>	
<b>PC Setup and Other Features</b> .....	<b>1</b>
1-1 PC Setup .....	3
1-2 Basic PC Operation and I/O Processes .....	16
1-3 Pulse Output Function (CQM1 Only) .....	22
1-4 Pulse Output Function (CPM1A Only) .....	35
1-5 CQM1 Interrupt Functions .....	38
1-6 CPM1/CPM1A Interrupt Functions .....	67
1-7 SRM1 Interrupt Functions .....	83
1-8 CompoBus/S Distributed I/O Functions (SRM1 Only) .....	86
1-9 Communications Functions .....	87
1-10 Calculating with Signed Binary Data .....	111
<b>SECTION 2</b>	
<b>Special Features</b> .....	<b>115</b>
2-1 Expansion Instructions (CQM1/SRM1 Only) .....	116
2-2 Advanced I/O Instructions (CQM1 Only) .....	118
2-3 Macro Function .....	128
2-4 Differential Monitor .....	129
2-5 Analog Settings (CQM1-CPU42-EV1/CPM1/CPM1A Only) .....	129
2-6 Quick-response Inputs (CPM1/CPM1A Only) .....	131
<b>SECTION 3</b>	
<b>Memory Areas</b> .....	<b>133</b>
3-1 CQM1 Memory Area Functions .....	134
3-2 CPM1/CPM1A Memory Area Functions .....	139
3-3 SRM1 Memory Area Functions .....	143
3-4 SRM1 Flash Memory .....	145
3-5 Using Memory Cassettes (CQM1 Only) .....	146
<b>SECTION 4</b>	
<b>Ladder-diagram Programming</b> .....	<b>151</b>
4-1 Basic Procedure .....	152
4-2 Instruction Terminology .....	152
4-3 Basic Ladder Diagrams .....	153
4-4 Controlling Bit Status .....	173
4-5 Work Bits (Internal Relays) .....	175
4-6 Programming Precautions .....	177
4-7 Program Execution .....	179

# TABLE OF CONTENTS

<b>SECTION 5</b>	
<b>Instruction Set</b> .....	<b>181</b>
5-1 Notation .....	184
5-2 Instruction Format .....	184
5-3 Data Areas, Definer Values, and Flags .....	184
5-4 Differentiated Instructions .....	186
5-5 Coding Right-hand Instructions .....	186
5-6 Instruction Tables .....	190
5-7 Ladder Diagram Instructions .....	196
5-8 Bit Control Instructions .....	197
5-9 NO OPERATION – NOP(00) .....	201
5-10 END – END(01) .....	201
5-11 INTERLOCK and INTERLOCK CLEAR – IL(02) and ILC(03) .....	201
5-12 JUMP and JUMP END – JMP(04) and JME(05) .....	203
5-13 User Error Instructions: FAILURE ALARM AND RESET – FAL(06) and SEVERE FAILURE ALARM – FALS(07) .....	205
5-14 Step Instructions: STEP DEFINE and STEP START–STEP(08)/SNXT(09) .....	206
5-15 Timer and Counter Instructions .....	208
5-16 Shift Instructions .....	224
5-17 Data Movement Instructions .....	232
5-18 Comparison Instructions .....	242
5-19 Conversion Instructions .....	252
5-20 BCD Calculation Instructions .....	278
5-21 Binary Calculation Instructions .....	289
5-22 Special Math Instructions .....	300
5-23 Logic Instructions .....	308
5-24 Increment/Decrement Instructions .....	311
5-25 Subroutine Instructions .....	313
5-26 Special Instructions .....	315
5-27 Communications Instructions .....	340
5-28 Advanced I/O Instructions .....	345
<b>SECTION 6</b>	
<b>Host Link Commands</b> .....	<b>349</b>
6-1 Communications Procedure .....	350
6-2 Command and Response Formats .....	352
6-3 Host Link Commands .....	356
<b>SECTION 7</b>	
<b>PC Operations and Processing Time</b> .....	<b>381</b>
7-1 CQM1 Cycle Time and I/O Response Time .....	382
7-2 CPM1/CPM1A Cycle Time and I/O Response Time .....	400
7-3 SRM1 Cycle Time and I/O Response Time .....	411
<b>SECTION 8</b>	
<b>Troubleshooting</b> .....	<b>423</b>
8-1 Introduction .....	424
8-2 Programming Console Operation Errors .....	424
8-3 Programming Errors .....	425
8-4 User-defined Errors .....	426
8-5 Operating Errors .....	427
8-6 Error Log .....	430
8-7 Host Link Errors .....	432
8-8 Troubleshooting Flowcharts .....	434

# TABLE OF CONTENTS

## Appendices

A Programming Instructions .....	441
B Error and Arithmetic Flag Operation .....	447
C Memory Areas .....	451
D Using the Clock Function .....	469
E I/O Assignment Sheet .....	471
F Program Coding Sheet .....	473
G List of FAL Numbers .....	477
H Extended ASCII .....	479
I CPM1/CPM1A and CQM1 Memory Area Comparison .....	481
<b>Glossary .....</b>	<b>483</b>
<b>Index .....</b>	<b>499</b>
<b>Revision History .....</b>	<b>507</b>

# PRECAUTIONS

This section provides general precautions for using the Programmable Controller (PC) and related devices.

**The information contained in this section is important for the safe and reliable application of the Programmable Controller. You must read this section and understand the information contained before attempting to set up or operate a PC system.**

1 Intended Audience .....	xiv
2 General Precautions .....	xiv
3 Safety Precautions .....	xiv
4 Application Precautions .....	xiv



## 1 Intended Audience

This manual is intended for the following personnel, who must also have knowledge of electrical systems (an electrical engineer or the equivalent).

- Personnel in charge of installing FA systems.
- Personnel in charge of designing FA systems.
- Personnel in charge of managing FA systems and facilities.


## 2 General Precautions

The user must operate the product according to the performance specifications described in the operation manuals.


Before using the product under conditions which are not described in the manual or applying the product to nuclear control systems, railroad systems, aviation systems, vehicles, combustion systems, medical equipment, amusement machines, safety equipment, and other systems, machines, and equipment that may have a serious influence on lives and property if used improperly, consult your OMRON representative.


Make sure that the ratings and performance characteristics of the product are sufficient for the systems, machines, and equipment, and be sure to provide the systems, machines, and equipment with double safety mechanisms.

This manual provides information for programming and operating the Unit. Be sure to read this manual before attempting to use the Unit and keep this manual close at hand for reference during operation.

 **WARNING** It is extremely important that a PC and all PC Units be used for the specified purpose and under the specified conditions, especially in applications that can directly or indirectly affect human life. You must consult with your OMRON representative before applying a PC System to the above-mentioned applications.


## 3 Safety Precautions

 **Caution** Execute online edit only after confirming that no adverse effects will be caused by extending the cycle time. Otherwise, the input signals may not be readable.

 **Caution** Confirm safety at the destination node before transferring a program to another node or changing the I/O memory area. Doing either of these without confirming safety may result in injury.


## 4 Application Precautions


Observe the following precautions when using the PC System.


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

- Fail-safe measures must be taken by the customer to ensure safety in the event of incorrect, missing, or abnormal signals caused by broken signal lines, momentary power interruptions, or other causes.
- Interlock circuits, limit circuits, and similar safety measures in external circuits (i.e., not in the Programmable Controller) must be provided by the customer.

- Check the user program for proper execution before actually running it on the Unit. Not checking the program may result in an unexpected operation.
- Confirm that no adverse effect will occur in the system before attempting any of the following. Not doing so may result in an unexpected operation.
  - Changing the operating mode of the PC.
  - Force-setting/force-resetting any bit in memory.
  - Changing the present value of any word or any set value in memory.
- Resume operation only after transferring to the new CPU Unit the contents of the DM and HR Areas required for resuming operation. Not doing so may result in an unexpected operation.
- Do not place objects on top of the cables. Doing so may break the cables.
- Before touching the Unit, be sure to first touch a grounded metallic object in order to discharge any static built-up. Not doing so may result in malfunction or damage.
- Do not touch the Expansion I/O Unit Connecting Cable while the power is being supplied in order to prevent any malfunction due to static electricity.

 **Caution** Always clear memory before beginning to program the CPM1, CPM1A or SRM1. Although memory is cleared before the CPU Unit is shipped (except for bits with specific functions), AR 1314, which turns ON when the internal capacitor cannot back up memory, may have turned ON during shipment.

 **Caution** If the CPM1 or CPM1A will be turned OFF for periods exceeding the data backup period of the internal capacitor, design the system so that it will not be influenced if data in the DM, HR, and CNT areas is cleared when power is turned OFF.

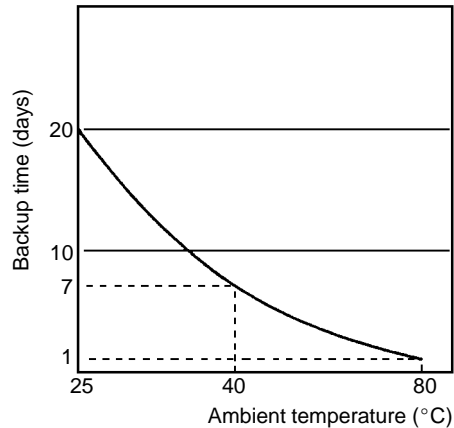
 **Caution** Either switch the CPM1 or CPM1A to RUN or MONITOR mode, or turn OFF and ON power to the CPM1 or CPM1A after changing from a Programming Device any data that is backed up in flash memory. This data includes the user program, read-only DM area (DM 6144 to DM 6599), and the PC Setup (DM 6600 to DM 6655).

- The user program and memory area data in the CPM1 or CPM1A are backed up either by an internal capacitor or in flash memory as shown in the following table.

Backup method	Data
Internal capacitor	Read/write DM area (DM 0000 to DM 0999, DM 1022, and DM 1023) Error log area (DM 1000 to DM 1021) HR area (HR 00 to HR 19) Counter area (CNT 000 to CNT 127)
Flash memory	User program Read-only DM area (DM 6144 to DM 6599) PC Setup (DM 6600 to DM 6655)

- Note**
1. The IR, TR, LR, and timer areas are not normally backed up when power is turned OFF and all contents will be cleared the next time power is turned ON. (The PC Setup setting in DM 6601 can be used to back up this data. Refer to details on the PC Setup later in this manual for details.)
  2. The bits in the AR and SR areas have special functions and are set according to these functions when power is turned ON.

- The capacitor backup time depends on the ambient temperature, as shown in the following graph. The backup time, however, assumes that the capacitor is fully charged, which requires that power be supplied to the CPU Unit continuously for at least 15 minutes.



If the power remains OFF for a period exceeding the data backup period, AR 1314 will turn ON to indicate that the capacitor can no longer back up data and the data backed up by the capacitor will be cleared. AR 1314 will remain ON unless it is turned OFF using I/O monitor operations, using memory clear operations, or from the user program.

If desired, the PC Setup setting in DM 6604 can be set to create a fatal error and thus stop the system when AR 1314 goes ON.

- The data stored in flash memory will not be lost even if power remains OFF for a period exceeding the data backup period, because the data stored in flash memory will be read to the CPU Unit when the CPM1 or CPM1A is turned ON.
- If the power is turned OFF without changing the mode from PROGRAM mode to RUN or MONITOR mode after having made changes in the data that is backed up in flash memory, the changes will not be written to flash memory. If the power is then left OFF for more than 20 days (at 25°C), the changes (i.e., the contents of the RAM) will be erased and the data values will become undefined.



### Caution

Be sure that the SRM1 system is not influenced by any undefined data if the data in the DM, HR, or CNT area is cleared when the SRM1 has been turned OFF for a period exceeding the data backup period of the internal lithium battery. If the AR 1414 flag is ON, the data will be held unless it is turned OFF using the I/O Monitor operation, instructions, etc. The system can be stopped by designating DM 6604 in the PC Setup so that a memory error occurs when the power interruption hold area is not held (with AR 1314 ON)

- A lithium battery in the CPU Unit is used to back up the counter values and the contents of the DM area, and HR area. The deterioration of the lithium battery capacity depends on the ambient temperature. The standard service life is 12 years at an ambient temperature of 40°C when operating 8 hours a day.

If the power remains off for a period exceeding the data backup period, the contents of the Data Memory (DM), Hold Relay (HR), and Counter (CNT) Areas in the CPU Unit may be cleared and the AR 1314 flag (which turns ON when the power interruption hold area is not held) may turn ON.

If the contents of the CPU Unit's program area are lost, the program stored in flash memory will be read to the CPU Unit's program area when the SRM1 is started up because the contents in the read-only area (DM 6144 through DM 6599) and PC Setup (DM 6600 through DM 6655) will be written to flash memory.

- However, if the power is turned OFF without changing the mode even if changes are made in the read-only DM area (DM 6144 through DM 6599), or PC Setup (DM 6600 through DM 6655) using a peripheral device, the contents of changes will not be written to flash memory. Although the data in these areas is backed up by the lithium battery, contents of changes will disappear if the service life of the lithium battery expires. In this case, programs in the flash memory will be automatically read into the user program memory.

The changes can be saved by switching the SRM1 to RUN or MONITOR mode or turning OFF and restarting the SRM1 soon after the changes are made.

# SECTION 1

## PC Setup and Other Features

This section explains the PC Setup and other CQM1/CPM1/CPM1A/SRM1 features, including interrupt processing and communications. The PC Setup can be used to control the operating parameters of the CQM1/CPM1/CPM1A/SRM1. To change the PC Setup, refer to the *CQM1 Operation Manual*, *CPM1 Operation Manual*, *CPM1A Operation Manual* or *SRM1 Master Control Units Operation Manual* for Programming Console procedures. Refer to the *SSS Operation Manual: C-series PCs* for SSS procedures.

If you are not familiar with OMRON PCs or ladder diagram program, you can read *1-5 PC Setup* as an overview of the operating parameters available for the CQM1/CPM1/CPM1A/SRM1, but may then want to read *Section 3 Memory Areas*, *Section 4 Ladder-diagram Programming*, and related instructions in *Section 5 Instruction Set* before completing this section.

1-1	PC Setup .....	3
1-1-1	Changing the PC Setup .....	3
1-1-2	CQM1 PC Setup Settings .....	4
1-1-3	CPM1/CPM1A PC Setup Settings .....	9
1-1-4	SRM1 PC Setup Settings .....	13
1-2	Basic PC Operation and I/O Processes .....	16
1-2-1	Startup Mode .....	16
1-2-2	Hold Bit Status .....	17
1-2-3	Program Memory Write-protection (CPM1/CPM1A Only) .....	17
1-2-4	RS-232C Port Servicing Time (CQM1/SRM1 Only) .....	18
1-2-5	Peripheral Port Servicing Time .....	18
1-2-6	Cycle Time .....	18
1-2-7	Input Time Constants .....	19
1-2-8	High-speed Timers (CQM1 Only) .....	20
1-2-9	DSW(87) Input Digits & Output Refresh Method (CQM1 Only) .....	21
1-2-10	Error Log Settings .....	21
1-3	Pulse Output Function (CQM1 Only) .....	22
1-3-1	Types of Pulse Outputs .....	22
1-3-2	Standard Pulse Output from an Output Point .....	23
1-3-3	Standard Pulse Output from Ports 1 and 2 .....	25
1-3-4	Variable-duty-ratio Pulse Output from Ports 1 and 2 .....	32
1-3-5	Determining the Status of Ports 1 and 2 .....	34
1-4	Pulse Output Function (CPM1A Only) .....	35
1-4-1	Programming Example in Continuous Mode .....	36
1-4-2	Programming Example in Independent Mode .....	36
1-4-3	Using Pulse Output Instructions .....	36
1-4-4	Changing the Frequency .....	37
1-4-5	Stopping Pulse Output .....	37
1-5	CQM1 Interrupt Functions .....	38
1-5-1	Types of Interrupts .....	38
1-5-2	Input Interrupts .....	39
1-5-3	Masking All Interrupts .....	44
1-5-4	Interval Timer Interrupts .....	44
1-5-5	High-speed Counter 0 Interrupts .....	47
1-5-6	High-speed Counter 0 Overflows/Underflows .....	53
1-5-7	High-speed Counter 1 and 2 Interrupts (CQM1-CPU43-EV1) .....	55
1-5-8	Absolute High-speed Counter Interrupts (CQM1-CPU44-EV1) .....	62
1-6	CPM1/CPM1A Interrupt Functions .....	67
1-6-1	Types of Interrupts .....	67
1-6-2	Input Interrupts .....	69
1-6-3	Masking All Interrupts .....	73
1-6-4	Interval Timer Interrupts .....	74
1-6-5	High-speed Counter Interrupts .....	76

1-7	SRM1 Interrupt Functions .....	83
1-7-1	Types of Interrupts .....	83
1-7-2	Interval Timer Interrupts .....	83
1-8	CompoBus/S Distributed I/O Functions (SRM1 Only) .....	86
1-9	Communications Functions .....	87
1-9-1	CQM1 PC Setup .....	88
1-9-2	Wiring Ports .....	91
1-9-3	CQM1 Host Link Communications .....	91
1-9-4	CPM1/CPM1A Host Link Communications .....	93
1-9-5	SRM1 Host Link Communications .....	95
1-9-6	RS-232C Communications (CQM1/SRM1 Only) .....	98
1-9-7	CQM1 One-to-one Link Communications .....	100
1-9-8	CPM1/CPM1A One-to-one Link Communications .....	101
1-9-9	CPM1/CPM1A NT Link Communications .....	103
1-9-10	SRM1 One-to-one Link Communications .....	104
1-9-11	SRM1 NT Link Communications .....	106
1-9-12	SRM1 No Protocol Communications .....	107
1-9-13	Transmission Data Configuration .....	110
1-9-14	Transmission Flags .....	110
1-9-15	No Protocol Communications Program Example .....	111
1-10	Calculating with Signed Binary Data .....	111
1-10-1	Definition of Signed Binary Data .....	112
1-10-2	Arithmetic Flags .....	113
1-10-3	Inputting Signed Binary Data Using Decimal Values .....	113
1-10-4	Using Signed-binary Expansion Instructions (CQM1 Only) .....	113
1-10-5	Application Example Using Signed Binary Data .....	114

## 1-1 PC Setup

The PC Setup comprises various operating parameters that control CQM1/CPM1/CPM1A/SRM1 operation. In order to make the maximum use of CQM1/CPM1/CPM1A/SRM1 functionality when using interrupt processing and communications functions, the PC Setup may be customized according to operating conditions.

At the time of shipping, the defaults are set for general operating conditions, so that the CQM1/CPM1/CPM1A/SRM1 can be used without having to change the settings. You are, however, advised to check the default values before operation.

### Default Values

The default values for the PC Setup are 0000 for all words. The default values can be reset at any time by turning ON SR 25210.



### Caution

When data memory (DM) is cleared from a Programming Device, the PC Setup settings will also be cleared to all zeros.

### 1-1-1 Changing the PC Setup

PC Setup settings are accessed at various times depending on the setting, as described below.

- DM 6600 to DM 6614: Accessed only when PC's power supply is turned on.
- DM 6615 to DM 6644: Accessed only when program execution begins.
- DM 6645 to DM 6655: Accessed regularly when the power is on.

Since changes in the PC Setup become effective only at the times given above, the CQM1/CPM1/CPM1A/SRM1 will have to be restarted to make changes in DM 6600 to DM 6614 effective, and program execution will have to be restarted to make changes in DM 6615 to DM 6644 effective.

When DM 6602 bits 00 to 03 are set to protect the program memory, DM 6602 cannot be changed using the PC Setup operation of the Support Software. To change DM 6602, use the I/O Monitor or DM Edit operation.

### Making Changes from a Peripheral Device

The PC Setup can be read, but not written into, from the user program. Writing can be done only by using a Programming Device.

Although the PC Setup is stored in DM 6600 to DM 6655, settings can be made and changed only from a Programming Device (e.g., SSS, or Programming Console). DM 6600 to DM 6644 can be set or changed only while in PROGRAM mode. DM 6645 to DM 6655 can be set or changed while in either PROGRAM mode or MONITOR mode.

The following settings can be made in PROGRAM mode from the SSS using menu operations. All other settings must be made using the hexadecimal setting operation.

- Startup Mode (DM 6600)
- I/O Hold Bit Status and Forced Status Hold Bit Status (DM 6601)
- Cycle Monitor Time (DM 6618)
- Cycle Time (DM 6619)
- RS-232C Port Settings (DM 6645 to DM 6649)

**Note** The RS-232C Port Settings (DM 6645 to DM 6649) are not used in CPM1/CPM1A PCs because these PCs aren't equipped with an RS-232C port.

### Errors in the PC Setup

If an incorrect PC Setup setting is accessed, a non-fatal error (error code 9B) will be generated, the corresponding error flag (AR 2400 to AR 2402 in the CQM1, AR 1300 to AR 1302 in the CPM1/CPM1A/SRM1) will be turned ON, and the default setting will be used instead of the incorrect setting.

## 1-1-2 CQM1 PC Setup Settings

The PC Setup is broadly divided into four categories: 1) Settings related to basic CQM1 operation and I/O processes, 2) Settings related to pulse output functions, 3) Settings related to interrupts, and 4) Settings related to communications. This section will explain the settings according to these classifications.

The following table shows the setting in order in the DM area. For details, refer to the page numbers shown.

Word(s)	Bit(s)	Function	Page
<b>Startup Processing (DM 6600 to DM 6614)</b>			
The following settings are effective after transfer to the PC only after the PC is restarted.			
DM 6600	00 to 07	Startup mode (effective when bits 08 to 15 are set to 02). 00: PROGRAM; 01: MONITOR 02: RUN	16
	08 to 15	Startup mode designation 00: Programming Console switch 01: Continue operating mode last used before power was turned off 02: Setting in 00 to 07	
DM 6601	00 to 07	Not used.	17
	08 to 11	IOM Hold Bit (SR 25212) Status 0: Reset; 1: Maintain	
	12 to 15	Forced Status Hold Bit (SR 25211) Status 0: Reset; 1: Maintain	
DM 6602 to DM 6610	00 to 15	Not used.	
DM 6611	00 to 15	CQM1-CPU43-EV1: Mode setting for ports 1 and 2 0000: High-speed counter mode; 0001: Pulse output mode CQM1-CPU44-EV1: Origin compensation setting for port 1 (4-digit BCD)	25, 63
DM 6612	00 to 15	CQM1-CPU44-EV1: Origin compensation setting for port 2 (4-digit BCD)	63
<b>Pulse Output and Cycle Time Settings (DM 6615 to DM 6619)</b>			
The following settings are effective after transfer to the PC the next time operation is started.			
DM 6615	00 to 07	Word for pulse output. 00: IR 100; 01: IR101; 02: IR 102... 15: IR 115	23
	08 to 15	Not used.	
DM 6616	00 to 07	Servicing time for RS-232C port (effective when bits 08 to 15 are set to 01) 00 to 99 (BCD): Percentage of cycle time used to service RS-232C port.	18
	08 to 15	RS-232C port servicing setting enable 00: 5% of the cycle time 01: Use time in 00 to 07.	
DM 6617	00 to 07	Servicing time for peripheral port (effective when bits 08 to 15 are set to 01) 00 to 99 (BCD): Percentage of cycle time used to service peripheral.	18
	08 to 15	Peripheral port servicing setting enable 00: 5% of the cycle time 01: Use time in 00 to 07.	
DM 6618	00 to 07	Cycle monitor time (effective when bits 08 to 15 are set to 01, 02, or 03) 00 to 99 (BCD): Setting (see 08 to 15)	21
	08 to 15	Cycle monitor enable (Setting in 00 to 07 x unit; 99 s max.) 00: 120 ms (setting in bits 00 to 07 disabled) 01: Setting unit: 10 ms 02: Setting unit: 100 ms 03: Setting unit: 1 s	
DM 6619	00 to 15	Cycle time 0000: Variable (no minimum) 0001 to 9999 (BCD): Minimum time in ms	18



Word(s)	Bit(s)	Function	Page
<b>Interrupt Processing (DM 6620 to DM 6639)</b>			
The following settings are effective after transfer to the PC the next time operation is started.			
DM 6620	00 to 03	Input constant for IR 00000 to IR 00007 0: 8 ms; 1: 1 ms; 2: 2 ms; 3: 4 ms; 4: 8 ms; 5: 16 ms; 6: 32 ms; 7: 64 ms; 8: 128 ms	19
	04 to 07	Input constant for IR 00008 to IR 00015 (Setting same as bits 00 to 03)	
	08 to 15	Input constant for IR 001 00: 8 ms; 01: 1 ms; 02: 2 ms; 03: 4 ms; 04: 8 ms; 05: 16 ms; 06: 32 ms; 07: 64 ms; 08: 128 ms	
DM 6621	00 to 07	Input constant for IR 002 (Setting same as for IR 001.)	
	08 to 15	Input constant for IR 003 (Setting same as for IR 001.)	
DM 6622	00 to 07	Input constant for IR 004 (Setting same as for IR 001.)	
	08 to 15	Input constant for IR 005 (Setting same as for IR 001.)	
DM 6623	00 to 07	Input constant for IR 006 (Setting same as for IR 001.)	
	08 to 15	Input constant for IR 007 (Setting same as for IR 001.)	
DM 6624	00 to 07	Input constant for IR 008 (Setting same as for IR 001.)	
	08 to 15	Input constant for IR 009 (Setting same as for IR 001.)	
DM 6625	00 to 07	Input constant for IR 010 (Setting same as for IR 001.)	
	08 to 15	Input constant for IR 011 (Setting same as for IR 001.)	
DM 6626	00 to 07	Input constant for IR 012 (Setting same as for IR 001.)	
	08 to 15	Input constant for IR 013 (Setting same as for IR 001.)	
DM 6627	00 to 07	Input constant for IR 014 (Setting same as for IR 001.)	
	08 to 15	Input constant for IR 015 (Setting same as for IR 001.)	
DM 6628	00 to 03	Interrupt enable for IR 00000 (0: Normal input; 1: Interrupt input)	40
	04 to 07	Interrupt enable for IR 00001 (0: Normal input; 1: Interrupt input)	
	08 to 11	Interrupt enable for IR 00002 (0: Normal input; 1: Interrupt input)	
	12 to 15	Interrupt enable for IR 00003 (0: Normal input; 1: Interrupt input)	
DM 6629	00 to 07	Number of high-speed timers for interrupt refreshing 00 to 15 (BCD; e.g., set 15 for 00 to 14)	20
	08 to 15	High-speed timer interrupt refresh enable 00: 16 timers (setting in bits 00 to 07 disabled) 01: Use setting in 00 to 07	
DM 6630	00 to 07	First input refresh word for I/O interrupt 0: 00 to 11 (BCD)	40
	08 to 15	Number of input refresh words for I/O interrupt 0: 00 to 12 (BCD)	
DM 6631	00 to 07	First input refresh word for I/O interrupt 1: 00 to 11 (BCD)	
	08 to 15	Number of input refresh words for I/O interrupt 1: 00 to 12 (BCD)	
DM 6632	00 to 07	First input refresh word for I/O interrupt 2: 00 to 11 (BCD)	
	08 to 15	Number of input refresh words for I/O interrupt 2: 00 to 12 (BCD)	
DM 6633	00 to 07	First input refresh word for I/O interrupt 3: 00 to 11 (BCD)	
	08 to 15	Number of input refresh words for I/O interrupt 3: 00 to 12 (BCD)	
DM 6634	00 to 07	First input refresh word for high-speed counter 1: 00 to 11 (BCD)	40
	08 to 15	Number of input refresh words for high-speed counter 1: 00 to 12 (BCD)	
DM 6635	00 to 07	First input refresh word for high-speed counter 1: 00 to 11 (BCD)	40
	08 to 15	Number of input refresh words for high-speed counter 1: 00 to 12 (BCD)	
DM 6636	00 to 07	First input refresh word for interval timer 0: 00 to 07 (BCD)	45, 50
	08 to 15	Number of input refresh words for interval timer 0: 00 to 08 (BCD)	
DM 6637	00 to 07	First input refresh word for interval timer 1: 00 to 07 (BCD)	
	08 to 15	Number of input refresh words for interval timer 1: 00 to 08 (BCD)	
DM 6638	00 to 07	First input refresh word for interval timer 2 (also used for high-speed counter 0): 00 to 07 (BCD)	
	08 to 15	Number of input refresh words for interval timer 2: 00 to 08 (BCD) (also used for high-speed counter 0)	

Word(s)	Bit(s)	Function	Page
DM 6639	00 to 07	Output refresh method 00: Cyclic; 01: Direct	21, 383
	08 to 15	Number of digits for DIGITAL SWITCH (DSW(87)) instruction 00: 4 digits; 01: 8 digits	21, 122
<b>High-speed Counter Settings (DM 6640 to DM 6644)</b>			
The following settings are effective after transfer to the PC the next time operation is started.			
DM 6640, DM 6641	00 to 15	Not used.	
DM 6642	00 to 03	High-speed counter 0 mode 0: Up/down counter mode; 4: Incrementing counter mode	50
	04 to 07	High-speed counter 0 reset mode 0: Z phase and software reset; 1: Software reset only	
	08 to 15	High-speed counter 0 enable 00: Don't use high-speed counter; 01: Use high-speed counter with settings in 00 to 07	
DM 6643	00 to 03	CQM1-CPU43-EV1: Port 1 input setting 0: Differential phase input; 1: Pulse/Direction input; 2: Up/Down input  CQM1-CPU44-EV1: Port 1 input setting 0: 8-bit input; 1: 10-bit input; 2: 12-bit input	55, 62
	04 to 07	CQM1-CPU43-EV1: Port 1 reset setting 0: Z phase and software reset; 1: Software reset only  CQM1-CPU44-EV1: Not used. Set to 0.	57
	08 to 11	CQM1-CPU43-EV1: Port 1 counting mode setting 0: Linear mode; 1: Ring mode  CQM1-CPU44-EV1: Port 1 mode setting 0: BCD mode; 1: 360° mode	55, 62
	12 to 15	CQM1-CPU43-EV1: Port 1 pulse type setting 0: Standard pulse output (0.5 duty ratio); 1: Variable-duty-ratio pulse output  CQM1-CPU44-EV1: Not used. Set to 0.	25, 32
DM 6644	00 to 15	Port 2 settings (Identical to the port 1 settings in DM 6643.)	

Word(s)	Bit(s)	Function	Page																																																															
<b>RS-232C Port Settings</b>																																																																		
The following settings are effective after transfer to the PC.																																																																		
DM 6645	00 to 07	Port settings 00: Standard (1 start bit, 7-bit data, even parity, 2 stop bits, 9,600 bps) 01: Settings in DM 6646	88																																																															
	08 to 11	Link words for 1:1 link (Effective when bits 12 to 15 are set to 3.) 0: LR 00 to LR 63; 1: LR 00 to LR 31; 2: LR 00 to LR 15																																																																
	12 to 15	Communications mode 0: Host link; 1: RS-232C (no protocol); 2: 1:1 data link slave; 3: 1:1 data link master; 4: NT Link (1:1) (NT Link setting is only applicable to the CQM1-CPU4□-EV1 CPU Units.)																																																																
DM 6646	00 to 07	Baud rate 00: 1.2K, 01: 2.4K, 02: 4.8K, 03: 9.6K, 04: 19.2K																																																																
	08 to 15	Frame format <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th></th> <th>Start</th> <th>Length</th> <th>Stop</th> <th>Parity</th> </tr> </thead> <tbody> <tr><td>00:</td><td>1 bit</td><td>7 bits</td><td>1 bit</td><td>Even</td></tr> <tr><td>01:</td><td>1 bit</td><td>7 bits</td><td>1 bit</td><td>Odd</td></tr> <tr><td>02:</td><td>1 bit</td><td>7 bits</td><td>1 bit</td><td>None</td></tr> <tr><td>03:</td><td>1 bit</td><td>7 bits</td><td>2 bit</td><td>Even</td></tr> <tr><td>04:</td><td>1 bit</td><td>7 bits</td><td>2 bit</td><td>Odd</td></tr> <tr><td>05:</td><td>1 bit</td><td>7 bits</td><td>2 bit</td><td>None</td></tr> <tr><td>06:</td><td>1 bit</td><td>8 bits</td><td>1 bit</td><td>Even</td></tr> <tr><td>07:</td><td>1 bit</td><td>8 bits</td><td>1 bit</td><td>Odd</td></tr> <tr><td>08:</td><td>1 bit</td><td>8 bits</td><td>1 bit</td><td>None</td></tr> <tr><td>09:</td><td>1 bit</td><td>8 bits</td><td>2 bit</td><td>Even</td></tr> <tr><td>10:</td><td>1 bit</td><td>8 bits</td><td>2 bit</td><td>Odd</td></tr> <tr><td>11:</td><td>1 bit</td><td>8 bits</td><td>2 bit</td><td>None</td></tr> </tbody> </table>			Start	Length	Stop	Parity	00:	1 bit	7 bits	1 bit	Even	01:	1 bit	7 bits	1 bit	Odd	02:	1 bit	7 bits	1 bit	None	03:	1 bit	7 bits	2 bit	Even	04:	1 bit	7 bits	2 bit	Odd	05:	1 bit	7 bits	2 bit	None	06:	1 bit	8 bits	1 bit	Even	07:	1 bit	8 bits	1 bit	Odd	08:	1 bit	8 bits	1 bit	None	09:	1 bit	8 bits	2 bit	Even	10:	1 bit	8 bits	2 bit	Odd	11:	1 bit	8 bits
	Start	Length	Stop	Parity																																																														
00:	1 bit	7 bits	1 bit	Even																																																														
01:	1 bit	7 bits	1 bit	Odd																																																														
02:	1 bit	7 bits	1 bit	None																																																														
03:	1 bit	7 bits	2 bit	Even																																																														
04:	1 bit	7 bits	2 bit	Odd																																																														
05:	1 bit	7 bits	2 bit	None																																																														
06:	1 bit	8 bits	1 bit	Even																																																														
07:	1 bit	8 bits	1 bit	Odd																																																														
08:	1 bit	8 bits	1 bit	None																																																														
09:	1 bit	8 bits	2 bit	Even																																																														
10:	1 bit	8 bits	2 bit	Odd																																																														
11:	1 bit	8 bits	2 bit	None																																																														
DM 6647	00 to 15	Transmission delay (Host Link) 0000 to 9999 (BCD): Set in units of 10 ms, e.g., setting of 0001 equals 10 ms																																																																
DM 6648	00 to 07	Node number (Host link, effective when bits 12 to 15 of DM 6645 are set to 0.) 00 to 31 (BCD)																																																																
	08 to 11	Start code enable (RS-232C, effective when bits 12 to 15 of DM 6645 are set to 1.) 0: Disable; 1: Set																																																																
	12 to 15	End code enable (RS-232C, effective when bits 12 to 15 of DM 6645 are set to 1.) 0: Disable (number of bytes received) 1: Set (specified end code) 2: CR, LF																																																																
DM 6649	00 to 07	Start code (RS-232C) 00 to FF (binary)	88																																																															
	08 to 15	When bits 12 to 15 of DM 6648 are set to 0: Number of bytes received 00: Default setting (256 bytes) 01 to FF: 1 to 255 bytes  When bits 12 to 15 of DM 6648 are set to 1: End code (RS-232C) 00 to FF (binary)																																																																

Word(s)	Bit(s)	Function	Page																																																																
<b>Peripheral Port Settings</b>																																																																			
The following settings are effective after transfer to the PC. These settings are effective when a CQM1-CIF01 Connecting Cable is used. They are not effective when a CQM1-CIF11 Connecting Cable or Programming Console is used.																																																																			
DM 6650	00 to 07	Port settings 00: Standard (1 start bit, 7-bit data, even parity, 2 stop bits, 9,600 bps) 01: Settings in DM 6651	88																																																																
	08 to 11	Not used.																																																																	
	12 to 15	Communications mode 0: Host link; 1: RS-232C	88																																																																
DM 6651	00 to 07	Baud rate 00: 1.2K, 01: 2.4K, 02: 4.8K, 03: 9.6K, 04: 19.2K																																																																	
	08 to 15	Frame format <table border="0" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;"></th> <th style="text-align: left;">Start</th> <th style="text-align: left;">Length</th> <th style="text-align: left;">Stop</th> <th style="text-align: left;">Parity</th> </tr> </thead> <tbody> <tr><td>00:</td><td>1 bit</td><td>7 bits</td><td>1 bit</td><td>Even</td></tr> <tr><td>01:</td><td>1 bit</td><td>7 bits</td><td>1 bit</td><td>Odd</td></tr> <tr><td>02:</td><td>1 bit</td><td>7 bits</td><td>1 bit</td><td>None</td></tr> <tr><td>03:</td><td>1 bit</td><td>7 bits</td><td>2 bit</td><td>Even</td></tr> <tr><td>04:</td><td>1 bit</td><td>7 bits</td><td>2 bit</td><td>Odd</td></tr> <tr><td>05:</td><td>1 bit</td><td>7 bits</td><td>2 bit</td><td>None</td></tr> <tr><td>06:</td><td>1 bit</td><td>8 bits</td><td>1 bit</td><td>Even</td></tr> <tr><td>07:</td><td>1 bit</td><td>8 bits</td><td>1 bit</td><td>Odd</td></tr> <tr><td>08:</td><td>1 bit</td><td>8 bits</td><td>1 bit</td><td>None</td></tr> <tr><td>09:</td><td>1 bit</td><td>8 bits</td><td>2 bit</td><td>Even</td></tr> <tr><td>10:</td><td>1 bit</td><td>8 bits</td><td>2 bit</td><td>Odd</td></tr> <tr><td>11:</td><td>1 bit</td><td>8 bits</td><td>2 bit</td><td>None</td></tr> </tbody> </table>		Start	Length	Stop	Parity	00:	1 bit	7 bits	1 bit	Even	01:	1 bit	7 bits	1 bit	Odd	02:	1 bit	7 bits	1 bit	None	03:	1 bit	7 bits	2 bit	Even	04:	1 bit	7 bits	2 bit	Odd	05:	1 bit	7 bits	2 bit	None	06:	1 bit	8 bits	1 bit	Even	07:	1 bit	8 bits	1 bit	Odd	08:	1 bit	8 bits	1 bit	None	09:	1 bit	8 bits	2 bit	Even	10:	1 bit	8 bits	2 bit	Odd	11:	1 bit	8 bits	2 bit	None
	Start	Length	Stop	Parity																																																															
00:	1 bit	7 bits	1 bit	Even																																																															
01:	1 bit	7 bits	1 bit	Odd																																																															
02:	1 bit	7 bits	1 bit	None																																																															
03:	1 bit	7 bits	2 bit	Even																																																															
04:	1 bit	7 bits	2 bit	Odd																																																															
05:	1 bit	7 bits	2 bit	None																																																															
06:	1 bit	8 bits	1 bit	Even																																																															
07:	1 bit	8 bits	1 bit	Odd																																																															
08:	1 bit	8 bits	1 bit	None																																																															
09:	1 bit	8 bits	2 bit	Even																																																															
10:	1 bit	8 bits	2 bit	Odd																																																															
11:	1 bit	8 bits	2 bit	None																																																															
DM 6652	00 to 15	Transmission delay (Host Link) 0000 to 9999: In ms.																																																																	
DM 6653	00 to 07	Node number (Host link, effective when bits 12 to 15 of DM 6650 are set to 0.) 00 to 31 (BCD)																																																																	
	08 to 11	Start code enable (RS-232C, effective when bits 12 to 15 of DM 6650 are set to 1.) 0: Disable; 1: Set																																																																	
	12 to 15	End code enable (RS-232C, effective when bits 12 to 15 of DM 6650 are set to 1.) 0: Disable (number of bytes received) 1: Set (specified end code) 2: CR, LF																																																																	
DM 6654	00 to 07	Start code (RS-232C, effective when bits 08 to 11 of DM 6653 are set to 1.) 00 to FF (binary)																																																																	
	08 to 15	When bits 12 to 15 of DM 6653 are set to 0: Number of bytes received 00: Default setting (256 bytes) 01 to FF: 1 to 255 bytes  When bits 12 to 15 of DM 6653 are set to 1: End code (RS-232C) 00 to FF (binary)																																																																	
<b>Error Log Settings (DM 6655)</b>																																																																			
The following settings are effective after transfer to the PC.																																																																			
DM 6655	00 to 03	Style 0: Shift after 10 records have been stored 1: Store only first 10 records (no shifting) 2 to F: Do not store records	22																																																																
	04 to 07	Not used.																																																																	
	08 to 11	Cycle time monitor enable 0: Detect long cycles as non-fatal errors 1: Do not detect long cycles	22																																																																
	12 to 15	Low battery error enable 0: Detect low battery voltage as non-fatal error 1: Do not detect low batter voltage																																																																	

### 1-1-3 CPM1/CPM1A PC Setup Settings

The PC Setup is broadly divided into four categories: 1) Settings related to basic PC operation and I/O processes, 2) Settings related to the cycle time, 3) Settings related to interrupts, and 4) Settings related to communications. This section will explain the settings according to these classifications.

The following table shows the settings for CPM1/CPM1A PCs in order. Refer to the page number in the last column for more details on that setting.

Word(s)	Bit(s)	Function	Page
<b>Startup Processing (DM 6600 to DM 6614)</b>			
The following settings are effective after transfer to the PC only after the PC is restarted.			
DM 6600	00 to 07	Startup mode (effective when bits 08 to 15 are set to 02). 00: PROGRAM; 01: MONITOR 02: RUN	16
	08 to 15	Startup mode designation 00: Programming Console switch 01: Continue operating mode last used before power was turned off. (See note 1.) 02: Setting in 00 to 07	
DM 6601	00 to 07	Not used.	17
	08 to 11	IOM Hold Bit (SR 25212) Status at Startup 0: Reset; 1: Maintain (See note 3.)	
	12 to 15	Forced Status Hold Bit (SR 25211) Status at Startup 0: Reset; 1: Maintain (See note 3.)	
DM 6602	00 to 03	Program memory write-protection 0: Program memory unprotected 1: Program memory write-protected (except DM 6602 itself)	17
	04 to 07	Programming Console display language 0: English; 1: Japanese	
	08 to 15	Not used.	
DM 6603	00 to 15	Not used.	
DM 6604	00 to 07	00: If data could not be saved with the built-in capacitor (AR 1314 ON), a memory error will not be generated. 01: If data could not be saved with the built-in capacitor (AR 1314 ON), a memory error will be generated.	
	08 to 15	Not used.	
DM 6605 to DM 6614	00 to 15	Not used.	
<b>Cycle Time Settings (DM 6615 to DM 6619)</b>			
The following settings are effective after transfer to the PC the next time operation is started.			
DM 6615, DM 6616	00 to 15	Not used.	
DM 6617	00 to 07	Servicing time for peripheral port (effective when bits 08 to 15 are set to 01) 00 to 99 (BCD): Percentage of cycle time used to service peripheral.	18
	08 to 15	Peripheral port servicing setting enable 00: 5% of the cycle time 01: Use time in 00 to 07.	
DM 6618	00 to 07	Cycle monitor time (effective when bits 08 to 15 are set to 01, 02, or 03) 00 to 99 (BCD): Setting (see 08 to 15)	21
	08 to 15	Cycle monitor enable (Setting in 00 to 07 x unit; 99 s max.) 00: 120 ms (setting in bits 00 to 07 disabled) 01: Setting unit: 10 ms 02: Setting unit: 100 ms 03: Setting unit: 1 s	
DM 6619	00 to 15	Cycle time 0000: Variable (no minimum) 0001 to 9999 (BCD): Minimum time in ms	18

Word(s)	Bit(s)	Function	Page
<b>Interrupt Processing (DM 6620 to DM 6639)</b>			
The following settings are effective after transfer to the PC the next time operation is started.			
DM 6620	00 to 03	Input constant for IR 00000 to IR 00002 0: 8 ms; 1: 1 ms; 2: 2 ms; 3: 4 ms; 4: 8 ms; 5: 16 ms; 6: 32 ms; 7: 64 ms; 8: 128 ms	19
	04 to 07	Input constant for IR 00003 and IR 00004 (Setting same as bits 00 to 03)	
	08 to 11	Input constant for IR 00005 and IR 00006 (Setting same as bits 00 to 03)	
	12 to 15	Input constant for IR 00007 and IR 00011 (Setting same as bits 00 to 03)	
DM 6621	00 to 07	Input constant for IR 001 00: 8 ms; 01: 1 ms; 02: 2 ms; 03: 4 ms; 04: 8 ms; 05: 16 ms; 06: 32 ms; 07: 64 ms; 08: 128 ms	
	08 to 15	Input constant for IR 002 (Setting same as for IR 001.)	
DM 6622	00 to 07	Input constant for IR 003 (Setting same as for IR 001.)	
	08 to 15	Input constant for IR 004 (Setting same as for IR 001.)	
DM 6623	00 to 07	Input constant for IR 005 (Setting same as for IR 001.)	
	08 to 15	Input constant for IR 006 (Setting same as for IR 001.)	
DM 6624	00 to 07	Input constant for IR 007 (Setting same as for IR 001.)	
	08 to 15	Input constant for IR 008 (Setting same as for IR 001.)	
DM 6625	00 to 07	Input constant for IR 009 (Setting same as for IR 001.)	
	08 to 15	Not used.	
DM 6626 to DM 6627	00 to 15	Not used.	
DM6628	00 to 03	Interrupt enable for IR 00003 (0: Normal input; 1: Interrupt input; 2: Quick-response)	40
	04 to 07	Interrupt enable for IR 00004 (0: Normal input; 1: Interrupt input; 2: Quick-response)	
	08 to 11	Interrupt enable for IR 00005 (0: Normal input; 1: Interrupt input; 2: Quick-response)	
	12 to 15	Interrupt enable for IR 00006 (0: Normal input; 1: Interrupt input; 2: Quick-response)	
DM 6629 to DM 6641	00 to 15	Not used.	40
<b>High-speed Counter Settings (DM 6640 to DM 6644)</b>			
The following settings are effective after transfer to the PC the next time operation is started.			
DM 6640 to DM 6641	00 to 15	Not used.	
DM 6642	00 to 03	High-speed counter mode 0: Up/down counter mode; 4: Incrementing counter mode	50
	04 to 07	High-speed counter reset mode 0: Z phase and software reset; 1: Software reset only	
	08 to 15	High-speed counter enable 00: Don't use high-speed counter; 01: Use high-speed counter with settings in 00 to 07	
DM 6643, DM 6644	00 to 15	Not used.	

Word(s)	Bit(s)	Function	Page																																																															
<b>Peripheral Port Settings</b>																																																																		
The following settings are effective after transfer to the PC.																																																																		
DM 6645 to DM 6649	00 to 15	Not used.	88																																																															
DM 6650	00 to 07	Port settings 00: Standard (1 start bit, 7-bit data, even parity, 2 stop bits, 9,600 bps) 01: Settings in DM 6651 (Other settings will cause a non-fatal error and AR 1302 will turn ON.)																																																																
	08 to 11	Link area for one-to-one PC link via peripheral port: 0: LR 00 to LR 15																																																																
	12 to 15	Communications mode 0: Host link; 2: One-to-one PC link (slave); 3: One-to-one PC link (master); 4: NT Link (Other settings will cause a non-fatal error and AR 1302 will turn ON.)																																																																
DM 6651	00 to 07	Baud rate 00: 1.2K, 01: 2.4K, 02: 4.8K, 03: 9.6K, 04: 19.2K, 05 to 07: Cannot be used (see note 2) (Other settings will cause a non-fatal error and AR 1302 will turn ON.)																																																																
	08 to 15	Frame format <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th></th> <th>Start</th> <th>Length</th> <th>Stop</th> <th>Parity</th> </tr> </thead> <tbody> <tr><td>00:</td><td>1 bit</td><td>7 bits</td><td>1 bit</td><td>Even</td></tr> <tr><td>01:</td><td>1 bit</td><td>7 bits</td><td>1 bit</td><td>Odd</td></tr> <tr><td>02:</td><td>1 bit</td><td>7 bits</td><td>1 bit</td><td>None</td></tr> <tr><td>03:</td><td>1 bit</td><td>7 bits</td><td>2 bit</td><td>Even</td></tr> <tr><td>04:</td><td>1 bit</td><td>7 bits</td><td>2 bit</td><td>Odd</td></tr> <tr><td>05:</td><td>1 bit</td><td>7 bits</td><td>2 bit</td><td>None</td></tr> <tr><td>06:</td><td>1 bit</td><td>8 bits</td><td>1 bit</td><td>Even</td></tr> <tr><td>07:</td><td>1 bit</td><td>8 bits</td><td>1 bit</td><td>Odd</td></tr> <tr><td>08:</td><td>1 bit</td><td>8 bits</td><td>1 bit</td><td>None</td></tr> <tr><td>09:</td><td>1 bit</td><td>8 bits</td><td>2 bit</td><td>Even</td></tr> <tr><td>10:</td><td>1 bit</td><td>8 bits</td><td>2 bit</td><td>Odd</td></tr> <tr><td>11:</td><td>1 bit</td><td>8 bits</td><td>2 bit</td><td>None</td></tr> </tbody> </table> (Other settings will cause a non-fatal error and AR 1302 will turn ON.)			Start	Length	Stop	Parity	00:	1 bit	7 bits	1 bit	Even	01:	1 bit	7 bits	1 bit	Odd	02:	1 bit	7 bits	1 bit	None	03:	1 bit	7 bits	2 bit	Even	04:	1 bit	7 bits	2 bit	Odd	05:	1 bit	7 bits	2 bit	None	06:	1 bit	8 bits	1 bit	Even	07:	1 bit	8 bits	1 bit	Odd	08:	1 bit	8 bits	1 bit	None	09:	1 bit	8 bits	2 bit	Even	10:	1 bit	8 bits	2 bit	Odd	11:	1 bit	8 bits
	Start	Length	Stop	Parity																																																														
00:	1 bit	7 bits	1 bit	Even																																																														
01:	1 bit	7 bits	1 bit	Odd																																																														
02:	1 bit	7 bits	1 bit	None																																																														
03:	1 bit	7 bits	2 bit	Even																																																														
04:	1 bit	7 bits	2 bit	Odd																																																														
05:	1 bit	7 bits	2 bit	None																																																														
06:	1 bit	8 bits	1 bit	Even																																																														
07:	1 bit	8 bits	1 bit	Odd																																																														
08:	1 bit	8 bits	1 bit	None																																																														
09:	1 bit	8 bits	2 bit	Even																																																														
10:	1 bit	8 bits	2 bit	Odd																																																														
11:	1 bit	8 bits	2 bit	None																																																														
DM 6652	00 to 15	Transmission delay (Host Link) (See note 4.) 0000 to 9999: In ms. (Other settings will cause a non-fatal error and AR 1302 will turn ON.)																																																																
DM 6653	00 to 07	Node number (Host link) 00 to 31 (BCD) (Other settings will cause a non-fatal error and AR 1302 will turn ON.)																																																																
	08 to 15	Not used.																																																																
DM 6654	00 to 15	Not used.																																																																
<b>Error Log Settings (DM 6655)</b>																																																																		
The following settings are effective after transfer to the PC.																																																																		
DM 6655	00 to 03	Style 0: Shift after 7 records have been stored 1: Store only first 7 records (no shifting) 2 to F: Do not store records	22																																																															
	04 to 07	Not used.																																																																
	08 to 11	Cycle time monitor enable 0: Detect long cycles as non-fatal errors 1: Do not detect long cycles																																																																
	12 to 15	Not used.																																																																

- Note** 1. When the startup mode is set to continue the operating mode last used before the power was turned off, that operating mode will be retained by the built-in capacitor. If the power remains off for longer than the backup time of the capacitor, the data may be lost. (For details on the holding time, refer to the *CPM1A* or *CPM1 Operation Manual*.)

2. Do not set to "05" to "07." If set to this value, the CPM1/CPM1A will not operate properly and the RUN PC Setup Error Flag (AR 1302 ON) will not turn ON.

3. **Retention of IOM Hold Bit (SR 25212) Status**

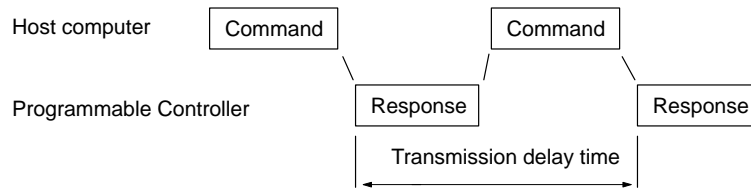
If the "IOM Hold Bit Status at Startup" (DM 6601, bits 08 to 11) is set to "Maintain" with the IOM Hold Bit (SR 25212) turned ON, operation can be started with the I/O memory (I/O, IR, LR) status just as it was before the power was turned OFF. (The input area is refreshed at startup, however, so it is overwritten by the most recently updated input status.)

**Retention of Forced Status Hold Bit (SR 25211) Status**

If the "Forced Status Hold Bit Status at Startup" (DM 6601, bits 12 to 15) is set to "Maintain" with the Forced Status Hold Bit (SR 25211) turned ON, operation can be started with the forced set/reset status just as it was before the power was turned OFF. (When starting up in RUN Mode, however, the forced set/reset status is cleared.)

Even if the "IOM Hold Bit Status at Startup" or "Forced Status Hold Bit Status at Startup" is set to "Maintain," the IOM Hold Bit (SR 25212) or Forced Status Hold Bit (SR 25211) status may be cleared if the power remains OFF for longer than the backup time of the built-in capacitor. (For details on the holding time, refer to the *CPM1A* or *CPM1 Operation Manual*.) At this time the I/O memory will also be cleared, so set up the system so that clearing the I/O memory will not cause problems.

4. The transmission delay is the delay between the previous transmission and the next transmission.



5. If an out-of-range value is set, the following communications conditions will result. In that case, reset the value so that it is within the permissible range.

Communications mode: Host Link  
 Communications format: Standard settings  
 (1 start bit, 7-bit data; even parity, 2 stop bits, 9,600 bps)  
 Transmission delay: No  
 Node number: 00



## 1-1-4 SRM1 PC Setup Settings

The PC Setup is broadly divided into three categories: 1) Settings related to basic PC operation and I/O processes, 2) Settings related to the cycle time, and 3) Settings related to communications. This section will explain the settings according to these classifications.

The following table shows the settings for SRM1 PCs in order. Refer to the page number in the last column for more details on that setting.

Word(s)	Bit(s)	Function	Page
<b>Startup Processing (DM 6600 to DM 6614)</b>			
The following settings are effective after transfer to the PC only after the PC is restarted.			
DM 6600	00 to 07	Startup mode (effective when bits 08 to 15 are set to 02). 00: PROGRAM; 01: MONITOR 02: RUN	16
	08 to 15	Startup mode designation 00: Programming Console switch 01: Continue operating mode last used before power was turned off 02: Setting in 00 to 07	
DM 6601	00 to 07	Not used.	17
	08 to 11	IOM Hold Bit (SR 25212) Status 0: Reset; 1: Maintain (See caution on page 17.)	
	12 to 15	Forced Status Hold Bit (SR 25211) Status 0: Reset; 1: Maintain	
DM 6602	00 to 03	Program memory write-protection 0: Program memory unprotected 1: Program memory write-protected (except DM 6602 itself)	17
	04 to 07	Programming Console display language 0: English; 1: Japanese	
	08 to 11	Expansion Instructions 0: Default settings; 1: User settings	
	12 to 15	Not used.	
DM 6603	00 to 03	Maximum number of CompoBus/S devices 0: Max. no. 32 1: Max. no. 16	
	04 to 15	Not used.	
DM 6604	00 to 07	00: If data could not be saved for a power interruption (AR 1314 ON), a memory error will not be generated. 01: If data could not be saved for a power interruption (AR 1314 ON), a memory error will be generated.	
	08 to 15	Not used.	
DM 6605 to DM 6614	00 to 15	Not used.	
<b>Cycle Time Settings (DM 6615 to DM 6619)</b>			
The following settings are effective after transfer to the PC the next time operation is started.			
DM 6615	00 to 15	Not used.	
DM 6616	00 to 07	Servicing time for RS-232C port (effective when bits 08 to 15 are set) 00 to 99 (BCD): Percentage for cycle time used to service peripheral.	18
	08 to 15	RS-232C port servicing enable 00: 5% of the cycle time 01: Use time in 00 to 07.	
DM 6617	00 to 07	Servicing time for peripheral port (effective when bits 08 to 15 are set to 01) 00 to 99 (BCD): Percentage of cycle time used to service peripheral.	18
	08 to 15	Peripheral port servicing setting enable 00: 5% of the cycle time 01: Use time in 00 to 07.	

Word(s)	Bit(s)	Function	Page																																																															
DM 6618	00 to 07	Cycle monitor time (effective when bits 08 to 15 are set to 01, 02, or 03) 00 to 99 (BCD): Setting (see 08 to 15)	21																																																															
	08 to 15	Cycle monitor enable (Setting in 00 to 07 x unit; 99 s max.) 00: 120 ms (setting in bits 00 to 07 disabled) 01: Setting unit: 10 ms 02: Setting unit: 100 ms 03: Setting unit: 1 s																																																																
DM 6619	00 to 15	Cycle time 0000: Variable (no minimum) 0001 to 9999 (BCD): Minimum time in ms	18																																																															
DM 6620 to DM 6644	00 to 15	Not used.																																																																
<b>RS-232C Port Settings</b>																																																																		
The following settings are effective after transfer to the PC.																																																																		
DM 6645	00 to 03	Port settings 0: Standard (1 start bit, 7-bit data, even parity, 2 stop bits, 9,600 bps) 1: Settings in DM 6646	95																																																															
	04 to 07	CTS control settings 0: Disable; 1: Set																																																																
	08 to 11	Link words for 1:1 link 0: LR 00 to LR 15; Other: Not effective																																																																
	12 to 15	Communications mode 0: Host link; 1: RS-232C (no protocol); 2: 1:1 data link slave; 3: 1:1 data link master; 4: NT Link																																																																
DM 6646	00 to 07	Baud rate 00: 1.2K, 01: 2.4K, 02: 4.8K, 03: 9.6K, 04: 19.2K																																																																
	08 to 15	Frame format <table border="1" style="margin-left: 20px;"> <thead> <tr> <th></th> <th>Start</th> <th>Length</th> <th>Stop</th> <th>Parity</th> </tr> </thead> <tbody> <tr><td>00:</td><td>1 bit</td><td>7 bits</td><td>1 bit</td><td>Even</td></tr> <tr><td>01:</td><td>1 bit</td><td>7 bits</td><td>1 bit</td><td>Odd</td></tr> <tr><td>02:</td><td>1 bit</td><td>7 bits</td><td>1 bit</td><td>None</td></tr> <tr><td>03:</td><td>1 bit</td><td>7 bits</td><td>2 bit</td><td>Even</td></tr> <tr><td>04:</td><td>1 bit</td><td>7 bits</td><td>2 bit</td><td>Odd</td></tr> <tr><td>05:</td><td>1 bit</td><td>7 bits</td><td>2 bit</td><td>None</td></tr> <tr><td>06:</td><td>1 bit</td><td>8 bits</td><td>1 bit</td><td>Even</td></tr> <tr><td>07:</td><td>1 bit</td><td>8 bits</td><td>1 bit</td><td>Odd</td></tr> <tr><td>08:</td><td>1 bit</td><td>8 bits</td><td>1 bit</td><td>None</td></tr> <tr><td>09:</td><td>1 bit</td><td>8 bits</td><td>2 bit</td><td>Even</td></tr> <tr><td>10:</td><td>1 bit</td><td>8 bits</td><td>2 bit</td><td>Odd</td></tr> <tr><td>11:</td><td>1 bit</td><td>8 bits</td><td>2 bit</td><td>None</td></tr> </tbody> </table>			Start	Length	Stop	Parity	00:	1 bit	7 bits	1 bit	Even	01:	1 bit	7 bits	1 bit	Odd	02:	1 bit	7 bits	1 bit	None	03:	1 bit	7 bits	2 bit	Even	04:	1 bit	7 bits	2 bit	Odd	05:	1 bit	7 bits	2 bit	None	06:	1 bit	8 bits	1 bit	Even	07:	1 bit	8 bits	1 bit	Odd	08:	1 bit	8 bits	1 bit	None	09:	1 bit	8 bits	2 bit	Even	10:	1 bit	8 bits	2 bit	Odd	11:	1 bit	8 bits
	Start	Length	Stop	Parity																																																														
00:	1 bit	7 bits	1 bit	Even																																																														
01:	1 bit	7 bits	1 bit	Odd																																																														
02:	1 bit	7 bits	1 bit	None																																																														
03:	1 bit	7 bits	2 bit	Even																																																														
04:	1 bit	7 bits	2 bit	Odd																																																														
05:	1 bit	7 bits	2 bit	None																																																														
06:	1 bit	8 bits	1 bit	Even																																																														
07:	1 bit	8 bits	1 bit	Odd																																																														
08:	1 bit	8 bits	1 bit	None																																																														
09:	1 bit	8 bits	2 bit	Even																																																														
10:	1 bit	8 bits	2 bit	Odd																																																														
11:	1 bit	8 bits	2 bit	None																																																														
DM 6647	00 to 15	Transmission delay (Host Link) 0000 to 9999 (BCD): Set in units of 10 ms, e.g., setting of 0001 equals 10 ms																																																																
DM 6648	00 to 07	Node number (Host link, effective when bits 12 to 15 of DM 6645 are set to 0.) 00 to 31 (BCD)																																																																
	08 to 11	Start code enable (RS-232C, effective when bits 12 to 15 of DM 6645 are set to 1.) 0: Disable; 1: Set																																																																
	12 to 15	End code enable (RS-232C, effective when bits 12 to 15 of DM 6645 are set to 1.) 0: Disable (number of bytes received) 1: Set (specified end code) 2: CR, LF																																																																
DM 6649	00 to 07	Start code (RS-232C) 00 to FF (binary)	95																																																															
	08 to 15	When bits 12 to 15 of DM 6648 are set to 0: Number of bytes received 00: Default setting (256 bytes) 01 to FF: 1 to 255 bytes  When bits 12 to 15 of DM 6648 are set to 1: End code (RS-232C) 00 to FF (binary)																																																																

Word(s)	Bit(s)	Function	Page																																																															
<b>Peripheral Port Settings</b>																																																																		
The following settings are effective after transfer to the PC.																																																																		
DM 6650	00 to 03	Port settings 00: Standard (1 start bit, 7-bit data, even parity, 2 stop bits, 9,600 bps) 01: Settings in DM 6651 (Other settings will cause a non-fatal error and AR 1302 will turn ON.)	95																																																															
	04 to 07	Not used.																																																																
	08 to 11	Not used.																																																																
	12 to 15	Communications mode 0: Host link; 1: No protocol (Other settings will cause a non-fatal error and and AR 1302 will turn ON.)																																																																
DM 6651	00 to 07	Baud rate 00: 1.2K, 01: 2.4K, 02: 4.8K, 03: 9.6K, 04: 19.2K																																																																
	08 to 15	Frame format <table border="1"> <thead> <tr> <th></th> <th>Start</th> <th>Length</th> <th>Stop</th> <th>Parity</th> </tr> </thead> <tbody> <tr> <td>00:</td> <td>1 bit</td> <td>7 bits</td> <td>1 bit</td> <td>Even</td> </tr> <tr> <td>01:</td> <td>1 bit</td> <td>7 bits</td> <td>1 bit</td> <td>Odd</td> </tr> <tr> <td>02:</td> <td>1 bit</td> <td>7 bits</td> <td>1 bit</td> <td>None</td> </tr> <tr> <td>03:</td> <td>1 bit</td> <td>7 bits</td> <td>2 bit</td> <td>Even</td> </tr> <tr> <td>04:</td> <td>1 bit</td> <td>7 bits</td> <td>2 bit</td> <td>Odd</td> </tr> <tr> <td>05:</td> <td>1 bit</td> <td>7 bits</td> <td>2 bit</td> <td>None</td> </tr> <tr> <td>06:</td> <td>1 bit</td> <td>8 bits</td> <td>1 bit</td> <td>Even</td> </tr> <tr> <td>07:</td> <td>1 bit</td> <td>8 bits</td> <td>1 bit</td> <td>Odd</td> </tr> <tr> <td>08:</td> <td>1 bit</td> <td>8 bits</td> <td>1 bit</td> <td>None</td> </tr> <tr> <td>09:</td> <td>1 bit</td> <td>8 bits</td> <td>2 bit</td> <td>Even</td> </tr> <tr> <td>10:</td> <td>1 bit</td> <td>8 bits</td> <td>2 bit</td> <td>Odd</td> </tr> <tr> <td>11:</td> <td>1 bit</td> <td>8 bits</td> <td>2 bit</td> <td>None</td> </tr> </tbody> </table> (Other settings will cause a non-fatal error and AR 1302 will turn ON.)			Start	Length	Stop	Parity	00:	1 bit	7 bits	1 bit	Even	01:	1 bit	7 bits	1 bit	Odd	02:	1 bit	7 bits	1 bit	None	03:	1 bit	7 bits	2 bit	Even	04:	1 bit	7 bits	2 bit	Odd	05:	1 bit	7 bits	2 bit	None	06:	1 bit	8 bits	1 bit	Even	07:	1 bit	8 bits	1 bit	Odd	08:	1 bit	8 bits	1 bit	None	09:	1 bit	8 bits	2 bit	Even	10:	1 bit	8 bits	2 bit	Odd	11:	1 bit	8 bits
	Start	Length	Stop	Parity																																																														
00:	1 bit	7 bits	1 bit	Even																																																														
01:	1 bit	7 bits	1 bit	Odd																																																														
02:	1 bit	7 bits	1 bit	None																																																														
03:	1 bit	7 bits	2 bit	Even																																																														
04:	1 bit	7 bits	2 bit	Odd																																																														
05:	1 bit	7 bits	2 bit	None																																																														
06:	1 bit	8 bits	1 bit	Even																																																														
07:	1 bit	8 bits	1 bit	Odd																																																														
08:	1 bit	8 bits	1 bit	None																																																														
09:	1 bit	8 bits	2 bit	Even																																																														
10:	1 bit	8 bits	2 bit	Odd																																																														
11:	1 bit	8 bits	2 bit	None																																																														
DM 6652	00 to 15	Transmission delay (Host Link) 0000 to 9999 (BCD): Set in units of 10 ms. (Other settings will cause a non-fatal error and AR 1302 will turn ON.)																																																																
DM 6653	00 to 07	Node number (Host link) 00 to 31 (BCD) (Other settings will cause a non-fatal error and AR 1302 will turn ON.)																																																																
	08 to 11	Start code enable (RS-232C, effective when bits 12 to 15 of DM6650 are set to 1.) 0: Disable 1: Set																																																																
	12 to 15	End code enable (RS-232C, effective when bits 12 to 15 of DM6650 are set to 1.) 0: Disable (number of bytes received) 1: Set (specified end code) 2: CR, LF																																																																
DM 6654	00 to 07	Start code (effective when bits 08 to 11 of DM6650 are set to 1.) 00: 256 bytes 01 to FF: 1 to 255 bytes																																																																
	08 to 15	End code When bits 12 to 15 of DM6653 are set to 0: 00: 256 bytes 01 to FF: 1 to 255 bytes When bits 12 to 15 of DM6653 are set to 1: Setting: 00 to FF (binary)																																																																

Word(s)	Bit(s)	Function	Page
<b>Error Log Settings (DM 6655)</b>			
The following settings are effective after transfer to the PC.			
DM 6655	00 to 03	Style 0: Shift after 7 records have been stored 1: Store only first 7 records Errors will not be stored if other values are set.	22
	04 to 07	Not used.	
	08 to 11	Cycle time monitor enable 0: Detect long cycles as non-fatal errors 1: Do not detect long cycles	
	12 to 15	Not used.	

**Note** If an out-of-range value is set, the following communications conditions will result. In that case, reset the value so that it is within the permissible range.

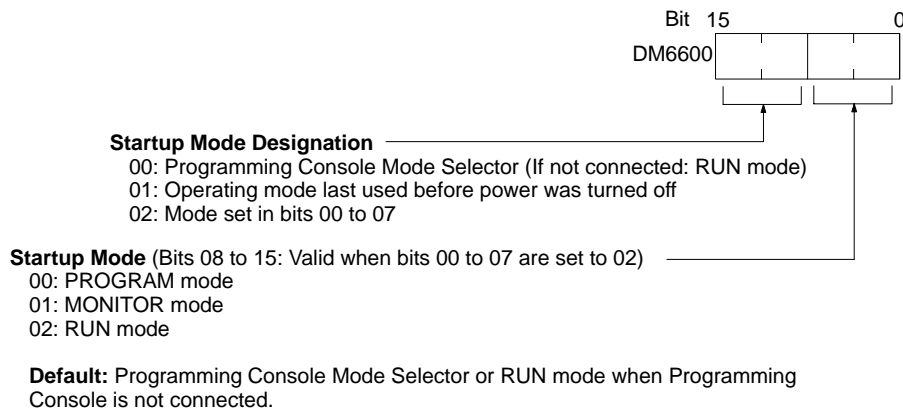
- Communications mode: Host Link
- Communications format: Standard settings  
(1 start bit, 7-bit data; even parity, 2 stop bits, 9,600 bps)
- Transmission delay: No
- Node number: 00

## 1-2 Basic PC Operation and I/O Processes

This section explains the PC Setup settings related to basic operation and I/O processes.

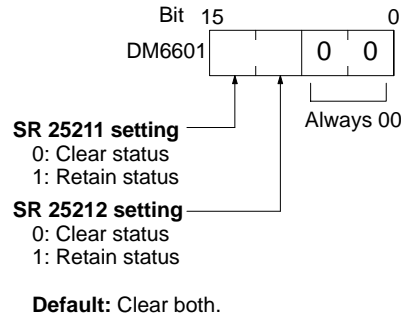
### 1-2-1 Startup Mode

The operation mode the PC will start in when power is turned on can be set as shown below.



### 1-2-2 Hold Bit Status

Make the settings shown below to determine whether, when the power supply is turned on, the Forced Status Hold Bit (SR 25211) and/or IOM Hold Bit (SR 25212) will retain the status that was in effect when the power was last turned off, or whether the previous status will be cleared.



The Forced Status Hold Bit (SR 25211) determines whether or not the forced set/reset status is retained when changing from PROGRAM mode to MONITOR mode.

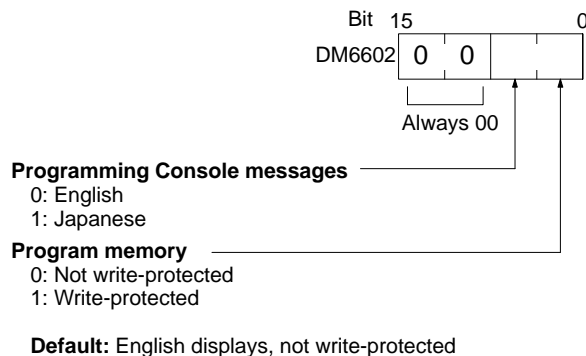
The IOM Hold Bit (SR 25212) determines whether or not the status of IR bits and LR bits is retained when PC operation is started and stopped.

**Caution** Do not use the I/O Hold Bit Status and Forced Status Hold Bit Status Bits (DM 6601) when the power to the PC is going to be turned off longer than the memory backup time of the internal capacitor. If the memory backup time is exceeded, memory status will be unstable even if the I/O Hold Bit Status and Forced Status Hold Bit Status Bits are used. Unpredictable results may occur if operation is attempted with unstable memory status.

- Note**
1. The memory backup time of the internal capacitor varies with the ambient temperature, but is 20 days at 25°C. Refer to hardware specifications for more details.
  2. The memory backup time assumes that the internal capacitor is fully charged before power is turned off. Fully charging the capacitor requires that power is supplied to the CPU Unit for at least 15 minutes.

### 1-2-3 Program Memory Write-protection (CPM1/CPM1A Only)

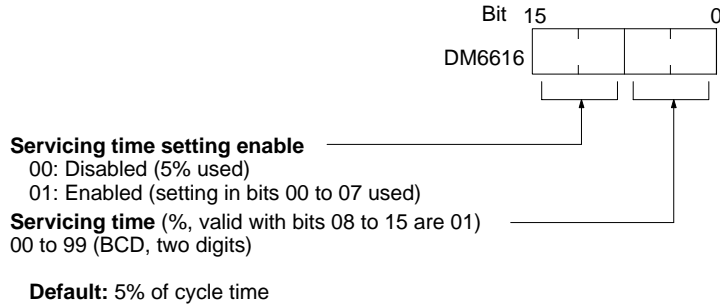
In CPM1/CPM1A PCs the program memory can be protected by setting bits 00 to 03 of DM 6602 to 0. Bits 04 to 07 determine whether Programming Console messages are displayed in English or Japanese.



**Note** DM 6602 itself can still be changed after the program memory has been write-protected by setting bits 04 to 07 of DM 6602 to 1.

### 1-2-4 RS-232C Port Servicing Time (CQM1/SRM1 Only)

The following settings are used to determine the percentage of the cycle time devoted to servicing the RS-232C port.



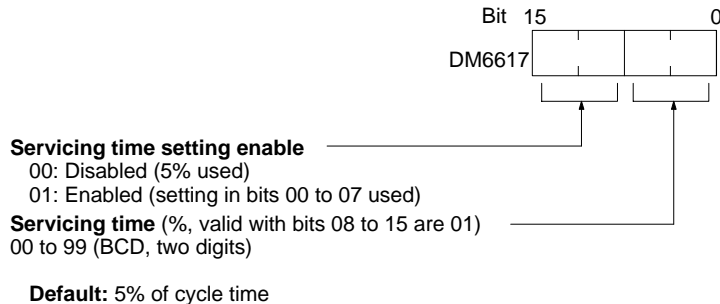
**Example:** If DM 6616 is set to 0110, the RS-232C port will be serviced for 10% of the cycle time.

The servicing time will be 0.34 ms minimum.

The entire servicing time will not be used unless processing requests exist.

### 1-2-5 Peripheral Port Servicing Time

The following settings are used to determine the percentage of the cycle time devoted to servicing the peripheral port.



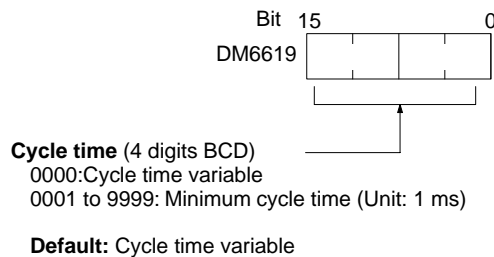
**Example:** If DM 6617 is set to 0115, the peripheral port will be serviced for 15% of the cycle time.

The servicing time will be 0.34 ms minimum.

The entire servicing time will not be used unless processing requests exist.

### 1-2-6 Cycle Time

Make the settings shown below to standardize the cycle time and to eliminate variations in I/O response time by setting a minimum cycle time.

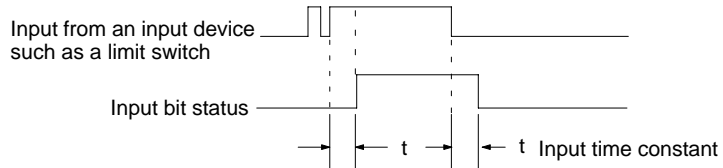


If the actual cycle time is shorter than the minimum cycle time, execution will wait until the minimum time has expired. If the actual cycle time is longer than the minimum cycle time, then operation will proceed according to the actual cycle time. AR 2405 will turn ON if the minimum cycle time is exceeded.

### 1-2-7 Input Time Constants

Make the settings shown below to set the time from when the actual inputs from the DC Input Unit are turned ON or OFF until the corresponding input bits are updated (i.e., until their ON/OFF status is changed). Make these settings when you want to adjust the time until inputs stabilize.

Increasing the input time constant can reduce the effects from chattering and external noise.

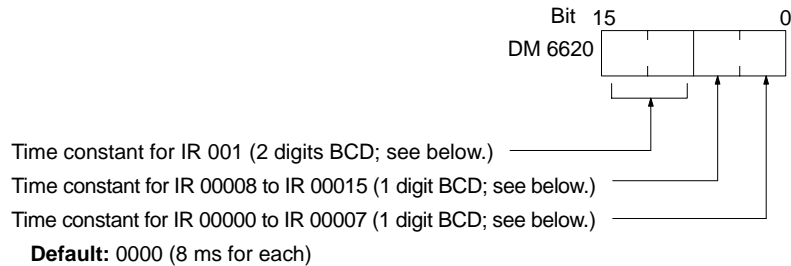


The SRM1 does not have this setting.

#### CQM1 PCs

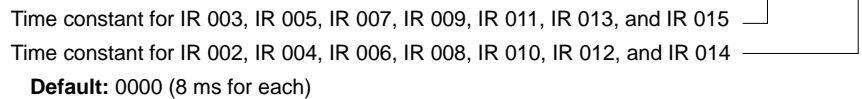
DM 6620 contains the input time constants for both IR 000 and IR 001.

#### Input Time Constants for IR 000 and IR 001



#### Input Time Constants for IR 002 to IR 015

- DM 6621: IR 002 and IR 003
- DM 6622: IR 004 and IR 005
- DM 6623: IR 006 and IR 007
- DM 6624: IR 008 and IR 009
- DM 6625: IR 010 and IR 011
- DM 6626: IR 012 and IR 013
- DM 6627: IR 014 and IR 015



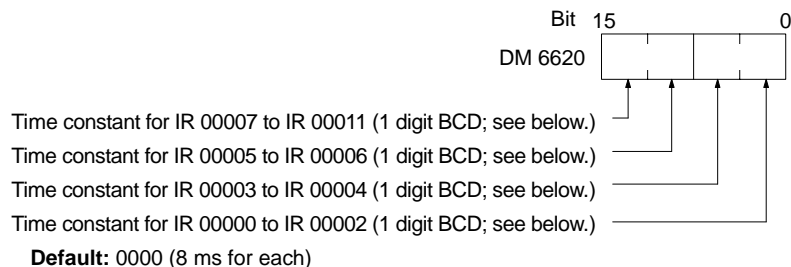
The nine possible settings for the input time constant are shown below. Set only the rightmost digit for IR 000.

- |          |          |          |           |         |
|----------|----------|----------|-----------|---------|
| 0: 8 ms  | 1: 1 ms  | 2: 2 ms  | 3: 4 ms   | 4: 8 ms |
| 5: 16 ms | 6: 32 ms | 7: 64 ms | 8: 128 ms |         |

#### CPM1/CPM1A PCs

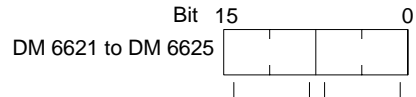
Set the input time constants for CPM1/CPM1A inputs from a Peripheral Device.

#### Input Time Constants for IR 000



**Input Time Constants for IR 001 to IR 009**

- DM 6621: IR 001 and IR 002
- DM 6622: IR 003 and IR 004
- DM 6623: IR 005 and IR 006
- DM 6624: IR 007 and IR 008
- DM 6625: IR 009



Time constant for IR 002, IR 004, IR 006, and IR 008  
 Time constant for IR 001, IR 003, IR 005, IR 007, and IR 009  
**Default:** 0000 (8 ms for each)

The nine possible settings for the input time constant are shown below. Set only the rightmost digit for IR 000.

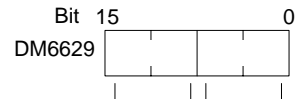
- |          |          |          |           |         |
|----------|----------|----------|-----------|---------|
| 0: 8 ms  | 1: 1 ms  | 2: 2 ms  | 3: 4 ms   | 4: 8 ms |
| 5: 16 ms | 6: 32 ms | 7: 64 ms | 8: 128 ms |         |

The CPM1/CPM1A's I/O response time is the input time constant (1 ms to 128 ms; default is 8 ms) + the cycle time.

Refer to 7-2 CPM1/CPM1A Cycle Time and I/O Response Time for more details.

**1-2-8 High-speed Timers (CQM1 Only)**

Make the settings shown below to set the number of high-speed timers created with TIMH(15) that will use interrupt processing.



**High-speed timer interrupt setting enable**  
 00: Setting disabled (Interrupt processing for all high-speed timers, TIM 000 to TIM 015)  
 01: Enabled (Use setting in bits 00 to 07.)

**Number of high-speed timer for interrupts** (valid when bits 08 to 15 are 01)  
 00 to 15 (2 digits BCD)

**Default:** Interrupt processing for all high-speed timers, TIM 000 to TIM 015.

The setting indicates the number of timers that will use interrupt processing beginning with TIM 000. For example, if "0108" is specified, then eight timers, TIM 000 to TIM 007 will use interrupt processing.

**Note** High-speed timers will not be accurate without interrupt processing unless the cycle time is 10 ms or less.

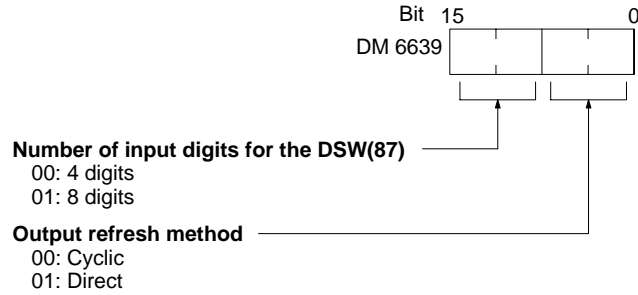
Interrupt response time for other interrupts will be improved if interrupt processing is set to 00 when high-speed timer processing is not required. This includes any time the cycle time is less than 10 ms.

**Note** If the SPED(64) instruction is used and pulses are output at a frequency of 500 Hz or greater, then set the number of high-speed timers with interrupt processing to four or less. Refer to information on the SPED(64) instruction for details.



### 1-2-9 DSW(87) Input Digits & Output Refresh Method (CQM1 Only)

Make the settings shown below to set the number of input digits the DSW(87) instruction, and to set the output refresh method.

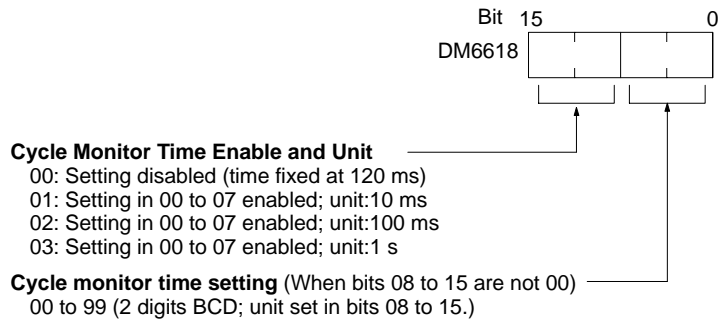


**Default:** The number of input digits for the DSW(87) instruction is set to "4" and the output refresh method is cyclic.

Refer to *Section 2 New Features* for details on the DSW(87) instruction and to *Section 7 PC Operations and Processing Time* for details on I/O refresh methods.

### 1-2-10 Error Log Settings

Make the settings shown below for detecting errors and storing the error log.  
**Cycle Monitor Time (DM 6618)**



**Default:** 120 ms.

The cycle monitor time is used for checking for extremely long cycle times, as can happen when the program goes into an infinite loop. If the cycle time exceeds the cycle monitor setting, a fatal error (FALS 9F) will be generated.

**Note** 1. The unit used for the maximum and current cycle times recorded in the AR area (AR 26 and AR 27 in the CQM1, AR 14 and AR 15 in the CPM1/CPM1A/SRM1) depend on the unit set for the cycle monitor time in DM 6618, as shown below.

- Bits 08 to 15 set to 01: 0.1 ms
- Bits 08 to 15 set to 02: 1 ms
- Bits 08 to 15 set to 03: 10 ms

2. If the cycle time is 1 s or longer, the cycle time read from Programming Devices will be 999.9 ms. The correct maximum and current cycle times will be recorded in the AR area.

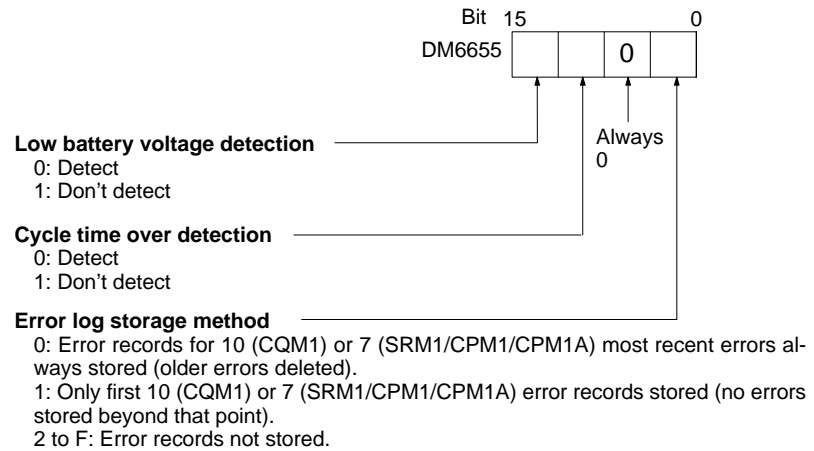
**Example**

If 0230 is set in DM 6618, an FALS 9F error will not occur until the cycle time exceeds 3 s. If the actual cycle time is 2.59 s, the current cycle time stored in the AR area will be 2590 (ms), but the cycle time read from a Programming Device will be 999.9 ms.

A "cycle time over" error (non-fatal) will be generated when the cycle time exceeds 100 ms unless detection of long cycle times is disable using the setting in DM 6655.

**Error Detection and Error Log Operation (DM 6655)**

Make the settings shown below to determine whether or not a non-fatal error is to be generated when the cycle time exceeds 100 ms or when the voltage of the built-in battery drops (CQM1 only), and to set the method for storing records in the error log when errors occur.



**Default:** Low battery voltage and cycle time over errors detected, and error records stored for the 10 most recent errors.

Battery errors and cycle time overrun errors are non-fatal errors.

For details on the error log, refer to *Section 8 Troubleshooting*.

**Note** The low battery error is applicable to CQM1 only. This digit isn't used in CPM1/CPM1A/SRM1 PCs.

## 1-3 Pulse Output Function (CQM1 Only)

This section explains the settings and methods for using the CQM1 pulse output function. Refer to the *CQM1 Operation Manual* for details on hardware connections to output points and ports.

### 1-3-1 Types of Pulse Outputs

All of the CQM1 PCs can output standard pulses from an output bit and the CQM1-CPU43-EV1 can also output standard or variable-duty-ratio pulses from ports 1 and 2. Standard pulse outputs have a duty ratio ( $t_{on}/T$ ) of 50%. The duty ratio for variable-duty-ratio pulse outputs can be set from 1% to 99% in 1% increments.

**Note** With the CQM1-CPU43-EV1, the pulse outputs described below can be output from three ports simultaneously. Furthermore, two ports can be used for counter inputs independent of the pulse output.

#### Standard Pulse Output from an Output Point

Standard pulses (duty ratio = 50%) can be output from an output point with a frequency from 20 Hz to 1 kHz. The I/O word is specified in the PC Setup and the bit is specified in the pulse output instruction itself.

Refer to page 23 for more details.

#### Standard Pulse Output from Ports 1 and 2

With the CQM1-CPU43-EV1, standard pulses (duty ratio = 50%) can be output from port 1 and/or 2 with a frequency from 10 Hz to 50 kHz (20 kHz max. to a stepping motor). The pulse output can be either clockwise (CW) or counter-clockwise (CCW) and frequency changes can be made smoothly.

PLS2(—) and mode 0 of ACC(—) cannot be used when the PC Setup (DM 6611) is set to high-speed counter mode. CTBL(63) cannot be used with ports 1 and 2 when the PC Setup (DM 6611) is set to pulse output mode.

Refer to page 25 for more details.

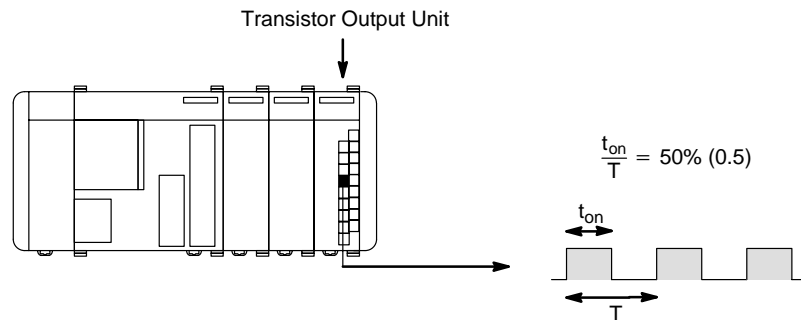
**Variable-duty-ratio Pulse Output from Ports 1 and 2**

With the CQM1-CPU43-EV1, variable-duty-ratio pulses (duty ratio = 0% to 99%) can be output from port 1 and/or 2 with frequencies of 91.6 Hz, 1.5 kHz, or 5.9 kHz. Only one direction can be output and the pulse output will continue until stopped with INI(61).

Refer to page 32 for more details.

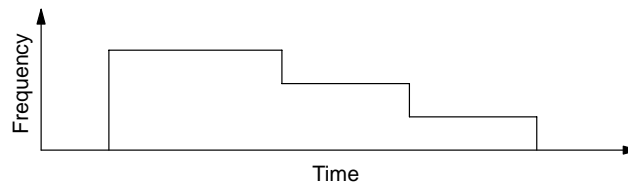
**1-3-2 Standard Pulse Output from an Output Point**

Standard pulses can be output from a specified output bit using SPED(64). Pulses can be output from just one bit at a time. The following diagram shows the pulses being output from the output point of the Transistor Output Unit mounted on a CQM1 PC. The duty ratio of the pulse output is 50% and the frequency can be set from 20 Hz to 1 kHz.



- Note**
1. A Transistor Output Unit must be used for this application.
  2. Pulses cannot be output when interval timer 0 is operating.
  3. When a pulse output higher than 500 Hz is being output, set the number of high-speed timers with interrupt processing to 4 by setting DM 6629 to 0104.

When outputting pulses from an output point, the frequency can be changed in steps by executing SPED(64) again with different frequencies, as shown in the following diagram.



There are two ways to stop the pulse output:

- 1, 2, 3... 1. After executing SPED(64), the pulse output will stop if INI(61) is executed with C=003 or SPED(64) is executed again with the frequency set to 0.
2. The total number of pulses that will be output can be set with PULS(65) before execution of SPED(64). In this case, SPED(64) must be executed in independent mode. The pulse output stops automatically when the number of pulses set by PULS(65) have been output.

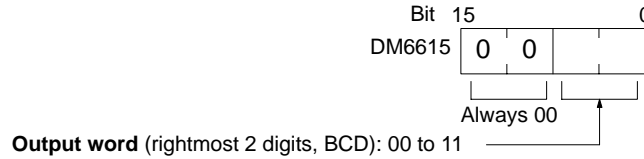
**Note** Refer to the sections on SPED(64) and PULS(65) for more details on these instructions.

**PC Setup Settings**

Before executing SPED(64) to output pulses from an Output Unit, set the PC to PROGRAM mode and make the following settings in the PC Setup.

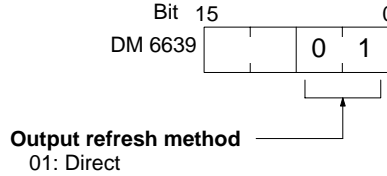
In DM 6615, specify the output word that will be used for SPED(64) pulse output to Output Units. (The bit is specified in the first operand in SPED(64).)

The content of DM 6615 (0000 to 0011) specifies output words IR 100 to IR 111. For example, if DM 6615 is set to 0002, pulses will be output to IR 102.



Default: Pulse output to IR 100.

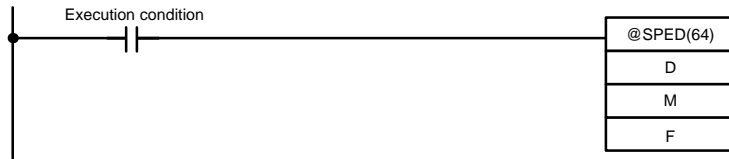
In the CQM1-CPU11/21-E CPUs, set direct output refreshing in DM 6639, as shown below. (In CQM1-CPU4□-EV1 CPU Units the output refresh method can be set to either direct or cyclic.)



Default: The default output refresh method is cyclic.

**Continuous Pulse Output**

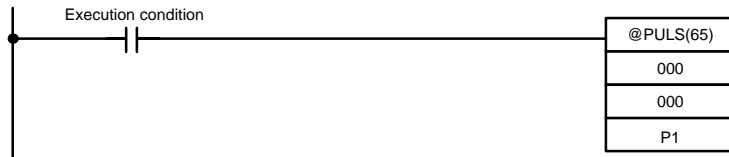
Pulses will begin to be output at the specified output bit when SPED(64) is executed. Set the output bit from 00 to 15 (D=000 to 150) and the frequency from 20 Hz to 1000 Hz (F=0002 to 0100). Set the mode to continuous mode (M=001).



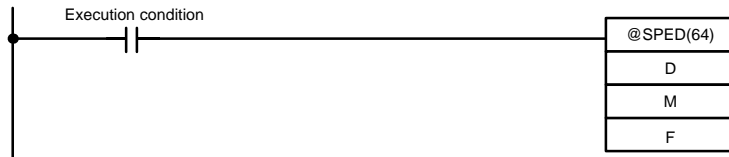
The pulse output can be stopped by executing INI(61) with C=003 or executing SPED(64) again with the frequency set to 0. The frequency can be changed by executing SPED(64) again with a different frequency setting.

**Setting the Number of Pulses**

The total number of pulses that will be output can be set with PULS(65) before executing SPED(64) in independent mode. The pulse output will stop automatically when the number of pulses set by PULS(65) have been output.



PULS(65) sets the 8-digit number of pulses P1+1, P1. These pulses can be set from 00000001 to 16777215. The number of pulses set with PULS(65) is accessed when SPED(64) is executed in independent mode. (The number of pulses cannot be changed for pulses that are being output.)



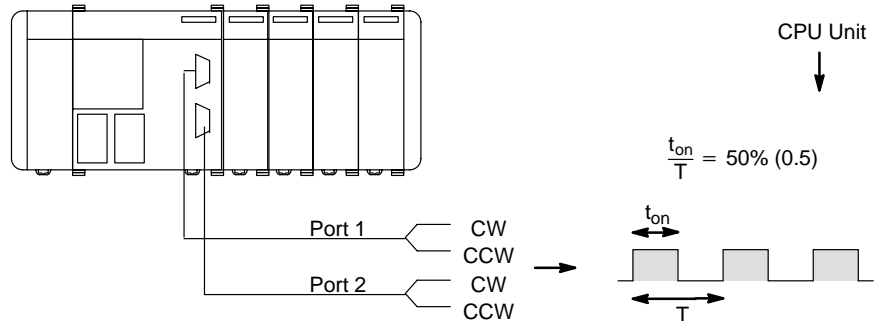
When SPED(64) is executed, pulses will begin to be output at the specified output bit (D=000 to 150: bit 00 to 15) at the specified frequency (F=0002 to 0100: 20 Hz to 1000 Hz). Set the mode to independent mode (M=001) to output the number of pulses set with PULS(65). The frequency can be changed by executing SPED(64) again with a different frequency setting.

**Changing the Frequency**

The frequency of the pulse output can be changed by executing SPED(64) again with a different frequency setting. Use the same output bit (P) and mode (M) settings that were used to start the pulse output. The new frequency can be frequency 20 Hz to 1000 Hz (F=0002 to 0100).

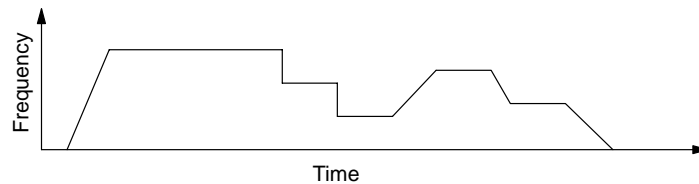
**1-3-3 Standard Pulse Output from Ports 1 and 2**

With the CQM1-CPU43-EV1, standard pulses can be output from ports 1 and 2 using SPED(64), PLS2(—), or ACC(—). The pulse frequency can be set from 10 Hz to 50 kHz (20 kHz max. to a stepping motor). The pulse output can be either clockwise (CW) or counter-clockwise (CCW) and frequency changes can be made smoothly.



**Note** Only the CQM1-CPU43-EV1 CPU Unit can output pulses from ports 1 and 2.

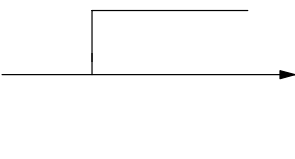
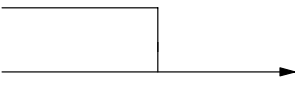
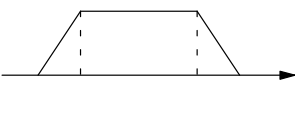
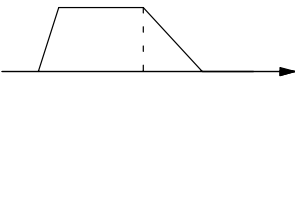
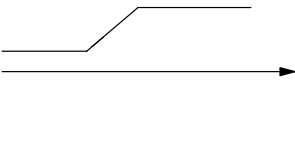
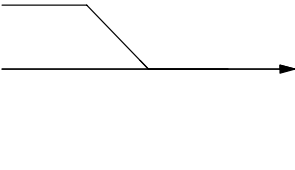
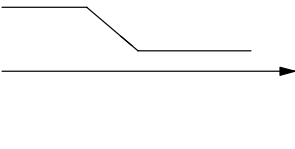
When outputting pulses from a port, the frequency can be changed smoothly or in steps with SPED(64), PLS2(—), and ACC(—), as shown in the following diagram.



There are two ways to stop the pulse output:

- 1, 2, 3... 1. After executing SPED(64), the pulse output will stop if INI(61) is executed with C=003 or SPED(64) is executed again with the frequency set to 0.
2. The total number of pulses that will be output can be set with PULS(65) before execution of SPED(64). In this case, SPED(64) must be executed in independent mode. The pulse output stops automatically when the number of pulses set by PULS(65) have been output.

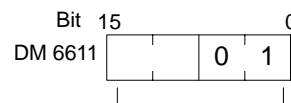
The following table shows the types of frequency changes that can be made with combinations of PULS(65), SPED(64), INI(61), PLS2(—), and ACC(—).

Frequency change	Instruction	Operand settings	Page
 <p>Start pulse output at the specified frequency. Outputs continuously or until the specified number of pulses have been output. (Execute PULS(65) and then SPED(64).)</p>	PULS(65)	CW/CCW (Number of pulses)	27
	SPED(64)	Port Mode Frequency	
 <p>Stop pulse output with an instruction. (Execute SPED(64) or INI(61).)</p>	SPED(64)	Port Frequency= 0	28
	INI(61)	Control word=0	
 <p>Outputs a specified number of pulses. Accelerates pulse output to the target frequency at the specified rate. Decelerates at the same rate.</p>	PLS2(—)	Port CW/CCW Acceleration rate Target frequency Number of pulses	29
 <p>Outputs a specified number of pulses. Accelerates pulse output to target frequency 1 at the specified rate. Decelerates to target frequency 2 at another rate. (Execute PULS(65) and then ACC(—).)</p>	PULS(65)	CW/CCW Number of pulses Deceleration point	30
	ACC(—) (Mode 0)	Port Acceleration rate Target frequency 1 Deceleration rate Target frequency 2	
 <p>Accelerates pulse output from the current frequency to the target frequency at the specified rate. Pulse output will continue. (Execute PULS(65) and then ACC(—).)</p>	PULS(65)	CW/CCW	30
	ACC(—) (Mode 1)	Port Acceleration rate Target frequency	
 <p>Decelerates pulse output from the current frequency to the target frequency at the specified rate. Pulse output will stop when the specified number of pulses have been output. (Execute PULS(65) and then ACC(—).)</p>	PULS(65)	CW/CCW Number of pulses	31
	ACC(—) (Mode 2)	Port Deceleration rate Target frequency	
 <p>Decelerates pulse output from the current frequency to the target frequency at the specified rate. Pulse output will continue. (Execute PULS(65) and then ACC(—).)</p>	PULS(65)	CW/CCW	31
	ACC(—) (Mode 3)	Port Deceleration rate Target frequency	

**PC Setup Settings**

Before outputting pulses from port 1 or 2, switch the PC to PROGRAM mode and make the following settings in the PC Setup.

In DM 6611, specify the mode setting for ports 1 and 2.



**Port 1 and 2 Mode Setting**  
 0000: High-speed counter mode  
 0001: Pulse output mode

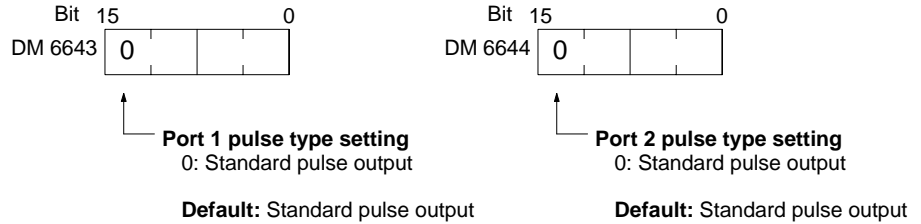
**Default:** The default mode setting is high-speed counter mode.

Some instructions cannot be used depending on the mode setting in DM 6611.

DM 6611 setting	Affected instructions
High-speed counter mode (0000)	PLS2(—) and mode 0 of ACC(—) cannot be used.
Pulse output mode (0001)	CTBL(63) cannot be used with ports 1 and 2.

The setting in DM 6611 is read only when the CQM1 is started. If this setting is changed, be sure to turn the PC off and then on again to make the new setting effective.

Specify standard pulse outputs in DM 6643 (port 1) and/or DM 6644 (port 2).

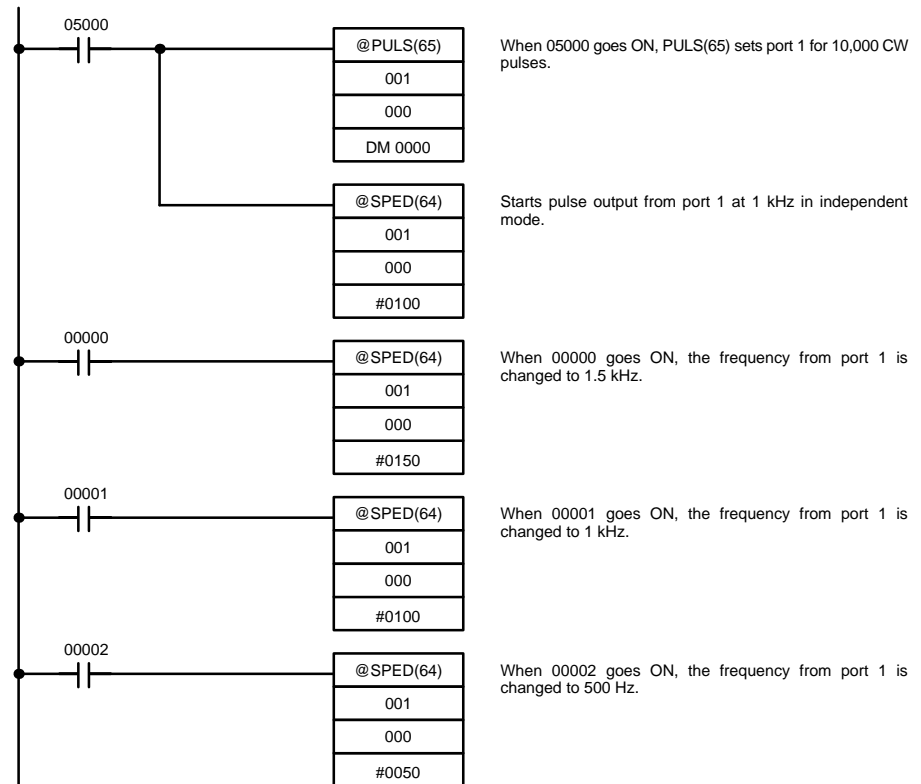


Variable-duty-ratio pulses cannot be output from a port if it has been set for standard pulse output in DM 6643 or DM 6644.

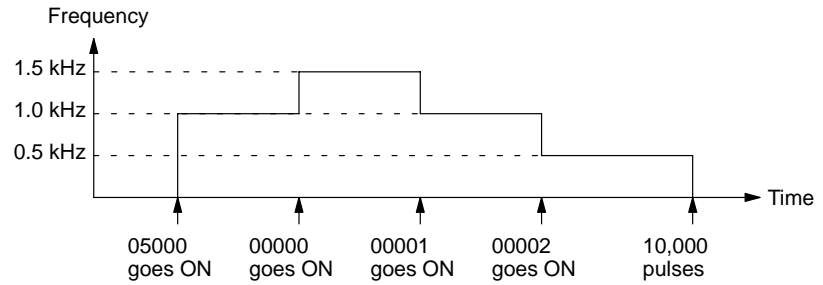
**Example 1:  
Starting Pulse Output with  
PULS(65) and SPED(64)**

The following example shows PULS(65) and SPED(64) used to control a pulse output from port 1. The number of pulses specified in PULS(65) (10,000) are output as the frequency is changed by executions of SPED(64) with different frequency settings.

Before executing the program make sure that DM 6611 is set to 0001 (pulse output mode) and DM 6643 is set to 0000 (standard pulse setting for port 1).



The following diagram shows the frequency of pulse outputs from port 1 as the program is executed.

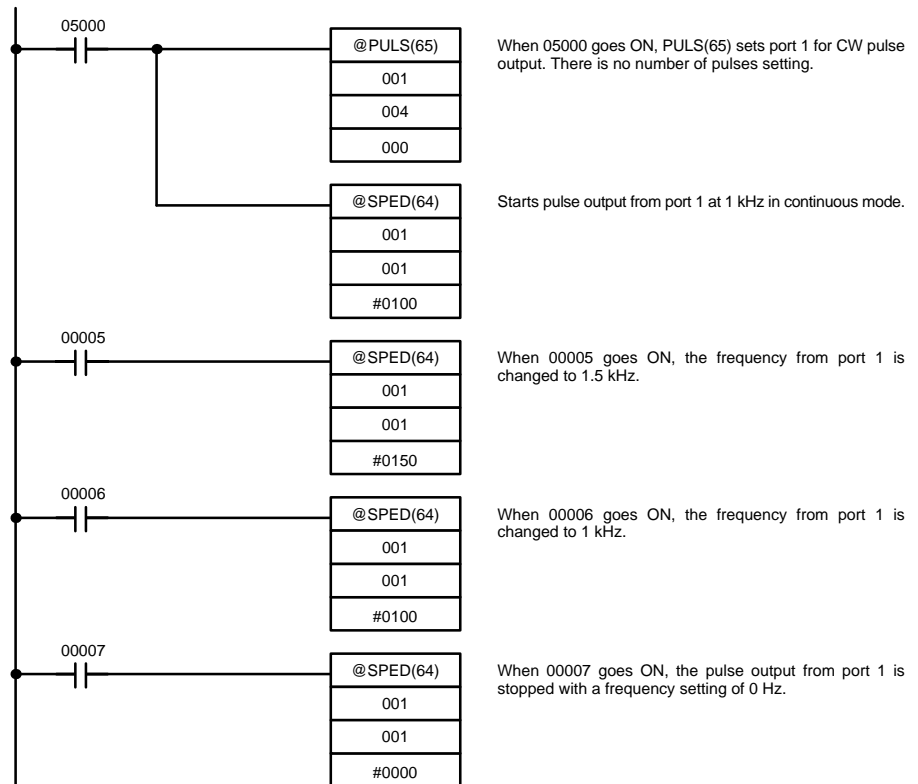


**Caution** Be sure that the pulse frequency is within the motor’s self-starting frequency range when starting and stopping the motor.

**Note** Speed control timing will be very accurate when frequency changes are performed as input interrupt processes.

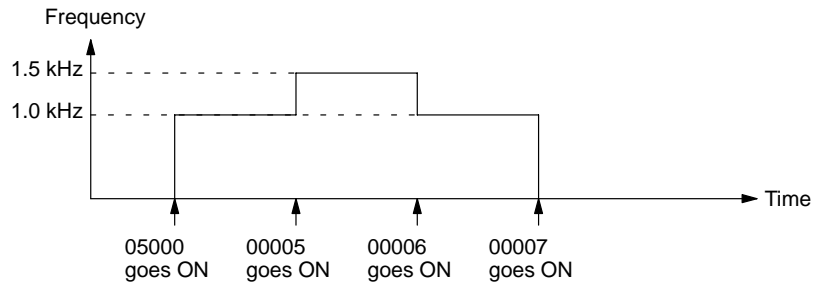
**Example 2: Stopping Pulse Output with SPED(64)**

The following example shows PULS(65) and SPED(64) used to control a pulse output from port 1. The frequency is changed by executions of SPED(64) with different frequency settings and finally stopped with a frequency setting of 0.





The following diagram shows the frequency of pulse outputs from port 1 as the program is executed.



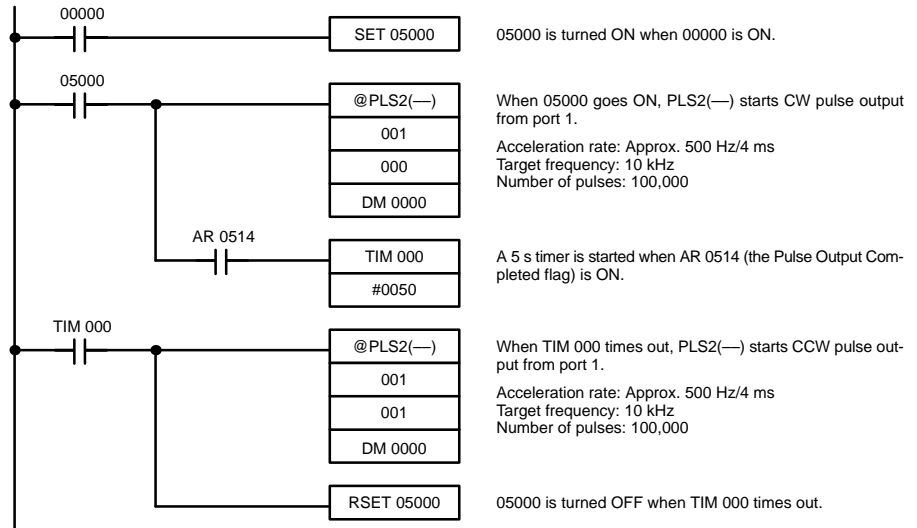
**Caution** Be sure that the pulse frequency is within the motor’s self-starting frequency range when starting and stopping the motor.

**Example 3: PLS2(—)**

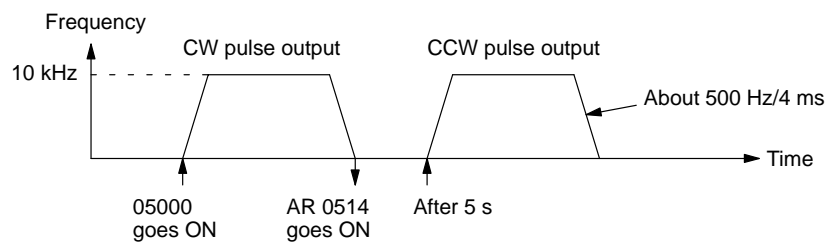
The following example shows PLS2(—) used to output 100,000 CW pulses from port 1. The frequency is accelerated to 10 kHz at approximately 500 Hz/4 ms and decelerated at the same rate.

Five seconds after the CW pulses have been output, another PLS2(—) instruction outputs 100,000 CCW pulses with the same settings.

DM 0000	0050
DM 0001	1000
DM 0002	0000
DM 0003	0010



The following diagram shows the frequency of pulse outputs from port 1 as the program is executed.

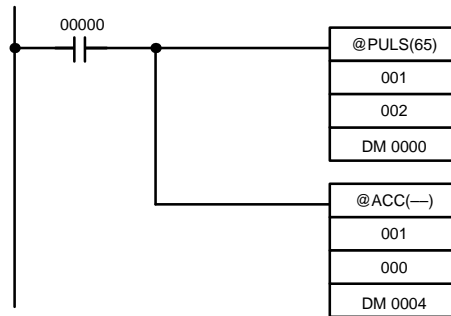


**Example 4: ACC(—) Mode 0**

The following example shows mode 0 of ACC(—) used to output 10,000 CW pulses from port 1. The frequency is accelerated to 10 kHz at approximately 1 kHz/4 ms and decelerated to 1 kHz at approximately 250 Hz/4 ms. Deceleration begins after 9,100 pulses have been output.

DM 0000	0000
DM 0001	0001
DM 0002	9100
DM 0003	0000

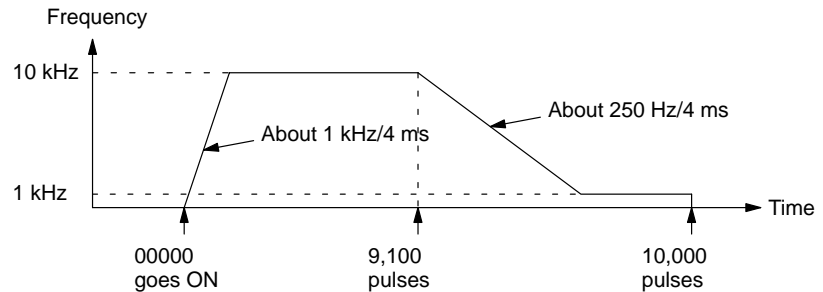
DM 0004	0100
DM 0005	1000
DM 0006	0025
DM 0007	0050



When 00000 goes ON, PULS(65) sets port 1 for CW pulse output. The total number of pulses is set to 10,000 and the deceleration point is set to 9,100 pulses.

Starts CW pulse output from port 1.  
 Acceleration rate: Approx. 1000 Hz/4 ms  
 Frequency after acceleration: 10 kHz  
 Deceleration rate: Approx. 250 Hz/4 ms  
 Frequency after deceleration: 1 kHz

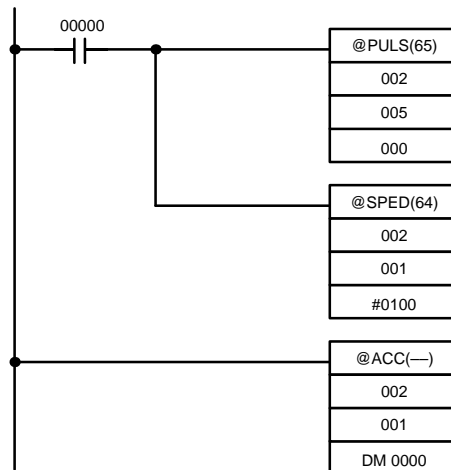
The following diagram shows the frequency of pulse outputs from port 1 as the program is executed.



**Example 5: ACC(—) Mode 1**

The following example shows mode 1 of ACC(—) used to increase the frequency of a pulse output from port 1. The frequency is accelerated from 1 kHz to 20 kHz at approximately 500 Hz/4 ms.

DM 0000	0050
DM 0001	2000

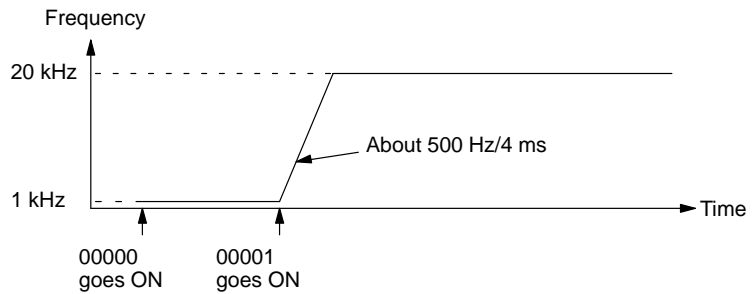


When 00000 goes ON, PULS(65) sets port 2 for CCW pulse output. The number of pulses is not set.

Starts 1 kHz pulse output from port 2 in continuous mode.

When 00001 goes ON, ACC(—) begins accelerating the port 2 pulse output at about 500 Hz/4 ms until it reaches the target frequency of 20 kHz.

The following diagram shows the frequency of pulse outputs from port 2 as the program is executed.



**Example 6: ACC(—) Mode 2**

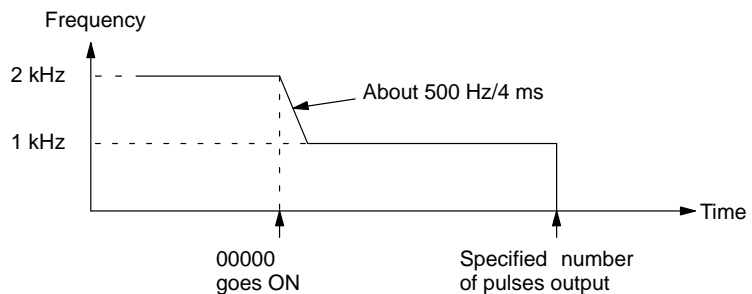
The following example shows mode 2 of ACC(—) used decrease the frequency of a pulse output from port 1. The 2-kHz pulse output is already in progress in independent mode and stops automatically when the number of pulses is reached.

DM 0000	0050
DM 0001	0001



When 00000 goes ON, ACC(—) begins decelerating the port 1 pulse output at about 500 Hz/4 ms until it reaches the target frequency of 10 Hz.

The following diagram shows the frequency of pulse outputs from port 1 as the program is executed.



**Note** The pulse output can be stopped by executing ACC(—) mode 2 with a target frequency of 0, but the pulse output cannot be stopped at the correct number of pulses, so this method should not be used except for emergency stops.

**Example 7: ACC(—) Mode 3**

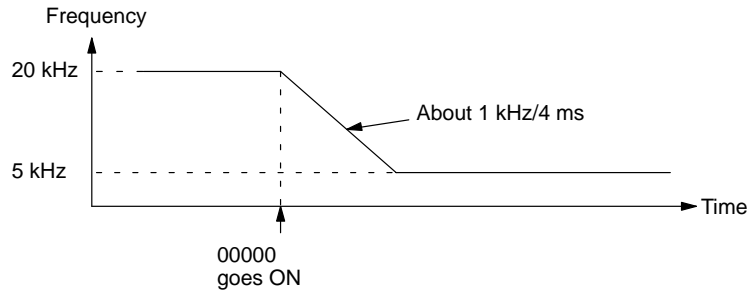
The following example shows mode 3 of ACC(—) used decrease the frequency of a pulse output from port 1. The 20-kHz pulse output is already in progress in continuous mode.

DM 0000	0100
DM 0001	0500



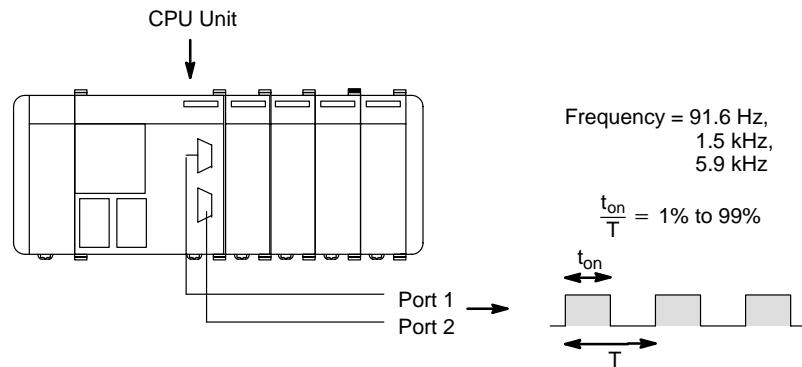
When 00000 goes ON, ACC(—) begins decelerating the port 1 pulse output at about 1 kHz/4 ms until it reaches the target frequency of 5 kHz.

The following diagram shows the frequency of pulse outputs from port 1 as the program is executed.



### 1-3-4 Variable-duty-ratio Pulse Output from Ports 1 and 2

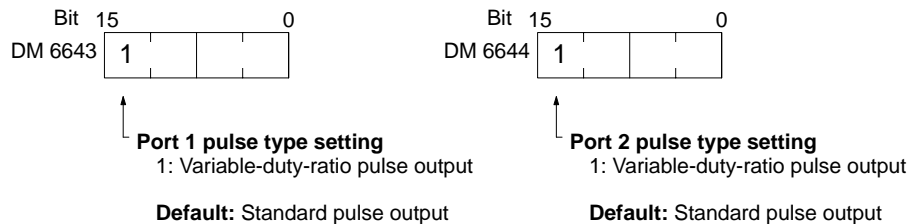
With the CQM1-CPU43-EV1, variable-duty-ratio pulses can be output from ports 1 and/or 2 using PWM(—). The pulse frequency can be set to 91.6 Hz, 1.5 kHz, or 5.9 kHz. This function can be used for various kinds of control outputs, such as light intensity output or speed control output to an inverter.



**Note** Only the CQM1-CPU43-EV1 CPU Unit can output pulses from ports 1 and 2.

#### PC Setup Settings

Before outputting variable-duty-ratio pulses from port 1 or 2, switch the PC to PROGRAM mode and make the following settings in the PC Setup. Specify variable-duty-ratio pulse outputs in DM 6643 (port 1) and/or DM 6644 (port 2).



Standard pulses cannot be output from a port if it has been set for standard variable-duty-ratio pulse output in DM 6643 or DM 6644.

#### Starting the Pulse Output

Pulses will begin to be output from the specified port when PWM(—) is executed. Specify port 1 or 2 (P=001 to 002). Set the frequency to 5.9 kHz, 1.5 kHz, or 91.6 Hz (F=000, 001, or 002). Set the duty ratio from 1% to 99% (D=0001 to 0099, BCD).



The pulse output will continue with the specified frequency and duty ratio until PWM(—) is executed again with different settings or INI(61) is executed to stop pulse outputs from the specified port.

**Stopping the Pulse Output**

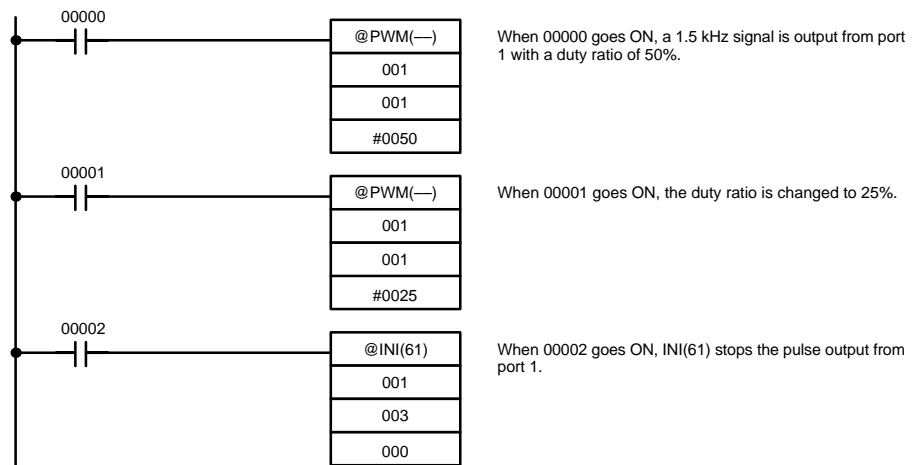
The pulse output from a port can be stopped by executing INI(61) with C=003. Specify port 1 or 2 (P=001 to 002).



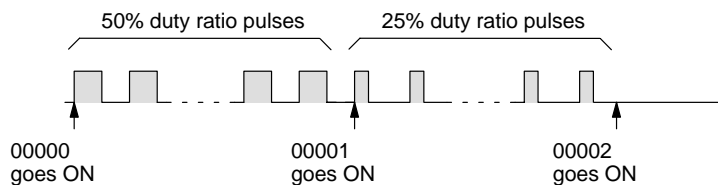
**Example: Using PWM(—)**

The following example shows PWM(—) used to start a 1.5 kHz pulse output from port 1 and change the duty ratio from 50% to 25%. The pulse output is then stopped with INI(61).

Before executing the program make sure that DM 6643 is set to 1000 (variable-duty-ratio pulse setting for port 1).



The following diagram shows the duty ratio of the pulse output from port 1 as the program is executed.



### 1-3-5 Determining the Status of Ports 1 and 2

The status of pulse outputs (for standard or variable-duty-ratio pulses) of ports 1 and 2 can be determined either by reading the status of the relevant flags in the SR and AR areas or by executing PRV(62).

#### Reading Flag Status

The status of pulse outputs can be determined by reading the contents of the words and flags shown in the following table.

Word(s)	Bit(s)	Function	Description
SR 236 and SR 237	00 to 15	Port 1 PV	Indicates the 8-digit present value of the number of pulses output from port 1. SR 237 contains the higher four digits.
SR 238 and SR 239	00 to 15	Port 2 PV	Indicates the 8-digit present value of the number of pulses output from port 2. SR 239 contains the higher four digits.
AR 04	08 to 15	Pulse output status	Indicates pulse output status. 00: normal 01 or 02: Hardware error 03: PC Setup error 04: Operation stopped during pulse output
AR 05	12	Port 1 Deceleration flag	Indicates deceleration. (0: Not specified; 1: Specified.)
	13	Port 1 Number of Pulses flag	Indicates whether the number of pulses are set. (0: Not specified; 1: Specified.)
	14	Port 1 Pulse Output Completed flag	Indicates whether pulse output has been completed. (0: Not completed; 1: Completed.)
	15	Port 1 Pulse Output Status flag	Indicates whether pulses are being output. (0: No output; 1: Output in progress.)
AR 06	12	Port 2 Deceleration flag	Indicates deceleration. (0: Not specified; 1: Specified.)
	13	Port 2 Number of Pulses flag	Indicates whether the number of pulses are set. (0: Not specified; 1: Specified.)
	14	Port 2 Pulse Output Completed flag	Indicates whether pulse output has been completed. (0: Not completed; 1: Completed.)
	15	Port 2 Pulse Output Status flag	Indicates whether pulses are being output. (0: No output; 1: Output in progress.)

#### Executing PRV(62)

The status of pulse outputs can be determined by executing PRV(62). Specify port 1 or 2 (P=001 to 002) and the destination word D. The port status information will be written to bits 04 to 07 of D and bits 00 to 03 and 08 to 15 will be set to 0.

When PRV(62) is used to read the port's status, the most recent information will be read, so the PC's cycle time will not be a factor.



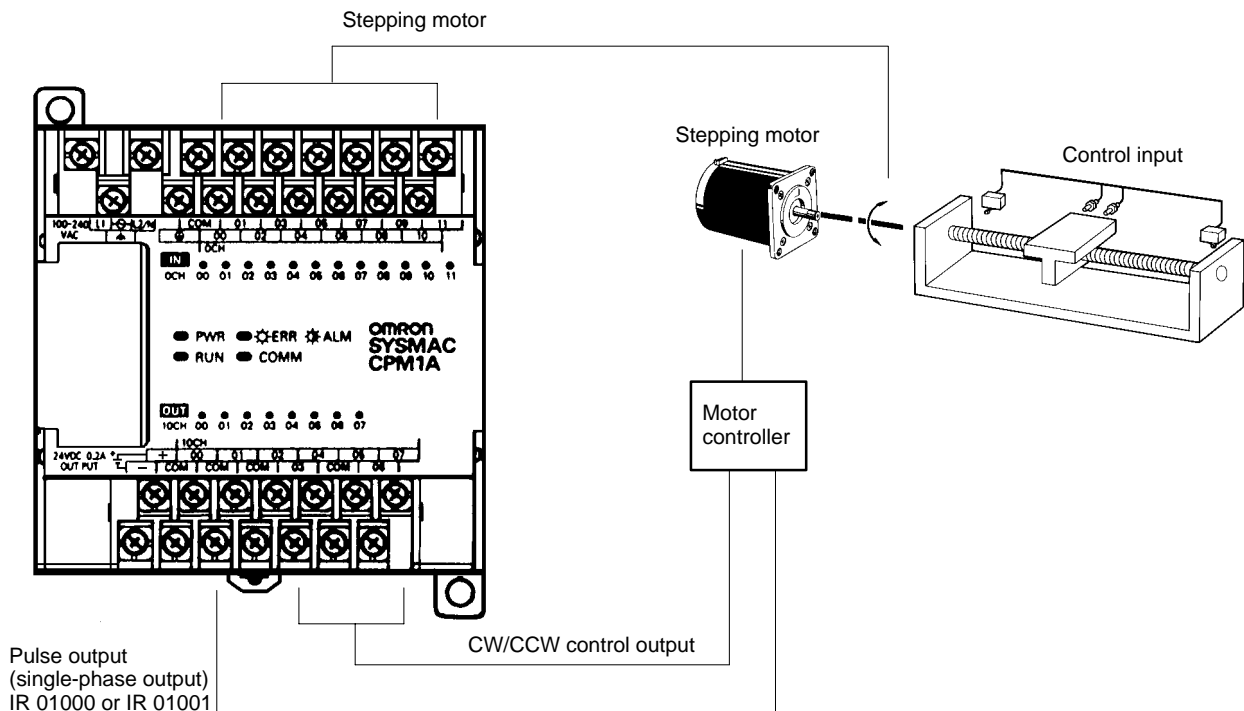
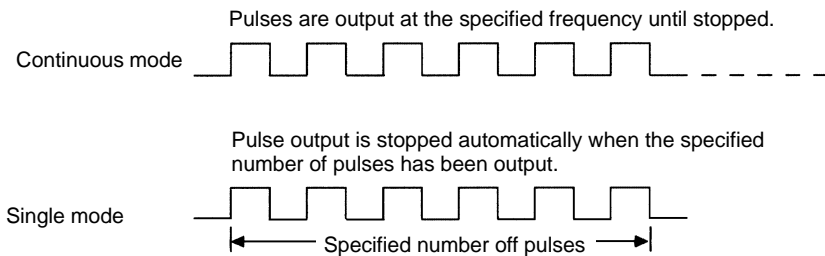
Bits 04 through 07 of D contain the specified port's status information.

Bit	Function	Description
04	Deceleration flag	Indicates deceleration. (0: Not decelerating; 1: Decelerating)
05	Number of Pulses flag	Indicates whether the total number of pulses have been specified. (0: Not specified; 1: Specified.)
06	Pulse Output Completed flag	Indicates whether pulse output has been completed. (0: Not completed; 1: Completed.)
07	Pulse Output Status flag	Indicates whether pulses are being output. (0: No output; 1: Output in progress.)

# 1-4 Pulse Output Function (CPM1A Only)

The CPM1A PCs with transistor outputs have a pulse output function capable of outputting a pulse of 20 Hz to 2 kHz (single-phase). Either IR 01000 or IR 01001 can be selected for pulse output, and the pulse output can be set to either the continuous mode, under which the output can be stopped by an instruction, or the independent mode, under which the output is stopped after a preset number of pulses (1 to 16,777,215).

Refer to the *CPM1A Operation Manual* for details on hardware connections to output points and ports.

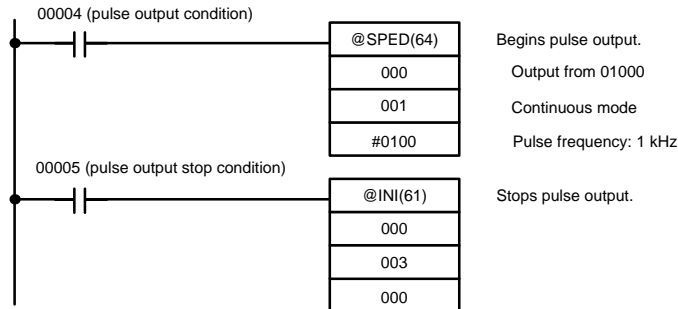


- Note**
1. The CPM1A uses a single-phase pulse output. The control signal for the direction of rotation (CW/CCW) for the motor driver must be written in the program.
  2. Be sure to use a CPU Unit with transistor outputs.

### 1-4-1 Programming Example in Continuous Mode

In this example program, pulse output begins from IR 01000 when input IR 00004 turns ON, and is stopped when input IR 00005 turns ON.

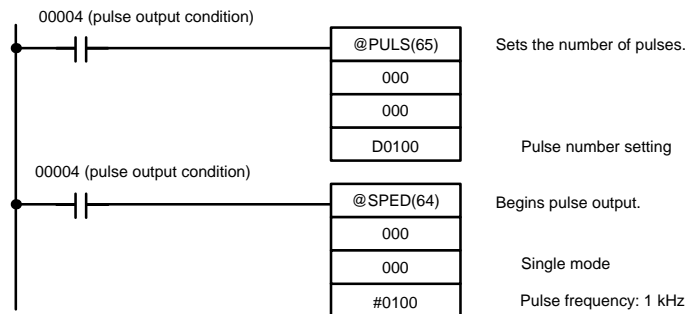
SPED(64) can be used to stop pulse output. When using SPED(64) for that purpose, specify #0000 (constant or word contents) as the pulse frequency.



•

### 1-4-2 Programming Example in Independent Mode

In this example program, pulse output begins from IR 01000 when input IR 00004 turns ON, and is stopped after the specified number of pulses have been output. The pulse amount is set in DM 0100 and DM 0101.

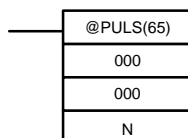


•

### 1-4-3 Using Pulse Output Instructions

#### Setting the Number of Pulses

Before beginning pulse output using the independent mode use PULS(65) as shown below to set the number of pulses to be output. This setting is not required for the continuous mode.



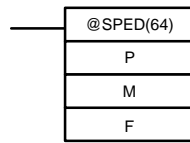
In N, set the beginning word address of the words where the number of pulses is set. Store the number of pulses in words N and N+1, in eight digits BCD, with the leftmost four digits in N+1 and the rightmost four digits in N.

Make the setting within a range of 00000001 to 16777215 (BCD).



**Beginning Pulse Output**

With SPED(64), set the bit location for pulse outputs (IR 01000 or IR 01001), the output mode (independent, continuous), and the pulse frequency to begin the pulse output.



**P (3 digits BCD)** 000: Outputs to IR 01000  
010: Outputs to IR 01001

**M (3 digits BCD)** 000: Independent mode  
001 Continuous mode

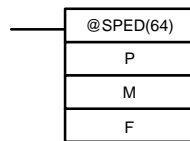
**F (4 digits BCD)** For the beginning pulse output frequency, specify a constant or word contents. The specified value and set frequency are as follows:

Specified value: 0002 to 0200  
Set frequency: 20 to 2,000 Hz

- Note**
1. Pulses can be output from only one bit at a time.
  2. When pulse output is begun in independent mode, the number of pulses is read when SPED(64) is executed. PULS(65) cannot be used to change the number of pulses while pulses are being output.

**1-4-4 Changing the Frequency**

To change the frequency during pulse output, change the frequency setting with SPED(64). At that time, set the operands other than the frequency to the same settings as at the beginning of pulse output.



**P (3 digits BCD)** Same as at beginning of pulse output.

**M (3 digits BCD)** Same as at beginning of pulse output.

**F (4 digits BCD)** For the changed pulse output frequency, specify a constant or word contents. The specified value and set frequency are as follows:

Specified value: 0002 to 0200  
Set frequency: 20 to 2,000 Hz

**1-4-5 Stopping Pulse Output**

When pulses are output in the independent mode, the pulse output will automatically stop after the number of pulses specified with PULS(65) has been output. When pulses are output in the continuous mode, either of the following two methods can be used to stop the pulse output.

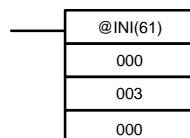
1. Use SPED(64) to set the frequency to 0.
2. Use INI(61) to stop the pulse output.

**Using SPED(64)**

The first method is to use SPED(64) to stop the pulse output by setting the frequency to 0. For details, refer to *1-4-4 Changing the Frequency*.

**Using INI(61)**

The second method is to use INI(61) to stop the pulse output, as follows:



## 1-5 CQM1 Interrupt Functions

This section explains the settings and methods for using the CQM1 interrupt functions.

### 1-5-1 Types of Interrupts

The CQM1 has three types of interrupt processing, as outlined below.

#### Input interrupts:

Interrupt processing is executed when an input from an external source turns ON one of CPU bits 00000 to 00003.

#### Interval timer interrupts:

Interrupt processing is executed by an interval timer with a precision of 0.1 ms.

#### High-speed counter interrupts:

Interrupt processing is executed according to the present value (PV) of a built-in high-speed counter. All CQM1 CPU Units are equipped with high-speed counter 0, which counts pulse inputs to one of CPU bits 00004 to 00006. Two-phase pulses up to 2.5 kHz can be counted.

The CQM1-CPU43/44-EV1 CPU Units can also count inputs to ports 1 and 2:

CQM1-CPU 43-EV1: High-speed counters 1 and 2 count high-speed pulse inputs to ports 1 and 2. Two-phase pulses up to 25 kHz can be counted.

CQM1-CPU44-EV1: High-speed counters 1 and 2 count absolute rotary encoder codes input to ports 1 and 2.

#### Interrupt Processing

When an interrupt is generated, the specified interrupt processing routine is executed. Interrupts have the following priority ranking. (Input interrupt 0 has the highest priority and high-speed counter interrupt 0 has the lowest.)

- 1, 2, 3...**
1. Input interrupt 0 > Input interrupt 1 > Input interrupt 2 > Input interrupt 3
  2. High-speed counter interrupt 1 > High-speed counter interrupt 2
  3. Interval timer interrupt 0 > Interval timer interrupt 1 > Interval timer interrupt 2 (Interval timer interrupt 2 is high-speed counter interrupt 0.)

When an interrupt with a higher priority is received during interrupt processing, the current processes will be stopped and the newly received interrupt will be processed instead. After that routine has been completely executed, then processing of the previous interrupt will be resumed.

When an interrupt with a lower or equal priority is received during interrupt processing, then the newly received interrupt will be processed as soon as the routine currently being processed has been completely executed.

Just as with ordinary subroutines, interrupt processing routines are defined using SBN(92) and RET(93) at the end of the main program.

When interrupt processing routines are executed, a specified range of input bits can be refreshed.

When an interrupt processing routine is defined, a "no SBS error" will be generated during the program check but execution will proceed normally. If this error occurs, check all normal subroutines to be sure that SBS(91) has been programmed before proceeding.

#### Pulse Output Instructions and Interrupts

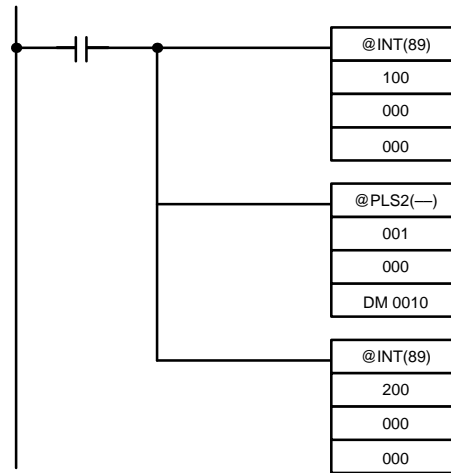
With the CQM1-CPU43/44-EV1 CPU Units, the following instructions cannot be executed in an interrupt subroutine when an instruction that controls pulse I/O or high-speed counters is being executed in the main program: (25503 turns ON)

INI(61), PRV(62), CTBL(63), SPED(64), PULS(65), PWM(—), PLS2(—) and ACC(—)

The following methods can be used to circumvent this limitation:

**Method 1**

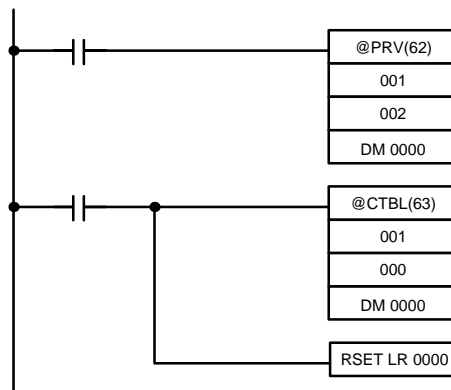
All interrupt processing can be masked while the instruction is being executed.



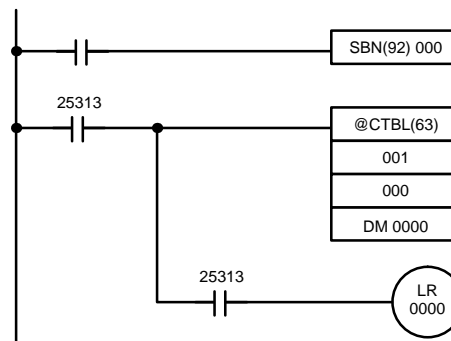
**Method 2**

Execute the instruction again in the main program.

This is the program section from the main program:



This is the program section from the interrupt subroutine:



**1-5-2 Input Interrupts**

The CPU Unit's inputs allocated IR 00000 to IR 00003 can be used for interrupts from external sources. Input interrupts 0 through 3 correspond respectively to these bits and are always used to call the subroutines numbered 000 through 003 respectively. When input interrupts are not used, subroutine numbers 000 to 003 can be used for ordinary subroutines.

**Processing**

There are two modes for processing input interrupts. The first is the Input Interrupt Mode, in which the interrupt is carried out in response to an external input. The second is the Counter Mode, in which signals from an external source are counted at high speed, and an interrupt is carried out once for every certain number of signals.

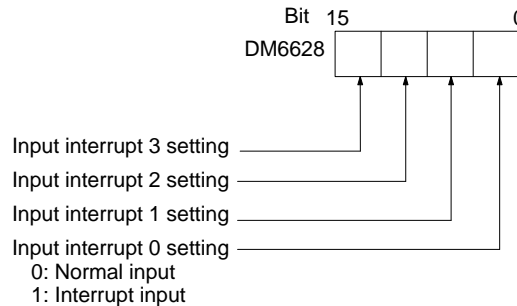
In the Input Interrupt Mode, signals with a length of 100  $\mu$ s or more can be detected. In the Counter Mode, signals up to 1 kHz can be counted.

**PC Setup Parameters**

Before executing the program, make the following settings in the PC Setup in PROGRAM mode.

**Interrupt Input Settings (DM 6628)**

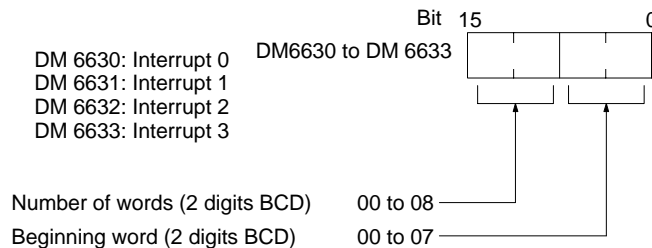
If these settings are not made, interrupts cannot be used in the program.



**Default:** All normal inputs.

**Input Refresh Word Settings (DM 6630 to DM 6633)**

Make these settings when it is necessary to refresh inputs.



**Default:** No input refresh

**Example:** If DM 6630 is set to 0100, IR 000 will be refreshed when a signal is received for interrupt 0.

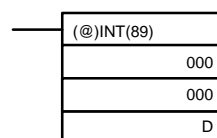
**Note** If input refreshing is not used, input signal status within the interrupt routine will not be reliable. This includes even the status of the interrupt input bit that activated the interrupt. For example, IR 00000 would not be ON in interrupt routine for input interrupt 0 unless it was refreshed (in this case, the Always ON Flag, SR 25313 could be used in place of IR 00000).

**Input Interrupt Mode**

Use the following instructions to program input interrupts using the Input Interrupt Mode.

**Masking of Interrupts**

With the INT(89) instruction, set or clear input interrupt masks as required.



Make the settings with the D bits 0 to 3, which correspond to input interrupts 0 to 3.  
 0: Mask cleared. (Input interrupt permitted.)  
 1: Mask set. (Input interrupt not permitted.)

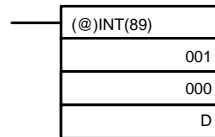
At the beginning of operation, all of the input interrupts are masked.

**Clearing Masked Interrupts**

If the bit corresponding to an input interrupt turns ON while masked, that input interrupt will be saved in memory and will be executed as soon as the mask is cleared. In order for that input interrupt not to be executed when the mask is cleared, the interrupt must be cleared from memory.

Only one interrupt signal will be saved in memory for each interrupt number.

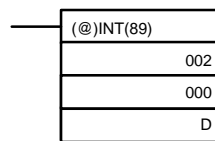
With the INT(89) instruction, clear the input interrupt from memory.



If D bits 0 to 3, which correspond to input interrupts 0 to 3, are set to "1," then the input interrupts will be cleared from memory.  
 0: Input interrupt retained.  
 1: Input interrupt cleared.

**Reading Mask Status**

With the INT(89) instruction, read the input interrupt mask status.



The status of the rightmost digit of the data stored in word D (bits 0 to 3) show the mask status.  
 0: Mask cleared. (Input interrupt permitted.)  
 1: Mask set. (Input interrupt not permitted.)

**Counter Mode**

Use the following steps to program input interrupts using the Input Interrupt Mode.

**Note** The SR words used in the Counter Mode (SR 244 to SR 251) all contain binary (hexadecimal) data (not BCD).

- 1, 2, 3...**
1. Write the set values for counter operation to SR words correspond to interrupts 0 to 3. The set values are written between 0000 and FFFF (0 to 65,535). A value of 0000 will disable the count operation until a new value is set and step 2, below, is repeated.

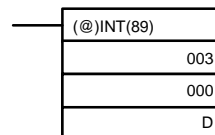
**Note** These SR bits are cleared at the beginning of operation, and must be written from the program.

That maximum input signal that can be counted is 1 kHz.

Interrupt	Word
Input interrupt 0	SR 244
Input interrupt 1	SR 245
Input interrupt 2	SR 246
Input interrupt 3	SR 247

If the Counter Mode is not used, these SR bits can be used as work bits.

2. With the INT(89) instruction, refresh the Counter Mode set value and enable interrupts.



If D bits 0 to 3, which correspond to input interrupts 0 to 3, are set to "0," then the set value will be refreshed and interrupts will be permitted.  
 0: Counter mode set value refreshed and mask cleared.  
 1: Nothing happens. (Set to 1 the bits for all interrupts that are not being changed.)

The input interrupt for which the set value is refreshed will be enabled in Counter Mode. When the counter reaches the set value, an interrupt will occur, the counter will be reset, and counting/interrupts will continue until the counter is stopped.

- Note**
1. If the INT(89) instruction is used during counting, the present value (PV) will return to the set value (SV). You must, therefore, use the differentiated form of the instruction or an interrupt may never occur.

2. The set value will be set when the INT(89) instruction is executed. If interrupts are already in operation, then the set value will not be changed just by changing the content of SR 244 to SR 247, i.e., if the contents is changed, the set value must be refreshed by executing the INT(89) instruction again.

Interrupts can be masked using the same process as for the Input Interrupt Mode, but is masked are cleared using the same process, the Counter Mode will not be maintained and the Input Interrupt Mode will be used instead. Interrupt signals received for masked interrupts can also be cleared using the same process as for the Input Interrupt Mode.

#### Counter PV in Counter Mode

When input interrupts are used in Counter Mode, the counter PV will be stored in the SR word corresponding to input interrupts 0 to 3. Values are 0000 to FFFE (0 to 65,534) and will equal the counter PV minus one.

Interrupt	Word
Input interrupt 0	SR 248
Input interrupt 1	SR 249
Input interrupt 2	SR 250
Input interrupt 3	SR 251

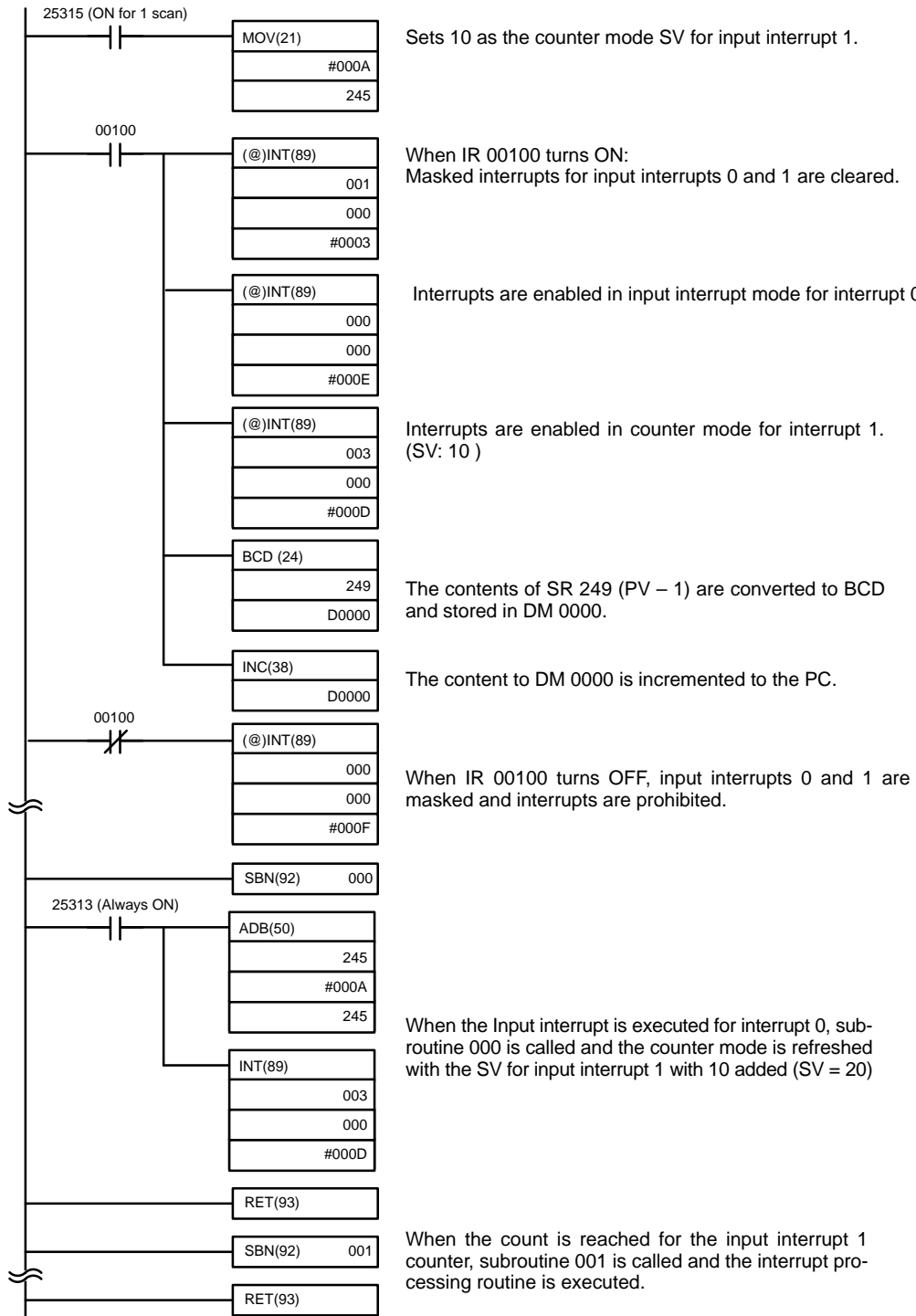
**Example:** The present value for an interrupt whose set value is 000A will be recorded as 0009 immediately after INT(89) is executed.

**Note** Even if input interrupts are not used in Counter Mode, these SR bits cannot be used as work bits.

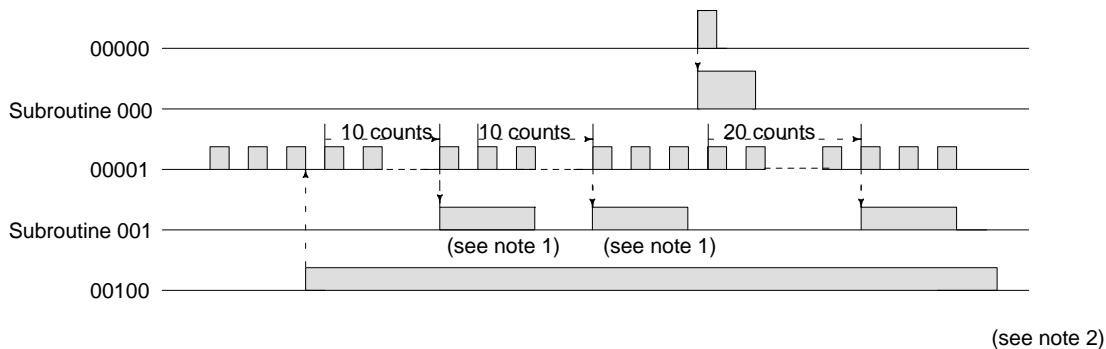
**Application Example**

In this example, input interrupt 0 is used in Input Interrupt Mode and input interrupt 1 is used in Counter Mode. Before executing the program, check to be sure the PC Setup.

PC Setup: DM 6628: 0011 (IR 00000 and IR 00001 used for input interrupts) The default settings are used for all other PC Setup parameters. (Inputs are not refreshed at the time of interrupt processing.)



When the program is executed, operation will be as shown in the following diagram.



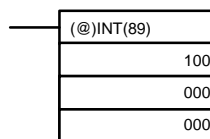
- Note**
1. The counter will continue operating even while the interrupt routine is being executed.
  2. The input interrupt will remain masked.

### 1-5-3 Masking All Interrupts

All interrupts, including input interrupts, interval timer interrupts, and high-speed counter interrupts, can be masked and unmasked as a group by means of the INT(89) instruction. The mask is in addition to any masks on the individual types of interrupts. Furthermore, clearing the masks for all interrupts does not clear the masks on the individual types of interrupts, but restores them to the masked conditions that existed before INT(89) was executed to mask them as a group. Do not use INT(89) to mask interrupts unless it is necessary to temporarily mask all interrupts and always use INT(89) instructions in pairs to do so, using the first INT(89) instruction to mask and the second one to unmask interrupts. INT(89) cannot be used to mask and unmask all interrupts from within interrupt routines.

#### Masking Interrupts

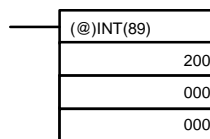
Use the INT(89) instruction to disable all interrupts.



If an interrupt is generated while interrupts are masked, interrupt processing will not be executed but the interrupt will be recorded for the input, interval timer, and high-speed counter interrupts. The interrupts will then be serviced as soon as interrupts are unmasked.

#### Unmasking Interrupts

Use the INT(89) instruction to unmask interrupts as follows:



### 1-5-4 Interval Timer Interrupts

High-speed, high-precision timer interrupt processing can be executed using interval timers. The CQM1 provides three interval timers, numbered from 0 to 2.

- Note**
1. Interval timer 0 cannot be used when pulses are being output to Output Units by means of the SPED(64) instruction.
  2. Interval timer 2 cannot be used at the same time as the high-speed counter.



**Processing**

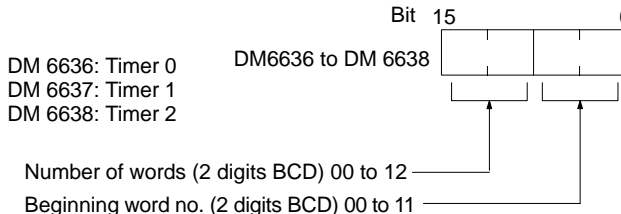
There are two modes for interval timer operation, the One-shot Mode, in which only one interrupt will be executed when time expires, and the Scheduled Interrupt Mode in which the interrupt is repeated at a fixed interval.

**PC Setup**

When using interval timer interrupts, make the following settings in the PC Setup in PROGRAM mode before executing the program.

**Input Refresh Word Settings (DM 6636 to DM 6638)**

Make these settings when it is necessary to refresh inputs.



**Default:** No input refresh

**High-speed Counter Settings (DM 6642)**

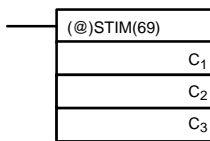
When using interval timer 2, check before beginning operation to be sure that the high-speed counter (PC Setup: DM 6642) is set to the default setting (0000: High-speed counter not used).

**Operation**

Use the following instruction to activate and control the interval timer.

**Starting Up in One-Shot Mode**

Use the STIM(69) instruction to start the interval timer in the one-shot mode.



- C<sub>1</sub>: Interval timer no.  
Interval timer 0: 000  
Interval timer 1: 001  
Interval timer 2: 002
- C<sub>2</sub>: Timer set value (first word address)
- C<sub>3</sub>: Subroutine no. (4 digits BCD): 0000 to 0255

C<sub>2</sub>: Decrementing counter set value (4 digits BCD): 0000 to 9999  
C<sub>2</sub> + 1: Decrementing time interval (4 digits BCD; unit: 0.1 ms): 0005 to 0320 (0.5 ms to 32 ms)

Each time that the interval specified in word C<sub>2</sub> + 1 elapses, the decrementing counter will decrement the present value by one. When the PV reaches 0, the designated subroutine will be called just once and the timer will stop.

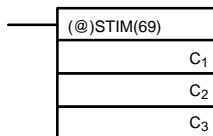
The time from when the STIM(69) instruction is executed until time elapses is calculated as follows:

$$(\text{Contents of word } C_2) \times (\text{Contents of word } C_2 + 1) \times 0.1 \text{ ms} = (0.5 \text{ to } 319,968 \text{ ms})$$

If a constant is set for C<sub>2</sub>, then the set value of the decrementing counter will take that value and the decrementing time interval will be 10 (1 ms). (The set value is expressed in ms.)

**Starting Up in Scheduled Interrupt Mode**

Use the STIM(69) instruction to start the interval timer in the scheduled interrupt mode.



- C<sub>1</sub>: Interval timer no. + 3  
Interval timer 0: 003  
Interval timer 1: 004  
Interval timer 2: 005
- C<sub>2</sub>: Timer set value (leading word no.)
- C<sub>3</sub>: Subroutine no. (4 digits BCD): 0000 to 0255

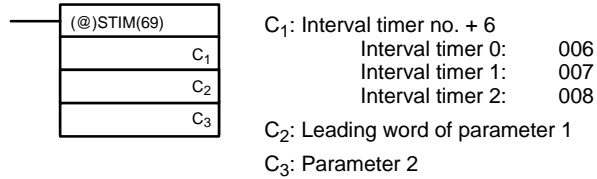
C<sub>2</sub>: Decrementing counter set value (4 digits BCD): 0000 to 9999  
C<sub>2</sub> + 1: Decrementing time interval (4 digits BCD; unit: 0.1 ms): 0005 to 0320 (0.5 ms to 32 ms)

The meanings of the settings are the same as for the one-shot mode, but in the scheduled interrupt mode the timer PV will be reset to the set value and decrementing will begin again after the subroutine has been called. In the scheduled interrupt mode, interrupts will continue to be repeated at fixed intervals until the operation is stopped.

**Note** CQM1-CPU11-E/CPU21-E support subroutine numbers 0000 to 0127 only.

**Reading the Timer’s Elapsed Time**

Use the STIM(69) instruction to read the timer’s elapsed time.



C<sub>2</sub>: Number of times the decrementing counter has ben decremented (4 digits BCD)

C<sub>2</sub> + 1: Decrementing counter time interval (4 digits BCD; unit: 0.1 ms)

C<sub>3</sub>: Elapsed time from previous decrement (4 digits BCD; unit: 0.1 ms)

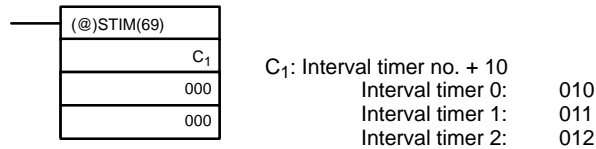
The time from when the interval timer is started until the execution of this instruction is calculated as follows:

$$\{(\text{Contents of word C2}) \times (\text{Contents of word C2} + 1) + (\text{Contents of word C3})\} \times 0.1 \text{ ms}$$

If the specified interval timer is stopped, then “0000” will be stored.

**Stopping Timers**

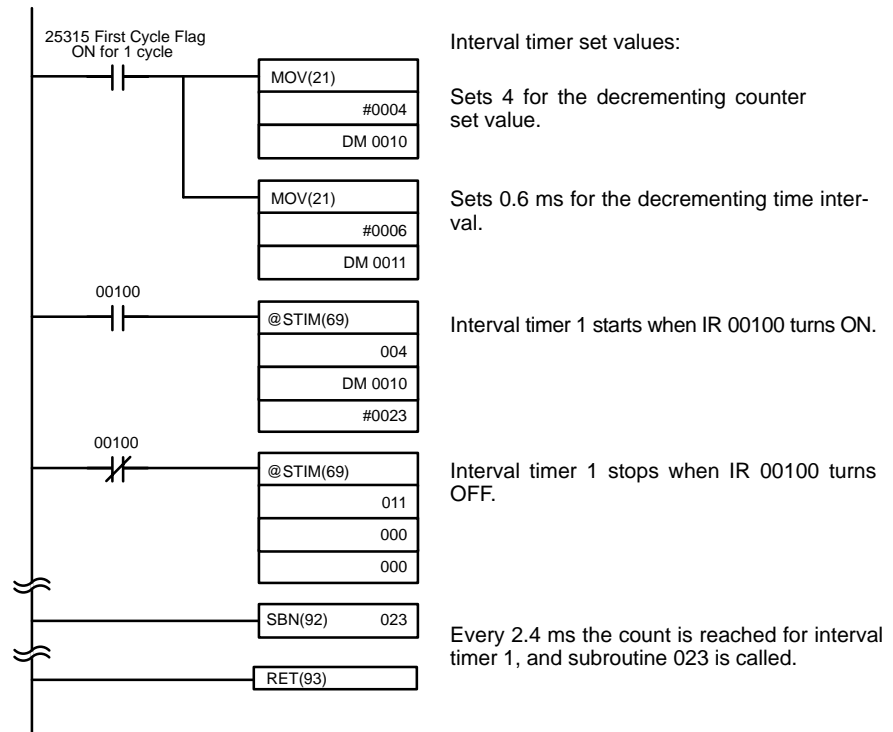
Use the STIM(69) instruction to stop the interval timer.



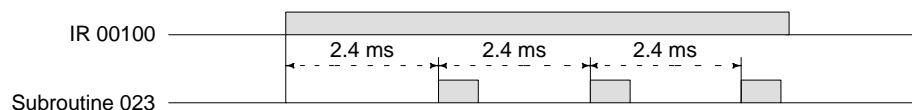
The specified interval timer will stop.

**Application Example**

In this example, an interrupt is executed every 2.4 ms (0.6 ms x 4) by means of interval timer 1. Assume the default settings for all of the PC Setup. (Inputs are not refreshed for interrupt processing.)



When the program is executed, subroutine 023 will be executed every 2.4 ms while IR 00100 is ON.



**1-5-5 High-speed Counter 0 Interrupts**

Pulse signals from a pulse encoder to CPU bits 00004 through 00006 can be counted at high speed, and interrupt processing can be executed according to the count.

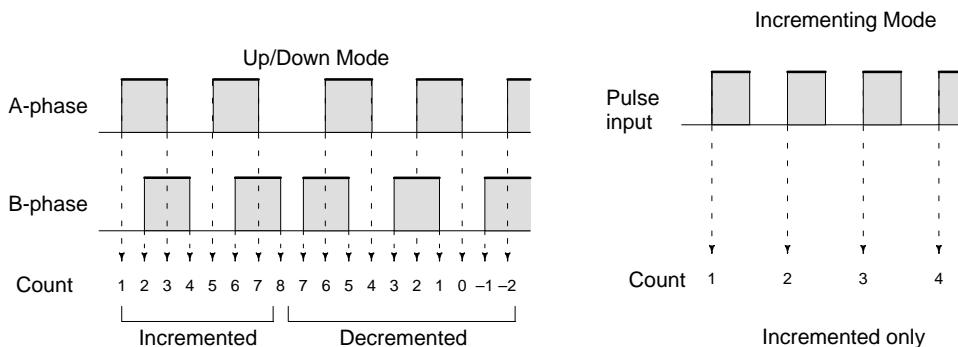
**Processing**

**Input Signal Types and Count Modes**

Two types of signals can be input from a pulse encoder. The count mode used for high-speed counter 0 will depend on the signal type.

Up/Down Mode: A phase-difference 4X two-phase signal (A-phase and B-phase) and a Z-phase signal are used for inputs. The count is incremented or decremented according to differences in the 2-phase signals.

Incrementing mode: One single-phase pulse signal and a count reset signal are used for inputs. The count is incremented according to the single-phase signal.



**Note** One of the methods in the following section should always be used to reset the counter when restarting it. The counter will be automatically reset when program execution is started or stopped.

The following signal transitions are handled as forward (incrementing) pulses: A-phase leading edge to B-phase leading edge to A-phase trailing edge to B-phase trailing edge. The following signal transitions are handled as reverse (decrementing) pulses: B-phase leading edge to A-phase leading edge to B-phase trailing edge to A-phase trailing edge.

The count range is from -32,767 to 32,767 for Up/Down Mode, and from 0 to 65,535 for Incrementing Mode. Pulse signals can be counted at up to 2.5 kHz in Up/Down Mode, and up to 5.0 kHz in Incrementing Mode.

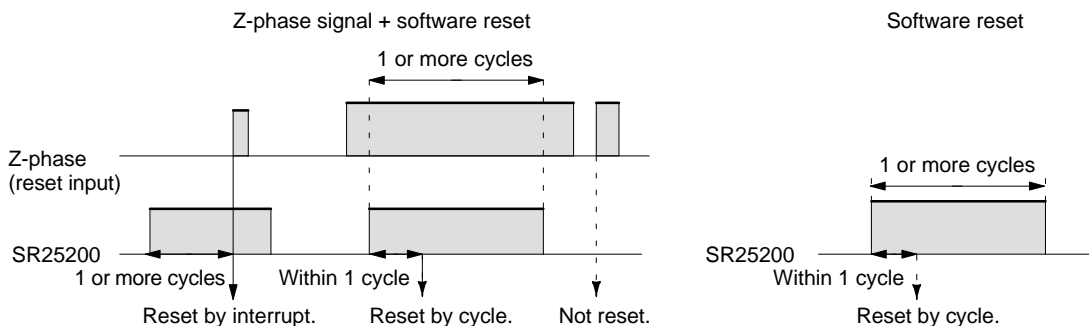
The Up/Down Mode always uses a 4X phase-difference input. The number of counts for each encoder revolution would be 4 times the resolution of the counter. Select the encoder based on the countable ranges.

**Reset Methods**

Either of the two methods described below may be selected for resetting the PV of the count (i.e., setting it to 0).

Z-phase signal + software reset: The PV is reset when the Z-phase signal (reset input) turns ON after the High-speed Counter 0 Reset Bit (SR 25200) is turned ON.

Software reset: The PV is reset when the High-speed Counter 0 Reset Bit (SR 25200) is turned ON.



**Note** The High-speed Counter 0 Reset Bit (SR 25200) is refreshed once every cycle, so in order for it to be read reliably it must be ON for at least one cycle.

The “Z” in “Z-phase” is an abbreviation for “Zero.” It is a signal that shows that the encoder has completed one cycle.

### High-speed Counter Interrupt Count

For high-speed counter 0 interrupts, a comparison table is used instead of a “count up.” The count check can be carried out by either of the two methods described below. In the comparison table, comparison conditions (for comparing to the PV) and interrupt routine combinations are saved.

Target value:

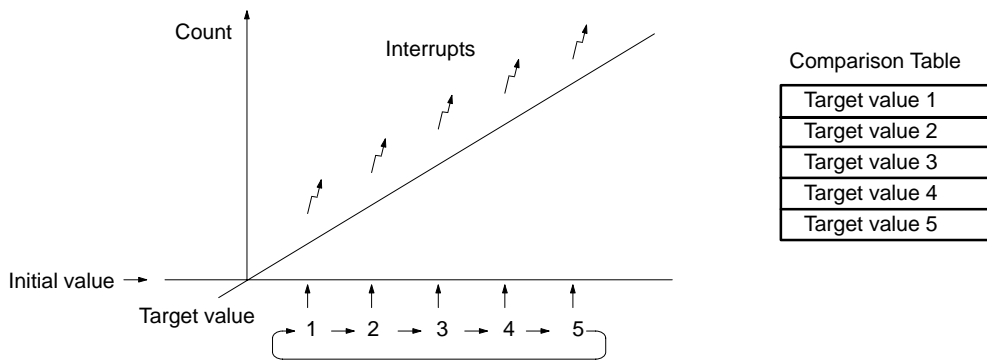
A maximum of 16 comparison conditions (target values and count directions) and interrupt routine combinations are saved in the comparison table. When the counter PV and the count direction match the comparison conditions, then the specified interrupt routine is executed.

Range comparison:

Eight comparison conditions (upper and lower limits) and interrupt routine combinations are saved in the comparison table. When the PV is greater than or equal to the lower limit and less than or equal to the upper limit, then the specified interrupt routine is executed.

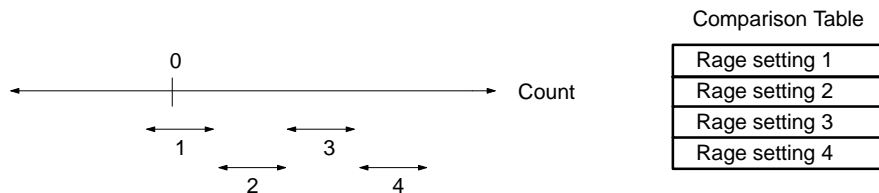
#### Target Value Comparisons

The current count is compared to the target values in the order that target values are set in the comparison table and interrupts are generated as the count equals each target value. Once the count has equaled all of the target values in the table, the target value is set to the first target value in the table, which is again compared to the current counted until the two values are equal.



#### Range Comparisons

The current count is compared in cyclic fashion to all of the ranges at the same time and interrupts are generated based on the results of the comparisons.



**Note** When performing target value comparisons, do not repeatedly use the INI instruction to change the current value of the count and start the comparison operation. The interrupt operation may not work correctly if the comparison operation is started immediately after changing the current value from the program. (The comparison operation will automatically return to the first target value once an interrupt has been generated for the last target value. Repetitious operation is thus possible merely by changing the current value.)

#### Wiring

Depending on the count mode, the input signals from the pulse encoder to the CPU Unit’s input terminal are as shown below.

Terminal no.	Up/Down Mode	Incrementing Mode
4	Encoder A-phase	Pulse count input
5	Encoder B-phase	---
6	Encoder Z-phase	Reset input

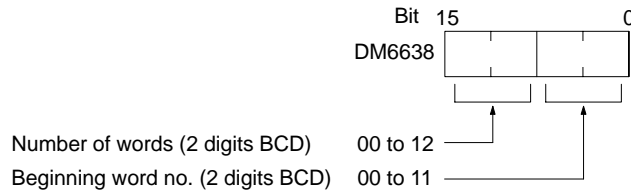
If only the software reset is to be used, terminal 6 can be used as an ordinary input. When in Incrementing Mode, terminal 5 can be used as an ordinary input.

**PC Setup**

When using high-speed counter 0 interrupts, make the settings in PROGRAM mode shown below before executing the program.

**Input Refresh Word Settings (DM 6638)**

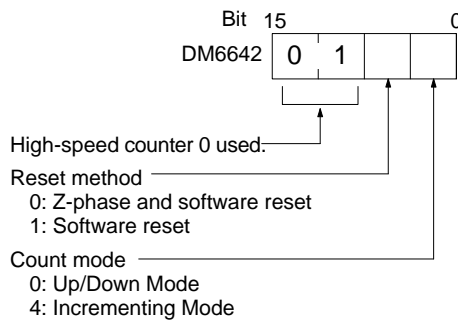
Make these settings when it is necessary to refresh inputs. The setting is the same as that for interval timer 2.



**Default:** No input refresh

**High-speed Counter 0 Settings (DM 6642)**

If these settings are not made, high-speed counter 0 cannot be used in the program.



**Default:** High-speed counter 0 not used.

Changes in the setting in DM 6642 are effective only when power is turned on or PC program execution is started.

**Programming**

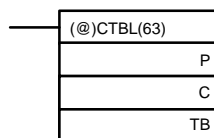
Use the following steps to program high-speed counter 0.

High-speed counter 0 begins the counting operation when the proper PC Setup settings are made, but comparisons will not be made with the comparison table and interrupts will not be generated unless the CTBL(63) instruction is executed. High-speed counter 0 is reset to "0" when power is turned ON and when operation begins.

The present value of high-speed counter 0 is maintained in SR 230 and SR 231.

**Controlling High-speed Counter 0 Interrupts**

- 1, 2, 3... 1. Use the CTBL(63) instruction to save the comparison table in the CQM1 and begin comparisons.



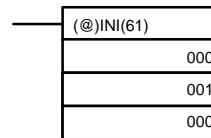
C: (3 digits BCD)  
000: Target table set and comparison begun  
001: Range table set and comparison begun  
002: Target table set only  
003: Range table set only  
TB: Beginning word of comparison table

If C is set to 000, then comparisons will be made by the target matching method; if 001, then they will be made by the range comparison method. The comparison table will be saved, and, when the save operation is complete, then comparisons will begin. While comparisons are being executed, high-speed interrupts will be executed according to the comparison table. For details on the contents of the comparison tables that are saved, refer to the explanation of the CTBL(63) instruction in *Section 5 Instruction Set*.

**Note** The comparison results are normally stored in AR 1100 through AR 1107 while the range comparison is being executed.

If C is set to 002, then comparisons will be made by the target matching method; if 003, then they will be made by the range comparison method. For either of these settings, the comparison table will be saved, but comparisons will not begin, and the INI(61) instruction must be used to begin comparisons.

2. To stop comparisons, execute the INI(61) instruction as shown below.



To start comparisons again, set the second operand to "000" (execute comparison), and execute the INI(61) instruction.

Once a table has been saved, it will be retained in the CQM1 during operation (i.e., during program execution) as long as no other table is saved.

**Reading the PV**

There are two ways to read the PV. The first is to read it from SR 230 and SR 231, and the second to use the PRV(62) instruction.

**Reading SR 230 and SR 231**

The PV of high-speed counter 0 is stored in SR 230 and SR 231 as shown below. The leftmost bit will be F for negative values.

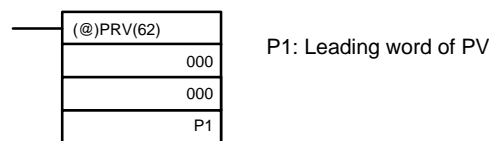
Leftmost 4 digits	Rightmost 4 digits	Up/Down Mode	Incrementing Mode
SR 231	SR 230	F0032767 to 00032767 (-32767)	00000000 to 00065535

**Note** These words are refreshed only once every cycle, so there may be a difference from the actual PV.

When high-speed counter 0 is not being used, the bits in these words can be used as work bits.

**Using the PRV(62) Instruction**

Read the PV of high-speed counter 0 by using the PRV(62) instruction.



The PV of high-speed counter 0 is stored as shown below. The leftmost bit will be F for negative values.

Leftmost 4 digits	Rightmost 4 digits	Up/Down Mode	Incrementing Mode
P1+1	P1	F0032767 to 00032767 (-32767)	00000000 to 00065535

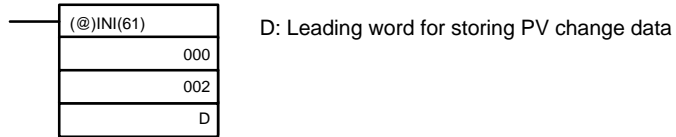
The PV is read when the PRV(62) instruction is actually executed.

**Changing the PV**

There are two ways to change the PV of high-speed counter 0. The first way is to reset it by using the reset methods. (In this case the PV is reset to 0.) The second way is to use the INI(61) instruction.

The method using the INI(61) instruction is explained here. For an explanation of the reset method, refer to the beginning of this description of high-speed counter 0.

Change the timer PV by using the INI(61) instruction as shown below.



Leftmost 4 digits

D+1

Rightmost 4 digits

D

Up/Down Mode

F0032767 to 00032767

Incrementing Mode

00000000 to 00065535

To specify a negative number, set F in the leftmost digit.

**Operation Example**

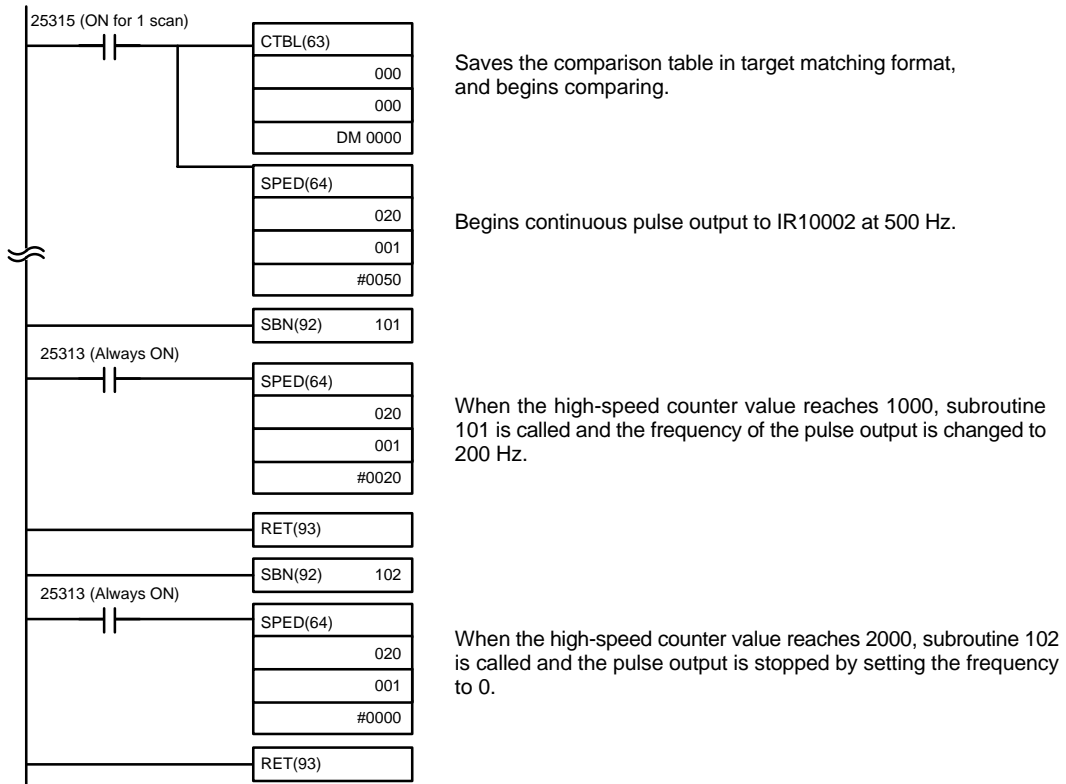
This example shows a program for using high-speed counter 0 in the Incrementing Mode, making comparisons by means of the target matching method, and changing the frequency of pulse outputs according to the counter's PV. Before executing the program, set the PC Setup as follows:

DM 6642: 0114 (High-speed counter 0 used with software reset and Incrementing Mode). For all other PC Setup, use the default settings. (Inputs are not refreshed at the time of interrupt processing, and pulse outputs are executed for IR 100.)

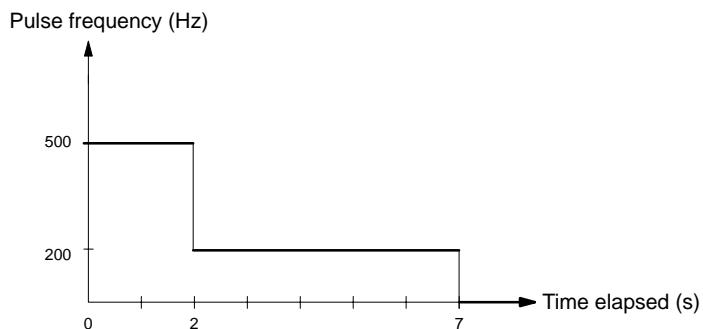


In addition, the following data is stored for the comparison table:

DM 0000	0002	Number of comparison conditions: 2
DM 0001	1000	Target value 1: 1000
DM 0002	0000	
DM 0003	0101	Comparison 1 interrupt processing routine no.: 101
DM 0004	2000	Target value 1: 2000
DM 0005	0000	
DM 0006	0102	Comparison 2 interrupt processing routine no.: 102



When the program is executed, operation will be as follows:



### 1-5-6 High-speed Counter 0 Overflows/Underflows

If the allowable counting range for high-speed counter 0 is exceeded, and underflow or overflow status will occur and the counter's PV will remain at OFFF FFFF for overflows and FFFF FFFF for underflows until the overflow/underflow status is cleared by resetting the counter. The allowable counting ranges are as follows:

Up/Down Mode: F003 2767 to 0003 2767  
 Incrementing Mode: 0000 0000 to 0006 5535

- Note**
1. The values given above are theoretical and assume a reasonably short cycle time. The values will actually be those that existed one cycle before the overflow/underflow existed.
  2. The 6th and 7th digits of high-speed counter 0's PV are normally 00, but can be used as "Overflow/Underflow Flags" by detecting values beyond the allowable counting ranges.

High-speed counter 0 can be reset as described in the previous section or it can be reset automatically by restarting program execution. High-speed counter 0 and related operations will not function normally until the overflow/underflow status is cleared. Operations during overflow/underflow status will be as follows.

- Comparison table operation will stop.
- The comparison table will not be cleared.
- Interrupt routines for the high-speed counter will not be executed.
- CTBL(63) can be used only to register the comparison table. If an attempt is made to start comparison table operation, operation will not start and the comparison table will not be registered.
- INI(61) cannot be used to start or stop comparison table operation or to change the present value.
- PRV(62) will read out only 0FFF FFFF or FFFF FFFF as the present value.

## Recovery

Use the following procedure to recover from overflow/underflow status.

### With Comparison Table Registered

- 1, 2, 3...**
1. Reset the counter.
  2. Set the PV with PRV(62) if necessary.
  3. Set the comparison table with CTBL(63) if necessary
  4. Start comparison table operation with INI(61).

### Without Comparison Table Registered

- 1, 2, 3...**
1. Reset the counter.
  2. Set the PV with PRV(62) if necessary.
  3. Set the comparison table and start operation with CTBL(63) and INI(61).

- Note** The range comparison results in AR 11 will remain after recovery. The interrupt routine for a interrupt condition meet immediately after recovery will not be executed if the interrupt condition was already met before the overflow/underflow status occurred. If interrupt routine execution is necessary, clear AR 11 before proceeding.

## Reset Operation

When high-speed counter 0 is reset, the PV will be set to 0, counting will begin from 0, and the comparison table, execution status, and execution results will be maintained.

## Startup Counter Status

When high-speed counter 0 is started, the counter mode in the PC Setup will be read and used, the PV will be set to 0, overflow/underflow status will be cleared, the comparison table registration and execution status will be cleared, and range execution results will be cleared. (Range execution results are always cleared when operation is begun or when the comparison table is registered.)

## Stopped Counter Status

When high-speed counter 0 is stopped, the PV will be maintained, the comparison table registration and execution status will be cleared, and range execution results will be maintained.

### 1-5-7 High-speed Counter 1 and 2 Interrupts (CQM1-CPU43-EV1)

Pulse signals from a pulse encoder to ports 1 and 2 of the CQM1-CPU43-EV1 can be counted at high speed, and interrupt processing can be executed according to the count.

The 2 ports can be operated separately. The counter for port 1 is called high-speed counter 1 and the counter for port 2 is called high-speed counter 2. This section describes how to use high-speed counters 1 and 2. Refer to the *CQM1 Operation Manual* for hardware information such as equipment and wiring specifications.

- Note**
- High-speed counters 1 and 2 can be used with the CQM1-CPU43-E/-EV1 only.
  - Some instructions cannot be used when the PC Setup (DM 6611) is set to high-speed counter mode.

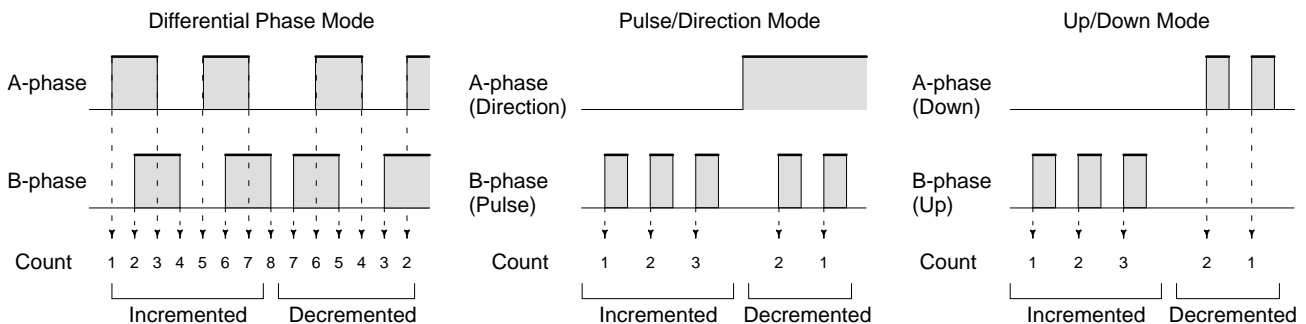
DM 6611 setting	Affected instructions
High-speed counter mode (0000)	PLS2(—) and mode 0 of ACC(—) cannot be used.
Pulse output mode (0001)	CTBL(63) cannot be used with ports 1 and 2.

#### Processing

#### Input Signals and Count Modes

Three types of signals can be input to ports 1 and 2. The count modes used for high-speed counters 1 and 2 are set in DM 6643 and DM 6644 respectively.

- 1, 2, 3...** Differential Phase Mode (Counting Rate = 25 kHz):  
A phase-difference 4X two-phase signal (A-phase and B-phase) and a Z-phase signal are used for inputs. The count is incremented or decremented according to differences in the 2-phase signals. This mode is identical to high-speed counter 0's up/down mode.
- Pulse/Direction Mode (Counting Rate = 50 kHz):  
The A-phase is the direction signal and the B-phase is the count pulse. The counter increments when the A-phase signal is OFF and decrements when it is ON.
- Up/Down Mode (Counting Rate = 50 kHz):  
The A-phase is the decrementing signal and the B-phase is the incrementing signal. The counter decrements when an A-phase pulse is detected and increments when a B-phase pulse is detected.



#### Counting Modes

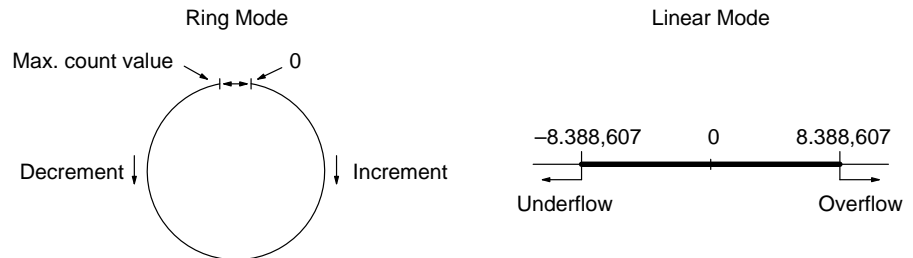
The counting modes (ring mode or linear mode) for high-speed counters 1 and 2 are specified in DM 6643 and DM 6644 respectively.

- 1, 2, 3...** Ring Mode:  
In ring mode, the maximum count value +1 is set in CTBL(63). The counter will go from the maximum count value to 0 when incrementing, and from 0 to the maximum count value when decrementing. There are no negative values.

The number of points on the ring (maximum count value +1) can be set from 1 to 65,000.

2. Linear Mode:

The counting range in linear mode is  $-8,388,607$  to  $8,388,607$ . If the allowable counting range for high-speed counter 1 or 2 is exceeded, an underflow or overflow status will occur and the counter's PV will remain at 0838 8607 for overflows and F838 8607 for underflows, counting or comparison will be stopped, and AR 0509 (port 1) or AR 0609 (port 2) will be turned ON.



- Note**
1. One of the methods in the following section should always be used to reset the counter when restarting it. The counter will be automatically reset when program execution is started or stopped.
  2. The following signal transitions are handled as forward (incrementing) pulses: A-phase leading edge to B-phase leading edge to A-phase trailing edge to B-phase trailing edge. The following signal transitions are handled as reverse (decrementing) pulses: B-phase leading edge to A-phase leading edge to B-phase trailing edge to A-phase trailing edge.

**Reset Methods**

Either the Z-phase signal + software reset or software reset may be selected for resetting the PV of the count (i.e., setting it to 0). These resets operate the same as they do for high-speed counter 0. Refer to page 48 for details.

- Note**
1. The reset bits for high-speed counters 1 and 2 (SR 25201 and SR 25202) are refreshed once every cycle. Make sure that a reset bit is ON for at least one full cycle so it can be read reliably.
  2. The comparison table, execution status, and range comparison results will be retained through a reset. (A comparison will be continued after a reset is performed.)

**High-speed Counter Interrupt Count**

The comparison tables used for high-speed counters 1 and 2 are just like the one used for high-speed counter 0. Refer to page 49 for details.

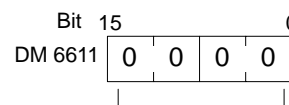
**PC Setup**

When using high-speed counter 1 and/or 2 interrupts, make the settings in PROGRAM mode shown below before executing the program.

**Port 1 and 2 Mode Setting (DM 6611)**

Specify high-speed counter mode for ports 1 and 2. If high-speed counter mode is not specified, CTBL(63) cannot be used to make count comparisons.

This setting is read when the PC is turned ON. If it is changed, the PC must be turned off and then on again before executing the program.



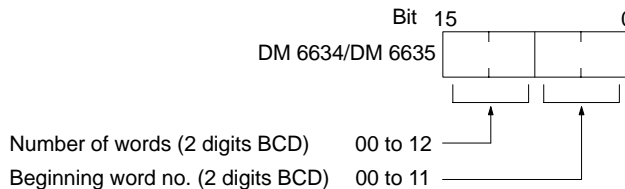
**Port 1 and 2 Mode Setting**  
0000: High-speed counter mode

**Default:** The default mode setting is high-speed counter mode.

**Note** If DM 6611 is set to pulse output mode, another comparison instruction such as BCMP(68) can be used to compare the PV of high-speed counters 1 and 2.

**Input Refresh Word Settings (DM 6634 and DM 6635)**

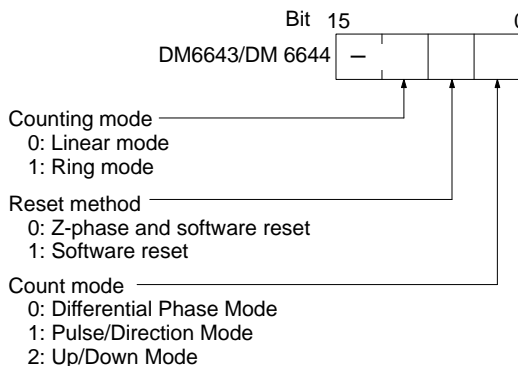
DM 6634 contains the input refresh word settings for high-speed counter 1, and DM 6635 contains the settings for high-speed counter 2. Make these settings when it is necessary to refresh inputs.



**Default:** No input refresh

**High-speed Counter 1 and 2 Settings (DM 6643 and DM 6644)**

DM 6643 contains the settings for high-speed counter 1, and DM 6644 contains the settings for high-speed counter 2. These settings determine the operating parameters for these high-speed counters.



**Defaults:** Linear Mode, Z-phase and software reset, Differential Phase Mode

**Programming**

Use the following steps to program high-speed counters 1 and 2.

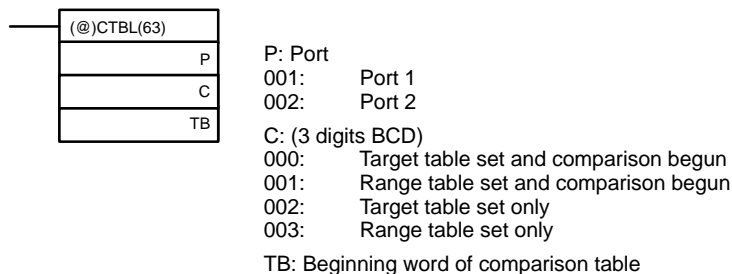
High-speed counters 1 and 2 begin counting when the proper PC Setup settings are made, but comparisons will not be made with the comparison table and interrupts will not be generated unless the CTBL(63) instruction is executed.

High-speed counters 1 and 2 are reset to “0” when power is turned ON, when operation begins, and when operation stops.

The present value of high-speed counter 1 is maintained in SR 232 and SR 233 and the present value of high-speed counter 2 is maintained in SR 234 and SR 235.

**Controlling High-speed Counter 1 and 2 Interrupts**

- 1, 2, 3... 1. Use the CTBL(63) instruction to save the comparison table in the CQM1 and begin comparisons.

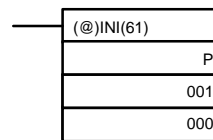


If C is set to 000, then comparisons will be made by the target matching method; if 001, then they will be made by the range comparison method. The comparison table will be saved, and, when the save operation is complete, then comparisons will begin. While comparisons are being executed, high-speed interrupts will be executed according to the comparison table. For details on the contents of the comparison tables that are saved, refer to the explanation of the CTBL(63) instruction in *Section 5 Instruction Set*.

**Note** The comparison results are normally stored in AR 1100 through AR 1107 while the range comparison is being executed.

If C is set to 002, then comparisons will be made by the target matching method; if 003, then they will be made by the range comparison method. For either of these settings, the comparison table will be saved, but comparisons will not begin, and the INI(61) instruction must be used to begin comparisons.

- To stop comparisons, execute the INI(61) instruction as shown below. Specify port 1 or 2 in P (P=001 or 002).



To start comparisons again, set the second operand to “000” (execute comparison), and execute the INI(61) instruction.

Once a table has been saved, it will be retained in the CQM1 during operation (i.e., during program execution) as long as no other table is saved.

**Reading the PV**

There are two ways to read the PV. The first is to read it from SR 232 and SR 233 (port 1) or SR 234 and SR 235 (port 2), and the second is to use PRV(62).

**Reading SR 232 and SR 233 or SR 234 and SR 235**

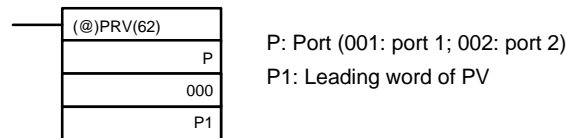
The PV of high-speed counter 1 is stored in SR 232 and SR 233, and the PV of high-speed counter 2 is stored in SR 234 and SR 235 as shown below. In linear mode, the leftmost digit will be F for negative values.

	Leftmost 4 digits	Rightmost 4 digits	Linear Mode	Ring Mode
Port 1:	SR 233	SR 232	F8388607 to 08388607 (-8,388,607 to 8,388,607)	00000000 to 0006499
Port 2:	SR 235	SR 234		

**Note** These words are refreshed only once every cycle, so there may be a difference from the actual PV.

**Using the PRV(62) Instruction**

Read the PV of high-speed counter 0 by using the PRV(62) instruction. Specify high-speed counter 1 or 2 in P (P=001 or 002).



The PV of the specified high-speed counter is stored as shown below. In linear mode, the leftmost bit will be F for negative values.

	Leftmost 4 digits	Rightmost 4 digits	Linear Mode	Ring Mode
	D+1	D	F8388607 to 08388607 (-8,388,607 to 8,388,607)	00000000 to 0006499

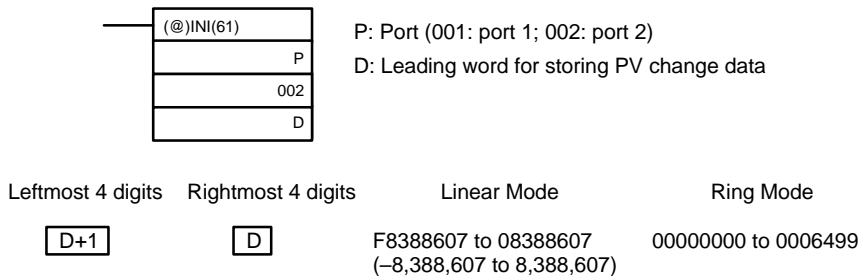
The PV is read when the PRV(62) instruction is actually executed.

**Changing the PV**

There are two ways to change the PV of high-speed counters 1 and 2. The first way is to reset it by using the reset methods. (In this case the PV is reset to 0.) The second way is to use the INI(61) instruction.

The method using the INI(61) instruction is explained here. For an explanation of the reset method, refer to the beginning of this description of high-speed counters 1 and 2.

Change the timer PV by using the INI(61) instruction as shown below.



To specify a negative number in linear mode, set F in the leftmost digit.

**Note** Do not use INI(61) to repeatedly change the PV and start comparison for target value comparison while pulses are being input. If the comparison operation is started immediately after forcing the PV to change, the interrupts may not work properly. The target value will return to the first target value following completion of the interrupt for it, enabling repeating operation by only changing the PV.

**High-speed Counter Status**

The status of high-speed counters 1 and 2 can be determined either by reading the status of the relevant flags in the AR area or executing PRV(62).

The following table shows the relevant AR area flags and their functions.

Word	Bit(s)	Function
AR 04	08 to 15	Indicates high-speed counter status. 00: Normal 01 or 02: Hardware error 03: PC Setup error
AR 05	00 to 07	High-speed Counter 1 Comparison Result flag for ranges 1 to 8. (0: Not in range; 1: In range)
	08	High-speed Counter 1 Comparison flag (0: Stopped; 1: Comparing)
	09	High-speed Counter 1 Overflow/Underflow flag (0: Normal; 1: Underflow or overflow occurred)
AR 06	00 to 07	High-speed Counter 2 Comparison Result flag for ranges 1 to 8. (0: Not in range; 1: In range)
	08	High-speed Counter 2 Comparison flag (0: Stopped; 1: Comparing)
	09	High-speed Counter 2 Overflow/Underflow flag (0: Normal; 1: Underflow or overflow occurred)

The status of high-speed counters 1 and 2 can also be determined by executing PRV(62). Specify high-speed counter 1 or 2 (P=001 to 002) and the destination word D. The status information will be written to bits 00 and 01 of D. Bits 02 to 15 will be set to 0.



Bits 00 and 01 of D contain the specified high-speed counter's status.

Bit	Function
00	Comparison flag (0: Stopped; 1: Comparing)
01	Overflow/Underflow flag (0: Normal; 1: Underflow or overflow occurred)

### Operation Example

This example shows a program that outputs standard pulses from port 1 while counting those pulses with high-speed counter 1. The high-speed counter operates in Up/Down Mode, with the pulse output's CW pulses incrementing the counter (B-phase input) and the CCW pulses decrementing the counter (A-phase input). Before executing the program, set the PC Setup as follows and restart the PC.

DM 6611: 0000 (High-speed counter mode).

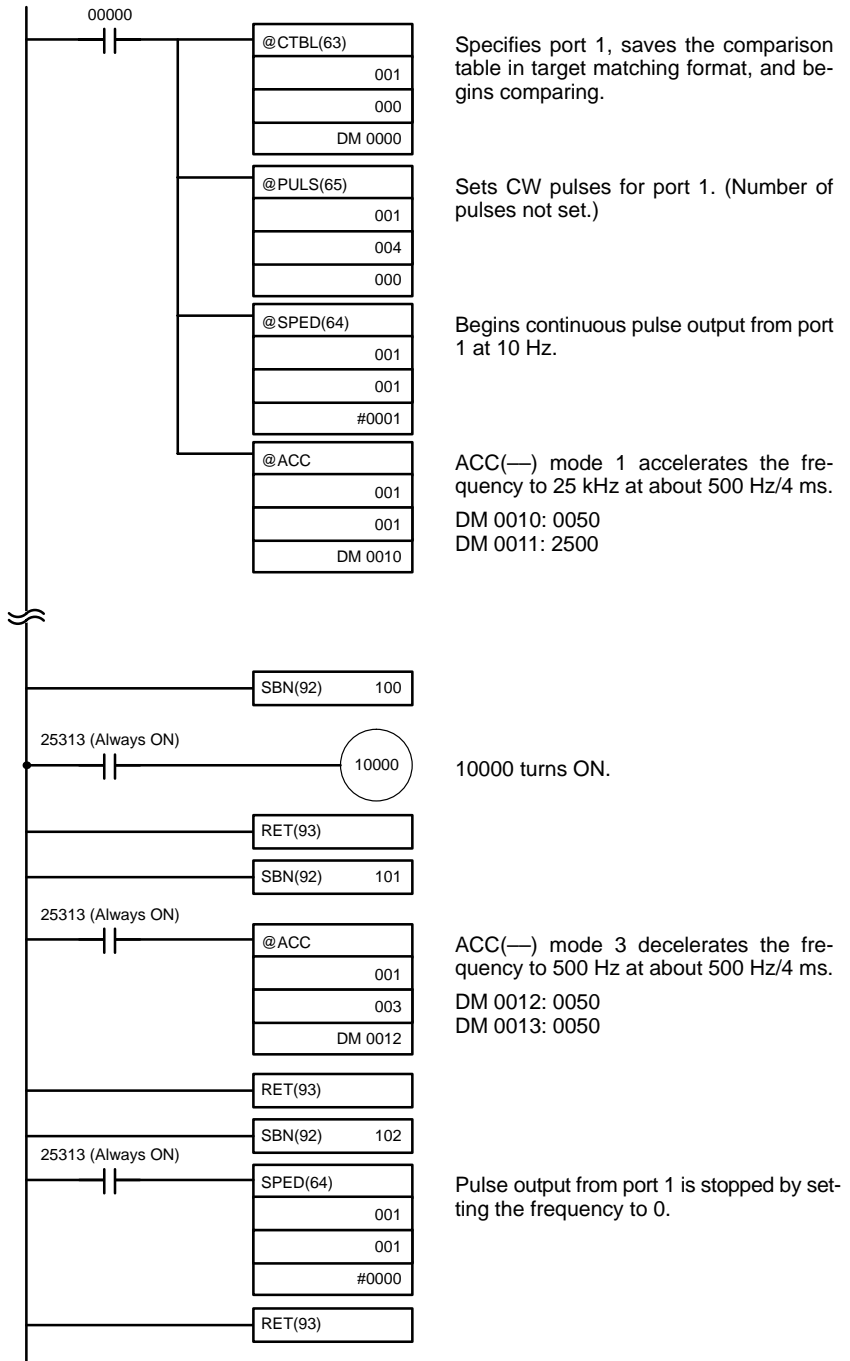
DM 6643: 0002 (Port 1: Standard pulse output, linear counting mode, Z-phase signal with software reset, and Up/Down Mode).

Other PC Setup settings use the default settings. (Inputs are not refreshed at the time of interrupt processing.)



In addition, the following data is stored for the comparison table:

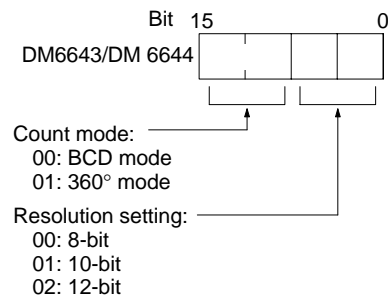
DM 0000	0003	Number of comparison conditions: 3
DM 0001	2500	Target value 1: 2,500
DM 0002	0000	
DM 0003	0100	Comparison 1 interrupt processing routine no.: 100
DM 0004	7500	Target value 2: 7,500
DM 0005	0000	
DM 0006	0101	Comparison 2 interrupt processing routine no.: 101
DM 0007	0000	Target value 2: 10,000
DM 0008	0001	
DM 0009	0102	Comparison 3 interrupt processing routine no.: 102





**Absolute High-speed Counter Settings (DM 6643 and DM 6644)**

DM 6643 contains the settings for absolute high-speed counter 1, and DM 6644 contains the settings for absolute high-speed counter 2. These words determine the count modes and resolution settings.



**Defaults:** BCD mode, 8-bit resolution

**Origin Compensation**

It is possible to compensate for an offset between an absolute rotary encoder's origin and the actual origin. This adjustment can be made separately for ports 1 and 2.

Follow the procedure below to set origin compensation. After origin compensation has been set, the data from the encoder will be adjusted before being output as the PV.

- 1, 2, 3... 1. Set the absolute rotary encoder to the desired origin location.
2. Make sure that pin 1 of the CPU Unit's DIP switch is OFF (enabling Peripheral Devices to overwrite DM 6614 through DM 6655) and switch the PC to PROGRAM mode.
3. Set the resolution setting in DM 6643 or DM 6644.
4. Make sure that a fatal error or FALS 9C error have not occurred.
5. Read the high-speed counter's PV from IR 232 and IR 233 (port 1) or IR 234 and IR 235 (port 2) to determine the PV before origin compensation.
6. Turn ON the Port 1 Origin Compensation flag (SR 25201) or Port 2 Origin Compensation flag (SR 25202) from a Peripheral Device.

The compensation value will be written to DM 6611 (port 1) or DM 6612 (port 2) and the Origin Compensation flag will be turned OFF automatically. The compensation value will be recorded in BCD between 0000 and 4095 whether the counter is set to BCD mode or 360° mode.

7. Read the high-speed counter's PV to determine the PV after origin compensation. The PV should be 0000 after origin compensation.

The compensation value will be valid until it is changed again by the procedure above.

**Programming**

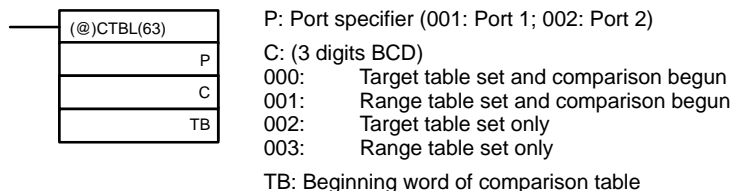
Use the following steps to program absolute high-speed counters 1 and 2.

Absolute high-speed counters 1 and 2 begin counting when the proper PC Setup settings are made, but comparisons will not be made with the comparison table and interrupts will not be generated unless the CTBL(63) instruction is executed.

The present value of absolute high-speed counter 1 is maintained in IR 232 and IR 233 and the present value of absolute high-speed counter 2 is maintained in IR 234 and IR 235.

**Controlling Absolute High-speed Counter Interrupts**

- 1, 2, 3... 1. Use the CTBL(63) instruction to save the comparison table in the CQM1 and begin comparisons.



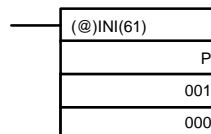
P specifies the port. Set P=001 to specify absolute high-speed counter 1, or P=002 to specify absolute high-speed counter 2.

If C is set to 000, then comparisons will be made by the target matching method; if 001, then they will be made by the range comparison method. The comparison table will be saved, and, when the save operation is complete, then comparisons will begin. While comparisons are being executed, high-speed interrupts will be executed according to the comparison table. Refer to 5-15-6 REGISTER COMPARISON TABLE –CTBL(63) for details on the structure of the comparison tables.

**Note** The comparison results are normally stored in AR 0500 through AR 0507 (port 1) and AR 0600 through AR 0607 (port 2) while the range comparison is being executed.

If C is set to 002, then comparisons will be made by the target matching method; if 003, then they will be made by the range comparison method. For either of these settings, the comparison table will be saved, but comparisons will not begin, and the INI(61) instruction must be used to begin comparisons.

2. To stop comparisons, execute the INI(61) instruction as shown below. Specify port 1 or 2 in P (P=001 or 002).



To start comparisons again, set the second operand to “000” (execute comparison), and execute the INI(61) instruction.

Once a table has been saved, it will be retained in the CQM1 during operation (i.e., during program execution) as long as no other table is saved.

**Reading the PV**

There are two ways to read the PV. The first is to read it from IR 232 and IR 233 (port 1) or IR 234 and IR 235 (port 2), and the second is to use PRV(62).

**Reading IR 232 and IR 233 or IR 234 and IR 235**

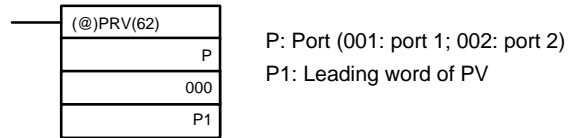
The PV of absolute high-speed counter 1 is stored in IR 232 and IR 233, and the PV of absolute high-speed counter 2 is stored in IR 234 and IR 235 as shown below.

	Leftmost 4 digits	Rightmost 4 digits	BCD Mode	360° Mode
Port 1:	IR 233	IR 232	0000 0000 to 0000 4095	0000 0000 to 0000 0359
Port 2:	IR 235	IR 234		

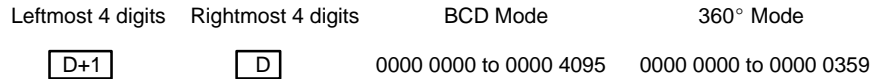
**Note** These words are refreshed only once every cycle, so there may be a difference from the actual PV.

**Using the PRV(62) Instruction**

Read the PV of an absolute high-speed counter by using the PRV(62) instruction. Specify absolute high-speed counter 1 or 2 in P (P=001 or 002).



The PV of the specified absolute high-speed counter is stored as shown below.



The PV is read when the PRV(62) instruction is actually executed.

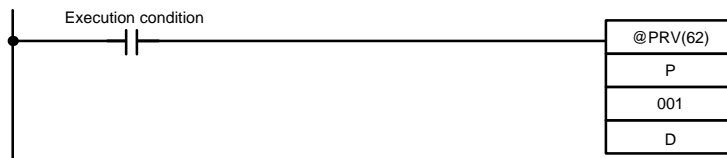
**Reading Absolute High-speed Counter Status**

The status of absolute high-speed counters 1 and 2 can be determined either by reading the status of the relevant flags in the AR area or executing PRV(62).

The following table shows the relevant AR area flags and their functions.

Word	Bit(s)	Function
AR 04	08 to 15	Indicates absolute high-speed counter status. 00: normal 01 or 02: Hardware error 03: PC Setup error
AR 05	00 to 07	Counter 1 Comparison Result flags for ranges 1 to 8. (0: Not in range; 1: In range)
	08	Counter 1 Comparison flag (0: Stopped; 1: Comparing)
AR 06	00 to 07	Counter 2 Comparison Result flags for ranges 1 to 8. (0: Not in range; 1: In range)
	08	Counter 2 Comparison flag (0: Stopped; 1: Comparing)

The comparison flag status of absolute high-speed counters 1 and 2 can also be determined by executing PRV(62). Specify absolute high-speed counter 1 or 2 (P=001 to 002) and the destination word D. The flag status (0: Stopped; 1: Comparing) will be written to bit 00 of D. Bits 01 to 15 will be set to 0.



**Operation Example**

This example shows a program that receives an input signal from an absolute rotary encoder at port 1 and uses this input to control outputs IR 10000 through IR 10003. Absolute high-speed counter 1 is set for 8-bit resolution and 360° Mode, and range comparisons are used. Before executing the program, set DM 6643 to 0100 (Port 1: 360° Mode, 8-bit resolution).

Other PC Setup settings use the default settings. (Inputs are not refreshed at the time of interrupt processing.)

In addition, the following data is stored for the comparison table:

DM 0000	0000	Lower limit #1 (0°)
DM 0001	0085	Upper limit #1 (85°)
DM 0002	0100	Subroutine number 100
DM 0003	0090	Lower limit #2 (90°)
DM 0004	0175	Upper limit #2 (175°)
DM 0005	0101	Subroutine number 101
DM 0006	0180	Lower limit #3 (180°)
DM 0007	0265	Upper limit #3 (265°)
DM 0008	0102	Subroutine number 102
DM 0009	0270	Lower limit #4 (270°)
DM 0010	0355	Upper limit #4 (355°)
DM 0011	0103	Subroutine number 103
DM 0012	0000	Lower limit #1 (0°)
DM 0013	0000	Upper limit #1 (0°)
DM 0014	FFFF	No subroutine number
DM 0015	0000	Lower limit #1 (0°)
DM 0016	0000	Upper limit #1 (0°)
DM 0017	FFFF	No subroutine number
DM 0018	0000	Lower limit #1 (0°)
DM 0019	0000	Upper limit #1 (0°)
DM 0020	FFFF	No subroutine number
DM 0021	0000	Lower limit #1 (0°)
DM 0022	0000	Upper limit #1 (0°)
DM 0023	FFFF	No subroutine number

→ First range setting (0° to 85°)

→ Second range setting (90° to 175°)

→ Third range setting (180° to 265°)

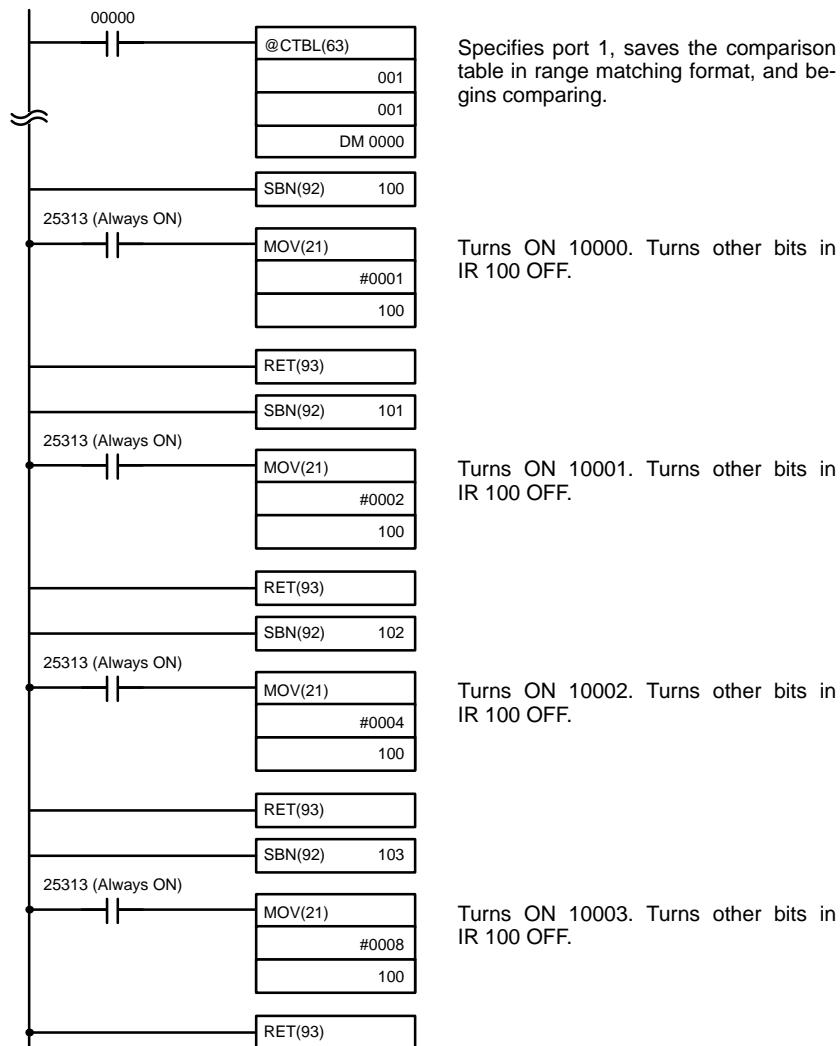
→ Fourth range setting (270° to 355°)

→ Fifth range setting (Not used.)

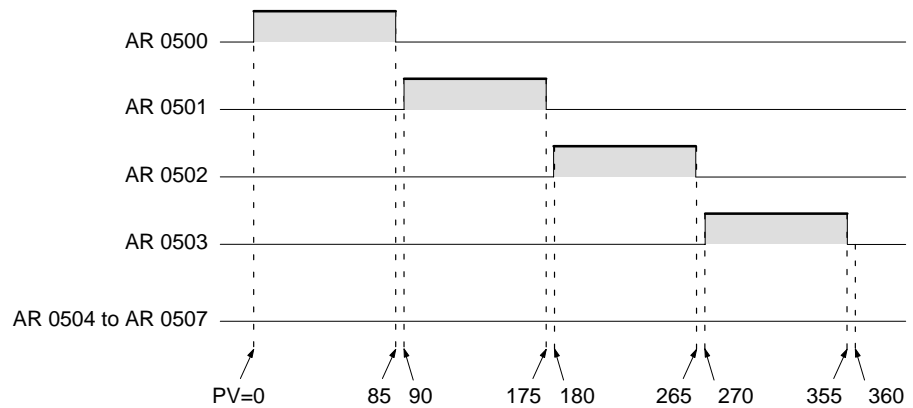
→ Sixth range setting (Not used.)

→ Seventh range setting (Not used.)

→ Eighth range setting (Not used.)



The following diagram shows the relationship between absolute high-speed counter 1's PV and Range Comparison Result flags AR 0500 to AR 0507 as the above program is executed.



## 1-6 CPM1/CPM1A Interrupt Functions

This section explains the settings and methods for using the CPM1/CPM1A interrupt functions.

### 1-6-1 Types of Interrupts

The CPM1/CPM1A has three types of interrupt processing, as outlined below.

#### Input Interrupts

CPM1/CPM1A PCs have two or four interrupt inputs. Interrupt processing is executed when one of these inputs is turned ON from an external source.

#### Interval Timer Interrupts

Interrupt processing is executed by an interval timer with a precision of 0.1 ms.

#### High-speed Counter Interrupts

The high-speed counter counts pulse inputs to one of CPU bits 00000 to 00002. Interrupt processing is executed when the count reaches the set value of a built-in high-speed counter.

#### Interrupt Priority

When an interrupt is generated, the specified interrupt processing routine is executed. Interrupts have the following priority ranking.

Input interrupts > Interval interrupts = High-speed counter interrupts

When an interrupt with a higher priority is received during interrupt processing, the current processes will be stopped and the newly received interrupt will be processed instead. After that routine has been completely executed, then processing of the previous interrupt will be resumed.

When an interrupt with a lower or equal priority is received during interrupt processing, then the newly received interrupt will be processed as soon as the routine currently being processed has been completely executed.

When two interrupts with equal priority are received at the same time, they are executed in the following order:

Input interrupt 0 > Input interrupt 1 > Input interrupt 2 > Input interrupt 3

Interval interrupts > High-speed counter interrupts

#### Interrupt Program Precautions

1, 2, 3...

Observe the following precautions when using interrupt programs:

1. A new interrupt can be defined within an interrupt program. Furthermore, an interrupt can be cleared from within an interrupt program.
2. Another interrupt program cannot be written within an interrupt program.
3. A subroutine program cannot be written within an interrupt program. Do not write a SUBROUTINE DEFINE instruction, SBN(92), within an interrupt program.

4. An interrupt program cannot be written within a subroutine program. Do not write an interrupt program between a SUBROUTINE DEFINE instruction (SBN(92)) and a RETURN instruction (RET(93)).

Inputs used as interrupts cannot be used as regular inputs.

**High-speed Counter Instructions and Interrupts**

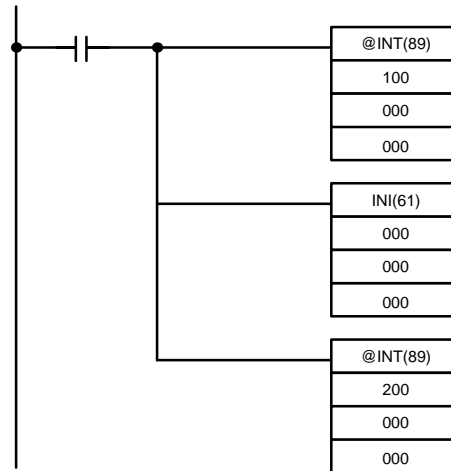
The following instructions cannot be executed in an interrupt subroutine when an instruction that controls high-speed counters is being executed in the main program:

INI(61), PRV(62), or CTBL(63)

The following methods can be used to circumvent this limitation:

**Method 1**

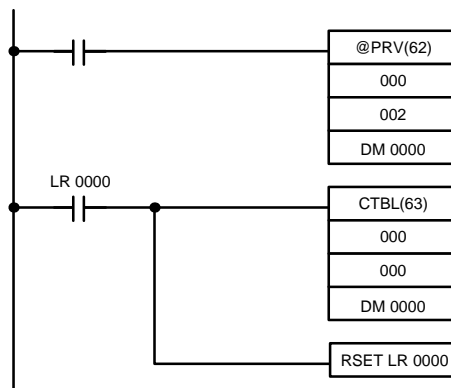
All interrupt processing can be masked while the instruction is being executed.



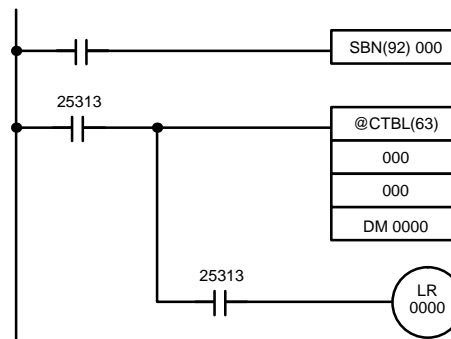
**Method 2**

Execute the instruction again in the main program.

1, 2, 3... 1. This is the program section from the main program:



2. This is the program section from the interrupt subroutine:





- Note**
1. Define interrupt routines at the end of the main program with SBN(92) and RET(93) instructions, just like regular subroutines.
  2. When defining an interrupt routine, a “SBS UNDEFD” error will occur during the program check operation, but the program will be executed normally.

### 1-6-2 Input Interrupts

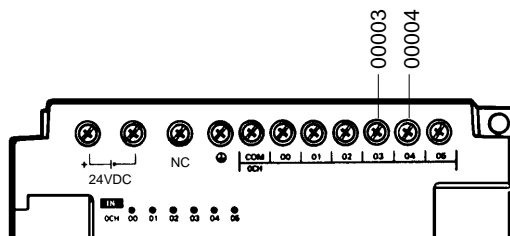
The 10-pt CPU Units (CPM1-10CDR-□ and CPM1A-10CDR-□) have two interrupt inputs (00003 and 00004).

The 20-, 30-, and 40-pt CPU Units (CPM1-20CDR-□, CPM1A-20CDR-□, CPM1-30CDR-□(-V1), CPM1A-30CDR-□ and CPM1A-40CDR-□) have four interrupt inputs (00003 to 00006).

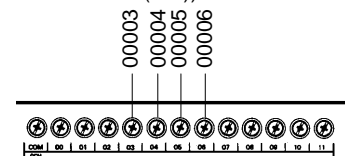
There are two modes for input interrupts: input interrupt mode and counter mode.

#### CPM1 PCs

10-pt CPU Units  
(CPM1-10CDR-□)

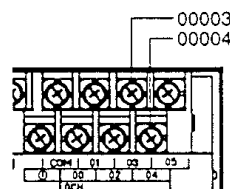


20- and 30-pt CPU Units  
(CPM1-20CDR-□ and CPM1-30CDR-□(-V1))

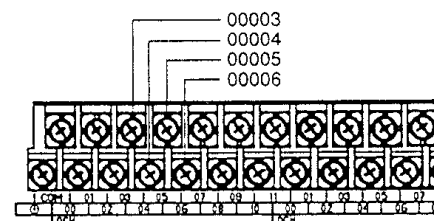


#### CPM1A PCs

10-pt CPU Units  
(CPM1A-10CDR-□)



20-, 30-, and 40-pt CPU Units  
(CPM1A-20CDR-□, CPM1A-30CDR-□, and CPM1A-40CDR-□)



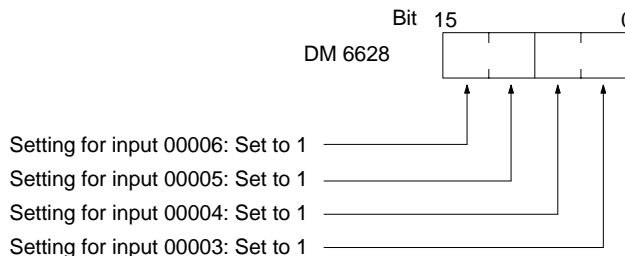
CPU Unit	Input	Interrupt number	Response time	
			Interrupt mode	Counter mode
CPM1-10CDR-□ CPM1A-10CD□-□	00003	00	0.3 ms max.  (Time until the interrupt program is executed.)	1 kHz
	00004	01		
CPM1-20CDR-□ CPM1A-20CD□-□	00003	00		
	00004	01		
CPM1-30CDR-□(-V1) CPM1A-30CD□-□	00003	02		
	00004	03		
CPM1A-40CD□-□	00004	03		

**Note** If input interrupts are not used, use inputs 00003 to 00006 as regular inputs.

**Input Interrupt Settings**

Inputs 00003 to 00006 must be set as interrupt inputs in DM 6628 if they are to be used for input interrupts in the CPM1/CPM1A. Set the corresponding digit to 1 if the input is to be used as an interrupt input (input interrupt or counter mode); set it to 0 if it is to be used as a regular input.

Word	Setting
DM 6628	0: Regular input (default setting) 1: Interrupt input 2: Quick-response input



**Interrupt Subroutines**

Interrupts from inputs 00003 to 00006 are allocated interrupt numbers 00 to 03 and call subroutines 000 to 003. If the input interrupts aren't being used, subroutines 000 to 003 can be used in regular subroutines.

Input number	Interrupt number	Subroutine number
00003	0	000
00004	1	001
00005	2	002
00006	3	003

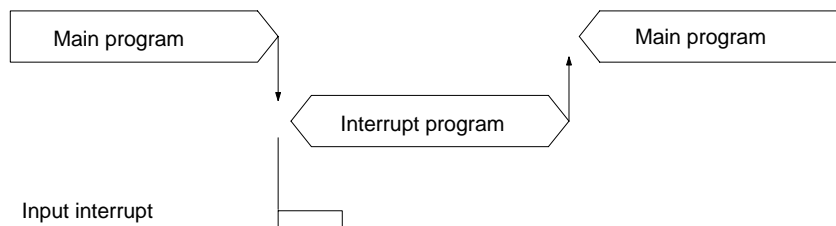
**Input Refreshing**

If input refreshing is not used, input signal status within the interrupt routine will not be reliable. Depending on the input time constant, the input signals might not go ON even if input refreshing is used. This includes the status of the interrupt input bit that activated the interrupt.

For example, IR 00000 would not be ON in interrupt routine for input interrupt 0 unless it was refreshed. In this case, use the Always ON Flag, SR 25313 in the interrupt routine instead of IR 00000.

**Input Interrupt Mode**

When an input interrupt signal is received, the main program is interrupted and the interrupt program is executed immediately, regardless of the point in the cycle in which the interrupt is received. The signal must be ON for 200 μs or more to be detected.



Use the following instructions to program input interrupts using the Input Interrupt Mode.

**Masking/Unmasking of Interrupts**

With the INT(89) instruction, set or clear input interrupt masks as required.

(@)INT(89)	
	000
	000
	D

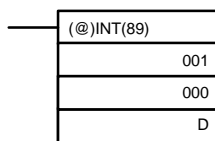
Make the settings with word D bits 0 to 3, which correspond to input interrupts 0 to 3.  
0: Mask cleared. (Input interrupt enabled.)  
1: Mask set. (Input interrupt disabled.)

All of the input interrupts are masked when PC operation is started. If input interrupt mode is being used, be sure to enable the inputs by executing INT(89) as shown above.

**Clearing Masked Interrupts**

If the bit corresponding to an input interrupt turns ON while masked, that input interrupt will be saved in memory and will be executed as soon as the mask is cleared. In order for that input interrupt not to be executed when the mask is cleared, the interrupt must be cleared from memory.

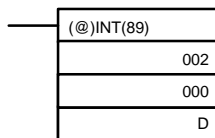
Only one interrupt signal will be saved in memory for each interrupt number. With the INT(89) instruction, clear the input interrupt from memory.



If D bits 0 to 3, which correspond to input interrupts 0 to 3, are set to "1," then the input interrupts will be cleared from memory.  
 0: Input interrupt retained.  
 1: Input interrupt cleared.

**Reading Mask Status**

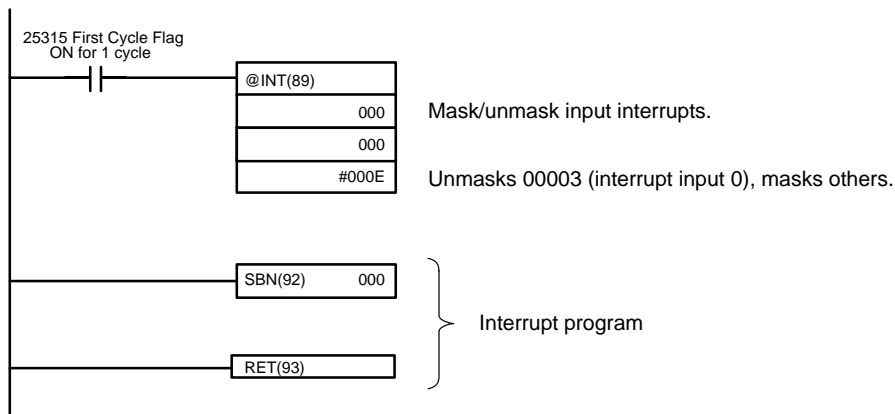
With the INT(89) instruction, read the input interrupt mask status.



The status of the rightmost digit of the data stored in word D (bits 0 to 3) show the mask status.  
 0: Mask cleared. (Input interrupt enabled.)  
 1: Mask set. (Input interrupt disabled.)

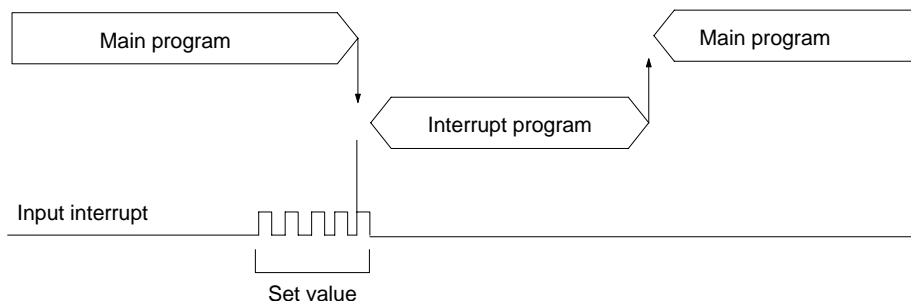
**Program Example**

When input 00003 (interrupt no. 0) goes ON, operation moves immediately to the interrupt program with subroutine number 000. Inputs for DM 6628 have been set to 0001.



**Counter Mode**

External signal inputs are counted at high speed and an interrupt is generated when the count reaches the set value. When an interrupt is generated, the main program is interrupted and the interrupt program is executed. Signals up to 1 kHz can be counted.



Use the following steps to program input interrupts using the Counter Mode.

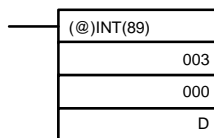
- 1, 2, 3...
1. Write the set values for counter operation to the SR words shown in the following table. The set values are written between 0000 and FFFF (0 to 65,535). A value of 0000 will disable the count operation until a new value is set and step 2, below, is repeated.

Interrupt	Word
Input interrupt 0	SR 240
Input interrupt 1	SR 241
Input interrupt 2	SR 242
Input interrupt 3	SR 243

The SR words used in the Counter Mode (SR 240 to SR 243) contain hexadecimal data, not BCD. If the Counter Mode is not used, these words can be used as work bits.

**Note** These SR words are cleared at the beginning of operation, and must be written from the program.

2. With the INT(89) instruction, refresh the Counter Mode set value and enable interrupts.



If D bits 0 to 3, which correspond to input interrupts 0 to 3, are set to "0," then the set value will be refreshed and interrupts will be permitted.  
 0: Counter mode set value refreshed and mask cleared.  
 1: Not refreshed.

Be sure to set the corresponding bit to 1 if an input interrupt isn't being controlled.

The input interrupt for which the set value is refreshed will be enabled in Counter Mode. When the counter reaches the set value, an interrupt will occur, the counter will be reset, and counting/interrupts will continue until the counter is stopped.

- Note**
1. If the INT(89) instruction is used during counting, the present value (PV) will return to the set value (SV). You must, therefore, use the differentiated form of the instruction or an interrupt may never occur.
  2. The set value will be set when the INT(89) instruction is executed. If interrupts are already in operation, then the set value will not be changed just by changing the content of SR 240 to SR 243, i.e., if the contents are changed, the set value must be refreshed by executing the INT(89) instruction again.

Interrupts can be masked using the same process used with the Input Interrupt Mode, but if the masked interrupts are cleared using the same process, the interrupts will operate in Input Interrupt Mode, not Counter Mode.

Interrupt signals received for masked interrupts can also be cleared using the same process as for the Input Interrupt Mode.

**Counter PV in Counter Mode**

When input interrupts are used in Counter Mode, the counter PV will be stored in the SR word corresponding to input interrupts 0 to 3. Values are 0000 to FFFE (0 to 65,534) and will equal the counter PV minus one.

Interrupt	Word
Input interrupt 0	SR 244
Input interrupt 1	SR 245
Input interrupt 2	SR 246
Input interrupt 3	SR 247

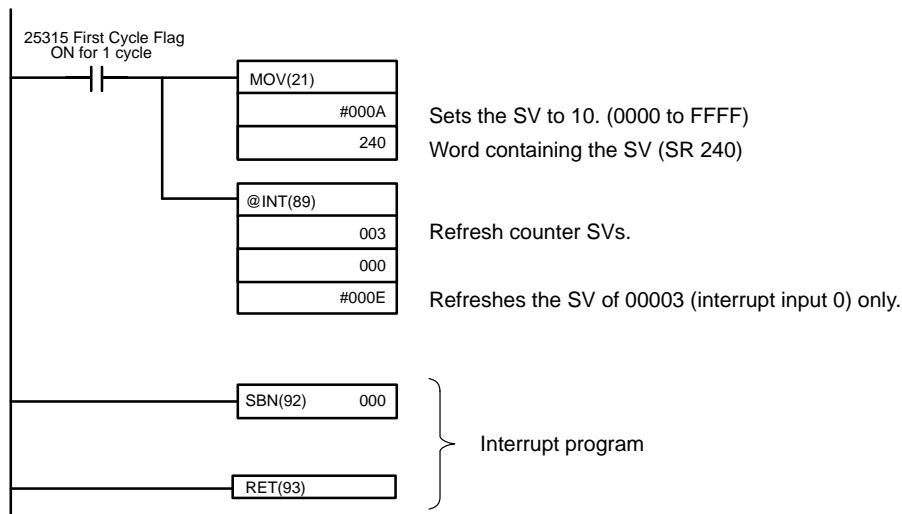
**Example:** The present value for an interrupt whose set value is 000A will be recorded as 0009 immediately after INT(89) is executed.

- Note** Even if input interrupts are not used in Counter Mode, these SR bits cannot be used as work bits.

**Program Example**

When input 00003 (interrupt no. 0) goes ON 10 times, operation moves immediately to the interrupt program with subroutine number 000. The following table shows where the counters' set values and present values -1 are stored. Inputs for DM 6628 have been set to 0001.

Interrupt	Word containing SV	Word containing PV-1
Input 00003 (input interrupt 0)	SR 240	SR 244
Input 00004 (input interrupt 1)	SR 241	SR 245
Input 00005 (input interrupt 2)	SR 242	SR 246
Input 00006 (input interrupt 3)	SR 243	SR 247



**1-6-3 Masking All Interrupts**

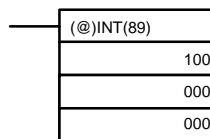
All interrupts, including input interrupts, interval timer interrupts, and high-speed counter interrupts, can be masked and unmasked as a group by means of the INT(89) instruction. The mask is in addition to any masks on the individual types of interrupts. Furthermore, clearing the masks for all interrupts does not clear the masks on the individual types of interrupts, but restores them to the masked conditions that existed before INT(89) was executed to mask them as a group.

Do not use INT(89) to mask interrupts unless it is necessary to temporarily mask all interrupts and always use INT(89) instructions in pairs to do so, using the first INT(89) instruction to mask and the second one to unmask interrupts.

INT(89) cannot be used to mask and unmask all interrupts from within interrupt routines.

**Masking Interrupts**

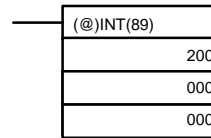
Use the INT(89) instruction to disable all interrupts.



If an interrupt is generated while interrupts are masked, interrupt processing will not be executed but the interrupt will be recorded for the input, interval timer, and high-speed counter interrupts. The interrupts will then be serviced as soon as interrupts are unmasked.

**Unmasking Interrupts**

Use the INT(89) instruction to unmask interrupts as follows:



**1-6-4 Interval Timer Interrupts**

The CPM1/CPM1A is equipped with one interval timer. When the interval timer times out, the main program is interrupted and the interrupt program is executed immediately, regardless of the point in the cycle.

There are two modes for interval timer operation, the One-shot Mode, in which only one interrupt will be executed when time expires, and the Scheduled Interrupt Mode in which the interrupt is repeated at a fixed interval.

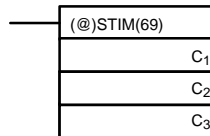
The interval timer's set value can be set anywhere from 0.5 to 319968 ms, in units of 0.1 ms.

**Operation**

Use the following instruction to activate and control the interval timer.

**Starting Up in One-Shot Mode**

Use the STIM(69) instruction to start the interval timer in the one-shot mode.



- C<sub>1</sub>: Interval timer, one-shot mode (000)
- C<sub>2</sub>: Timer set value (first word address)
- C<sub>3</sub>: Subroutine no. (4 digits BCD): 0000 to 0049

**1, 2, 3...**

1. When C<sub>2</sub> is entered as a word address:

C<sub>2</sub>: Decrementing counter set value (4 digits BCD): 0000 to 9999

C<sub>2</sub> + 1: Decrementing time interval (4 digits BCD; unit: 0.1 ms): 0005 to 0320 (0.5 ms to 32 ms)

Each time that the interval specified in word C<sub>2</sub> + 1 elapses, the decrementing counter will decrement the present value by one. When the PV reaches 0, the designated subroutine will be called just once and the timer will stop.

The time from when the STIM(69) instruction is executed until time elapses is calculated as follows:

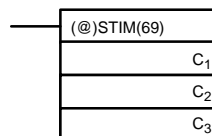
$$(\text{Content of } C_2) \times (\text{Content of } C_2 + 1) \times 0.1 \text{ ms} = (0.5 \text{ to } 319,968 \text{ ms})$$

2. When C<sub>2</sub> is entered as a constant:

The set value of the decrementing counter will equal the specified constant (in ms) and the decrementing time interval will be 10 (1 ms).

**Starting Up in Scheduled Interrupt Mode**

Use the STIM(69) instruction to start the interval timer in the scheduled interrupt mode.



- C<sub>1</sub>: Interval timer, scheduled interrupt mode (003)
- C<sub>2</sub>: Timer set value (leading word no.)
- C<sub>3</sub>: Subroutine no. (4 digits BCD): 0000 to 0049

**1, 2, 3...**

1. When C<sub>2</sub> is entered as a word address:

C<sub>2</sub>: Decrementing counter set value (4 digits BCD): 0000 to 9999

C<sub>2</sub> + 1: Decrementing time interval (4 digits BCD; unit: 0.1 ms): 0005 to 0320 (0.5 ms to 32 ms)

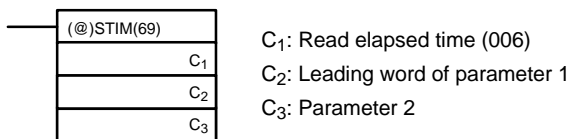
The meanings of the settings are the same as for the one-shot mode, but in the scheduled interrupt mode the timer PV will be reset to the set value and decrementing will begin again after the subroutine has been called. In the scheduled interrupt mode, interrupts will continue to be repeated at fixed intervals until the operation is stopped.

2. When C<sub>2</sub> is entered as a constant:

The settings are the same as for the one-shot mode, but interrupts will continue to be repeated at fixed intervals until the operation is stopped.

**Reading the Timer's Elapsed Time**

Use the STIM(69) instruction to read the timer's elapsed time.



C<sub>2</sub>: Number of times the decrementing counter has ben decremented (4 digits BCD)

C<sub>2</sub> + 1: Decrementing counter time interval (4 digits BCD; unit: 0.1 ms)

C<sub>3</sub>: Elapsed time from previous decrement (4 digits BCD; unit: 0.1 ms)

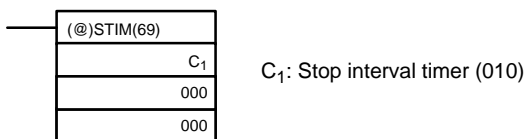
The time from when the interval timer is started until the execution of this instruction is calculated as follows:

$$\{(Content\ of\ C_2) \times (Content\ of\ C_2+1) + (Content\ of\ C_3)\} \times 0.1\ ms$$

If the specified interval timer is stopped, then "0000" will be stored.

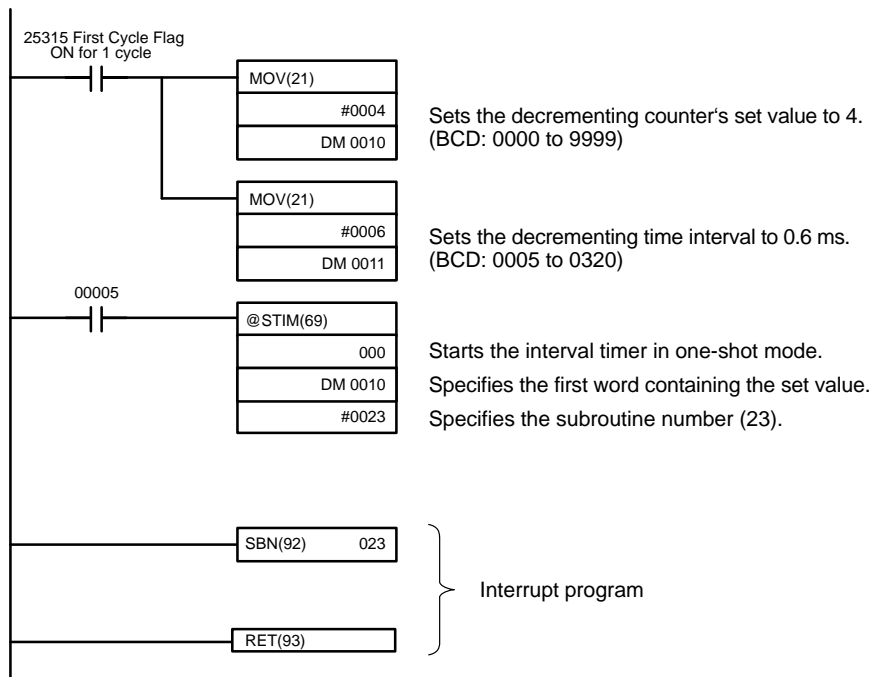
**Stopping the Timer**

Use the STIM(69) instruction to stop the interval timer. The interval timer will be stopped.



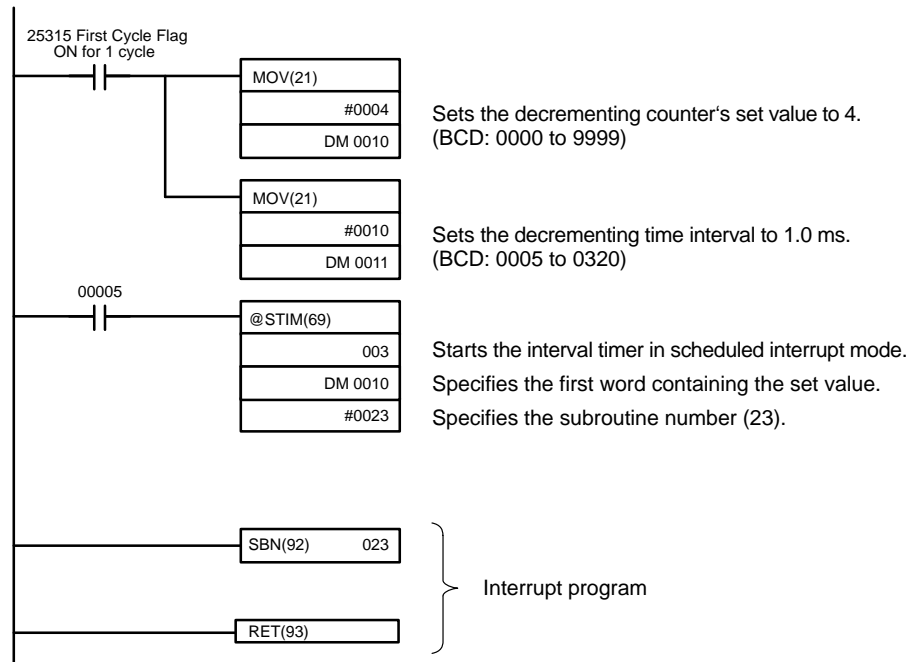
**Application Example (One-shot Mode)**

In this example, an interrupt is generated 2.4 ms (0.6 ms × 4) after input 00005 goes ON; the interrupt executes interrupt subroutine number 23.



**Application Example  
(Scheduled Interrupt Mode)**

In this example, an interrupt is generated every 4.0 ms (1.0 ms × 4) after input 00005 goes ON; the interrupts execute interrupt subroutine number 23.

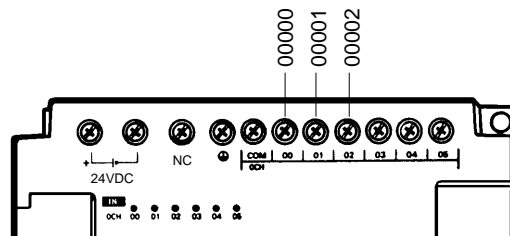


**1-6-5 High-speed Counter Interrupts**

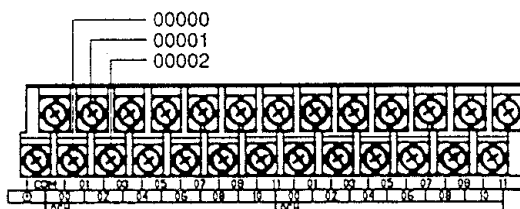
CPM1/CPM1A PCs have a high-speed counter function that can be used in incrementing mode or up/down mode. The high-speed counter can be combined with input interrupts to perform target value control or zone comparison control that isn't affected by the PC's cycle time.

High-speed counter signals can be input to CPU bits 00000 through 00002.

**CPM1 PCs**



**CPM1A PCs**





Mode	Input functions	Input method	Count frequency	Count range	Control methods
Up/Down	00000: A-phase input 00001: B-phase input 00002: Z-phase input	Phase-difference, 4× inputs	2.5 kHz max.	-32767 to 32767	Target value control: Up to 16 target values and interrupt subroutine numbers can be registered.
Incrementing	00000: Count input 00001: See note. 00002: Reset input	Individual inputs	5.0 kHz max.	0 to 65535	Zone comparison control: Up to 8 sets of upper limit values, lower limit values, and interrupt subroutine numbers can be registered.

**Note** In incrementing mode, input 00001 can be used as a regular input. When the reset method is used for the software reset, input 00002 can be used as a regular input. Also, even when used for the Z-phase signal and software reset, the input status is reflected in 00002 of the I/O memory.

**High-speed Counter Settings** The following settings must be made in DM 6642 when using the CPM1/CPM1A's high-speed counter function.

DM 6642 Bits	Function	Settings		
		Incrementing	Up/Down	Not used
00 to 03	Sets the counter mode: 0: Up/down 4: Incrementing	4	0	0 or 4
04 to 07	Sets the reset method: 0: Z-phase + software reset 1: Software reset	0 or 1	0 or 1	0 or 1
08 to 15	Sets the counter: 00: Counter not being used. 01: Counter being used.	01	01	00

**Count Range**

The CPM1/CPM1A's high-speed counter uses linear operation and the count (present value) is stored in SR 248 and SR 249. (The upper four digits are stored in SR 249 and the lower four digits are stored in SR 248.)

Mode	Count range
Up/Down	F003 2767 to 0003 2767 (-32,767 to 32,767) The leftmost digit in SR 248 indicates the sign. F is negative, 0 is positive.
Incrementing	0000 0000 to 0006 5535 (0 to 65,535)

An overflow will occur if the count exceeds the upper limit in the count range and an underflow will occur if the count goes below the lower limit in the count range.

Error	Incrementing	Up/Down	Present value
Overflow	Occurs when the count is incremented from 65,535.	Occurs when the count is incremented from 32,767.	0FFF FFFF
Underflow	---	Occurs when the count is decremented from -32,767.	FFFF FFFF

**Processing**

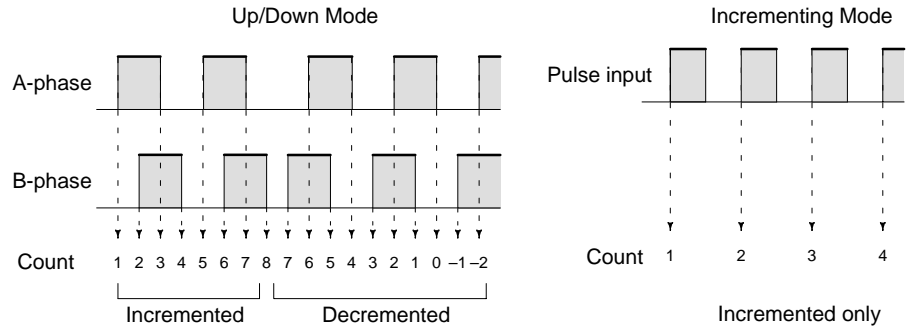
Two types of signals can be input from a pulse encoder. The count mode used for the high-speed counter will depend on the signal type. The count mode and reset mode are set in DM 6642; these settings become effective when the power is turned on or PC operation is started.

**Up/Down Mode:**

A phase-difference 4× two-phase signal (A-phase and B-phase) and a Z-phase signal are used for inputs. The count is incremented or decremented according to differences in the 2-phase signals.

**Incrementing Mode:**

One single-phase pulse signal and a count reset signal are used for inputs. The count is incremented according to the single-phase signal.



**Note** One of the reset methods described below should always be used to reset the counter when restarting it. The counter will be automatically reset when program execution is started or stopped.

The following signal transitions are handled as forward (incrementing) pulses: A-phase leading edge to B-phase leading edge to A-phase trailing edge to B-phase trailing edge. The following signal transitions are handled as reverse (decrementing) pulses: B-phase leading edge to A-phase leading edge to B-phase trailing edge to A-phase trailing edge.

The Up/Down Mode always uses a 4x phase-difference input. The number of counts for each encoder revolution would be 4 times the resolution of the counter. Select the encoder based on the countable ranges.

**Reset Methods**

Either of the two methods described below may be selected for resetting the PV of the count (i.e., setting it to 0).

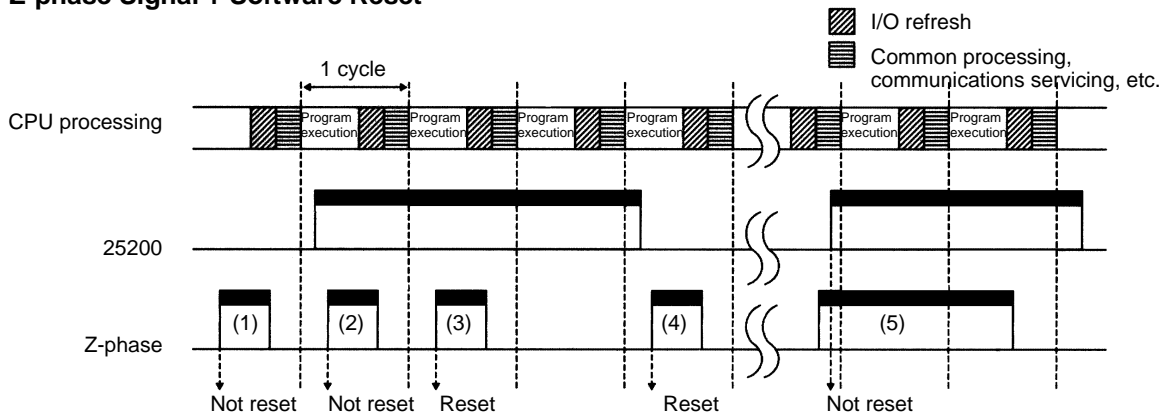
Z-phase signal + software reset:

The PV is reset when the Z-phase signal (reset input) is turned ON while the High-speed Counter Reset Bit (SR 25200) is ON.

Software reset:

The PV is reset when the High-speed Counter Reset Bit (SR 25200) is turned ON.

**Z-phase Signal + Software Reset**



No.	Operation timing	Reset
(1)	Z-phase signal turns ON when SR 25200 turns OFF.	Not reset.
(2)	Z-phase signal turns ON within one cycle after SR 25200 turns ON.	Not reset.
(3)	Z-phase signal turns ON after at least one cycle elapses after SR 25200 turns ON.	Reset with Z-phase leading edge.
(4)	Z-phase signal turns ON within one cycle after SR 25200 turns OFF.	Reset with Z-phase leading edge.
(5)	SR 25200 turns ON when Z-phase signal is ON.	Not reset.

**Note** The High-speed Counter Reset Bit (SR 25200) is refreshed once every cycle, so in order for it to be read reliably it must be ON for at least one cycle.

The “Z” in “Z-phase” is an abbreviation for “Zero.” It is a signal that shows that the encoder has completed one cycle.

**High-speed Counter Interrupt Count**

For high-speed counter 0 interrupts, a comparison table is used instead of a “count up.” The count check can be carried out by either of the two methods described below. In the comparison table, comparison conditions (for comparing to the PV) and interrupt routine combinations are saved.

Target value:

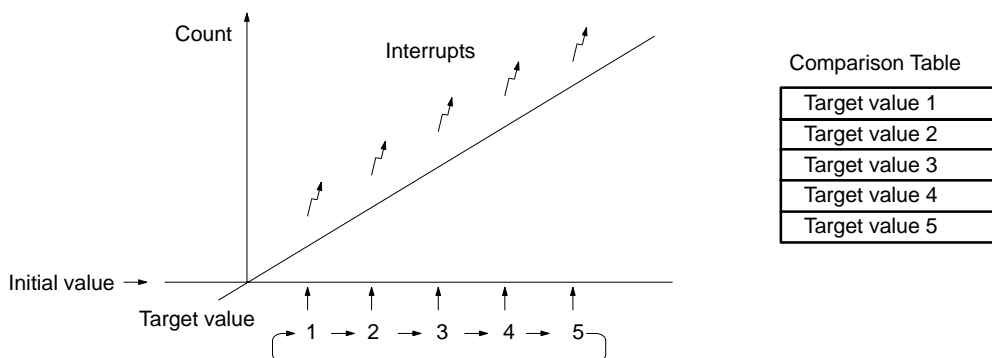
A maximum of 16 comparison conditions (target values and count directions) and interrupt routine combinations are saved in the comparison table. When the counter PV and the count direction match the comparison conditions, then the specified interrupt routine is executed.

Range (zone) comparison:

Eight comparison conditions (upper and lower limits) and interrupt routine combinations are saved in the comparison table. When the PV is greater than or equal to the lower limit and less than or equal to the upper limit, then the specified interrupt routine is executed.

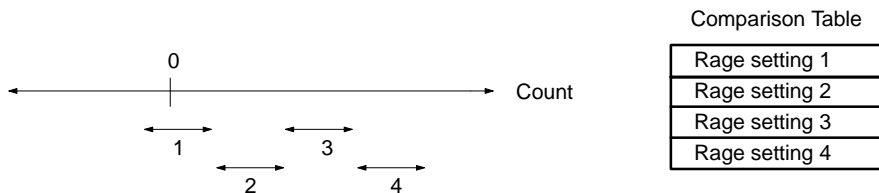
**Target Value Comparisons**

The current count is compared to the target values in the order that target values are set in the comparison table and interrupts are generated as the count equals each target value. Once the count has equaled all of the target values in the table, the target value is set to the first target value in the table, which is again compared to the current counted until the two values are equal.



**Range Comparisons**

The current count is compared in cyclic fashion to all of the ranges at the same time and interrupts are generated based on the results of the comparisons.



**Note** When performing target value comparisons, do not repeatedly use the INI instruction to change the current value of the count and start the comparison operation. The interrupt operation may not work correctly if the comparison operation is started immediately after changing the current value from the program. (The comparison operation will automatically return to the first target value once an interrupt has been generated for the last target value. Repetitious operation is thus possible merely by changing the current value.)

**Programming**

Use the following steps to program the high-speed counter.

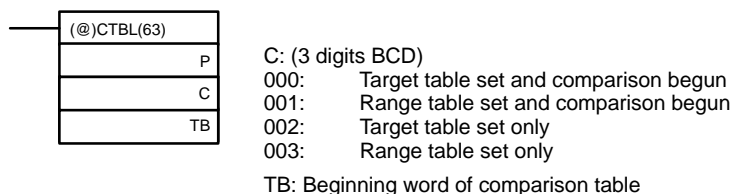
The high-speed counter begins the counting operation when the proper PC Set-up settings are made, but comparisons will not be made with the comparison table and interrupts will not be generated unless the CTBL(63) instruction is executed.

The high-speed counter is reset to "0" when power is turned ON and when operation begins.

The present value of high-speed counter is maintained in SR 248 and SR 249.

**Controlling High-speed Counter Interrupts**

- 1, 2, 3... 1. Use the CTBL(63) instruction to save the comparison table in the CPM1/CPM1A and begin comparisons.

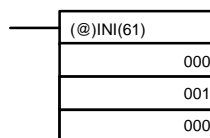


If C is set to 000, then comparisons will be made by the target matching method; if 001, then they will be made by the range comparison method. The comparison table will be saved, and, when the save operation is complete, then comparisons will begin. While comparisons are being executed, high-speed interrupts will be executed according to the comparison table. For details on the contents of the comparison tables that are saved, refer to the explanation of the CTBL(63) instruction in *Section 5 Instruction Set*.

**Note** The comparison results are normally stored in AR 1100 through AR 1107 while the range comparison is being executed.

If C is set to 002, then comparisons will be made by the target matching method; if 003, then they will be made by the range comparison method. For either of these settings, the comparison table will be saved, but comparisons will not begin, and the INI(61) instruction must be used to begin comparisons.

2. To stop comparisons, execute the INI(61) instruction as shown below.



To start comparisons again, set the second operand to "000" (execute comparison), and execute the INI(61) instruction.

Once a table has been saved, it will be retained in the CPM1/CPM1A during operation (i.e., during program execution) as long as no other table is saved.

**Reading the PV**

There are two ways to read the PV. The first is to read it from SR 248 and SR 249, and the second to use the PRV(62) instruction.

**Reading SR 248 and SR 249**

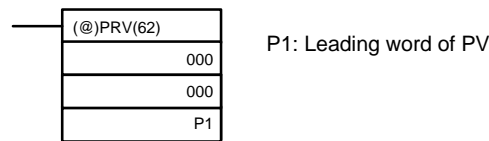
The PV of high-speed counter is stored in SR 248 and SR 249 as shown below. The leftmost bit will be F for negative values.

Leftmost 4 digits	Rightmost 4 digits	Up/Down Mode	Incrementing Mode
SR 249	SR 248	F0032767 to 00032767 (-32767)	00000000 to 00065535

- Note**
1. These words are refreshed only once every cycle, so there may be a difference from the actual PV.
  2. When high-speed counter is not being used, the bits in these words can be used as work bits.

**Using the PRV(62) Instruction**

Read the PV of the high-speed counter by using the PRV(62) instruction.



The PV of the high-speed counter is stored as shown below. The leftmost bit will be F for negative values.

Leftmost 4 digits	Rightmost 4 digits	Up/Down Mode	Incrementing Mode
P1+1	P1	F0032767 to 00032767 (-32767)	00000000 to 00065535

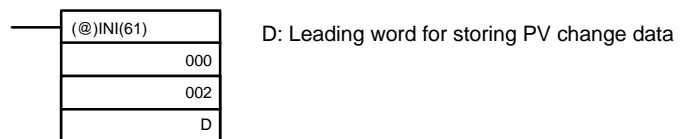
The PV is read when the PRV(62) instruction is actually executed.

**Changing the PV**

There are two ways to change the PV of high-speed counter. The first way is to reset it by using the reset methods. (In this case the PV is reset to 0.) The second way is to use the INI(61) instruction.

The method using the INI(61) instruction is explained here. For an explanation of the reset method, refer to the beginning of this description of high-speed counter.

Change the timer PV by using the INI(61) instruction as shown below.



Leftmost 4 digits	Rightmost 4 digits	Up/Down Mode	Incrementing Mode
D+1	D	F0032767 to 00032767	00000000 to 00065535

To specify a negative number in up/down mode, set F in the leftmost digit.

This example shows a program that uses the high-speed counter with single-phase inputs in the Incrementing Mode, making comparisons by means of the target matching method.

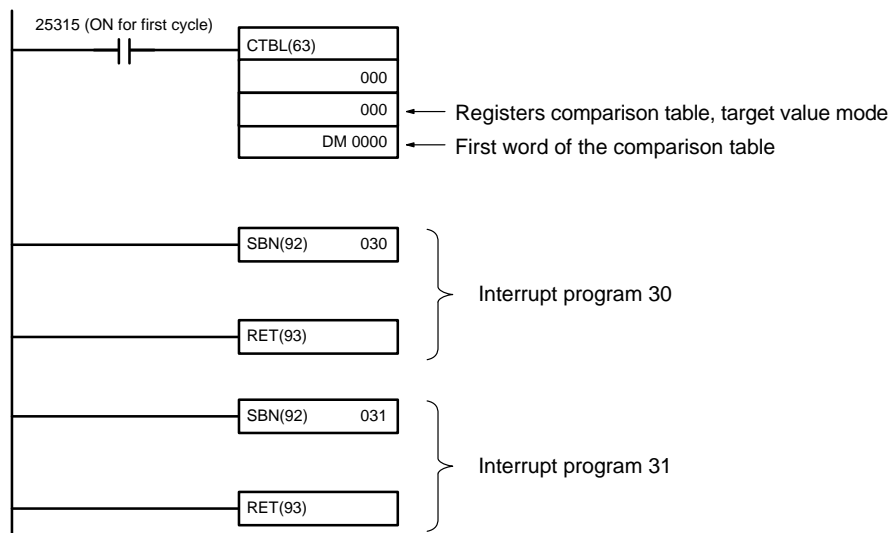
**Application Example  
(Incrementing Mode)**

The comparison conditions (target values and count directions) are stored in the comparison table with the subroutine numbers. Up to 16 target values can be stored. The corresponding subroutine is executed when the counter's PV matches the target value.

The following data is stored for the comparison table:

DM 0000	0002	Number of comparison conditions: 2
DM 0001	1000	Target value 1: 1000
DM 0002	0000	
DM 0003	0030	Comparison 1 interrupt subroutine no.: 30
DM 0004	2000	Target value 2: 2000
DM 0005	0000	
DM 0006	0031	Comparison 2 interrupt subroutine no.: 31

The following diagram shows the example ladder program. DM 6642 must be set to 01□4, where □ is the reset method which can be set to 0 or 1.



**Application Example (Up/Down Mode)**

This example shows a program that uses the high-speed counter with phase-difference inputs in the Up/Down Mode, making comparisons by means of the range comparison method.

The comparison conditions (upper/lower limits of the ranges) are stored in the comparison table with the subroutine numbers. Up to 8 separate ranges can be defined. The corresponding subroutine is executed when the counter's PV is within the range.

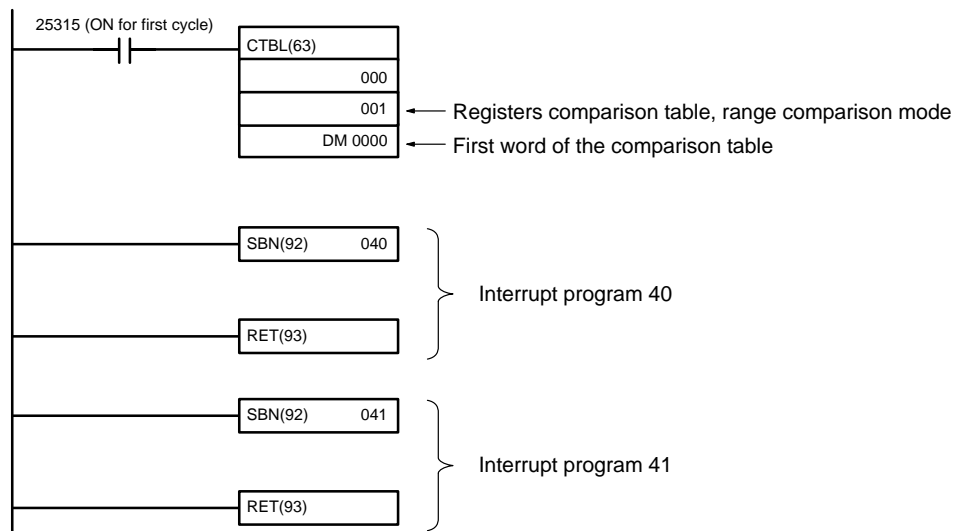
**Note** Always set 8 ranges. If fewer than 8 ranges are needed, set the remaining subroutine numbers to FFFF. A value of FFFF indicates that no subroutine is to be executed.

The following data is stored for the comparison table:

DM 0000	1500	
DM 0001	0000	Lower limit 1: 1,500 counts
DM 0002	3000	
DM 0003	0000	Upper limit 1: 3,000 counts
DM 0004	0040	Range 1 interrupt subroutine no.: 40
DM 0005	7500	
DM 0006	0000	Lower limit 2: 7,500 counts
DM 0007	0000	
DM 0008	0001	Upper limit 2: 10,000 counts
DM 0009	0041	Range 2 interrupt subroutine no.: 41
DM 0010	0000	
DM 0011	0000	

DM 0012	0000	
DM 0013	0000	
DM 0014	FFFF	Range 3 interrupt subroutine not executed
.	.	.
.	.	.
.	.	.
.	.	.
DM 0035	0000	
DM 0036	0000	
DM 0037	0000	
DM 0038	0000	
DM 0039	FFFF	Range 8 interrupt subroutine not executed

The following diagram shows the example ladder program. DM 6642 must be set to 01□0, where □ is the reset method which can be set to 0 or 1.



## 1-7 SRM1 Interrupt Functions

This section explains the settings and methods for using the SRM1 interrupt functions.

### 1-7-1 Types of Interrupts

The SRM1 has only one type of interrupt processing, as outlined below.

#### Interval Timer Interrupts

Interrupt processing is executed by an interval timer with a precision of 0.1 ms.

### 1-7-2 Interval Timer Interrupts

The SRM1 is equipped with one interval timer. When the interval timer times out, the main program is interrupted and the interrupt program is executed immediately, regardless of the point in the cycle.

There are two modes for interval timer operation, the One-shot Mode, in which only one interrupt will be executed when time expires, and the Scheduled Interrupt Mode in which the interrupt is repeated at a fixed interval.

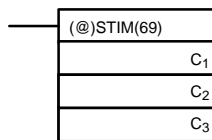
The interval timer's set value can be set anywhere from 0.5 to 319968 ms, in units of 0.1 ms.

**Operation**

Use the following instruction to activate and control the interval timer.

**Starting Up in One-Shot Mode**

Use the STIM(69) instruction to start the interval timer in the one-shot mode.



- C<sub>1</sub>: Interval timer, one-shot mode (000)  
 C<sub>2</sub>: Timer set value (first word address)  
 C<sub>3</sub>: Subroutine no. (4 digits BCD): 0000 to 0049

**1, 2, 3...**

1. When C<sub>2</sub> is entered as a word address:

C<sub>2</sub>: Decrementing counter set value (4 digits BCD): 0000 to 9999

C<sub>2</sub> + 1: Decrementing time interval (4 digits BCD; unit: 0.1 ms): 0005 to 0320 (0.5 ms to 32 ms)

Each time that the interval specified in word C<sub>2</sub> + 1 elapses, the decrementing counter will decrement the present value by one. When the PV reaches 0, the designated subroutine will be called just once and the timer will stop.

The time from when the STIM(69) instruction is executed until time elapses is calculated as follows:

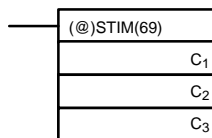
$$(\text{Content of } C_2) \times (\text{Content of } C_2 + 1) \times 0.1 \text{ ms} = (0.5 \text{ to } 319,968 \text{ ms})$$

2. When C<sub>2</sub> is entered as a constant:

The set value of the decrementing counter will equal the specified constant (in ms) and the decrementing time interval will be 10 (1 ms).

**Starting Up in Scheduled Interrupt Mode**

Use the STIM(69) instruction to start the interval timer in the scheduled interrupt mode.



- C<sub>1</sub>: Interval timer, scheduled interrupt mode (003)  
 C<sub>2</sub>: Timer set value (leading word no.)  
 C<sub>3</sub>: Subroutine no. (4 digits BCD): 0000 to 0049

**1, 2, 3...**

1. When C<sub>2</sub> is entered as a word address:

C<sub>2</sub>: Decrementing counter set value (4 digits BCD): 0000 to 9999

C<sub>2</sub> + 1: Decrementing time interval (4 digits BCD; unit: 0.1 ms): 0005 to 0320 (0.5 ms to 32 ms)

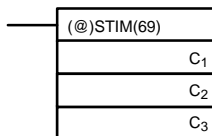
The meanings of the settings are the same as for the one-shot mode, but in the scheduled interrupt mode the timer PV will be reset to the set value and decrementing will begin again after the subroutine has been called. In the scheduled interrupt mode, interrupts will continue to be repeated at fixed intervals until the operation is stopped.

2. When C<sub>2</sub> is entered as a constant:

The settings are the same as for the one-shot mode, but interrupts will continue to be repeated at fixed intervals until the operation is stopped.

**Reading the Timer's Elapsed Time**

Use the STIM(69) instruction to read the timer's elapsed time.



- C<sub>1</sub>: Read elapsed time (006)  
 C<sub>2</sub>: Leading word of parameter 1  
 C<sub>3</sub>: Parameter 2

C<sub>2</sub>: Number of times the decrementing counter has been decremented (4 digits BCD)

C<sub>2</sub> + 1: Decrementing counter time interval (4 digits BCD; unit: 0.1 ms)

C<sub>3</sub>: Elapsed time from previous decrement (4 digits BCD; unit: 0.1 ms)



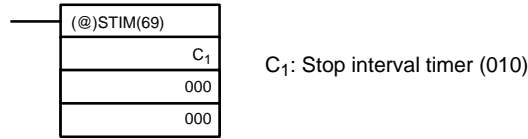
The time from when the interval timer is started until the execution of this instruction is calculated as follows:

$$\{(\text{Content of C2}) \times (\text{Content of C2}+1) + (\text{Content of C3})\} \times 0.1 \text{ ms}$$

If the specified interval timer is stopped, then "0000" will be stored.

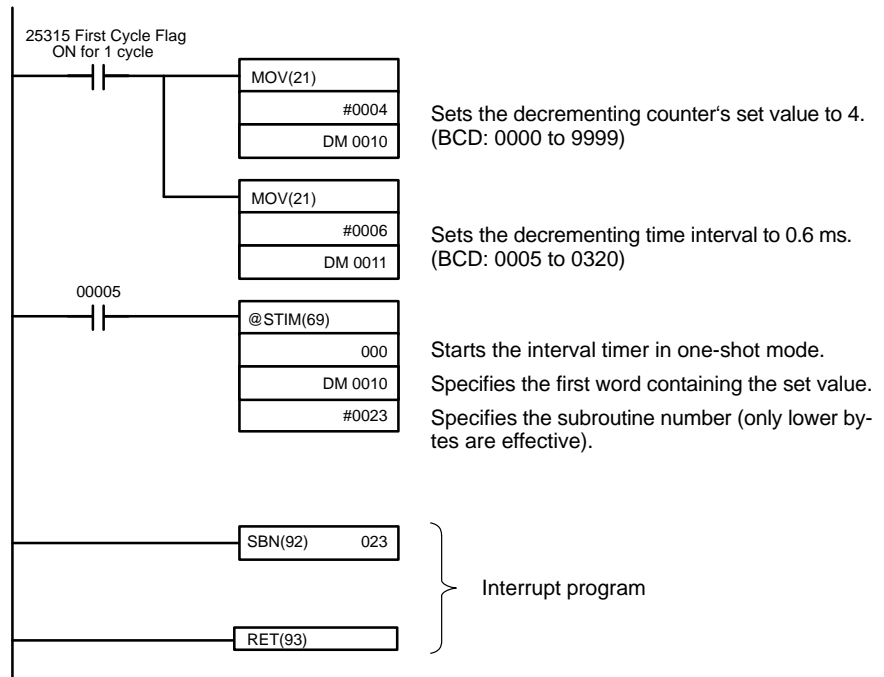
**Stopping the Timer**

Use the STIM(69) instruction to stop the interval timer. The interval timer will be stopped.



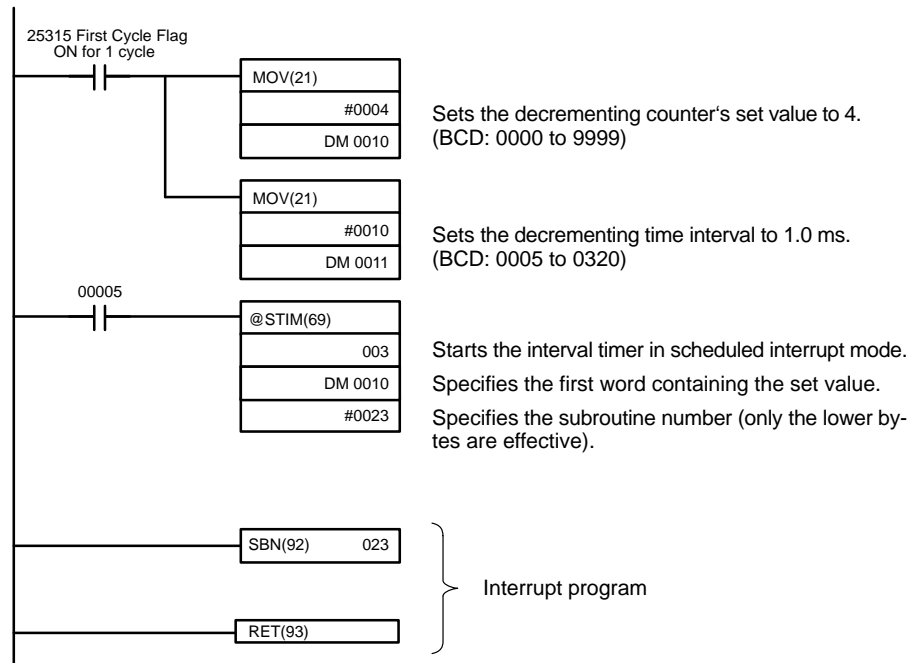
**Application Example  
(One-shot Mode)**

In this example, an interrupt is generated 2.4 ms (0.6 ms × 4) after input 00005 goes ON; the interrupt executes interrupt subroutine number 23.



**Application Example  
(Scheduled Interrupt Mode)**

In this example, an interrupt is generated every 4.0 ms (1.0 ms × 4) after input 00005 goes ON; the interrupts execute interrupt subroutine number 23.



## 1-8 CompoBus/S Distributed I/O Functions (SRM1 Only)

**No. of Connected Nodes**

A maximum of either 16 or 32 CompoBus/S nodes may be connected.

No. of nodes set	Communications response time
32	0.8 ms
16	0.5 ms

The maximum no. of nodes can be set from a Programming Device by making the following settings in DM 6603.

Word	Bit(s)	Function	Setting
DM 6603	00 to 03	Sets the maximum no. of CompoBus/S nodes to either 16 or 32. 00: 32 nodes 01: 16 nodes	00 or 01
	04 to 15	Not used.	00

**Note** When changes are made to these settings, always turn the power off and on again to make the new setting effective.

**Slave Interrupts**

Input bits in IR 000 to IR 007 and output bits in IR 010 to IR 017 are used as interrupts for CompoBus/S I/O Terminals. The CompoBus/S I/O Terminal interrupts (IN 0 to 15 and OUT 0 to 15) are allocated as indicated in the following table.

IN0 to IN15 are the node addresses for the Input Terminals and OUT0 to OUT15 are the node addresses for the Output Terminals.

Word		Bit															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Input	IR 000	IN1								IN0							
	IR 001	IN3								IN2							
	IR 002	IN5								IN4							
	IR 003	IN7								IN6							
	IR 004	IN9								IN8							
	IR 005	IN11								IN10							
	IR 006	IN13								IN12							
	IR 007	IN15								IN14							
Output	IR 010	OUT1								OUT0							
	IR 011	OUT3								OUT2							
	IR 012	OUT5								OUT4							
	IR 013	OUT7								OUT6							
	IR 014	OUT9								OUT8							
	IR 015	OUT11								OUT10							
	IR 016	OUT13								OUT12							
	IR 017	OUT15								OUT14							

- Note**
1. When the maximum number of CompoBus/S nodes is set to 16, IN8 to IN15 can be used as work bits.
  2. CompoBus/S Terminals with less than 8 points are allocated bit addresses from either 0 or 8.
  3. CompoBus/S Terminals with 16 points can be set for only even number addresses.

**Status Flags**

The communications status between CompoBus/S terminals is output through AR04 to AR07 Slave Add Flags and Slave Communications Error Flags.

Word	Uppermost bits: Slave Communications Error Flags								Lower Bits: Slave Add Flags							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AR04	OUT 7	OUT 6	OUT 5	OUT 4	OUT 3	OUT 2	OUT 1	OUT 0	OUT 7	OUT 6	OUT 5	OUT 4	OUT 3	OUT 2	OUT 1	OUT 0
AR05	IN7	IN6	IN5	IN4	IN3	IN2	IN1	IN0	IN7	IN6	IN5	IN4	IN3	IN2	IN1	IN0
AR06	OUT 15	OUT 14	OUT 13	OUT 12	OUT 11	OUT 10	OUT 9	OUT 8	OUT 15	OUT 14	OUT 13	OUT 12	OUT 11	OUT 10	OUT 9	OUT 8
AR07	IN15	IN14	IN13	IN12	IN11	IN10	IN9	IN8	IN15	IN14	IN13	IN12	IN11	IN10	IN9	IN8

- Note**
1. IN0 to IN15 are the input terminals and OUT0 to OUT15 are the output terminals.
  2. When the maximum number of CompoBus/S units is set to 16, IN8 to IN15 and OUT8 to OUT15 cannot be used.
  3. The Slave Add Flag turns ON when a slave joins the communications. When the power to the CPU Unit is turned OFF and ON again all bits will turn OFF.
  4. The Slave Communications Error Flag turns ON when a slave participating in the network is separated from the network. The bit will turn OFF when the slave re-enters the network.

## 1-9 Communications Functions

**CQM1 Communications**

The following types of communications can be executed through the ports of the CQM1.

- Host link communications with a host computer
- RS-232C communications with a computer or other device
- One-to-one link communications with another CQM1

**Note** These types of communications cannot be carried out with the CQM1-CPU11-E, which is equipped with only a peripheral port.

This section explains the required PC Setup and methods for using these types of communications.

**CPM1/CPM1A Communications**

The CPM1/CPM1A can execute a variety of communications through its peripheral port via an RS-232C Adapter or an RS-422 Adapter.

**Host Link Communications**

The CPM1/CPM1A PCs are compatible with the Host Link System, which allows up to 32 PCs to be controlled from a host computer. An RS-232C Adapter is used for 1-to-1 communications and an RS-422 Adapter and B500-AL004 Link Adapter are used for 1-to-n communications.

A CPM1/CPM1A equipped with an RS-232C Adapter can also communicate with a Programmable Terminal using host link commands.

Refer to *1-9-4 CPM1/CPM1A Host Link Communications* in this manual and *1-2-2 Host Link Communications* in the *CPM1 Operation Manual* or *1-2-2 Host Link Communications* in the *CPM1A Operation Manual* for more details.

**1-to-1 Link**

A data link can be created with a data area in another CPM1, CPM1A, CQM1, or C200HS PC. An RS-232C Adapter is used to make the 1-to-1 connection.

Refer to *1-9-8 CPM1/CPM1A One-to-one Link Communications* in this manual and *1-2-3 1-to-1 Communications Links* in the *CPM1 Operation Manual* or *1-2-3 1-to-1 Communications Links* in the *CPM1A Operation Manual* for more details.

**NT Link**

Using the NT link, the CPM1/CPM1A PC can be connected to the Programmable Terminal (NT Link Interface) through an RS-232C Adapter.

Refer to *1-9-9 CPM1/CPM1A NT Link Communications* in this manual and *1-2-4 NT Link Communications* in the *CPM1 Operation Manual* or *NT Link Communications* in the *CPM1A Operation Manual* for more details.

**SRM1 Communications**

The following types of communications can be executed through the ports of the SRM1.

- Host link communications with a host computer
- RS-232C communications with a computer or other device
- One-to-one link communications with another SRM1
- NT Link communications with Programmable Terminals

**Note** NT Link communications are not possible with the SRM1-C01, which is equipped with only a peripheral port. The SRM1-C01 may be connected to a PT through an RS-232C Adapter in Host Link mode.

**1-9-1 CQM1 PC Setup**

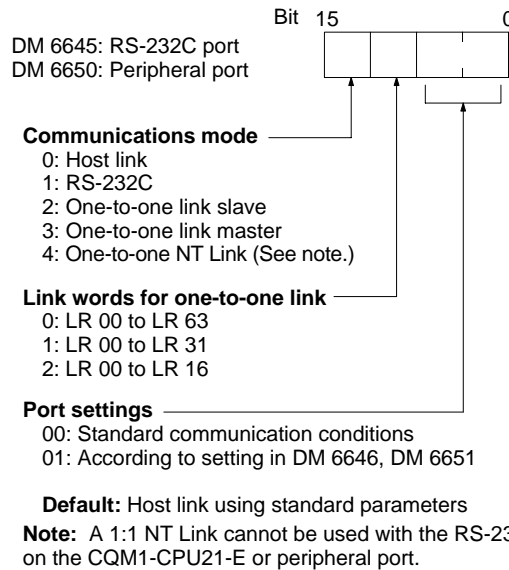
The PC Setup parameters in DM 6645 through DM 6654 are used to set parameters for the communications ports. The parameters for the RS-232C port in DM 6645 through DM 6649 can be set from menu operations using the SSS.

If pin 5 on the CQM1's DIP switch is turned ON, the PC Setup communications parameters will be ignored and the following parameters will be used.

Mode:	Host link
Node number:	00
Start bits:	1 bit
Data length:	7 bits
Stop bits:	2 bit
Parity:	Even
Baud rate:	9,600 bps
Transmission delay:	None

**Note** The above setting apply to CPU Units manufactured from July 1995 (lot number \*\*75 for July 1995). For CPU Units manufactured before July 1995 (lot number \*\*65 for June 1995), only 1 stop bit will be set and the baud rate will be 2,400 bps.

The settings in DM 6645 and DM 6650 determine the main communications parameters, as shown in the following diagram.



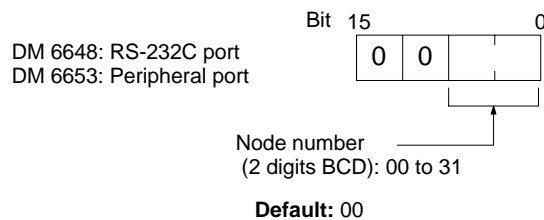
**One-to-one Links**

To use a 1:1 link, the only settings necessary are the communications mode and the link words. Set the communications mode for one of the PCs to the 1:1 master and the other to the 1:1 slave, and then set the link words in the PC designated as the master. Bits 08 to 11 are valid only for the master for link one-to-one.

**Note** One-to-one link communications are possible for the RS-232C port only. This setting is not possible for the peripheral port. A 1:1 NT Link cannot be used with the CQM1-CPU21-E.

**Host Link Node Number**

A node number must be set for host link communications to differentiate between nodes when multiple nodes are participating in communications. This setting is required only for host link communications. To use host link communications, the host link must be specified as the communications mode and the communications parameters must be set (see following section).

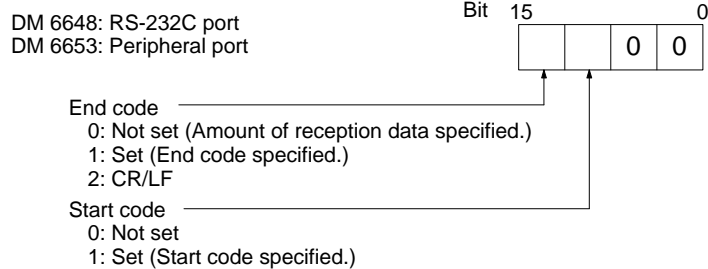


Set the node number to 00 unless multiple nodes are connected in a network.

**RS-232C Start and End Codes and Data Received**

Start and end codes or the amount of data to be received can be set as shown in the following diagrams if required for RS-232C communications. This setting is required only for RS-232C communications. To use RS-232C communications, the RS-232C must be specified as the communications mode and the communications parameters must be set (see next section).

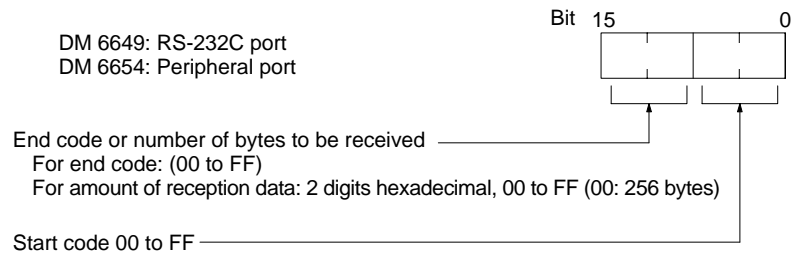
### Enabling Start and End Codes



**Defaults:** No start code; data reception complete at 256 bytes.

Specify whether or not a start code is to be set at the beginning of the data, and whether or not an end code is to be set at the end. Instead of setting the end code, it is possible to specify the number of bytes to be received before the reception operation is completed. Both the codes and the number of bytes of data to be received are set in DM 6649 or DM 6654.

### Setting the Start Code, End Code, and Amount of Reception Data



**Defaults:** No start code; data reception complete at 256 bytes.

## Host Link and RS-232C Communications Parameters

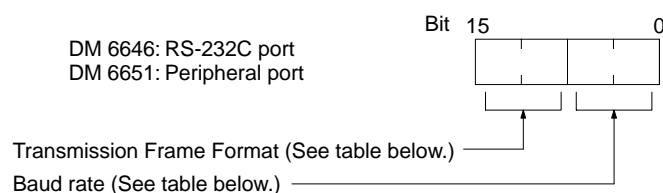
Select either host link or RS-232C communications and then set the communications parameters as described next. Match the communications conditions to the settings at the device with which communications are being carried out.

### Standard Communications

If the following settings are satisfactory for these communications conditions, then set the two rightmost digits to 00. The settings in DM 6646 and DM 6651 will be ignored for this setting.

- Start bits: 1 bit
- Data length: 7 bits
- Stop bits: 2 bits
- Parity: Even
- Baud rate: 9,600 bps

### Setting Communications Conditions



**Default:** Standard communication conditions.

**Transmission Frame Format**

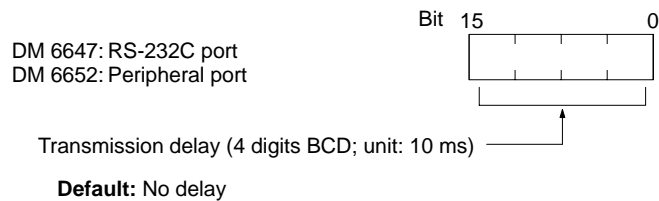
Setting	Stop bits	Data length	Stop bits	Parity
00	1	7	1	Even
01	1	7	1	Odd
02	1	7	1	None
03	1	7	2	Even
04	1	7	2	Odd
05	1	7	2	None
06	1	8	1	Even
07	1	8	1	Odd
08	1	8	1	None
09	1	8	2	Even
10	1	8	2	Odd
11	1	8	2	None

**Baud Rate**

Setting	Baud rate
00	1,200 bps
01	2,400 bps
02	4,800 bps
03	9,600 bps
04	19,200 bps

**Transmission Delay Time**

Depending on the devices connected to the RS-232C port, it may be necessary to allow time for transmission. When that is the case, set the transmission delay to regulate the amount of time allowed.



**1-9-2 Wiring Ports**

Refer to the *CQM1 Operation Manual*, *CPM1 Operation Manual*, *CPM1A Operation Manual* or *SRM1 Master Control Units Operation Manual* for information on wiring the communications ports.

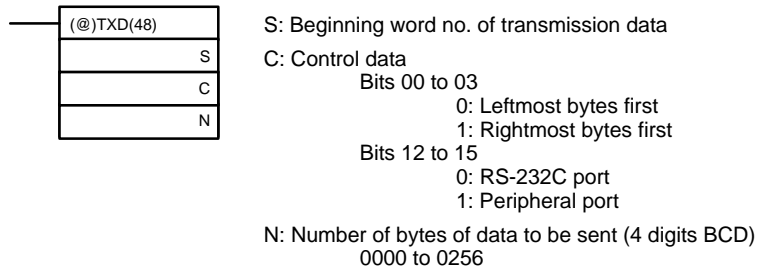
**1-9-3 CQM1 Host Link Communications**

Host link communications were developed by OMRON for the purpose of connecting PCs and one or more host computers by RS-232C cable, and controlling PC communications from the host computer. Normally the host computer issues a command to a PC, and the PC automatically sends back a response. Thus the communications are carried out without the PCs being actively involved. The PCs also have the ability to initiate data transmissions when direct involvement is necessary.

In general, there are two means for implementing host link communications. One is based on C-mode commands, and the other on FINS (CV-mode) commands. The CQM1 supports C-mode commands only. For details on host link communications, refer to *Section 6 Host Link Commands*.

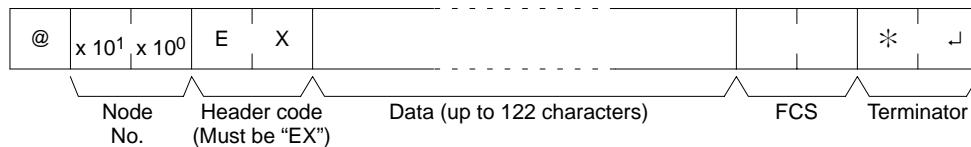
**Communications Procedure** This section explains how to use the host link to execute data transmissions from the CQM1. Using this method enables automatic data transmission from the CQM1 when data is changed, and thus simplifies the communications process by eliminating the need for constant monitoring by the computer.

- 1, 2, 3... 1. Check to see that AR 0805 (RS-232C Port Transmit Ready Flag) is ON.
2. Use the TXD(48) instruction to transmit the data.



From the time this instruction is executed until the data transmission is complete, AR 0805 (or AR 0813 for the peripheral port) will remain OFF. It will turn ON again upon completion of the data transmission. The TXD(48) instruction does not provide for a response, so in order to receive confirmation that the computer has received the data, the computer's program must be written so that it gives notification when data is written from the CQM1.

The transmission data frame is as shown below for data transmitted in the Host Link Mode by means of the TXD(48) instruction.

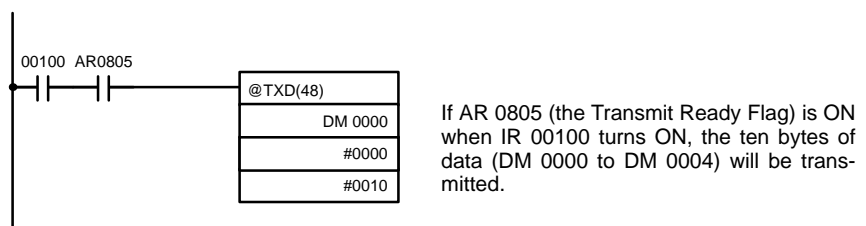


To reset the RS-232C port (i.e., to restore the initial status), turn ON SR 25209. To reset the peripheral port, turn ON SR 25208. These bits will turn OFF automatically after the reset.

If the TXD(48) instruction is executed while the CQM1 is in the middle of responding to a command from the computer, the response transmission will first be completed before the transmission is executed according to the TXD(48) instruction. In all other cases, data transmission based on a TXD(48) instruction will be given first priority.

**Application Example**

This example shows a program for using the RS-232C port in the Host Link Mode to transmit 10 bytes of data (DM 0000 to DM 0004) to the computer. The default values are assumed for all the PC Setup (i.e., the RS-232C port is used in Host Link Mode, the node number is 00, and the standard communications conditions are used.) From DM 0000 to DM 0004, "1234" is stored in every word. From the computer, execute a program to receive CQM1 data with the standard communications conditions.





The following type of program must be prepared in the host computer to receive the data. This program allows the computer to read and display the data received from the PC while a host link read command is being executed to read data from the PC.

```

10 'CQM1 SAMPLE PROGRAM FOR EXCEPTION
20 CLOSE 1
30 CLS
40 OPEN "COM:E73" AS #1
50 *KEYIN
60 INPUT "DATA -----",S$
70 IF S$=" " THEN GOTO 190
80 PRINT "SEND DATA = ";S$
90 ST$=S$
100 INPUT "SEND OK? Y or N?=",B$
110 IF B$="Y" THEN GOTO 130 ELSE GOTO *KEYIN
120 S$=ST$
130 PRINT #1,S$                                ' Sends command to PC
140 INPUT #1,R$                                ' Receives response from PC
150 PRINT "RCV DATA = ";R$
160 IF MID$(R$,4,2)="EX" THEN GOTO 210 ' Identifies command from PC
170 IF RIGHT$(R$,1)<>"*" THEN S$=" ":GOTO 130
180 GOTO *KEYIN
190 CLOSE 1
200 END
210 PRINT "EXCEPTION!! DATA"
220 GOTO 140
    
```

The data received by the computer will be as shown below. (FCS is "59.")  
 "@00EX1234123412341234123459\*CR"

### 1-9-4 CPM1/CPM1A Host Link Communications

Host link communications were developed by OMRON for the purpose of connecting PCs and one or more host computers by RS-232C cable, and controlling PC communications from the host computer. Normally the host computer issues a command to a PC, and the PC automatically sends back a response. Thus the communications are carried out without the PCs being actively involved. The PCs also have the ability to initiate data transmissions when direct involvement is necessary.

In general, there are two means for implementing host link communications. One is based on C-mode commands, and the other on FINS (CV-mode) commands. The CPM1/CPM1A supports C-mode commands only. For details on host link communications, refer to *Section 6 Host Link Commands*.

#### PC Setup Settings

The CPM1/CPM1A's peripheral port settings must be set properly in order to use the host link communications, as shown in the following table.

Word	Bit	Function	Setting
DM 6650	00 to 07	Port settings <sup>1</sup> 00: Standard (1 start bit, 7-bit data, even parity, 2 stop bits, 9,600 bps) 01: Settings in DM 6651	00
	08 to 11	Link area for one-to-one PC link via peripheral port 0: LR 00 to LR 15	0 (Any value is OK)
	12 to 15	Communications mode <sup>1</sup> 0: Host link; 2: One-to-one PC link (slave); 3: One-to-one PC link (master); 4: NT link	0

Word	Bit	Function	Setting																																																																
DM 6651	00 to 07	Baud rate <sup>1</sup> 00: 1.2K, 01: 2.4K, 02: 4.8K, 03: 9.6K, 04: 19.2K	00 (Any value is OK)																																																																
	08 to 15	Frame format <sup>1</sup> <table border="0" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;"></th> <th style="width: 15%;">Start</th> <th style="width: 15%;">Length</th> <th style="width: 15%;">Stop</th> <th style="width: 15%;">Parity</th> </tr> </thead> <tbody> <tr><td>00:</td><td>1 bit</td><td>7 bits</td><td>1 bit</td><td>Even</td></tr> <tr><td>01:</td><td>1 bit</td><td>7 bits</td><td>1 bit</td><td>Odd</td></tr> <tr><td>02:</td><td>1 bit</td><td>7 bits</td><td>1 bit</td><td>None</td></tr> <tr><td>03:</td><td>1 bit</td><td>7 bits</td><td>2 bit</td><td>Even</td></tr> <tr><td>04:</td><td>1 bit</td><td>7 bits</td><td>2 bit</td><td>Odd</td></tr> <tr><td>05:</td><td>1 bit</td><td>7 bits</td><td>2 bit</td><td>None</td></tr> <tr><td>06:</td><td>1 bit</td><td>8 bits</td><td>1 bit</td><td>Even</td></tr> <tr><td>07:</td><td>1 bit</td><td>8 bits</td><td>1 bit</td><td>Odd</td></tr> <tr><td>08:</td><td>1 bit</td><td>8 bits</td><td>1 bit</td><td>None</td></tr> <tr><td>09:</td><td>1 bit</td><td>8 bits</td><td>2 bit</td><td>Even</td></tr> <tr><td>10:</td><td>1 bit</td><td>8 bits</td><td>2 bit</td><td>Odd</td></tr> <tr><td>11:</td><td>1 bit</td><td>8 bits</td><td>2 bit</td><td>None</td></tr> </tbody> </table>		Start	Length	Stop	Parity	00:	1 bit	7 bits	1 bit	Even	01:	1 bit	7 bits	1 bit	Odd	02:	1 bit	7 bits	1 bit	None	03:	1 bit	7 bits	2 bit	Even	04:	1 bit	7 bits	2 bit	Odd	05:	1 bit	7 bits	2 bit	None	06:	1 bit	8 bits	1 bit	Even	07:	1 bit	8 bits	1 bit	Odd	08:	1 bit	8 bits	1 bit	None	09:	1 bit	8 bits	2 bit	Even	10:	1 bit	8 bits	2 bit	Odd	11:	1 bit	8 bits	2 bit	None
	Start	Length	Stop	Parity																																																															
00:	1 bit	7 bits	1 bit	Even																																																															
01:	1 bit	7 bits	1 bit	Odd																																																															
02:	1 bit	7 bits	1 bit	None																																																															
03:	1 bit	7 bits	2 bit	Even																																																															
04:	1 bit	7 bits	2 bit	Odd																																																															
05:	1 bit	7 bits	2 bit	None																																																															
06:	1 bit	8 bits	1 bit	Even																																																															
07:	1 bit	8 bits	1 bit	Odd																																																															
08:	1 bit	8 bits	1 bit	None																																																															
09:	1 bit	8 bits	2 bit	Even																																																															
10:	1 bit	8 bits	2 bit	Odd																																																															
11:	1 bit	8 bits	2 bit	None																																																															
DM 6652	00 to 15	Transmission delay (Host Link) <sup>1</sup> 0000 to 9999: In ms.	0000																																																																
DM 6653	00 to 07	Node number (Host link) <sup>1</sup> 00 to 31 (BCD)	00 to 31																																																																
	08 to 15	Not used.	00 (Any value is OK)																																																																

- Note**
1. If an improper setting is used, a non-fatal error will occur, AR 1302 will be turned ON, and the default setting (0, 00, or 0000) will be used.
  2. For information on the host link settings for another OMRON PC, refer to that PC's Operation Manual.
  3. If an out-of-range value is set, the following communications conditions will result. In that case, reset the value so that it is within the permissible range.
 

Communications mode:	Host Link
Communications format:	Standard settings (1 start bit, 7-bit data; even parity, 2 stop bits, 9,600 bps)
Transmission delay:	No
Node number:	00

**Example Program**

This example shows a BASIC program that reads the status of the CPM1's inputs in IR 000. For more details, refer to *Section 6 Host Link Commands*.

An FCS (frame check sequence) check isn't performed on the received response data in this program. Be sure that the host computer's RS-232C port is configured correctly before executing the program.

```

1010 'CPM1 SAMPLE PROGRAM
1020 'SET THE COMMAND DATA
1030 S$="@00RR00000001"
1040 FCS=0
1050 FOR I=1 TO LEN(S$)
1060 FCS=FCS XOR ASC(MID$(S$,I,1))
1070 NEXT I
1080 FCS$=(FCS):IF LEN(FCS$)=1 THEN FCS$="0"+FCS$
1090 CLOSE 1
1100 CLS
1110 PRINT "SENDING COMMAND"
1120 OPEN "COM:E73" AS #1
1130 PRINT #1,S$ + FCS + CHR$(13);
1140 CLS
1150 PRINT "RECEIVING RESPONSE DATA"
1160 LINE INPUT #1,A$
1170 PRINT A$
1180 END
    
```

**1-9-5 SRM1 Host Link Communications**

Host link communications were developed by OMRON for the purpose of connecting PCs and one or more host computers by RS-232C cable, and controlling PC communications from a host computer. Normally the host computer issues a command to a PC, and the PC automatically sends back a response. Thus the communications are carried out without the PCs being actively involved. The PCs also have the ability to initiate data transmissions when direct involvement is necessary.

In general, there are two means for implementing host link communications. One is based on C-mode commands, and the other on FINS (CV-mode) commands. The SRM1 supports C-mode commands only. For details on host link communications, refer to *Section 6 Host Link Commands*.

**PC Setup Settings**

The SRM1's peripheral port and RS-232C port settings must be set properly in order to use the host link communications, as shown in the following table.

Word	Bit	Function	Setting
<b>Peripheral Port Settings</b>			
The following settings are effective after transfer to the PC.			
DM 6650	00 to 03	Port settings 0: Standard (1 start bit, 7-bit data, even parity, 2 stop bits, 9,600 bps) 1: Settings in DM 6651  (Other settings will cause a non-fatal error, the default setting (0) will be used, and AR 1302 will turn ON.)	To match host parameters
	04 to 07	Not used.	0
	08 to 11	Not used.	0
	12 to 15	Communications mode 0: Host link; 1: No protocol  (Other settings will cause a non-fatal error, the default setting (0) will be used, and AR 1302 will turn ON.)	0: Host link

Word	Bit	Function	Setting																																																																
DM 6651	00 to 07	Baud rate 00: 1.2K, 01: 2.4K, 02: 4.8K, 03: 9.6K, 04: 19.2K	To match host parameters																																																																
	08 to 15	Frame format <table border="0"> <tr> <td></td> <td>Start</td> <td>Length</td> <td>Stop</td> <td>Parity</td> </tr> <tr> <td>00:</td> <td>1 bit</td> <td>7 bits</td> <td>1 bit</td> <td>Even</td> </tr> <tr> <td>01:</td> <td>1 bit</td> <td>7 bits</td> <td>1 bit</td> <td>Odd</td> </tr> <tr> <td>02:</td> <td>1 bit</td> <td>7 bits</td> <td>1 bit</td> <td>None</td> </tr> <tr> <td>03:</td> <td>1 bit</td> <td>7 bits</td> <td>2 bit</td> <td>Even</td> </tr> <tr> <td>04:</td> <td>1 bit</td> <td>7 bits</td> <td>2 bit</td> <td>Odd</td> </tr> <tr> <td>05:</td> <td>1 bit</td> <td>7 bits</td> <td>2 bit</td> <td>None</td> </tr> <tr> <td>06:</td> <td>1 bit</td> <td>8 bits</td> <td>1 bit</td> <td>Even</td> </tr> <tr> <td>07:</td> <td>1 bit</td> <td>8 bits</td> <td>1 bit</td> <td>Odd</td> </tr> <tr> <td>08:</td> <td>1 bit</td> <td>8 bits</td> <td>1 bit</td> <td>None</td> </tr> <tr> <td>09:</td> <td>1 bit</td> <td>8 bits</td> <td>2 bit</td> <td>Even</td> </tr> <tr> <td>10:</td> <td>1 bit</td> <td>8 bits</td> <td>2 bit</td> <td>Odd</td> </tr> <tr> <td>11:</td> <td>1 bit</td> <td>8 bits</td> <td>2 bit</td> <td>None</td> </tr> </table> (Other settings will cause a non-fatal error, the default setting (00) will be used, and AR 1302 will turn ON.)		Start	Length	Stop	Parity	00:	1 bit	7 bits	1 bit	Even	01:	1 bit	7 bits	1 bit	Odd	02:	1 bit	7 bits	1 bit	None	03:	1 bit	7 bits	2 bit	Even	04:	1 bit	7 bits	2 bit	Odd	05:	1 bit	7 bits	2 bit	None	06:	1 bit	8 bits	1 bit	Even	07:	1 bit	8 bits	1 bit	Odd	08:	1 bit	8 bits	1 bit	None	09:	1 bit	8 bits	2 bit	Even	10:	1 bit	8 bits	2 bit	Odd	11:	1 bit	8 bits	2 bit	None
	Start	Length	Stop	Parity																																																															
00:	1 bit	7 bits	1 bit	Even																																																															
01:	1 bit	7 bits	1 bit	Odd																																																															
02:	1 bit	7 bits	1 bit	None																																																															
03:	1 bit	7 bits	2 bit	Even																																																															
04:	1 bit	7 bits	2 bit	Odd																																																															
05:	1 bit	7 bits	2 bit	None																																																															
06:	1 bit	8 bits	1 bit	Even																																																															
07:	1 bit	8 bits	1 bit	Odd																																																															
08:	1 bit	8 bits	1 bit	None																																																															
09:	1 bit	8 bits	2 bit	Even																																																															
10:	1 bit	8 bits	2 bit	Odd																																																															
11:	1 bit	8 bits	2 bit	None																																																															
DM 6652	00 to 15	Transmission delay (Host Link) 0000 to 9999 (BCD): Set in units of 10 ms.  (Other settings will cause a non-fatal error, the default setting (0000) will be used, and AR 1302 will turn ON.)	To match host parameters																																																																
DM 6653	00 to 07	Node number (Host link) 00 to 31 (BCD)  (Other settings will cause a non-fatal error, the default setting (0000) will be used, and AR 1302 will turn ON.)	00 to 31																																																																
	08 to 11	Start code enable (RS-232C, effective when bits 12 to 15 of DM 6650 are set to 1.) 0: Disable 1: Set	Any																																																																
	12 to 15	End code enable (RS-232C, effective when bits 12 to 15 of DM 6650 are set to 1.) 0: Disable (number of bytes received) 1: Set (specified end code) 2: CR, LF	Any																																																																
DM 6654	00 to 07	Start code (effective when bits 08 to 11 of DM6650 are set to 1.) 00: 256 bytes 01 to FF: 1 to 255 bytes	Any																																																																
	08 to 15	End code (no protocol)  When bits 12 to 15 of DM6653 are set to 0: 00: 256 bytes 01 to FF: 1 to 255 bytes  When bits 12 to 15 of DM6653 are set to 1: Setting: 00 to FF (binary)	Any																																																																

**Note** If an out-of-range value is set, the following communications conditions will result. In that case, reset the value so that it is within the permissible range.

Communications mode: Host Link  
 Communications format: Standard settings  
 (1 start bit, 7-bit data; even parity, 2 stop bits, 9,600 bps)  
 Transmission delay: No  
 Node number: 00

Word	Bit	Function	Setting
<b>RS-232C Port Settings</b>			
The following settings are effective after transfer to the PC.			
DM 6645	00 to 03	Port settings 0: Standard (1 start bit, 7-bit data, even parity, 2 stop bits, 9,600 bps) 1: Settings in DM 6646	To match host parameters

Word	Bit	Function	Setting																																																																
	04 to 07	CTS control settings 0: Disable; 1: Set	0																																																																
	08 to 11	Link words for 1:1 link 0: LR 00 to LR 15; Other: Not effective																																																																	
	12 to 15	Communications mode 0: Host link; 1: RS-232C (no protocol); 2: 1:1 data link slave; 3: 1:1 data link master; 4: NT Link	0: Host link																																																																
DM 6646	00 to 07	Baud rate 00: 1.2K, 01: 2.4K, 02: 4.8K, 03: 9.6K, 04: 19.2K	To match host parameters																																																																
	08 to 15	Frame format <table border="0" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;"></th> <th style="text-align: left;">Start</th> <th style="text-align: left;">Length</th> <th style="text-align: left;">Stop</th> <th style="text-align: left;">Parity</th> </tr> </thead> <tbody> <tr> <td>00:</td> <td>1 bit</td> <td>7 bits</td> <td>1 bit</td> <td>Even</td> </tr> <tr> <td>01:</td> <td>1 bit</td> <td>7 bits</td> <td>1 bit</td> <td>Odd</td> </tr> <tr> <td>02:</td> <td>1 bit</td> <td>7 bits</td> <td>1 bit</td> <td>None</td> </tr> <tr> <td>03:</td> <td>1 bit</td> <td>7 bits</td> <td>2 bit</td> <td>Even</td> </tr> <tr> <td>04:</td> <td>1 bit</td> <td>7 bits</td> <td>2 bit</td> <td>Odd</td> </tr> <tr> <td>05:</td> <td>1 bit</td> <td>7 bits</td> <td>2 bit</td> <td>None</td> </tr> <tr> <td>06:</td> <td>1 bit</td> <td>8 bits</td> <td>1 bit</td> <td>Even</td> </tr> <tr> <td>07:</td> <td>1 bit</td> <td>8 bits</td> <td>1 bit</td> <td>Odd</td> </tr> <tr> <td>08:</td> <td>1 bit</td> <td>8 bits</td> <td>1 bit</td> <td>None</td> </tr> <tr> <td>09:</td> <td>1 bit</td> <td>8 bits</td> <td>2 bit</td> <td>Even</td> </tr> <tr> <td>10:</td> <td>1 bit</td> <td>8 bits</td> <td>2 bit</td> <td>Odd</td> </tr> <tr> <td>11:</td> <td>1 bit</td> <td>8 bits</td> <td>2 bit</td> <td>None</td> </tr> </tbody> </table>		Start	Length	Stop	Parity	00:	1 bit	7 bits	1 bit	Even	01:	1 bit	7 bits	1 bit	Odd	02:	1 bit	7 bits	1 bit	None	03:	1 bit	7 bits	2 bit	Even	04:	1 bit	7 bits	2 bit	Odd	05:	1 bit	7 bits	2 bit	None	06:	1 bit	8 bits	1 bit	Even	07:	1 bit	8 bits	1 bit	Odd	08:	1 bit	8 bits	1 bit	None	09:	1 bit	8 bits	2 bit	Even	10:	1 bit	8 bits	2 bit	Odd	11:	1 bit	8 bits	2 bit	None
	Start	Length	Stop	Parity																																																															
00:	1 bit	7 bits	1 bit	Even																																																															
01:	1 bit	7 bits	1 bit	Odd																																																															
02:	1 bit	7 bits	1 bit	None																																																															
03:	1 bit	7 bits	2 bit	Even																																																															
04:	1 bit	7 bits	2 bit	Odd																																																															
05:	1 bit	7 bits	2 bit	None																																																															
06:	1 bit	8 bits	1 bit	Even																																																															
07:	1 bit	8 bits	1 bit	Odd																																																															
08:	1 bit	8 bits	1 bit	None																																																															
09:	1 bit	8 bits	2 bit	Even																																																															
10:	1 bit	8 bits	2 bit	Odd																																																															
11:	1 bit	8 bits	2 bit	None																																																															
DM 6647	00 to 15	Transmission delay (Host Link) 0000 to 9999 (BCD): Set in units of 10 ms, e.g., setting of 0001 equals 10 ms	To match host parameters																																																																
DM 6648	00 to 07	Node number (Host link, effective when bits 12 to 15 of DM 6645 are set to 0.) 00 to 31 (BCD)	00 to 31																																																																
	08 to 11	Start code enable (RS-232C, effective when bits 12 to 15 of DM 6645 are set to 1.) 0: Disable; 1: Set	Any																																																																
	12 to 15	End code enable (RS-232C, effective when bits 12 to 15 of DM 6645 are set to 1.) 0: Disable (number of bytes received) 1: Set (specified end code) 2: CR, LF	Any																																																																
DM 6649	00 to 07	Start code (RS-232C) 00: 256 bytes 01 to FF: 1 to 255 bytes	Any																																																																
	08 to 15	End code enable (RS-232C) 00 to FF (BIN)	Any																																																																

**Note** If an out-of-range value is set, the following communications conditions will result. In that case, reset the value so that it is within the permissible range.

Communications mode: Host Link  
 Communications format: Standard settings  
 (1 start bit, 7-bit data; even parity, 2 stop bits, 9,600 bps)  
 Transmission delay: No  
 Node number: 00

**Example Program**

This example shows a BASIC program that reads the status of the SRM1's inputs in IR 000. For more details, refer to *Section 6 Host Link Commands*.

An FCS (frame check sequence) check isn't performed on the received response data in this program. Be sure that the host computer's RS-232C port is configured correctly before executing the program.

```

1000 '-----
1010 'SRM1 Sample Program for PC-9801 N88-BASIC
1020 '
1050 '-----
1060 '----Set value RS-232C SPEED:9600BPS,PAR-
ITY:EVEN,DATA:7,STOP:2 -
1070 OPEN "COM:E73" AS #1
1080 *REPEAT
1090 '----Transmission data input -----
1100 INPUT "send data:",SEND$
1110 '----FCS Calculation -----
1120 FCS=0
1130 FOR IFCS=1 TO LEN(SEND$)
1140 FCS=FCS XOR ASC(MID$(SEND$,IFCS,1))
1150 NEXT
1160 FCS$=RIGHT$("0"+HEX$(FCS),2)
1170 '----Communications execute-----
1180 ZZZ$=SEND$+FCS$+"*" +CHR$(13)
1190 PRINT #1,ZZZ$;
1200 '----Response check -----
1210 RECCNT=0:TMP$=""
1220 *DRECLOOP
1230 IF LOC(1)<>0 THEN *DRECL
1240 RECCNT=RECCNT+1
1250 IF RECCNT=5000 THEN *DRECERR ELSE *DRECLOOP
1260 *DRECL
1270 TMP$=TMP$+INPUT$(LOC(1),#1)
1280 IF RIGHT$(TMP$,1)=CHR$(13) THEN *DRECEND ELSE
RECCNT=0:GOTO *DRECLOOP
1290 *DRECERR
1300 TMP$="No response!!"+CHR$(13)
1310 *DRECEND
1320 RECV$=TMP$
1330 PRINT "receive data: ";RECV$
1340 '----Go to transmission data input -----
1350 GOTO *REPEAT
1360 '----Processing complete -----
1370 CLOSE #1
1380 END

```

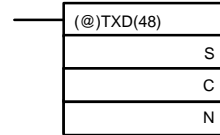
**1-9-6 RS-232C Communications (CQM1/SRM1 Only)**

This section explains RS-232C communications. By using RS-232C communications, the data can be printed out by a printer or read by a bar code reader. Handshaking is not supported for RS-232C communications.

**Communications Procedure****Transmissions**

- 1, 2, 3...** 1. Check to see that AR 0805 (the RS-232C Port Transmit Ready Flag) has turned ON.

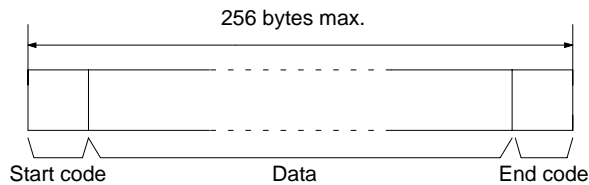
2. Use the TXD(48) instruction to transmit the data.



S: Leading word no. of data to be transmitted  
 C: Control data  
 N: Number of bytes to be transmitted (4 digits BCD), 0000 to 0256

From the time this instruction is executed until the data transmission is complete, AR 0805 ( or AR0813 for the peripheral port) will remain OFF. (It will turn ON again upon completion of the data transmission.)

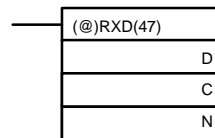
Start and end codes are not included when the number of bytes to be transmitted is specified. The largest transmission that can be sent with or without start and end codes in 256 bytes, N will be between 254 and 256 depending on the designations for start and end codes. If the number of bytes to be sent is set to 0000, only the start and end codes will be sent.



To reset the RS-232C port (i.e., to restore the initial status), turn on SR 25209. To reset the peripheral port, turn on SR 25208. These bits will turn OFF automatically after the reset.

**Receptions**

- 1, 2, 3... 1. Confirm that AR 0806 (RS-232C Reception Complete Flag) or AR 0814 (Peripheral Reception Complete Flag) is ON.  
 2. Use the RXD(47) instruction to receive the data.



D: Leading word no. for storing reception data  
 C: Control data  
     Bits 00 to 03  
         0: Leftmost bytes first  
         1: Rightmost bytes first  
     Bits 12 to 15  
         0: RS-232C port  
         1: Peripheral port  
 N: Number of bytes stored (4 digits BCD), 0000 to 0256

3. The results of reading the data received will be stored in the AR area. Check to see that the operation was successfully completed. The contents of these bits will be reset each time RXD(47) is executed.

RS-232C port	Peripheral port	Error
AR 0800 to AR 0803	AR 0808 to AR 0811	RS-232C port error code (1 digit BCD) 0: Normal completion 1: Parity error 2: Framing error 3: Over-run error
AR 0804	AR0812	Communications error
AR 0807	AR0815	Reception Overrun Flag (After reception was completed, the subsequent data was received before the data was read by means of the RXD(47) instruction.)
AR 09	AR10	Number of bytes received

To reset the RS-232C port (i.e., to restore the initial status), turn ON SR 25209. To reset the peripheral port, turn ON SR 25208. These bits will turn OFF automatically after the reset.

The start code and end code are not included in AR 09 or AR 10 (number of bytes received).

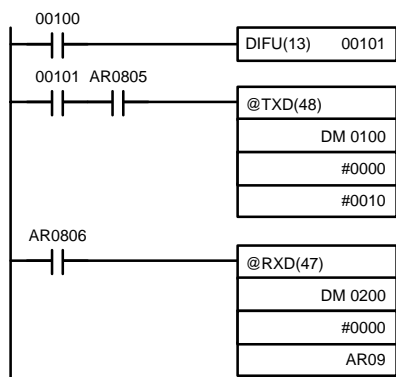
**Application Example**

This example shows a program for using the RS-232C port in the RS-232C Mode to transmit 10 bytes of data (DM 0100 to DM 0104) to the computer, and to store the data received from the computer in the DM area beginning with DM 0200. Before executing the program, the following PC Setup setting must be made.

DM 6645: 1000 (RS-232C port in RS-232C Mode; standard communications conditions)

DM 6648: 2000 (No start code; end code CR/LF)

The default values are assumed for all other PC Setup settings. From DM 0100 to DM 0104, 3132 is stored in every word. From the computer, execute a program to receive CQM1 data with the standard communications conditions.



If AR 0805 (the Transmit Ready Flag) is ON when IR 00100 turns ON, the ten bytes of data (DM 0100 to DM 0104) will be transmitted, leftmost bytes first.

When AR 0806 (Reception Completed Flag) goes ON, the number of bytes of data specified in AR 09 will be read from the CQM1's reception buffer and stored in memory starting at DM 0200, leftmost bytes first.

The data will be as follows: "31323132313231323132CR LF"

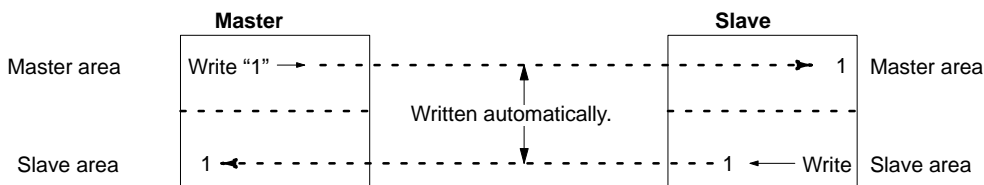
**1-9-7 CQM1 One-to-one Link Communications**

If two CQM1s are linked one-to-one by connecting them together through their RS-232C ports, they can share common LR areas. When two CQM1s are linked one-to-one, one of them will serve as the master and the other as the slave.

**Note** The peripheral port cannot be used for 1:1 links.

**One-to-one Links**

A one-to-one link allows two CQM1s to share common data in their LR areas. As shown in the diagram below, when data is written into a word the LR area of one of the linked Units, it will automatically be written identically into the same word of the other Unit. Each PC has specified words to which it can write and specified words that are written to by the other PC. Each can read, but cannot write, the words written by the other PC.



The word used by each PC will be as shown in the following table, according to the settings for the master, slave, and link words.

DM 6645 setting	LR 00 to LR 63	LR 00 to LR 31	LR 00 to LR 15
Master words	LR00 to LR31	LR00 to LR15	LR00 to LR07
Slave words	LR32 to LR63	LR16 to LR31	LR08 to LR15

**Communications Procedure**

If the settings for the master and the slave are made correctly, then the one-to-one link will be automatically started up simply by turning on the power supply to both the CQM1s and operation will be independent of the CQM1 operating modes.



**Link Errors**

If a slave does not received a response from the master within one second, the 1:1 Link Error Flag (AR 0802) and the Communications Error Flag (AR 0804) will be turned ON.

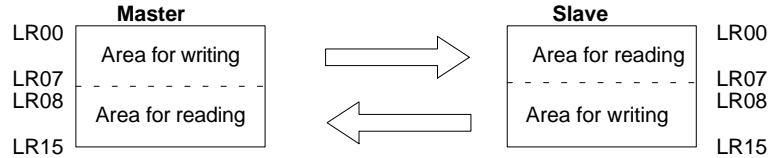
**Application Example**

This example shows a program for verifying the conditions for executing a one-to-one link using the RS-232C ports. Before executing the program, set the following PC Setup parameters.

Master: DM 6645: 3200 (one-to-one link master; Area used: LR 00 to LR 15)

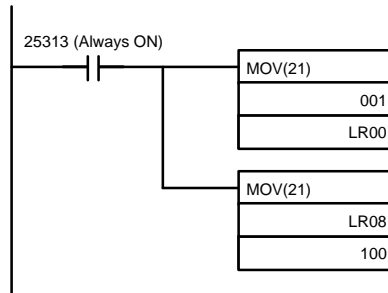
Slave: DM 6645: 2000 (one-to-one link slave)

The defaults are assumed for all other PC Setup parameters. The words used for the one-to-one link are as shown below.

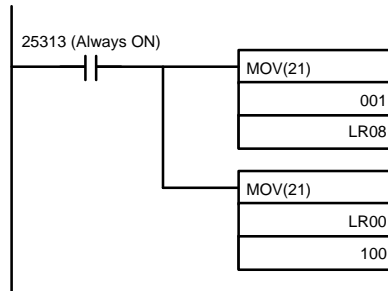


When the program is executed at both the master and the slave, the status of IR 001 of each Unit will be reflected in IR 100 of the other Unit. Likewise, the status of the other Unit's IR 001 will be reflected in IR 100 of each Unit. IR 001 is an input word and IR 100 is an output word

**In the Master**



**In the Slave**

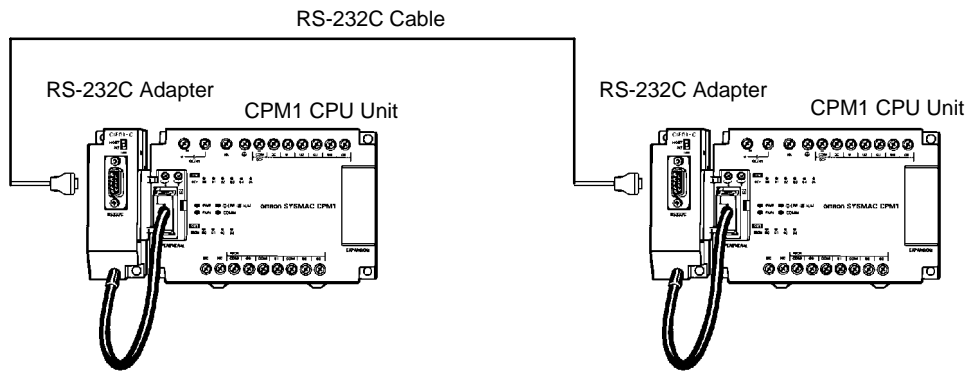


**1-9-8 CPM1/CPM1A One-to-one Link Communications**

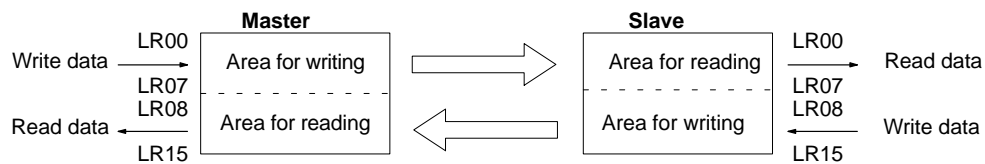
In a one-to-one link, a CPM1/CPM1A is linked to another CPM1/CPM1A, a CQM1, or a C200HS through an RS-232C Adapter and standard RS-232C cable. One of the PCs will serve as the master and the other as the slave. The one-to-one link can connect up to 256 bits (LR 0000 to LR 1515) in the two PCs.

**One-to-one CPM1/CPM1A Links**

The following diagram shows a one-to-one link between two CPM1s PCs. Refer to the *CPM1A Operation Manual* for the corresponding information on the CPM1A.



The words used for the one-to-one link are as shown below.



**Limitations of One-to-one Links with a CPM1/CPM1A**

Only the 16 LR words from LR 00 to LR 15 can be linked in the CPM1/CPM1A, so use only those 16 words in the CQM1 or C200HS when making a one-to-one link with one of those PCs. A one-to-one link cannot be made to a CPM1/CPM1A PC using LR 16 through LR 63 in the CQM1 or C200HS.

**PC Setup Settings**

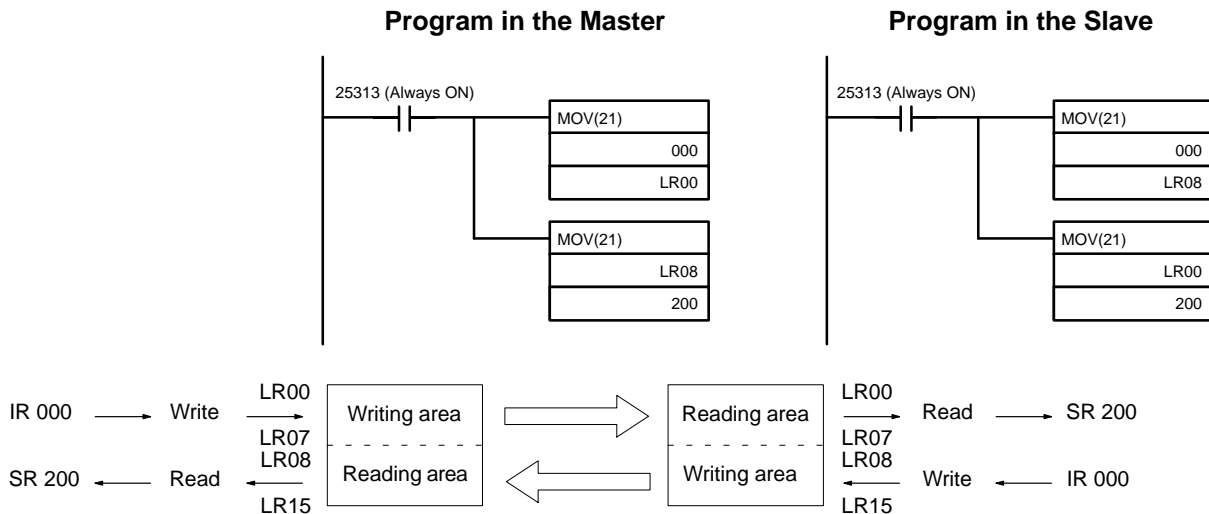
The settings relating to 1-to-1 PC communications must be set as shown in the following table.

Word	Bit	Function	Setting (Master)	Setting (Slave)
DM 6650	00 to 07	Port settings <sup>1</sup> 00: Standard (1 start bit, 7-bit data, even parity, 2 stop bits, 9,600 bps) 01: Settings in DM 6651	00 (Any value is OK)	00 (Any value is OK)
	08 to 11	Link area for one-to-one PC link via peripheral port 0: LR 00 to LR 15	0	0 (Any value is OK)
	12 to 15	Communications mode <sup>1</sup> 0: Host link; 2: 1-to-1 PC link (slave); 3: 1-to-1 PC link (master); 4: NT link	3	2

- Note**
1. If an improper setting is used, a non-fatal error will occur, AR 1302 will be turned ON, and the default setting (0 or 00) will be used.
  2. For information on the 1-to-1 link settings for another OMRON PC, refer to that PC's Operation Manual.
  3. For information on CPM1/CPM1A one-to-one link connections and wiring diagrams refer to 3-4-7 *Host Link Connections* in the *CPM1 Operation Manual* or *CPM1A Operation Manual*. For the SRM1 refer to 3-4-4 *RS-232C Port Wiring* in the *SRM1 Master Control Unit Operation Manual*.
  4. If an out-of-range value is set, the following communications conditions will result. In that case, reset the value so that it is within the permissible range.
    - Communications mode: Host Link
    - Communications format: Standard settings  
(1 start bit, 7-bit data; even parity, 2 stop bits, 9,600 bps)
    - Transmission delay: No
    - Node number: 00

**Example Program**

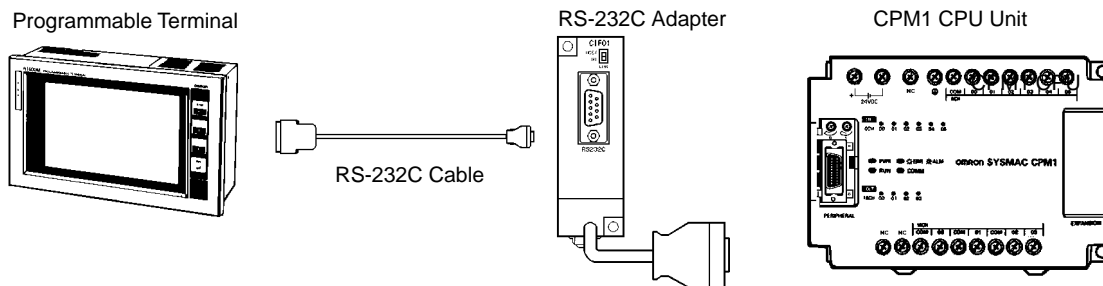
This example shows ladder programs that copy the status of IR 000 in each CPM1/CPM1A to SR 200 in the other CPM1/CPM1A.



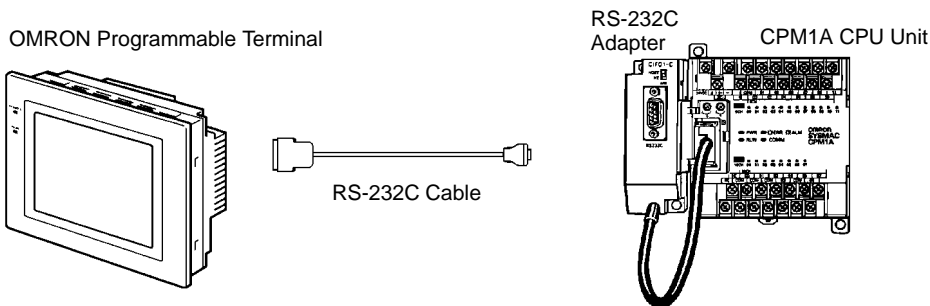
**1-9-9 CPM1/CPM1A NT Link Communications**

Using the NT link, the CPM1/CPM1A PC can be connected to the Programmable Terminal (NT Link Interface) through an RS-232C Adapter.

**CPM1 PCs**



**CPM1A PCs**



**PC Setup Settings**

The settings relating to NT link PC communications must be set as shown in the following table.

Word	Bit	Function	Setting
DM 6650	00 to 07	Port settings <sup>1</sup> 00: Standard (1 start bit, 7-bit data, even parity, 2 stop bits, 9,600 bps) 01: Settings in DM 6651	00 (Any value is OK)
	08 to 11	Link area for one-to-one PC link via peripheral port 0: LR 00 to LR 15	0 (Any value is OK)
	12 to 15	Communications mode <sup>1</sup> 0: Host link; 2: 1-to-1 PC link (slave); 3: 1-to-1 PC link (master); 4: NT link	4

- Note**
1. If an improper setting is used, a non-fatal error will occur, AR 1302 will be turned ON, and the default setting (0 or 00) will be used.
  2. For information on the NT link settings for another OMRON PC, refer to that PC's Operation Manual.
  3. If an out-of-range value is set, the following communications conditions will result. In that case, reset the value so that it is within the permissible range.

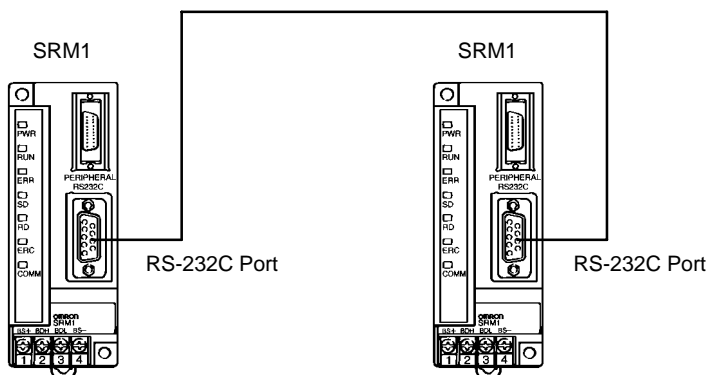
Communications mode: Host Link  
 Communications format: Standard settings  
 (1 start bit, 7-bit data; even parity, 2 stop bits, 9,600 bps)  
 Transmission delay: No  
 Node number: 00

### 1-9-10 SRM1 One-to-one Link Communications

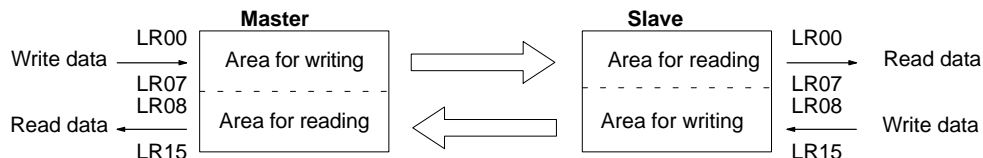
In a one-to-one link, a SRM1 is linked to another SRM1, CPM1/CPM1A, a CQM1, or a C200HS through an RS-232C Adapter and standard RS-232C cable. One of the PCs will serve as the master and the other as the slave. The one-to-one link can connect up to 256 bits (LR 0000 to LR 1515) in the two PCs.

#### One-to-one SRM1 Links

The following diagram shows a one-to-one link between two SRM1s.



The words used for the one-to-one link are as shown below.



#### Limitations of One-to-one Links with a SRM1

Only the 16 LR words from LR 00 to LR 15 can be linked in the SRM1, so use only those 16 words in the CQM1 or C200HS when making a one-to-one link with one of those PCs. A one-to-one link cannot be made to a SRM1 PC using LR 16 through LR 63 in the CQM1 or C200HS.

**PC Setup Settings**

The settings relating to 1-to-1 PC communications must be set as shown in the following table.

Word	Bit	Function	Setting (Master)	Setting (Slave)
DM 6645	00 to 03	Port settings <sup>1</sup> 00: Standard (1 start bit, 7-bit data, even parity, 2 stop bits, 9,600 bps) 01: Settings in DM 6651	Any	Any
	04 to 07	CTS Control settings 0: Disable 1: Set	0	0
	08 to 11	Link area for one-to-one PC link via peripheral port 0: LR 00 to LR 15	0	0
	12 to 15	Communications mode <sup>1</sup> 0: Host link; 1: No protocol; 2: 1-to-1 PC link (slave); 3: 1-to-1 PC link (master); 4: NT link	3	2

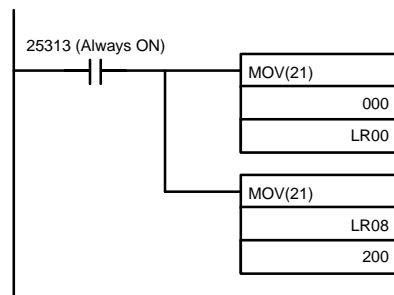
- Note**
1. If an improper setting is used, a non-fatal error will occur, AR 1302 will be turned ON, and the default setting (0 or 00) will be used.
  2. For information on the 1-to-1 link settings for another OMRON PC, refer to that PC's Operation Manual.
  3. If an out-of-range value is set, the following communications conditions will result. In that case, reset the value so that it is within the permissible range.

Communications mode: Host Link  
 Communications format: Standard settings  
 (1 start bit, 7-bit data; even parity, 2 stop bits, 9,600 bps)  
 Transmission delay: No  
 Node number: 00

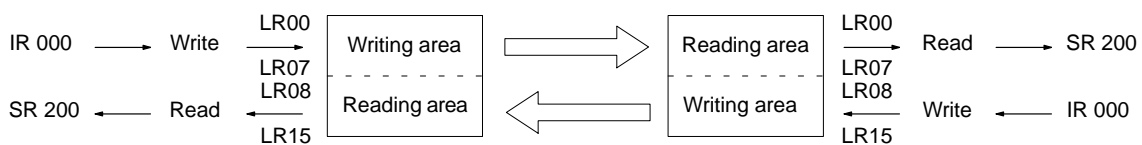
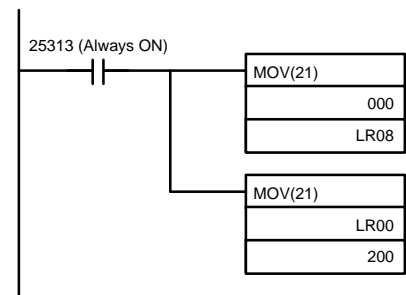
**Example Program**

This example shows ladder programs that copy the status of IR 000 in each SRM1 to SR 200 in the other SRM1.

**Program in the Master**

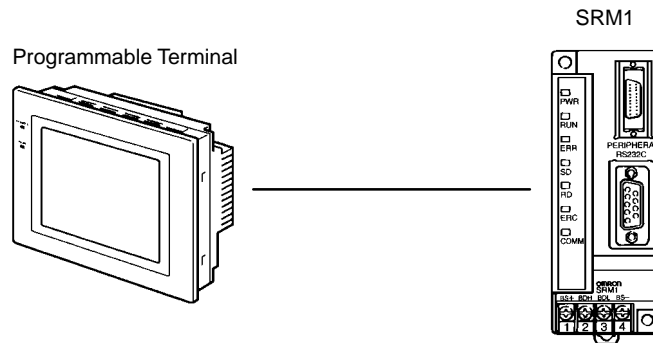


**Program in the Slave**



### 1-9-11 SRM1 NT Link Communications

Using the NT link, the SRM1 PC can be connected to the Programmable Terminal (NT Link Interface.) The RS-232C port can be used to the NT Link.



NT Link is possible only with the SRM1-C02-V1, which has an RS-232C port.

#### PC Setup Settings

The settings relating to NT link PC communications must be set as shown in the following table.

Word	Bit	Function	Setting
DM 6645	00 to 03	Port settings <sup>1</sup> 00: Standard (1 start bit, 7-bit data, even parity, 2 stop bits, 9,600 bps) 01: Settings in DM 6646	00 (Any value is OK)
	04 to 07	CTS control settings 0: Disable 1: Set	0 (Any value is OK)
	08 to 11	Link area for one-to-one PC link via peripheral port 0: LR 00 to LR 15	0
	12 to 15	Communications mode <sup>1</sup> 0: Host link; 1: No protocol; 2: 1-to-1 PC link (slave); 3: 1-to-1 PC link (master); 4: NT link	4

- Note**
1. If an improper setting is used, a non-fatal error will occur, AR 1302 will be turned ON, and the default setting (0 or 00) will be used.
  2. For information on the NT link settings for another OMRON PC, refer to that PC's Operation Manual.
  3. If an out-of-range value is set, the following communications conditions will result. In that case, reset the value so that it is within the permissible range.

Communications mode: Host Link  
 Communications format: Standard settings  
 (1 start bit, 7-bit data; even parity, 2 stop bits, 9,600 bps)  
 Transmission delay: No  
 Node number: 00

### 1-9-12 SRM1 No Protocol Communications

**Peripheral Port Settings** When the peripheral port is used to conduct no protocol communications, the following settings must be made from the peripheral device to DM 6650 to DM 6653 in the SRM1.

Word	Bit	Function	Setting																																																																
<b>Peripheral Port Settings</b>																																																																			
The following settings are effective after transfer to the PC.																																																																			
DM 6650	00 to 03	Port settings 0: Standard (1 start bit, 7-bit data, even parity, 2 stop bits, 9,600 bps) 1: Settings in DM 6651  (Other settings will cause a non-fatal error, the default setting (0) will be used, and AR 1302 will turn ON.)	As required																																																																
	04 to 07	Not used.	0																																																																
	08 to 11	Not used.	0																																																																
	12 to 15	Communications mode 0: Host link; 1: No protocol  (Other settings will cause a non-fatal error, the default setting (0) will be used, and AR 1302 will turn ON.)	1: No protocol																																																																
DM 6651	00 to 07	Baud rate 00: 1.2K, 01: 2.4K, 02: 4.8K, 03: 9.6K, 04: 19.2K	As required																																																																
	08 to 15	Frame format <table border="0" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;"></th> <th style="text-align: left;">Start</th> <th style="text-align: left;">Length</th> <th style="text-align: left;">Stop</th> <th style="text-align: left;">Parity</th> </tr> </thead> <tbody> <tr><td>00:</td><td>1 bit</td><td>7 bits</td><td>1 bit</td><td>Even</td></tr> <tr><td>01:</td><td>1 bit</td><td>7 bits</td><td>1 bit</td><td>Odd</td></tr> <tr><td>02:</td><td>1 bit</td><td>7 bits</td><td>1 bit</td><td>None</td></tr> <tr><td>03:</td><td>1 bit</td><td>7 bits</td><td>2 bit</td><td>Even</td></tr> <tr><td>04:</td><td>1 bit</td><td>7 bits</td><td>2 bit</td><td>Odd</td></tr> <tr><td>05:</td><td>1 bit</td><td>7 bits</td><td>2 bit</td><td>None</td></tr> <tr><td>06:</td><td>1 bit</td><td>8 bits</td><td>1 bit</td><td>Even</td></tr> <tr><td>07:</td><td>1 bit</td><td>8 bits</td><td>1 bit</td><td>Odd</td></tr> <tr><td>08:</td><td>1 bit</td><td>8 bits</td><td>1 bit</td><td>None</td></tr> <tr><td>09:</td><td>1 bit</td><td>8 bits</td><td>2 bit</td><td>Even</td></tr> <tr><td>10:</td><td>1 bit</td><td>8 bits</td><td>2 bit</td><td>Odd</td></tr> <tr><td>11:</td><td>1 bit</td><td>8 bits</td><td>2 bit</td><td>None</td></tr> </tbody> </table> (Other settings will cause a non-fatal error, the default setting (00) will be used, and AR 1302 will turn ON.)		Start	Length	Stop	Parity	00:	1 bit	7 bits	1 bit	Even	01:	1 bit	7 bits	1 bit	Odd	02:	1 bit	7 bits	1 bit	None	03:	1 bit	7 bits	2 bit	Even	04:	1 bit	7 bits	2 bit	Odd	05:	1 bit	7 bits	2 bit	None	06:	1 bit	8 bits	1 bit	Even	07:	1 bit	8 bits	1 bit	Odd	08:	1 bit	8 bits	1 bit	None	09:	1 bit	8 bits	2 bit	Even	10:	1 bit	8 bits	2 bit	Odd	11:	1 bit	8 bits	2 bit	None
	Start	Length	Stop	Parity																																																															
00:	1 bit	7 bits	1 bit	Even																																																															
01:	1 bit	7 bits	1 bit	Odd																																																															
02:	1 bit	7 bits	1 bit	None																																																															
03:	1 bit	7 bits	2 bit	Even																																																															
04:	1 bit	7 bits	2 bit	Odd																																																															
05:	1 bit	7 bits	2 bit	None																																																															
06:	1 bit	8 bits	1 bit	Even																																																															
07:	1 bit	8 bits	1 bit	Odd																																																															
08:	1 bit	8 bits	1 bit	None																																																															
09:	1 bit	8 bits	2 bit	Even																																																															
10:	1 bit	8 bits	2 bit	Odd																																																															
11:	1 bit	8 bits	2 bit	None																																																															
DM 6652	00 to 15	Transmission delay (Host Link) 0000 to 9999 (BCD): Set in units of 10 ms.  (Other settings will cause a non-fatal error, the default setting (0000) will be used, and AR 1302 will turn ON.)	To match host parameters																																																																
DM 6653	00 to 07	Node number (Host link) 00 to 31 (BCD)  (Other settings will cause a non-fatal error, the default setting (0000) will be used, and AR 1302 will turn ON.)	00 to 31																																																																
	08 to 11	Start code enable (RS-232C, effective when bits 12 to 15 of DM 6650 are set to 1.) 0: Disable 1: Set	As required																																																																
	12 to 15	End code enable (RS-232C, effective when bits 12 to 15 of DM 6650 are set to 1.) 0: Disable (number of bytes received) 1: Set (specified end code) 2: CR, LF	As required																																																																

Word	Bit	Function	Setting
DM 6654	00 to 07	Start code (effective when bits 08 to 11 of DM6650 are set to 1.) 00: 256 bytes 01 to FF: 1 to 255 bytes	As required
	08 to 15	End code (no protocol) When bits 12 to 15 of DM6653 are set to 0: 00: 256 bytes 01 to FF: 1 to 255 bytes When bits 12 to 15 of DM6653 are set to 1: Setting: 00 to FF (binary)	As required

**Note** If an out-of-range value is set, the following communications conditions will result. In that case, reset the value so that it is within the permissible range.

Communications mode: Host Link  
 Communications format: Standard settings  
 (1 start bit, 7-bit data; even parity, 2 stop bits, 9,600 bps)  
 Transmission delay: No  
 Node number: 00

**RS-232C Port Settings**

When the RS-232C port is used to conduct no protocol communications, the following settings must be made from the peripheral device to DM6645 to DM6649 in the SRM1.

Word	Bit	Function	Setting																																																																
<b>RS-232C Port Settings</b>																																																																			
The following settings are effective after transfer to the PC.																																																																			
DM 6645	00 to 03	Port settings 0: Standard (1 start bit, 7-bit data, even parity, 2 stop bits, 9,600 bps) 1: Settings in DM 6646	As required 0																																																																
	04 to 07	CTS control settings 0: Disable; 1: Set																																																																	
	08 to 11	Link words for 1:1 link 0: LR 00 to LR 15; Other: Not effective	0																																																																
	12 to 15	Communications mode 0: Host link; 1: RS-232C (no protocol); 2: 1:1 data link slave; 3: 1:1 data link master; 4: NT Link	1: No protocol																																																																
DM 6646	00 to 07	Baud rate 00: 1.2K, 01: 2.4K, 02: 4.8K, 03: 9.6K, 04: 19.2K	As required																																																																
	08 to 15	Frame format <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th></th> <th>Start</th> <th>Length</th> <th>Stop</th> <th>Parity</th> </tr> </thead> <tbody> <tr><td>00:</td><td>1 bit</td><td>7 bits</td><td>1 bit</td><td>Even</td></tr> <tr><td>01:</td><td>1 bit</td><td>7 bits</td><td>1 bit</td><td>Odd</td></tr> <tr><td>02:</td><td>1 bit</td><td>7 bits</td><td>1 bit</td><td>None</td></tr> <tr><td>03:</td><td>1 bit</td><td>7 bits</td><td>2 bit</td><td>Even</td></tr> <tr><td>04:</td><td>1 bit</td><td>7 bits</td><td>2 bit</td><td>Odd</td></tr> <tr><td>05:</td><td>1 bit</td><td>7 bits</td><td>2 bit</td><td>None</td></tr> <tr><td>06:</td><td>1 bit</td><td>8 bits</td><td>1 bit</td><td>Even</td></tr> <tr><td>07:</td><td>1 bit</td><td>8 bits</td><td>1 bit</td><td>Odd</td></tr> <tr><td>08:</td><td>1 bit</td><td>8 bits</td><td>1 bit</td><td>None</td></tr> <tr><td>09:</td><td>1 bit</td><td>8 bits</td><td>2 bit</td><td>Even</td></tr> <tr><td>10:</td><td>1 bit</td><td>8 bits</td><td>2 bit</td><td>Odd</td></tr> <tr><td>11:</td><td>1 bit</td><td>8 bits</td><td>2 bit</td><td>None</td></tr> </tbody> </table>		Start	Length	Stop	Parity	00:	1 bit	7 bits	1 bit	Even	01:	1 bit	7 bits	1 bit	Odd	02:	1 bit	7 bits	1 bit	None	03:	1 bit	7 bits	2 bit	Even	04:	1 bit	7 bits	2 bit	Odd	05:	1 bit	7 bits	2 bit	None	06:	1 bit	8 bits	1 bit	Even	07:	1 bit	8 bits	1 bit	Odd	08:	1 bit	8 bits	1 bit	None	09:	1 bit	8 bits	2 bit	Even	10:	1 bit	8 bits	2 bit	Odd	11:	1 bit	8 bits	2 bit	None
	Start	Length	Stop	Parity																																																															
00:	1 bit	7 bits	1 bit	Even																																																															
01:	1 bit	7 bits	1 bit	Odd																																																															
02:	1 bit	7 bits	1 bit	None																																																															
03:	1 bit	7 bits	2 bit	Even																																																															
04:	1 bit	7 bits	2 bit	Odd																																																															
05:	1 bit	7 bits	2 bit	None																																																															
06:	1 bit	8 bits	1 bit	Even																																																															
07:	1 bit	8 bits	1 bit	Odd																																																															
08:	1 bit	8 bits	1 bit	None																																																															
09:	1 bit	8 bits	2 bit	Even																																																															
10:	1 bit	8 bits	2 bit	Odd																																																															
11:	1 bit	8 bits	2 bit	None																																																															
DM 6647	00 to 15	Transmission delay (Host Link) 0000 to 9999 (BCD): Set in units of 10 ms, e.g., setting of 0001 equals 10 ms	As required																																																																



Word	Bit	Function	Setting
DM 6648	00 to 07	Node number (Host link, effective when bits 12 to 15 of DM 6645 are set to 0.) 00 to 31 (BCD)	As required
	08 to 11	Start code enable (RS-232C, effective when bits 12 to 15 of DM 6645 are set to 1.) 0: Disable; 1: Set	
	12 to 15	End code enable (RS-232C, effective when bits 12 to 15 of DM 6645 are set to 1.) 0: Disable (number of bytes received) 1: Set (specified end code) 2: CR, LF	
DM 6649	00 to 07	Start code (RS-232C) 00: 256 bytes 01 to FF: 1 to 255 bytes	
	08 to 15	End code enable (RS-232C) 00 to FF (BIN)	

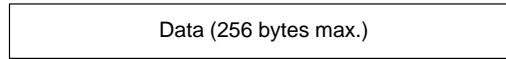
**Note** If an out-of-range value is set, the following communications conditions will result. In that case, reset the value so that it is within the permissible range.

Communications mode: Host Link  
 Communications format: Standard settings  
 (1 start bit, 7-bit data; even parity, 2 stop bits, 9,600 bps)  
 Transmission delay: No  
 Node number: 00

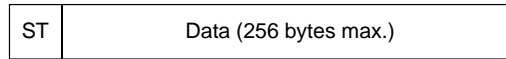
### 1-9-13 Transmission Data Configuration

When no-protocol communications are used, TXD(48) is used to send data and RXD(47) to receive data. The maximum amount of data that can be either sent or received is 259 bytes, including the start/end code.

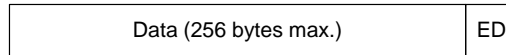
• **No Start or End Code:**



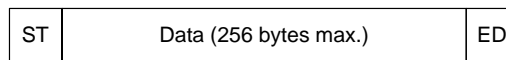
• **Only a Start Code:**



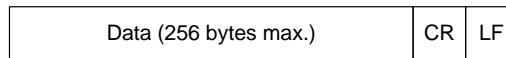
• **Only an End Code:**



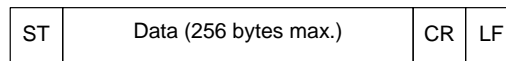
• **Both an Start and End Code:**



• **End Code of CR, LF:**



• **Start Code 00-FF/End Code CR,LF:**



- Note**
1. The start and end codes are set in DM 6648 to DM 6649 and DM 6653 to DM 6654 of the PC Setup.
  2. When there are several start and end codes, the first part of each will be effective.
  3. When the end code duplicates the transmission data and the transmission is stopped part way through, use CR or LF as the end code.
  4. The start and end codes are not stored.

### 1-9-14 Transmission Flags

When sending data from the SRM1, check that the Transmission Enable Flag is ON for executing the TXD(48) instruction. The Transmission Enable Flag will turn OFF while the data is being transmitted and will turn ON again when transmission is complete.

After the SRM1 has received data, the Receive Enable Flag turns ON. When the RXD instruction is executed, the data received will be written to the specified words and the Reception Complete Flag will turn OFF.

Flag	Peripheral port	RS-232C port
Transmission Enable Flag	AR 0813	AR 0805
Reception Complete Flag	AR 0814	AR 0806

**Note** The timing from data reception starting to completion for the SRM1 is as indicated below.

**Reception Start:**

- Without start code: Normal reception status
- With start code: After start code is received.

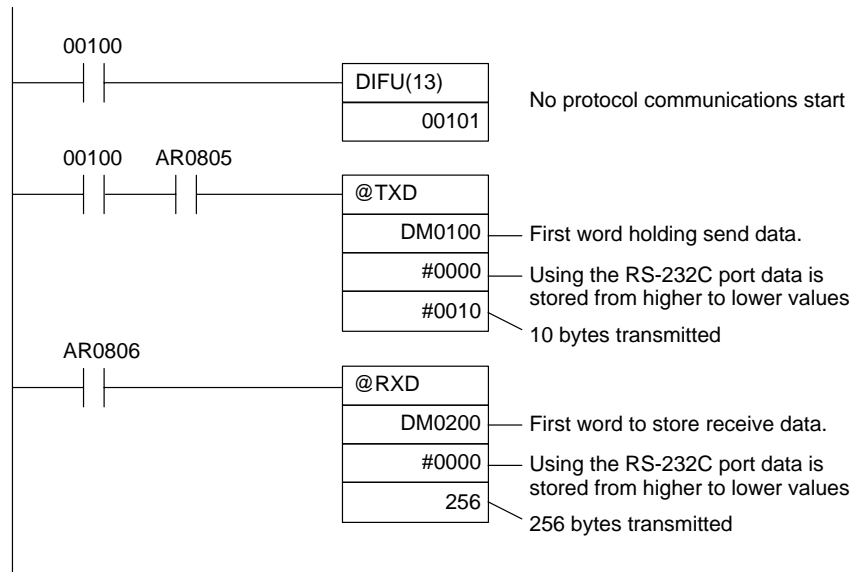
**Reception Complete:**

When either the end code, the specified no. of bytes, or 256 bytes are received.

**1-9-15 No Protocol Communications Program Example**

The following program example is for no protocol communication conducted through a RS-232C port using TXD(48) and RXD(47) instructions.

If AR 0805 (Transmission Enable Flag) is ON when 00100 is ON, then data from DM0100 to DM0104 is transmitted from higher to lower values. When AR 0806 (Reception Enable Flag) turns ON, 256 bytes of received data are read and written to DM 0200 from higher to lower values.



**1-10 Calculating with Signed Binary Data**

The CQM1/CPM1/CPM1A/SRM1 allow calculations on signed binary data. The following instructions manipulate signed binary data. Signed data is handled using 2's complements.

**CQM1 Instructions**

The following signed-binary instructions are available in CQM1 PCs:

**Single-word Instructions**

- 2'S COMPLEMENT – NEG(—)
- BINARY ADD – ADB(50)
- BINARY SUBTRACT – SBB(51)
- SIGNED BINARY MULTIPLY – MBS(—)
- SIGNED BINARY DIVIDE – DBS(—)

**Double-word (Long) Instructions**

- DOUBLE 2'S COMPLEMENT – NEGL(—)
- DOUBLE BINARY ADD – ADBL(—)
- DOUBLE BINARY SUBTRACT – SBBL(—)
- DOUBLE SIGNED BINARY MULTIPLY – MBSL(—)
- DOUBLE SIGNED BINARY DIVIDE – DBSL(—)

**CPM1/CPM1A/SRM1 Instructions**

The following signed-binary instructions are available in CPM1/CPM1A/SRM1 PCs:

- BINARY ADD – ADB(50)
- BINARY SUBTRACT – SBB(51)

Signed Data Calculations

**Addition**

$7 + 3 = 10$   
 $(-7) + 3 = -4$   
 $7 + (-3) = 4$   
 $(-7) + (-3) = -10$

**Multiplication**

$7 \times 3 = 21$   
 $(-7) \times 3 = -21$   
 $7 \times (-3) = -21$   
 $(-7) \times (-3) = 21$

**Subtraction**

$7 - 3 = 4$   
 $(-7) - 3 = -10$   
 $7 - (-3) = 10$   
 $(-7) - (-3) = -4$

**Division**

$7 \div 3 = 2$  with a remainder of 1  
 $(-7) \div 3 = -2$  with a remainder of -1  
 $7 \div (-3) = -2$  with a remainder of 1  
 $(-7) \div (-3) = 2$  with a remainder of -1

1-10-1 Definition of Signed Binary Data

The CQM1 provides instructions that operate on either one or two words of data; the CPM1/CPM1A/SRM1 provide just two instructions that operate on one word of data. Signed binary data is manipulated using 2's complements and the MSB of the one- or two-word data is used as the sign bit. The range of data that can be expressed using one or two words is thus as follows:

- **One-word data:**  
 -32,768 to 32,767 (8000 to 7FFF hexadecimal)
- **Two-word data:**  
 -2,147,483,648 to 2,147,483,647 (8000 0000 to 7FFF FFFF hexadecimal)

The following table shows equivalents between decimal and hexadecimal data.

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

## 1-10-2 Arithmetic Flags

The results of executing signed binary instructions is reflected in the arithmetic flags. The flags and the conditions under which it will turn ON are given in the following table. The flags will be OFF when these conditions are not met.

Flag	ON conditions
Carry Flag (SR 25504)	Carry in an addition. Negative results for subtraction.
Equals Flag (SR 25506)	The results of addition, subtraction, multiplication, or division is 0. Results of converting 2's complement is 0.
Overflow Flag (SR 25404)	32,767 (7FFF) was exceeded in results of 16-bit addition or subtraction. 2,147,483,647 (7FFF FFFF) was exceeded in results of 32-bit addition or subtraction.
Underflow Flag (SR 25405)	-32,768 (8000) was exceeded in results of 16-bit addition or subtraction, or conversion of 2's complement. -2,147,483,648 (8000 0000) was exceeded in results of 32-bit addition or subtraction, or conversion of 2's complement.

## 1-10-3 Inputting Signed Binary Data Using Decimal Values

Although calculations for signed binary data use hexadecimal expressions, inputs from the Programming Console or SSS can be done using decimal inputs and mnemonics for the instructions. The procedure to using the Programming Console to input using decimal values is shown in the *CQM1 Operation Manual*, *CPM1 Operation Manual*, *CPM1A Operation Manual* and *SRM1 Master Control Unit Operation Manual*. Refer to the *SSS Operation Manual: C-series PCs* for details on using the SSS.

### Input Instructions

Only 16-bit operands can be input for the following instructions: NEG(—), ADB(50), SBB(51), MBS(—), and DBS(—). Refer to the *CQM1 Operation Manual*, *CPM1 Operation Manual*, *CPM1A Operation Manual* or *SRM1 Master Control Unit Operation Manual* for details on inputting instructions from the Programming Console.

## 1-10-4 Using Signed-binary Expansion Instructions (CQM1 Only)

The following CQM1 instructions must be allocated function codes in the instructions table before they can be used.

- 2'S COMPLEMENT – NEG(—)
- DOUBLE 2'S COMPLEMENT – NEGL(—)
- DOUBLE BINARY ADD – ADBL(—)
- DOUBLE BINARY SUBTRACT – SBBL(—)
- SIGNED BINARY MULTIPLY – MBS(—)
- DOUBLE SIGNED BINARY MULTIPLY – MBSL(—)
- SIGNED BINARY DIVIDE – DBS(—)
- DOUBLE SIGNED BINARY DIVIDE – DBSL(—)

### Allocating Function Codes

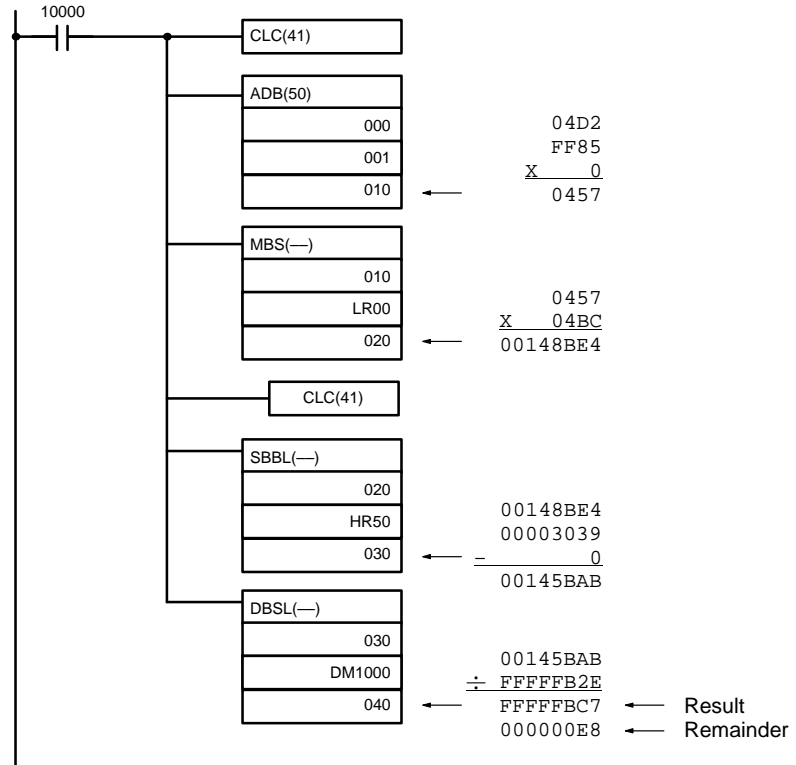
The procedure to using the Programming Console to allocate function codes is shown in the *CQM1 Operation Manual*. Be sure that pin 4 on the CQM1's DIP switch is turned ON to enable use of a user-set instruction table before performing this operation.

### 1-10-5 Application Example Using Signed Binary Data

The following programming can be used to performed calculations such as the following in the CQM1:

$$((1234 + (-123)) \times 1212 - 12345) \div (-1234) = -1081, \text{ Remainder of } 232$$

000	=	04D2	←	1234
001	=	FF85	←	-123
LR00	=	04BC	←	1212
HR50	=	3039	←	12345
HR51	=	0000	←	
DM1000	=	FB2E	←	-1234
DM1001	=	FFFF	←	



## SECTION 2

# Special Features

This section provides an introduction to some of the main CQM1/CPM1/CPM1A/SRM1 features, including the instructions available through expansion instructions, a monitoring feature called differential monitoring, and the analog setting function available in the CQM1-CPU42-EV1 and CPM1/CPM1A PCs.

If you are not familiar with OMRON PCs or ladder diagram program, you can skim over this section as an overview of these features, but will want to read *Section 3 Memory Areas* and *Section 4 Ladder-diagram Programming* before reading this section in detail. Also, details on programming instructions can be found in *Section 5 Instruction Set*.

2-1	Expansion Instructions (CQM1/SRM1 Only) .....	116
2-1-1	CQM1 Expansion Instructions .....	117
2-1-2	SRM1 Expansion Instructions .....	117
2-2	Advanced I/O Instructions (CQM1 Only) .....	118
2-2-1	TEN-KEY INPUT – TKY(18) .....	118
2-2-2	HEXADECIMAL KEY INPUT – HKY(—) .....	120
2-2-3	DIGITAL SWITCH INPUT – DSW(87) .....	122
2-2-4	7-SEGMENT DISPLAY OUTPUT – 7SEG(88) .....	125
2-2-5	Alternate I/O Bits .....	127
2-3	Macro Function .....	128
2-4	Differential Monitor .....	129
2-5	Analog Settings (CQM1-CPU42-EV1/CPM1/CPM1A Only) .....	129
2-6	Quick-response Inputs (CPM1/CPM1A Only) .....	131

## 2-1 Expansion Instructions (CQM1/SRM1 Only)

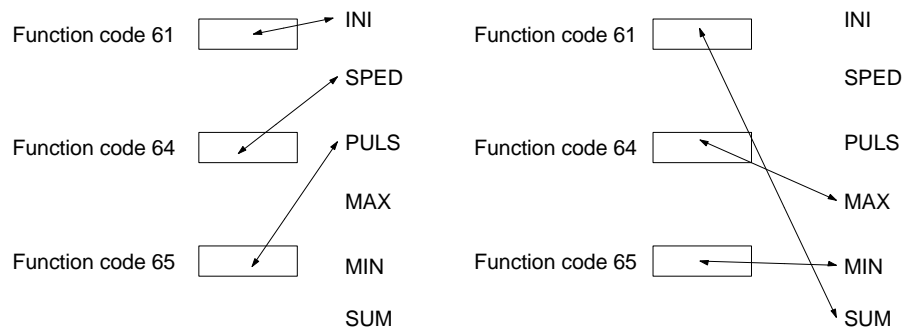
A set of expansion instructions is available for the CQM1/SRM1 to aid in special programming needs. Function codes can be assigned to up to 18 of the expansion instructions to enable using them in programs. This allows the user to pick the instructions needed by each CQM1 or SRM1 program to more effectively use the function codes required to input instructions.

The mnemonics of expansion instructions are followed by “(—)” as the function code to indicate that they must be assigned function codes by the user in the instructions table before they can be used in programming (unless they are used under their default settings).

Any of the instructions not assigned function codes will need to be assigned function codes in the instructions table used by the Programming Device and the CQM1 or SRM1 before they can be used in programming. The assignments of expansion instructions in the instructions table will change the meaning of instructions and operands, so be sure to set the instructions table before programming and transfer the proper instructions table to the CQM1 or SRM1 before program execution.

### Example: CQM1 PCs

The specific instructions used in the following example are for the CQM1. The concepts are the same for the SRM1.



At the time of shipping, the function codes are assigned as shown above. (In this example, the instructions all relate to pulse outputs.)

If pulse outputs are not being used, and if maximum values, minimum values, and sums are required, then the Set Instructions operation can be used as shown above to reassign instructions in the instruction table.

**Note** Set the PC model to “CQM1” when setting the expansion instructions for the SRM1 or CQM1 from the SSS.



## 2-1-1 CQM1 Expansion Instructions

The following 18 function codes can be used for expansion instructions: 17, 18, 19, 47, 48, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 87, 88, and 89

The expansion instructions that can be used are listed below, along with the default function codes that are assigned when the CQM1 is shipped. Instructions marked with a "\*" are available only in CQM1-CPU4□-EV1 CPU Units.

Mnemonic	Function code	Mnemonic	Function code	Mnemonic	Function code
ASFT	17	INT	89	ADBL*	---
TKY	18	HKY	---	SBBL*	---
MCMP	19	FPD	---	MBS*	---
RXD	47	SRCH	---	DBS*	---
TXD	48	MAX	---	MBSL*	---
CMPL	60	MIN	---	DBSL*	---
INI	61	APR	---	CPS*	---
PRV	62	LINE	---	CPSL*	---
CTBL	63	COLM	---	NEG*	---
SPED	64	SEC	---	NEGL*	---
PULS	65	HMS	---	ZCP*	---
SCL	66	SUM	---	ZCPL*	---
BCNT	67	FCS	---	XFRB*	---
BCMP	68	HEX	---	PLS2*	---
STIM	69	AVG	---	ACC*	---
DSW	87	PWM*	---	SCL2*	---
7SEG	88	PID*	---	SCL3*	---

With the CQM1, an instructions table can also be stored on Memory Cassettes when they are used. Exercise care when using a Memory Cassette that has been used with another CQM1 or SRM1 and be sure the proper instructions table is present.



### Caution

If pin no. 4 of the CQM1's DIP switch is set to OFF, only expansion instructions on the default instructions table can be used and the user-set instructions table will be ignored. The default instructions table will also be set whenever power is turned on, deleting any previous settings.

Make sure that pin 4 of the CPU Unit DIP switch is ON when reading a program from the Memory Cassette that has a user-set expansion instructions table. If pin 4 is OFF, the default instructions table will be used for expansion instructions in programs read from a Memory Cassette. (In this case, the program read from the Memory Cassette and the program on the Memory Cassette will not match when the two are compared.)

## 2-1-2 SRM1 Expansion Instructions

The following 18 function codes can be used for expansion instructions: 17, 18, 19, 47, 48, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 87, 88, and 89

The expansion instructions that can be used are listed below, along with the default function codes that are assigned when the SRM1 is shipped.

Mnemonic	Function code
ASFT	17
RXD	47
TXD	48
CMPL	60

Mnemonic	Function code
BCNT	67
BCMP	68
STIM	69
FCS	---
HEX	---
STUP	---

## 2-2 Advanced I/O Instructions (CQM1 Only)

Advanced I/O instructions enable control, with a single instruction, of previously complex operations involving external I/O devices (digital switches, 7-segment displays, etc.). This section introduces advanced I/O instructions, which are also covered at the end of *Section 5 Instruction Set*.

There are four advanced I/O instructions, as shown in the following table. All of these are expansion instructions and must be assigned to function codes before they can be used.

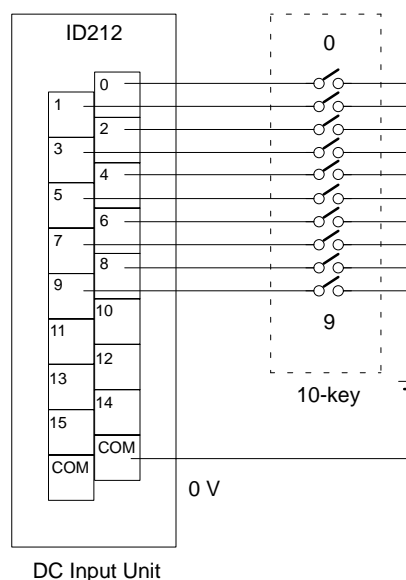
Name	Mnemonic	Function
TEN-KEY INPUT	TKY(18)	BCD input from 10-key keypad
HEXADECIMAL KEY INPUT	HKY(—)	Hexadecimal input from 16-key keypad
DIGITAL SWITCH INPUT	DSW(87)	SV input from digital switch
7-SEGMENT DISPLAY OUTPUT	7SEG(88)	BCD output to 7-segment display

### 2-2-1 TEN-KEY INPUT – TKY(18)

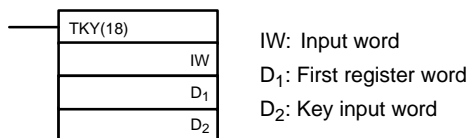
This instruction inputs 8 digits in BCD from a 10-key keypad and uses 10 input points.

#### Hardware

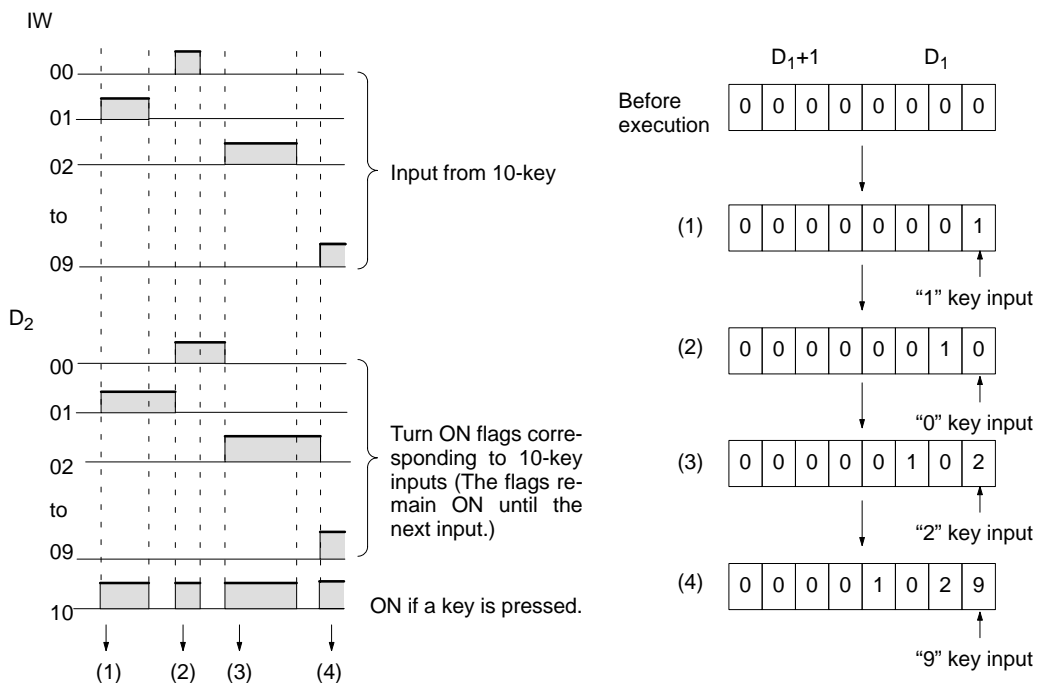
Prepare a 10-key keypad, and connect it so that the switches for numeric keys 0 through 9 are input to points 0 through 9 as shown in the following diagram. Either the input terminals on the CPU Unit or the inputs on a DC Input Unit with 16 or more input points can be used.



Using the Instruction



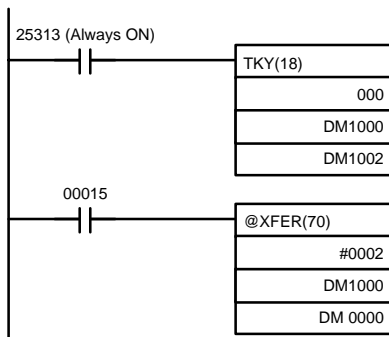
If the input word for connecting the 10-key keypad is specified for IW, then operation will proceed as shown below when the program is executed.



- Note**
1. While one key is being pressed, input from other keys will not be accepted.
  2. If more than eight digits are input, digits will be deleted beginning with the leftmost digit.
  3. Input bits not used here can be used as ordinary input bits.

Application Example

In this example, a program for inputting numbers from the 10-key is shown. Assume that the 10-key is connected to IR 000.



The 10-key information input to IR 000 using TKY(18) is converted to BCD and stored in DM 1000 and DM 1001. Key information is stored in DM 1002.

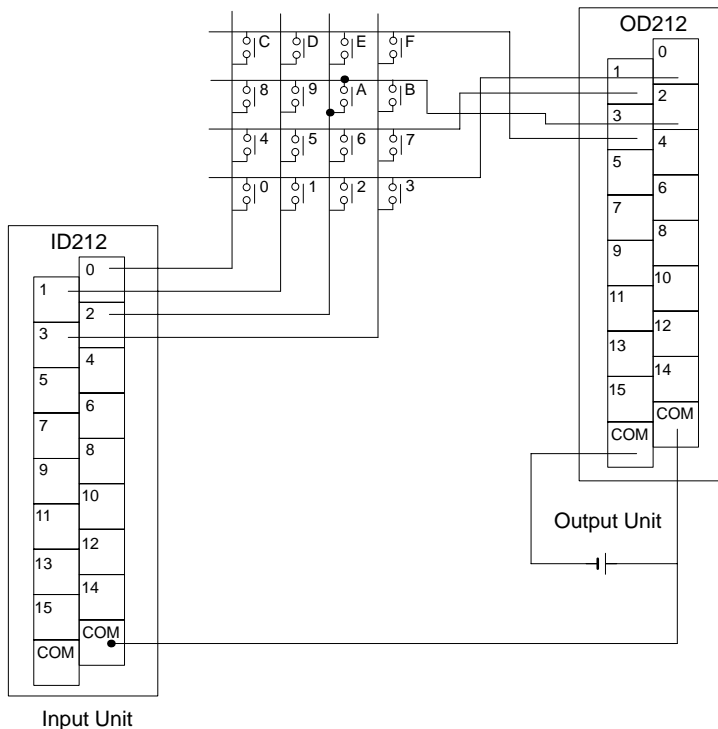
IR 00015 is used as an "ENTER key," and when IR 00015 turns ON, the data stored in DM 1000 and DM 1001 will be transferred to DM 0000 and DM 0001.

### 2-2-2 HEXADECIMAL KEY INPUT – HKY(—)

This instruction inputs 8 digits in hexadecimal from a hexadecimal keyboard. It utilizes 5 output bits and 4 input bits.

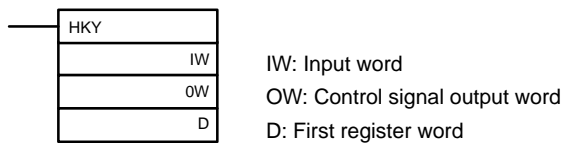
**Hardware**

Prepare the hexadecimal keyboard, and connect the 0 to F numeric key switches, as shown below, to input points 0 through 3 and output points 0 through 3. Output point 4 will be turned ON while any key is being pressed, but there is no need to connect it.

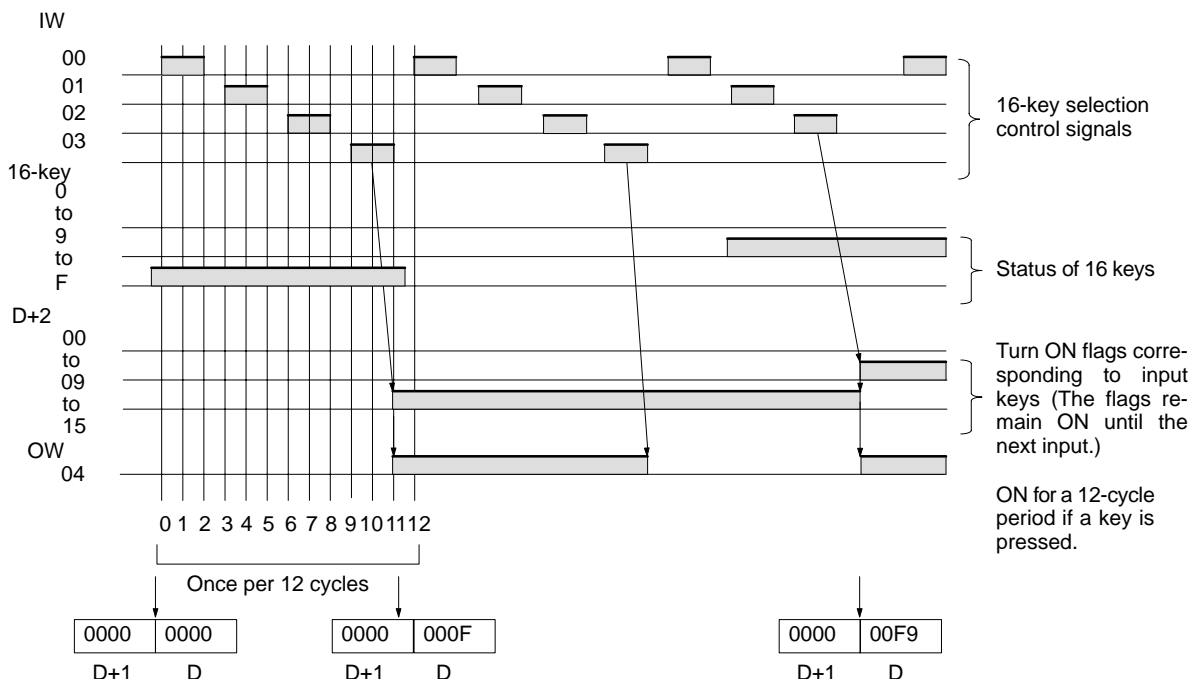


The inputs can be connected to the input terminals on the CPU Unit or a DC Input Unit with 8 or more input points and the outputs can be connected from a Transistor Output Unit with 8 points or more.

**Using the Instruction**



If the input word for connecting the hexadecimal keyboard is specified at IW, and the output word is specified at OW, then operation will proceed as shown below when the program is executed.



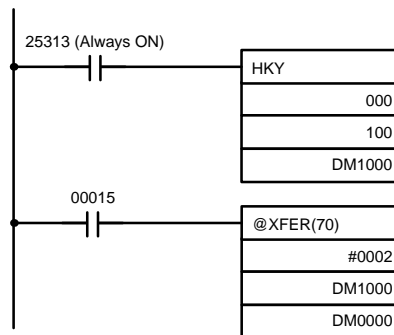
SR 25408 will turn ON while HKY(—) is being executed.

- Note**
1. Do not use HKY(—) more than once within the same program.
  2. When using HKY(—), set the input constant for the relevant input word to less than the cycle time. (Input constants can be changed from DM 6620 onwards.)
  3. While one key is being pressed, input from other keys will not be accepted.
  4. If more than eight digits are input, digits will be deleted beginning with the leftmost digit.
  5. Input and output bits not used here can be used as ordinary input and output bits.

With this instruction, one key input is read in 3 to 12 cycles. More than one cycle is required because the ON keys can only be determined as the outputs are turned ON to test them.

**Application Example**

This example shows a program for inputting numbers from a hexadecimal keyboard. Assume that the hexadecimal keyboard is connected to IR 000 (input) and IR 100 (output).



The hexadecimal key information that is input to IR 000 by HKY(—) is converted to hexadecimal and stored in words DM1000 and DM1001.

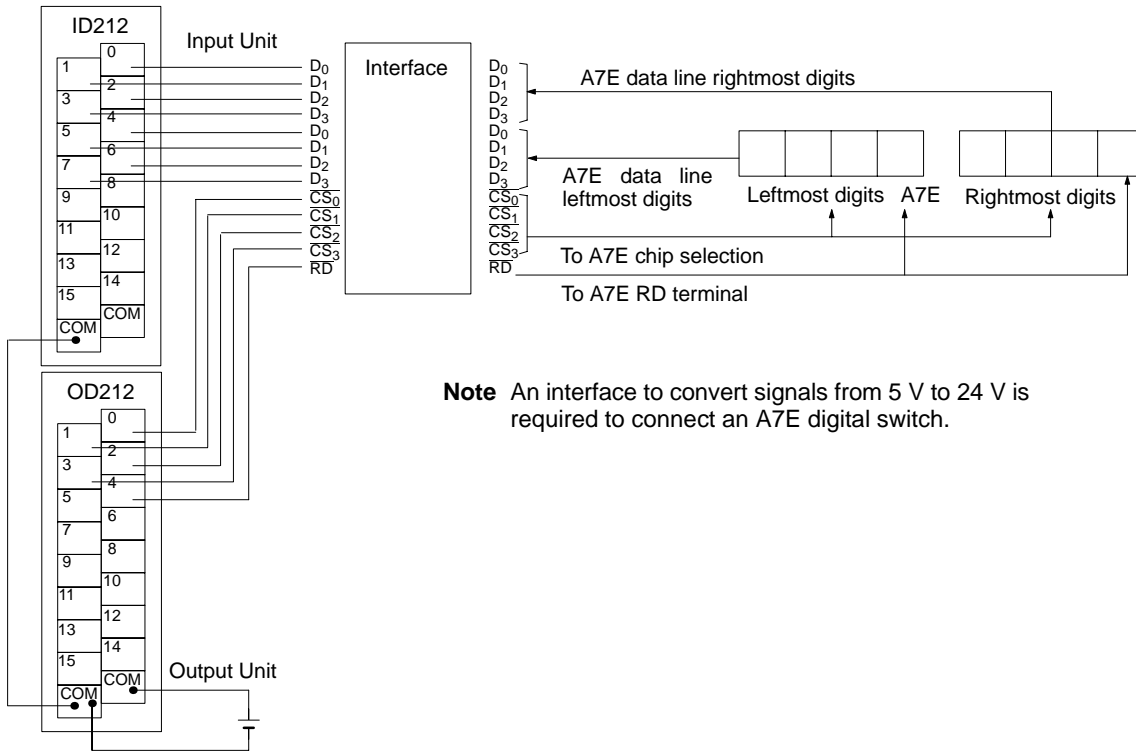
IR 00015 is used as an “ENTER key,” and when IR 00015 turns ON, the numbers stored in DM 1000 and DM 1001 are transferred to DM 0000 and DM 0001.

### 2-2-3 DIGITAL SWITCH INPUT – DSW(87)

With this instruction, 4-digit or 8-digit BCD set values are read from a digital switch. DSW(87) utilizes 5 output bits and either 4 input bits (for 4 digits) or 8 input bits (for 8 digits).

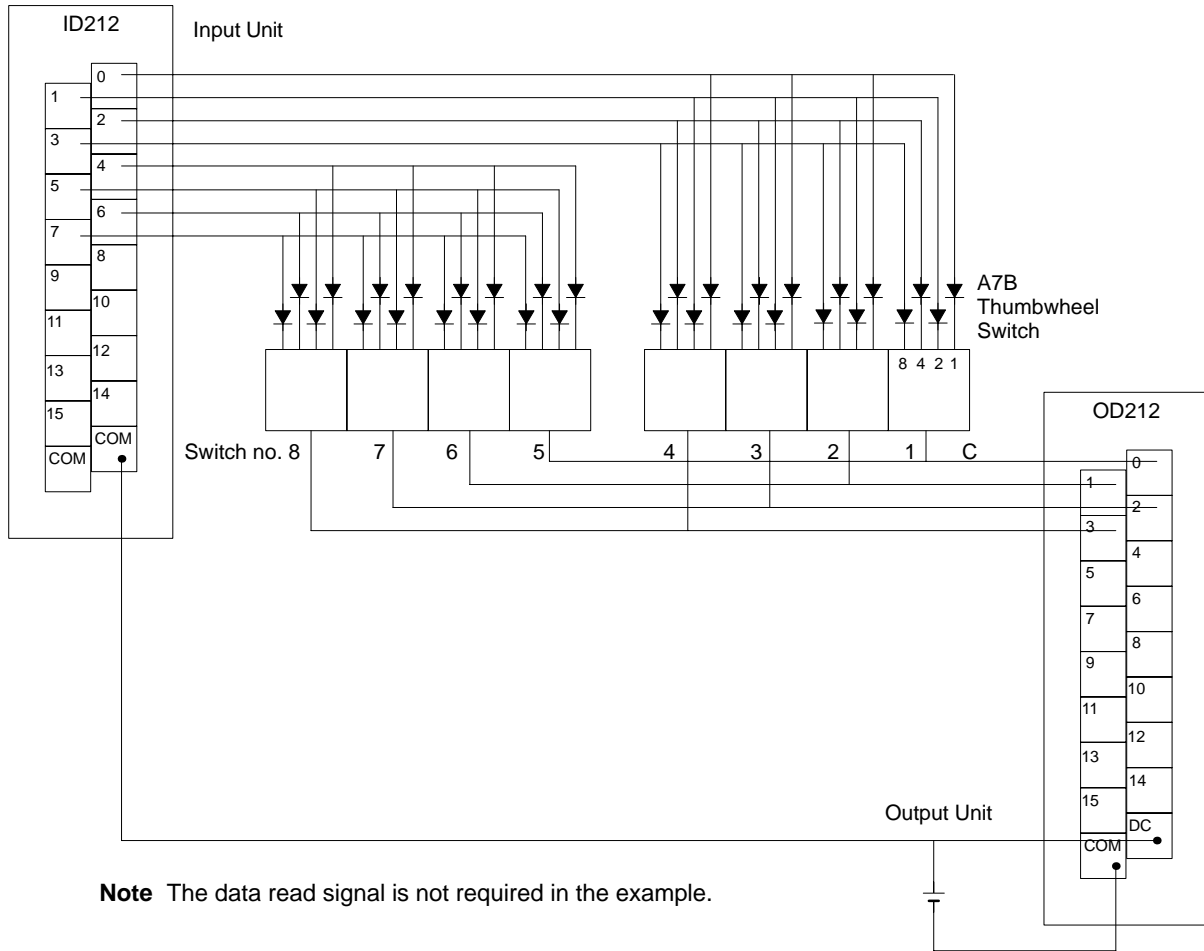
#### Hardware

Connect the digital switch and the Input and Output Units as shown in the diagram below. In the diagram, an 8-digit input is shown. When using a 4-digit input, connect D0 through D3 from the digital switch to input points 0 through 3. In either case, output point 5 will be turned ON when one round of data is read, but there is no need to connect output point 5 unless required for the application.



**Note** An interface to convert signals from 5 V to 24 V is required to connect an A7E digital switch.

The following example illustrates connections for an A7B Thumbwheel Switch.



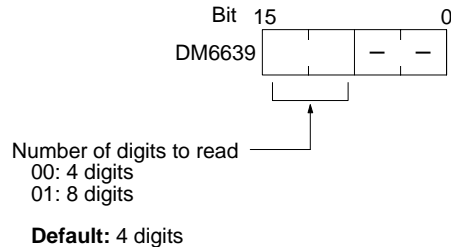
**Note** The data read signal is not required in the example.

The inputs can be connected to the CPU Unit's input terminals or a DC Input Unit with 8 or more input points and the outputs can be connected from a Transistor Output Unit with 8 or more output points.

**Preparations**

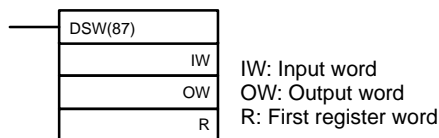
When using DSW(87), make the following setting in the PC Setup in PROGRAM mode before executing the program.

**Digital Switch Settings (PC Setup)**

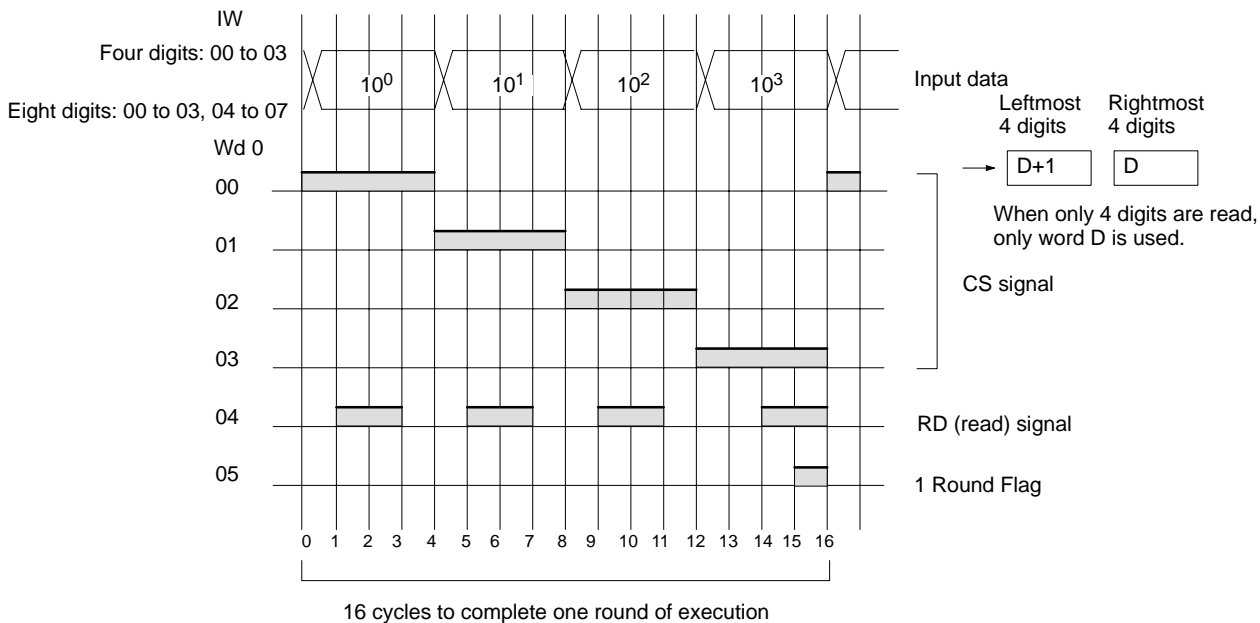


Do not make any changes to bits 0 to 7. They are not related to DSW(87).

**Using the Instruction**



If the input word for connecting the digital switch is specified at for IW, and the output word is specified for OW, then operation will proceed as shown below when the program is executed.



SR 25410 will turn ON while DSW(87) is being executed.

- Note**
1. Do not use DSW(87) more than once within the same program.
  2. When using DSW(87), set the input constant for the relevant input word to less than the cycle time. (Input constants can be changed from DM 6620 onwards.) The characteristics of the digital switch must also be considered in system and program design.
  3. Input and output bits not used here can be used as ordinary input and output bits.

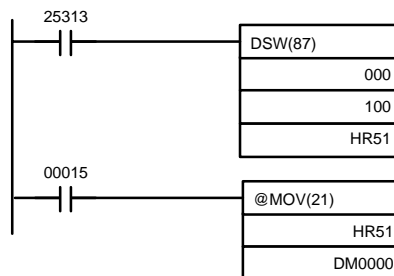
With this instruction, 4-digit or 8-digit set values can be read in 16 cycles.

**Application Example**

This example shows a program for reading 4 digits in BCD from the digital switch. Assume that the digital switch is connected to IR 000 (input) and IR 100 (output), and assume the default status for all the PC Setup (4 digits to read).

The data set from the digital switch by DSW(87) is stored in HR 51.

When IR 00015 turns ON, the value stored in HR 51 is moved to DM 0001.



- Note** Output point 5 (here, IR 10005) turns on when one round of data is read and can be used to time switching the data storage area and gate signal (CS signal) when DSW(87) is used to input data to different areas of memory.

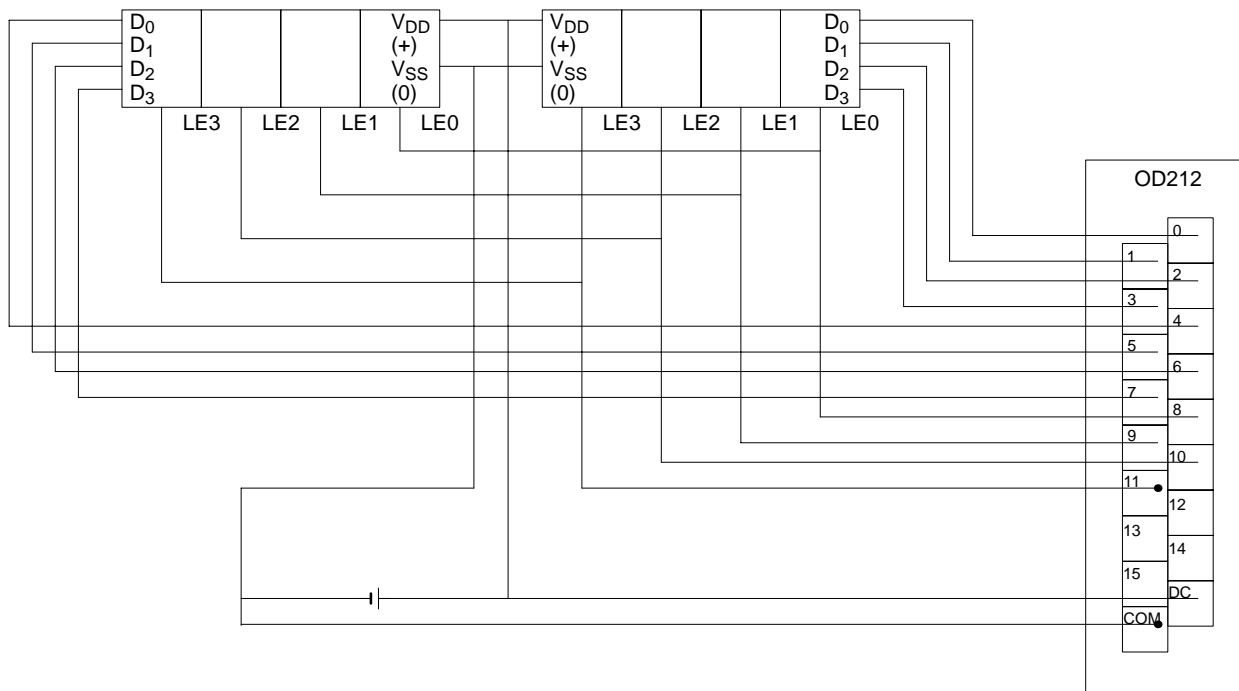


### 2-2-4 7-SEGMENT DISPLAY OUTPUT – 7SEG(88)

This instruction outputs word data to a 7-segment display. It utilizes either 8 (for 4 digits) or 12 (for 8 digits) output bits.

**Hardware**

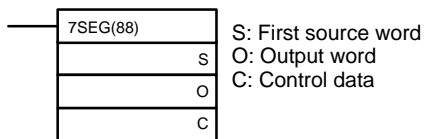
The 7-segment display is connected to an Output Unit as shown in the diagram below. For 4-digit display, the data outputs (D0 to D3) are connected to output points 0 through 3, and latch outputs (CS0 to CS3) are connected to output points 4 through 7. Output point 12 (for 8-digit display) or output point 8 (for 4-digit display) will be turned ON when one round of data is displayed, but there is no need to connect them unless required by the application.



The outputs can be connected from a Transistor Output Unit with 8 or more output points for four digits or 16 or more output points for eight digits.

- Note**
1. Output Unit outputs normally employ negative logic. (Only the PNP output type employs positive logic.)
  2. The 7-segment display may require either positive or negative logic, depending on the model.

**Using the Instruction**



If the first word holding the data to be displayed is specified at S, and the output word is specified at O, and the SV taken from the table below is specified at C, then operation will proceed as shown below when the program is executed.

**Data Storage Format**



If only four digits are displayed, then only word S will be used.

Set Values for Selecting Logic and Number of Digits (C)

Number of digits displayed	Display Unit data input and Output Unit logic	Display Unit latch input and Output Unit logic	C setting data
4 digits (4 digits, 1 block)	Same	Same	000
		Different	001
	Different	Same	002
		Different	003
8 digits (4 digits, 2 blocks)	Same	Same	004
		Different	005
	Different	Same	006
		Different	007

**Note** Do not set C to values other than 000 to 007.

Function	Bit(s) in O		Output status (Data and latch logic depends on C)
	(4 digits, 1 block)	(4 digits, 2 blocks)	
Data output	00 to 03	00 to 03 04 to 07	<p><b>Note</b> 0 to 3: Data output for word S 4 to 7: Data output for word S+1</p>
Latch output 0	04	08	
Latch output 1	05	09	
Latch output 2	06	10	
Latch output 3	07	11	
One Round Flag	08	12	

SR 25409 will turn ON while 7SEG(88) is being executed.

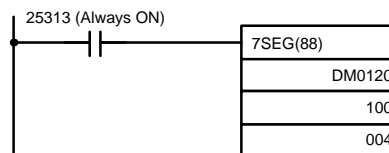
- Note**
1. Do not use 7SEG(88) more than once within the same program.
  2. Consider the cycle time and the characteristics of the 7-segment display when designing the system.
  3. Output bits not used here can be used as ordinary output bits.

With this instruction, 4 digits or 8 digits are displayed in 12 cycles.

Operation will proceed from the first execution without regard to the status prior to execution.

**Application Example**

This example shows a program for displaying the CQM1's 8-digit BCD numbers at the 7-segment LED display. Assume that the 7-segment display is connected to output word IR 100. Also assume that the Output Unit is using negative logic, and that the 7-segment display logic is also negative for data signals and latch signals.



The 8-digit BCD data in DM 0120 (rightmost 4 digits) and DM 0121 (leftmost 4 digits) are always displayed by means of 7SEG(88). When the contents of DM 0120 and DM 0121 change, the display will also change.

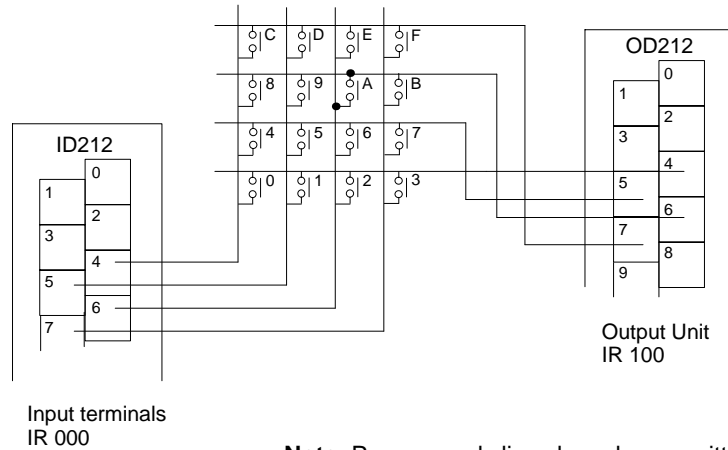
### 2-2-5 Alternate I/O Bits

Although the advanced I/O instructions generally using I/O bits starting from bit 00 of the specified words, they can be programmed through intermediate words to use other I/O bits. The following example shows how this can be achieved for HKY(—).

**Example**

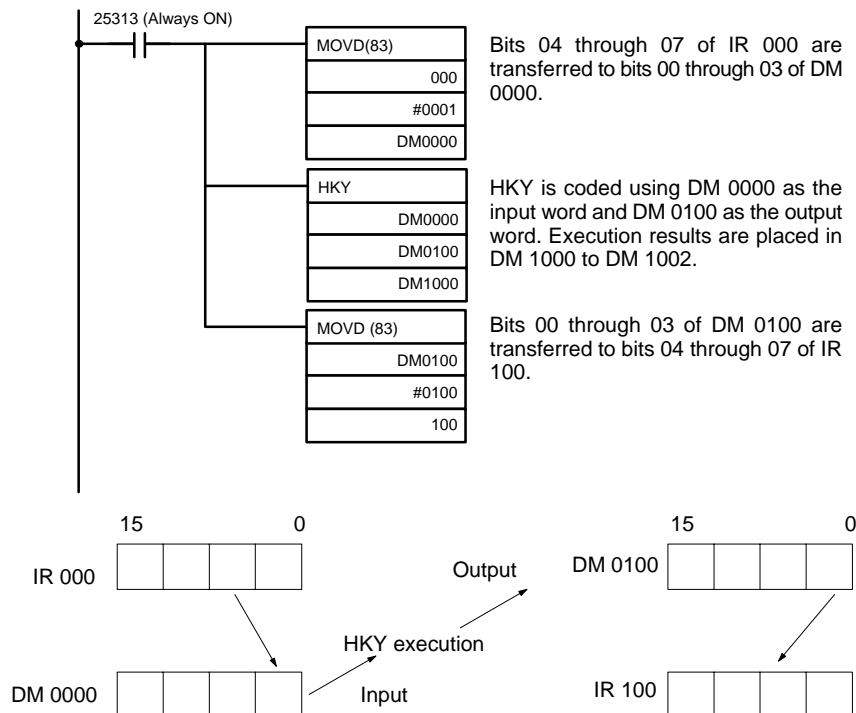
The following wiring and program examples show how to use input bits IR 00004 through IR 00007 and output bits IR 10004 through IR 10007 to input values from a hexadecimal keypad.

**Wiring Diagram (Not Complete)**



**Note** Power supply lines have been omitted.

**Program**

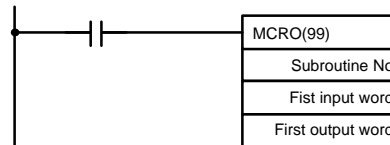


## 2-3 Macro Function

The macro function allows a single subroutine (programming pattern) to be used by simply changing the I/O word. A number of similar program sections can be managed with just one subroutine, thereby greatly reducing the number of steps in the program and making the program easier to understand.

### Using Macros

To use a macro, call a subroutine by means of the MACRO instruction, MCRO(99), as shown below, in stead of SBS(91) (SUBROUTINE ENTRY).



When MCRO(99) is executed, operation will proceed as follows:

- 1, 2, 3...
  1. The contents of the four consecutive words beginning with the first input word will be transferred to IR 096 through IR 099 (SR 232 through SR 235 in CPM1/CPM1A/SRM1 PCs). The contents of the four consecutive words beginning with the first output word will be transferred to IR 196 through IR 199 (SR 236 through SR 239 in CPM1, CPM1A and SRM1 PCs).
  2. The specified subroutine will be executed until RET(93) (Subroutine Return) is executed.
  3. The contents of IR 196 through IR 199 (SR 236 through SR 239 in CPM1/CPM1A/SRM1 PCs) will be transferred to the four consecutive words beginning with the first output word.
  4. MCRO(99) will then be finished.

When MCRO(99) is executed, the same instruction pattern can be used as needed simply by changing the first input word or the first output word.

The following restrictions apply when the macro function is used.

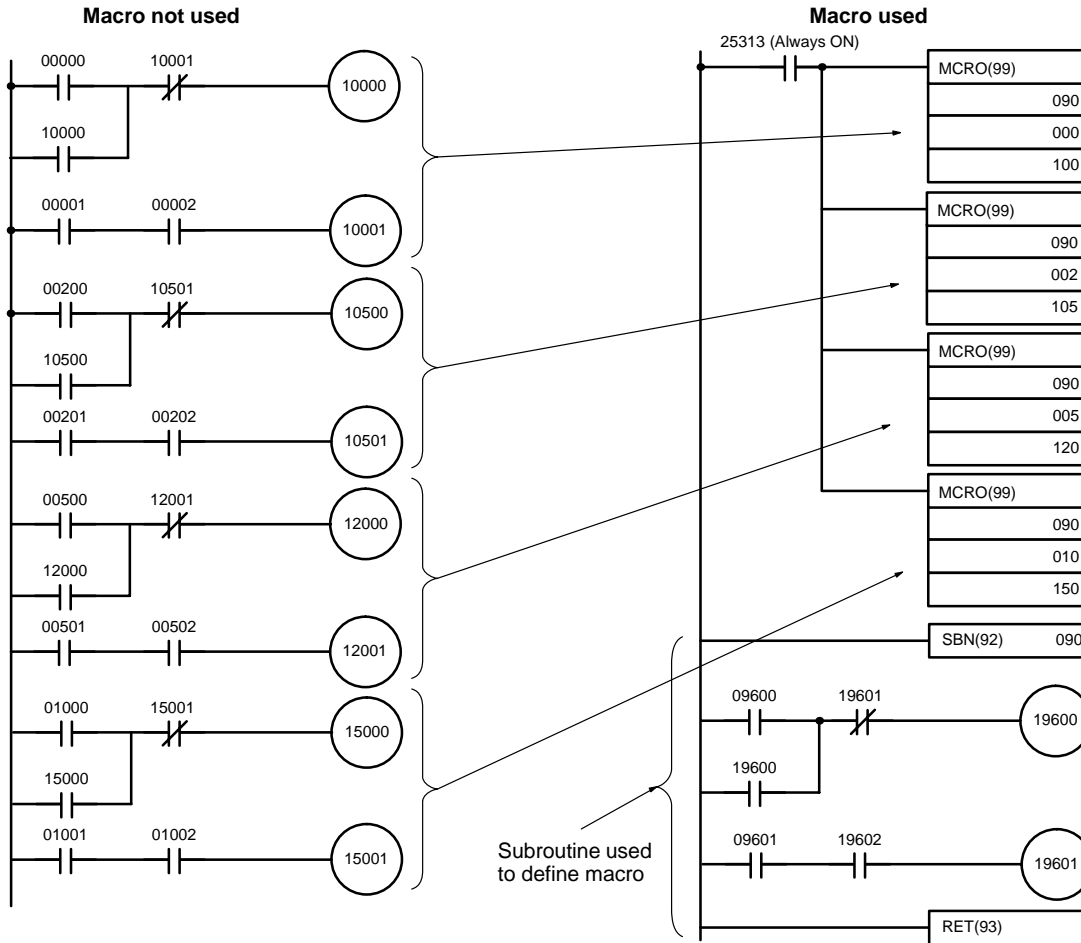
- The only words that can be used for each execution of the macro are the four consecutive words beginning with the first input word number (for input) and the four consecutive words beginning with the first output word (for output).
- The specified inputs and outputs must correctly correspond to the words used in the subroutine.
- Even when the direct output method is used for outputs, subroutine results will be actually reflected in the specified output words only when the subroutine has been completed (step 3 above).

- Note**
1. In CQM1 PCs, IR 096 to IR 099 and IR 196 to IR 199 can be used as work bits when MCRO(99) is not used.
  2. In CPM1/CPM1A/SRM1 PCs, SR 232 through SR 239 can be used as work bits when MCRO(99) is not used.

The first input word and the first output word can be specified not with I/O bits, but also with other bits (such as HR bits, work bits, etc.) or with DM words.

Subroutines called by MCRO(99) are defined by SBN(92) and RET(93), just as are ordinary subroutines.

**CQM1 Application Example** When a macro is used, the program can be simplified as shown below.



**CPM1/CPM1A/SRM1 Application Example**

The CPM1/CPM1A/SRM1 program can be simplified like the one shown above, but words SR 232 through SR 235 would be used instead of IR 096 through IR 099 and words SR 236 through SR 239 would be used instead of IR 196 through IR 199.

**2-4 Differential Monitor**

The CQM1/CPM1/CPM1A/SRM1 support differential monitoring from either the Programming Console or the SSS. The operator can detect on OFF-to-ON or ON-to-OFF transition in a specified bit. When the specified transition takes place, the transition is indicated on the display and a buzzer sounds to enable easy recognition of the transition.

Refer to the *CQM1 Operation Manual*, *CPM1 Operation Manual*, *CPM1A Operation Manual* or *SRM1 Master Control Units Operation Manual* for details on the Programming Console Differential Monitor procedure and to the *SSS Operation Manual: C-series PCs* for the SYSMAC Support Software procedure.

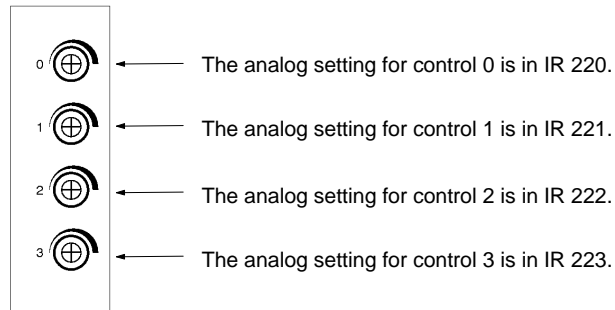
**2-5 Analog Settings (CQM1-CPU42-EV1/CPM1/CPM1A Only)**

In the CQM1-CPU42-EV1/CPM1/CPM1A PCs, the analog settings function automatically transfers the settings on the CPU Unit's adjustment switches to IR 220 through IR 223. This function is very useful when there are set values that need to be precisely adjusted during operation. These set values can be changed just by turning the adjustment switches on the CPU Unit.

The adjustment settings are stored in BCD and range from 0000 to 0200. Use a precision screwdriver to adjust the settings. (Turn the switches clockwise to increase the adjustment.)

**CQM1-CPU42-EV1 Settings**

The CQM1-CPU42-EV1 has four analog adjustment controls. The following diagram shows the CQM1-CPU42-EV1 adjustment controls and indicates the corresponding IR words that contain the adjustment settings.



In CQM1 CPU Units other than the CQM1-CPU42-EV1, IR 220 through IR 223 serve no special purpose. They can be used as work words in the program.

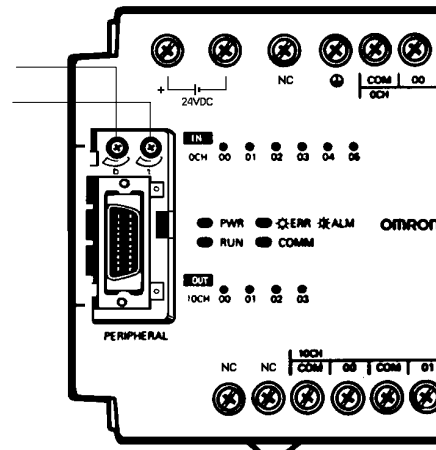
**Caution**

The CQM1-CPU42-EV1 continually refreshes IR 220 through IR 223 with the adjustment settings as long as the power is on. Do not overwrite the content of these words from the program or Peripheral Device.

**CPM1/CPM1A Settings**

The CPM1/CPM1A PCs have two analog adjustment controls. The following diagram shows the adjustment controls and indicates the corresponding SR words that contain the adjustment settings. Use a phillips-head screwdriver to adjust the settings.

The analog setting for control 0 is in SR 250.  
The analog setting for control 1 is in SR 251.



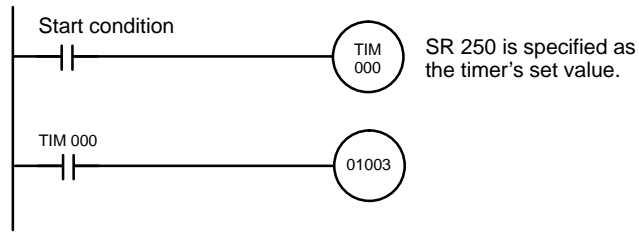
**Note** The above diagram shows the CPM1, but the settings are the same for CPM1A.

**Caution**

The analog setting may change with changing temperatures. Do not use the analog adjustment controls for applications that require a precise, fixed setting.

**CPM1/CPM1A Program Example**

The following ladder program uses the CPM1/CPM1A's analog settings. The analog setting in SR 250 (0000 to 0200 BCD) is determined by adjusting analog adjustment control 0. This value is used to adjust the timer's set value from 0.0 to 20.0 seconds.



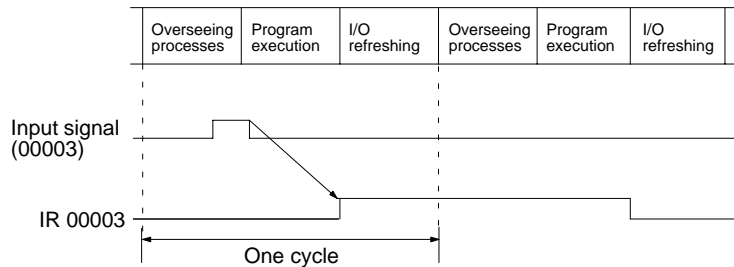
**2-6 Quick-response Inputs (CPM1/CPM1A Only)**

The CPM1/CPM1A have quick response inputs that can be used to enable inputting shorter signals.

All 10-point CPU Units have 2 quick-response input terminals and the 20-, 30-, and 40-point CPU Units have 4 quick-response input terminals. The same terminals are used for quick-response inputs and interrupt inputs.

**Quick-response Operation**

Quick-response inputs have an internal buffer, so input signals shorter than one cycle can be detected. Signals with a pulse width as short as 0.2 ms can be detected, regardless of their timing during the PC cycle.

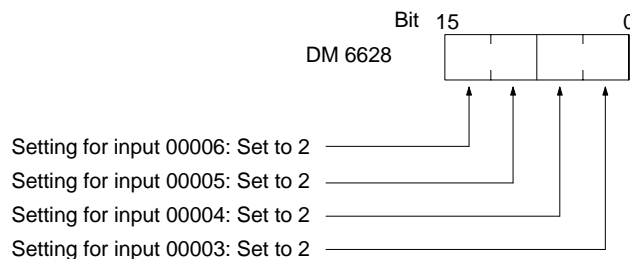


CPU Unit	Input bits	Min. input pulse width
10-point CPU Units	IR 00003 to IR 00004	0.2 ms
20-, 30-, 40-point CPU Units	IR 00003 to IR 00006	

**Setting Quick-response Inputs**

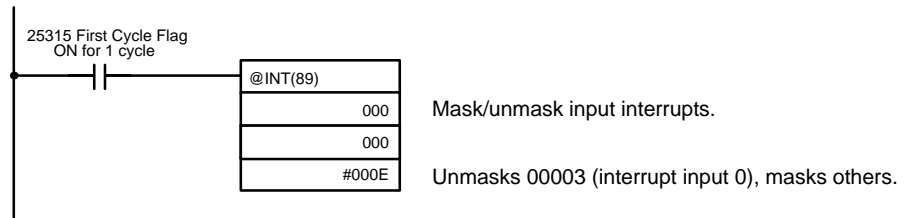
The input bits in the above table can be set as quick-response inputs in DM 6628, as shown in the following table.

Word	Settings
DM 6628	0: Normal input 1: Interrupt input 2: Quick-response input (Default setting: 0)



**Program Example**

In this example, DM 6628 has been set to 0002.





# SECTION 3

## Memory Areas

This section describes the structure of the PC memory areas, and explains how to use them. It also describes the CQM1 Memory Cassette operations used to transfer data between the CQM1 CPU Unit and a Memory Cassette.

3-1	CQM1 Memory Area Functions .....	134
3-1-1	Memory Area Structure .....	134
3-1-2	IR Area .....	135
3-1-3	SR Area .....	137
3-1-4	TR Area .....	137
3-1-5	HR Area .....	137
3-1-6	AR Area .....	138
3-1-7	LR Area .....	138
3-1-8	Timer/Counter Area .....	138
3-1-9	DM Area .....	139
3-1-10	UM Area .....	139
3-2	CPM1/CPM1A Memory Area Functions .....	139
3-2-1	Memory Area Structure .....	139
3-2-2	IR Area .....	140
3-2-3	SR Area .....	141
3-2-4	TR Area .....	142
3-2-5	HR Area .....	142
3-2-6	AR Area .....	142
3-2-7	LR Area .....	142
3-2-8	Timer/Counter Area .....	142
3-2-9	DM Area .....	142
3-3	SRM1 Memory Area Functions .....	143
3-3-1	Memory Area Structure .....	143
3-3-2	IR Area .....	144
3-3-3	SR Area .....	144
3-3-4	TR Area .....	144
3-3-5	HR Area .....	145
3-3-6	AR Area .....	145
3-3-7	LR Area .....	145
3-3-8	Timer/Counter Area .....	145
3-3-9	DM Area .....	145
3-4	SRM1 Flash Memory .....	145
3-5	Using Memory Cassettes (CQM1 Only) .....	146
3-5-1	Memory Cassettes and Contents .....	146
3-5-2	Memory Cassette Capacity and UM Area Size .....	147
3-5-3	Writing to the Memory Cassette .....	148
3-5-4	Reading from the Memory Cassette .....	148
3-5-5	Comparing Memory Cassette Contents .....	149

## 3-1 CQM1 Memory Area Functions

### 3-1-1 Memory Area Structure

The following memory areas can be used with the CQM1.

Data area		Size	Words	Bits	Function
IR area <sup>1</sup>	Input area	128, or 256 bits	IR 000 to IR 015	IR 00000 to IR 01515	CQM1-CPU11/21-E: Up to 8 words (128 bits) can be used for I/O bits. Up to 7 Units can be connected.
	Output area		IR 100 to IR 115	IR 10000 to IR 11515	CQM1-CPU4□-EV1: Up to 16 words (256 bits) can be used for I/O bits. Up to 11 Units can be connected.
	Work areas	2,720 bits min. <sup>2</sup>	IR 016 to IR 095	IR 01600 to IR 09515	Work bits do not have any specific function, and they can be freely used within the program.
			IR 116 to IR 195	IR 11600 to IR 19515	
IR 216 to IR 219			IR 21600 to IR 21915		
MACRO operand area <sup>1</sup>	Input area	64 bits	IR 096 to IR 099	IR 09600 to IR 09915	Used when the MACRO instruction, MCRO(99), is used. When the MACRO instruction is not used, these bits may be used as work bits.
	Output area	64 bits	IR 196 to IR 199	IR 19600 to IR 19915	
Analog SV area <sup>1</sup>		64 bits	IR 220 to IR 223	IR 22000 to IR 22315	CQM1-CPU42-EV1: Used to store the analog set values. (Cannot be used as work bits.) Can be used as work bits in other CPU Units.
High-speed Counter 0 PV <sup>1</sup>		32 bits	IR 230 to IR 231	IR 23000 to IR 23115	Used to store the present values of high-speed counter 0.
Port 1 and 2 Pulse Output PVs <sup>1</sup>		64 bits	IR 236 to IR 239	IR 23600 to IR 23915	CQM1-CPU43-EV1: Used to store the present values of pulse outputs for ports 1 and 2. (Cannot be used as work bits.) CQM1-CPU44-EV1: Used by the system. (Cannot be used as work bits.) Can be used as work bits in other CPU Units.
High-speed Counter 1 and 2 PVs <sup>1</sup>		64 bits	IR 232 to IR 235	IR 23200 to IR 23515	CQM1-CPU43/44-EV1: Used to store the present values of high-speed counters 1 and 2 for ports 1 and 2. (Cannot be used as work bits.) Can be used as work bits in other CPU Units.
Expansion Areas <sup>1</sup>		320 bits	IR 200 to IR 215 IR 240 to IR 243	IR 20000 to IR 21515 IR 24000 to IR 24315	These bits are expected to be used in planned function expansion.
SR area		184 bits	SR 244 to SR 255	SR 24400 to SR 25507	These bits serve specific functions such as flags and control bits. Can be used as work bits.
TR area		8 bits	---	TR 0 to TR 7	These bits are used to temporarily store ON/OFF status at program branches.
HR area		1,600 bits	HR 00 to HR 99	HR 0000 to HR 9915	These bits store data and retain their ON/OFF status when power is turned off.
AR area		448 bits	AR 00 to AR 27	AR 0000 to AR 2715	These bits serve specific functions such as flags and control bits.
LR area <sup>1</sup>		1,024 bits	LR 00 to LR 63	LR 0000 to LR 6315	Used for 1:1 data link through the RS-232 port.
Timer/Counter area <sup>3</sup>		512 bits	TC 000 to TC 511 (timer/counter numbers)		The same numbers are used for both timers and counters. TC 000 to TC 002 are used for interval timers.

Data area		Size	Words	Bits	Function
DM area	Read/write	1,024 words	DM 0000 to DM 1023	---	DM area data can be accessed in word units only. Word values are retained when the power is turned off.
		5,120 words	DM 1024 to DM 6143	---	Available in CQM1-CPU4□-EV1 CPU Units only. <sup>4</sup>
	Read-only <sup>5</sup>	425 words	DM 6144 to DM 6568	---	Cannot be overwritten from program.
	Error history area <sup>5</sup>	31 words	DM 6569 to DM 6599	---	Used to store the time of occurrence and error code of errors that occur.
	PC Setup <sup>5</sup>	56 words	DM 6600 to DM 6655	---	Used to store various parameters that control PC operation.
User program area (UM area)		3,200 or 7,200 words	---	---	Used to store the program. Retained when the power is turned off. CQM1-CPU11/21-E: 3,200 words CQM1-CPU4□-EV1: 7,200 words

- Note**
1. IR and LR bits that are not used for their allocated functions can be used as work bits.
  2. At least 2,720 bits can be used as work bits. The total number of bits that can be used depends on the configuration of the PC system.
  3. When accessing a PV, TC numbers are used as word data; when accessing Completion Flags, they are used as bit data.
  4. Although the CQM1-CPU11-E and CQM1-CPU21-E do not support DM 1024 through DM 6143, an error will not occur if they are addressed. Any attempt to write to these words will have no effect and any reads will produce all zeros.
  5. Data in DM 6144 to DM 6655 cannot be overwritten from the program.

### 3-1-2 IR Area

The functions of the IR area are explained below.

#### Input and Output Area

IR area bits are allocated to terminals on Input Units and Output Units. They reflect the ON/OFF status of input and output signals. Input bits begin at IR 00000, and output bits begin at IR 10000. With the CQM1, only IR 00000 through IR 01515 can be used as input bits and only IR 10000 through IR 11515 can be used as output bits.

For information on allocating input and output bits, refer to page 139.

- Note** Input bits cannot be used in output instructions. Do not use the same output bit in more than one OUT and/or OUT NOT instruction, or the program will not execute properly.

#### Work Areas

With the CQM1-CPU11/21-E and CQM1-CPU41-EV1 CPU Units, any of the bits between IR 001 and IR 243 not used for specific functions can be used as work bits. There are a few exceptions with the CQM1-CPU42/43/44-EV1 CPU Units, as shown in the following table.

CPU Unit	Bits Unavailable as Work Bits
CQM1-CPU42-EV1	IR 22000 to IR 22315
CQM1-CPU43/44-EV1	IR 23200 to IR 23915

The work bits can be used freely within the program. They can only be used within the program, however, and not for direct external I/O. Work bits are reset (i.e., turned OFF) when the CQM1 power supply is turned off or when operation begins or stops.

The bits in the ranges shown below have specific functions, but can still be used as work bits when their specific functions are not being used.

Range	Function
IR 001 to IR 015	When allocated to Input Units, these bits serve as input bits.
IR 096 to IR 099	When the MACRO instruction is used, these bits serve as operand and input bits.
IR 100 to IR 115	When allocated to Output Units, these bits serve as output bits.
IR 196 to IR 199	When the MACRO instruction is used, these bits serve as operand and output bits.
IR 220 to IR 223	In the CQM1-CPU42-EV1, these bits serve to store the analog SV. They can be used as work bits in other CPU Units.
IR 230 to IR 231	When high-speed counter 0 is used, these bits are used to store its present value.
IR 232 to IR 235	In the CQM1-CPU43/44-EV1, these bits are used to store the present values for high-speed counters 1 and 2. They can be used as work bits in other CPU Units.
IR 236 to IR 239	In the CQM1-CPU43-EV1, these bits are used to store the present values for pulse outputs from ports 1 and 2. In the CQM1-CPU44-EV1, they are used by the system. They can be used as work bits in other CPU Units.

IR 200 to IR 215 and IR 240 to IR 243 will be used in future functions, but they can be used as work bits for the time being.

LR 00 to LR 63 are used as link bits, but they can also be used as work bits when not linked to another CQM1.

### I/O Bit Allocation

I/O words are allocated in order from the left, beginning with IR 001 for the Input Unit and IR100 for the Output Unit. The CPU Unit's input points are allocated to IR 000. Even if the Input Unit and Output Unit are mounted at random the input words and output words are in separate portions of the IR area.

One word is allocated even for 8-point I/O Units. Bit usage for 8-point I/O Units are shown in the following table.

Unit	Bits 0 to 7	Bits 8 to 15
Input Unit	Input bits	Always OFF (0)
Output Unit	Output bits	Work bits

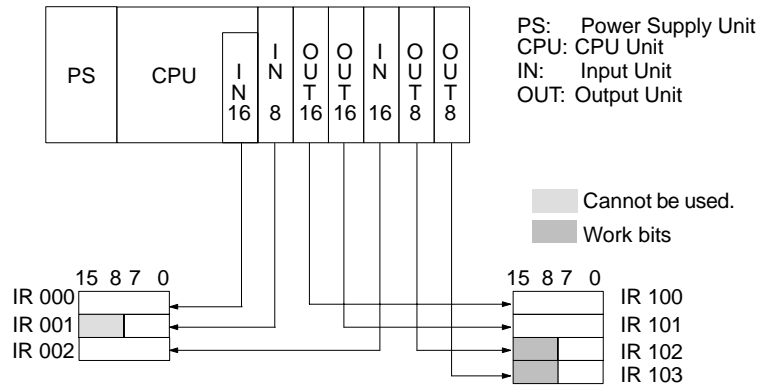
The number of I/O bits that can be allocated depends on the CQM1 CPU Unit being used, as shown in the following table.

CPU Unit model	Number of I/O bits
CQM1-CPU11/21-E	Up to 128 bits (8 words) can be used for I/O bits.
CQM1-CPU4□-EV1	Up to 256 bits (16 words) can be used for I/O bits.

As many as 256 I/O bits (16 words) can be allocated for the CQM1. A 16-point I/O Unit is allocated two words and must be counted as 16 points in calculating the total.

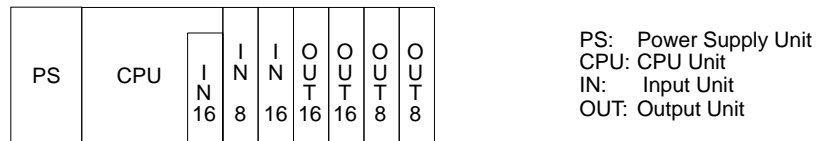
**Note** The SYSMAC-CPT will display or print only 12 I/O words in the I/O table display for the CQM1, i.e., IR 000 to IR 011. IR 012 to IR 015 will not be displayed for the CQM1-CPU4□-EV1 CPU Unit's I/O table, even though it has an I/O capacity of 256 pts (16 words). You will be able to input IR 012 to IR 015 for the CQM1-CPU4□-EV1 CPU Unit and the program will execute correctly for these bits, but "I" and "Q" will not be displayed to indicate input and output words/bits.

Word Allocation Example



All bits in words beyond the last input word and output word allocated can be used as work bits.

In order to make the word allocation easier to understand, and to help eliminate problems with noise, it is recommended that all Input Units be mounted directly following the CPU Unit. For the above example, the arrangement would be as shown below.



The number of allocated input words is stored in BCD in AR 2200 to AR 2207; the number of allocated output words in BCD in AR 2208 to AR 2215. The CQM1 PCs do not use an I/O table.

**Note** Up to 11 I/O Units can be mounted, regardless of the CPU Unit.

3-1-3 SR Area

These bits mainly serve as flags related to CQM1 operation. For details on the various bit functions, refer to relevant sections in this manual or to *Appendix C Memory Areas*.

SR 244 to SR 247 can also be used as work bits, when input interrupts are not used in Counter Mode.

3-1-4 TR Area

When a complex ladder diagram cannot be programmed in mnemonic code just as it is, these bits are used to temporarily store ON/OFF execution conditions at program branches. They are used only for mnemonic code. When programming directly with ladder diagrams using the Ladder Support Software (LSS) or the SYSMAC Support Software (SSS), TR bits are automatically processed for you.

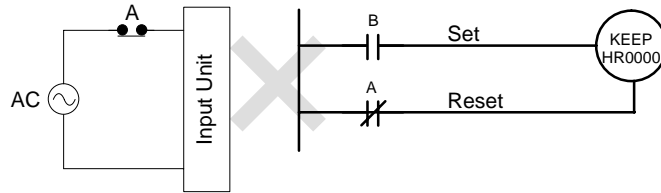
The same TR bits cannot be used more than once within the same instruction block, but can be used again in different instruction blocks. The ON/OFF status of TR bits cannot be monitored from a Peripheral Device.

Examples showing the use of TR bits in programming are provided on page 169.

3-1-5 HR Area

These bits retain their ON/OFF status even after the CQM1 power supply has been turned off or when operation begins or stops. They are used in the same way as work bits.

- Caution** Never use an input bit in a NC condition on the reset (R) for KEEP(11) when the input device uses an AC power supply (see diagram below). The delay in shutting down the PC's DC power supply relative to the AC power supply to the input device can cause the designate bit of KEEP(11) to be reset.



### 3-1-6 AR Area

These bits mainly serve as flags related to CQM1 operation. For details on the various bit functions, refer to relevant sections in this manual or to *Appendix C Memory Areas*.

With the exception of AR 23 (Power-off Counter), the status of AR words and bits is refreshed each cycle. (AR 23 is refreshed only for power interruptions.)

### 3-1-7 LR Area

When the CQM1 is linked one to one with another CQM1, these bits are used to share data. For details, refer to page 100.

LR bits can be used as work bits when not used for data links.

### 3-1-8 Timer/Counter Area

This area is used to manage timers and counters created with TIM, TIMH(15), CNT, and CNTR(12). The same numbers are used for both timers and counters and each number can be used only once in the user program. Do not use the same TC number twice even for different instructions.

TC number are used to create timers and counters, as well as to access Completion Flags and present values (PVs). If a TC number is designated for word data, it will access the present value (PV); if it is used for bit data, it access the Completion Flag for the timer/counter.

The Completion Flag turns ON when the PV of the timer/counter that is being used goes to 0.

Refer to instructions beginning on page 208 for details on timers and counters.

- Note**
1. TC numbers 000 through 015 and interrupt processing should be used for TIMH(15) whenever the cycle time is longer than 10 ms. Using other timer/counter numbers or not using interrupt processing will lead to inaccuracy in the high-speed timers. Interrupt processing can be set in DM 6629 of the PC Setup.
  2. When the input condition turns OFF for TIM or TIMH(15), the PV is reset and returns to the set value. The PV is also reset at the beginning of program execution or when the interlock condition goes OFF in a interlocked program section (IL-ILC). The PV for CNT or CNTR(12) is not reset like one for the timer instruction, but rather is reset only when the reset input goes ON.

### 3-1-9 DM Area

Data is accessed in word units. As shown below, the DM area contains both an area that can be freely used and areas with specific functions.

DM0000		This area has no specific functions and can be used freely. Both reading and writing are possible from the program.
DM1024		
(see note 1)	Fixed DM (see note 2)	This area cannot be written from the program. It is used for storing information that is not to be changed. Writing can be executed by means of peripheral devices only.
DM6144		
DM6569	Error Log	This area stores the error log. The user can only read this area, and cannot write into it.
DM6600	PC Setup (see note 2)	This area stores information related to CQM1 operation. The settings are made by means of peripheral devices.
DM6655		

- Note**
1. The CQM1-CPU11-E and CQM1-CPU21-E do not support DM 1024 through DM 6143.
  2. Turning ON pin 1 of the DIP switch on the CPU Unit will prevent writing even by means of peripheral devices.

Fixed DM contents, the PC Setup, the user program, and the instructions table can all be saved to and loaded from a Memory Cassette as a single unit. Refer to page 146 for details.

**Caution** Although the CQM1-CPU11-E and CQM1-CPU21-E do not support DM 1024 through DM 6143, an error will not occur if they are addressed. Any attempt to write to these words will have no effect and any reads will produce all zeros.

### 3-1-10 UM Area

The UM area stores the user's program. UM area contents can be read and written only as program data, and not as words. The following table shows the size of the UM area in the CQM1 CPU Units.

CPU Unit Model	UM area size
CQM1-CPU11/21-E	3.2 KW
CQM1-CPU4□-EV1	7.2 KW

## 3-2 CPM1/CPM1A Memory Area Functions

### 3-2-1 Memory Area Structure

The following memory areas can be used with the CPM1/CPM1A.

Data area	Words	Bits	Function
IR area <sup>1</sup>	Input area	IR 000 to IR 009 (10 words)	These bits can be allocated to the external I/O terminals.
	Output area	IR 010 to IR 019 (10 words)	
	Work area	IR 200 to IR 231 (32 words)	IR 20000 to IR 23115 (512 bits)
SR area	SR 232 to SR 255 (24 words)	SR 23200 to SR 25515 (384 bits)	These bits serve specific functions such as flags and control bits.
TR area	---	TR 0 to TR 7 (8 bits)	These bits are used to temporarily store ON/OFF status at program branches.
HR area <sup>2</sup>	HR 00 to HR 19 (20 words)	HR 0000 to HR 1915 (320 bits)	These bits store data and retain their ON/OFF status when power is turned off.

Data area		Words	Bits	Function
AR area <sup>2</sup>		AR 00 to AR 15 (16 words)	AR 0000 to AR 1515 (256 bits)	These bits serve specific functions such as flags and control bits.
LR area <sup>1</sup>		LR 00 to LR 15 (16 words)	LR 0000 to LR 1515 (256 bits)	Used for a 1:1 data link with another PC.
Timer/Counter area <sup>2</sup>		TC 000 to TC 127 (timer/counter numbers) <sup>3</sup>		The same numbers are used for both timers and counters.
DM area	Read/write <sup>2</sup>	DM 0000 to DM 0999 DM 1022 to DM 1023 (1,002 words)	---	DM area data can be accessed in word units only. Word values are retained when the power is turned off.
	Error log	DM 1000 to DM 1021 (22 words)	---	Used to store the time of occurrence and error code of errors that occur. These words can be used as ordinary read/write DM when the error log function isn't being used.
	Read-only <sup>4</sup>	DM 6144 to DM 6599 (456 words)	---	Cannot be overwritten from program.
	PC Setup <sup>4</sup>	DM 6600 to DM 6655 (56 words)	---	Used to store various parameters that control PC operation.

- Note**
1. IR and LR bits that are not used for their allocated functions can be used as work bits.
  2. The contents of the HR area, LR area, Counter area, and read/write DM area are backed up by a capacitor. The backup time varies with the ambient temperature, but at 25°C, the capacitor will back up memory for 20 days. If the power supply is off longer than the backup time, memory contents will be cleared and AR1314 will turn ON. (This flag turns ON when data can no longer be retained by the built-in capacitor.) Refer to 2-1-2 *Characteristics* in the *CPM1 and CPM1A Operation Manual* for a graph showing the backup time vs. temperature.
  3. When accessing a PV, TC numbers are used as word data; when accessing Completion Flags, they are used as bit data.
  4. Data in DM 6144 to DM 6655 cannot be overwritten from the program, but they can be changed from a Peripheral Device.

### 3-2-2 IR Area

The functions of the IR area are explained below.

#### I/O Bits

IR area bits from IR 00000 to IR 01915 are allocated to terminals on the CPU Unit and I/O Units. They reflect the ON/OFF status of input and output signals. Input bits begin at IR 00000, and output bits begin at IR 01000.



The following table shows which IR bits are allocated to the I/O terminals on the CPM1 CPU Units and CPM1-20EDR I/O Unit.

CPM1 CPU Unit	I/O	CPU Unit Terminals	I/O Unit Terminals
CPM1-10CDR-□	Inputs	6 points: 00000 to 00005	12 points: 00100 to 00111
	Outputs	4 points: 01000 to 01003	8 points: 01100 to 01107
CPM1-20CDR-□	Inputs	12 points: 00000 to 00011	12 points: 00100 to 00111
	Outputs	8 points: 01000 to 01007	8 points: 01100 to 01107
CPM1-30CDR-□	Inputs	18 points: 00000 to 00011, 00100 to 00105	12 points: 00200 to 00211
	Outputs	12 points: 01000 to 01007, 01100 to 01103	8 points: 01200 to 01207
CPM1-30CDR-□ -V1	Inputs	18 points: 00000 to 00011, 00100 to 00105	36 points: 00200 to 00211 00300 to 00311 00400 to 00411
	Outputs	12 points: 01000 to 01007, 01100 to 01103	24 points: 01200 to 01207 01300 to 01407 01400 to 01407

The following table shows which IR bits are allocated to the I/O terminals on the CPM1A's CPU Units and Expansion I/O Unit.

Number of I/O terminals on the CPU Unit	CPU Unit terminals		CPM1A-20ED□ Expansion I/O Unit Terminals						Power supply	Model number
	Inputs	Outputs	Inputs	Outputs	Inputs	Outputs	Inputs	Outputs		
10	6 points: 00000 to 00005	4 points: 01000 to 01003	---	---	---	---	---	---	AC	CPM1A-10CD□-A
									DC	CPM1A-10CD□-D
20	12 points: 00000 to 00011	8 points: 01000 to 01007	---	---	---	---	---	---	AC	CPM1A-20CD□-A
									DC	CPM1A-20CD□-D
30	18 points: 00000 to 00011 00100 to 00105	12 points: 01000 to 01007 01100 to 01103	12 points: 00200 to 00211	8 points: 01200 to 01207	12 points: 00300 to 00311	8 points: 01300 to 01307	12 points: 00400 to 00411	8 points: 01400 to 01407	AC	CPM1A-30CD□-A
									DC	CPM1A-30CD□-D
40	24 points: 00000 to 00011 00100 to 00111	16 points: 01000 to 01007 01100 to 01107	12 points: 00200 to 00211	8 points: 01200 to 01207	12 points: 00300 to 00311	8 points: 01300 to 01307	12 points: 00400 to 00411	8 points: 01400 to 01407	AC	CPM1A-40CD□-A
									DC	CPM1A-40CD□-D

### Work Bits

The work bits can be used freely within the program. They can only be used within the program, however, and not for direct external I/O.

### 3-2-3 SR Area

These bits mainly serve as flags related to CPM1/CPM1A operation or contain present and set values for various functions. For details on the various bit functions, refer to relevant sections in this manual or to *Appendix C Memory Areas*.

SR 244 to SR 247 can also be used as work bits, when input interrupts are not used in Counter Mode.

### 3-2-4 TR Area

When a complex ladder diagram cannot be programmed in mnemonic code just as it is, these bits are used to temporarily store ON/OFF execution conditions at program branches. They are used only for mnemonic code. When programming directly with ladder diagrams using the Ladder Support Software (LSS) or the SYSMAC Support Software (SSS), TR bits are automatically processed for you.

The same TR bits cannot be used more than once within the same instruction block, but can be used again in different instruction blocks. The ON/OFF status of TR bits cannot be monitored from a Peripheral Device.

Examples showing the use of TR bits in programming are provided on page 169.

### 3-2-5 HR Area

These bits retain their ON/OFF status even after the CPM1/CPM1A power supply has been turned off or when operation begins or stops. They are used in the same way as work bits.

### 3-2-6 AR Area

These bits mainly serve as flags related to CPM1/CPM1A operation. These bits retain their status even after the CPM1/CPM1A power supply has been turned off or when operation begins or stops. For details on the various bit functions, refer to relevant sections in this manual or to *Appendix C Memory Areas*.

### 3-2-7 LR Area

When the CPM1/CPM1A is linked one-to-one with another CPM1/CPM1A, a CQM1, an SRM1 or a C200HS PC, these bits are used to share data. For details, refer to page 101.

LR bits can be used as work bits when not used for data links.

### 3-2-8 Timer/Counter Area

This area is used to manage timers and counters created with TIM, TIMH(15), CNT, and CNTR(12). The same numbers are used for both timers and counters and each number can be used only once in the user program. Do not use the same TC number twice even for different instructions.

TC numbers are used to create timers and counters, as well as to access Completion Flags and present values (PVs). If a TC number is designated for word data, it will access the present value (PV); if it is used for bit data, it will access the Completion Flag for the timer/counter.

Refer to instructions beginning on page 208 for details on timers and counters.

### 3-2-9 DM Area

DM area data is accessed in word units only. The contents of the DM area are retained even after the CPM1/CPM1A power supply has been turned off or when operation begins or stops.

DM words DM 0000 through DM 0999, DM 1022, and DM 1023 can be used freely in the program; other DM words are allocated specific functions, described below.

#### Error Log

DM 1000 through DM 1021 contain the error log information. Refer to *Section 8 Troubleshooting* for details on the error log.

#### PC Setup

DM 6600 through DM 6655 contain the PC Setup. Refer to *1-5 PC Setup* for details.

## 3-3 SRM1 Memory Area Functions

### 3-3-1 Memory Area Structure

The following memory areas can be used with the SRM1.

Data area		Words	Bits	Function
IR area <sup>1</sup>	Input area	IR 000 to IR 009 (10 words)	IR 00000 to IR 00915 (160 bits)	These bits can be allocated to the external I/O terminals. The ON/OFF status of the I/O bits will be the same as the ON/OFF status of the I/O terminals Any I/O bits not used for I/O can be used as work bits.
	Output area	IR 010 to IR 019 (10 words)	IR 01000 to IR 01915 (160 bits)	
	Work area	IR 200 to IR 239 (40 words)	IR 20000 to IR 23915 (640 bits)	Work bits can be freely used within the program. IR 232 to IR 239 however, cannot be used as the the MACRO input area for the MACRO instruction.
SR area		SR 240 to SR 255 (16 words)	SR 24000 to SR 25507 (248 bits)	These bits serve as storage space for flags and function set values/present values for SRM1 operation. Refer to <i>SR Area</i> .
TR area		---	TR 0 to TR 7 (8 bits)	When a complicated ladder diagram cannot be recorded as a mnemonic these bits are used to temporarily store ON/OFF status at program branches. These temporary bits cannot be used within the same block but if the blocks are different several may be used. The ON/OFF status of these bits cannot be monitored using the monitoring function of a peripheral device.
HR area <sup>2</sup>		HR 00 to HR 19 (20 words)	HR 0000 to HR 1915 (320 bits)	These bits store data and retain their ON/OFF status when power is turned off, or operation starts or stops. They are used in the same way as work bits.
AR area <sup>2</sup>		AR 00 to AR 15 (16 words)	AR 0000 to AR 1515 (256 bits)	These bits serve specific functions such as flags and control bits. AR 04 to 07 are used as slaves. Refer to <i>AR Area</i> .
LR area <sup>1</sup>		LR 00 to LR 15 (16 words)	LR 0000 to LR 1515 (256 bits)	Used for a 1:1 data link with another SRM1, CQM1 or C200HS PC.
Timer/Counter area <sup>2</sup>		TC 000 to TC 127 (timer/counter numbers) <sup>3</sup>		Timers and counter use the TIM, TIMH(15), CNT and CNTR(12) instructions. The same numbers are used for both timers and counters.  Timer/counter numbers should be specified as bits when dealing with timer/counter present values. The counter data will be stored even when the SRM1 power is turned off or operation is stopped or started.  When timer/counter are treated as up-flags the number should be specified as relay data.

Data area		Words	Bits	Function
DM area	Read/write <sup>2</sup>	DM 0000 to DM 1999 (2,000 words)	---	DM area data can be accessed in word units only. Word values are retained when the power is turned off, or operation started or stopped.  Read/write areas can be read and written freely within the program.
	Error log <sup>4</sup>	DM 2000 to DM 2021 (22 words)	---	Used to store the time of occurrence and error code of errors that occur. Refer to 5-5 <i>Coding Right-hand Instructions</i> .
	Read-only <sup>4</sup>	DM 6144 to DM 6599 (456 words)	---	Cannot be overwritten from program.
	PC Setup <sup>4</sup>	DM 6600 to DM 6655 (56 words)	---	Used to store various parameters that control PC operation.

- Note**
1. IR and LR bits that are not used for their allocated functions can be used as work bits.
  2. The contents of the HR area, LR area, Counter area, and read/write DM area are backed up by a capacitor. At 25°C, the capacitor will back up memory for 20 days. Refer to 2-1-2 *Characteristics* in the *SRM1 Master Control Unit Operation Manual* for a graph showing the backup time vs. temperature.
  3. When accessing a PV, TC numbers are used as word data; when accessing Completion Flags, they are used as bit data.
  4. Data in DM 6144 to DM 6655 cannot be overwritten from the program, but they can be changed from a Peripheral Device.

### 3-3-2 IR Area

The functions of the IR area are explained below.

#### I/O Bits

IR area bits from IR 00000 to IR 01915 are allocated to terminals on the CPU Unit and I/O Unit. They reflect the ON/OFF status of input and output signals. Input bits begin at IR 00000, and output bits begin at IR 01000.

Refer to 1-4 *I/O and Data Area Allocation* in the *SRM1 Master Control Units Operation Manual* for further details.

#### Work Bits

The work bits can be used freely within the program. They can only be used within the program, however, and not for direct external I/O.

### 3-3-3 SR Area

These bits mainly serve as flags related to SRM1 operation or contain present and set values for various functions. For details on the various bit functions, refer to relevant sections in this manual or to *Appendix C Memory Areas*.

SR 240 to SR 247 and SR250, 251 can also be used as work bits, when input interrupts are not used in Counter Mode. SR232 to SR239 can also be used as work bits when the MCRO (99) instruction is not being used.

### 3-3-4 TR Area

When a complex ladder diagram cannot be programmed in mnemonic code just as it is, these bits are used to temporarily store ON/OFF execution conditions at program branches. They are used only for mnemonic code. When programming directly with ladder diagrams using the Ladder Support Software (LSS) or the SYSMAC Support Software (SSS), TR bits are automatically processed for you.

The same TR bits cannot be used more than once within the same instruction block, but can be used again in different instruction blocks. The ON/OFF status of TR bits cannot be monitored from a Peripheral Device.

Examples showing the use of TR bits in programming are provided on page 169.

### 3-3-5 HR Area

These bits retain their ON/OFF status even after the SRM1 power supply has been turned off or when operation begins or stops. They are used in the same way as work bits.

### 3-3-6 AR Area

These bits mainly serve as flags related to SRM1 operation. These bits retain their status even after the SRM1 power supply has been turned off or when operation begins or stops. For details on the various bit functions, refer to relevant sections in this manual or to *Appendix C Memory Areas*.

### 3-3-7 LR Area

When the SRM1 is linked one-to-one with another SRM1, a CQM1, an CPM1/CPM1A or a C200HS PC, these bits are used to share data. For details, refer to page 101.

LR bits can be used as work bits when not used for data links.

### 3-3-8 Timer/Counter Area

This area is used to manage timers and counters created with TIM, TIMH(15), CNT, and CNTR(12). The same numbers are used for both timers and counters and each number can be used only once in the user program. Do not use the same TC number twice even for different instructions.

TC numbers are used to create timers and counters, as well as to access Completion Flags and present values (PVs). If a TC number is designated for word data, it will access the present value (PV); if it is used for bit data, it will access the Completion Flag for the timer/counter.

Refer to instructions beginning on page 208 for details on timers and counters.

### 3-3-9 DM Area

DM area data is accessed in word units only. The contents of the DM area are retained even after the SRM1 power supply has been turned off or when operation begins or stops.

DM words DM 0000 through DM 1999, DM 2022, and DM 2047 can be used freely in the program; other DM words are allocated specific functions, described below.

#### Error Log

DM 2000 through DM 2021 contain the error log information. Refer to *Section 8 Troubleshooting* for details on the error log.

#### PC Setup

DM 6600 through DM 6655 contain the PC Setup. Refer to *1-5 PC Setup* for details.

## 3-4 SRM1 Flash Memory

The following settings must be made to use the flash memory area for SRM1 PCs.

#### Writing Data

In order to write the contents of the UM area, the DM read-only area (DM 6144 to DM 6599, and the PC Setup area (DM 6600 to DM 6655) to the flash memory, either one of the following operations must be performed.

- Switch the SRM1 to either the MONITOR or PROGRAM mode.
- Turn the power to the SRM1 off and on again.

**Note** If changes are made to the above memory areas, they are not written to the flash memory, and the power is switched off for 20 days or more (at 25°C), the changes (in RAM) will be lost. If this occurs, the unchanged contents will be read from the flash memory when the PC is started again.

**Changing Memory Areas**

When operating the SRM1 for the first time after changes have been made to the UM area, the DM read only area (DM 6144 to DM 6599, and the PC Setup area (DM 6600 to DM 6655), beware of the effect resulting from the SRM1's delay in the operation on other devices.

The first operation for the SRM1 after the above memory areas have been changed will be a maximum of 850 ms later than the normal first operation without changes.

**Cycle Times**

A cycle time overflow warning will not be issued when any of the following operations are performed in either MONITOR or OPERATION modes. Be careful of the effect of using online editing on SRM1 I/O response time.

- Changes to the program using online editing.
- Changes to the read-only DM area (DM 6144 to DM 6599.)
- Changes to the PC Setup area (DM 6600 to DM 6655.)

When any of the above operations are performed, the SRM1 cycle time will be increased by a maximum of 850 ms. During this tiny interrupts will be disabled while the program or memory contents is written.

## 3-5 Using Memory Cassettes (CQM1 Only)

When the optional Memory Cassette is used, the PC Setup, user's program, fixed DM, and the instructions tabled can be placed in ROM. This prevents unwanted changes from being accidentally written. In addition, when changing control processes, the settings and the program can be easily changed by simply replacing the Memory Cassette.

This section explains how to read, write, and compare information to and from the Memory Cassette.

### 3-5-1 Memory Cassettes and Contents

**Memory Cassettes**

There are six types of Memory Cassette, as shown in the following table.

Model	Remarks
CQM1-ME04K	EEPROM type (without clock) 4K words
CQM1-ME04R	EEPROM type (with clock) 4K words
CQM1-MP08K	EPR0M type (without clock) 8K words
CQM1-MP08R	EPR0M type (with clock) 8K words
CQM1-ME08K	EEPROM type (without clock) 8K words
CQM1-ME08R	EEPROM type (with clock) 8K words

The following EEPROM chips (sold separately) are required for EPROM-type Memory Cassettes.

Model	ROM version	Capacity	Access speed
ROM-ID-B	27128 or equivalent	8K words	150 ns
ROM-JD-B	27256 or equivalent	16K words	150 ns
ROM-KD-B	27512 or equivalent	32K words	150 ns

For the CQM1 CPU Units, 8K words max. of information can be written to the Memory Cassette. Therefore, any of the EPROM chips listed above would have sufficient capacity, and the choice can be made strictly on the basis of ease of purchase.

For instructions on using Memory Cassettes, refer to *CQM1 Operation Manual*.

**Contents**

The information that can be written to a Memory Cassette is shown in the following table.

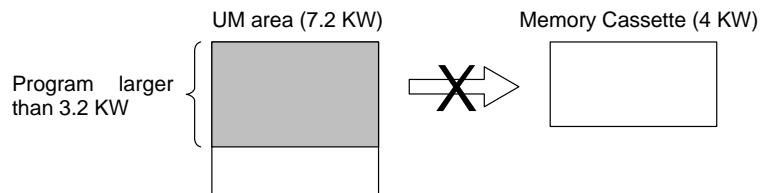
Information	Contents
Fixed DM	Fixed Data Memory cannot be written from the program. The range is DM 6144 to DM 6568. These words are available for the user.
PC Setup	The PC Setup sets the operating parameters of the CQM1 and it stored in DM 6600 to DM 6655.
Instructions Table	The instructions table assigns expansion instruction to function codes to enable using them in programming.
User Program Memory (UM)	The UM area holds the user's program.

The above information is cannot be read, written, or compared individually, and must be treated as a single unit.

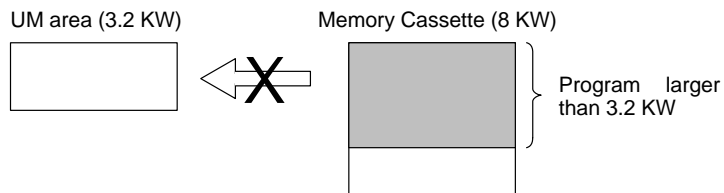
### 3-5-2 Memory Cassette Capacity and UM Area Size

A non-fatal error will occur if an attempt is made to transfer a program that is too large. There are two cases in which this can occur.

- 1, 2, 3...
1. When a 4-KW EEPROM Memory Cassette is installed in a CQM1-CPU4□-EV1 CPU Unit with a 7.2-KW UM area, programs up to 3.2 KW long can be written to the Memory Cassette. A non-fatal error will occur if an attempt is made to write a program larger than 3.2 KW to the Memory Cassette.



2. When a 8-KW or larger Memory Cassette is installed in a CQM1-CPU11/21-E CPU Unit with a 3.2-KW UM area, programs up to 3.2 KW long can be read from the Memory Cassette. A non-fatal error will occur if an attempt is made to read a program larger than 3.2 KW from the Memory Cassette.



**Note** The two transfers shown above would be completed normally if the program were 3.2 KW or smaller.

The approximate sizes of the programs in the UM area and Memory Cassette can be determined by the content of AR 15, as shown in the following table.

Program location	Bits	Content	Meaning
Memory Cassette	AR 1500 to AR 1507	00	No Memory Cassette is installed or no program is saved in the Memory Cassette.
		04	The program is less than 3.2 KW long and can be read from any CQM1 CPU Unit.
		08	The program is less than 7.2 KW long and can be read from CQM1-CPU4□-E/-EV1 CPU Units only.
UM area	AR 1508 to AR 1515	04	The program is less than 3.2 KW long and can be written to any Memory Cassette.
		08	The program is less than 7.2 KW long and can be written to 8-KW or larger Memory Cassettes only.

In CQM1-CPU11/21-E CPU Units, the content of AR 1508 to AR 1515 is normally 04, and the content of AR 1500 to AR 1507 is normally 04 when a 4-KW Memory Cassette is installed.

The size of the program indicated in AR 15 does not include the NOP(00) instructions after END(01), but will include any instructions other than NOP(00). Be sure to clear any unneeded instructions after END(01) to get an accurate measurement of the program's size.

### 3-5-3 Writing to the Memory Cassette

Writing to an EPROM-type Memory Cassette is executed using either the Ladder Support Software (LSS) or SYSMAC Support Software (SSS), and a PROM Writer. For instructions on using LSS, refer to the *LSS Operation Manual*. For instructions on using SSS, refer to the *SSS Operation Manual: C-series PCs*.

Follow the procedure outlined below for writing.

- 1, 2, 3... 1. Check to see that the write-protect switch on the Memory Cassette is OFF (i.e., writing enabled). If the switch is ON (i.e., writing not enabled), then turn the CQM1 power supply off and remove the Memory Cassette before changing the switch.
2. Check to see that the CQM1 is in PROGRAM mode. If it is in either RUN or MONITOR mode, use the LSS/SSS to change the mode.
3. Turn ON AR 1400 from the LSS/SSS. The information will be written from the CQM1 to the Memory Cassette.
4. With the LSS/SSS, clear the forced-set status to turn OFF AR 1400, which will not change even after the operation is complete. (If the Programming Console is being used, AR 1400 will automatically turn OFF.)

 **Caution** Data cannot be written to the Memory Cassette if a memory error has occurred.

**Note** If an error occurs while data is being transmitted, a non-fatal error (FAL 9D) will be generated and the appropriate AR bit (from AR 1412 to AR 1415) will turn ON/OFF. If this occurs, refer to *Section 8 Troubleshooting* and make the necessary corrections.

### 3-5-4 Reading from the Memory Cassette

There are two methods for reading data from the Memory Cassette to the CQM1: By using a Peripheral Device (e.g., LSS/SSS) or by automatic reading the contents when the CQM1 is started up.

If the program on the Memory Cassette has expansion instructions with function codes different from the default settings, make sure that pin 4 of the CPU Unit's DIP switch is ON (indicating user-allocated function codes).



**Note** When data is being read from the Memory Cassette to the CQM1, pin 1 of the DIP switch on the CQM1 must be OFF (i.e., writing to the DM must be enabled). Turn off the power to the CQM1 before turning this pin OFF.


Reading from the Memory Cassette can be executed regardless of the type of Memory Cassette.

If an error occurs while data is being transmitted, a non-fatal error (FAL 9D) will be generated and the appropriate AR bit (from AR 1412 to AR 1415 will turn ON/OFF. (If this occurs, refer to the Troubleshooting section and make the necessary correction.)

**Peripheral Device Operation** To use a peripheral device to read from the Memory Cassette, follow the procedure outlined below.

- 1, 2, 3...**
1. Check to see that the CQM1 is in PROGRAM mode. If it is in either RUN or MONITOR mode, use the peripheral device to change the mode.
  2. Use the peripheral device to turn ON AR 1401. The information will be read from the Memory Cassette to the CQM1.
  3. With the LSS/SSS, clear the forced-set status to turn OFF AR 1401, which will not change even after the operation is complete. (If the Programming Console is being used, AR 1401 will automatically turn OFF.)

**Automatic Reading** If pin 2 of the DIP switch on the CQM1 is turned ON (auto-boot), then data will automatically be read from the Memory Cassette when the power supply is turned on to the CQM1. Operation will not be possible if an error occurs during transfer of data between the Memory Cassette and CQM1 memory.

 **Caution** Be absolutely sure that the power is turned off before changing CQM1 DIP switch settings.

### 3-5-5 Comparing Memory Cassette Contents

The contents of the Memory Cassette can be compared to the contents of the CQM1 memory to check to see if they are the same. This comparison can be performed for any type of Memory Cassette.

Use the following procedure.

- 1, 2, 3...**
1. Check to see that the CQM1 is in PROGRAM mode. If it is in either RUN or MONITOR mode, use the peripheral device to change to the PROGRAM mode.
  2. Turn ON AR 1402 from the peripheral device. The contents of the Memory Cassette will be compared to the contents of CQM1 memory.
  3. With the LSS/SSS, clear the forced-set status to turn OFF AR 1402, which will not change even after the operation is complete. (If the Programming Console is being used, AR 1402 will automatically turn OFF.)
  4. Check the status of AR 1403 to see the results of the comparison. AR 1403 will be ON if the contents were not the same or if the comparison was not possible because the CQM1 was not in PROGRAM mode. If AR 1403 is OFF, the comparison was successful and the contents were the same.

AR 1403 cannot be controlled from the program or from a peripheral device. It is controlled by the results of comparison only.

If a comparison is attempted with the CQM1 in any mode but PROGRAM mode, a non-fatal error will occur (FAL 9D) and AR 1412 will turn ON. Although AR 1403 will also turn ON, no comparison will have been performed. AR 1403 will also turn ON if a comparison is attempted without a Memory Cassette mounted in the CQM1.

# SECTION 4

## Ladder-diagram Programming

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

4-1	Basic Procedure .....	152
4-2	Instruction Terminology .....	152
4-3	Basic Ladder Diagrams .....	153
4-3-1	Basic Terms .....	153
4-3-2	Mnemonic Code .....	154
4-3-3	Ladder Instructions .....	155
4-3-4	OUTPUT and OUTPUT NOT .....	158
4-3-5	The END Instruction .....	159
4-3-6	Logic Block Instructions .....	159
4-3-7	Coding Multiple Right-hand Instructions .....	168
4-3-8	Branching Instruction Lines .....	168
4-3-9	Jumps .....	172
4-4	Controlling Bit Status .....	173
4-4-1	SET and RESET .....	173
4-4-2	DIFFERENTIATE UP and DIFFERENTIATE DOWN .....	174
4-4-3	KEEP .....	174
4-4-4	Self-maintaining Bits (Seal) .....	175
4-5	Work Bits (Internal Relays) .....	175
4-6	Programming Precautions .....	177
4-7	Program Execution .....	179

## 4-1 Basic Procedure

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

- 1, 2, 3... 1. Obtain a list of all I/O devices and the I/O points that have been assigned to them and prepare a table that shows the I/O bit allocated to each I/O device.
2. If you are using LR bits to link two PCs, prepare sheet showing the used of these bits.
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 *5-15 Timer and Counter Instructions*; jump numbers are described later in this section.)
5. Draw the ladder diagram.
6. Input the program into the CPU Unit. 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 the *CQM1 Operation Manual*, *CPM1 Operation Manual*, the *CPM1A Operation Manual* and the *SRM1 Master Control Units Manual* and via the SSS in the *SSS Operation Manual: C-series PCs*.

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 the *CQM1 Operation Manual*, *CPM1 Operation Manual*, the *CPM1A Operation Manual*, the *SRM1 Master Control Units Manual*, and *SSS Operation Manual: C-series PCs*. *Section 8 Troubleshooting* also provides information required for debugging.

## 4-2 Instruction Terminology

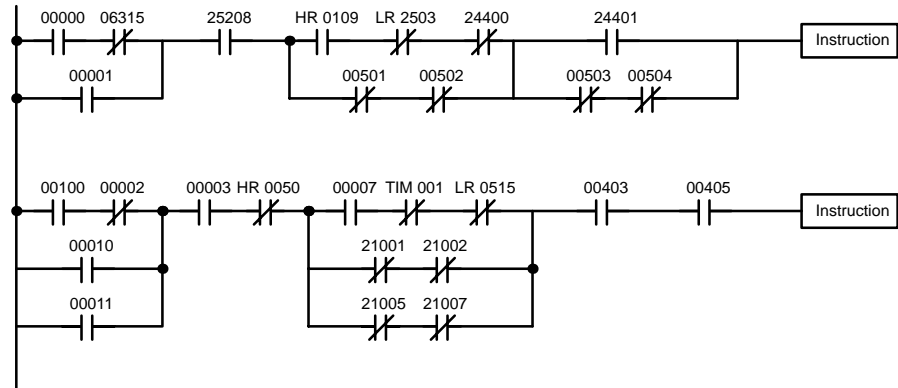
There are basically two types of instructions used in ladder-diagram programming: instructions that correspond to the conditions on the ladder diagram and are used in instruction form only when converting a program to mnemonic code and instructions that are used on the right side of the ladder diagram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



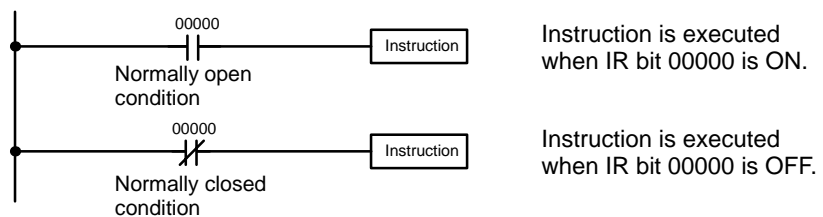
As shown in the diagram above, instruction lines can branch apart and they can join back together. The vertical pairs of lines are called conditions. Conditions without diagonal lines through them are called normally open conditions and correspond to a LOAD, AND, or OR instruction. The conditions with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



<b>Execution Conditions</b>	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.
<b>Operand Bits</b>	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 <i>4-3-8 Branching Instruction Lines</i> for details.
<b>Logic Blocks</b>	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.
<b>Instruction Block</b>	An instruction block consists of all the instructions that are interconnected across the ladder diagram. One instruction block thus consists of all the instructions between where you can draw a horizontal line across the ladder diagram without intersecting any vertical lines and the next place where you can draw the same type of horizontal line.

## 4-3-2 Mnemonic Code

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

<b>Program Memory Structure</b>	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	
00006	MOV(21)	
		000
		DM 0000
00007	CMP(20)	
		DM 0000
		HR 00
00008	AND	25505
00009	OUT	10000
00010	MOV(21)	
		DM 0000
		DM 0500
00011	LD	00502
00012	AND	00005
00013	OUT	10003

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.

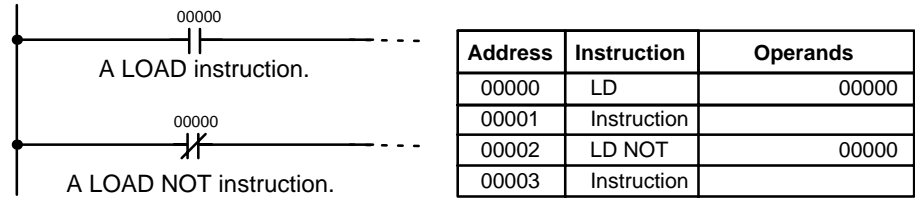
When programming, addresses are automatically displayed and do not have to be input unless for some reason a different location is desired for the instruction. When converting to mnemonic code, it is best to start at Program Memory address 00000 unless there is a specific reason for starting elsewhere.

### 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**

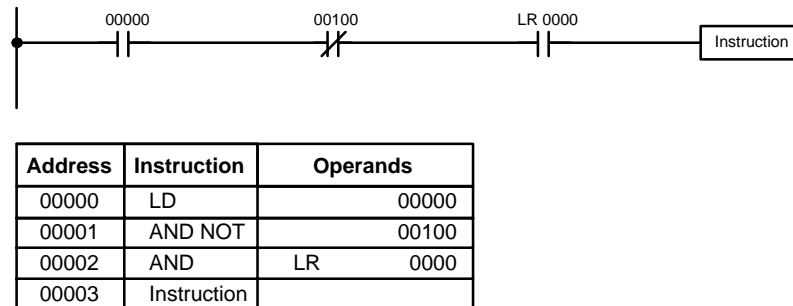
The first condition that starts any logic block within a ladder diagram corresponds to a LOAD or LOAD NOT instruction. Each of these instruction requires one line of mnemonic code. "Instruction" is used as a dummy instruction in the following examples and could be any of the right-hand instructions described later in this manual.



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.



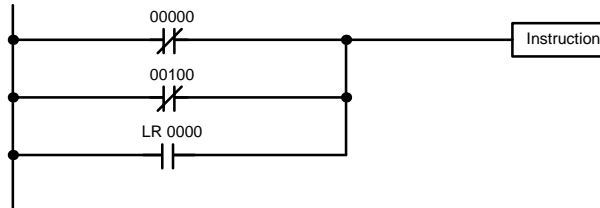
The instruction would have an ON execution condition only when all three conditions are ON, i.e., when IR 00000 was ON, IR 00100 was OFF, and LR 0000 was ON.

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



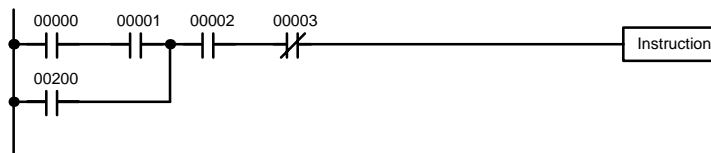
Address	Instruction	Operands
00000	LD NOT	00000
00001	OR NOT	00100
00002	OR	LR 0000
00003	Instruction	

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	

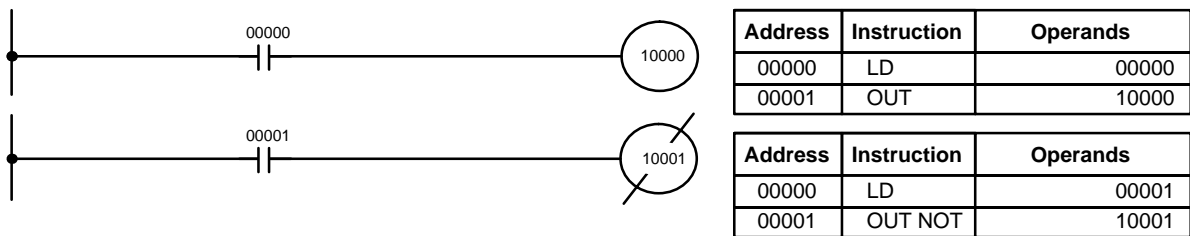
Here, an AND is taken between the status of IR 00000 and that of IR 00001 to determine the execution condition for an OR with the status of IR 00200. The result of this operation determines the execution condition for an AND with the status of IR 00002, which in turn determines the execution condition 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.

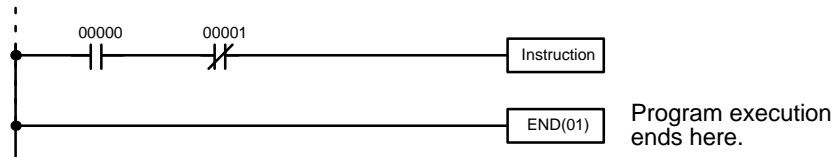


In the above examples, IR 10000 will be ON as long as IR 00000 is ON and IR 10001 will be OFF as long as IR 00001 is ON. Here, IR 00000 and IR 00001 would be input bits and IR 10000 and IR 10001 output bits assigned to the Units controlled by the PC, i.e., the signals coming in through the input points assigned IR 00000 and IR 00001 are controlling the output points assigned IR 10000 and IR 10001, 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 5-15-1 *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 Unit scans the program, it executes all instructions up to the first END instruction before returning to the beginning of the program and beginning execution again. Although an END instruction can be placed at any point in a program, which is sometimes done when debugging, no instructions past the first END instruction will be executed until it is removed. The number following the END instruction in the mnemonic code is its function code, which is used when inputted most instruction into the PC. These are described later. The END instruction requires no operands and no conditions can be placed on the same instruction line with it.



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.

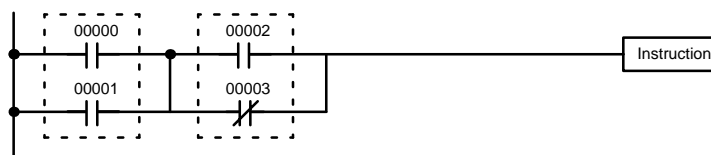
Now you have all of the instructions required to write simple input-output programs. Before we finish with ladder diagram basic and go onto inputting the program into the PC, let's look at logic block instruction (AND LOAD and OR LOAD), which are sometimes necessary even with simple diagrams.

### 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).

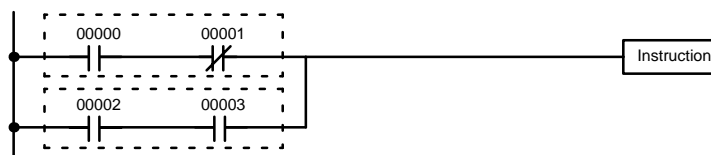
The above ladder diagram cannot, however, be converted to mnemonic code using AND and OR instructions alone. If an AND between IR 00002 and the results of an OR between IR 00000 and IR 00001 is attempted, the OR NOT between IR 00002 and IR 00003 is lost and the OR NOT ends up being an OR NOT between 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 it 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.



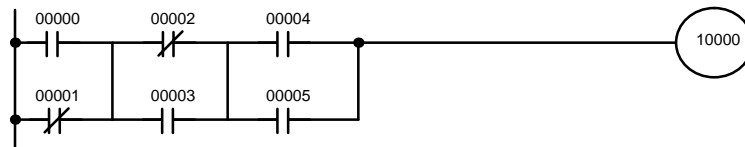
Address	Instruction	Operands
00000	LD	00000
00001	AND NOT	00001
00002	LD	00002
00003	AND	00003
00004	OR LD	---

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.

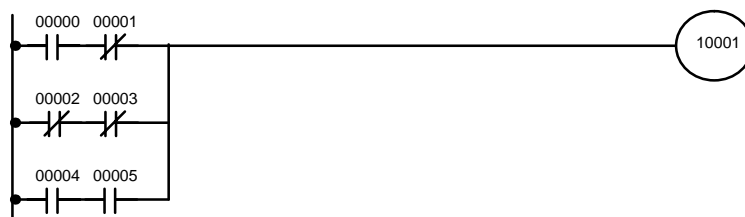


Address	Instruction	Operands
00000	LD	00000
00001	OR NOT	00001
00002	LD NOT	00002
00003	OR	00003
00004	AND LD	—
00005	LD	00004
00006	OR	00005
00007	AND LD	—
00008	OUT	10000

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	10000

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	10001

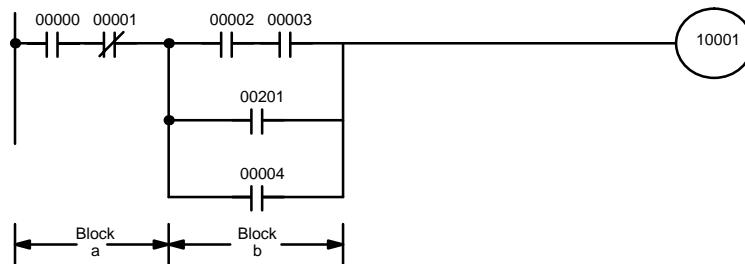
Address	Instruction	Operands
00000	LD	00000
00001	AND NOT	00001
00002	LD NOT	00002
00003	AND NOT	00003
00004	LD	00004
00005	AND	00005
00006	OR LD	—
00007	OR LD	—
00008	OUT	10001

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

**Combining AND LOAD and OR LOAD**

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

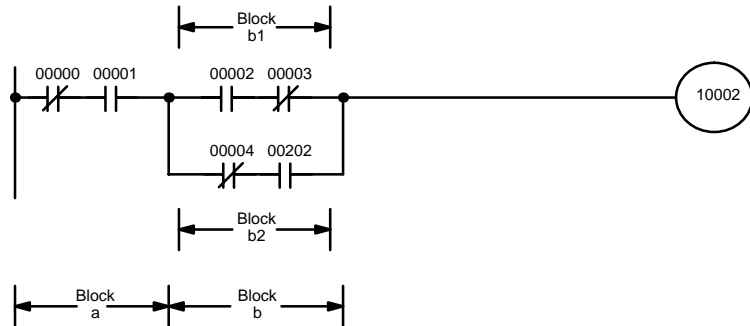
The following diagram contains only two logic blocks as shown. It is not necessary to further separate block b components, because it can be coded directly using only AND and OR.



Address	Instruction	Operands
00000	LD	00000
00001	AND NOT	00001
00002	LD	00002
00003	AND	00003
00004	OR	00201
00005	OR	00004
00006	AND LD	—
00007	OUT	10001

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.



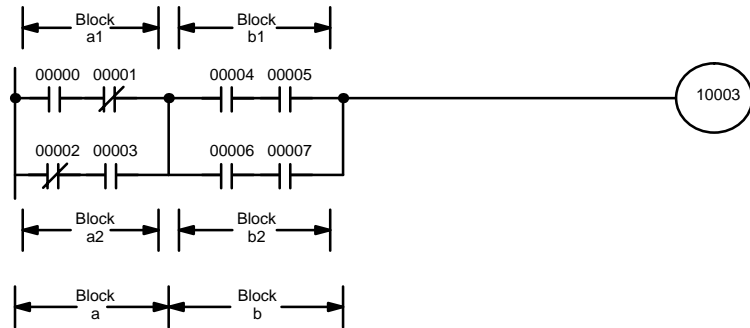
Address	Instruction	Operands
00000	LD NOT	00000
00001	AND	00001
00002	LD	00002
00003	AND NOT	00003
00004	LD NOT	00004
00005	AND	00202
00006	OR LD	—
00007	AND LD	—
00008	OUT	10002

**Complicated Diagrams**

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

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

The following diagram must be broken down into two blocks and each of these then broken into two blocks before it can be coded. As shown below, blocks a and b require an AND LOAD. Before AND LOAD can be used, however, OR LOAD must be used to combine the top and bottom blocks on both sides, i.e., to combine a1 and a2; b1 and b2.



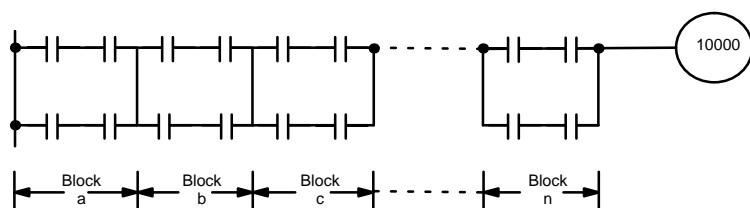
Address	Instruction	Operands
00000	LD	00000
00001	AND NOT	00001
00002	LD NOT	00002
00003	AND	00003
00004	OR LD	—
00005	LD	00004
00006	AND	00005
00007	LD	00006
00008	AND	00007
00009	OR LD	—
00010	AND LD	—
00011	OUT	10003

Blocks a1 and a2

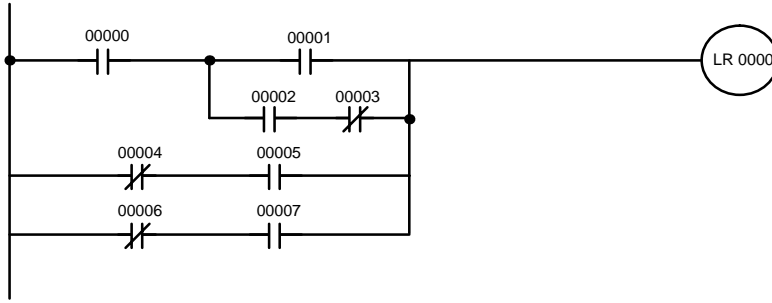
Blocks b1 and b2

Blocks a and b

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.

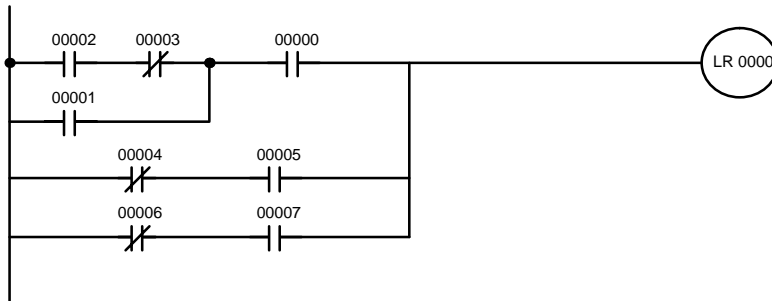


The following diagram requires an OR LOAD followed by an AND LOAD to code the top of the three blocks, and then two more OR LOADs to complete the mnemonic code.



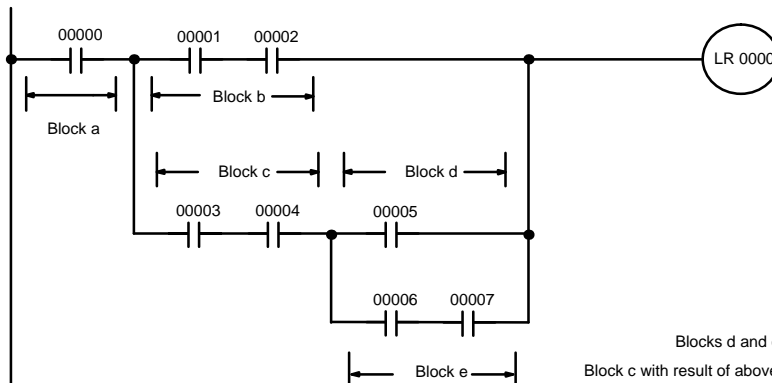
Address	Instruction	Operands
00000	LD	00000
00001	LD	00001
00002	LD	00002
00003	AND NOT	00003
00004	OR LD	--
00005	AND LD	--
00006	LD NOT	00004
00007	AND	00005
00008	OR LD	--
00009	LD NOT	00006
00010	AND	00007
00011	OR LD	--
00012	OUT	LR 0000

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.

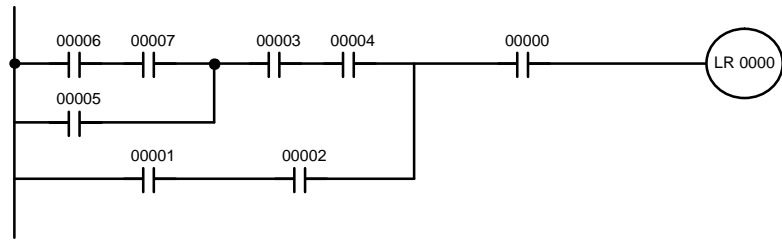


Blocks d and e  
 Block c with result of above  
 Block b with result of above  
 Block a with result of above

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

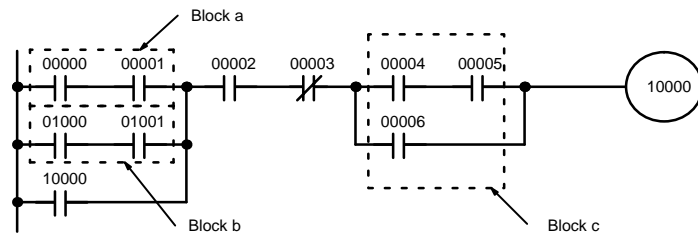


Again, this diagram can be redrawn as follows to simplify program structure and coding and to save memory space.

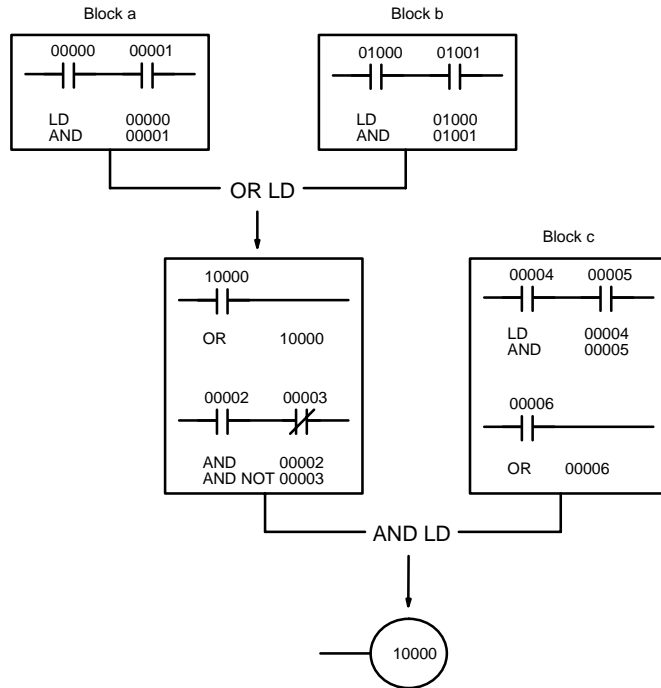


Address	Instruction	Operands
00000	LD	00006
00001	AND	00007
00002	OR	00005
00003	AND	00003
00004	AND	00004
00005	LD	00001
00006	AND	00002
00007	OR LD	--
00008	AND	00000
00009	OUT	LR 0000

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



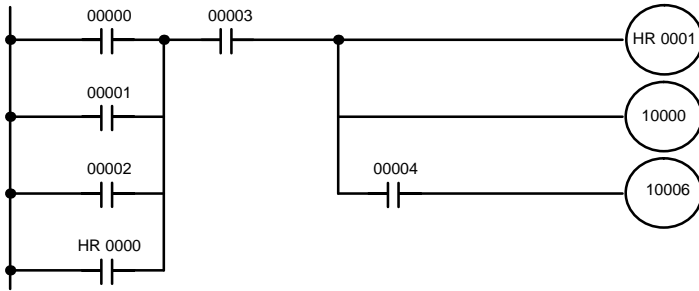
The first logic block instruction is used to combine the execution conditions resulting from blocks a and b, and the second one is to combine the execution condition of block c with the execution condition resulting from the normally closed condition assigned IR 00003. The rest of the diagram can be coded with OR, AND, and AND NOT instructions. The logical flow for this and the resulting code are shown below.



Address	Instruction	Operands
00000	LD	00000
00001	AND	00001
00002	LD	01000
00003	AND	01001
00004	OR LD	--
00005	OR	10000
00006	AND	00002
00007	AND NOT	00003
00008	LD	00004
00009	AND	00005
00010	OR	00006
00011	AND LD	--
00012	OUT	10000

### 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	10000
00007	AND	00004
00008	OUT	10006

### 4-3-8 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.

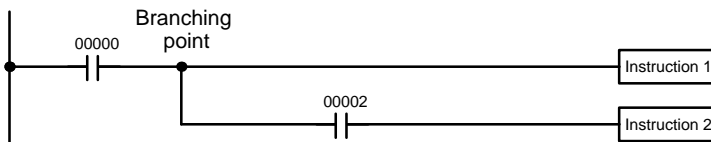


Diagram A: Correct Operation

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

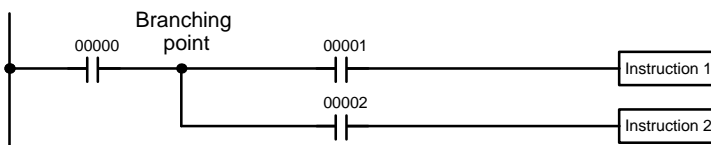


Diagram B: Incorrect Operation

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

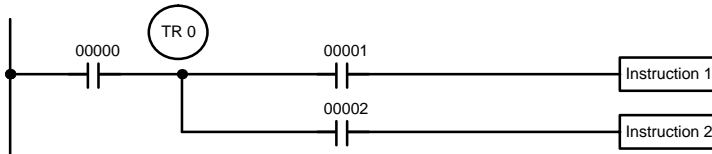
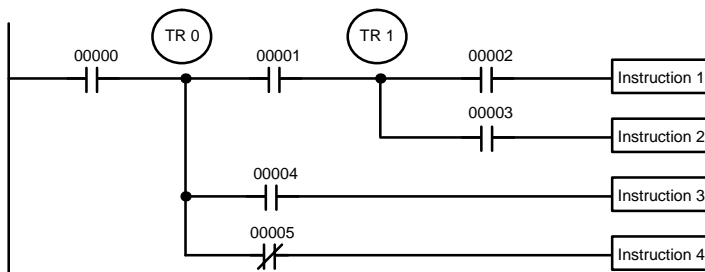


Diagram B: Corrected Using a TR bit

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.



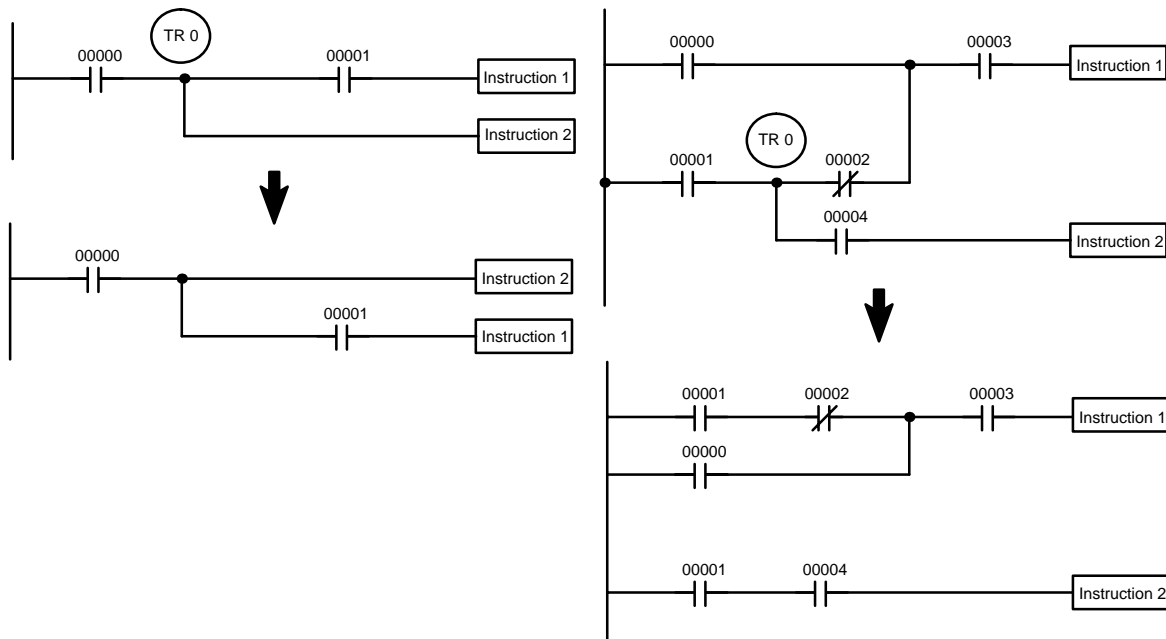
Address	Instruction	Operands
00000	LD	00000
00001	OUT	TR 0
00002	AND	00001
00003	OUT	TR 1
00004	AND	00002
00005	Instruction 1	
00006	LD	TR 1
00007	AND	00003
00008	Instruction 2	
00009	LD	TR 0
00010	AND	00004
00011	Instruction 3	
00012	LD	TR 0
00013	AND NOT	00005
00014	Instruction 4	

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 require 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.



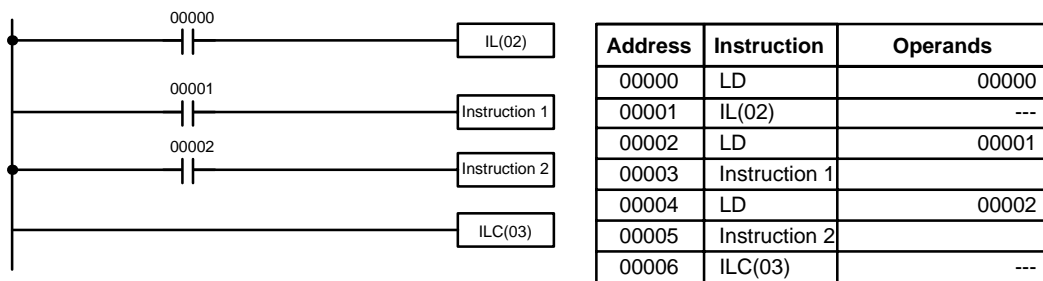
**Note** TR bits are must be input by the user only when programming using mnemonic code. They are not necessary when inputting ladder diagrams directly because they are processed for you automatically. The above limitations on the number of branching points requiring TR bits, and considerations on methods 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.

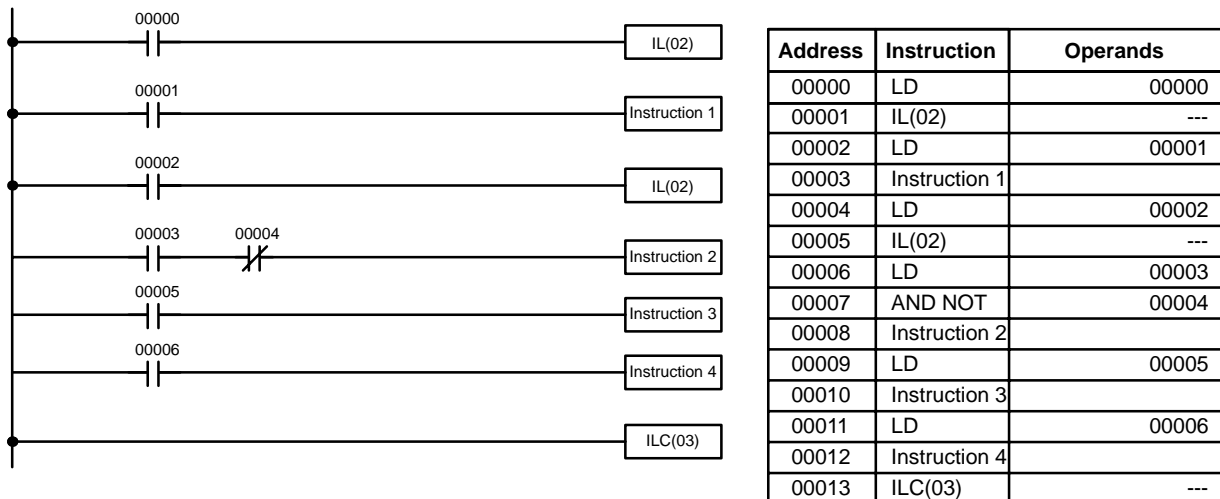
When an INTERLOCK instruction is placed before a section of a ladder program, the execution condition for the INTERLOCK instruction will control the execution of all instruction up to the next INTERLOCK CLEAR instruction. If the execution condition for the INTERLOCK instruction is OFF, all right-hand 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.



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

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



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

### 4-3-9 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.

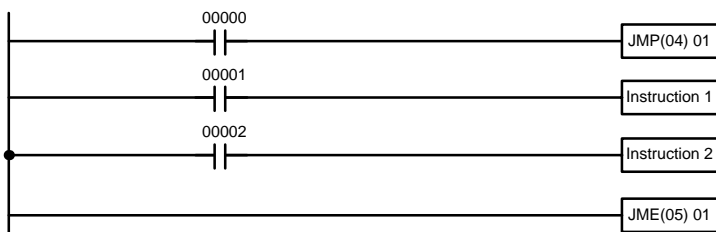


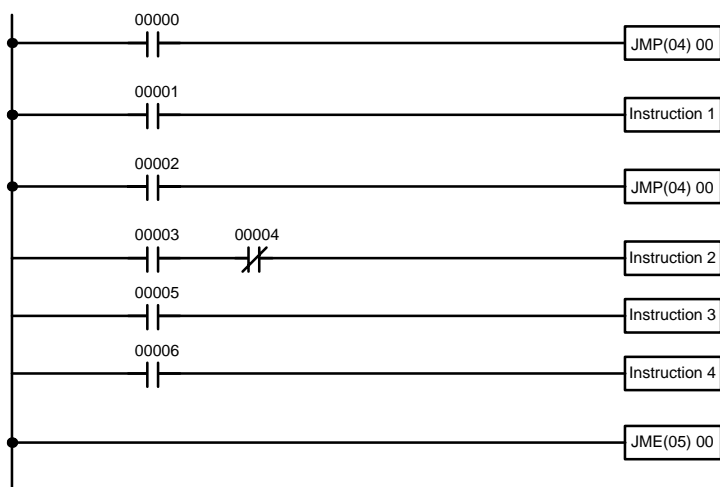
Diagram B: Corrected with a Jump

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 IR 00000 was OFF than any of the other versions.

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

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



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

## 4-4 Controlling Bit Status

There are seven basic instructions that can be used generally to control individual bit status. These are the OUTPUT, OUTPUT NOT, SET, RESET, 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 *5-8 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.

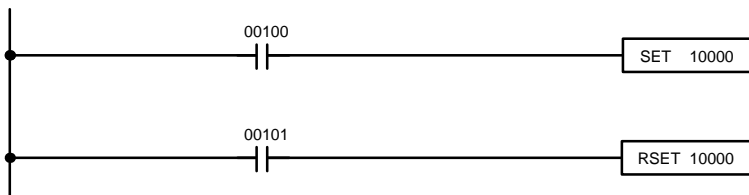
### 4-4-1 SET and RESET

The SET and RESET instructions are very similar to the OUTPUT and OUTPUT NOT instructions except that they only change the status of their operand bits for ON execution conditions. Neither instructions will affect the status of its operand bit when the execution condition is OFF.



SET will turn ON the operand bit when the execution condition goes ON, but unlike the OUTPUT instruction, SET will not turn OFF the operand bit when the execution condition goes OFF. RESET will turn OFF the operand bit when the execution condition goes OFF, but unlike OUTPUT NOT, RESET will not turn ON the operand bit when the execution condition goes OFF.

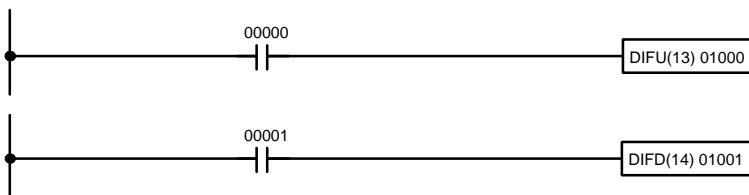
In the following example, IR 10000 will be turned ON when IR 00100 goes ON and will remain ON until IR 00101 goes ON, regardless of the status of IR 00100. When IR 00101 goes ON, RESET will turn IR 10000 OFF.



Address	Instruction	Operands
00000	LD	00100
00001	SET	10000
00002	LD	00101
00003	RSET	10000

### 4-4-2 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)	01000

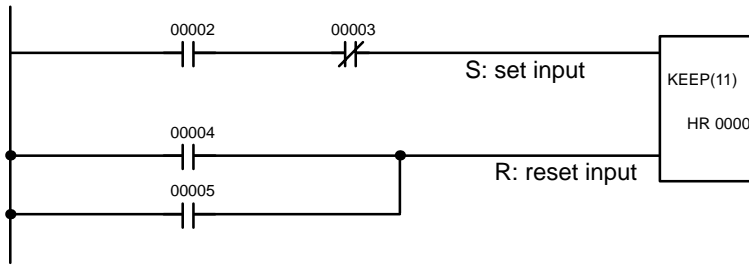
Address	Instruction	Operands
00000	LD	00001
00001	DIFD(14)	01001

Here, IR 01000 will be turned ON for one cycle after IR 00000 goes ON. The next time DIFU(13) 01000 is executed, IR 01000 will be turned OFF, regardless of the status of IR 00000. With the DIFFERENTIATE DOWN instruction, IR 01001 will be turned ON for one cycle after IR 00001 goes OFF (IR 01001 will be kept OFF until then), and will be turned OFF the next time DIFD(14) 01001 is executed.

### 4-4-3 KEEP

The KEEP instruction is used to maintain the status of the operand bit based on two execution conditions. To do this, the KEEP instruction is connected to two instruction lines. When the execution condition at the end of the first instruction line is ON, the operand bit of the KEEP instruction is turned ON. When the execution condition at the end of the second instruction line is ON, the operand bit of the KEEP instruction is turned OFF. The operand bit for the KEEP instruction will maintain its ON or OFF status even if it is located in an interlocked section of the diagram.

In the following example, 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.



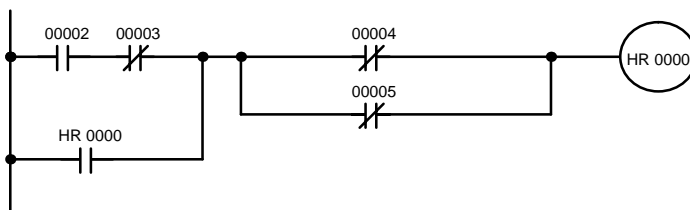
Address	Instruction	Operands
00000	LD	00002
00001	AND NOT	00003
00002	LD	00004
00003	OR	00005
00004	KEEP(11)	HR 0000

### 4-4-4 Self-maintaining Bits (Seal)

Although the KEEP instruction can be used to create self-maintaining bits, it is sometimes necessary to create self-maintaining bits in another way so that they can be turned OFF when in an interlocked section of a program.

To create a self-maintaining bit, the operand bit of an OUTPUT instruction is used as a condition for the same OUTPUT instruction in an OR setup so that the operand bit of the OUTPUT instruction will remain ON or OFF until changes occur in other bits. At least one other condition is used just before the OUTPUT instruction to function as a reset. Without this reset, there would be no way to control the operand bit of the OUTPUT instruction.

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

## 4-5 Work Bits (Internal Relays)

In programming, combining conditions to directly produce execution conditions is often extremely difficult. These difficulties are easily overcome, however, by using certain bits to trigger other instructions indirectly. Such programming is achieved by using work bits. Sometimes entire words are required for these purposes. These words are referred to as work words.

Work bits are not transferred to or from the PC. They are bits selected by the programmer to facilitate programming as described above. I/O bits and other dedicated bits cannot be used as work bits. All bits in the IR area that are not allocated as I/O bits, and certain unused bits in the AR area, are available for use as work bits. Be careful to keep an accurate record of how and where you use work bits. This helps in program planning and writing, and also aids in debugging operations.

**Work Bit Applications**

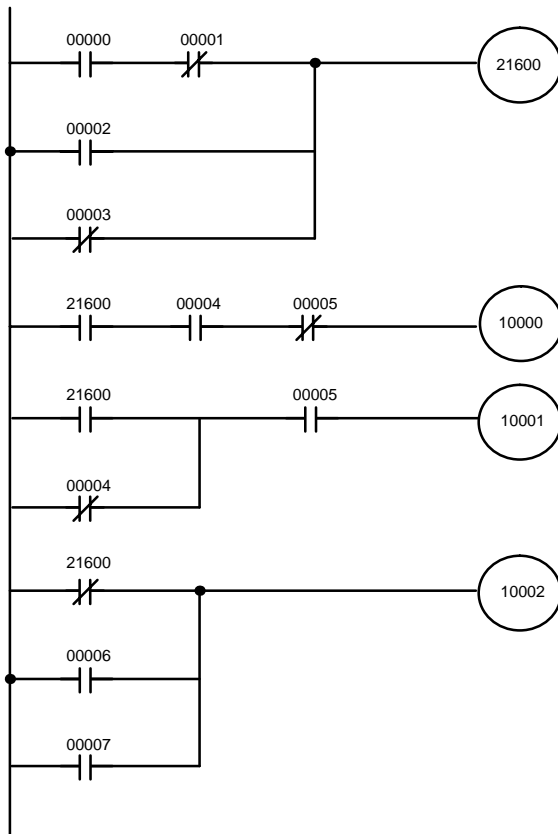
Examples given later in this subsection show two of the most common ways to employ work bits. These should act as a guide to the almost limitless number of ways in which the work bits can be used. Whenever difficulties arise in programming a control action, consideration should be given to work bits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

Work bits are often used with the OUTPUT, OUTPUT NOT, DIFFERENTIATE UP, DIFFERENTIATE DOWN, and KEEP instructions. The work bit is used first as the operand for one of these instructions so that later it can be used as a condition that will determine how other instructions will be executed. Work bits can also be used with other instructions, e.g., with the SHIFT REGISTER instruction (SFT(10)). An example of the use of work words and bits with the SHIFT REGISTER instruction is provided in 5-16-1 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 21600. IR 21600 is then combined with various other conditions to determine output conditions for IR 10000, IR 10001, and IR 10002, i.e., to turn the outputs allocated to these bits ON or OFF.

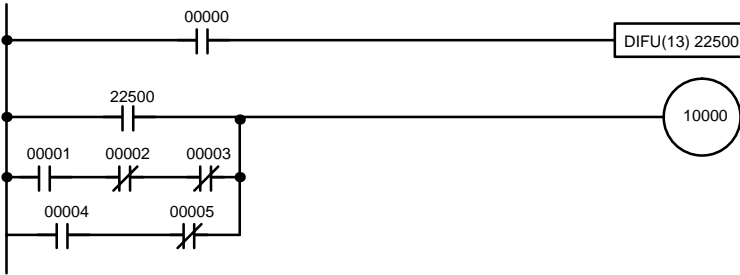


Address	Instruction	Operands
00000	LD	00000
00001	AND NOT	00001
00002	OR	00002
00003	OR NOT	00003
00004	OUT	21600
00005	LD	21600
00006	AND	00004
00007	AND NOT	00005
00008	OUT	10000
00009	LD	21600
00010	OR NOT	00004
00011	AND	00005
00012	OUT	10001
00013	LD NOT	21600
00014	OR	00006
00015	OR	00007
00016	OUT	10002

**Differentiated Conditions**

Work bits can also be used if differential treatment is necessary for some, but not all, of the conditions required for execution of an instruction. In this example, IR 10000 must be left ON continuously as long as IR 001001 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).

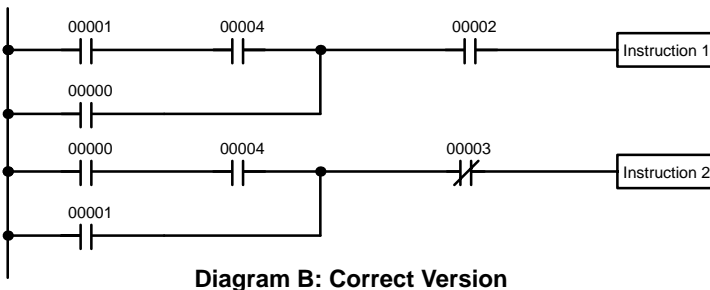
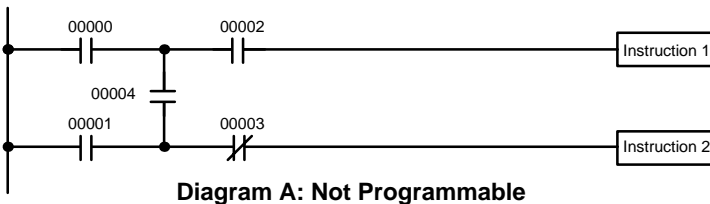
This action is easily programmed by using IR 22500 as a work bit as the operand of the DIFFERENTIATE UP instruction (DIFU(13)). When IR 00000 turns ON, IR 22500 will be turned ON for one cycle and then be turned OFF the next cycle by DIFU(13). Assuming the other conditions controlling IR 10000 are not keeping it ON, the work bit IR 22500 will turn IR 10000 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	10000

## 4-6 Programming Precautions

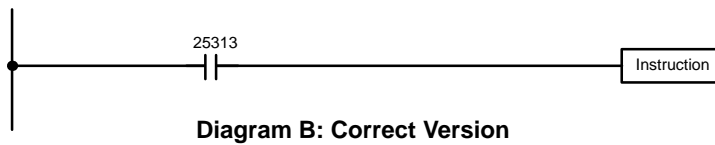
The number of conditions that can be used in series or parallel is unlimited as long as the memory capacity of the PC is not exceeded. Therefore, use as many conditions as required to draw a clear diagram. Although very complicated diagrams can be drawn with instruction lines, there must not be any conditions 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.



Address	Instruction	Operands
00000	LD	00001
00001	AND	00004
00002	OR	00000
00003	AND	00002
00004	Instruction 1	
00005	LD	00000
00006	AND	00004
00007	OR	00001
00008	AND NOT	00003
00009	Instruction 2	

The number of times any particular bit can be assigned to conditions is not limited, so use them as many times as required to simplify your program. Often, complicated programs are the result of attempts to reduce the number of times a bit is used.

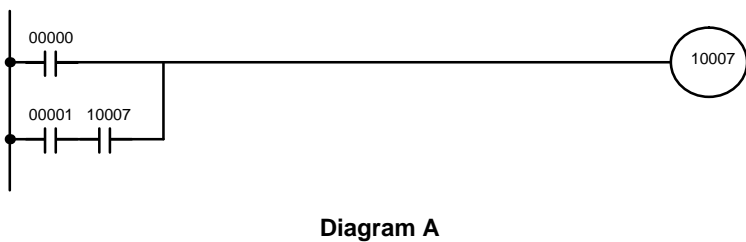
Except for instructions for which conditions are not allowed (e.g., INTERLOCK CLEAR and JUMP END, see below), every instruction line must also have at least one condition on it to determine the execution condition for the instruction at the right. Again, diagram A, below, must be drawn as diagram B. If an instruction must be continuously executed (e.g., if an output must always be kept ON while the program is being executed), the Always ON Flag (SR 25313) in the SR area can be used.



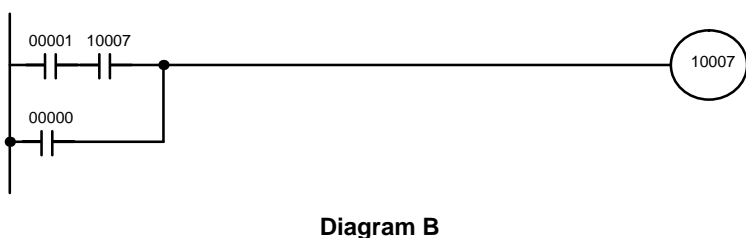
Address	Instruction	Operands
00000	LD	25313
00001	Instruction	

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

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



Address	Instruction	Operands
00000	LD	00000
00001	LD	00001
00002	AND	10007
00003	OR LD	---
00004	OUT	10007



Address	Instruction	Operands
00000	LD	00001
00001	AND	10007
00002	OR	00000
00003	OUT	10007

## **4-7 Program Execution**

When program execution is started, the CPU Unit 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 Unit as part of the cycle time. Refer to *Section 7 PC Operations and Processing Time* for details.

# SECTION 5

## Instruction Set

The CQM1 and CPM1 have large programming instruction sets that allow for easy programming of complicated control processes. This section explains instructions individually and provides the ladder diagram symbol, data areas, and flags used with each.

The many instructions provided by these PCs are organized in the following subsections by instruction group. These groups include Ladder Diagram Instructions, instructions with fixed function codes, and set 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	184
5-2	Instruction Format	184
5-3	Data Areas, Definer Values, and Flags	184
5-4	Differentiated Instructions	186
5-5	Coding Right-hand Instructions	186
5-6	Instruction Tables	190
5-6-1	CQM1 Function Codes	190
5-6-2	CPM1/CPM1A Function Codes	191
5-6-3	SRM1 Function Codes	192
5-6-4	Alphabetic List by Mnemonic	193
5-7	Ladder Diagram Instructions	196
5-7-1	LOAD, LOAD NOT, AND, AND NOT, OR, and OR NOT	196
5-7-2	AND LOAD and OR LOAD	197
5-8	Bit Control Instructions	197
5-8-1	OUTPUT and OUTPUT NOT – OUT and OUT NOT	197
5-8-2	SET and RESET – SET and RSET	198
5-8-3	KEEP – KEEP(11)	199
5-8-4	DIFFERENTIATE UP and DOWN – DIFU(13) and DIFD(14)	200
5-9	NO OPERATION – NOP(00)	201
5-10	END – END(01)	201
5-11	INTERLOCK and INTERLOCK CLEAR – IL(02) and ILC(03)	201
5-12	JUMP and JUMP END – JMP(04) and JME(05)	203
5-13	User Error Instructions:	
	FAILURE ALARM AND RESET – FAL(06) and SEVERE FAILURE ALARM – FALS(07)	205
5-14	Step Instructions:	
	STEP DEFINE and STEP START–STEP(08)/SNXT(09)	206
5-15	Timer and Counter Instructions	208
5-15-1	TIMER – TIM	209
5-15-2	COUNTER – CNT	210
5-15-3	REVERSIBLE COUNTER – CNTR(12)	211
5-15-4	HIGH-SPEED TIMER – TIMH(15)	212
5-15-5	INTERVAL TIMER – STIM(69)	213
5-15-6	REGISTER COMPARISON TABLE – CTBL(63)	215
5-15-7	MODE CONTROL – INI(61)	220
5-15-8	HIGH-SPEED COUNTER PV READ – PRV(62)	221
5-16	Shift Instructions	224
5-16-1	SHIFT REGISTER – SFT(10)	224
5-16-2	WORD SHIFT – WSFT(16)	225
5-16-3	ARITHMETIC SHIFT LEFT – ASL(25)	225
5-16-4	ARITHMETIC SHIFT RIGHT – ASR(26)	226
5-16-5	ROTATE LEFT – ROL(27)	226
5-16-6	ROTATE RIGHT – ROR(28)	227
5-16-7	ONE DIGIT SHIFT LEFT – SLD(74)	228
5-16-8	ONE DIGIT SHIFT RIGHT – SRD(75)	228
5-16-9	REVERSIBLE SHIFT REGISTER – SFTR(84)	229
5-16-10	ASYNCHRONOUS SHIFT REGISTER – ASFT(17)	230

5-17	Data Movement Instructions	232
5-17-1	MOVE – MOV(21)	232
5-17-2	MOVE NOT – MVN(22)	232
5-17-3	BLOCK TRANSFER – XFER(70)	233
5-17-4	BLOCK SET – BSET(71)	234
5-17-5	DATA EXCHANGE – XCHG(73)	235
5-17-6	SINGLE WORD DISTRIBUTE – DIST(80)	235
5-17-7	DATA COLLECT – COLL(81)	237
5-17-8	MOVE BIT – MOVB(82)	239
5-17-9	MOVE DIGIT – MOVD(83)	240
5-17-10	TRANSFER BITS – XFRB(—)	241
5-18	Comparison Instructions	242
5-18-1	COMPARE – CMP(20)	242
5-18-2	TABLE COMPARE – TCMP(85)	243
5-18-3	BLOCK COMPARE – BCMP(68)	244
5-18-4	DOUBLE COMPARE – CMPL(60)	246
5-18-5	MULTI-WORD COMPARE – MCMP(19)	247
5-18-6	SIGNED BINARY COMPARE – CPS(—)	248
5-18-7	DOUBLE SIGNED BINARY COMPARE – CPSL(—)	249
5-18-8	AREA RANGE COMPARE – ZCP(—)	250
5-18-9	DOUBLE AREA RANGE COMPARE – ZCPL(—)	252
5-19	Conversion Instructions	252
5-19-1	BCD-TO-BINARY – BIN(23)	252
5-19-2	BINARY-TO-BCD – BCD(24)	253
5-19-3	DOUBLE BCD-TO-DOUBLE BINARY – BINL(58)	254
5-19-4	DOUBLE BINARY-TO-DOUBLE BCD – BCDL(59)	254
5-19-5	4-TO-16 DECODER – MLPX(76)	255
5-19-6	16-TO-4 ENCODER – DMPX(77)	257
5-19-7	7-SEGMENT DECODER – SDEC(78)	259
5-19-8	ASCII CONVERT – ASC(86)	262
5-19-9	ASCII-TO-HEXADECIMAL – HEX(—)	263
5-19-10	SCALING – SCL(66)	266
5-19-11	SIGNED BINARY TO BCD SCALING – SCL2(—)	268
5-19-12	BCD TO SIGNED BINARY SCALING – SCL3(—)	269
5-19-13	HOURS-TO-SECONDS – SEC(—)	271
5-19-14	SECONDS-TO-HOURS – HMS(—)	272
5-19-15	COLUMN-TO-LINE – LINE(—)	273
5-19-16	LINE-TO-COLUMN – COLM(—)	274
5-19-17	2'S COMPLEMENT – NEG(—)	275
5-19-18	DOUBLE 2'S COMPLEMENT – NEGL(—)	276
5-20	BCD Calculation Instructions	278
5-20-1	SET CARRY – STC(40)	278
5-20-2	CLEAR CARRY – CLC(41)	278
5-20-3	BCD ADD – ADD(30)	278
5-20-4	BCD SUBTRACT – SUB(31)	279
5-20-5	BCD MULTIPLY – MUL(32)	281
5-20-6	BCD DIVIDE – DIV(33)	282
5-20-7	DOUBLE BCD ADD – ADDL(54)	283
5-20-8	DOUBLE BCD SUBTRACT – SUBL(55)	285
5-20-9	DOUBLE BCD MULTIPLY – MULL(56)	286
5-20-10	DOUBLE BCD DIVIDE – DIVL(57)	287
5-20-11	SQUARE ROOT – ROOT(72)	288



5-21	Binary Calculation Instructions	289
5-21-1	BINARY ADD – ADB(50)	289
5-21-2	BINARY SUBTRACT – SBB(51)	290
5-21-3	BINARY MULTIPLY – MLB(52)	291
5-21-4	BINARY DIVIDE – DVB(53)	292
5-21-5	DOUBLE BINARY ADD – ADBL(—)	293
5-21-6	DOUBLE BINARY SUBTRACT – SBBL(—)	294
5-21-7	SIGNED BINARY MULTIPLY – MBS(—)	296
5-21-8	DOUBLE SIGNED BINARY MULTIPLY – MBSL(—)	297
5-21-9	SIGNED BINARY DIVIDE – DBS(—)	298
5-21-10	DOUBLE SIGNED BINARY DIVIDE – DBSL(—)	299
5-22	Special Math Instructions	300
5-22-1	FIND MAXIMUM – MAX(—)	300
5-22-2	FIND MINIMUM – MIN(—)	301
5-22-3	AVERAGE VALUE – AVG(—)	302
5-22-4	SUM – SUM(—)	303
5-22-5	ARITHMETIC PROCESS – APR(—)	305
5-23	Logic Instructions	308
5-23-1	COMPLEMENT – COM(29)	308
5-23-2	LOGICAL AND – ANDW(34)	309
5-23-3	LOGICAL OR – ORW(35)	309
5-23-4	EXCLUSIVE OR – XORW(36)	310
5-23-5	EXCLUSIVE NOR – XNRW(37)	311
5-24	Increment/Decrement Instructions	311
5-24-1	BCD INCREMENT – INC(38)	311
5-24-2	BCD DECREMENT – DEC(39)	312
5-25	Subroutine Instructions	313
5-25-1	SUBROUTINE ENTER – SBS(91)	313
5-25-2	SUBROUTINE DEFINE and RETURN – SBN(92)/RET(93)	315
5-26	Special Instructions	315
5-26-1	TRACE MEMORY SAMPLING – TRSM(45)	315
5-26-2	MESSAGE DISPLAY – MSG(46)	317
5-26-3	I/O REFRESH – IORF(97)	318
5-26-4	MACRO – MCRO(99)	319
5-26-5	BIT COUNTER – BCNT(67)	320
5-26-6	FRAME CHECKSUM – FCS(—)	321
5-26-7	FAILURE POINT DETECTION – FPD(—)	322
5-26-8	INTERRUPT CONTROL – INT(89)	326
5-26-9	SET PULSES – PULS(65)	328
5-26-10	SPEED OUTPUT – SPED(64)	330
5-26-11	PULSE OUTPUT – PLS2(—)	332
5-26-12	ACCELERATION CONTROL – ACC(—)	334
5-26-13	PULSE WITH VARIABLE DUTY RATIO – PWM(—)	336
5-26-14	DATA SEARCH – SRCH(—)	337
5-26-15	PID CONTROL – PID(—)	338
5-27	Communications Instructions	340
5-27-1	RECEIVE – RXD(47)	340
5-27-2	TRANSMIT – TXD(48)	342
5-27-3	CHANGE RS-232C SETUP – STUP(—)	344
5-28	Advanced I/O Instructions	345
5-28-1	7-SEGMENT DISPLAY OUTPUT – 7SEG(88)	345
5-28-2	DIGITAL SWITCH INPUT – DSW(87)	346
5-28-3	HEXADECIMAL KEY INPUT – HKY(—)	347
5-28-4	TEN KEY INPUT – TKY(18)	347

## 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 A 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 Unit. A table of instructions listed in order of function codes is also provided in *Appendix A Programming Instructions*. Lists of instructions are also provided in *5-6 Instruction Tables*.

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.

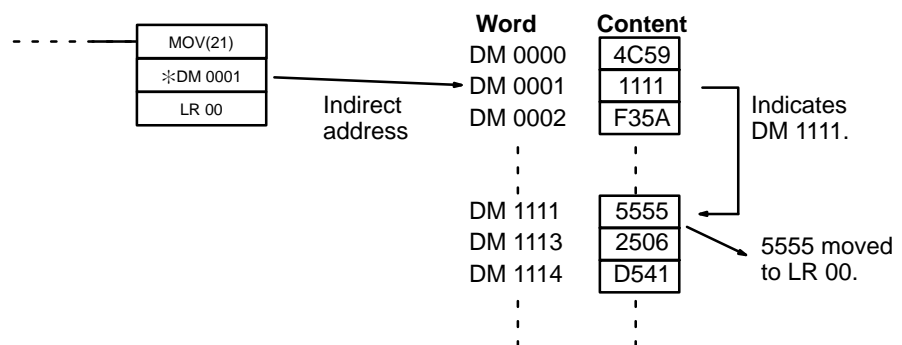
Abbreviation	Name	Bit
ER	Instruction Execution Error Flag	25503
CY	Carry Flag	25504
GR	Greater Than Flag	25505
EQ	Equals Flag	25506
LE	Less Than Flag	25507

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 B Error and Arithmetic Flag Operation*.

### Indirect Addressing

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

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



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

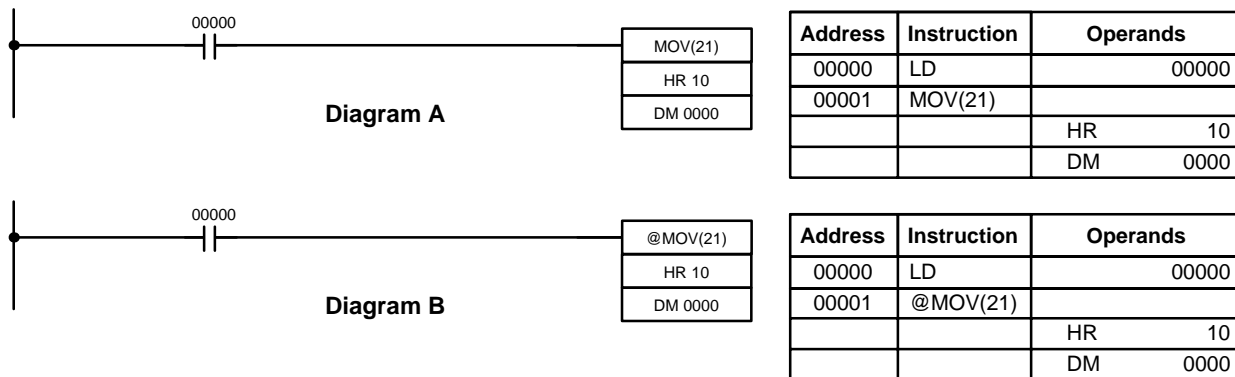
### Designating Constants

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

## 5-4 Differentiated Instructions

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

A non-differentiated instruction is executed each time it is 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.



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 once during the first cycle in which 00000 has changed from OFF to ON. Because the content of HR 10 could very well change during the 2 seconds while 00000 is ON, the final content of DM 0000 after the 2 seconds could be different depending on whether MOV(21) or @MOV(21) was used.

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

Refer to 5-11 INTERLOCK and INTERLOCK CLEAR – IL(02) and IL(03) for the effects of interlocks on differentiated instructions.

The CQM1 also provides differentiation instructions: DIFU(13) and DIFD(14). DIFU(13) operates the same as a differentiated instruction, but is used to turn ON a bit for one cycle. DIFD(14) also turns ON a bit for one cycle, but does it when the execution condition has changed from ON to OFF. Refer to 5-8-4 DIFFERENTIATE UP and 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 Ladder-diagram Programming. Converting the information in the ladder diagram symbol for all other instructions follows the same pattern, as described below, and is not specified for each instruction individually.

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

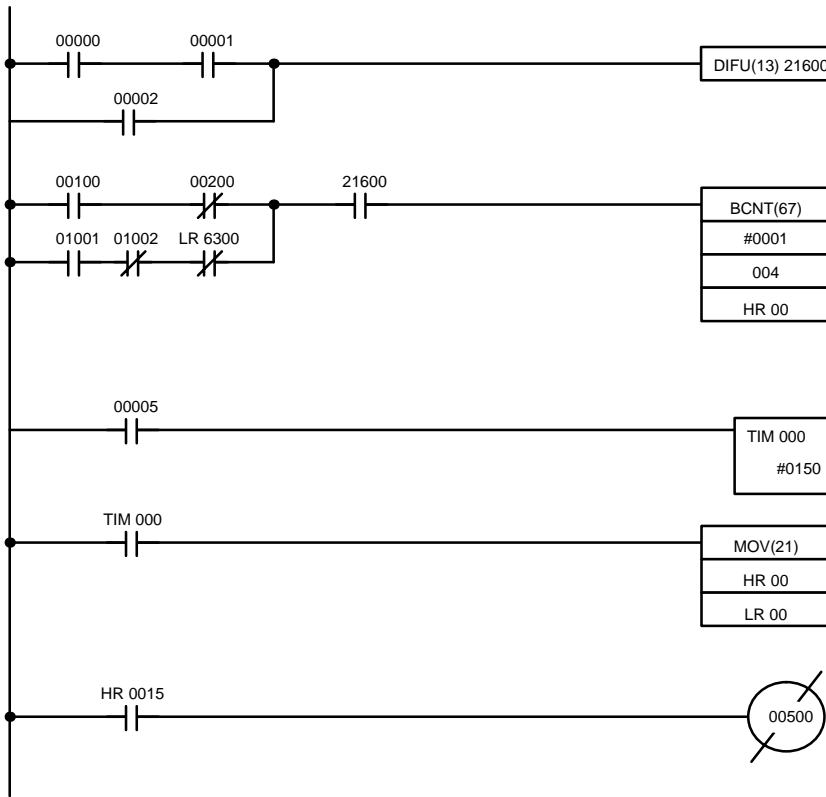
The address and instruction columns of the mnemonic code table are filled in for the instruction word only. For all other lines, the left two columns are left blank. If the instruction requires no definer or bit operand, the data column is left blank for first line. It is a good idea to cross through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

If an IR or SR address is used in the data column, the left side of the column is left blank. If any other data area is used, the data area abbreviation is placed on the left side and the address is placed 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.

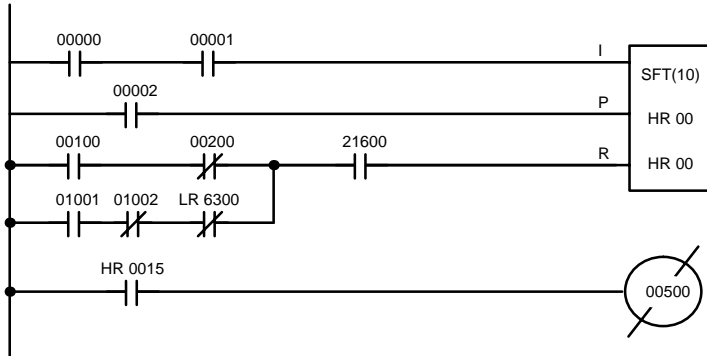
**Note** The mnemonics of expansion instructions are followed by “(—)” as the function code to indicate that they must be assigned function codes by the user in the instructions table before they can be used in programming. Refer to page 116 for details.

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)	21600
00004	LD	00100
00005	AND NOT	00200
00006	LD	01001
00007	AND NOT	01002
00008	AND NOT	LR 6300
00009	OR LD	—
00010	AND	21600
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	21600
00010	SFT(10)	HR 00
		HR 00
00011	LD	HR 0015
00012	OUT NOT	00500

## 5-6 Instruction Tables

This section provides tables of the instructions available in the CQM1. The first few tables can be used to find instructions by function code. The last table can be used to find instructions by mnemonic. In both tables, the @ symbol indicates instructions with differentiated forms.

### 5-6-1 CQM1 Function Codes

The following table lists the CQM1 instructions that have fixed function codes. Each instruction is listed by mnemonic and by instruction name. Use the numbers in the leftmost column as the left digit and the number in the column heading as the right digit of the function code.

Left digit	Right digit									
	0	1	2	3	4	5	6	7	8	9
0	NOP NO OPERATION	END END	IL INTERLOCK	ILC INTERLOCK CLEAR	JMP JUMP	JME JUMP END	(@) FAL FAILURE ALARM AND RESET	FALS SEVERE FAILURE ALARM	STEP STEP DEFINE	SNXT STEP START
1	SFT SHIFT REGISTER	KEEP KEEP	CNTR REVERSIBLE COUNTER	DIFU DIFFERENTIATE UP	DIFD DIFFERENTIATE DOWN	TIMH HIGH-SPEED TIMER	(@) WSFT WORD SHIFT	(@) ASFT ASYNCHRONOUS SHIFT REGISTER	(@) TKY TEN KEY INPUT	(@) MCMP MULTIWORD COMPARE
2	CMP COMPARE	(@) MOV MOVE	(@) MVN MOVE NOT	(@) BIN BCD TO BINARY	(@) BCD BINARY TO BCD	(@) ASL SHIFT LEFT	(@) ASR SHIFT RIGHT	(@) ROL ROTATE LEFT	(@) ROR ROTATE RIGHT	(@) COM COMPLEMENT
3	(@) ADD BCD ADD	(@) SUB BCD SUBTRACT	(@) MUL BCD MULTIPLY	(@) DIV BCD DIVIDE	(@) ANDW LOGICAL AND	(@) ORW LOGICAL OR	(@) XORW EXCLUSIVE OR	(@) XNRW EXCLUSIVE NOR	(@) INC INCREMENT	(@) DEC DECREMENT
4	(@) STC SET CARRY	(@) CLC CLEAR CARRY	---	---	---	TRSM TRACE MEMORY SAMPLE (SEE NOTE)	(@) MSG MESSAGE DISPLAY	(@) RXD RECEIVE	(@) TXD TRANSMIT	---
5	(@) ADB BINARY ADD	(@) SBB BINARY SUBTRACT	(@) MLB BINARY MULTIPLY	(@) DVB BINARY DIVIDE	(@) ADDL DOUBLE BCD ADD	(@) SUBL DOUBLE BCD SUBTRACT	(@) MULL DOUBLE BCD MULTIPLY	(@) DIVL DOUBLE BCD DIVIDE	(@) BINL DOUBLE BCD-TO-DOUBLE BINARY	(@) BCBL DOUBLE BINARY-TO-DOUBLE BCD
6	CMPL DOUBLE COMPARE	(@) INI MODE CONTROL	(@) PRV HIGH-SPEED COUNTER PV READ	(@) CTBL COMPARISON TABLE LOAD	(@) SPED SPEED OUTPUT	(@) PULS SET PULSES	(@) SCL SCALING	(@) BCNT BIT COUNTER	(@) BCMP BLOCK COMPARE	(@) STIM INTERVAL TIMER
7	(@) XFER BLOCK TRANSFER	(@) BSET BLOCK SET	(@) ROOT SQUARE ROOT	(@) XCHG DATA EXCHANGE	(@) SLD ONE DIGIT SHIFT LEFT	(@) SRD ONE DIGIT SHIFT RIGHT	(@) MLPX 4-TO-16 DECODER	(@) DMPX 16-TO-4 ENCODER	(@) SDEC 7-SEGMENT DECODER	---
8	(@) DIST SINGLE WORD DISTRIBUTE	(@) COLL DATA COLLECT	(@) MOVb MOVE BIT	(@) MOVD MOVE DIGIT	(@) SFTR REVERSIBLE SHIFT REGISTER	(@) TCMP TABLE COMPARE	(@) ASC ASCII CONVERT	(@) DSW DIGITAL SWITCH	(@) 7SEG 7-SEGMENT DISPLAY OUTPUT	(@) INT INTERRUPT CONTROL
9	---	(@) SBS SUBROUTINE ENTRY	SBN SUBROUTINE DEFINE	RET SUBROUTINE RETURN	---	---	---	(@) IORF I/O REFRESH	---	(@) MCRO MACRO

**Note** TRSM(45) cannot be used with the CQM1-CPU11/21-E CPU Units.



### 5-6-2 CPM1/CPM1A Function Codes

The following table lists the CPM1/CPM1A instructions that have fixed function codes. Each instruction is listed by mnemonic and by instruction name. Use the numbers in the leftmost column as the left digit and the number in the column heading as the right digit of the function code.

Left digit	Right digit									
	0	1	2	3	4	5	6	7	8	9
0	NOP NO OPERATION	END END	IL INTERLOCK	ILC INTERLOCK CLEAR	JMP JUMP	JME JUMP END	(@) FAL FAILURE ALARM AND RESET	FALS SEVERE FAILURE ALARM	STEP STEP DEFINE	SNXT STEP START
1	SFT SHIFT REGISTER	KEEP KEEP	CNTR REVERSIBLE COUNTER	DIFU DIFFERENTIATE UP	DIFD DIFFERENTIATE DOWN	TIMH HIGH-SPEED TIMER	(@) WSFT WORD SHIFT	(@) ASFT ASYNCHRONOUS SHIFT REGISTER	---	---
2	CMP COMPARE	(@) MOV MOVE	(@) MVN MOVE NOT	(@) BIN BCD TO BINARY	(@) BCD BINARY TO BCD	(@) ASL SHIFT LEFT	(@) ASR SHIFT RIGHT	(@) ROL ROTATE LEFT	(@) ROR ROTATE RIGHT	(@) COM COMPLEMENT
3	(@) ADD BCD ADD	(@) SUB BCD SUBTRACT	(@) MUL BCD MULTIPLY	(@) DIV BCD DIVIDE	(@) ANDW LOGICAL AND	(@) ORW LOGICAL OR	(@) XORW EXCLUSIVE OR	(@) XNRW EXCLUSIVE NOR	(@) INC INCREMENT	(@) DEC DECREMENT
4	(@) STC SET CARRY	(@) CLC CLEAR CARRY	---	---	---	---	(@) MSG MESSAGE DISPLAY	---	---	---
5	(@) ADB BINARY ADD	(@) SBB BINARY SUBTRACT	(@) MLB BINARY MULTIPLY	(@) DVB BINARY DIVIDE	(@) ADDL DOUBLE BCD ADD	(@) SUBL DOUBLE BCD SUBTRACT	(@) MULL DOUBLE BCD MULTIPLY	(@) DIVL DOUBLE BCD DIVIDE	---	---
6	CMPL DOUBLE COMPARE	(@) INI MODE CONTROL	(@) PRV HIGH-SPEED COUNTER PV READ	(@) CTBL COMPARISON TABLE LOAD	---	---	---	(@) BCNT BIT COUNTER	(@) BCMP BLOCK COMPARE	(@) STIM INTERVAL TIMER
7	(@) XFER BLOCK TRANSFER	(@) BSET BLOCK SET	---	(@) XCHG DATA EXCHANGE	(@) SLD ONE DIGIT SHIFT LEFT	(@) SRD ONE DIGIT SHIFT RIGHT	(@) MLPX 4-TO-16 DECODER	(@) DMPX 16-TO-4 ENCODER	(@) SDEC 7-SEGMENT DECODER	---
8	(@) DIST SINGLE WORD DISTRIBUTE	(@) COLL DATA COLLECT	(@) MOV B MOVE BIT	(@) MOV D MOVE DIGIT	(@) SFTR REVERSIBLE SHIFT REGISTER	(@) TCMP TABLE COMPARE	(@) ASC ASCII CONVERT	---	---	(@) INT INTERRUPT CONTROL
9	---	(@) SBS SUBROUTINE ENTRY	SBN SUBROUTINE DEFINE	RET SUBROUTINE RETURN	---	---	---	(@) IORF I/O REFRESH	---	(@) MCRO MACRO

### 5-6-3 SRM1 Function Codes

The following table lists the SRM1 instructions that have fixed function codes. Each instruction is listed by mnemonic and by instruction name. Use the numbers in the leftmost column as the left digit and the number in the column heading as the right digit of the function code.

Left digit	Right digit									
	0	1	2	3	4	5	6	7	8	9
0	NOP NO OPERATION	END END	IL INTERLOCK	ILC INTERLOCK CLEAR	JMP JUMP	JME JUMP END	(@) FAL FAILURE ALARM AND RESET	FALS SEVERE FAILURE ALARM	STEP STEP DEFINE	SNXT STEP START
1	SFT SHIFT REGISTER	KEEP KEEP	CNTR REVERSIBLE COUNTER	DIFU DIFFERENTIATE UP	DIFD DIFFERENTIATE DOWN	TIMH HIGH-SPEED TIMER	(@) WSFT WORD SHIFT	(@) ASFT ASYNCHRONOUS SHIFT REGISTER	---	---
2	CMP COMPARE	(@) MOV MOVE	(@) MVN MOVE NOT	(@) BIN BCD TO BINARY	(@) BCD BINARY TO BCD	(@) ASL SHIFT LEFT	(@) ASR SHIFT RIGHT	(@) ROL ROTATE LEFT	(@) ROR ROTATE RIGHT	(@) COM COMPLEMENT
3	(@) ADD BCD ADD	(@) SUB BCD SUBTRACT	(@) MUL BCD MULTIPLY	(@) DIV BCD DIVIDE	(@) ANDW LOGICAL AND	(@) ORW LOGICAL OR	(@) XORW EXCLUSIVE OR	(@) XNRW EXCLUSIVE NOR	(@) INC INCREMENT	(@) DEC DECREMENT
4	(@) STC SET CARRY	(@) CLC CLEAR CARRY	---	---	---	---	(@) MSG MESSAGE DISPLAY	---	---	---
5	(@) ADB BINARY ADD	(@) SBB BINARY SUBTRACT	(@) MLB BINARY MULTIPLY	(@) DVB BINARY DIVIDE	(@) ADDL DOUBLE BCD ADD	(@) SUBL DOUBLE BCD SUBTRACT	(@) MULL DOUBLE BCD MULTIPLY	(@) DIVL DOUBLE BCD DIVIDE	---	---
6	CMPL DOUBLE COMPARE	---	---	---	---	---	---	(@) BCNT BIT COUNTER	(@) BCMP BLOCK COMPARE	(@) STIM INTERVAL TIMER
7	(@) XFER BLOCK TRANSFER	(@) BSET BLOCK SET	---	(@) XCHG DATA EXCHANGE	(@) SLD ONE DIGIT SHIFT LEFT	(@) SRD ONE DIGIT SHIFT RIGHT	(@) MLPX 4-TO-16 DECODER	(@) DMPX 16-TO-4 ENCODER	(@) SDEC 7-SEGMENT DECODER	---
8	(@) DIST SINGLE WORD DISTRIBUTE	(@) COLL DATA COLLECT	(@) MOVB MOVE BIT	(@) MOVD MOVE DIGIT	(@) SFTR REVERSIBLE SHIFT REGISTER	(@) TCMP TABLE COMPARE	(@) ASC ASCII CONVERT	---	---	---
9	---	(@) SBS SUBROUTINE ENTRY	SBN SUBROUTINE DEFINE	RET SUBROUTINE RETURN	---	---	---	(@) IORF I/O REFRESH	---	(@) MCRO MACRO

### 5-6-4 Alphabetic List by Mnemonic

Dashes (“—”) in the *Code* column indicate expansion instructions, which do not have fixed function codes. “None” indicates instructions for which function codes are not used.

Mnemonic	Code	Words	Name	CPU Units	Page
7SEG	88	4	7-SEGMENT DISPLAY OUTPUT	CQM1 only	345
ACC (@)	—	4	ACCELERATION CONTROL	CQM-CPU43-E/-EV1 only	334
ADB (@)	50	4	BINARY ADD	All	289
ADBL (@)	—	4	DOUBLE BINARY ADD	CQM1-CPU4□-E/-EV1 only	293
ADD (@)	30	4	BCD ADD	All	278
ADDL (@)	54	4	DOUBLE BCD ADD	All	284
AND	None	1	AND	All	196
AND LD	None	1	AND LOAD	All	197
AND NOT	None	1	AND NOT	All	196
ANDW (@)	34	4	LOGICAL AND	All	309
APR (@)	—	4	ARITHMETIC PROCESS	CQM1 only	305
ASC (@)	86	4	ASCII CONVERT	All	262
ASFT(@)	17	4	ASYNCHRONOUS SHIFT REGISTER	All	230
ASL (@)	25	2	ARITHMETIC SHIFT LEFT	All	225
ASR (@)	26	2	ARITHMETIC SHIFT RIGHT	All	226
AVG	—	4	AVERAGE VALUE	CQM1 only	302
BCD (@)	24	3	BINARY TO BCD	All	253
BCDL (@)	59	3	DOUBLE BINARY-TO-DOUBLE BCD	CQM1 only	254
BCMP (@)	68	4	BLOCK COMPARE	All	244
BCNT (@)	67	4	BIT COUNTER	All	320
BIN (@)	23	3	BCD-TO-BINARY	All	252
BINL (@)	58	3	DOUBLE BCD-TO-DOUBLE BINARY	CQM1 only	254
BSET (@)	71	4	BLOCK SET	All	234
CLC (@)	41	1	CLEAR CARRY	All	278
CMP	20	3	COMPARE	All	242
CMPL	60	4	DOUBLE COMPARE	All	246
CNT	None	2	COUNTER	All	210
CNTR	12	3	REVERSIBLE COUNTER	All	211
COLL (@)	81	4	DATA COLLECT	All	237
COLM(@)	—	4	LINE TO COLUMN	CQM1 only	274
COM (@)	29	2	COMPLEMENT	All	308
CPS	—	4	SIGNED BINARY COMPARE	CQM1-CPU4□-E/-EV1 only	248
CPSL	—	4	DOUBLE SIGNED BINARY COMPARE	CQM1-CPU4□-E/-EV1 only	249
CTBL(@)	63	4	COMPARISON TABLE LOAD	All	215
DBS (@)	—	4	SIGNED BINARY DIVIDE	CQM1-CPU4□-E/-EV1 only	298
DBSL (@)	—	4	DOUBLE SIGNED BINARY DIVIDE	CQM1-CPU4□-E/-EV1 only	299
DEC (@)	39	2	BCD DECREMENT	All	312
DIFD	14	2	DIFFERENTIATE DOWN	All	200
DIFU	13	2	DIFFERENTIATE UP	All	200
DIST (@)	80	4	SINGLE WORD DISTRIBUTE	All	235
DIV (@)	33	4	BCD DIVIDE	All	282
DIVL (@)	57	4	DOUBLE BCD DIVIDE	All	287
DMPX (@)	77	4	16-TO-4 ENCODER	All	257
DSW	87	4	DIGITAL SWITCH	CQM1 only	346

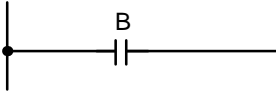
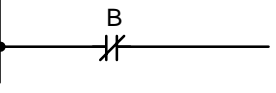
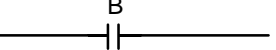
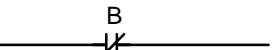
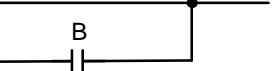
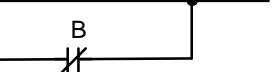
Mnemonic	Code	Words	Name	CPU Units	Page
DVB (@)	53	4	BINARY DIVIDE	All	292
END	01	1	END	All	201
FAL (@)	06	2	FAILURE ALARM AND RESET	All	205
FALS	07	2	SEVERE FAILURE ALARM	All	205
FCS (@)	—	4	FCS CALCULATE	CQM1/SRM1 only	321
FPD	—	4	FAILURE POINT DETECT	CQM1 only	322
HEX (@)	—	4	ASCII-TO-HEXADECIMAL	CQM1/SRM1 only	263
HKY	—	4	HEXADECIMAL KEY INPUT	CQM1 only	347
HMS	—	4	SECONDS TO HOURS	CQM1 only	272
IL	02	1	INTERLOCK	All	201
ILC	03	1	INTERLOCK CLEAR	All	201
INC (@)	38	2	INCREMENT	All	311
INI (@)	61	4	MODE CONTROL	All	220
INT (@)	89	4	INTERRUPT CONTROL	All	326
IORF (@)	97	3	I/O REFRESH	All	318
JME	05	2	JUMP END	All	203
JMP	04	2	JUMP	All	203
KEEP	11	2	KEEP	All	199
LD	None	1	LOAD	All	196
LD NOT	None	1	LOAD NOT	All	196
LINE	—	4	LINE	CQM1 only	273
MAX (@)	—	4	FIND MAXIMUM	CQM1 only	300
MBS (@)	—	4	SIGNED BINARY MULTIPLY	CQM1-CPU4□-E/-EV1 only	296
MBSL (@)	—	4	DOUBLE SIGNED BINARY MULTIPLY	CQM1-CPU4□-E/-EV1 only	297
MCMP (@)	19	4	MULTI-WORD COMPARE	CQM1 only	301
MCRO (@)	99	4	MACRO	All	319
MIN (@)	—	4	FIND MINIMUM	CQM1 only	301
MLB (@)	52	4	BINARY MULTIPLY	All	291
MLPX (@)	76	4	4-TO-16 DECODER	All	255
MOV (@)	21	3	MOVE	All	232
MOVB (@)	82	4	MOVE BIT	All	239
MOVD (@)	83	4	MOVE DIGIT	All	240
MSG (@)	46	2	MESSAGE	All	317
MUL (@)	32	4	BCD MULTIPLY	All	281
MULL (@)	56	4	DOUBLE BCD MULTIPLY	All	286
MVN (@)	22	3	MOVE NOT	All	232
NEG (@)	—	4	2'S COMPLEMENT	CQM1-CPU4□-E/-EV1 only	275
NEGL (@)	—	4	DOUBLE 2'S COMPLEMENT	CQM1-CPU4□-E/-EV1 only	276
NOP	00	1	NO OPERATION	All	201
OR	None	1	OR	All	196
OR LD	None	1	OR LOAD	All	197
OR NOT	None	1	OR NOT	All	196
ORW (@)	35	4	LOGICAL OR	All	309
OUT	None	2	OUTPUT	All	197
OUT NOT	None	2	OUTPUT NOT	All	197
PID	—	4	PID CONTROL	CQM1-CPU4□-E/-EV1 only	338
PLS2 (@)	—	4	PULSE OUTPUT	CQM1-CPU43-E/-EV1 only	332
PRV (@)	62	4	HIGH-SPEED COUNTER PV READ	All	221

Mnemonic	Code	Words	Name	CPU Units	Page
PULS (@)	65	4	SET PULSES	CQM1 and CPM1A-□□CDT/CDT1 only	328
PWM (@)	—	4	PULSE WITH VARIABLE DUTY RATIO	CQM1-CPU43-E/-EV1 only	336
RET	93	1	SUBROUTINE RETURN	All	315
ROL (@)	27	2	ROTATE LEFT	All	226
ROOT (@)	72	3	SQUARE ROOT	CQM1 only	288
ROR (@)	28	2	ROTATE RIGHT	All	227
RSET	None	2	RESET	All	198
RXD (@)	47	4	RECEIVE	CQM1/SRM1 only	340
SBB (@)	51	4	BINARY SUBTRACT	All	290
SBBL (@)	—	4	DOUBLE BINARY SUBTRACT	CQM1-CPU4□-E/-EV1 only	294
SBN	92	2	SUBROUTINE DEFINE	All	315
SBS (@)	91	2	SUBROUTINE ENTRY	All	313
SCL (@)	66	4	SCALING	CQM1 only	266
SCL2 (@)	—	4	SIGNED BINARY TO BCD SCALING	CQM1-CPU4□-E/-EV1 only	268
SCL3 (@)	—	4	BCD TO SIGNED BINARY SCALING	CQM1-CPU4□-E/-EV1 only	269
SDEC (@)	78	4	7-SEGMENT DECODER	All	259
SEC	—	4	HOURS TO SECONDS	CQM1 only	271
SET	None	2	SET	All	198
SFT	10	3	SHIFT REGISTER	All	224
SFTR (@)	84	4	REVERSIBLE SHIFT REGISTER	All	229
SLD (@)	74	3	ONE DIGIT SHIFT LEFT	All	228
SNXT	09	2	STEP START	All	206
SPED (@)	64	4	SPEED OUTPUT	CQM1 and CPM1A-□□CDT/CDT1 only	330
SRCH (@)	—	4	DATA SEARCH	CQM1 only	337
SRD (@)	75	3	ONE DIGIT SHIFT RIGHT	All	228
STC (@)	40	1	SET CARRY	All	278
STEP	08	2	STEP DEFINE	All	206
STIM (@)	69	4	INTERVAL TIMER	All	213
STUP	—	3	CHANGE RS-232C SETUP	SRM1 only	344
SUB (@)	31	4	BCD SUBTRACT	All	279
SUBL (@)	55	4	DOUBLE BCD SUBTRACT	All	285
SUM (@)	—	4	SUM	CQM1 only	303
TCMP (@)	85	4	TABLE COMPARE	All	243
TIM	None	2	TIMER	All	209
TIMH	15	3	HIGH-SPEED TIMER	All	212
TKY (@)	18	4	TEN KEY INPUT	CQM1 only	347
TRSM	45	1	TRACE MEMORY SAMPLE	CQM1-CPU4□-E/-EV1 only	315
TXD (@)	48	4	TRANSMIT	CQM1/SRM1 only	342
WSFT (@)	16	3	WORD SHIFT	All	225
XCHG (@)	73	3	DATA EXCHANGE	All	235
XFER (@)	70	4	BLOCK TRANSFER	All	233
XFRB (@)	—	4	TRANSFER BITS	CQM1-CPU4□-E/-EV1 only	241
XNRW (@)	37	4	EXCLUSIVE NOR	All	311
XORW (@)	36	4	EXCLUSIVE OR	All	310
ZCP	—	4	AREA RANGE COMPARE	CQM1-CPU4□-E/-EV1 only	250
ZCPL	—	4	DOUBLE AREA RANGE COMPARE	CQM1-CPU4□-E/-EV1 only	252

## 5-7 Ladder Diagram Instructions

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

### 5-7-1 LOAD, LOAD NOT, AND, AND NOT, OR, and OR NOT

	Ladder Symbols	Operand Data Areas		
LOAD – LD		<table border="1"> <tr><td>B: Bit</td></tr> <tr><td>IR, SR, AR, HR, TC, LR, TR</td></tr> </table>	B: Bit	IR, SR, AR, HR, TC, LR, TR
B: Bit				
IR, SR, AR, HR, TC, LR, TR				
LOAD NOT – LD NOT		<table border="1"> <tr><td>B: Bit</td></tr> <tr><td>IR, SR, AR, HR, TC, LR</td></tr> </table>	B: Bit	IR, SR, AR, HR, TC, LR
B: Bit				
IR, SR, AR, HR, TC, LR				
AND – AND		<table border="1"> <tr><td>B: Bit</td></tr> <tr><td>IR, SR, AR, HR, TC, LR</td></tr> </table>	B: Bit	IR, SR, AR, HR, TC, LR
B: Bit				
IR, SR, AR, HR, TC, LR				
AND NOT – AND NOT		<table border="1"> <tr><td>B: Bit</td></tr> <tr><td>IR, SR, AR, HR, TC, LR</td></tr> </table>	B: Bit	IR, SR, AR, HR, TC, LR
B: Bit				
IR, SR, AR, HR, TC, LR				
OR – OR		<table border="1"> <tr><td>B: Bit</td></tr> <tr><td>IR, SR, AR, HR, TC, LR</td></tr> </table>	B: Bit	IR, SR, AR, HR, TC, LR
B: Bit				
IR, SR, AR, HR, TC, LR				
OR NOT – OR NOT		<table border="1"> <tr><td>B: Bit</td></tr> <tr><td>IR, SR, AR, HR, TC, LR</td></tr> </table>	B: Bit	IR, SR, AR, HR, TC, LR
B: Bit				
IR, SR, AR, HR, TC, LR				

**Limitations**

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

**Description**

These six basic instructions correspond to the conditions on a ladder diagram. As described in *Section 4 Ladder-diagram 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.

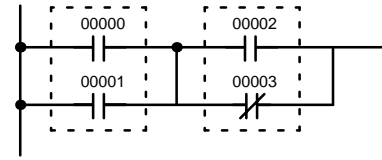
**Flags**

There are no flags affected by these instructions.

### 5-7-2 AND LOAD and OR LOAD

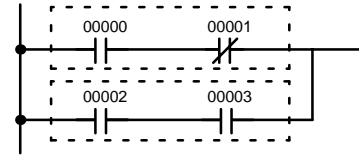
#### AND LOAD – AND LD

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 SSS. 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 4-3-6 *Logic Block Instructions*.

#### Flags

There are no flags affected by these instructions.

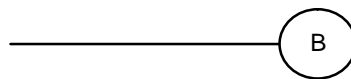
## 5-8 Bit Control Instructions

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

### 5-8-1 OUTPUT and OUTPUT NOT – OUT and OUT NOT

#### OUTPUT – OUT

Ladder Symbol

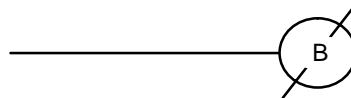


Operand Data Areas

<b>B: Bit</b>
IR, SR, AR, HR, LR, TR

#### OUTPUT NOT – OUT NOT

Ladder Symbol



Operand Data Areas

<b>B: Bit</b>
IR, SR, AR, HR, LR

#### Limitations

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

#### Description

OUT and OUT NOT are used to control the status of the designated bit according to the execution condition.

OUT turns ON the designated bit for an ON execution condition, and turns OFF the designated bit for an OFF execution condition. With a TR bit, OUT appears at a branching point rather than at the end of an instruction line. Refer to 4-3-8 *Branching Instruction Lines* for details.

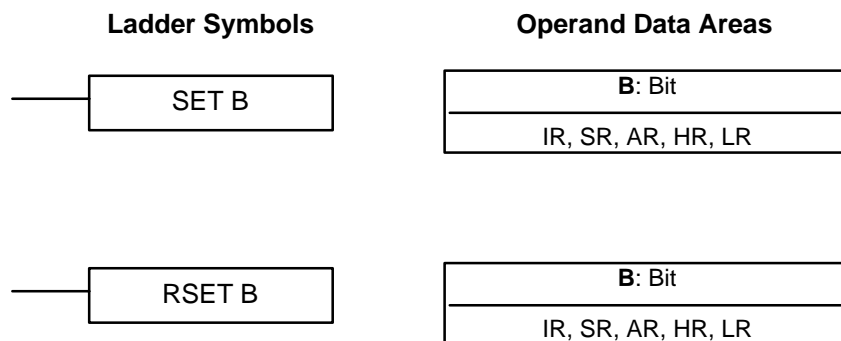
OUT NOT turns ON the designated bit for a OFF execution condition, and turns OFF the designated bit for an ON execution condition.

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 5-15-1 *TIMER – TIM* for details.

**Flags** There are no flags affected by these instructions.

### 5-8-2 SET and RESET – SET and RSET



**Description** SET turns the operand bit ON when the execution condition is ON, and does not affect the status of the operand bit when the execution condition is OFF. RSET turns the operand bit OFF when the execution condition is ON, and does not affect the status of the operand bit when the execution condition is OFF.

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

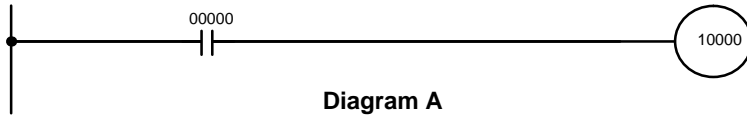
**Precautions** The status of operand bits for SET and RSET programmed between IL(02) and ILC(03) or JMP(04) and JME(05) will not change when the interlock or jump condition is met (i.e., when IL(02) or JMP(04) is executed with an OFF execution condition).

**Flags** There are no flags affected by these instructions.

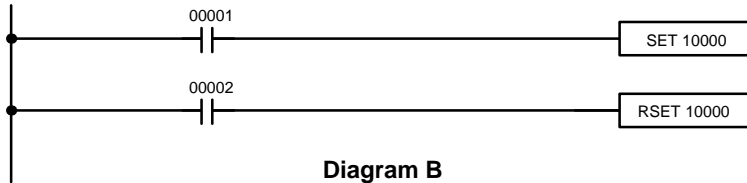
**Examples** The following examples demonstrate the difference between OUT and SET/RSET. In the first example (Diagram A), IR 10000 will be turned ON or OFF whenever IR 00000 goes ON or OFF.



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



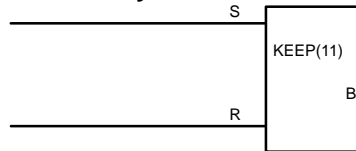
Address	Instruction	Operands
00000	LD	00000
00001	OUT	10000



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

### 5-8-3 KEEP – KEEP(11)

#### Ladder Symbol



#### Operand Data Areas

<b>B:</b> Bit
IR, SR, AR, HR, LR

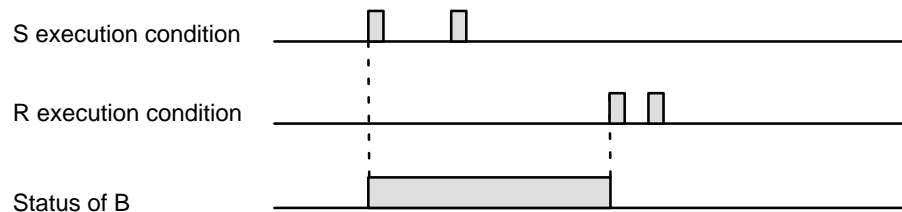
#### Limitations

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

#### 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.

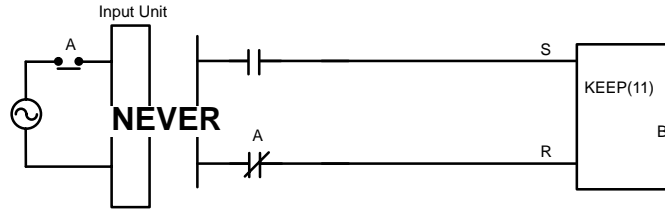


#### Flags

There are no flags affected by this instruction.

**Precautions**

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



Bits used in KEEP are not reset in interlocks. Refer to the 5-11 INTERLOCK – and INTERLOCK CLEAR IL(02) and ILC(03) for details.

**5-8-4 DIFFERENTIATE UP and DOWN – DIFU(13) and DIFD(14)**

**Ladder Symbols**



**Operand Data Areas**

<b>B: Bit</b>
IR, SR, AR, HR, LR

<b>B: Bit</b>
IR, SR, AR, HR, LR

**Limitations**

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

**Description**

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

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

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

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

**Flags**

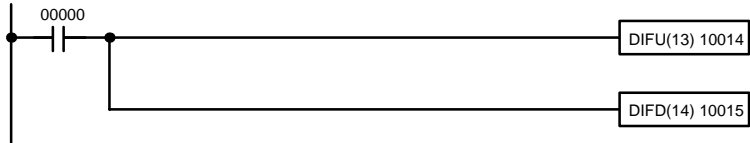
There are no flags affected by these instructions.

**Precautions**

rDIFU(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 5-11 INTERLOCK and INTERLOCK CLEAR – IL(02) and ILC(03), 5-12 JUMP and JUMP END – JMP(04) and JME(05), 5-25 Subroutine Instructions, and 5-26-8 INTERRUPT CONTROL – INT(89).

**Example**

In this example, IR 10014 will be turned ON for one cycle when IR 00000 goes from OFF to ON. IR 10015 will be turned ON for one cycle when IR 00000 goes from ON to OFF.



Address	Instruction	Operands
00000	LD	00000
00001	DIFU(13)	10014
00002	DIFD(14)	10015

### 5-9 NO OPERATION – NOP(00)

**Description**

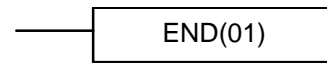
NOP(00) is not generally required in programming and there is no ladder symbol for it. When NOP(00) is found in a program, nothing is executed and the program execution moves to the next instruction. When memory is cleared prior to programming, NOP(00) is written at all addresses. NOP(00) can be input through the 00 function code.

**Flags**

There are no flags affected by NOP(00).

### 5-10 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**

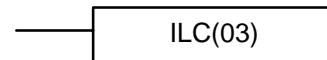
END(01) turns OFF the ER, CY, GR, EQ, and LE flags.

### 5-11 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 4-3-8 Branching Instruction Lines for basic descriptions of both methods.

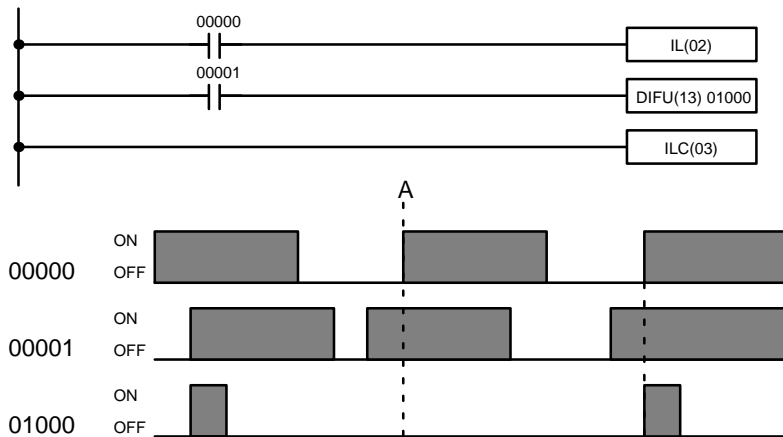
If the execution condition for IL(02) is OFF, the interlocked section between IL(02) and ILC(03) will be treated as shown in the following table:

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 other instructions	The instructions are not executed, and all IR, AR, LR, HR, and SR bits and words written to as operands in the instructions are turned OFF.

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.

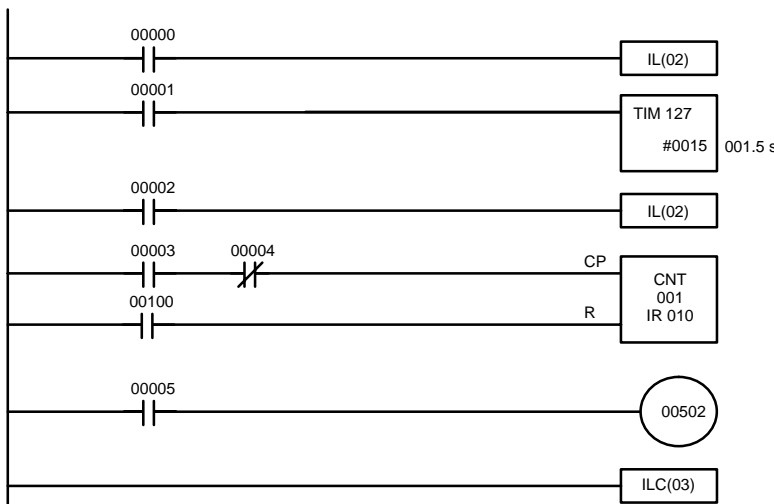
When more than one IL(02) is used with a single ILC(03), an error message will appear when the program check is performed, but execution will proceed normally.

**Flags**

There are no flags affected by these instructions.

**Example**

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

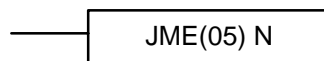
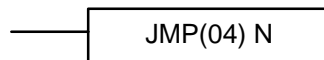


Address	Instruction	Operands
00000	LD	00000
00001	IL(02)	
00002	LD	00001
00003	TIM	127
		# 0015
00004	LD	00002
00005	IL(02)	
00006	LD	00003
00007	AND NOT	00004
00008	LD	00100
00009	LD	00100
00010	CNT	001
		010
00011	LD	00005
00012	OUT	00502
00013	ILC(03)	

When the execution condition for the first IL(02) is OFF, TIM 127 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 127 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.

## 5-12 JUMP and JUMP END – JMP(04) and JME(05)

**Ladder Symbols**



**Definer Values**

N: Jump number
#

N: Jump number
#

**Limitations**

Jump numbers 01 through 99 (00 through 49 in CPM1/CPM1A/SRM1 PCs) 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.

Jump numbers run from 00 through 99 in the CQM1 PCs and from 00 through 49 in the CPM1/CPM1A/SRM1 PCs.

**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) is ON, no jump is made and the program is executed consecutively as written. When the execution condition for JMP(04) is OFF, a jump is made to the JME(05) with the same jump number and the instruction following JME(05) is executed next.

If the jump number for JMP(04) is between 01 and 99, jumps, when made, will go immediately to JME(05) with the same jump number without executing any instructions in between. The status of timers, counters, bits used in OUT, bits used in OUT NOT, and all other status bits controlled by the instructions between JMP(04) and JME(05) will not be changed. Each of these jump numbers can be used to define only one jump. Because all of instructions between JMP(04) and JME(05) are skipped, jump numbers 01 through 99 (01 through 49 in CPM1/CPM1A/SRM1 PCs) can be used to reduce cycle time.

**Jump Number 00**

If the jump number for JMP(04) is 00, the CPU Unit 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 JME(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. This message also appears if JMP(04) 00 and JME(05) 00 are not used in pairs, but the program will execute properly as written.

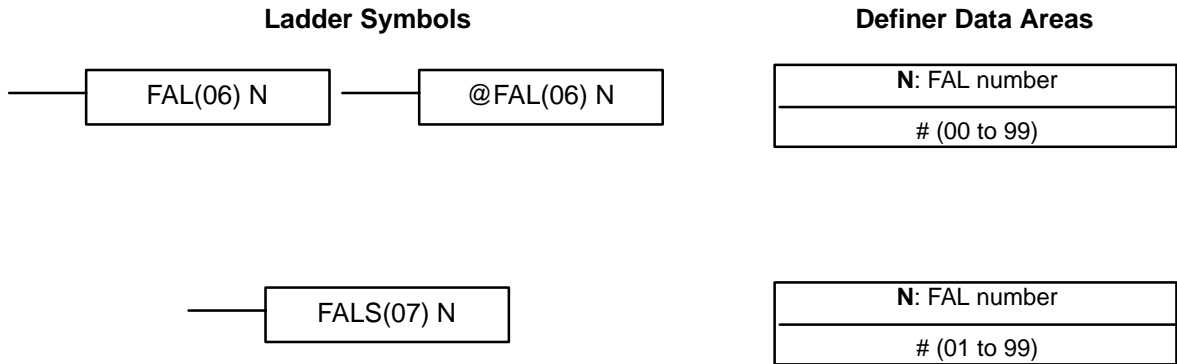
**Flags**

There are no flags affected by these instructions.

**Examples**

Examples of jump programs are provided in *4-3-9 Jumps*.

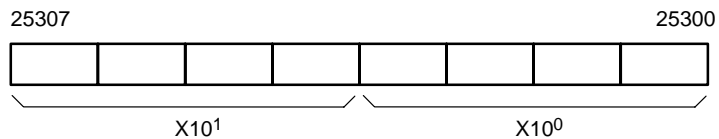
## 5-13 User Error Instructions: FAILURE ALARM AND RESET – FAL(06) and SEVERE FAILURE ALARM – FALS(07)



### Description

FAL(06) and FALS(07) are provided so that the programmer can output error numbers for use in operation, maintenance, and debugging. When executed with an ON execution condition, either of these instructions will output a FAL number to bits 00 to 07 of SR 253. The FAL number that is output can be between 01 and 99 and is input as the definer for FAL(06) or FALS(07). FAL(06) with a definer of 00 is used to reset this area (see below).

### FAL Area



FAL(06) produces a non-fatal error and FALS(07) produces a fatal error. When FAL(06) is executed with an ON execution condition, the ALARM/ERROR indicator on the front of the CPU Unit will flash, but PC operation will continue. When FALS(07) is executed with an ON execution condition, the ALARM/ERROR indicator will light and PC operation will stop.

The system also generates error codes to the FAL area.

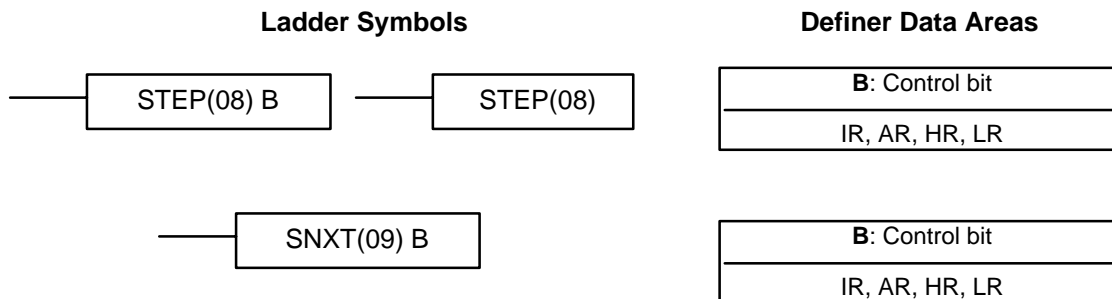
### Resetting Errors

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

FAL(06) 00 is also used to clear message programmed with the instruction, MSG(46).

If the FAL area cannot be cleared, as is generally the case when FALS(07) is executed, first remove the cause of the error and then clear the FAL area through the Programming Console or SSS.

## 5-14 Step Instructions: STEP DEFINE and STEP START–STEP(08)/SNXT(09)



### Limitations

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

### Description

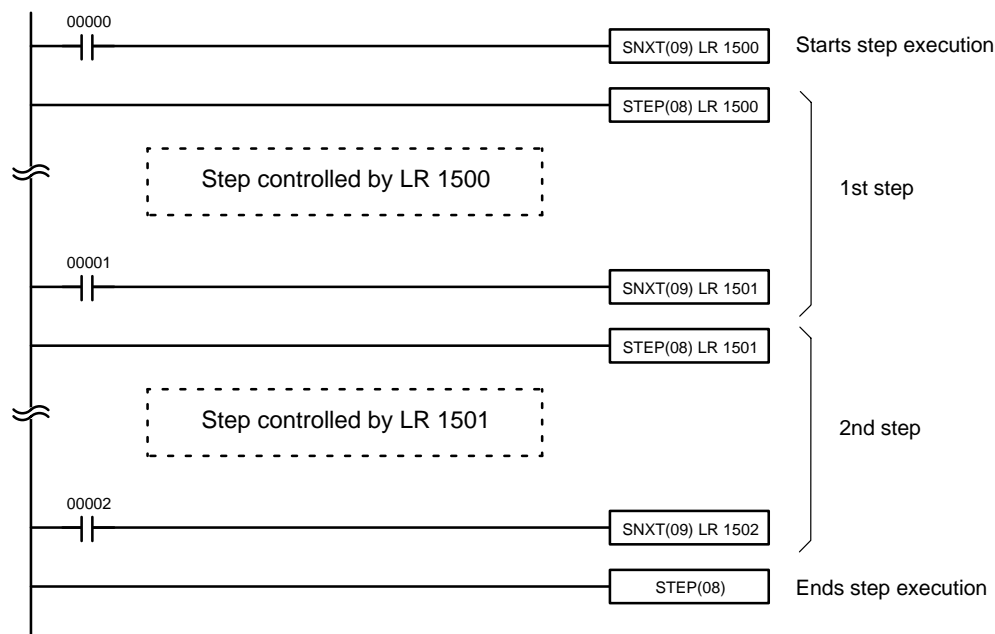
The step instructions STEP(08) and SNXT(09) are used together to set up breakpoints between sections in a large program so that the sections can be executed as units and reset upon completion. A section of program will usually be defined to correspond to an actual process in the application. (Refer to the application examples later in this section.) A step is like a normal programming code, except that certain instructions (i.e., END(01), IL(02)/ILC(03), JMP(04)/JME(05), and SBN(92)) may not be included.

STEP(08) uses a control bit in the IR or HR areas to define the beginning of a section of the program called a step. STEP(08) does not require an execution condition, i.e., its execution is controlled through the control bit. To start execution of the step, SNXT(09) is used with the same control bit as used for STEP(08). If SNXT(09) is executed with an ON execution condition, the step with the same control bit is executed. If the execution condition is OFF, the step is not executed. The SNXT(09) instruction must be written into the program so that it is executed before the program reaches the step it starts. It can be used at different locations before the step to control the step according to two different execution conditions (see example 2, below). Any step in the program that has not been started with SNXT(09) will not be executed.

Once SNXT(09) is used in the program, step execution will continue until STEP(08) is executed without a control bit. STEP(08) without a control bit must be preceded by SNXT(09) with a dummy control bit. The dummy control bit may be any unused IR or HR bit. It cannot be a control bit used in a STEP(08).



Execution of a step is completed either by execution of the next SNXT(09) or by turning OFF the control bit for the step (see example 3 below). When the step is completed, all of the IR and HR bits in the step are turned OFF and all timers in the step are reset to their SVs. Counters, shift registers, and bits used in KEEP(11) maintain status. Two simple steps are shown below.



Address	Instruction	Operands
00000	LD	00000
00001	SNXT(09)	LR 1500
00002	STEP(08)	LR 1500
Step controlled by LR 1500.		
00100	LD	00001
00101	SNXT(09)	LR 1501

Address	Instruction	Operands
00102	STEP(08)	LR 1501
Step controlled by LR 1501.		
00200	LD	00002
00201	SNXT(09)	LR 1502
00202	STEP(08)	---

Steps can be programmed in consecutively. Each step must start with STEP(08) and generally ends with SNXT(09) (see example 3, below, for an exception). When steps are programmed in series, three types of execution are possible: sequential, branching, or parallel. The execution conditions for, and the positioning of, SNXT(09) determine how the steps are executed. The three examples given below demonstrate these three types of step execution.

**Precautions**

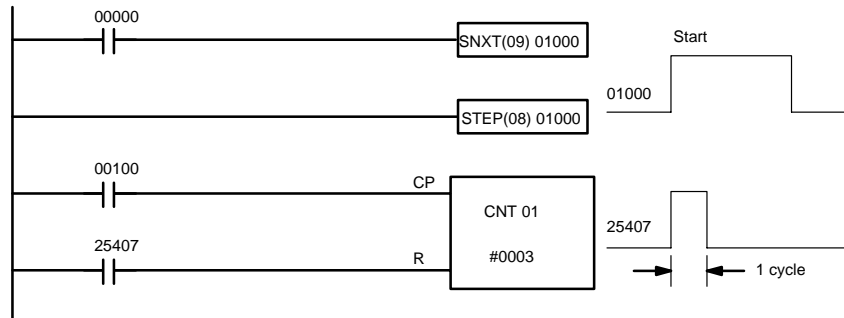
Interlocks, jumps, SBN(92), and END(01) cannot be used within step programs.

Bits used as control bits must not be used anywhere else in the program unless they are being used to control the operation of the step (see example 3, below). All control bits must be in the same word and must be consecutive.

If IR or LR bits are used for control bits, their status will be lost during any power interruption. If it is necessary to maintain status to resume execution at the same step, HR bits must be used.

Flags

**25407:** Step Start Flag; turns ON for one cycle when STEP(08) is executed and can be used to reset counters in steps as shown below if necessary.



Address	Instruction	Operands
00000	LD	00000
00001	SNXT(09)	01000
00002	STEP(08)	01000
00003	LD	00100

Address	Instruction	Operands
00004	LD	25407
00005	CNT	01
		# 0003

## 5-15 Timer and Counter Instructions

TIM and TIMH(15) are decrementing ON-delay timer instructions which require a TC number and a set value (SV). STIM(69) is used to control the interval timers, which are used to activate interrupt routines.

CNT is a decrementing counter instruction and CNTR(12) 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. CTBL(63), INT(89), and PRV(62) are used to manage the high-speed counter. INT(89) is also used to stop pulse output.

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

TC numbers run from 000 through 511 in the CQM1 PCs and from 000 through 127 in the CPM1/CPM1A/SRM1 PCs. No prefix is required when using a TC number as a definer in a timer or counter instruction. Once defined as a timer, a TC number can be prefixed with TIM for use as an operand in certain instructions. The TIM prefix is used regardless of the timer instruction that was used to define the timer. Once defined as a counter, a TC number can be prefixed with CNT for use as an operand in certain instructions. The CNT is also used regardless of the counter instruction that was used to define the counter.

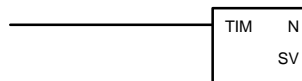
TC numbers can be designated as operands that require either bit or word data. When designated as an operand that requires bit data, the TC number accesses a bit that functions as a 'Completion Flag' that indicates when the time/count has expired, i.e., the bit, which is normally OFF, will turn ON when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

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

An SV can be input as a constant or as a word address in a data area. If an IR area word assigned to an Input Unit is designated as the word address, the Input Unit can be wired so that the SV can be set externally through thumbwheel switches or similar devices. Timers and counters wired in this way can only be set externally during RUN or MONITOR mode. All SVs, including those set externally, must be in BCD.

### 5-15-1 TIMER – TIM

#### Ladder Symbol



#### Definer Values

<b>N:</b> TC number
#

#### Operand Data Areas

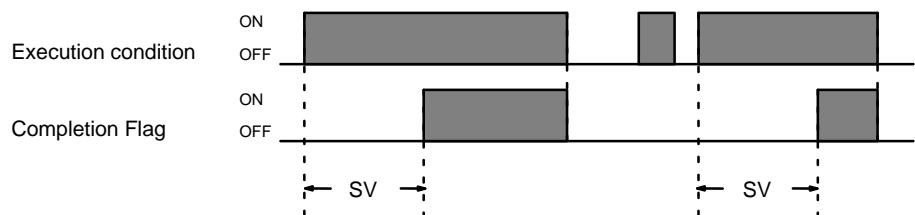
<b>SV:</b> Set value (word, BCD)
IR, SR, AR, DM, HR, LR, #

#### 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 numbers run from 000 through 511 in the CQM1 PCs and from 000 through 127 in the CPM1, CPM1A/SRM1 PCs. TC 000 through TC 015 (TC 000 through TC 003 in the CPM1/CPM1A/SRM1) should not be used in TIM if they are required for TIMH(15). Refer to 5-15-4 HIGH-SPEED TIMER – TIMH(15) for details.

#### Description

A timer is activated when its execution condition goes ON and is reset (to SV) when the execution condition goes OFF. Once activated, TIM measures in units of 0.1 second from the SV. If the execution condition remains ON long enough for TIM to time down to zero, the Completion Flag for the TC number used will turn ON and will remain ON until TIM is reset (i.e., until its execution condition is goes OFF). The following figure illustrates the relationship between the execution condition for TIM and the Completion Flag assigned to it.



#### Precautions

Timers in interlocked program sections are reset when the execution condition for IL(02) is OFF. Power interruptions also reset timers. If a timer that is not reset under these conditions is desired, SR area clock pulse bits can be counted to produce timers using CNT. Refer to 5-15-2 COUNTER – CNT for details.

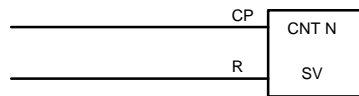
**Flags**

**ER:** SV is not in BCD.

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

**5-15-2 COUNTER – CNT**

**Ladder Symbol**



**Definer Values**

<b>N:</b> TC number
#

**Operand Data Areas**

<b>SV:</b> Set value (word, BCD)
IR, SR, AR, DM, HR, LR, #

**Limitations**

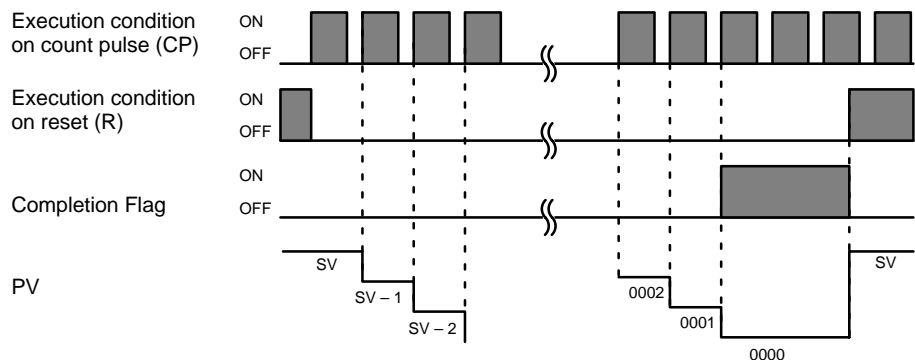
Each TC number can be used as the definer in only one TIMER or COUNTER instruction. TC numbers run from 000 through 511 in the CQM1 PCs and from 000 through 127 in the CPM1/CPM1A/SRM1 PCs.

**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**

In the following example, CNT is used to create extended timers by counting SR area clock pulse bits.

CNT 001 counts the number of times the 1-second clock pulse bit (SR 25502) goes from OFF to ON. Here again, IR 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 IR 01602 being turned ON.

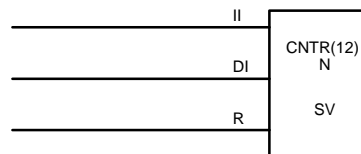


Address	Instruction	Operands
00000	LD	00000
00001	AND	25502
00002	LD NOT	00001
00003	CNT	001
		# 0700
00004	LD	CNT 001
00005	OUT	01602

**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.

**5-15-3 REVERSIBLE COUNTER – CNTR(12)**

**Ladder Symbol**



**Definer Values**

<b>N:</b> TC number
#

**Operand Data Areas**

<b>SV:</b> Set value (word, BCD)
IR, SR, AR, DM, HR, LR, #

**Limitations**

Each TC number can be used as the definer in only one TIMER or COUNTER instruction. TC numbers run from 000 through 511 in the CQM1 PCs and from 000 through 127 in the CPM1/CPM1A/SRM1 PCs.

**Description**

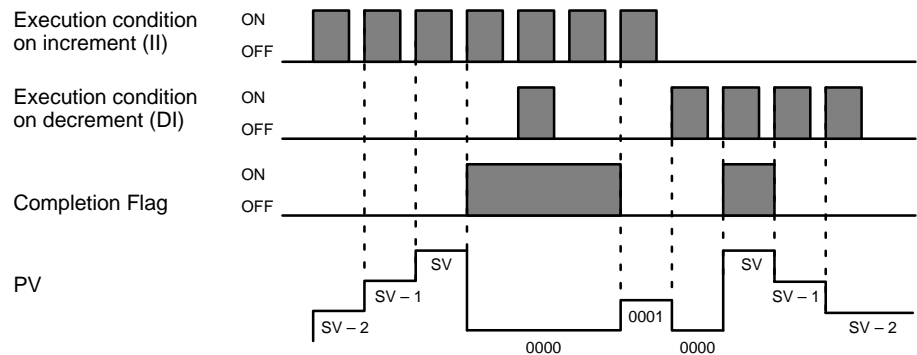
The CNTR(12) is a reversible, up/down circular counter, i.e., it is used to count between zero and SV according to changes in two execution conditions, those in the increment input (II) and those in the decrement input (DI).

The present value (PV) will be incremented by one whenever CNTR(12) is executed with an ON execution condition for II and the last execution condition for II was OFF. The present value (PV) will be decremented by one whenever CNTR(12) is executed with an ON execution condition for DI and the last execution condition for DI was OFF. If OFF to ON changes have occurred in both II and DI since the last execution, the PV will not be changed.

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

When decremented from 0000, the present value is set to SV and the Completion Flag is turned ON until the PV is decremented again. When incremented past the SV, the PV is set to 0000 and the Completion Flag is turned ON until the PV is incremented again.

CNTR(12) is reset with a reset input, R. When R goes from OFF to ON, the PV is reset to zero. The PV will not be incremented or decremented while R is ON. Counting will begin again when R goes OFF. The PV for CNTR(12) will not be reset in interlocked program sections or by the effects of power interruptions. Changes in II and DI execution conditions, the Completion Flag, and the PV are illustrated below starting from part way through CNTR(12) operation (i.e., when reset, counting begins from zero). PV line height is meant to indicate changes in the PV only.



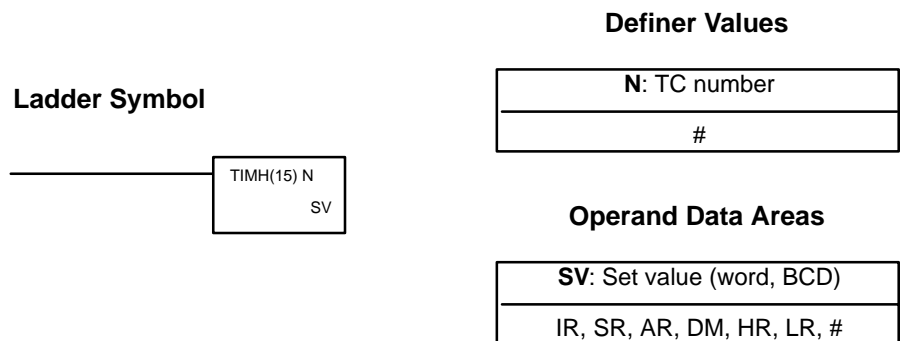
**Precautions**

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

**Flags**

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

**5-15-4 HIGH-SPEED TIMER – TIMH(15)**



**Limitations**

SV is between 00.00 and 99.99. (Although 00.00 and 00.01 may be set, 00.00 will disable the timer, i.e., turn ON the Completion Flag immediately, and 00.01 is not reliably scanned.) The decimal point is not entered. Each TC number can be used as the definer in only one TIMER or COUNTER instruction. Use TC numbers from 000 through 015 in the CQM1 PCs and from 000 through 003 in the CPM1/CPM1A/SRM1 PCs. High-speed timers with timer numbers TC 016 through TC 511 (TC 004 through TC 127 in the CPM1/CPM1A/SRM1) should not be used if the cycle time exceeds 10 ms.

**Description**

TIMH(15) operates in the same way as TIM except that TIMH measures in units of 0.01 second. Refer to 5-15-1 TIMER – TIM for operational details.

**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 5-15-2 COUNTER – CNT for details.

Timers in jumped program sections will not be reset when the execution condition for JMP(04) is OFF, but the timer will stop timing if jump number 00 is used. The timers will continue timing if jump numbers 01 through 99 (01 through 49 in CPM1/CPM1A/SRM1 PCs) are used.

**CQM1 Precautions**

High-speed timers with timer numbers TC 000 through TC 015 will not be inaccurate when the PC Setup (DM 6629) is set to perform interrupt processing on these timers.

High-speed timers with timer numbers TC 016 through TC 511 will be inaccurate when the cycle time exceeds 10 ms. If the cycle time is greater than 10 ms, use TC 000 through TC 015 and set DM 6629 for interrupt processing of the timer numbers used.

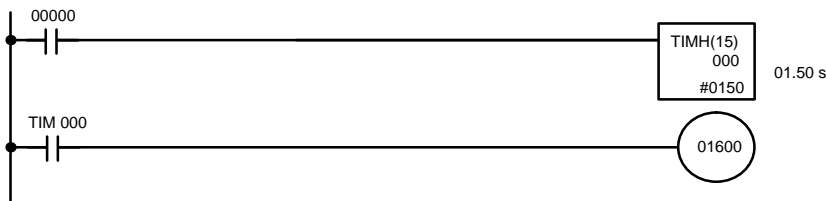
**Flags**

**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**

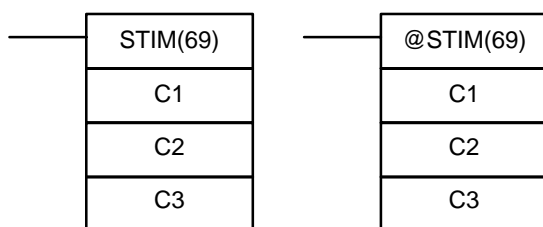
The following example shows a timer set with a constant. 01600 will be turned ON after 00000 goes ON and stays ON for at least 1.5 seconds. When 00000 goes OFF, the timer will be reset and 01600 will be turned OFF.



Address	Instruction	Operands
00000	LD	00000
00001	TIMH(15)	000
		# 0150
00002	LD	TIM 000
00003	OUT	01600

**5-15-5 INTERVAL TIMER – STIM(69)**

**Ladder Symbols**



**Operand Data Areas**

<b>C1:</b> Control data #1
000 to 008, 010 to 012
<b>C2:</b> Control data #2
IR, SR, AR, DM, HR, TC, LR, #
<b>C3:</b> Control data #3
IR, SR, AR, DM, HR, TC, LR, #

**Note** STIM(69) is an expansion instruction for the SRM1. The function code 69 is the factory setting and can be changed for the SRM1 if desired.

**Limitations (CQM1)**

C1 must be 000 to 008 or 010 to 012.

If C1 is 000 to 005, a constant greater than 0255 cannot be used for C3.

If C1 is 006 to 008, constants and DM 6143 to DM 6655 cannot be used for C2 or C3. If C1 is 010 to 012, both C2 and C3 must be set to 000.

**Limitations (CPM1/CPM1A/SRM1)**

C1 must be 000, 003, 006, or 010.

If C1 is 000 or 003, a constant greater than 0049 cannot be used for C3.

If C1 is 006, constants and DM 6143 to DM 6655 cannot be used for C2 or C3.

If C1 is 010, both C2 and C3 must be set to 000.

**Description**

STIM(69) is used to control the interval timers by performing four basic functions: starting the timer for a non-shot interrupt, starting the timer for scheduled interrupts, stopping the timer, and reading the timer's PV. Set the value of C1 to specify which of these functions will be performed and which of the three interval timers it will be performed on, as shown in the following table. Refer to 1-5-4, 1-6-4, and 1-7-2 for more detailed descriptions of using interval timer interrupts. STIM(69) is also described in more detail after the table.

Function	Timer	C1 value	Applicable PCs
Starting timers	0	000	CQM1/CPM1/ CPM1A/SRM1
	1	001	CQM1 only
	2	002	
Starting scheduled interrupts	0	003	CQM1/CPM1/ CPM1A/SRM1
	1	004	CQM1 only
	2	005	
Reading timer PV	0	006	CQM1/CPM1/ CPM1A/SRM1
	1	007	CQM1 only
	2	008	
Stopping timers	0	010	CQM1/CPM1/ CPM1A/SRM1
	1	011	CQM1 only
	2	012	

- Note**
1. In CQM1 PCs, interval timer 0 cannot be used when a pulse output is being output by the SPED(64) instruction.
  2. In CQM1 PCs, interval timer 2 cannot be used when high-speed counter 0 operation has been enabled in DM 6642 of the PC Setup.

**Starting Interrupts**

Set C1=000 to 002 to start timers 0 to 2 to activate a one-shot interrupt. Set C1=003 to 005 to start scheduled interrupts using timers 0 to 2.

C2, which specifies the timer's SV, can be a constant or the first of two words containing the SV. The settings are slightly different depending on the method used.

If C2 is a constant, it specifies the initial value of the decremting counter (BCD, 0000 to 9999). The decremting time interval is 1 ms.

If C2 is a word address, C2 specifies the initial value of the decremting counter (BCD, 0000 to 9999), and C2+1 specifies the decremting time interval (BCD, 0005 to 0320) in units of 0.1 ms. The decremting time interval can thus be 0.5 to 32 ms.

C3 specifies subroutine number 0000 to 0255 (0000 to 0127 in the CQM1-CPU11/21-E, 0000 to 0049 in the CPM1/CPM1A/SRM1 PCs).

- Note** The time required from interval timer start-up to time-up is:  
 (the content of C2) × (the content of C2+1) × 0.1 ms

**Reading Timer PVs**

Set C1=006 to 008 to read the PVs of timers 0 to 2.

C2 specifies the first of two destination words that will receive the timer's PV. C2 will receive the number of times the decremting counter has been decremting (BCD, 0000 to 9999) and C2+1 will receive the decremting time interval (BCD in 0.1 ms units).

C3 specifies the destination word that will receive the time which has elapsed since the last time the timer was decremting (BCD in 0.1 ms units).  
 (Must be equal to or less than the decremting time interval set in C2+1.)

- Note** The time that has elapsed since the timer was started is computed as follows:  
 (Content of C2 × (Content of C2 + 1) + Content of C3) × 0.1 ms



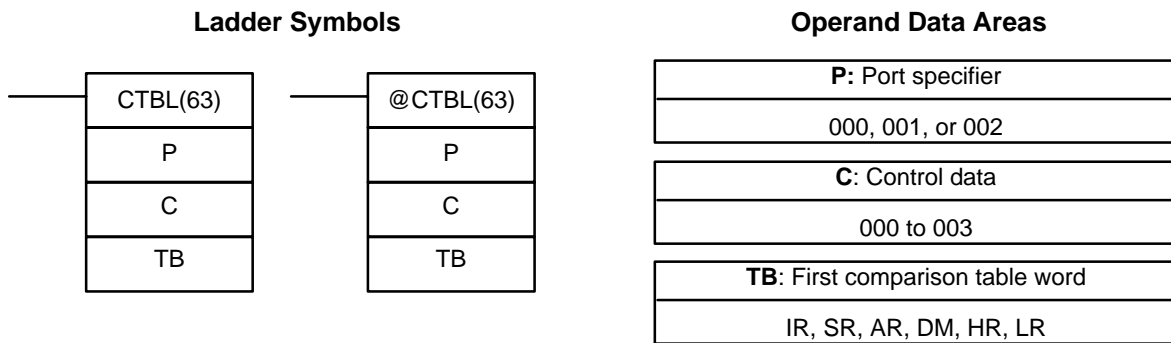
**Stopping Timers**

Set C1=010 to 012 to stop timers 0 to 2.  
C2 and C3 have no function and should both be set to 000.

**Flags**

**ER:** Interval timer 0 is started while a pulse output is operating. (C1=000 only)  
Interval timer 2 is started while the high-speed counter 0 is enabled (C1=002 only)  
Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)  
A data area boundary has been exceeded.

**5-15-6 REGISTER COMPARISON TABLE – CTBL(63)**



This instruction is not available for SRM1 PCs.

**Limitations**

The first and last comparison table words must be in the same data area. (The length of the comparison table varies according to the settings.)  
In the CQM1-CPU43-EV1, CTBL(63) cannot be used if the PC Setup (DM 6611) is set to pulse output mode.  
In the CPM1/CPM1A PCs, P must be 000.

**Description**

When the execution condition is OFF, CTBL(63) is not executed. When the execution condition is ON, CTBL(63) registers a comparison table for use with the high-speed counter PV. Depending on the value of C, comparison with the high-speed counter PV can begin immediately or it can be started separately with INI(61).  
The port specifier (P) specifies the high-speed counter that will be used in the comparison.

P	Function	Applicable PCs
000	Specifies high-speed counter 0.	CQM1/CPM1/CPM1A
001	Specifies high-speed counter 1.	CQM1 only
002	Specifies high-speed counter 2.	

The function of CTBL(63) is determined by the control data, C, as shown in the following table. These functions are described after the table.

C	CTBL(63) function
000	Registers a target value comparison table and starts comparison.
001	Registers a range comparison table and starts comparison.
002	Registers a target value comparison table. Start comparison with INI(61).
003	Registers a range comparison table. Start comparison with INI(61).

When the PV agrees with a target value or falls within a specified range, the specified subroutine is called and executed. Refer to 1-5-5 High-speed Counter 0 Interrupts (CQM1 PCs) or 1-6-5 High-speed Counter Interrupts (CPM1/CPM1A PCs) for more details on table comparison.

If the high-speed counter is enabled in the PC Setup (DM 6642), it will begin counting from zero when the CQM1 begins operation. The PV will not be compared to the comparison table until the table is registered and comparison is initiated with INI(61) or CTBL(63). Comparison can be stopped and started, or the PV can be reset with INI(61).

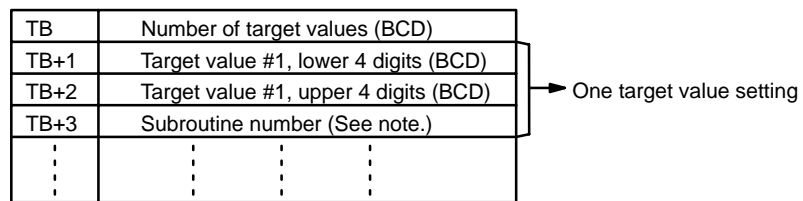
Once a comparison table has been registered, it is valid until the CQM1 is halted or until a error occurs in attempting to register a new table. The differentiated form of CTBL(63) is recommended when possible to reduce cycle time.

**Target Value Comparison**

A target value comparison table contains up to sixteen target values. For the CQM1-CPU4□-EV1 CPU Units, up to 48 target values can be registered. A sub-routine number is also registered for each target value. The corresponding sub-routine is called and executed when the PV matches a target value. (When interrupt processing is not required, an undefined subroutine number may be entered.)

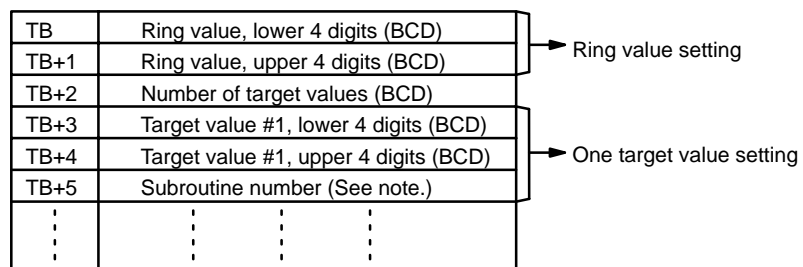
Target value comparisons are performed one item at a time in order of the comparison table. When the PV reaches the first target value in the table, the interrupt subroutine is executed and comparison continues to the next value in the table. When processing has been completed for the last target value in the table, comparison returns to the first value in the table and the process is repeated.

The following diagram shows the structure of a target value comparison table for use with high-speed counter 0, or high-speed counters 1 or 2 in linear mode.

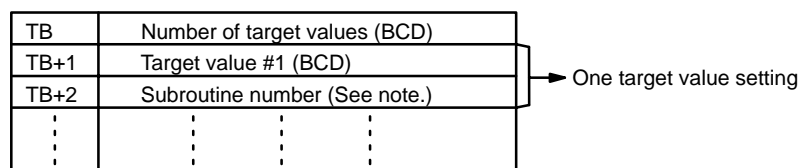


The following diagram shows the structure of a target value comparison table for use with high-speed counters 1 or 2 in ring mode. Input the target values in ascending or descending order.

The ring value specifies the number of points in the ring and the maximum count value (ring value = max. count value+1). Do not change the ring value while a comparison is in progress.



The following diagram shows the structure of a target value comparison table for use with absolute high-speed counters 1 and 2 (CQM1-CPU44-E/-EV1 only). Input the target values in ascending or descending order.



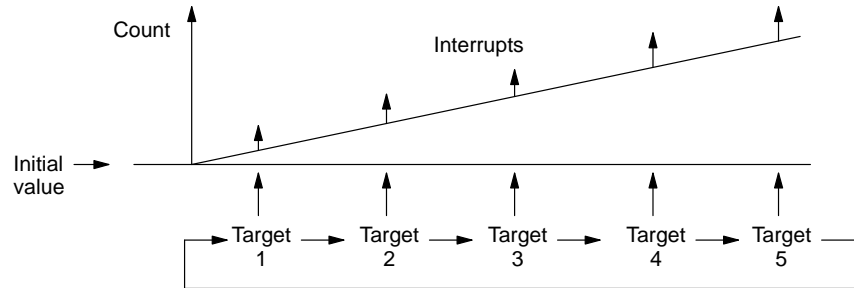
- Note** 1. The subroutine number can be 0000 to 0049 for CPM1/CPM1A PCs. In CQM1 PCs, the subroutine number can be F000 to F255 (F000 to F127)

for the CQM1-CPU11/21-E) to activate the subroutine when decrementing and can be 0000 to 0255 (0000 to 0127 for the CQM1-CPU11/21-E) to activate the subroutine when incrementing.

2. Allow an interval of at least 0.2 ms for interrupt processing when setting the target value for high-speed counters 1 and 2.

**Target Value Comparison Operation**

The following diagram illustrates the operation of target value comparisons for target values 1 through 5 set consecutively in the comparison table.



As illustrated above, the current count is compared with each target value in the order that they are registered in the target value comparison table. When the count is the same as the current target value, an interrupt is generated, and comparison starts with the next target value. When all target values in the comparison table have been matched and interrupts for them generated, the target value is reset to the first target value in the table and the operation is repeated.

**Range Comparison**

A range comparison table contains 8 ranges which are defined by an 8-digit lower limit and an 8-digit upper limit, as well as their corresponding subroutine numbers. The corresponding subroutine is called and executed when the PV falls within a given range. (When interrupt processing is not required, an undefined subroutine number may be entered.)

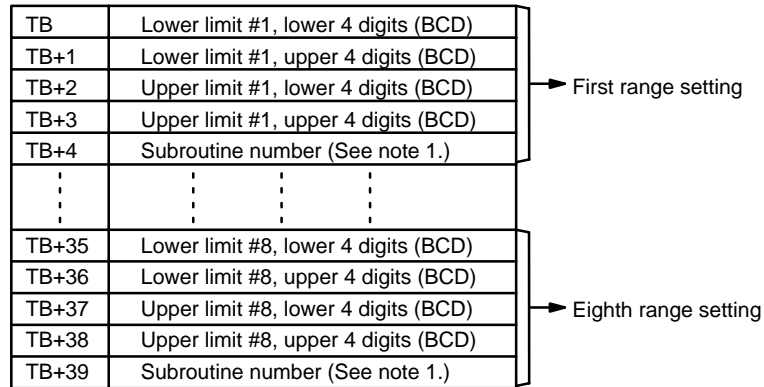
Always set 8 ranges. If fewer than 8 ranges are needed, set the remaining subroutine numbers to FFFF. If more than 8 ranges are needed, another comparison instruction such as BCMP(68) can be used to compare ranges with the high-speed counter PVs in IR 230 through IR 235 (SR 248 and SR 249 in CPM1/CPM1A PCs). Bear in mind that these words are refreshed just once each cycle.

There are flags in the AR area which indicate when a high-speed counter's PV falls within one or more of the 8 ranges. The flags turn ON when a PV is within the corresponding range.

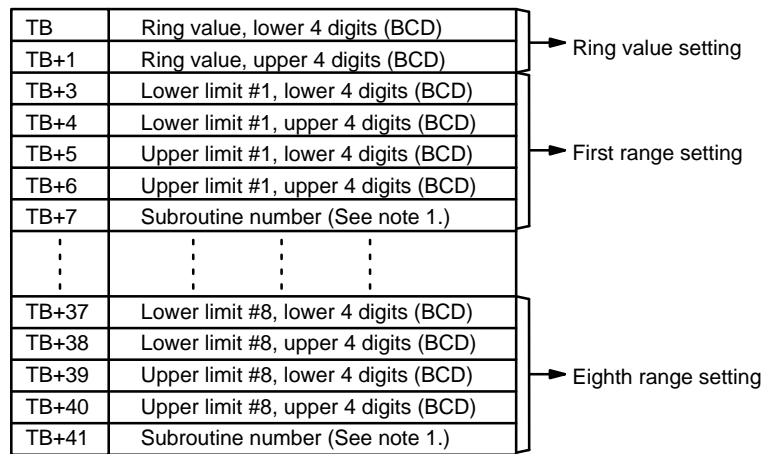
Counter	AR area flags
High-speed counter 0	AR 1100 to AR 1107 correspond to ranges 1 to 8.
High-speed counter 1	AR 0500 to AR 0507 correspond to ranges 1 to 8.
High-speed counter 2	AR 0600 to AR 0607 correspond to ranges 1 to 8.

**Note** CPM1/CPM1A PCs are equipped with high-speed counter 0 only.

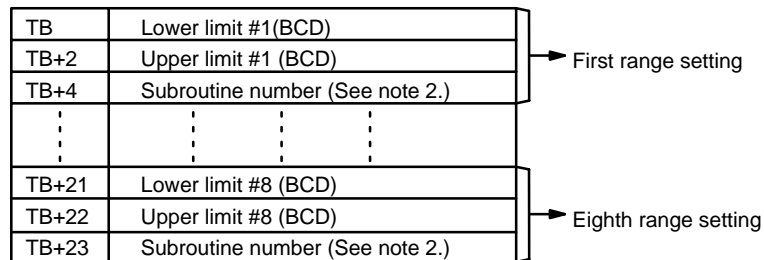
The following diagram shows the structure of a range comparison table for use with high-speed counter 0, or high-speed counters 1 or 2 in linear mode.



The following diagram shows the structure of a range comparison table for use with high-speed counters 1 or 2 in ring mode (CQM1 only). The ring value specifies the number of points in the ring and the maximum count value (ring value = max. count value+1). Do not change the ring value while a comparison is in progress.



The following diagram shows the structure of a range comparison table for use with absolute high-speed counters 1 and 2 (CQM1-CPU44-E/-EV1 only).



- Note**
1. The subroutine number can be 0000 to 0255 (0000 to 0127 for the CQM1-CPU11/21-E, 0000 to 0049 for the CPM1/CPM1A) and the subroutine will be executed as long as the counter's PV is within the specified range. A value of FFFF indicates that no subroutine is to be executed.
  2. The subroutine number can be 0000 to 0255 (0000 to 0127 for the CQM1-CPU11/21-E, 0000 to 0049 for the CPM1/CPM1A) to activate the subroutine when incrementing.
  3. Allow a time interval of at least 2 ms between the lower and upper limits (upper limit – lower limit > 0.002 × input pulse frequency) in range comparisons with high-speed counters 1 and 2.

The following table shows the possible values for target values, lower limit values, and upper limit values. The hexadecimal value F in the most significant digit of indicates that the value is negative.

Counter	Possible values
High-speed counter 0	Up/Down mode: F003 2767 to 0003 2767 Incrementing mode: 0000 0000 to 0006 5535
High-speed counters 1 and 2	Linear mode: F838 8607 to 0838 8607 Ring mode: 0000 0000 to 0006 4999
Absolute high-speed counters 1 and 2	BCD mode: 0000 to 4095 360° mode: 0000 to 0355 (5° units)

In 360° mode the absolute high-speed counter's angular values are internally converted to binary values. The binary value after conversion depends on the resolution selected in the PC Setup (DM 6643 and/or DM 6644). The following table shows the converted values for 5° to 45°.

Resolution	Converted value								
	5°	10°	15°	20°	25°	30°	35°	40°	45°
8-bit (0 to 255)	4	7	11	14	18	21	25	28	32
10-bit (0 to 1023)	14	28	43	57	71	85	100	114	128
12-bit (0 to 4095)	57	114	171	228	284	341	398	455	512

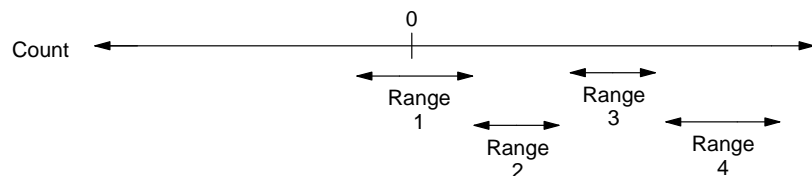
For higher values, find the converted value to the nearest 45° and add the remainder from the table. For example, to convert 145° into 8-bit resolution: 32×3 (for 135°) + 7 (for 10°) = 103.



**Caution** With 10-bit and 12-bit resolution, interrupt processing might not be triggered when the angular value matches the comparison value because the converted values do not match exactly.

**Range Comparison Operation**

The following diagram illustrates the operation of range comparisons for range settings 1 through 4 set consecutively in the comparison table.



As illustrated above, the current count is compared against all the comparison ranges at the same time and the result for each range is output.

**Flags**

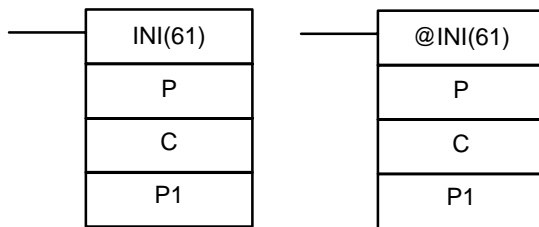
- ER:** There is an error in the high-speed counter's settings.  
The specified port and function are not compatible.
- There is a CTBL(63) instruction in the subroutine called by another CTBL(63) instruction.
- A CTBL(63) instruction using a different comparison format is executed during comparison.
- Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)
- The comparison table exceeds the data area boundary, or there is an error in the comparison table settings.
- CTBL(63) is executed in an interrupt subroutine while a pulse I/O or high-speed counter instruction is being executed in the main program.

- AR 05:** Flags AR 0500 to AR 0507 are turned ON to indicate when the PV of high-speed counter 1 is in ranges 1 to 8.
- AR 06:** Flags AR 0600 to AR 0607 are turned ON to indicate when the PV of high-speed counter 2 is in ranges 1 to 8.
- AR 11:** Flags AR 1100 to AR 1107 are turned ON to indicate when the PV of high-speed counter 0 is in ranges 1 to 8.

Subroutines are executed only once when the execution conditions are first met. AR status is refreshed only once per cycle. If conditions are met for more than one item in the table at the same time, the first item in the table takes priority.

### 5-15-7 MODE CONTROL – INI(61)

#### Ladder Symbols



#### Operand Data Areas

<b>P:</b> Port specifier
000, 001, or 002
<b>C:</b> Control data
000 to 003
<b>P1:</b> First PV word
IR, SR, AR, DM, HR, LR

This instruction is not available for SRM1 PCs.

#### Limitations

In the CPM1/CPM1A PCs, P must be 000 and C must be 000 to 003.  
 In CQM1 PCs, P must be 000, 001, or 002 and C must be 000 to 003.  
 P1 must be 000 unless C is 002.  
 P1 and P1+1 must be in the same data area.  
 DM 6143 to DM 6655 cannot be used for P1.

#### Description

When the execution condition is OFF, INI(61) is not executed. When the execution condition is ON, INI(61) is used to control high-speed counter operation and stop pulse output.  
 The port specifier (P) specifies the high-speed counter or pulse output that will be controlled.

P	Function
000	Specifies high-speed counter 0 or a pulse output from a bit.
001	Specifies high-speed counter 1 or a pulse output from port 1.
002	Specifies high-speed counter 2 or a pulse output from port 2.

**Note** CPM1/CPM1A PCs are equipped with high-speed counter 0 only.

The function of INI(61) is determined by the control data, C. (P1 and P1+1 contain the new high-speed counter PV when changing the PV.)

C	P1	INI(61) function
000	000	Starts CTBL(63) table comparison.
001	000	Stops CTBL(63) table comparison.
002	New high-speed counter PV	Changes high-speed counter PV.
003	000	Stops pulse output.

#### CTBL(63) Table Comparison

If C is 000 or 001, INI(61) starts or stops comparison of the high-speed counter's PV to the comparison table registered with CTBL(63). Refer to 1-5-5 *High-speed Counter 0 Interrupts* (CQM1 PCs) or 1-6-5 *High-speed Counter Interrupts* (CPM1/CPM1A PCs) for details on table comparison.

**PV Change**

If C is 002, INI(61) changes the high-speed counter's PV to the 8-digit value in P1 and P1+1.

With high-speed counter 0, the PV can be F003 2767 to 0003 2767 in Up/Down Mode, or 0000 0000 to 0006 5535 in Incremental Mode. The hexadecimal value F in the most significant digit of PV indicates that PV is negative.

Leftmost 4 digits P1+1	Rightmost 4 digits P1	Up/Down Mode F0032767 to 00032767	Incrementing Mode 00000000 to 00065535
---------------------------	--------------------------	--------------------------------------	---

With high-speed counters 1 and 2 (CQM1 only), the PV can be F838 8607 to 0838 8607 in Linear Mode, or 0000 0000 to 0006 4999 in Ring Mode.

The hexadecimal value F in the most significant digit of PV indicates that PV is negative.

Leftmost 4 digits P1+1	Rightmost 4 digits P1	Linear Mode F8388607 to 08388607 (-8,388,607 to 8,388,607)	Ring Mode 00000000 to 00064999
---------------------------	--------------------------	--	-----------------------------------

**Note** The PV of absolute high-speed counters 1 and 2 (CQM1-CPU44-E/-EV1 only) cannot be changed.

**Stop Pulse Output**

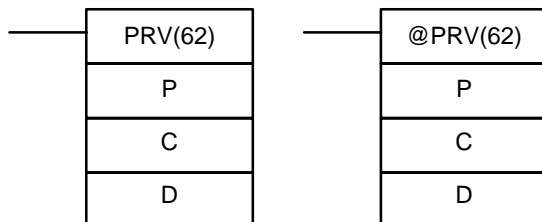
If C is 003, INI(61) stops pulse output. Refer to *1-3 Pulse Output Functions (CQM1 Only)* and *1-4 Pulse Output Function (CPM1A Only)* for details on pulse output.

**Flags**

- ER:** The specified port and function are not compatible.
- Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)
- P1+1 exceeds the data area boundary. (C=002)
- There is an error in the operand settings.
- INI(61) is executed in an interrupt subroutine while a pulse I/O or high-speed counter instruction is being executed in the main program.

**5-15-8 HIGH-SPEED COUNTER PV READ – PRV(62)**

**Ladder Symbols**



**Operand Data Areas**

<b>P:</b> Port specifier
000, 001, or 002
<b>C:</b> Control data
000, 001, or 002
<b>D:</b> First destination word
IR, SR, AR, DM, HR, LR

This instruction is not available for SRM1 PCs.

**Limitations**

- In the CPM1/CPM1A PCs, P and C must be 000.
- In CQM1 PCs, P and C must be 000, 001, or 002.
- D and D+1 must be in the same data area.
- DM 6143 to DM 6655 cannot be used for D.

**Description**

When the execution condition is OFF, PRV(62) is not executed. When the execution condition is ON, PRV(62) reads data specified by P and C and writes it to D or D and D+1.

The port specifier (P) specifies the high-speed counter or pulse output.

P	Function
000	Specifies high-speed counter 0 or a pulse output from a bit.
001	Specifies high-speed counter 1 or a pulse output from port 1.
002	Specifies high-speed counter 2 or a pulse output from port 2.

**Note** CPM1/CPM1A PCs are equipped with high-speed counter 0 only.

The control data, C, determines which type of data will be accessed.

C	Data	Destination word(s)
000	High-speed counter PV	D and D+1
001	Status of high-speed counter or pulse output	D
002	Range comparison results	D

**High-speed Counter PV (C=000)**

If C is 000, PRV(62) reads the specified high-speed counter's PV and writes the 8-digit value in D and D+1.

With high-speed counter 0, the PV can be F003 2767 to 0003 2767 in Up/Down Mode, or 0000 0000 to 0006 5535 in Incremental Mode. The hexadecimal value F in the most significant digit of PV indicates that the PV is negative.

Leftmost 4 digits	Rightmost 4 digits	Up/Down Mode	Incrementing Mode
D+1	D	F0032767 to 00032767	00000000 to 00065535

With high-speed counters 1 and 2 (CQM1 only), the PV can be F838 8607 to 0838 8607 in Linear Mode, or 0000 0000 to 0006 4999 in Ring Mode. The hexadecimal value F in the most significant digit of PV indicates that the PV is negative.

Leftmost 4 digits	Rightmost 4 digits	Linear Mode	Ring Mode
D+1	D	F8388607 to 08388607 (-8,388,607 to 8,388,607)	00000000 to 00064999

With absolute high-speed counters 1 and 2, the PV can be 0000 0000 to 0000 4095 in BCD Mode, or 0000 0000 to 0000 0359 in 360° Mode.

Leftmost 4 digits	Rightmost 4 digits	BCD Mode	360° Mode
D+1	D	0000 0000 to 0000 4095	0000 0000 to 0000 0359

**High-speed Counter or Pulse Output Status (C=001)**

If C is 001 (CQM1 only), PRV(62) reads the operating status of the specified high-speed counter or pulse output and writes the data to D. Refer to 1-3-5 Determining the Status of Ports 1 and 2 for details on determining the status of pulse outputs.

The following table shows the function of bits in D for high-speed counters 1 and 2, and pulse outputs from ports 1 and 2 (CQM1-CPU43-E/-EV1 only). Bits not listed in the table are not used and will always be 0.

Bit	Function
00	High-speed counter comparison status. (0: Stopped; 1: Comparing)
01	High-speed counter underflow/overflow. (0: Normal; 1: Underflow/Overflow occurred.)
04	Deceleration of pulse frequency. (0: Not specified; 1: Specified.)
05	Total number of pulses specified. (0: Not specified; 1: Specified.)
06	Pulse output completed. (0: Not completed; 1: Completed)
07	Pulse output status (0: Stopped; 1: Outputting)

For absolute high-speed counters 1 and 2 (CQM1-CPU44-E/-EV1 only), Bit 00 of D indicates the comparison status (0: Stopped; 1: Comparing). The other bits in D (01 through 15) are not used and will always be 0.



**Note** These flags are in AR 05 and AR 06, but those words are normally refreshed only once each cycle, so the data obtained with PRV(62) will be more up-to-date.

**Range Comparison Results (C=002)** If C is 002 (CQM1 only), PRV(62) reads the results of the comparison of the PV to the 8 ranges defined by CTBL(63) and writes this data to D. Bits 00 through 07 of D contain the Comparison Result flags for ranges 1 to 8. (0: Not in range; 1: In range)

**Note** These flags are in AR 05 and AR 06, but those words are normally refreshed only once each cycle, so the data obtained with PRV(62) will be more up-to-date.

### Flags

**ER:** The specified port and function are not compatible.  
Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)

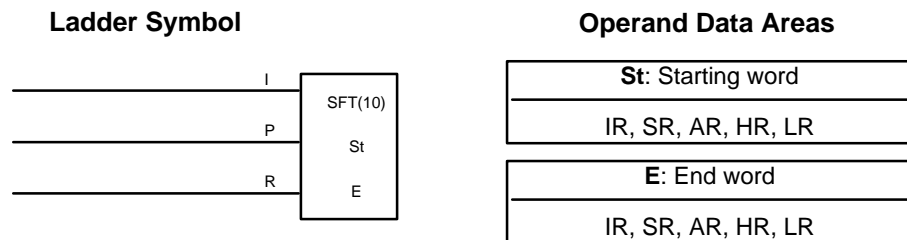
D+1 exceeds the data area boundary. (C=000)

There is an error in the operand settings.

PRV(62) is executed in an interrupt subroutine while a pulse I/O or high-speed counter instruction is being executed in the main program.

## 5-16 Shift Instructions

### 5-16-1 SHIFT REGISTER – SFT(10)



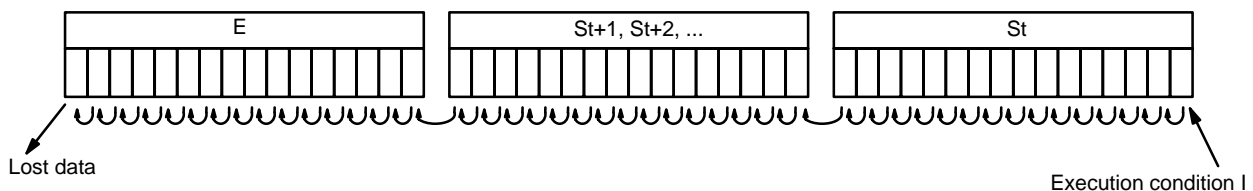
#### Limitations

E must be greater 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)), an error (“COIL/OUT 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.

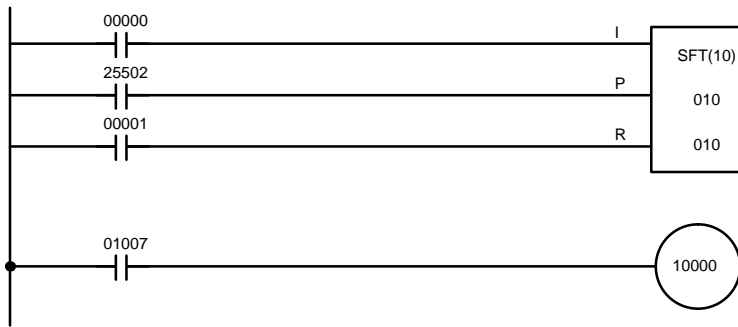
When execution condition R goes ON, all bits in the shift register will be turned OFF (i.e., set to 0) and the shift register will not operate until R goes OFF again.

#### Flags

There are no flags affected by SFT(10).

**Example**

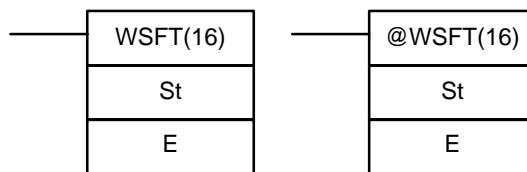
The following example uses the 1-second clock pulse bit (25502) so that the execution condition produced by 00000 is shifted into IR 010 every second. Output 10000 is turned ON whenever a “1” is shifted into 01007.



Address	Instruction	Operands
00000	LD	00000
00001	LD	25502
00002	LD	00001
00003	SFT(10)	010 010
00004	LD	01007
00005	OUT	10000

**5-16-2 WORD SHIFT – WSFT(16)**

**Ladder Symbols**



**Operand Data Areas**

<b>St: Starting word</b>
IR, SR, AR, DM, HR, LR
<b>E: End word</b>
IR, SR, AR, DM, HR, LR

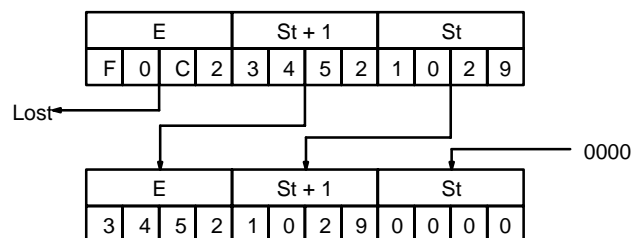
**Limitations**

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

DM 6144 to DM 6655 cannot be used for St or E.

**Description**

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

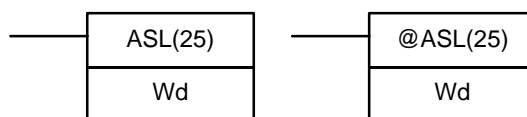


**Flags**

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

**5-16-3 ARITHMETIC SHIFT LEFT – ASL(25)**

**Ladder Symbols**



**Operand Data Areas**

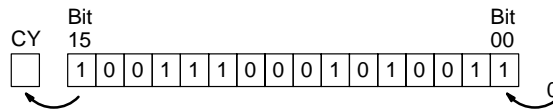
<b>Wd: Shift word</b>
IR, SR, AR, DM, HR, LR

**Limitations**

DM 6144 to DM 6655 cannot be used for Wd.

**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.



**Precautions**

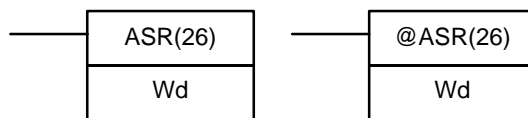
A 0 will be shifted into bit 00 every cycle if the undifferentiated form of ASL(25) is used. Use the differentiated form (@ASL(25)) or combine ASL(25) with DIFU(13) or DIFD(14) to shift just one time.

**Flags**

- ER:** Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)
- CY:** Receives the status of bit 15.
- EQ:** ON when the content of Wd is zero; otherwise OFF.

**5-16-4 ARITHMETIC SHIFT RIGHT – ASR(26)**

**Ladder Symbols**



**Operand Data Areas**

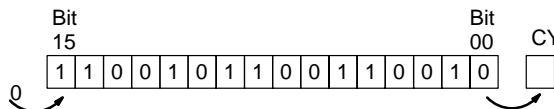
<b>Wd:</b> Shift word
IR, SR, AR, DM, HR, LR

**Limitations**

DM 6144 to DM 6655 cannot be used for Wd.

**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.



**Precautions**

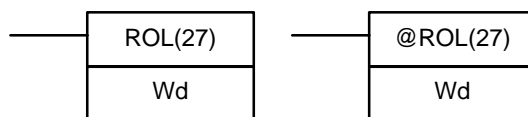
A 0 will be shifted into bit 15 every cycle if the undifferentiated form of ASR(26) is used. Use the differentiated form (@ASR(26)) or combine ASR(26) with DIFU(13) or DIFD(14) to shift just one time.

**Flags**

- ER:** Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)
- CY:** Receives the data of bit 00.
- EQ:** ON when the content of Wd is zero; otherwise OFF.

**5-16-5 ROTATE LEFT – ROL(27)**

**Ladder Symbols**



**Operand Data Areas**

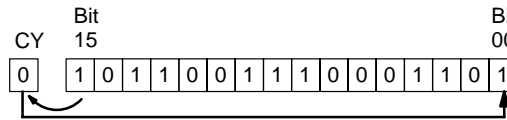
<b>Wd:</b> Rotate word
IR, SR, AR, DM, HR, LR

**Limitations**

DM 6144 to DM 6655 cannot be used for Wd.

**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 executing ROL(27).

CY will be shifted into bit 00 every cycle if the undifferentiated form of ROL(27) is used. Use the differentiated form (@ROL(27)) or combine ROL(27) with DIFU(13) or DIFD(14) to shift just one time.

**Flags**

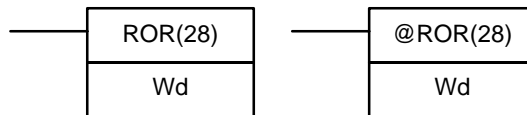
**ER:** Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)

**CY:** Receives the data of bit 15.

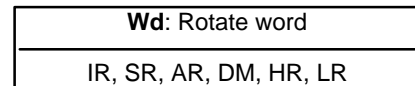
**EQ:** ON when the content of Wd is zero; otherwise OFF.

**5-16-6 ROTATE RIGHT – ROR(28)**

**Ladder Symbols**



**Operand Data Areas**

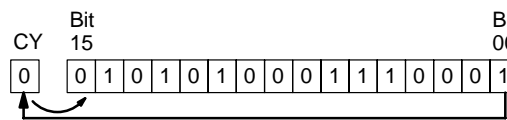


**Limitations**

DM 6144 to DM 6655 cannot be used for Wd.

**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).

CY will be shifted into bit 15 every cycle if the undifferentiated form of ROR(28) is used. Use the differentiated form (@ROR(28)) or combine ROR(28) with DIFU(13) or DIFD(14) to shift just one time.

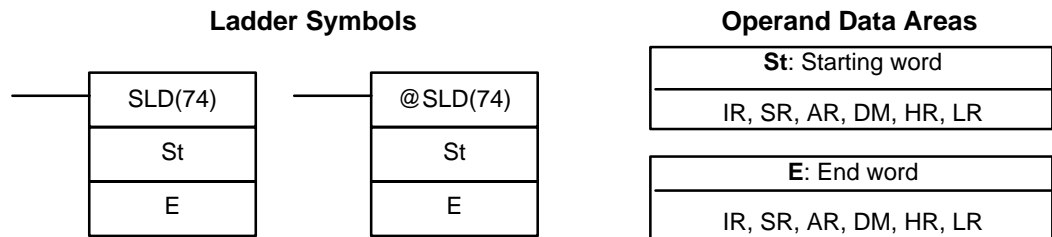
**Flags**

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

**CY:** Receives the data of bit 00.

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

### 5-16-7 ONE DIGIT SHIFT LEFT – SLD(74)



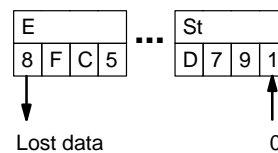
**Limitations**

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

DM 6144 to DM 6655 cannot be used for St or E.

**Description**

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



**Precautions**

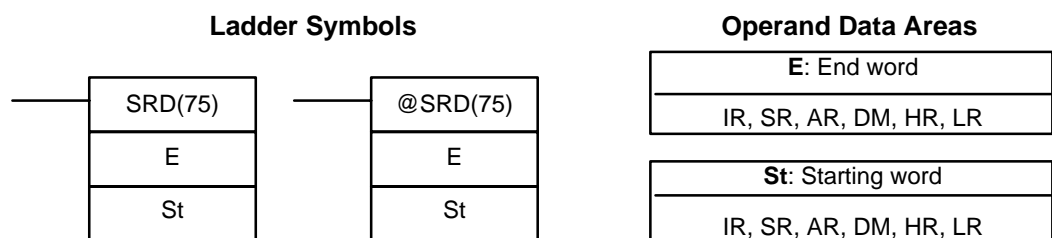
If a power failure occurs during a shift operation across more than 50 words, the shift operation might not be completed.

A 0 will be shifted into the least significant digit of St every cycle if the undifferentiated form of SLD(74) is used. Use the differentiated form (@SLD(74)) or combine SLD(74) with DIFU(13) or DIFD(14) to shift just one time.

**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.)

### 5-16-8 ONE DIGIT SHIFT RIGHT – SRD(75)

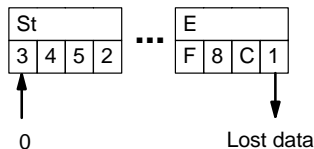


**Limitations**

St and E must be in the same data area, and E must be less than or equal to St.  
DM 6144 to DM 6655 cannot be used for St or E.

**Description**

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



**Precautions**

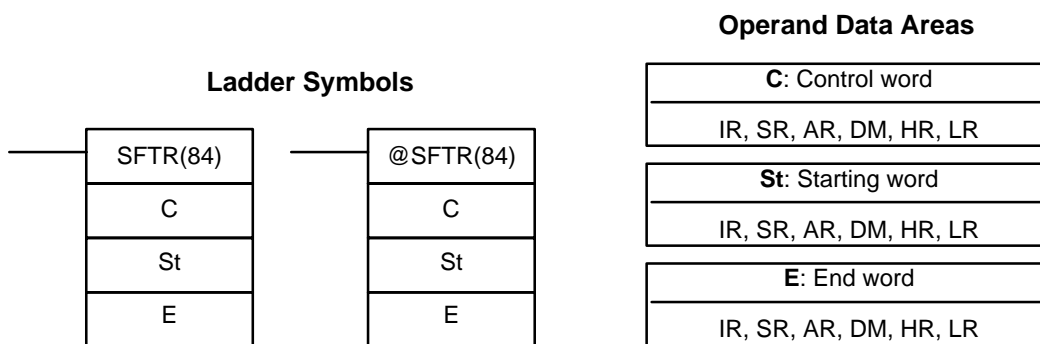
If a power failure occurs during a shift operation across more than 50 words, the shift operation might not be completed.

A 0 will be shifted into the most significant digit of St every cycle if the undifferentiated form of SRD(75) is used. Use the differentiated form (@SRD(75)) or combine SRD(75) with DIFU(13) or DIFD(14) to shift just one time.

**Flags**

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

**5-16-9 REVERSIBLE SHIFT REGISTER – SFTR(84)**



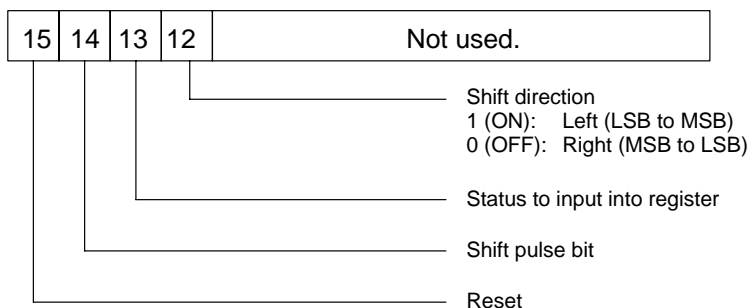
**Limitations**

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

DM 6144 to DM 6655 cannot be used for C, St, or E.

**Description**

SFTR(84) is used to create a single- or multiple-word shift register that can shift data to either the right or the left. To create a single-word register, designate the same word for St and E. The control word provides the shift direction, the status to be put into the register, the shift pulse, and the reset input. The control word is allocated as follows:



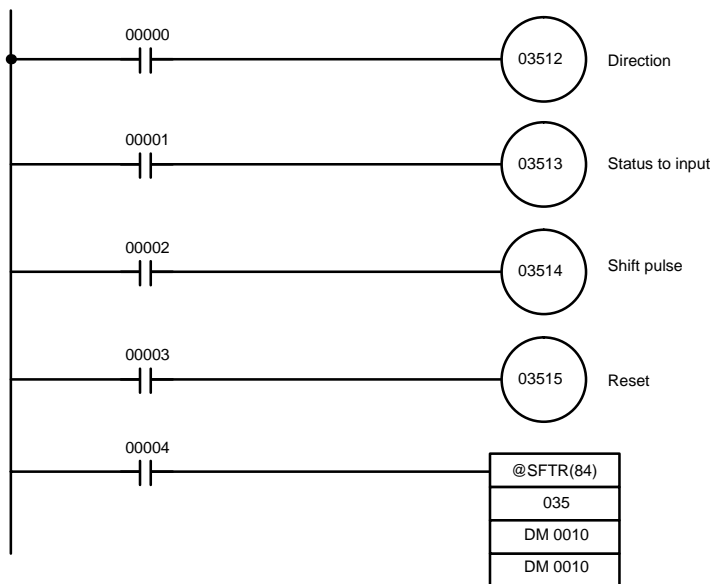
The data in the shift register will be shifted one bit in the direction indicated by bit 12, shifting one bit out to CY and the status of bit 13 into the other end whenever SFTR(84) is executed with an ON execution condition as long as the reset bit is OFF and as long as bit 14 is ON. If SFTR(84) is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

**Flags**

- ER:** St and E are not in the same data area or ST is greater than E.  
Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)
- CY:** Receives the status of bit 00 of St or bit 15 of E, depending on the shift direction.

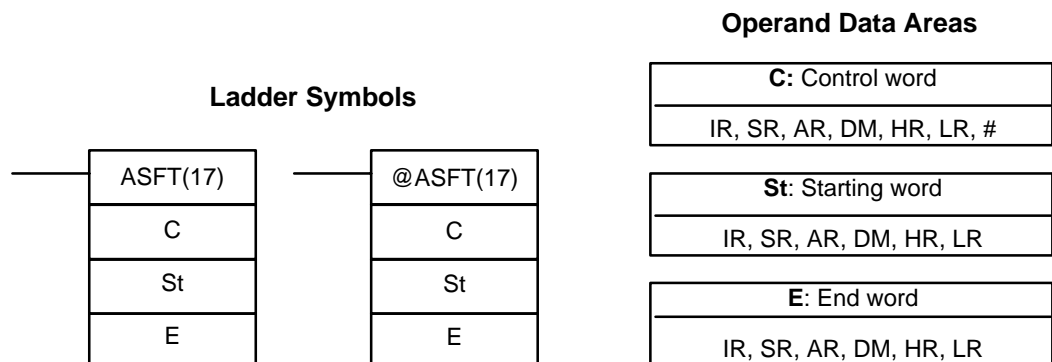
**Example**

In the following example, IR 00000, IR 00001, IR 00002, and IR 00003 are used to control the bits of C used in @SFTR(84). The shift register is in DM 0010, and it is controlled through IR 00004.



Address	Instruction	Operands
00000	LD	00000
00001	OUT	03512
00002	LD	00001
00003	OUT	03513
00004	LD	00002
00005	OUT	03514
00006	LD	00003
00007	OUT	03515
00008	LD	00004
00009	@SFT(10)	
		035
		DM 0010
		DM 0010

**5-16-10 ASYNCHRONOUS SHIFT REGISTER – ASFT(17)**



**Note** ASFT(17) is an expansion instruction for the SRM1. The function code 17 is the factory setting and can be changed for the SRM1 if desired.

**Limitations**

- St and E must be in the same data area, and E must be greater than or equal to St.
- DM 6144 to DM 6655 cannot be used for St or E.



**Description**

When the execution condition is OFF, ASFT(17) does nothing and the program moves to the next instruction. When the execution condition is ON, ASFT(17) is used to create and control a reversible asynchronous word shift register between St and E. This register only shifts words when the next word in the register is zero, e.g., if no words in the register contain zero, nothing is shifted. Also, only one word is shifted for each word in the register that contains zero. When the contents of a word are shifted to the next word, the original word's contents are set to zero. In essence, when the register is shifted, each zero word in the register trades places with the next word. (See *Example* below.)

The shift direction (i.e. whether the "next word" is the next higher or the next lower word) is designated in C. C is also used to reset the register. All of any portion of the register can be reset by designating the desired portion with St and E.

**Control Word**

Bits 00 through 12 of C are not used. Bit 13 is the shift direction: turn bit 13 ON to shift down (toward lower addressed words) and OFF to shift up (toward higher addressed words). Bit 14 is the Shift Enable Bit: turn bit 14 ON to enable shift register operation according to bit 13 and OFF to disable the register. Bit 15 is the Reset bit: the register will be reset (set to zero) between St and E when ASFT(17) is executed with bit 15 ON. Turn bit 15 OFF for normal operation.

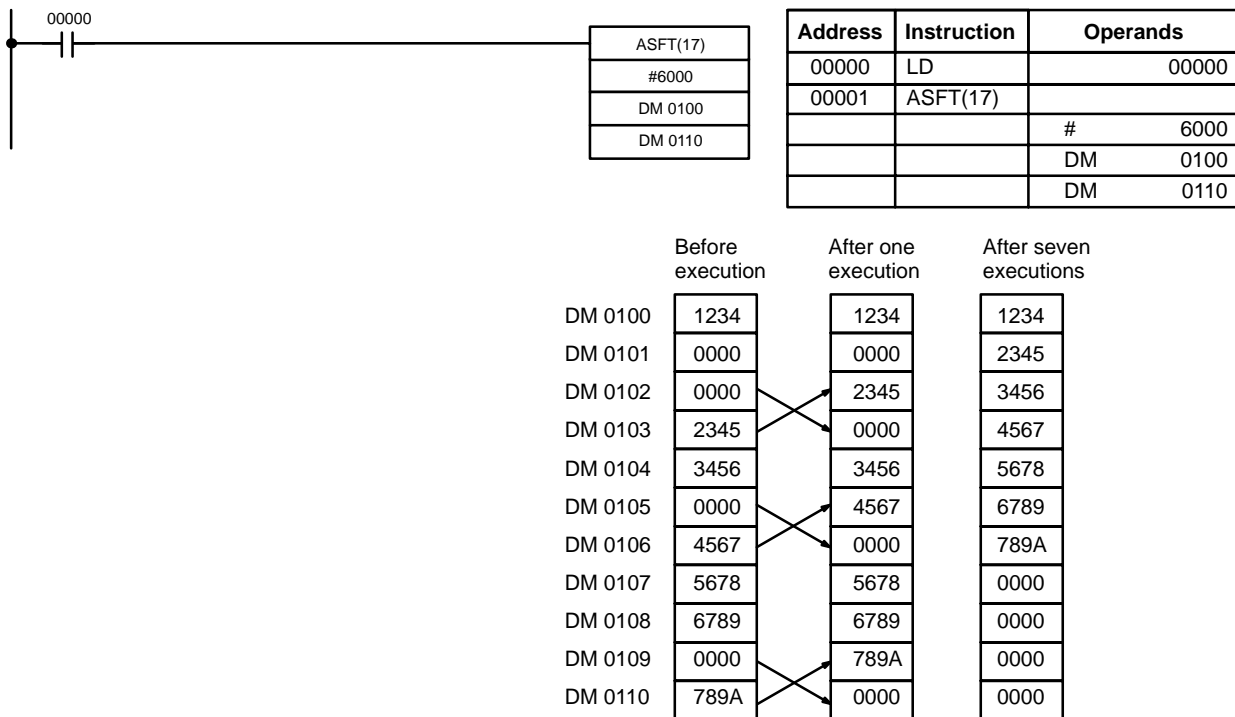
**Note** If the non-differentiated form of ASFT(17) is used, data will be shifted every cycle while the execution condition is ON. Use the differentiated form to prevent this.

**Flags**

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

**Example**

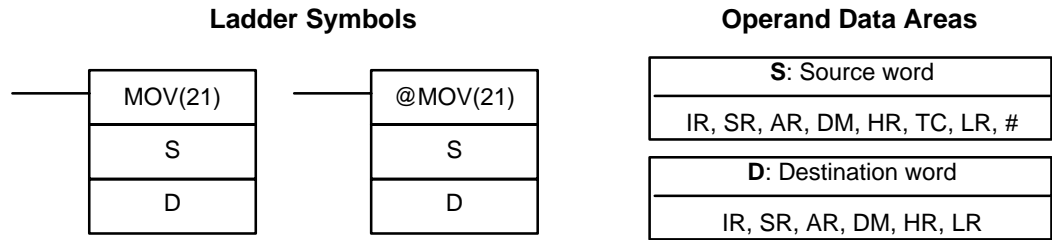
The following example shows instruction ASFT(17) used to shift words in an 11-word shift register created between DM 0100 and DM 0110 with C=#6000. Non-zero data is shifted towards St (DM 0110).



**Note** The zeroes are shifted "upward" if C=4000, and the entire shift register is set to zero if C=8000.

## 5-17 Data Movement Instructions

### 5-17-1 MOVE – MOV(21)

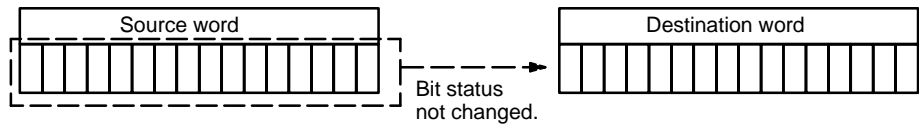


**Limitations**

DM 6144 to DM 6655 cannot be used for D.

**Description**

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



**Precautions**

TC numbers cannot be designated as D to change the PV of the timer or counter. You can, however, easily change the PV of a timer or a counter by using BSET(71).

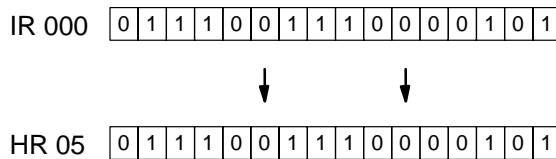
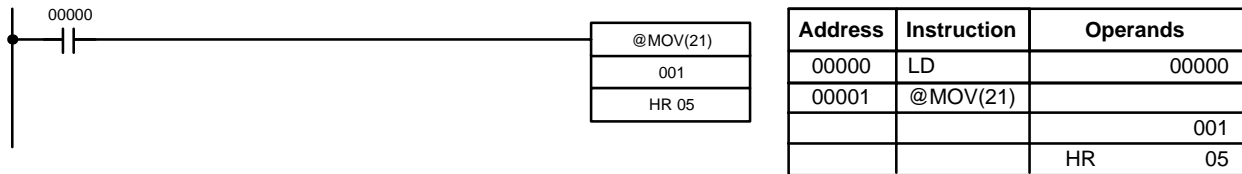
**Flags**

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

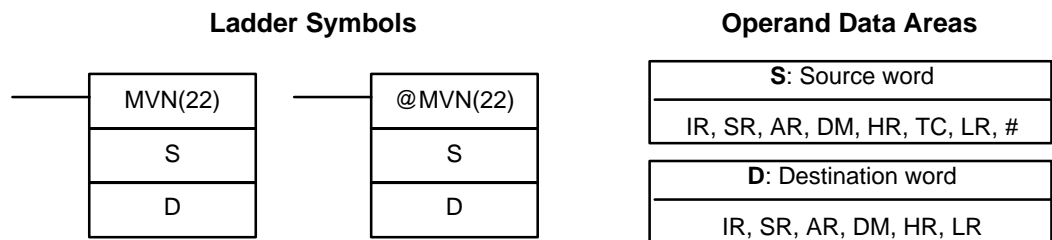
**EQ:** ON when all zeros are transferred to D.

**Example**

The following example shows @MOV(21) being used to copy the content of IR 001 to HR 05 when IR 00000 goes from OFF to ON.



### 5-17-2 MOVE NOT – MVN(22)



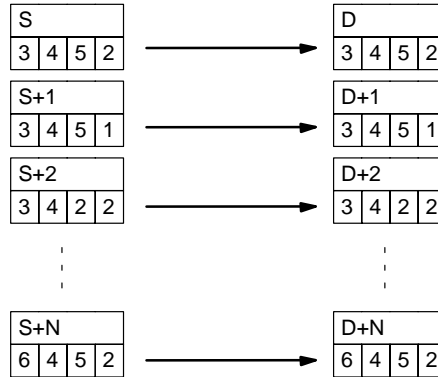
**Limitations**

DM 6144 to DM 6655 cannot be used for D.



**Description**

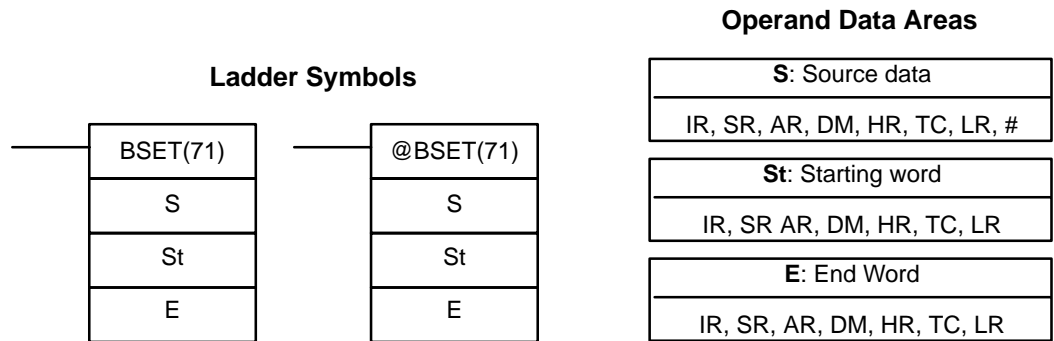
When the execution condition is OFF, XFER(70) is not executed. When the execution condition is ON, XFER(70) copies the contents of S, S+1, ..., S+N to D, D+1, ..., D+N.



**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.)

**5-17-4 BLOCK SET – BSET(71)**

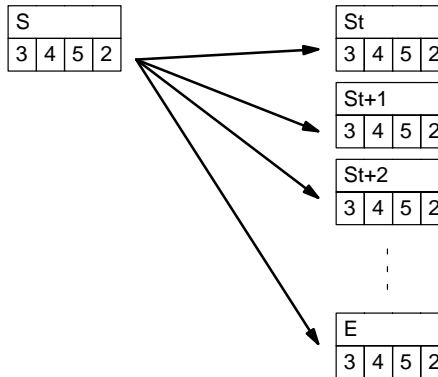


**Limitations**

St must be less than or equal to E, and St and E must be in the same data area. DM 6144 to DM 6655 cannot be used for St or E.

**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.



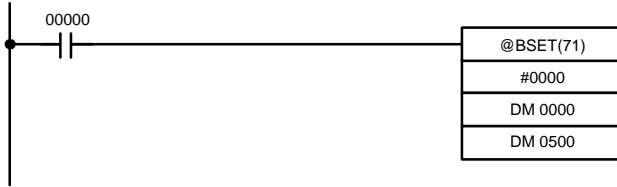
BSET(71) can be used to change timer/counter PV. (This cannot be done with MOV(21) or MVN(22).) BSET(71) can also be used to clear sections of a data area, i.e., the DM area, to prepare for executing other instructions. It can also be used to clear words by transferring all zeros.

**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**

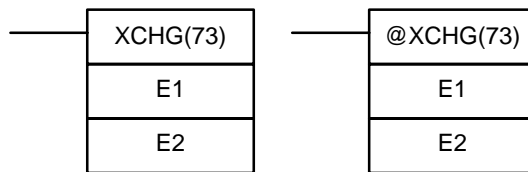
The following example shows how to use BSET(71) to copy a constant (#0000) to a block of the DM area (DM 0000 to DM 0500) when IR 00000 is ON.



Address	Instruction	Operands
00000	LD	00000
00001	@BSET(71)	
		# 0000
		DM 0000
		DM 0500

### 5-17-5 DATA EXCHANGE – XCHG(73)

**Ladder Symbols**



**Operand Data Areas**

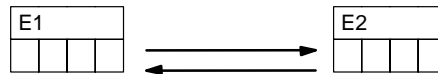
<b>E1: Exchange word 1</b>
IR, SR, AR, DM, HR, TC, LR
<b>E2: Exchange word 2</b>
IR, SR, AR, DM, HR, TC, LR

**Limitations**

DM 6144 to DM 6655 cannot be used for E1 or E2.

**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.



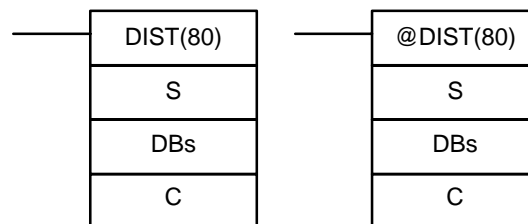
If you want to exchange content of blocks whose size is greater than 1 word, use work words as an intermediate buffer to hold one of the blocks using XFER(70) three times.

**Flags**

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

### 5-17-6 SINGLE WORD DISTRIBUTE – DIST(80)

**Ladder Symbols**



**Operand Data Areas**

<b>S: Source data</b>
IR, SR, AR, DM, HR, TC, LR, #
<b>DBs: Destination base word</b>
IR, SR, AR, DM, HR, TC, LR
<b>C: Control word (BCD)</b>
IR, SR, AR, DM, HR, TC, LR, #

**Limitations**

C must be BCD.  
DM 6144 to DM 6655 cannot be used for DBs or C.

**Description**

DIST(80) can be used for single-word distribution or for a stack operation depending on the content of the control word, C.

**Single-word Distribution**

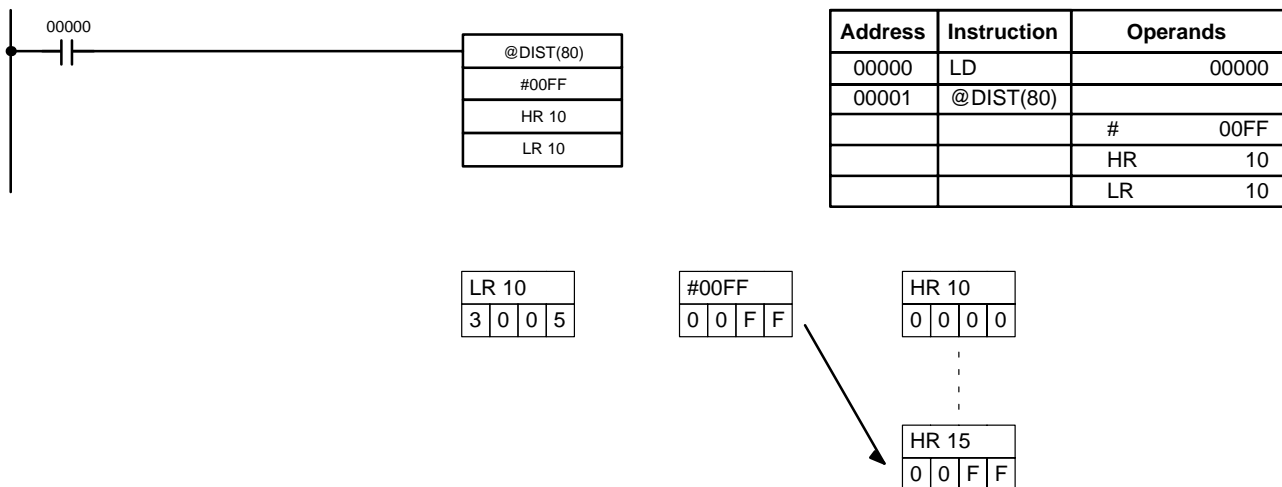
When bits 12 to 15 of C=0 to 8, DIST(80) can be used for a single word distribute operation. The entire contents of C specifies an offset, Of.

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.

**Note** DBs and DBs+Of must be in the same data area and cannot be between DM 6144 and DM 6655.

**Example**

The following example shows how to use DIST(80) to copy #00FF to HR 10 + Of. The content of LR 10 is #3005, so #00FF is copied to HR 15 (HR 10 + 5) when IR 00000 is ON.



**Stack Operation**

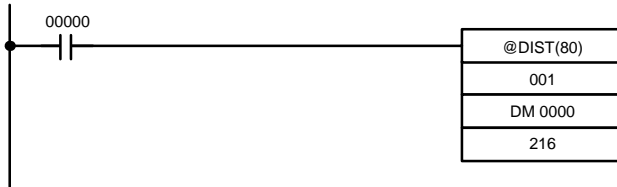
When bits 12 to 15 of C=9, DIST(80) can be used for a stack operation. The other 3 digits of C specify the number of words in the stack (000 to 999). The content of DBs is the stack pointer.

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+1+the content of DBs. In other words, 1 and the content of DBs are added to DBs to determine the destination word. The content of DBs is then incremented by 1.

- Note**
1. DIST(80) will be executed every cycle unless the differentiated form (@DIST(80)) is used or DIST(80) is used with DIFU(13) or DIFD(14).
  2. Be sure to initialize the stack pointer before using DIST(80) as a stack operation.

**Example**

The following example shows how to use DIST(80) to create a stack between DM 0001 and DM 0005. DM 0000 acts as the stack pointer.



Address	Instruction	Operands
00000	LD	00000
00001	@DIST(80)	
		001
		DM 0000
		216

IR 001	FFFF
IR 216	9005

DM 0000	0000
DM 0001	0000
DM 0002	0000
DM 0003	0000
DM 0004	0000
DM 0005	0000

First execution  
 →  
 Stack pointer incremented

DM 0000	0001
DM 0001	FFFF
DM 0002	0000
DM 0003	0000
DM 0004	0000
DM 0005	0000

Second execution  
 →  
 Stack pointer incremented

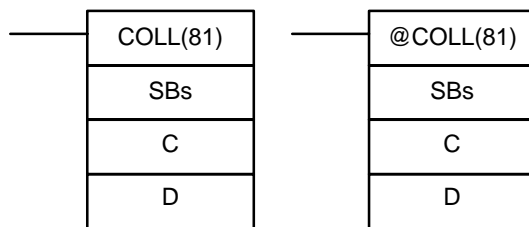
DM 0000	0002
DM 0001	FFFF
DM 0002	FFFF
DM 0003	0000
DM 0004	0000
DM 0005	0000

**Flags**

- ER:** The offset or stack length in the control word 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.)  
 During stack operation, the value of the stack pointer+1 exceeds the length of the stack.
- EQ:** ON when the content of S is zero; otherwise OFF.

**5-17-7 DATA COLLECT – COLL(81)**

**Ladder Symbols**



**Operand Data Areas**

<b>SBs:</b> Source base word
IR, SR, AR, DM, HR, TC, LR
<b>C:</b> Control word (BCD)
IR, SR, AR, DM, HR, TC, LR, #
<b>D:</b> Destination word
IR, SR, AR, DM, HR, TC, LR

**Limitations**

- C must be BCD.
- DM 6144 to DM 6655 cannot be used for D.

**Description**

COLL(81) can be used for data collection, an FIFO stack operation, or an LIFO stack operation depending on the content of the control word, C.

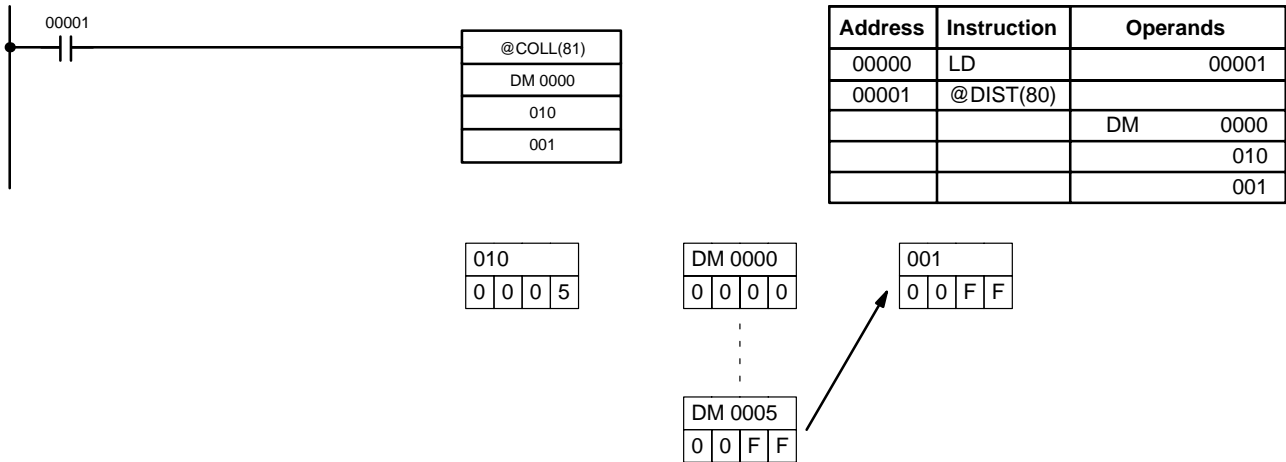
**Data Collection**

When bits 12 to 15 of C=0 to 7, COLL(81) is used for data collection. The entire contents of C specifies an offset, Of.  
 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.

**Note** SBs and SBs+Of must be in the same data area.

**Example**

The following example shows how to use COLL(81) to copy the content of DM 0000+Of to IR 001. The content of O10 is #0005, so the content of DM 0005 (DM 0000 + 5) is copied to IR 001 when IR 00001 is ON.



**FIFO Stack Operation**

When bits 12 to 15 of C=9, COLL(81) can be used for an FIFO stack operation. The other 3 digits of C specify the number of words in the stack (000 to 999). The content of SBs is the stack pointer.

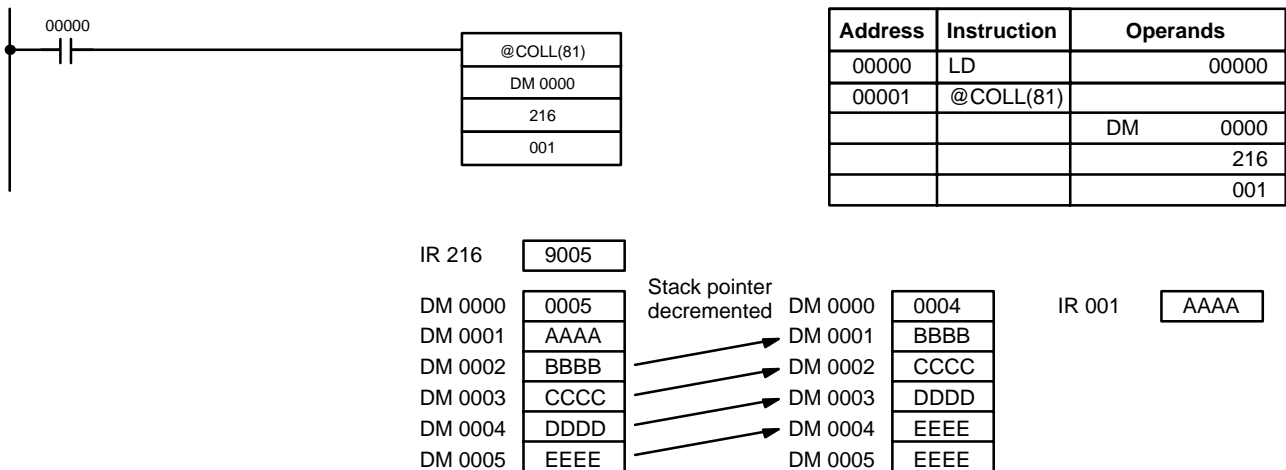
When the execution condition is ON, COLL(81) shifts the contents of each word within the stack down by one address, finally shifting the data from SBs+1 (the first value written to the stack) to the destination word (D). The content of the stack pointer (SBs) is then decremented by one.

**Note** COLL(81) will be executed every cycle unless the differentiated form (@COLL(81)) is used or COLL(81) is used with DIFU(13) or DIFD(14).

**Example**

The following example shows how to use COLL(81) to create a stack between DM 0001 and DM 0005. DM 0000 acts as the stack pointer.

When IR 00000 goes from OFF to ON, COLL(81) shifts the contents of DM 0002 to DM 0005 down by one address, and shifts the data from DM 0001 to IR 001. The content of the stack pointer (DM 0000) is then decremented by one.



**LIFO Stack Operation**

When bits 12 to 15 of C=8, COLL(81) can be used for an LIFO stack operation. The other 3 digits of C specify the number of words in the stack (000 to 999). The content of SBs is the stack pointer.



When the execution condition is ON, COLL(81) copies the data from the word indicated by the stack pointer (SBs+the content of SBs) to the destination word (D). The content of the stack pointer (SBs) is then decremented by one. The stack pointer is the only word changed in the stack.

**Note** COLL(81) will be executed every cycle unless the differentiated form (@DIST(80)) is used or DIST(80) is used with DIFU(13) or DIFD(14).

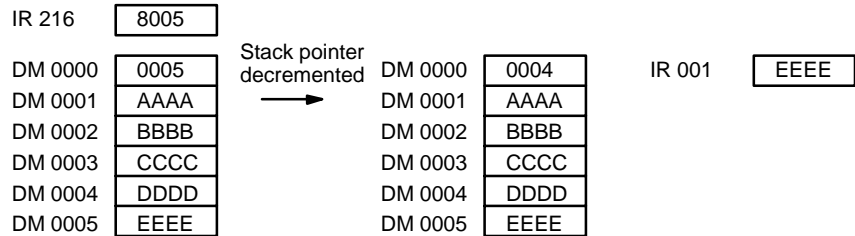
**Example**

The following example shows how to use COLL(81) to create a stack between DM 0001 and DM 0005. DM 0000 acts as the stack pointer.

When IR 00000 goes from OFF to ON, COLL(81) copies the content of DM 0005 (DM 0000 + 5) to IR 001. The content of the stack pointer (DM 0000) is then decremented by one.



Address	Instruction	Operands
00000	LD	00000
00001	@COLL(81)	
		DM 0000
		216
		001



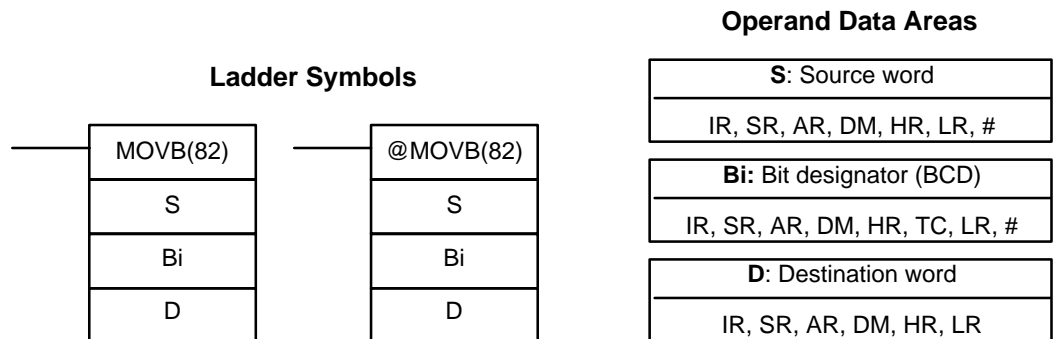
**Flags**

**ER:** The offset or stack length in the control word 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.)

During stack operation, the value of the stack pointer exceeds the length of the stack; an attempt was made to write to a word beyond the end of the stack.

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

**5-17-8 MOVE BIT – MOVB(82)**



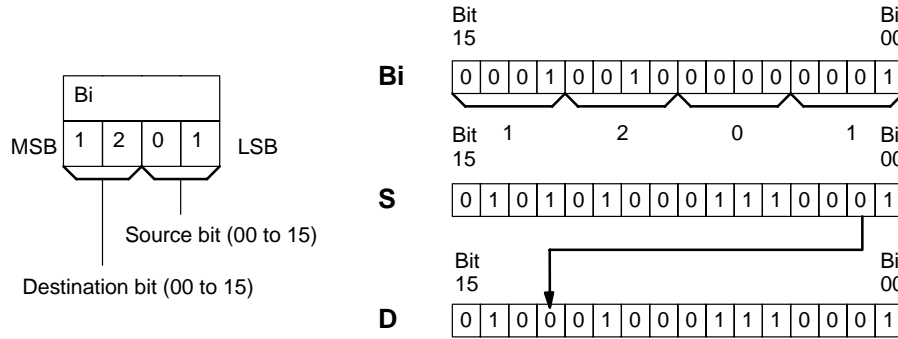
**Limitations**

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

DM 6144 to DM 6655 cannot be used for Bi or D.

**Description**

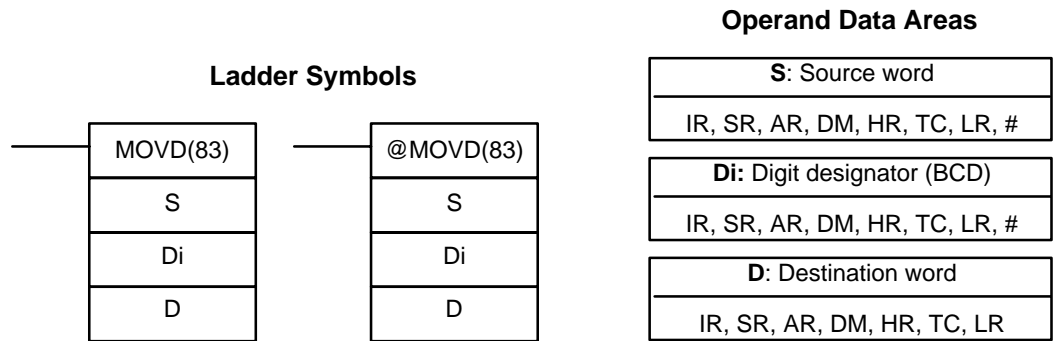
When the execution condition is OFF, MOV<sub>B</sub>(82) is not executed. When the execution condition is ON, MOV<sub>B</sub>(82) copies the specified bit of S to the specified bit in D. The bits in S and D are specified by Bi. The rightmost two digits of Bi designate the source bit; the leftmost two 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.)

**5-17-9 MOVE DIGIT – MOV<sub>D</sub>(83)**

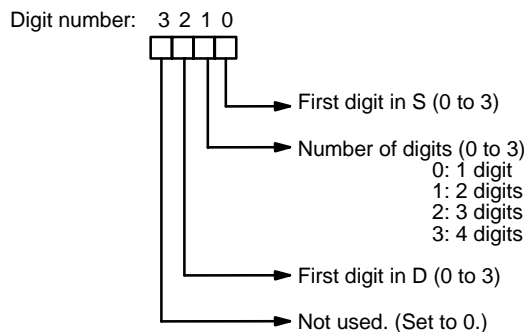


**Limitations**

The rightmost three digits of Di must each be between 0 and 3.  
DM 6144 to DM 6655 cannot be used for Di or D.

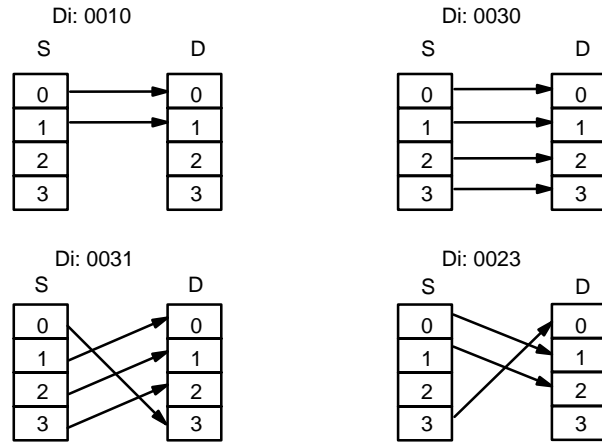
**Description**

When the execution condition is OFF, MOV<sub>D</sub>(83) is not executed. When the execution condition is ON, MOV<sub>D</sub>(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

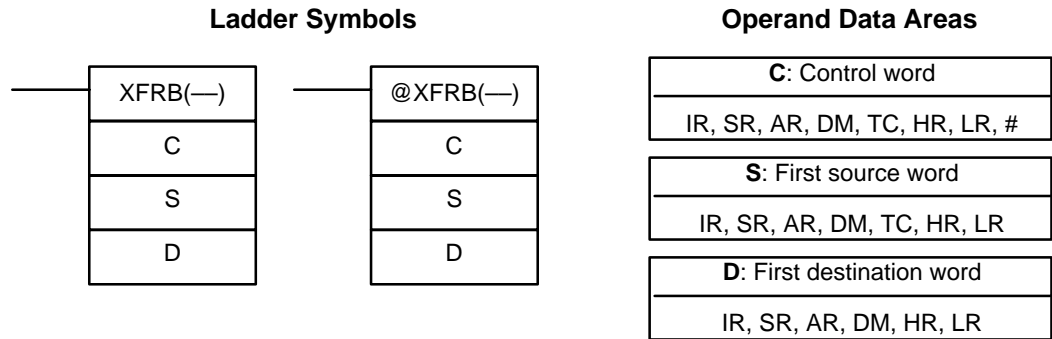
The following show examples of the data movements for various values of Di.



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.)

5-17-10 TRANSFER BITS – XFRB(—)

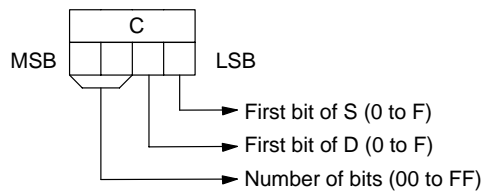


Limitations

This instruction is available in the **CQM1-CPU4□-E/-EV1 only**.  
The specified source bits must be in the same data area.  
The specified destination bits must be in the same data area.  
DM 6144 to DM 6655 cannot be used for D.

Description

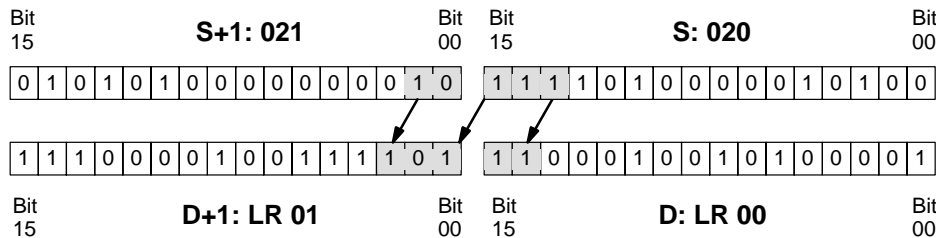
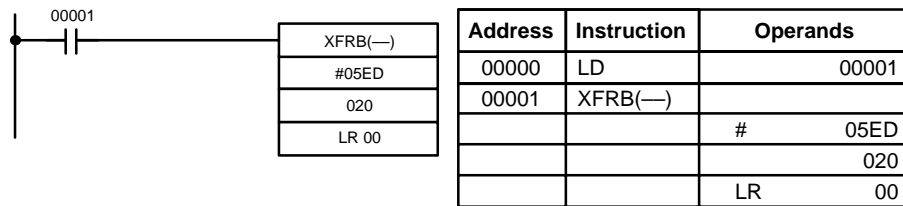
When the execution condition is OFF, XFRB(—) is not executed. When the execution condition is ON, XFRB(—) copies the specified source bits to the specified destination bits. The two rightmost digits of C specify the starting bits in S and D and the leftmost two digits indicate the number of bits that will be copied.



**Note** Up to 255 (FF) bits can be copied at one time.

**Example**

In the following example, XFRB(—) is used to transfer 5 bits from IR 020 and IR 021 to LR 00 and LR 01. The starting bit in IR 020 is D (13), and the starting bit in LR 00 is E (14), so IR 02013 to IR 02101 are copied to LR 0014 to LR 0102.



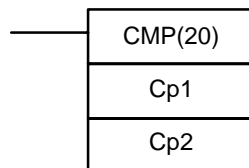
**Flags**

**ER:** The specified source bits are not all in the same data area.  
 The specified destination bits are not all in the same data area.  
 Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)

## 5-18 Comparison Instructions

### 5-18-1 COMPARE – CMP(20)

**Ladder Symbols**



**Operand Data Areas**

<b>Cp1:</b> First compare word
IR, SR, AR, DM, HR, TC, LR, #
<b>Cp2:</b> 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.

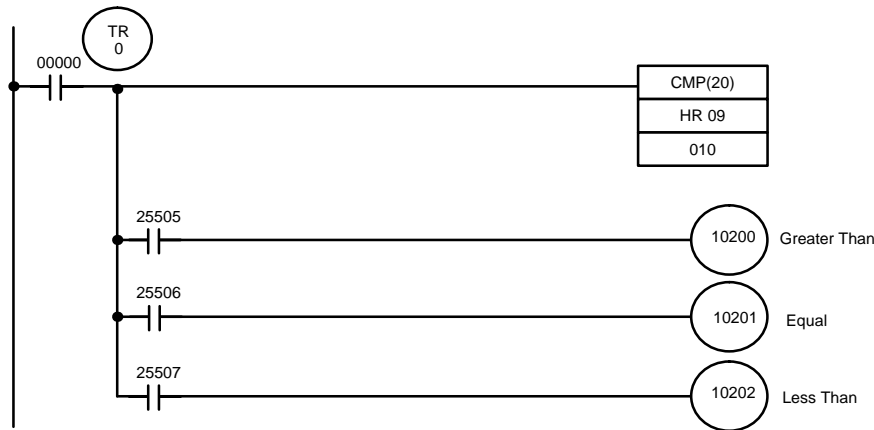
**Flags**

**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.

Flag	Address	C1 < C2	C1 = C2	C1 > C2
GR	25505	OFF	OFF	ON
EQ	25506	OFF	ON	OFF
LE	25507	ON	OFF	OFF

**Example:  
Saving CMP(20) Results**

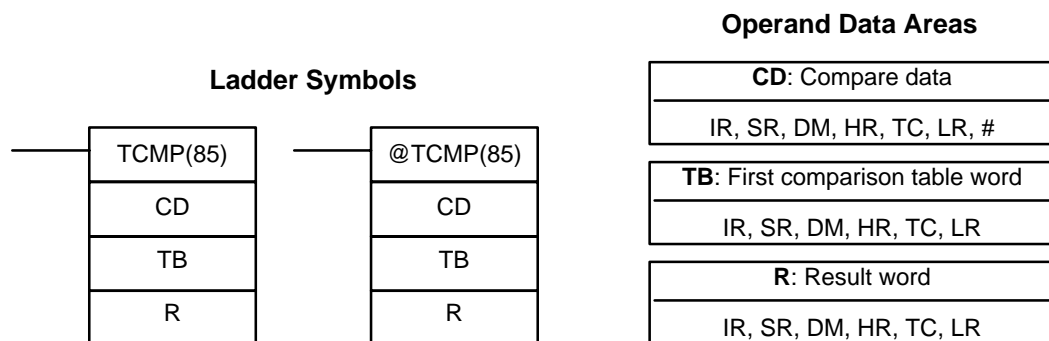
The following example shows how to save the comparison result immediately. If the content of HR 09 is greater than that of 010, 10200 is turned ON; if the contents are equal, 10201 is turned ON; if content of HR 09 is less than that of 010, 10202 is turned ON. In some applications, only one of the three OUTs would be necessary, making the use of TR 0 unnecessary. With this type of programming, 10200, 10201, and 10202 are changed only when CMP(20) is executed.



Address	Instruction	Operands
00000	LD	00000
00001	OUT	TR 0
00002	CMP(20)	
		HR 09
		010
00003	AND	25505
00004	OUT	10200

Address	Instruction	Operands
00005	LD	TR 0
00006	AND	25506
00007	OUT	10201
00008	LD	TR 0
00009	AND	25507
00010	OUT	10202

**5-18-2 TABLE COMPARE – TCMP(85)**



**Limitations**

DM 6144 to DM 6655 cannot be used for R.

**Description**

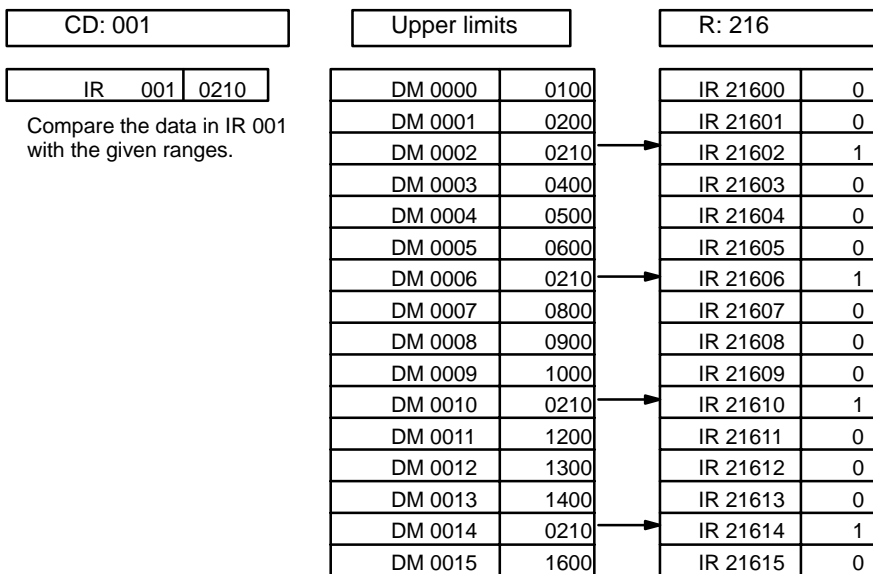
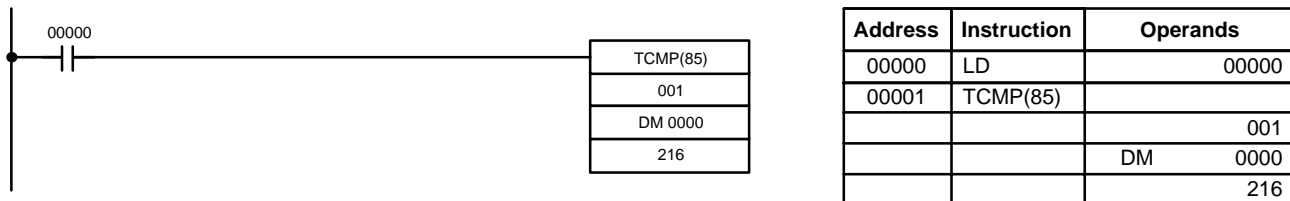
When the execution condition is OFF, TCMP(85) is not executed. When the execution condition is ON, TCMP(85) compares CD to the content of TB, TB+1, TB+2, ..., and TB+15. If CD is equal to the content of any of these words, the corresponding bit in R is set, e.g., if the CD equals the content of TB, bit 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**

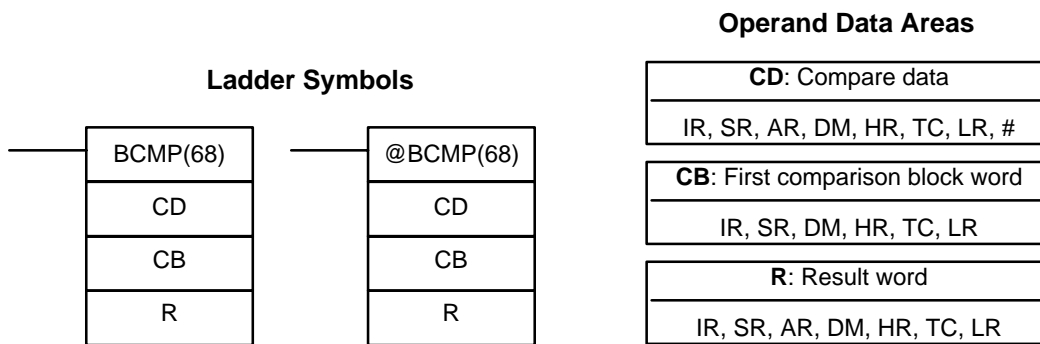
**ER:** The comparison table (i.e., TB through TB+15) exceeds the data area. Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)

**Example**

The following example shows the comparisons made and the results provided for TCMP(85). Here, the comparison is made during each cycle when IR 00000 is ON.



**5-18-3 BLOCK COMPARE – BCMP(68)**



**Note** BCMP(68) is an expansion instruction for the SRM1. The function code 68 is the factory setting and can be changed for the SRM1 if desired.

**Limitations**

Each lower limit word in the comparison block must be less than or equal to the upper limit.

DM 6144 to DM 6655 cannot be used for R.

**Description**

When the execution condition is OFF, BCMP(68) is not executed. When the execution condition is ON, BCMP(68) compares CD to the ranges defined by a block consisting of CB, CB+1, CB+2, ..., CB+31. Each range is defined by two words, the first one providing the lower limit and the second word providing the upper limit. If CD is found to be within any of these ranges (inclusive of the upper and lower limits), the corresponding bit in R is set. The comparisons that are

made and the corresponding bit in R that is set for each true comparison are shown below. The rest of the bits in R will be turned OFF.

CB ≤ CD ≤ CB+1	Bit 00
CB+2 ≤ CD ≤ CB+3	Bit 01
CB+4 ≤ CD ≤ CB+5	Bit 02
CB+6 ≤ CD ≤ CB+7	Bit 03
CB+8 ≤ CD ≤ CB+9	Bit 04
CB+10 ≤ CD ≤ CB+11	Bit 05
CB+12 ≤ CD ≤ CB+13	Bit 06
CB+14 ≤ CD ≤ CB+15	Bit 07
CB+16 ≤ CD ≤ CB+17	Bit 08
CB+18 ≤ CD ≤ CB+19	Bit 09
CB+20 ≤ CD ≤ CB+21	Bit 10
CB+22 ≤ CD ≤ CB+23	Bit 11
CB+24 ≤ CD ≤ CB+25	Bit 12
CB+26 ≤ CD ≤ CB+27	Bit 13
CB+28 ≤ CD ≤ CB+29	Bit 14
CB+30 ≤ CD ≤ CB+31	Bit 15

**Flags**

**ER:** The comparison block (i.e., CB through CB+31) exceeds the data area. Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)

**Example**

The following example shows the comparisons made and the results provided for BCMP(68). Here, the comparison is made during each cycle when IR 00000 is ON.



Address	Instruction	Operands
00000	LD	00000
00001	BCMP(68)	
		001
		DM 0010
		LR 05

CD 001	
--------	--

001	0210
-----	------

Compare data in IR 001 (which contains 0210) with the given ranges.

Lower limits	
--------------	--

DM 0010	0000
DM 0012	0101
DM 0014	0201
DM 0016	0301
DM 0018	0401
DM 0020	0501
DM 0022	0601
DM 0024	0701
DM 0026	0801
DM 0028	0901
DM 0030	1001
DM 0032	1101
DM 0034	1201
DM 0036	1301
DM 0038	1401
DM 0040	1501

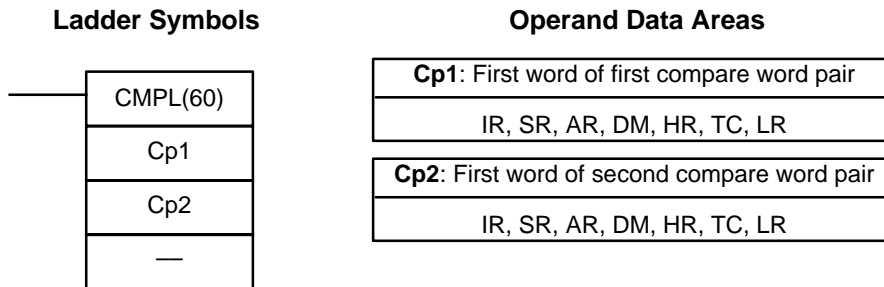
Upper limits	
--------------	--

DM 0011	0100
DM 0013	0200
DM 0015	0300
DM 0017	0400
DM 0019	0500
DM 0021	0600
DM 0023	0700
DM 0025	0800
DM 0027	0900
DM 0029	1000
DM 0031	1100
DM 0033	1200
DM 0035	1300
DM 0037	1400
DM 0039	1500
DM 0041	1600

R:LR 05	
---------	--

LR 0500	0
LR 0501	0
LR 0502	1
LR 0503	0
LR 0504	0
LR 0505	0
LR 0506	0
LR 0507	0
LR 0508	0
LR 0509	0
LR 0510	0
LR 0511	0
LR 0512	0
LR 0513	0
LR 0514	0
LR 0515	0

### 5-18-4 DOUBLE COMPARE – CMPL(60)



**Note** CMPL(60) is an expansion instruction for the SRM1. The function code 60 is the factory setting and can be changed for the SRM1 if desired.

**Limitations**

Cp1 and Cp1+1 must be in the same data area.  
 Cp2 and Cp2+1 must be in the same data area.  
 Set the third operand to 000.

**Description**

When the execution condition is OFF, CMPL(60) is not executed. When the execution condition is ON, CMPL(60) joins the 4-digit hexadecimal content of Cp1+1 with that of Cp1, and that of Cp2+1 with that of Cp2 to create two 8-digit hexadecimal numbers, Cp+1,Cp1 and Cp2+1,Cp2. The two 8-digit numbers are then compared and the result is output to the GR, EQ, and LE flags in the SR area.

**Precautions**

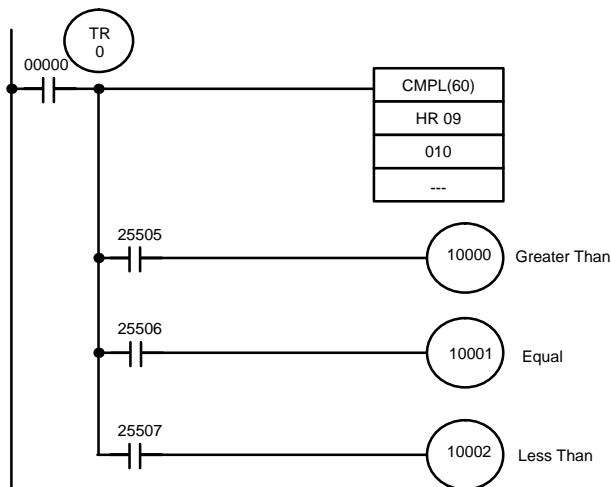
Placing other instructions between CMPL(60) and the operation which accesses the EQ, LE, and GR flags may change the status of these flags. Be sure to access them before the desired status is changed.

**Flags**

- ER:** Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)
- GR:** ON if Cp1+1,Cp1 is greater than Cp2+1,Cp2.
- EQ:** ON if Cp1+1,Cp1 equals Cp2+1,Cp2.
- LE:** ON if Cp1+1,Cp1 is less than Cp2+1,Cp2.

**Example:  
Saving CMPL(60) Results**

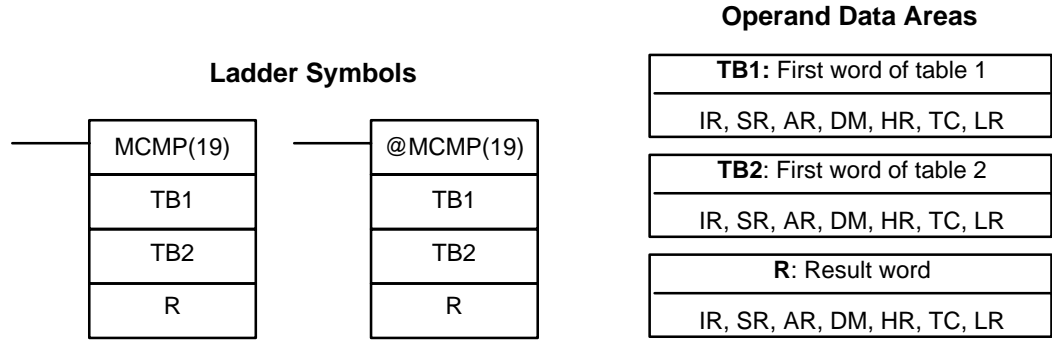
The following example shows how to save the comparison result immediately. If the content of HR 10, HR 09 is greater than that of 011, 010, then 10000 is turned ON; if the two contents are equal, 10001 is turned ON; if content of HR 10, HR 09 is less than that of 011, 010, then 10002 is turned ON. In some applications, only one of the three OUTs would be necessary, making the use of TR 0 unnecessary. With this type of programming, 10000, 10001, and 10002 are changed only when CMPL(60) is executed.



Address	Instruction	Operands
00000	LD	00000
00001	OUT	TR 0
00002	CMPL(60)	
		HR 09
		010
00003	AND	25505
00004	OUT	10000
00005	LD	TR 0
00006	AND	25506
00007	OUT	10001
00008	LD	TR 0
00009	AND	25507
00010	OUT	10002



### 5-18-5 MULTI-WORD COMPARE – MCMP(19)



**Limitations**

This instruction is available in the **CQM1 only**.  
 TB1 and TB1+15 must be in the same data area.  
 TB2 and TB2+15 must be in the same data area.  
 DM 6144 to DM 6655 cannot be used for R.

**Description**

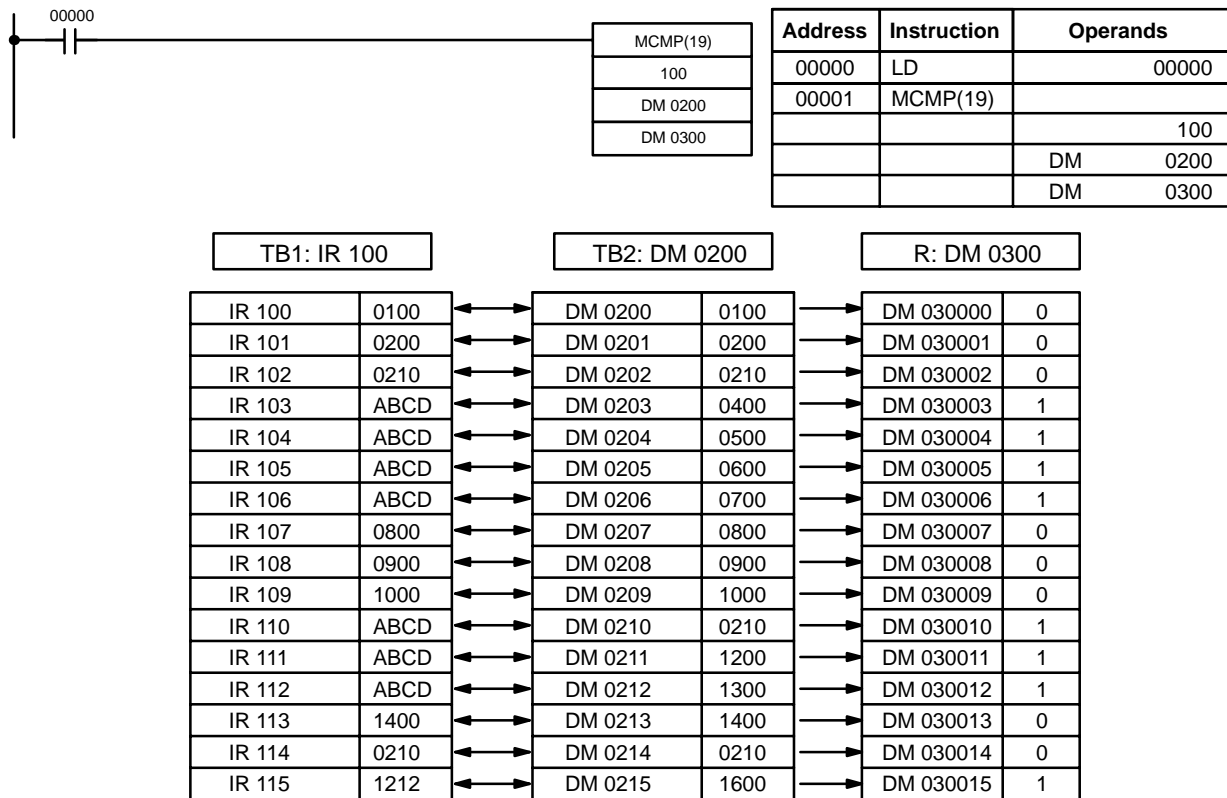
When the execution condition is OFF, MCMP(19) is not executed. When the execution condition is ON, MCMP(19) compares the content of TB1 to TB2, TB1+1 to TB2+1, TB1+2 to TB2+2, ..., and TB1+15 to TB2+15. If the first pair is equal, the first bit in R is turned OFF, etc., i.e., if the content of TB1 equals the content of TB2, bit 00 is turned OFF, if the content of TB1+1 equals the content of TB2+1, bit 01 is turned OFF, etc. The rest of the bits in R will be turned ON.

**Flags**

- ER:** One of the tables (i.e., TB1 through TB1+15, or TB2 through TB2+15) exceeds the data area.  
 Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)
- EQ:** ON if the entire contents of both tables are equal and R=0000.

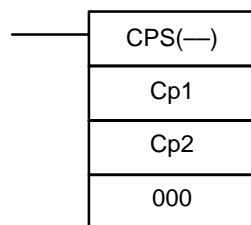
**Example**

The following example shows the comparisons made and the results provided for MCMP(19). Here, the comparison is made during each cycle when 00000 is ON.



**5-18-6 SIGNED BINARY COMPARE – CPS(—)**

**Ladder Symbols**



**Operand Data Areas**

<b>Cp1: First compare word</b>
IR, SR, AR, DM, HR, TC, LR, #
<b>Cp2: Second compare word</b>
IR, SR, AR, DM, HR, TC, LR, #
<b>000</b>
Not used. Set to 000.

**Limitations**

This instruction is available in the **CQM1-CPU4□-E/-EV1 only**.

**Description**

When the execution condition is OFF, CPS(—) is not executed. When the execution condition is ON, CPS(—) compares the 16-bit (4-digit) signed binary contents in Cp1 and Cp2 and outputs the result to the GR, EQ, and LE flags in the SR area.

**Precautions**

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

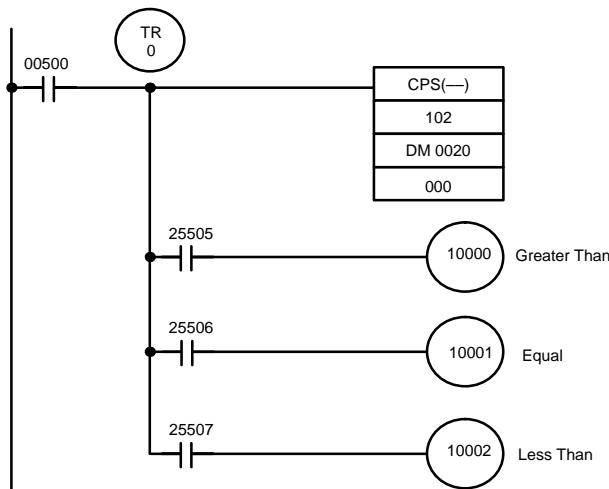
**Flags**

- ER:** Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)
- EQ:** ON if Cp1 equals Cp2.
- LE:** ON if Cp1 is less than Cp2.
- GR:** ON if Cp1 is greater than Cp2.

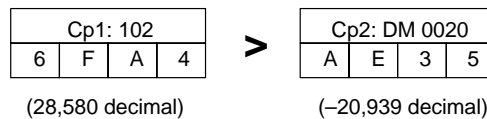
Comparison result	Flag status		
	GR (SR 25505)	EQ (SR 25506)	LE (SR 25507)
Cp1 < Cp2	0	0	1
Cp1 = Cp2	0	1	0
Cp1 > Cp2	1	0	0

**Example**

In the following example, the content of 102 is greater than that of DM 0020, so 10000 is turned ON and the other bits, 10001 and 10002, are turned OFF.

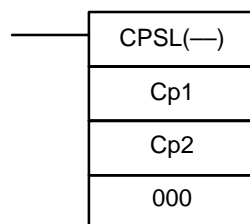


Address	Instruction	Operands
00000	LD	00500
00001	OUT	TR 0
00002	CPS(—)	
		102
		DM 0020
		000
00003	AND	25505
00004	OUT	10000
00005	LD	TR 0
00006	AND	25506
00007	OUT	10001
00008	LD	TR 0
00009	AND	25507
00010	OUT	10002



**5-18-7 DOUBLE SIGNED BINARY COMPARE – CPSL(—)**

**Ladder Symbols**



**Operand Data Areas**

<b>Cp1:</b> First compare word
IR, SR, AR, DM, HR, TC, LR, #
<b>Cp2:</b> Second compare word
IR, SR, AR, DM, HR, TC, LR, #
<b>000</b>
Not used. Set to 000.

**Limitations**

This instruction is available in the **CQM1-CPU4□-E/-EV1 only**.

**Description**

When the execution condition is OFF, CPSL(—) is not executed. When the execution condition is ON, CPSL(—) compares the 32-bit (8-digit) signed binary contents in Cp1+1, Cp1 and Cp2+1, Cp2 and outputs the result to the GR, EQ, and LE flags in the SR area.

**Precautions**

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

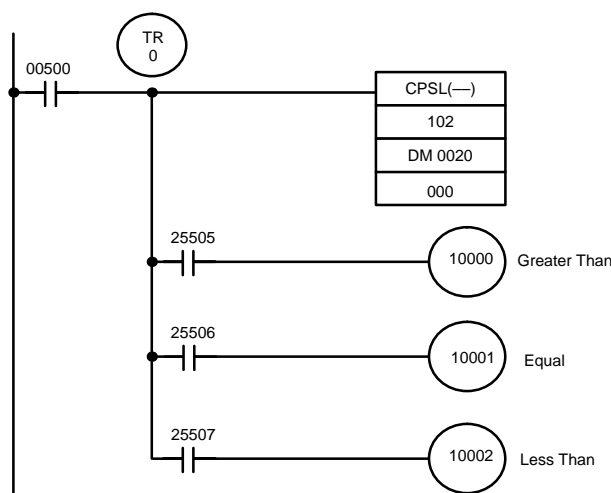
**Flags**

- ER:** Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)
- EQ:** ON if Cp1+1, Cp1 equals Cp2+1, Cp2.
- LE:** ON if Cp1+1, Cp1 is less than Cp2+1, Cp2.
- GR:** ON if Cp1+1, Cp1 is greater than Cp2+1, Cp2.

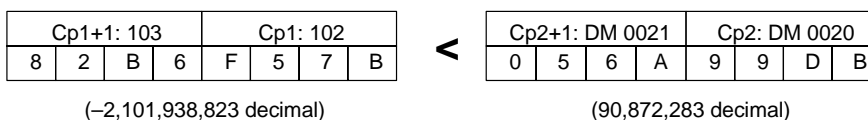
Comparison result	Flag status		
	GR (SR 25505)	EQ (SR 25506)	LE (SR 25507)
Cp1+1, Cp1 < Cp2+1, Cp2	0	0	1
Cp1+1, Cp1 = Cp2+1, Cp2	0	1	0
Cp1+1, Cp1 > Cp2+1, Cp2	1	0	0

**Example**

In the following example, the content of 103, 102 is less than that of DM 0021, DM 0020, so 10002 is turned ON and the other bits, 10000 and 10001, are turned OFF.

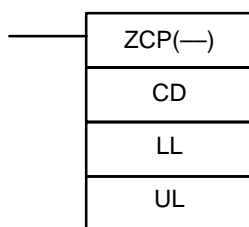


Address	Instruction	Operands
00000	LD	00500
00001	OUT	TR 0
00002	CPSL(—)	
		102
		DM 0020
		000
00003	AND	25505
00004	OUT	10000
00005	LD	TR 0
00006	AND	25506
00007	OUT	10001
00008	LD	TR 0
00009	AND	25507
00010	OUT	10002



**5-18-8 AREA RANGE COMPARE – ZCP(—)**

**Ladder Symbol**



**Operand Data Areas**

<b>CD:</b> Compare data
IR, SR, AR, DM, HR, TC, LR, #
<b>LL:</b> Lower limit of range
IR, SR, AR, DM, HR, TC, LR, #
<b>UL:</b> Upper limit of range
IR, SR, AR, DM, HR, TC, LR, #

**Limitations**

This instruction is available in the **CQM1-CPU4□-E/-EV1** only. LL must be less than or equal to UL.

**Description**

When the execution condition is OFF, ZCP(—) is not executed. When the execution condition is ON, ZCP(—) compares CD to the range defined by lower limit LL and upper limit UL and outputs the result to the GR, EQ, and LE flags in the SR area. The resulting flag status is shown in the following table.

Comparison result	Flag status		
	GR (SR 25505)	EQ (SR 25506)	LE (SR 25507)
CD < LL	0	0	1
LL ≤ CD ≤ UL	0	1	0
UL < CD	1	0	0

**Precautions**

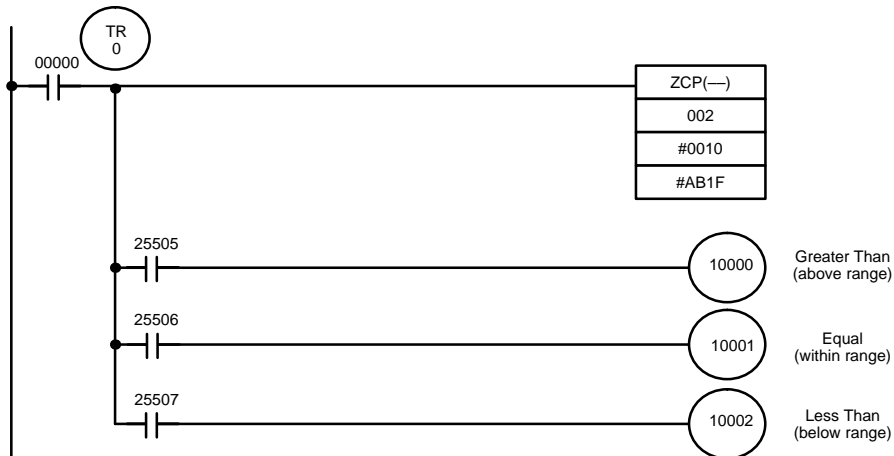
Placing other instructions between ZCP(—) and the operation which accesses the EQ, LE, and GR flags may change the status of these flags. Be sure to access them before the desired status is changed.

**Flags**

- ER:** Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)  
LL is greater than UL.
- EQ:** ON if LL ≤ CD ≤ UL
- LE:** ON if CD < LL.
- GR:** ON if CD > UL.

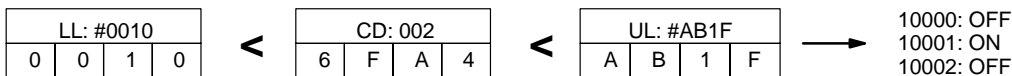
**Example**

In the following example, the content of IR 002 (#6FA4) is compared to the range #0010 to #AB1F. Since #0010 ≤ #6FA4 ≤ #AB1F, the EQ flag and IR 10001 are turned ON.

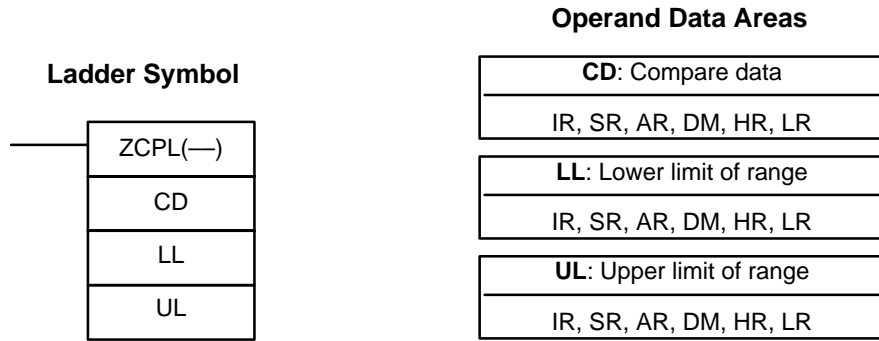


Address	Instruction	Operands
00000	LD	00000
00001	OUT	TR 0
00002	ZCP(—)	
		002
		# 0010
		# AB1F
00003	AND	25505

Address	Instruction	Operands
00004	OUT	10000
00005	LD	TR 0
00006	AND	25506
00007	OUT	10001
00008	LD	TR 0
00009	AND	25507
00010	OUT	10002



### 5-18-9 DOUBLE AREA RANGE COMPARE – ZCPL(—)



**Limitations**

This instruction is available in the **CQM1-CPU4□-E/-EV1 only**.

The 8-digit value in LL+1,LL must be less than or equal to UL+1,UL.

**Description**

When the execution condition is OFF, ZCPL(—) is not executed. When the execution condition is ON, ZCPL(—) compares the 8-digit value in CD, CD+1 to the range defined by lower limit LL+1,LL and upper limit UL+1,UL and outputs the result to the GR, EQ, and LE flags in the SR area. The resulting flag status is shown in the following table.

Comparison result	Flag status		
	GR (SR 25505)	EQ (SR 25506)	LE (SR 25507)
CD, CD+1 < LL+1,LL	0	0	1
LL+1,LL ≤ CD, CD+1 ≤ UL+1,UL	0	1	0
UL+1,UL < CD, CD+1	1	0	0

**Precautions**

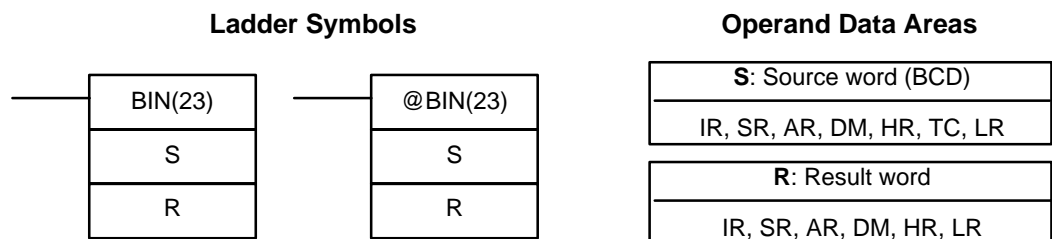
Placing other instructions between ZCPL(—) and the operation which accesses the EQ, LE, and GR flags may change the status of these flags. Be sure to access them before the desired status is changed.

**Flags**

- ER:** Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)  
LL+1,LL is greater than UL+1,UL.
- EQ:** ON if LL+1,LL ≤ CD, CD+1 ≤ UL+1,UL
- LE:** ON if CD, CD+1 < LL+1,LL.
- GR:** ON if CD, CD+1 > UL+1,UL.

## 5-19 Conversion Instructions

### 5-19-1 BCD-TO-BINARY – BIN(23)

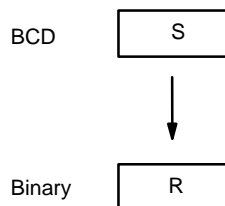


**Limitations**

DM 6144 to DM 6655 cannot be used for R.

**Description**

When the execution condition is OFF, BIN(23) is not executed. When the execution condition is ON, BIN(23) converts the BCD content of S into the numerically equivalent binary bits, and outputs the binary value to R. Only the content of R is changed; the content of S is left unchanged.

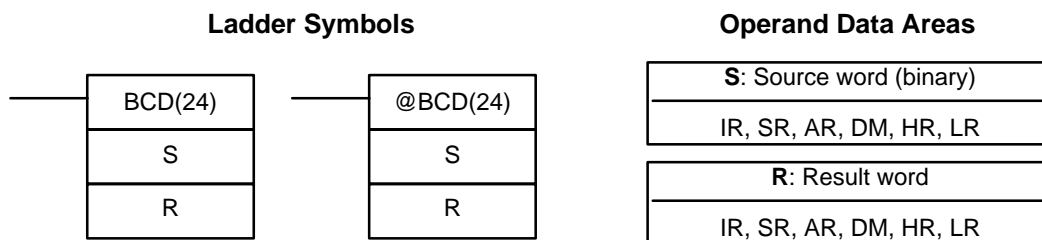


BIN(23) can be used to convert BCD to binary so that displays on the Programming Console or any other programming device will appear in hexadecimal rather than decimal. It can also be used to convert to binary to perform binary arithmetic operations rather than BCD arithmetic operations, e.g., when BCD and binary values must be added.

**Flags**

- ER:** The content of S is not BCD.  
Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)
- EQ:** ON when the result is zero.

**5-19-2 BINARY-TO-BCD – BCD(24)**

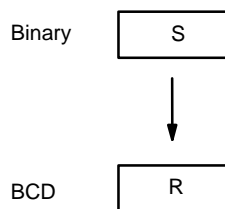


**Limitations**

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.  
DM 6144 to DM 6655 cannot be used for R.

**Description**

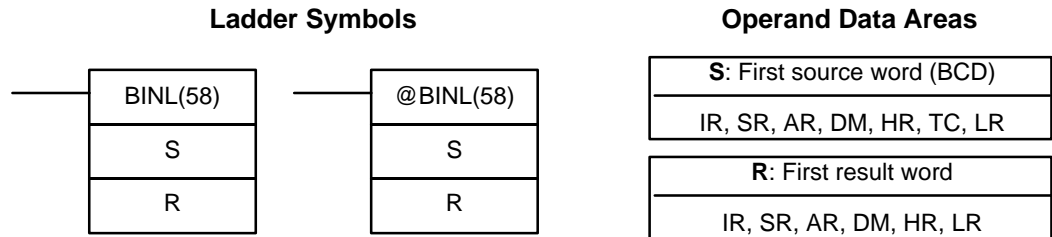
BCD(24) converts the binary (hexadecimal) content of S into the numerically equivalent BCD bits, and outputs the BCD bits to R. Only the content of R is changed; the content of S is left unchanged.



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.

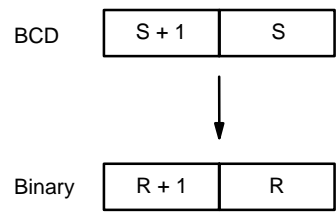
- Flags**
- ER:** Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)
  - EQ:** ON when the result is zero.

### 5-19-3 DOUBLE BCD-TO-DOUBLE BINARY – BINL(58)



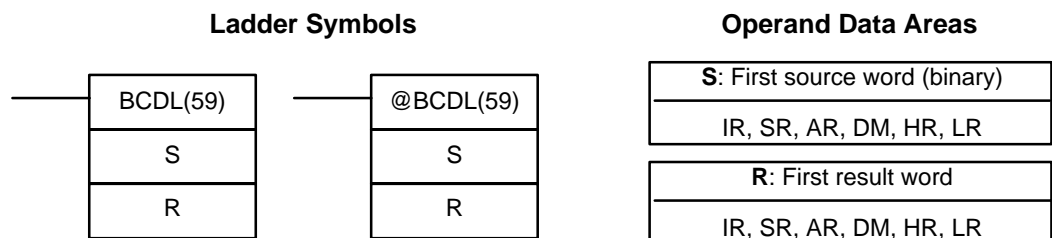
**Limitations** This instruction is available in the **CQM1 only**.  
DM 6143 to DM 6655 cannot be used for R.

**Description** When the execution condition is OFF, BINL(58) is not executed. When the execution condition is ON, BINL(58) converts an eight-digit number in S and S+1 into 32-bit binary data, and outputs the converted data to R and R+1.



- Flags**
- ER:** The contents of S and/or S+1 words are not BCD.  
Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)
  - EQ:** ON when the result is zero.

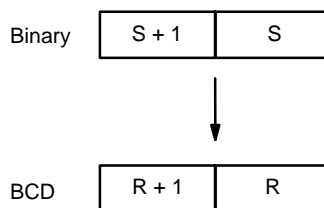
### 5-19-4 DOUBLE BINARY-TO-DOUBLE BCD – BCDL(59)



**Limitations** This instruction is available in the **CQM1 only**.  
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.  
DM 6143 to DM 6655 cannot be used for R.



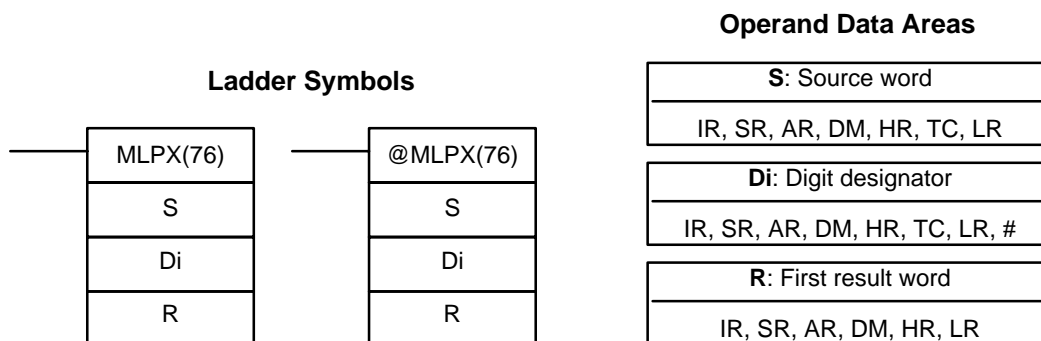
**Description** BCDL(59) converts the 32-bit binary content of S and S+1 into eight digits of BCD data, and outputs the converted data to R and R+1.



**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 the result is zero.

### 5-19-5 4-TO-16 DECODER – MLPX(76)

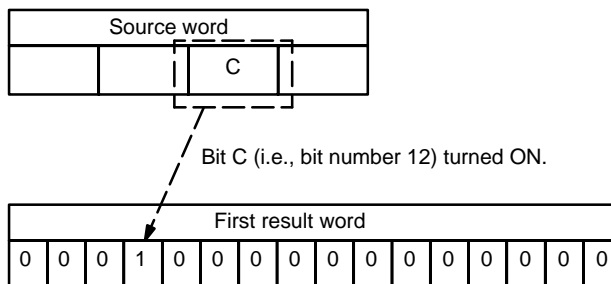


**Limitations**

- The rightmost two digits of Di must each be between 0 and 3.
- All result words must be in the same data area.
- DM 6144 to DM 6655 cannot be used for R.

**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.)

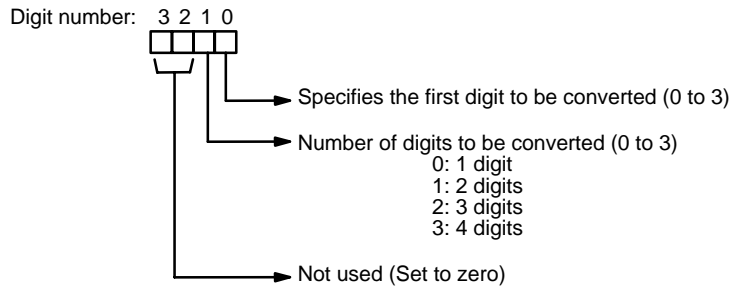
The following is an example of a one-digit decode operation from digit number 1 of S, i.e., here Di would be 0001.



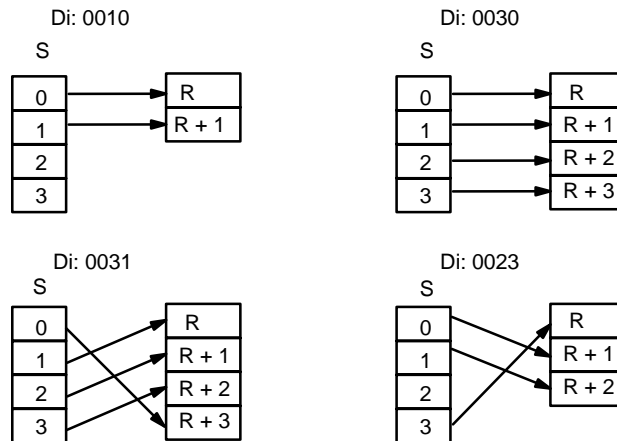
The first digit and the number of digits to be converted are designated in Di. If more digits are designated than remain in S (counting from the designated first digit), the remaining digits will be taken starting back at the beginning of S. The final word required to store the converted result (R plus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

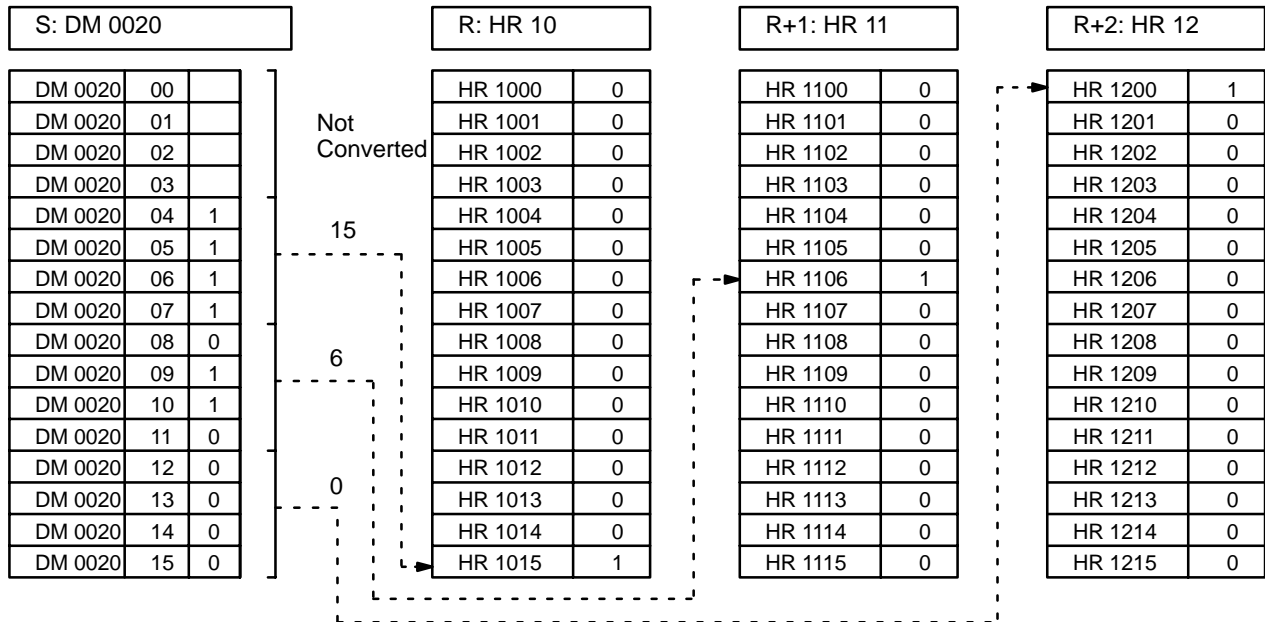
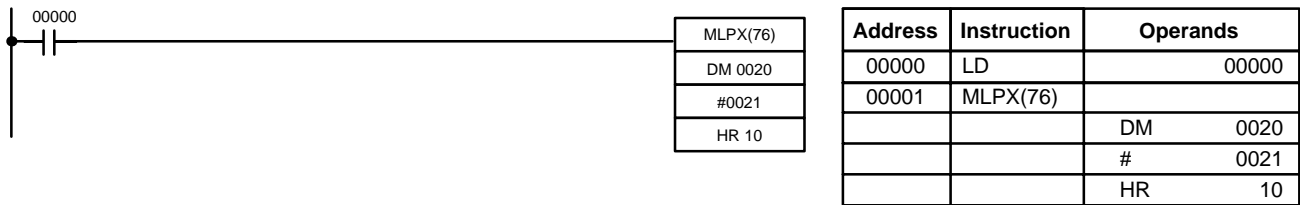


**Flags**

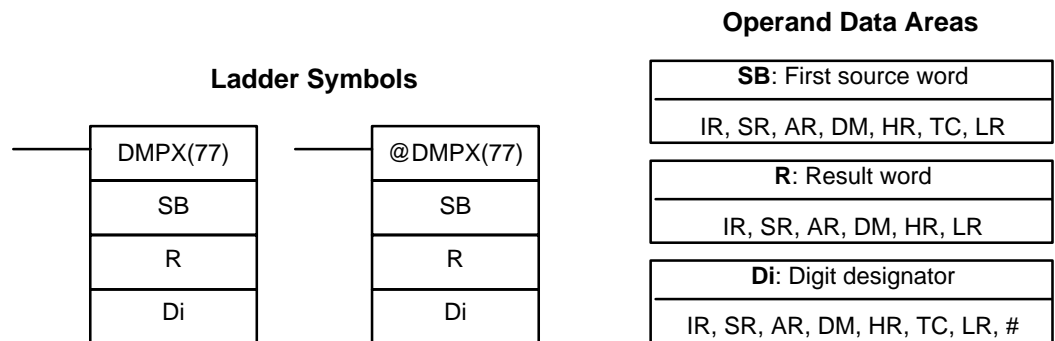
**ER:** Undefined digit designator, or R plus number of digits exceeds a data area.  
 Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD or the DM area boundary has been exceeded.)

**Example**

The following program converts digits 1 to 3 of data from DM 0020 to bit positions and turns ON the corresponding bits in three consecutive words starting with HR 10. Digit 0 is not converted.



**5-19-6 16-TO-4 ENCODER – DMPX(77)**



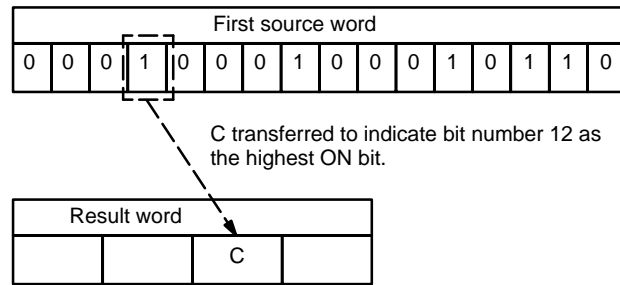
**Limitations**

The rightmost two digits of Di must each be between 0 and 3.  
 All source words must be in the same data area.  
 DM 6144 to DM 6655 cannot be used for SB, R, or Di.

**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.

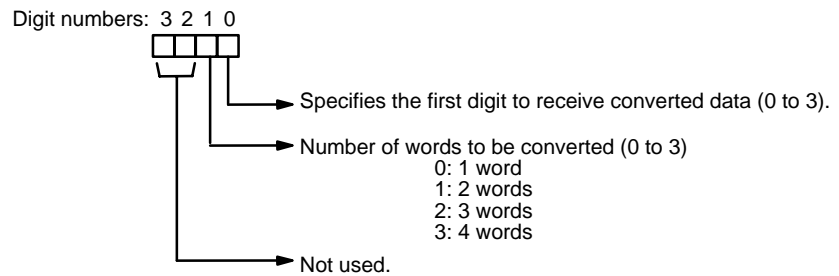
The following is an example of a one-digit encode operation to digit number 1 of R, i.e., here Di would be 0001.



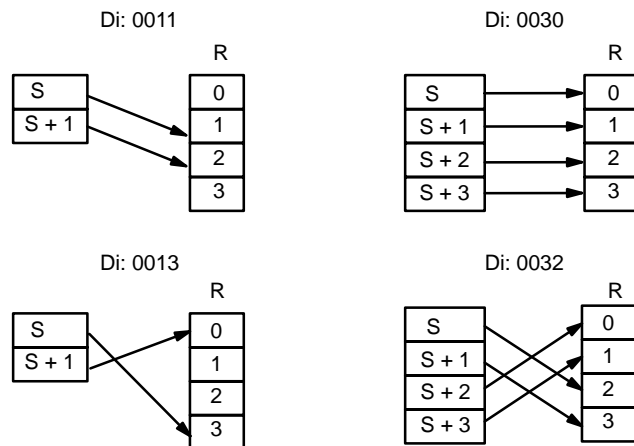
Up to four digits from four consecutive source words starting with S may be encoded and the digits written to R in order from the designated first digit. If more digits are designated than remain in R (counting from the designated first digit), the remaining digits will be placed at digits starting back 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.

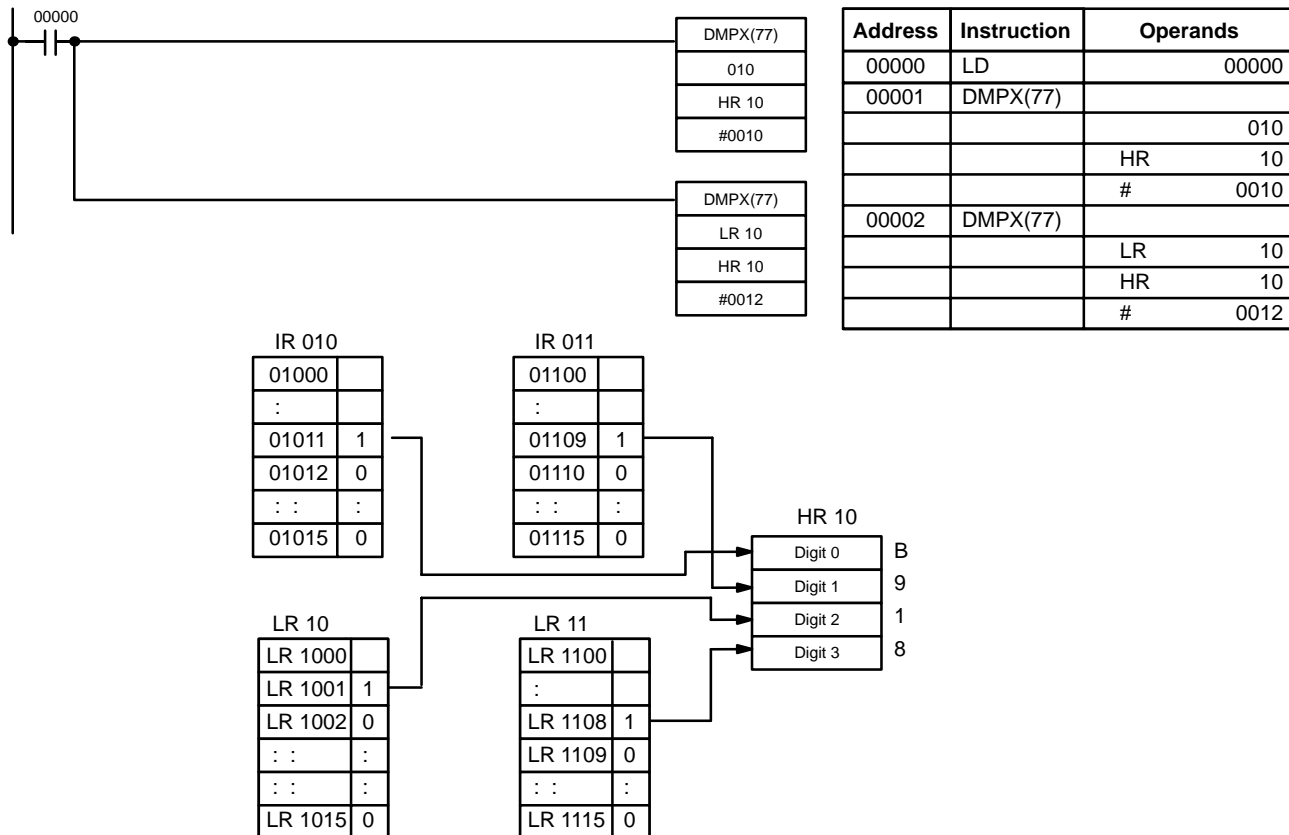


**Flags**

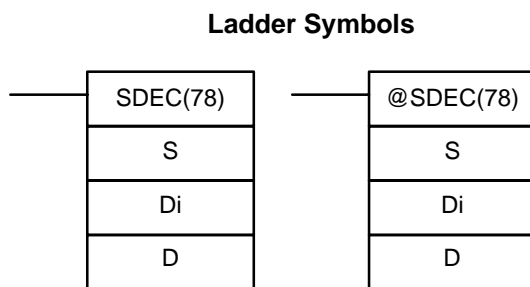
- ER:** Undefined digit designator, or S plus number of digits exceeds a data area.
- Content of a source word is zero.
- Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)

**Example**

When 00000 is ON, the following diagram encodes IR words 010 and 011 to the first two digits of HR 10 and then encodes LR 10 and 11 to the last two digits of HR 10. 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.



**5-19-7 7-SEGMENT DECODER – SDEC(78)**



**Operand Data Areas**

<b>S:</b> Source word (binary)
IR, SR, AR, DM, HR, TC, LR
<b>Di:</b> Digit designator
IR, SR, AR, DM, HR, TC, LR, #
<b>D:</b> First destination word
IR, SR, AR, DM, HR, LR

**Limitations**

- Di must be within the values given below.
- All destination words must be in the same data area.
- DM 6144 to DM 6655 cannot be used for D.

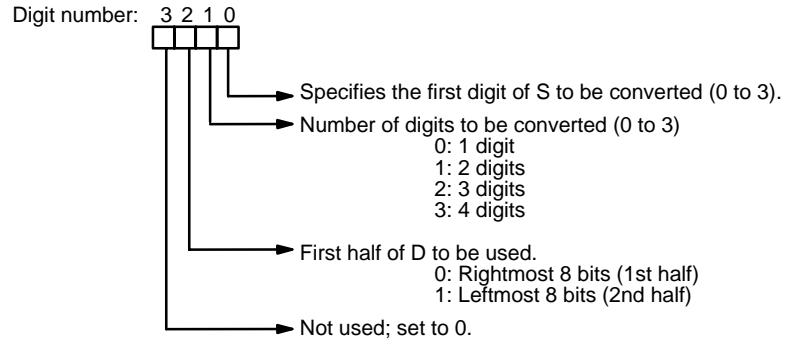
**Description**

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.

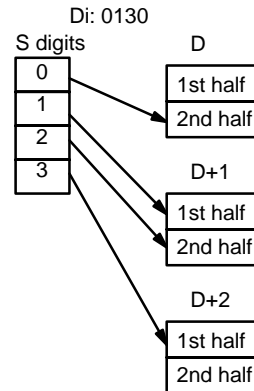
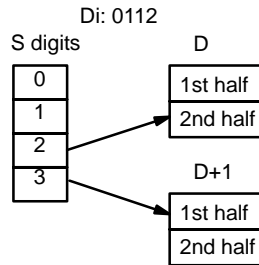
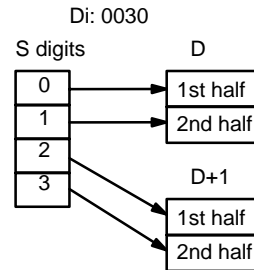
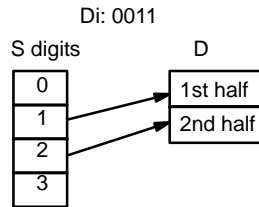
Any or all of the digits in S may be converted in sequence from the designated first digit. The first digit, the number of digits to be converted, and the half of D to receive the first 7-segment display code (rightmost or leftmost 8 bits) are designated in Di. If multiple digits are designated, they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

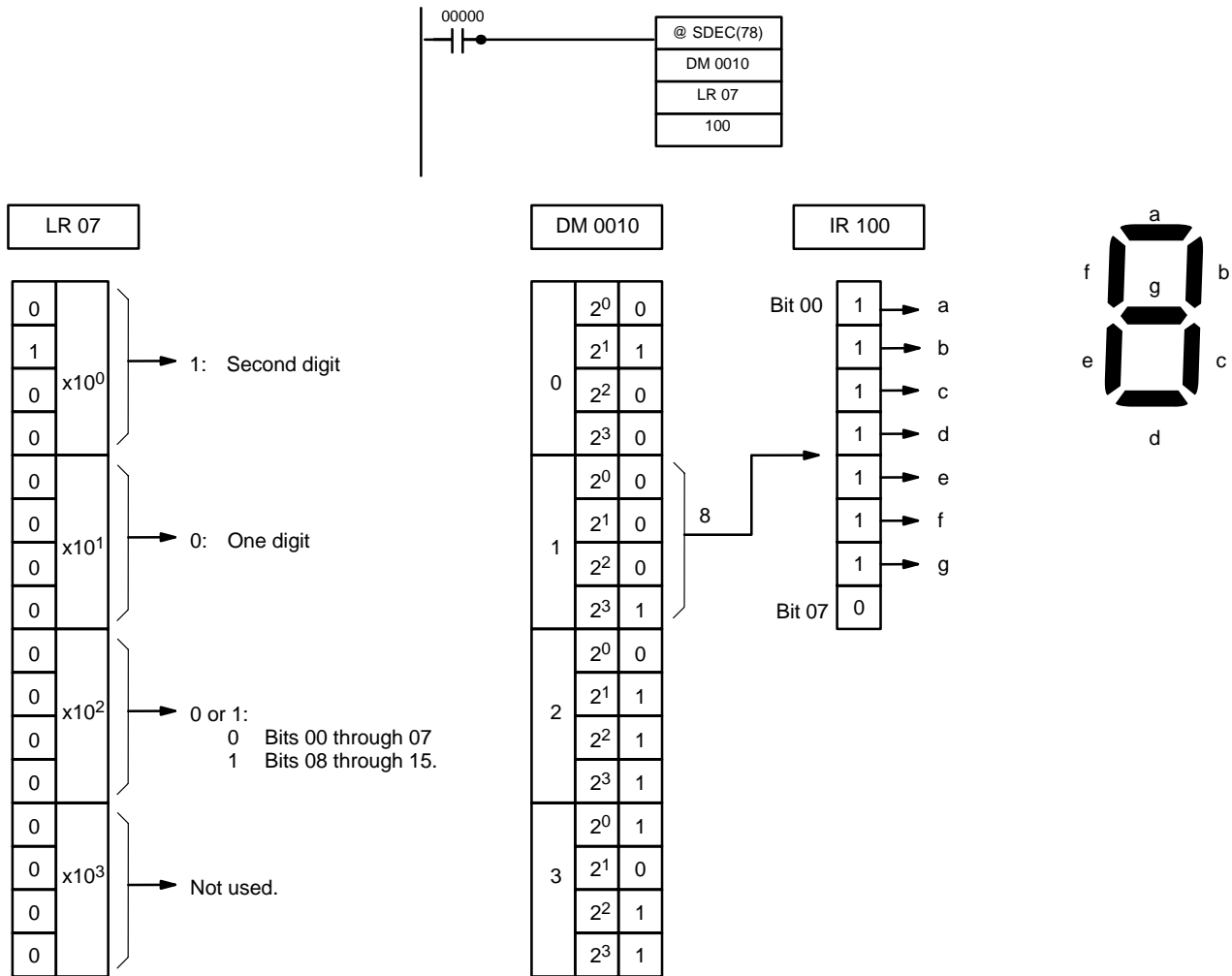


Some example Di values and the 4-bit binary to 7-segment display conversions that they produce are shown below.



**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.

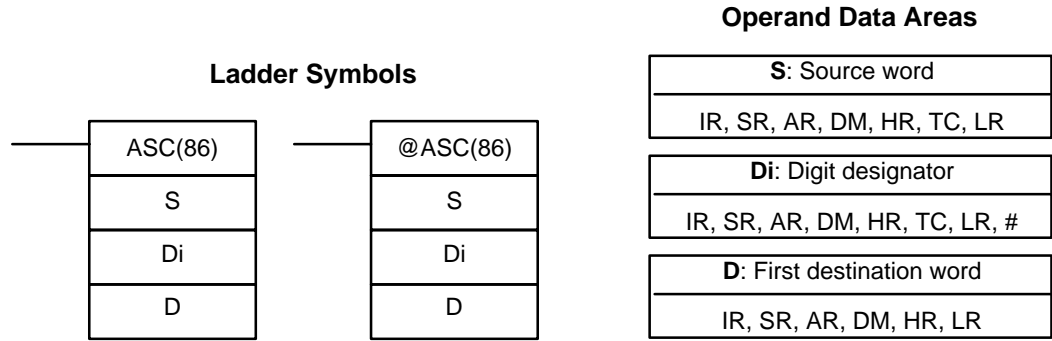


Original data					Converted code (segments)								Display
Digit	Bits				-	g	f	e	d	c	b	a	
0	0	0	0	0	0	0	1	1	0	0	0	0	0
1	0	0	0	1	0	0	1	1	0	0	0	0	1
2	0	0	1	0	0	0	1	1	0	0	1	1	0
3	0	0	1	1	0	0	1	1	0	0	1	1	0
4	0	1	0	0	0	0	1	1	0	1	0	0	0
5	0	1	0	1	0	0	1	1	0	1	0	1	0
6	0	1	1	0	0	0	1	1	0	1	0	1	0
7	0	1	1	1	0	0	1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

**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.)

**5-19-8 ASCII CONVERT – ASC(86)**



**Limitations**

Di must be within the values given below.  
All destination words must be in the same data area.  
DM 6144 to DM 6655 cannot be used for D.

**Description**

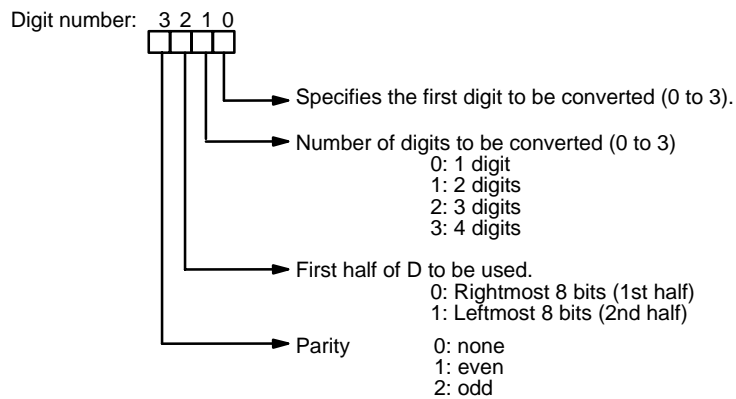
When the execution condition is OFF, ASC(86) is not executed. When the execution condition is ON, ASC(86) converts the designated digit(s) of S into the equivalent 8-bit ASCII code and places it into the destination word(s) beginning with D.

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.

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

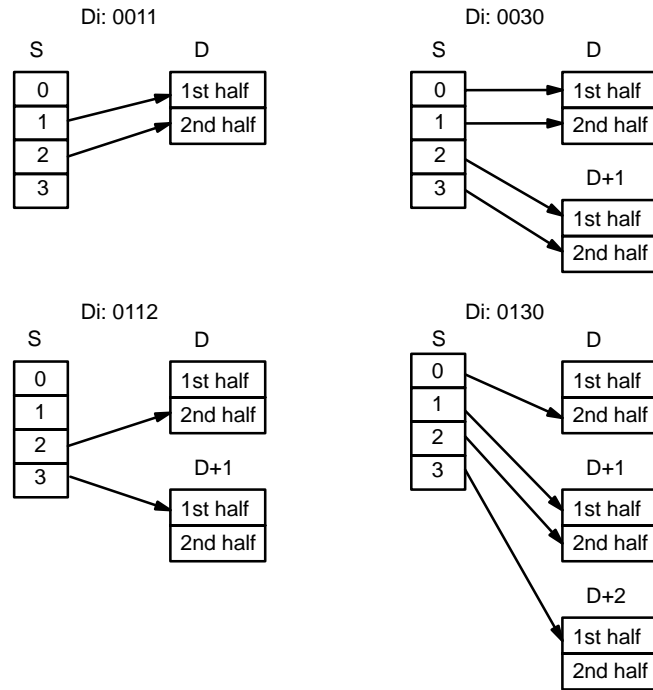
**Digit Designator**

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.

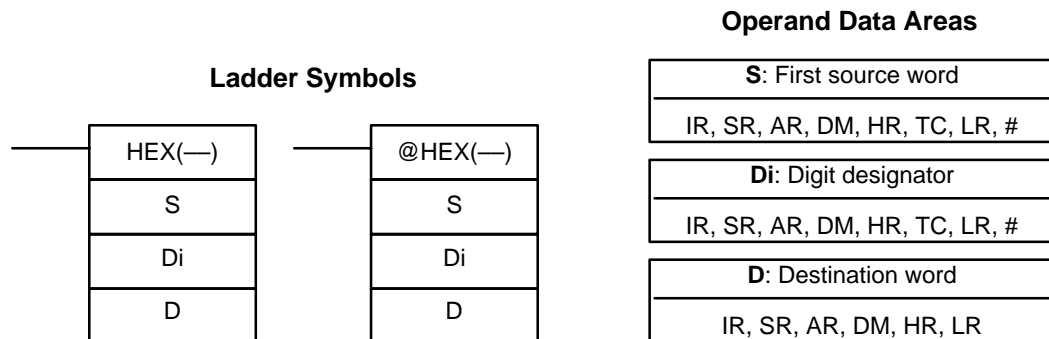
When even parity is designated, the leftmost bit will be adjusted so that the total number of ON bits is even, e.g., when adjusted for even parity, ASCII “31” (00110001) will be “B1” (10110001: parity bit turned ON to create an even number of ON bits); ASCII “36” (00110110) will be “36” (00110110: parity 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.)

**5-19-9 ASCII-TO-HEXADECIMAL – HEX(—)**



**Limitations**

This instruction is available in the **CQM1/SRM1** only.

Di must be within the values given below.

All source words must be in the same data area.

Bytes in the source words must contain the ASCII code equivalent of hexadecimal values, i.e., 30 to 39 (0 to 9) or 41 to 46 (A to F).

DM 6144 to DM 6655 cannot be used for D.

**Description**

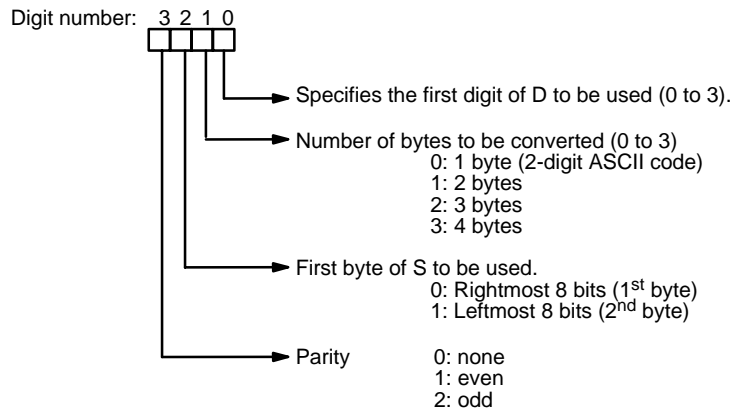
When the execution condition is OFF, HEX(—) is not executed. When the execution condition is ON, HEX(—) converts the designated byte(s) of ASCII code from the source word(s) into the hexadecimal equivalent and places it into D.

Up to 4 ASCII codes may be converted beginning with the designated first byte of S. The converted hexadecimal values are then placed in D in order from the designated digit. The first byte (rightmost or leftmost 8 bits), the number of bytes to be converted, and the digit of D to receive the first hexadecimal value are designated in Di. If multiple bytes are designated, they will be converted in order starting from the designated half of S and continuing to S+1 and S+2, if necessary.

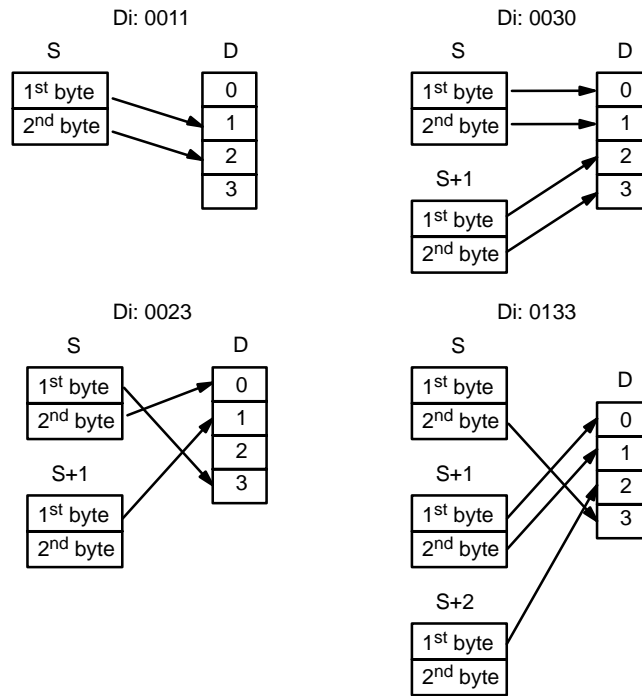
If more digits are designated than remain in D (counting from the designated first digit), further digits will be used starting back at the beginning of D. Digits in D that do not receive converted data will not be changed.

**Digit Designator**

The digits of Di are set as shown below.



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



**ASCII Code Table**

The following table shows the ASCII codes before conversion and the hexadecimal values after conversion. Refer to *Appendix H* for a table of ASCII characters.

ASCII Code	Original data								Converted data				
	Bit status (See note.)								Digit	Bits			
30	*	0	1	1	0	0	0	0	0	0	0	0	0
31	*	0	1	1	0	0	0	1	1	0	0	0	1
32	*	0	1	1	0	0	1	0	2	0	0	1	0
33	*	0	1	1	0	0	1	1	3	0	0	1	1
34	*	0	1	1	0	1	0	0	4	0	1	0	0
35	*	0	1	1	0	1	0	1	5	0	1	0	1
36	*	0	1	1	0	1	1	0	6	0	1	1	0
37	*	0	1	1	0	1	1	1	7	0	1	1	1
38	*	0	1	1	1	0	0	0	8	1	0	0	0
39	*	0	1	1	1	0	0	1	9	1	0	0	1
41	*	1	0	1	0	0	0	1	A	1	0	1	0
42	*	1	0	1	0	0	1	0	B	1	0	1	1
43	*	1	0	1	0	0	1	1	C	1	1	0	0
44	*	1	0	1	0	1	0	0	D	1	1	0	1
45	*	1	0	1	0	1	0	1	E	1	1	1	0
46	*	1	0	1	0	1	1	0	F	1	1	1	1

**Note** The leftmost bit of each ASCII code is adjusted for parity.

**Parity**

The leftmost bit of each ASCII character (2 digits) is automatically adjusted for either even or odd parity.

With no parity, the leftmost bit should always be zero. With odd or even parity, the leftmost bit of each ASCII character should be adjusted so that there is an odd or even number of ON bits.

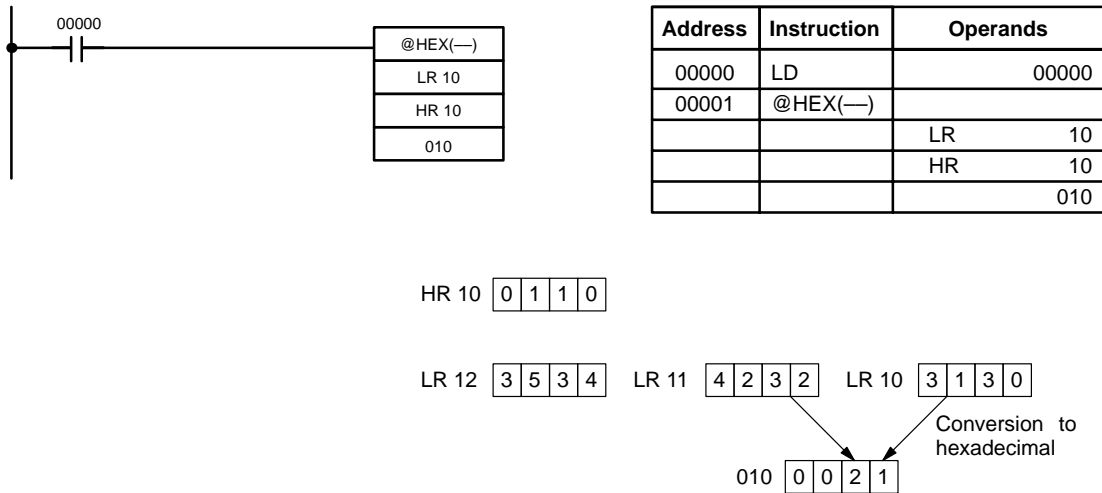
If the parity of the ASCII code in S does not agree with the parity specified in Di, the ER Flag (SR 25503) will be turned ON and the instruction will not be executed.

**Flags**

**ER:** Incorrect digit designator, or data area for destination exceeded.  
Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)

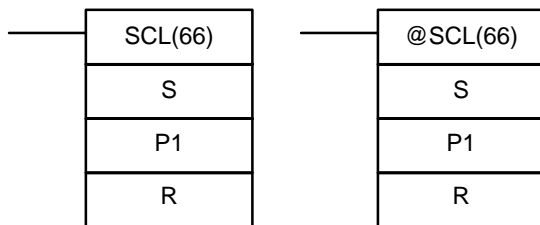
**Example**

In the following example, the 2<sup>nd</sup> byte of LR 10 and the 1<sup>st</sup> byte of LR 11 are converted to hexadecimal values and those values are written to the first and second bytes of IR 010.



**5-19-10 SCALING – SCL(66)**

**Ladder Symbols**



**Operand Data Areas**

<b>S:</b> Source word
IR, SR, AR, DM, HR, TC, LR, #
<b>P1:</b> First parameter word
IR, SR, AR, DM, HR, TC, LR
<b>R:</b> Result word
IR, SR, AR, DM, HR, LR

**Limitations**

This instruction is available in the **CQM1 only**.  
S must be BCD.  
P1 through P1+3 must be in the same data area.  
DM 6144 to DM 6655 cannot be used for P1 through P1+3 or R.

**Description**

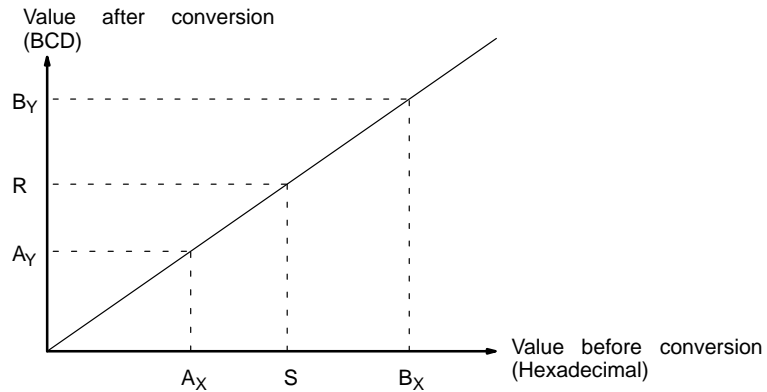
SCL(66) is used to linearly convert a 4-digit hexadecimal value to a 4-digit BCD value. Unlike BCD(24), which converts a 4-digit hexadecimal value to its 4-digit BCD equivalent ( $S_{hex} \rightarrow S_{BCD}$ ), SCL(66) can convert the hexadecimal value according to a specified linear relationship. The conversion line is defined by two points specified in the parameter words P1 to P1+3.

When the execution condition is OFF, SCL(66) is not executed. When the execution condition is ON, SCL(66) converts the 4-digit hexadecimal value in S to the 4-digit BCD value on the line defined by points (P1, P1+1) and (P1+2, P1+3) and places the results in R. The results is rounded off to the nearest integer. If the results is less than 0000, then 0000 is written to R, and if the result is greater than 9999, then 9999 is written to R.

The following table shows the functions and ranges of the parameter words:

Parameter	Function	Range	Comments
P1	BCD point #1 (A <sub>Y</sub> )	0000 to 9999	---
P1+1	Hex. point #1 (A <sub>X</sub> )	0000 to FFFF	Do not set P1+1=P1+3.
P1+2	BCD point #2 (B <sub>Y</sub> )	0000 to 9999	---
P1+3	Hex. point #2 (B <sub>X</sub> )	0000 to FFFF	Do not set P1+3=P1+1.

The following diagram shows the source word, S, converted to D according to the line defined by points (A<sub>Y</sub>, A<sub>X</sub>) and (B<sub>Y</sub>, B<sub>X</sub>).



The results can be calculated by first converting all values to BCD and then using the following formula.

$$\text{Results} = B_Y - [(B_Y - A_Y)/(B_X - A_X) \times (B_X - S)]$$

**Flags**

- ER:** The value in P1+1 equals that in P1+3.  
Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)  
  
P1 and P1+3 are not in the same data area, or other setting error.
- EQ:** ON when the result, R, is 0000.

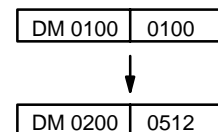
**Example**

When 00000 is turned ON in the following example, the BCD source data in DM 0100 (#0100) is converted to hexadecimal according to the parameters in DM 0150 to DM 0153. The result (#0512) is then written to DM 0200.



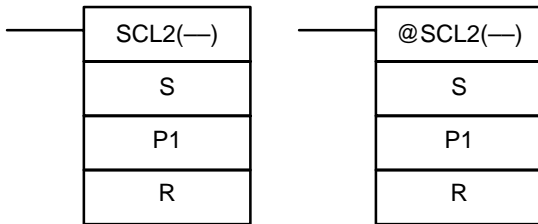
Address	Instruction	Operands
00000	LD	00000
00001	@SCL(66)	
		DM 0100
		DM 0150
		DM 0200

DM 0150	0010
DM 0151	0005
DM 0152	0050
DM 0153	0019



### 5-19-11 SIGNED BINARY TO BCD SCALING – SCL2(—)

**Ladder Symbols**



**Operand Data Areas**

<b>S:</b> Source word
IR, SR, AR, DM, HR, LR
<b>P1:</b> First parameter word
IR, SR, AR, DM, HR, LR
<b>R:</b> Result word
IR, SR, AR, DM, HR, LR

**Limitations**

This instruction is available in the **CQM1-CPU4□-E/-EV1 only**.

S must be BCD.

P1 through P1+2 must be in the same data area.

DM 6144 to DM 6655 cannot be used for R.

**Description**

SCL2(—) is used to linearly convert a 4-digit signed hexadecimal value to a 4-digit BCD value. Unlike BCD(24), which converts a 4-digit hexadecimal value to its 4-digit BCD equivalent ( $S_{hex} \rightarrow S_{BCD}$ ), SCL2(—) can convert the signed hexadecimal value according to a specified linear relationship. The conversion line is defined by the x-intercept and the slope of the line specified in the parameter words P1 to P1+2.

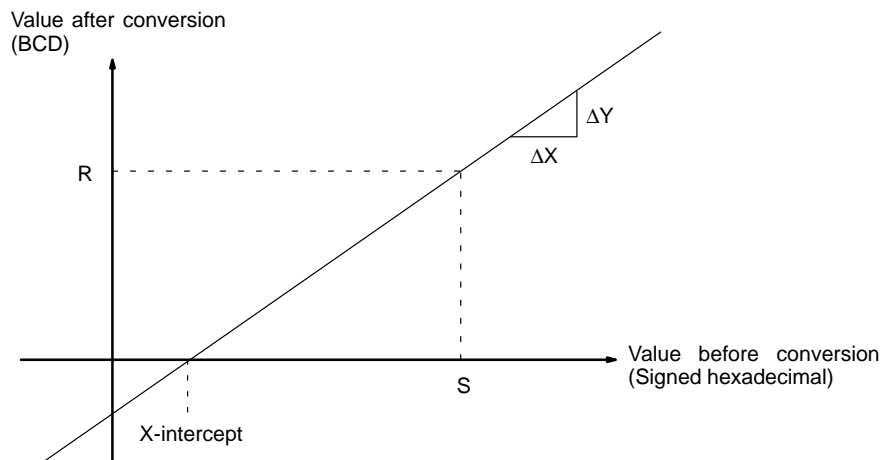
When the execution condition is OFF, SCL2(—) is not executed. When the execution condition is ON, SCL2(—) converts the 4-digit signed hexadecimal value in S to the 4-digit BCD value on the line defined by the x-intercept (P1, 0) and the slope (P1+2 ÷ P1+1) and places the results in R. The result is rounded off to the nearest integer.

If the result is negative, then CY is set to 1. If the result is less than -9999, then -9999 is written to R. If the result is greater than 9999, then 9999 is written to R.

The following table shows the functions and ranges of the parameter words:

Parameter	Function	Range
P1	x-intercept (signed hex.)	8000 to 7FFF (-32,768 to 32,767)
P1+1	$\Delta X$ (signed hex.)	8000 to 7FFF (-32,768 to 32,767)
P1+2	$\Delta Y$ (BCD)	0000 to 9999

The following diagram shows the source word, S, converted to R according to the line defined by the point (P1, 0) and slope  $\Delta Y/\Delta X$ .



The result can be calculated by first converting all signed hexadecimal values to BCD and then using the following formula.

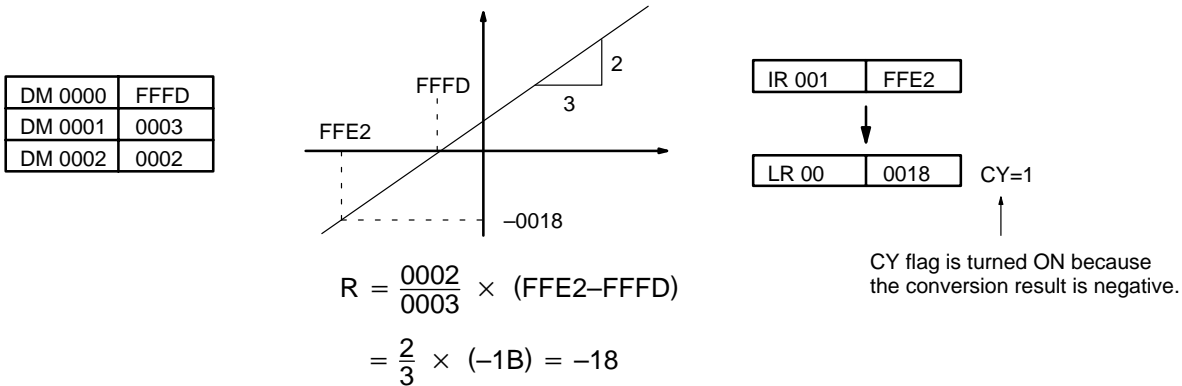
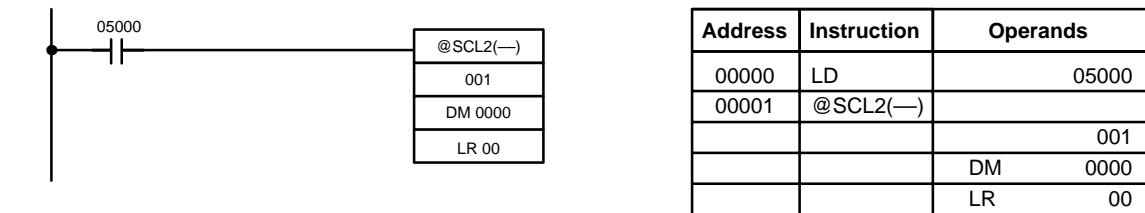
$$R = \frac{?Y}{?X} \times (S-P1)$$

**Flags**

- ER:** Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)  
P1 and P1+2 are not in the same data area, or other setting error.
- CY:** ON when the result, R, is negative.
- EQ:** ON when the result, R, is 0000.

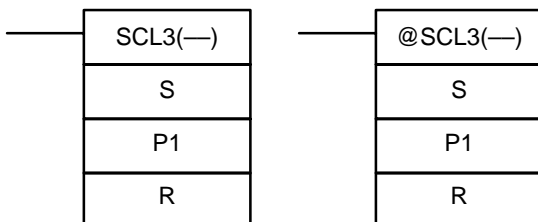
**Example**

When 05000 is turned ON in the following example, the signed binary source data in 001 (#FFE2) is converted to BCD according to the parameters in DM 0000 to DM 0002. The result (#0018) is then written to LR 00 and CY is turned ON because the result is negative.



**5-19-12 BCD TO SIGNED BINARY SCALING – SCL3(—)**

**Ladder Symbols**



**Operand Data Areas**

<b>S:</b> Source word
IR, SR, AR, DM, HR, LR
<b>P1:</b> First parameter word
IR, SR, AR, DM, HR, LR
<b>R:</b> Result word
IR, SR, AR, DM, HR, LR

**Limitations**

- This instruction is available in the **CQM1-CPU4□-E/-EV1 only**.
- P1+1 must be BCD.
- P1 through P1+4 must be in the same data area.
- DM 6144 to DM 6655 cannot be used for R.

**Description**

SCL3(—) is used to linearly convert a 4-digit 4-digit BCD value to 4-digit signed hexadecimal. SCL3(—) converts the BCD value according to a specified linear relationship. The conversion line is defined by the y-intercept and the slope of the line specified in the parameter words P1 to P1+2.

When the execution condition is OFF, SCL3(—) is not executed. When the execution condition is ON, SCL3(—) converts the 4-digit BCD value in S to the 4-digit signed hexadecimal value on the line defined by the y-intercept (0, P1) and the slope (P1+2 ÷ P1+1) and places the result in R. The result is rounded off to the nearest integer.

The content of S can be 0000 to 9999, but S will be treated as a negative value if CY=1, so the effective range of S is actually -9999 to 9999. Be sure to set the desired sign in CY using STC(40) or CLC(41).

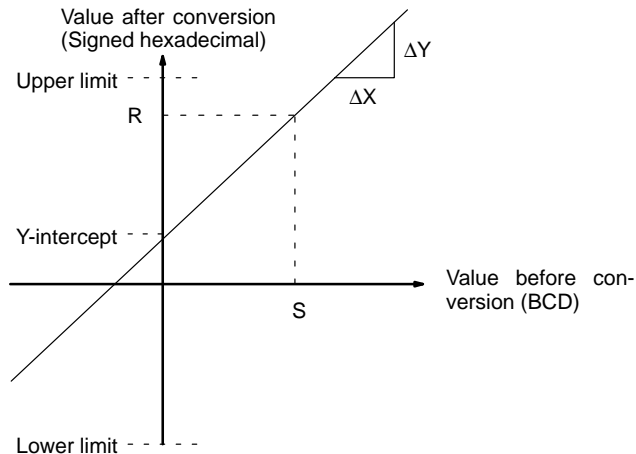
Parameter words P1+3 and P1+4 define upper and lower limits for the result. If the result is greater than the upper limit in P1+3, then the upper limit is written to R. If the result is less than the lower limit in P1+4, then the lower limit is written to R.

**Note** The upper and lower limits for a 12-bit Analog Input Unit would be 07FF and F800.

The following table shows the functions and ranges of the parameter words:

Parameter	Function	Range
P1	x-intercept (signed hex.)	8000 to 7FFF (-32,768 to 32,767)
P1+1	ΔX (BCD)	0000 to 9999
P1+2	ΔY (signed hex.)	8000 to 7FFF (-32,768 to 32,767)
P1+3	Upper limit (signed hex.)	8000 to 7FFF (-32,768 to 32,767)
P1+4	Lower limit (signed hex.)	8000 to 7FFF (-32,768 to 32,767)

The following diagram shows the source word, S, converted to R according to the line defined by the point (0, P1) and slope ΔY/ΔX.



The result can be calculated by first converting all BCD values to signed binary and then using the following formula.

$$R = \left[ \frac{\Delta Y}{\Delta X} \times S \right] + P1$$

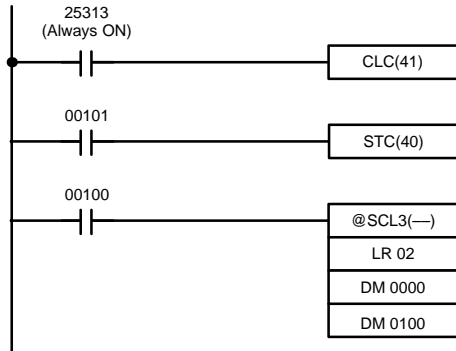
**Flags**

- ER:** Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)  
The content of S is not BCD.
- CY:** CY is not changed by SCL3(—). (CY shows the sign of S before execution.)
- EQ:** ON when the result, R, is 0000.

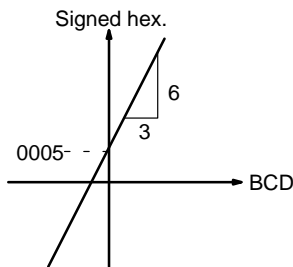


**Example**

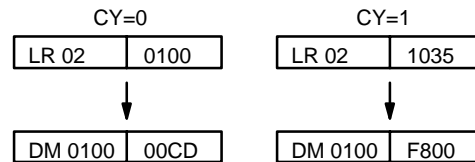
The status of 00101 determines the sign of the BCD source word in the following example. If 00101 is ON, then the source word is negative. When 00100 is turned ON, the BCD source data in LR 02 is converted to signed binary according to the parameters in DM 0000 to DM 0004. The result is then written to DM 0100. (In the second conversion, the signed binary equivalent of -1035 is less than the lower limit specified in DM 0004, so the lower limit is written to DM 0100.)



Address	Instruction	Operands
00000	LD	25313
00001	CLC(41)	
00002	LD	00101
00101	STC(40)	
00004	LD	00100
00005	SCL3(—)	
		LR 02
		DM 0000
		DM 0100

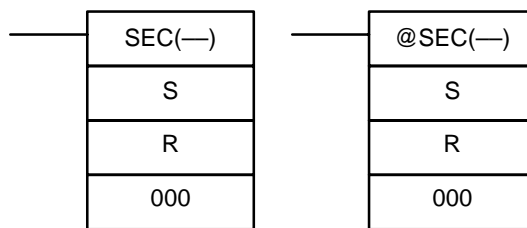


DM 0000	0005
DM 0001	0003
DM 0002	0006
DM 0003	07FF
DM 0004	F800



**5-19-13 HOURS-TO-SECONDS – SEC(—)**

**Ladder Symbols**



**Operand Data Areas**

<b>S:</b> Beginning source word (BCD)
IR, SR, AR, DM, HR, TC, LR
<b>R:</b> Beginning result word (BCD)
IR, SR, AR, DM, HR, TC, LR
<b>000:</b> No function
000

**Limitations**

This instruction is available in the **CQM1 only**.

S and S+1 must be within the same data area. R and R+1 must be within the same data area. S and S+1 must be BCD and must be in the proper hours/minutes/seconds format.

DM 6143 to DM 6655 cannot be used for R.

**Description**

SEC(—) is used to convert time notation in hours/minutes/seconds to an equivalent in just seconds.

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

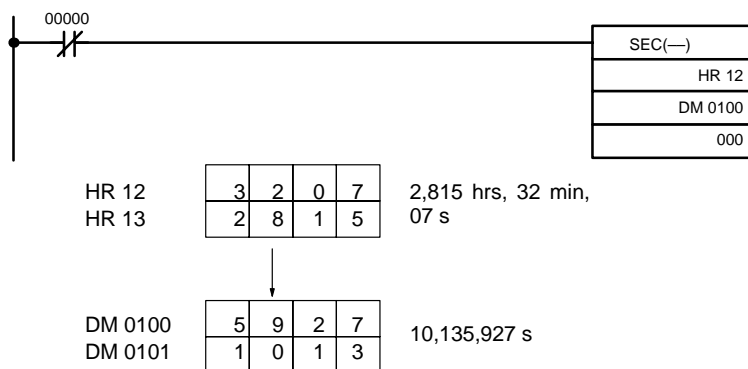
The result is output to R and R+1. The maximum obtainable value is 35,999,999 seconds.

**Flags**

- ER:** S and S+1 or R and R+1 are not in the same data area.  
S and/or S+1 do not contain BCD.  
Number of seconds and/or minutes exceeds 59.  
Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)
- EQ:** ON when the result is zero.

**Example**

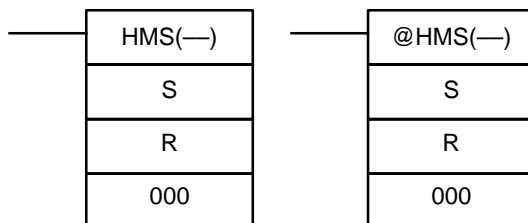
When 00000 is OFF (i.e., when the execution condition is ON), the following instruction would convert the hours, minutes, and seconds given in HR 12 and HR 13 to seconds and store the results in DM 0100 and DM 0101 as shown.



Address	Instruction	Operands
00000	LD NOT	00000
00001	SEC(→)	
		HR 12
		DM 0100
		000

**5-19-14 SECONDS-TO-HOURS – HMS(→)**

**Ladder Symbols**



**Operand Data Areas**

<b>S:</b> Beginning source word (BCD)
IR, SR, AR, DM, HR, TC, LR
<b>R:</b> Beginning result word (BCD)
IR, SR, AR, DM, HR, TC, LR
<b>000:</b> No function
000

**Limitations**

This instruction is available in the **CQM1 only**.  
S and S+1 must be within the same data area. R and R+1 must be within the same data area. S and S+1 must be BCD and must be between 0 and 35,999,999 seconds.  
DM 6143 to DM 6655 cannot be used for R.

**Description**

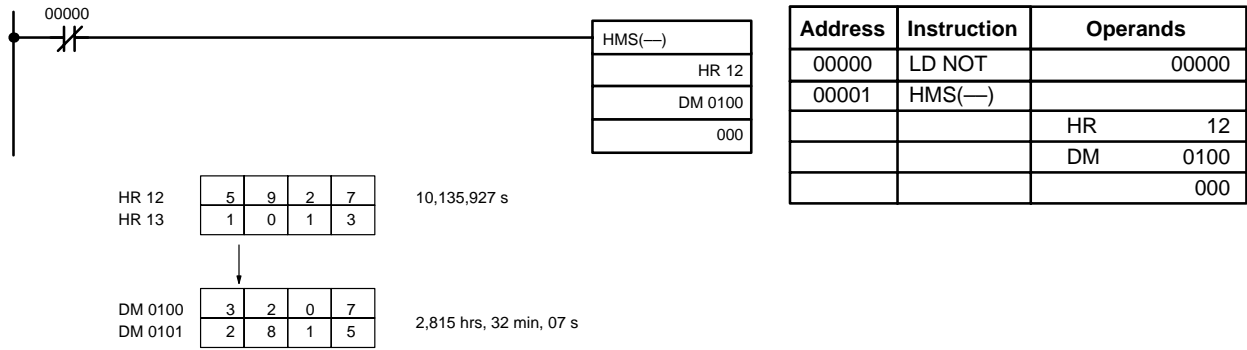
HMS(→) is used to convert time notation in seconds to an equivalent in hours/minutes/seconds.  
The number of seconds designated in S and S+1 is converted to hours/minutes/seconds and placed in R and R+1.  
For the results, the seconds are placed in bits 00 through 07 and the minutes are placed in bits 08 through 15 of R. The hours are placed in R+1. The maximum is 9,999 hours, 59 minutes, and 59 seconds.

**Flags**

- ER:** S and S+1 or R and R+1 are not in the same data area.  
S and/or S+1 do not contain BCD or exceed 36,000,000 seconds.  
Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)
- EQ:** ON when the result is zero.

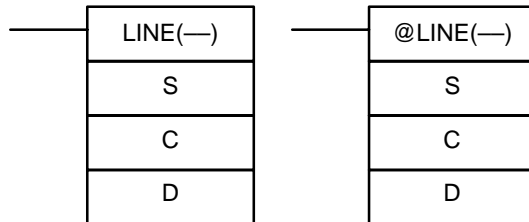
**Example**

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



**5-19-15 COLUMN-TO-LINE – LINE(—)**

**Ladder Symbols**



**Operand Data Areas**

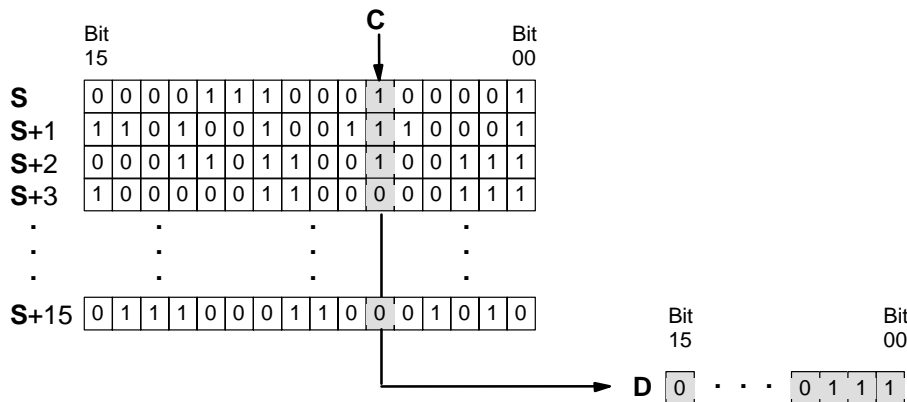
<b>S:</b> First word of 16 word source set
IR, SR, AR, DM, HR, TC, LR
<b>C:</b> Column bit designator (BCD)
IR, SR, AR, DM, HR, TC, LR, #
<b>D:</b> Destination word
IR, SR, AR, DM, HR, TC, LR

**Limitations**

This instruction is available in the **CQM1 only**.  
 S and S+15 must be in the same data area.  
 C must be BCD between #0000 and #0015.  
 DM 6144 to DM 6655 cannot be used for D.

**Description**

When the execution condition is OFF, LINE(—) is not executed. When the execution condition is ON, LINE(—) copies bit column C from the 16-word set (S to S+15) to the 16 bits of word D (00 to 15).



**Flags**

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

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

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

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

**Example**

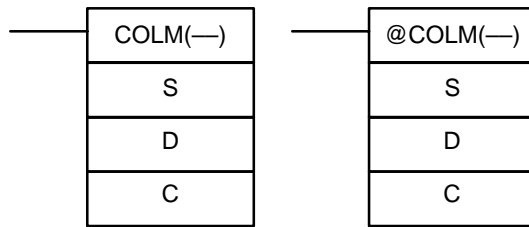
The following example shows how to use LINE(—) to move bit column 07 from the set (IR 100 to IR 115) to DM 0100.



Address	Instruction	Operands
00000	LD	00000
00001	LINE(—)	
		100
		# 0007
		DM 0100

**5-19-16 LINE-TO-COLUMN – COLM(—)**

**Ladder Symbols**



**Operand Data Areas**

<b>S:</b> Source word
IR, SR, AR, DM, HR, TC, LR

<b>D:</b> First word of the destination set
IR, SR, AR, DM, HR, TC LR

<b>C:</b> Column bit designator (BCD)
IR, SR, AR, DM, HR, TC, LR, #

**Limitations**

This instruction is available in the **CQM1 only**.

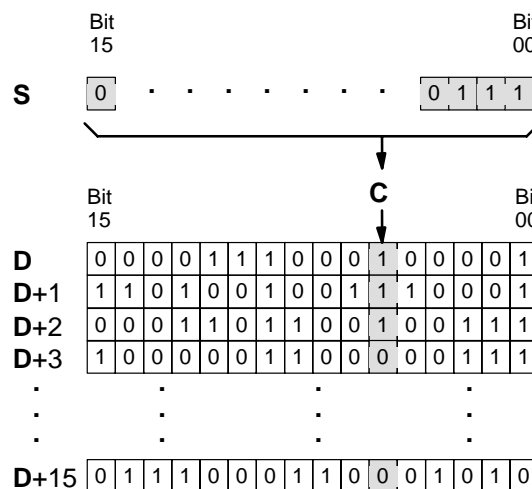
D and D+15 must be in the same data area.

DM 6129 to DM 6655 cannot be used for D.

C must be BCD between #0000 and #0015.

**Description**

When the execution condition is OFF, COLM(—) is not executed. When the execution condition is ON, COLM(—) copies the 16 bits of word S (00 to 15) to the column of bits, C, of the 16-word set (D to D+15).



**Flags**

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

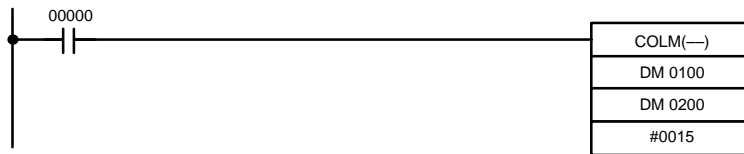
Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)

D and D+15 are not in the same data area.

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

**Example**

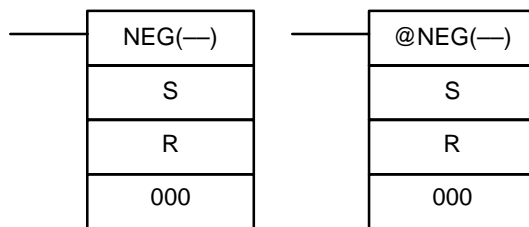
The following example shows how to use COLM(—) to move the contents of word DM 0100 (00 to 15) to bit column 15 of the set (DM 0200 to DM 0215).



Address	Instruction	Operands
00000	LD	00000
00001	COLM(—)	
		DM 0100
		DM 0200
		# 0015

**5-19-17 2'S COMPLEMENT – NEG(—)**

**Ladder Symbols**



**Operand Data Areas**

<b>S:</b> Source word
IR, SR, AR, DM, HR, TC, LR, #
<b>R:</b> Result word
IR, SR, AR, DM, HR, LR
<b>000</b>
Not used. Set to 000.

**Limitations**

This instruction is available in the **CQM1-CPU4□-E/-EV1 only**.  
DM 6144 to DM 6655 cannot be used for R.

**Description**

Converts the four-digit hexadecimal content of the source word (S) to its 2's complement and outputs the result to the result word (R). This operation is effectively the same as subtracting S from 0000 and outputting the result to R; it will calculate the absolute value of negative signed binary data.

If the content of S is 0000, the content of R will also be 0000 after execution and EQ (SR 25506) will be turned on.

If the content of S is 8000, the content of R will also be 8000 after execution and UF (SR 25405) will be turned on.

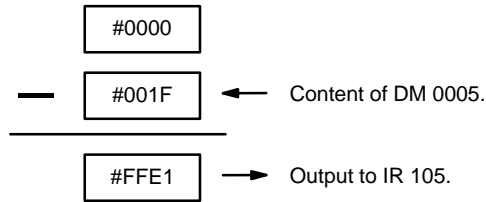
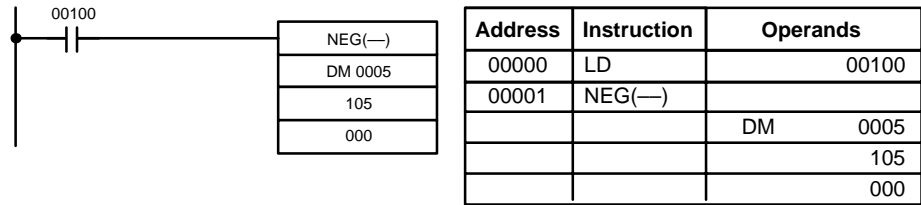
**Note** Refer to 1-10 *Calculating with Signed Binary Data* for more details.

**Flags**

- ER:** Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)
- EQ:** ON when the content of R is zero after execution; otherwise OFF.
- UF:** ON when the content of S is 8000; otherwise OFF.

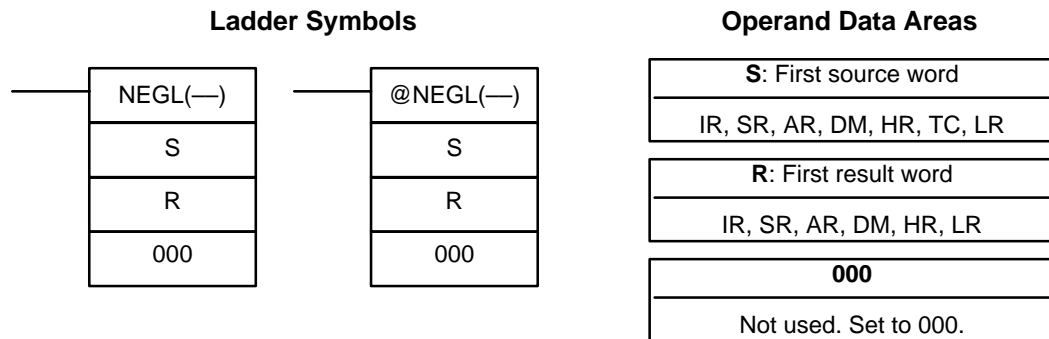
**Example**

The following example shows how to use NEG(—) to find the 2's complement of the content of DM 0005 and output the result to IR 105.



**5-19-18**

**DOUBLE 2'S COMPLEMENT – NEGL(—)**



**Limitations**

This instruction is available in the **CQM1-CPU4□-E/-EV1 only**.  
 DM 6143 to DM 6655 cannot be used for R.  
 S and S+1 must be in the same data area, as must R and R+1.

**Description**

Converts the eight-digit hexadecimal content of the source words (S and S+1) to its 2's complement and outputs the result to the result words (R and R+1). This operation is effectively the same as subtracting the eight-digit content S and S+1 from \$0000 0000 and outputting the result to R and R+1; it will calculate the absolute value of negative signed binary data.

If the content of S is 0000 0000, the content of R will also be 0000 0000 after execution and EQ (SR 25506) will be turned on.

If the content of S is 8000 0000, the content of R will also be 8000 0000 after execution and UF (SR 25405) will be turned on.

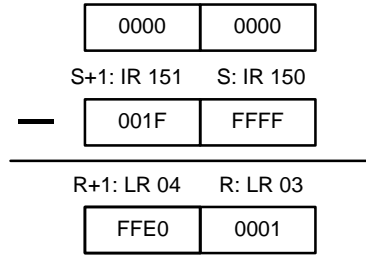
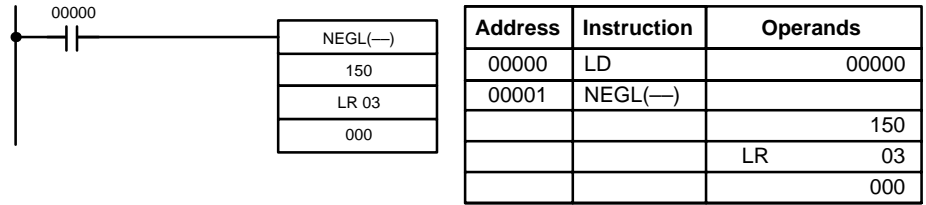
**Note** Refer to 1-10 *Calculating with Signed Binary Data* for more details.

**Flags**

- ER:** Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)
- EQ:** ON when the content of R+1, R is zero after execution; otherwise OFF.
- UF:** ON when the content of S+1, S is 8000 0000; otherwise OFF.

**Example**

The following example shows how to use NEGL(—) to find the 2's complement of the hexadecimal value in IR 151, IR 150 (001F FFFF) and output the result to HR 04, HR 03.



## 5-20 BCD Calculation Instructions

### 5-20-1 SET CARRY – STC(40)

#### Ladder Symbols



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

**Note** Refer to *Appendix B Error and Arithmetic Flag Operation* for a table listing the instructions that affect CY.

### 5-20-2 CLEAR CARRY – CLC(41)

#### Ladder Symbols



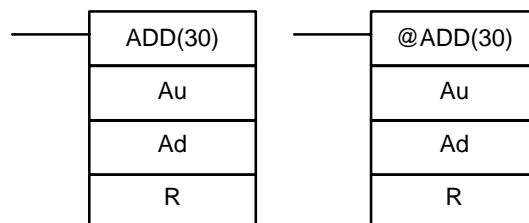
When the execution condition is OFF, CLC(41) is not executed. When the execution condition is ON, CLC(41) turns OFF CY (SR 25504).

CLEAR CARRY is used to reset (turn OFF) CY (SR 25504) to "0."

**Note** Refer to *Appendix B Error and Arithmetic Flag Operation* for a table listing the instructions that affect CY.

### 5-20-3 BCD ADD – ADD(30)

#### Ladder Symbols



#### Operand Data Areas

<b>Au:</b> Augend word (BCD)
IR, SR, AR, DM, HR, TC, LR, #
<b>Ad:</b> Addend word (BCD)
IR, SR, AR, DM, HR, TC, LR, #
<b>R:</b> Result word
IR, SR, AR, DM, HR, LR

#### Limitations

DM 6144 to DM 6655 cannot be used for R.

#### Description

When the execution condition is OFF, ADD(30) is not executed. When the execution condition is ON, ADD(30) adds the contents of Au, Ad, and CY, and places the result in R. CY will be set if the result is greater than 9999.

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

#### 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.)

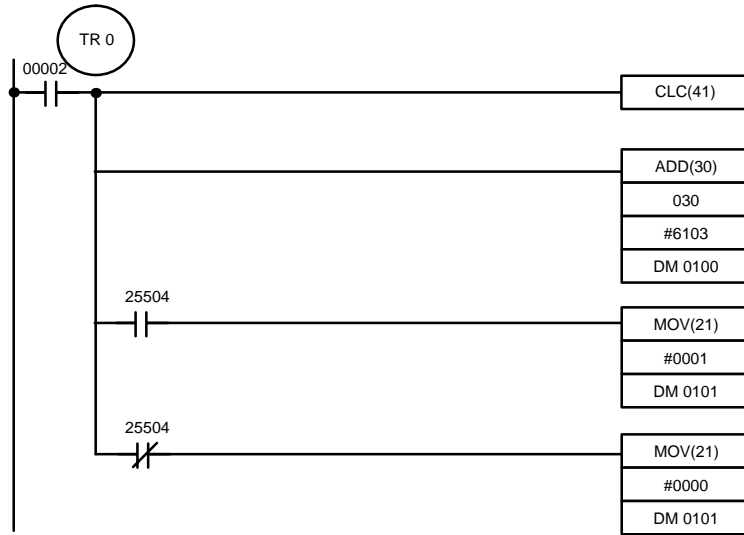
**CY:** ON when there is a carry in the result.

**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 IR 030 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.

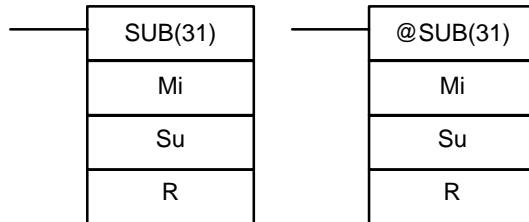


Address	Instruction	Operands
00000	LD	00002
00001	OUT	TR 0
00002	CLC(41)	
00003	ADD(30)	
		030
		# 6103
		DM 0100
00004	AND	25504
00005	MOV(21)	
		# 0001
		DM 0101
00006	LD	TR 0
00007	AND NOT	25504
00008	MOV(21)	
		# 0000
		DM 0101

Although two ADD(30) can be used together to perform eight-digit BCD addition, ADDL(54) is designed specifically for this purpose.

**5-20-4 BCD SUBTRACT – SUB(31)**

**Ladder Symbols**



**Operand Data Areas**

<b>Mi:</b> Minuend word (BCD)
IR, SR, AR, DM, HR, TC, LR, #
<b>Su:</b> Subtrahend word (BCD)
IR, SR, AR, DM, HR, TC, LR, #
<b>R:</b> Result word
IR, SR, AR, DM, HR, LR

**Limitations**

DM 6144 to DM 6655 cannot be used for R.

**Description**

When the execution condition is OFF, SUB(31) is not executed. When the execution condition is ON, SUB(31) subtracts the contents of Su and CY from Mi, and places the result in R. If the result is negative, CY is set and the 10's complement of the actual result is placed in R. To convert the 10's complement to the true result, subtract the content of R from zero (see example below).

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

**Flags**

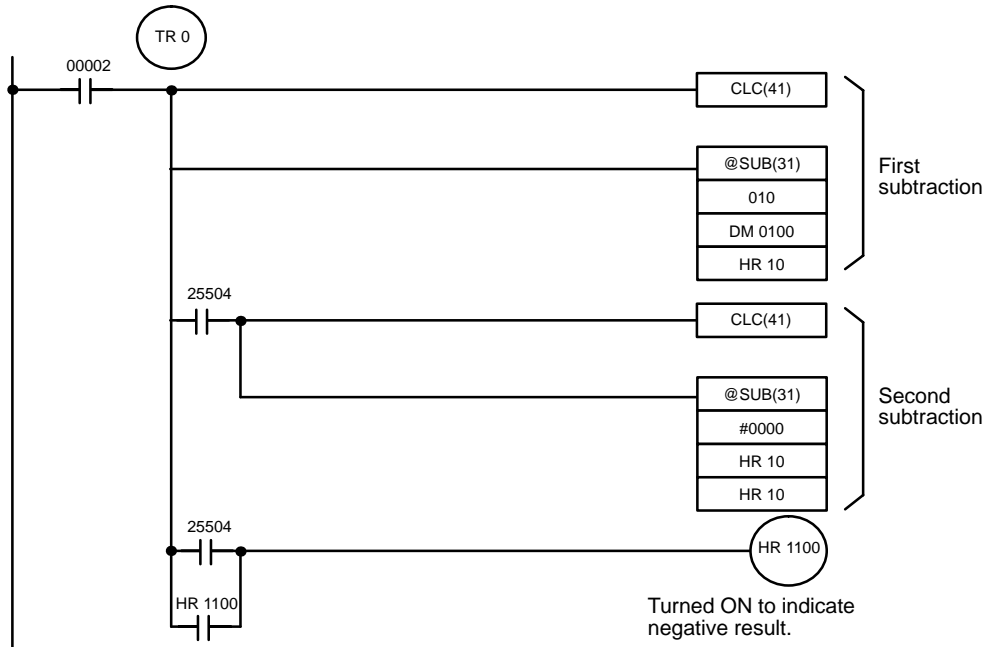
- ER:** Mi and/or Su is not BCD.  
Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)
- CY:** ON when the result is negative, i.e., when Mi is less than Su plus CY.
- EQ:** ON when the result is 0.

**Caution** Be sure to clear the carry flag with CLC(41) before executing SUB(31) if its previous status is not required, and check the status of CY after doing a subtraction with SUB(31). If CY is ON as a result of executing SUB(31) (i.e., if the result is negative), the result is output as the 10's complement of the true answer. To convert the output result to the true value, subtract the value in R from 0.

**Example**

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 10. If CY is set by executing SUB(31), the result in HR 10 is subtracted from zero (note that CLC(41) is again required to obtain an accurate result), the result is placed back in HR 10, and HR 1100 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 1100 is not turned ON. HR 1100 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 rescanned.

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 1100) 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 10
00004	AND	25504
00005	CLC(41)	
00006	@SUB(31)	
		# 0000
		HR 10
		HR 10
00007	LD	TR 0
00008	LD	25504
00009	OR	HR 1100
00010	AND LD	
00011	OUT	HR 1100

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**

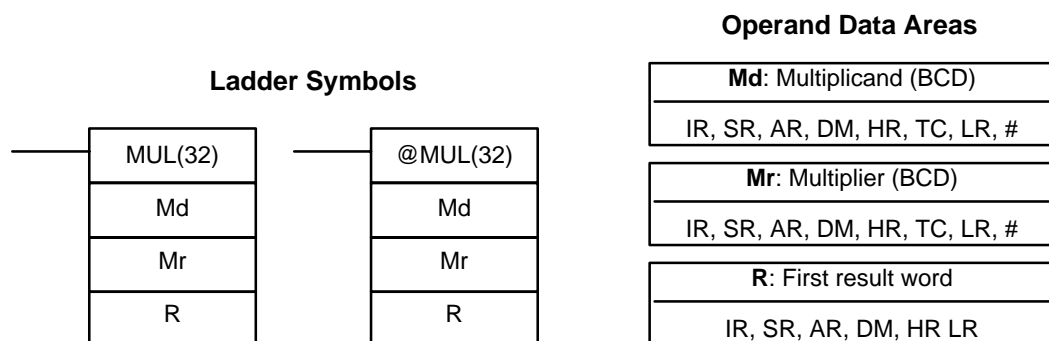
IR 010 1029  
 DM 0100 - 3452  
CY - 0  
 HR 10 7577 (1029 + (10000 - 3452))  
 CY 1 (negative result)

**Second Subtraction**

0000  
 HR 10 -7577  
CY - 0  
 HR 10 2423 (0000 + (10000 - 7577))  
 CY 1 (negative result)

In the above case, the program would turn ON HR 1100 to indicate that the value held in HR 10 is negative.

**5-20-5 BCD MULTIPLY – MUL(32)**

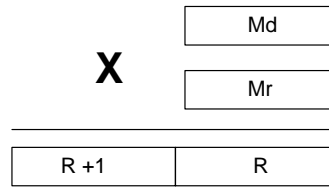


**Limitations**

DM 6143 to DM 6655 cannot be used for R.

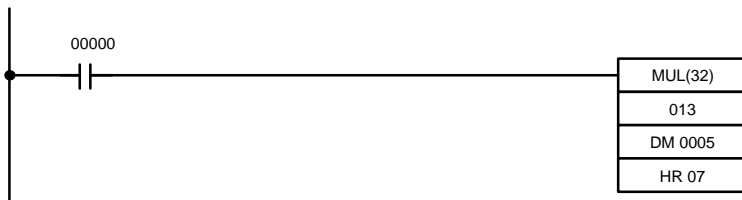
**Description**

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 0005
		HR 07

Md: IR 013			
3	3	5	6

X

Mr: DM 0005			
0	0	2	5

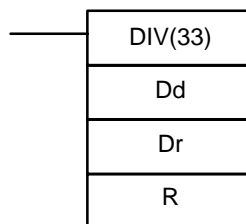
R+1: HR 08	R: HR 07
0 0 0 8	3 9 0 0

**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.

**5-20-6 BCD DIVIDE – DIV(33)**

**Ladder Symbol**



**Operand Data Areas**

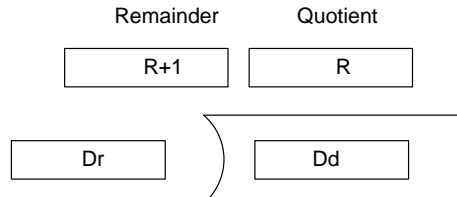
<b>Dd:</b> Dividend word (BCD)
IR, SR, AR, DM, HR, TC, LR, #
<b>Dr:</b> Divisor word (BCD)
IR, SR, AR, DM, HR, TC, LR, #
<b>R:</b> First result word (BCD)
IR, SR, AR, DM, HR, LR

**Limitations**

R and R+1 must be in the same data area. DM 6143 to DM 6655 cannot be used for R.

**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**

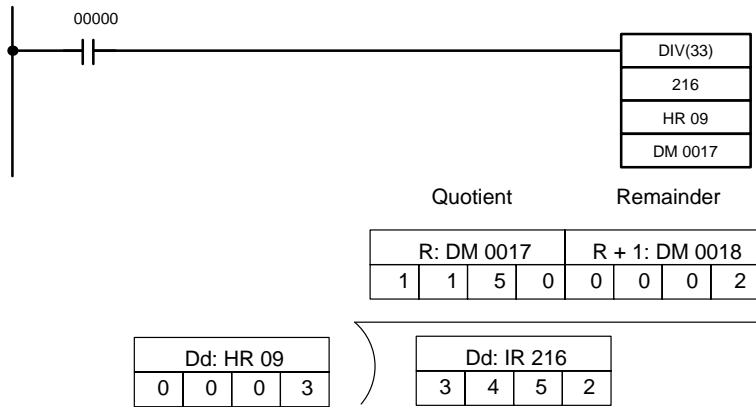
**ER:** Dd or Dr is not in BCD.

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**

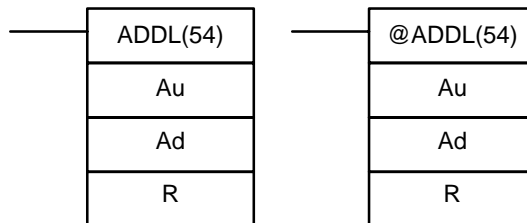
When IR 00000 is ON with the following program, the content of IR 216 is divided by the content of HR 09 and the result is placed in DM 0017 and DM 0018. Example data and calculations are shown below the program.



Address	Instruction	Operands
00000	LD	00000
00001	DIV(33)	
		216
		HR 09
		DM 0017

**5-20-7 DOUBLE BCD ADD – ADDL(54)**

**Ladder Symbols**



**Operand Data Areas**

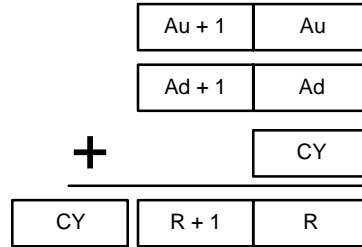
<b>Au:</b> First augend word (BCD)
IR, SR, AR, DM, HR, TC, LR
<b>Ad:</b> First addend word (BCD)
IR, SR, AR, DM, HR, TC, LR
<b>R:</b> First result word
IR, SR, AR, DM, HR, LR

**Limitations**

DM 6143 to DM 6655 cannot be used for R.

**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.)

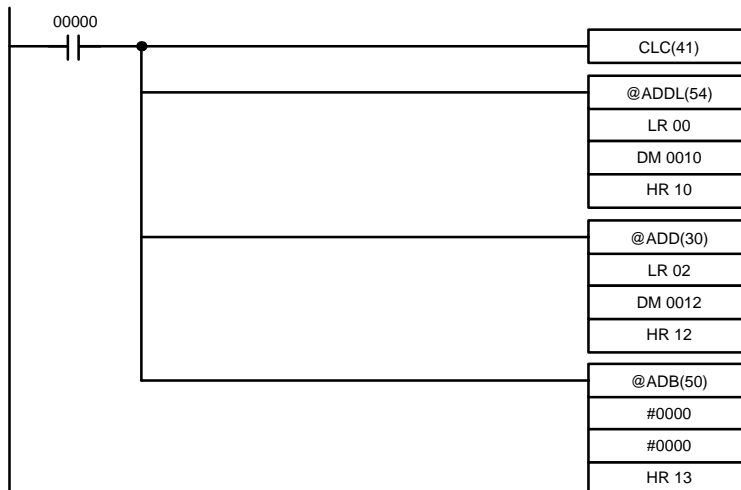
**CY:** ON when there is a carry in the result.

**EQ:** ON when the result is 0.

**Example**

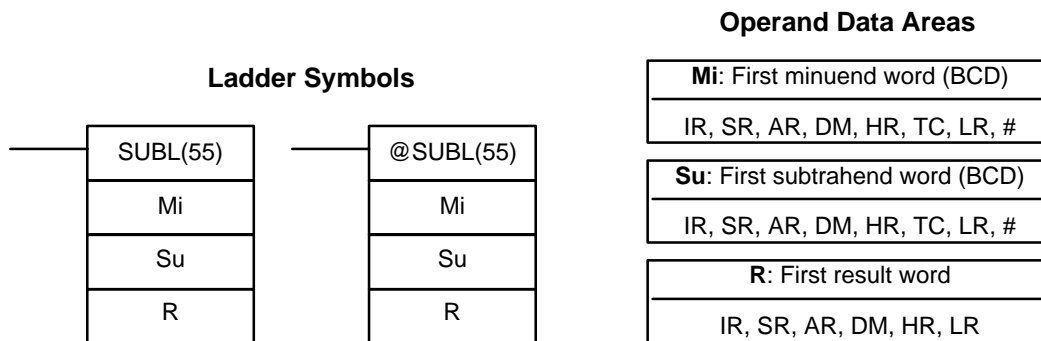
When 00000 is ON, the following program section adds two 12-digit numbers, the first contained in LR 00 through LR 02 and the second in DM 0010 through DM 0012. The result is placed in HR 10 through HR 13.

The rightmost 8 digits of the two numbers are added using ADDL(54), i.e., the contents of LR 00 and LR 01 are added to DM 0010 and DM 0011 and the results is placed in HR 10 and HR 11. The second addition adds the leftmost 4 digits of each number using ADD(30), and includes any carry from the first addition. The last instruction, ADB(50) (see 5-21-1 BINARY ADD – ADB(50)) adds two all-zero constants to place any carry from the second addition into HR 13.



Address	Instruction	Operands
00000	LD	00000
00001	CLC(41)	
00002	@ADDL(54)	
		LR 00
		DM 0010
		HR 10
00003	@ADD(30)	
		LR 02
		DM 0012
		HR 12
00004	@ADB(50)	
		# 0000
		# 0000
		HR 13

### 5-20-8 DOUBLE BCD SUBTRACT – SUBL(55)

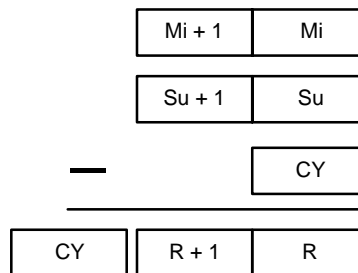


**Limitations**

DM 6143 to DM 6655 cannot be used for R.

**Description**

When the execution condition is OFF, SUBL(55) is not executed. When the execution condition is ON, SUBL(55) subtracts CY and the 8-digit contents of Su and Su+1 from the 8-digit value in Mi and Mi+1, and places the result in R and R+1. If the result is negative, CY is set and the 10's complement of the actual result is placed in R. To convert the 10's complement to the true result, subtract the content of R from zero. Since an 8-digit constant cannot be directly entered, use the BSET(71) instruction (see 5-17-4 BLOCK SET – BSET(71)) to create an 8-digit constant.



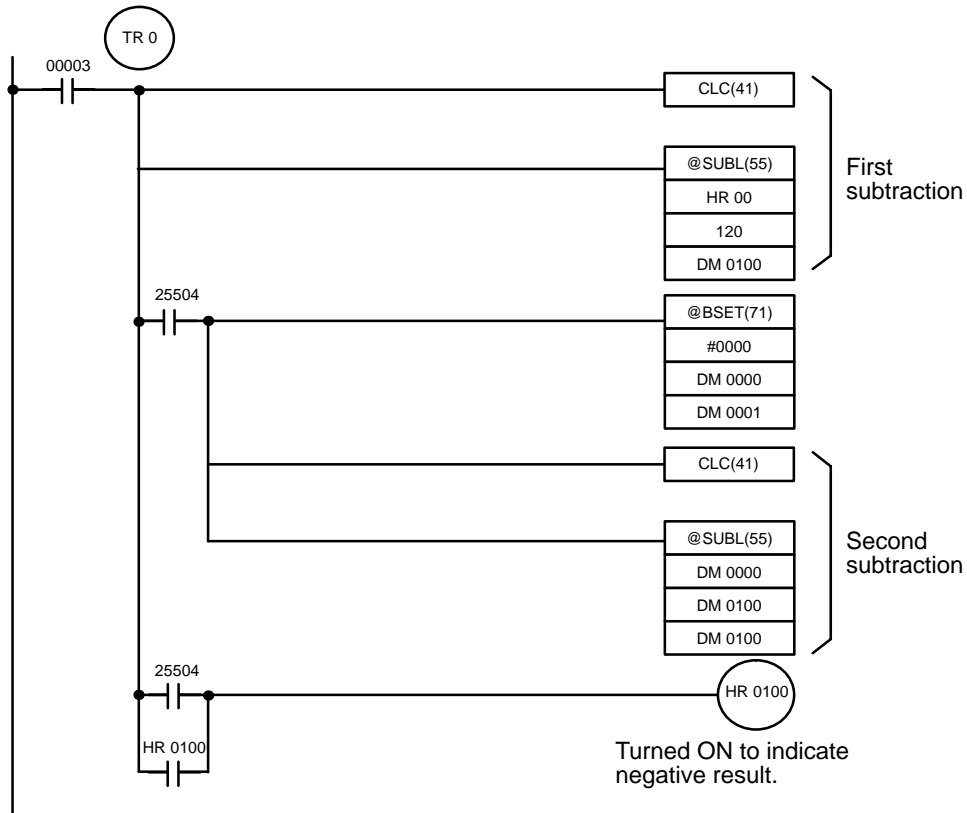
**Flags**

- 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).

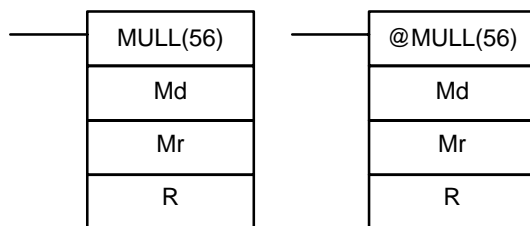


Address	Instruction	Operands
00000	LD	00003
00001	OUT	TR 0
00002	CLC(41)	
00003	@SUBL(55)	
		HR 00
		120
		DM 0100
00004	AND	25504
00005	@BSET(71)	
		# 0000
		DM 0000
		DM 0001

Address	Instruction	Operands
00006	CLC(41)	
00007	@SUBL(55)	
		DM 0000
		DM 0100
		DM 0100
00008	LD	TR 0
00009	LD	25504
00010	OR	HR 0100
00011	AND LD	
00012	OUT	HR 0100

### 5-20-9 DOUBLE BCD MULTIPLY – MULL(56)

#### Ladder Symbols



#### Operand Data Areas

<b>Md:</b> First multiplicand word (BCD)
IR, SR, AR, DM, HR, TC, LR
<b>Mr:</b> First multiplier word (BCD)
IR, SR, AR, DM, HR, TC, LR
<b>R:</b> First result word
IR, SR, AR, DM, HR LR

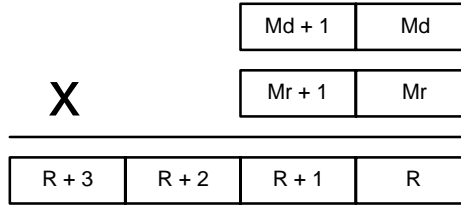


**Limitations**

DM 6141 to DM 6655 cannot be used for R.

**Description**

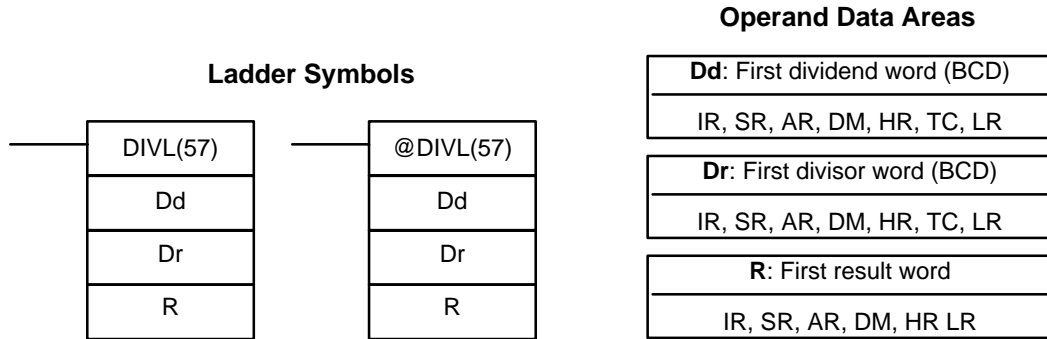
When the execution condition is OFF, MULL(56) is not executed. When the execution condition is ON, MULL(56) multiplies the eight-digit content of Md and Md+1 by the content of Mr and Mr+1, and places the result in R to R+3.



**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.

**5-20-10 DOUBLE BCD DIVIDE – DIVL(57)**

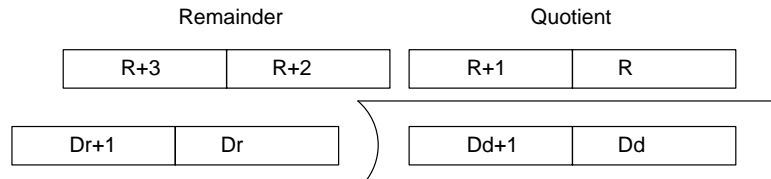


**Limitations**

DM 6141 to DM 6655 cannot be used for R.

**Description**

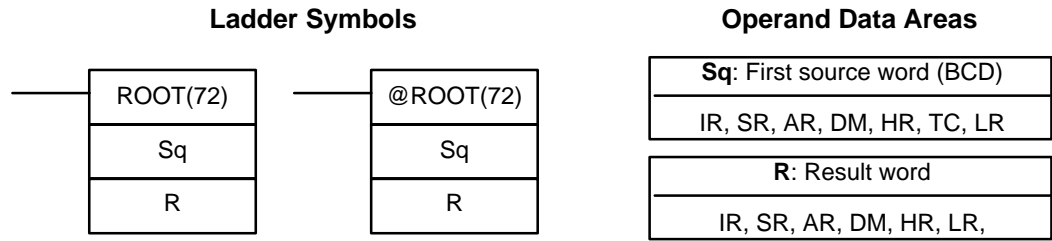
When the execution condition is OFF, DIVL(57) is not executed. When the execution condition is ON, DIVL(57) the eight-digit content of Dd and D+1 is divided by the content of Dr and Dr+1 and the result is placed in R to R+3: the quotient in R and R+1, the remainder in R+2 and R+3.



**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.

### 5-20-11 SQUARE ROOT – ROOT(72)

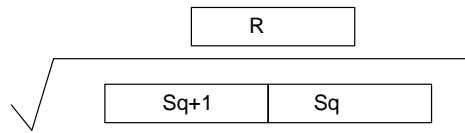


**Limitations**

This instruction is available in the **CQM1 only**.  
DM 6144 to DM 6655 cannot be used for R.

**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.



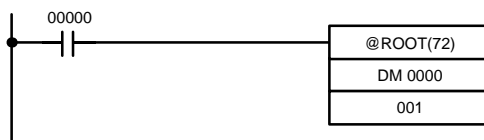
**Flags**

**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 an eight digit number. The result is a four-digit number, with the remainder rounded off. and then round the result.

In this example,  $\sqrt{63250561} = 7953.0221\dots$ , which is rounded off to 7953.



Address	Instruction	Operands
00000	LD	00000
00001	@ROOT(72)	
		DM 0000
		001

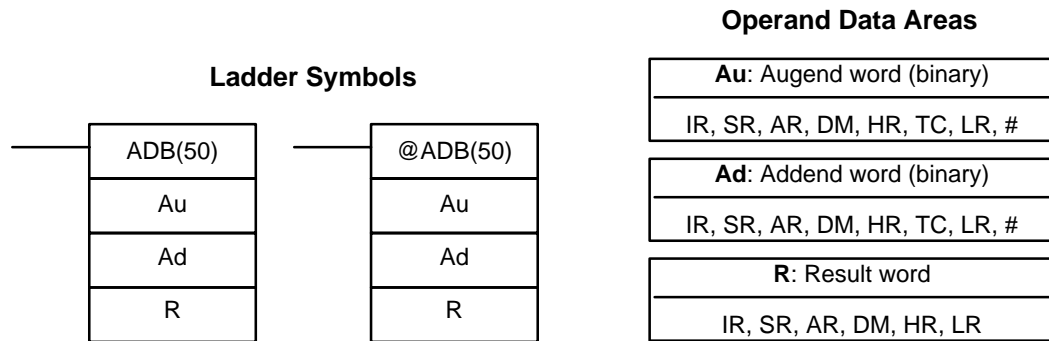
DM 0001				DM 0000			
6	3	2	5	0	5	6	1

$\sqrt{63,250,561} = 7953.0221$   
(The remainder is rounded off.)

001
7 9 5 3

# 5-21 Binary Calculation Instructions

## 5-21-1 BINARY ADD – ADB(50)



**Limitations**

DM 6144 to DM 6655 cannot be used for R.

**Description**

When the execution condition is OFF, ADB(50) is not executed. When the execution condition is ON, ADB(50) adds the contents of Au, Ad, and CY, and places the result in R. CY will be set if the result is greater than FFFF.

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

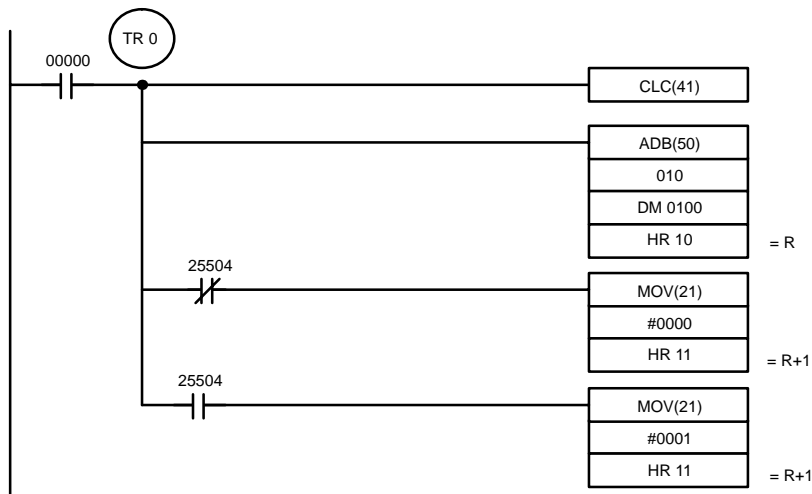
ADB(50) can also be used to add signed binary data. With the CQM1-CPU4□-EV1, CPM1A, and SRM1, the underflow and overflow flags (SR 25404 and SR 25405) indicate whether the result has exceeded the lower or upper limits of the 16-bit signed binary data range.

**Flags**

- ER:** Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)
- CY:** ON when the result is greater than FFFF.
- EQ:** ON when the result is 0.
- OF:** ON when the result exceeds +32,767 (7FFF). (CQM1-CPU4□-E/-EV1 only)
- UF:** ON when the result is below -32,768 (8000). (CQM1-CPU4□-E/-EV1 only)

**Example**

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



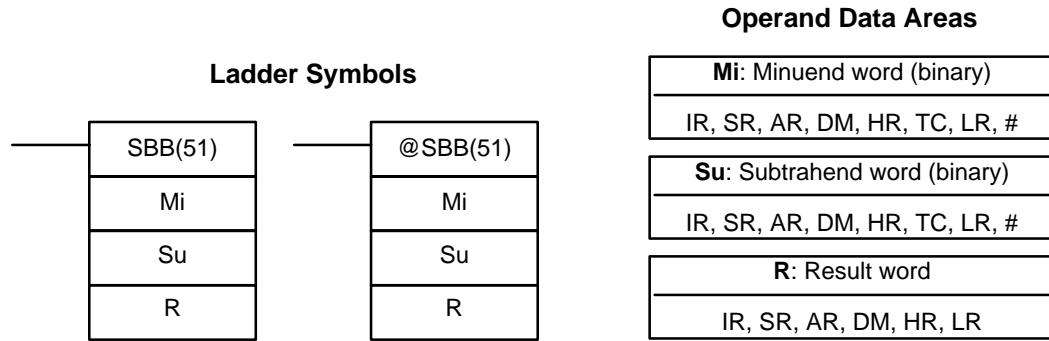
Address	Instruction	Operands
00000	LD	00000
00001	OUT	TR 0
00002	CLC(41)	
00003	ADB(50)	
		010
		DM 0100
		HR 10
00004	AND NOT	25504
00005	MOV(21)	# 0000
		HR 11
00006	LD	TR 0
00007	AND	25504
00008	MOV(21)	# 00001
		HR 11

In the case below, A6E2 + 80C5 = 127A7. The result is a 5-digit number, so CY (SR 25504) = 1, and the content of R + 1 becomes #0001.

				Au: IR 010							
				A	6	E	2				
				+							
				Ad: DM 0100							
				8	0	C	5				
-----											
				R+1: HR 11				R: HR 10			
0	0	0	1	2	7	A	7				

**Note** For signed binary calculations, the status of the UF and OF flags indicate whether the result has exceeded the signed binary data range (–32,768 (8000) to +32,767 (7FFF)). (CQM1-CPU4□-EV1 only)

### 5-21-2 BINARY SUBTRACT – SBB(51)



**Limitations**

DM 6144 to DM 6655 cannot be used for R.

**Description**

When the execution condition is OFF, SBB(51) is not executed. When the execution condition is ON, SBB(51) subtracts the contents of Su and CY from Mi and places the result in R. If the result is negative, CY is set and the 2’s complement of the actual result is placed in R.

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

SBB(51) can also be used to subtract signed binary data. With CQM1-CPU4□-EV1, CPM1A, and SRM1, the underflow and overflow flags (SR 25404 and SR 25405) indicate whether the result has exceeded the lower or upper limits of the 16-bit signed binary data range.

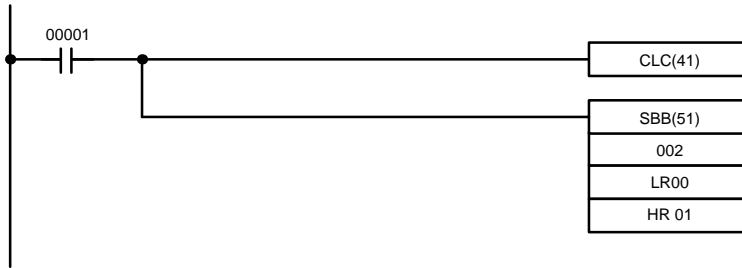
**Flags**

- ER:** Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)
- CY:** ON when the result is negative, i.e., when Mi is less than Su plus CY.
- EQ:** ON when the result is 0.
- OF:** ON when the result exceeds +32,767 (7FFF). (CQM1-CPU4□-EV1 only)
- UF:** ON when the result is below –32,768 (8000). (CQM1-CPU4□-EV1 only)

**Example**

The following example shows a four-digit subtraction. When IR 00001 is ON, the content of LR 00 and CY are subtracted from the content of IR 002 and the result is written to HR 01.

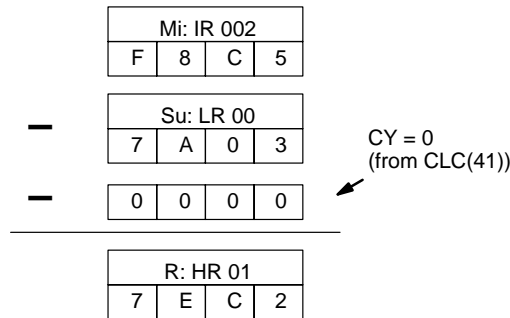
CY is turned ON if the result is negative. If normal data is being used, a negative result (signed binary) must be converted to normal data using NEG(—). Refer to 5-19-17 2's COMPLEMENT – NEG(—) for details.



Address	Instruction	Operands
00000	LD	00001
00001	OUT	TR 1
00002	CLC(41)	
00003	SBB(51)	
		002
		LR 00
		HR 01

In the case below, the content of LR 00 (#7A03) and CY are subtracted from IR 002 (#F8C5). Since the result is positive, CY is 0.

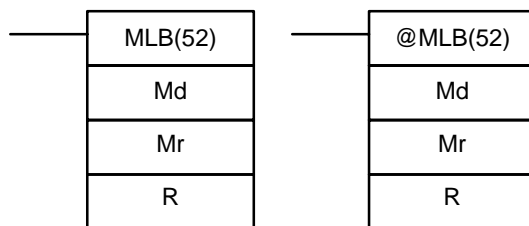
If the result had been negative, CY would have been set to 1. For normal (unsigned) data, the result would have to be converted to its 2's complement.



**Note** For signed binary calculations, the status of the UF and OF flags indicate whether the result has exceeded the signed binary data range (–32,768 (8000) to +32,767 (7FFF)). (CQM1-CPU4□-EV1 only)

### 5-21-3 BINARY MULTIPLY – MLB(52)

#### Ladder Symbols



#### Operand Data Areas

<b>Md:</b> Multiplicand word (binary)
IR, SR, AR, DM, HR, TC, LR, #
<b>Mr:</b> Multiplier word (binary)
IR, SR, AR, DM, HR, TC, LR, #
<b>R:</b> First result word
IR, SR, AR, DM, HR LR

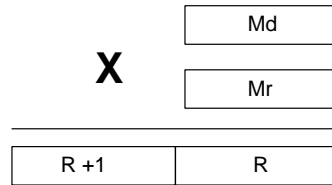
#### Limitations

DM 6143 to DM 6655 cannot be used for R.

MLB(52) cannot be used to multiply signed binary data. In CQM1 PCs, MBS(—) can be used. Refer to 5-21-7 SIGNED BINARY MULTIPLY – MBS(—).

**Description**

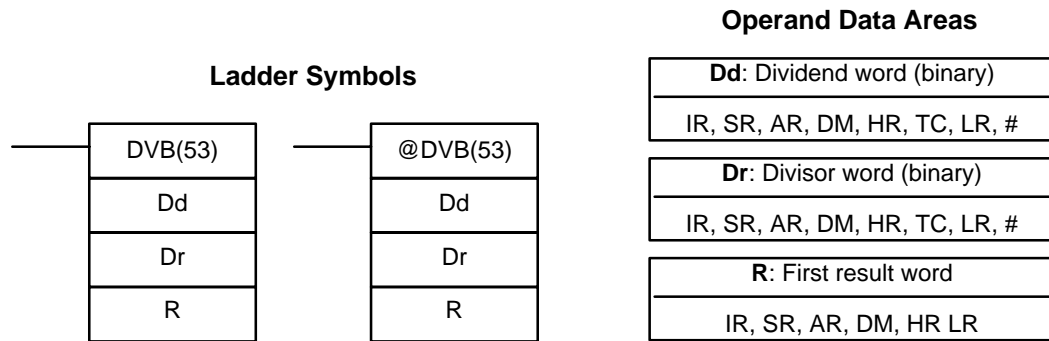
When the execution condition is OFF, MLB(52) is not executed. When the execution condition is ON, MLB(52) multiplies the content of Md by the contents of Mr, places the rightmost four digits of the result in R, and places the leftmost four digits in R+1.



**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.

**5-21-4 BINARY DIVIDE – DVB(53)**

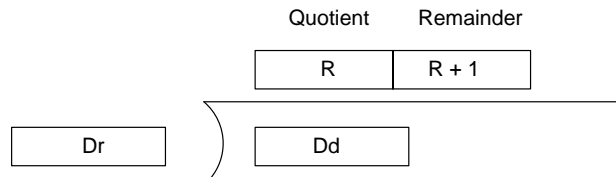


**Limitations**

DM 6143 to DM 6655 cannot be used for R.  
 DVB(53) cannot be used to divide signed binary data. In CQM1 PCs, DBS(—) can be used. Refer to 5-21-9 SIGNED BINARY DIVIDE – DBS(—) for details.

**Description**

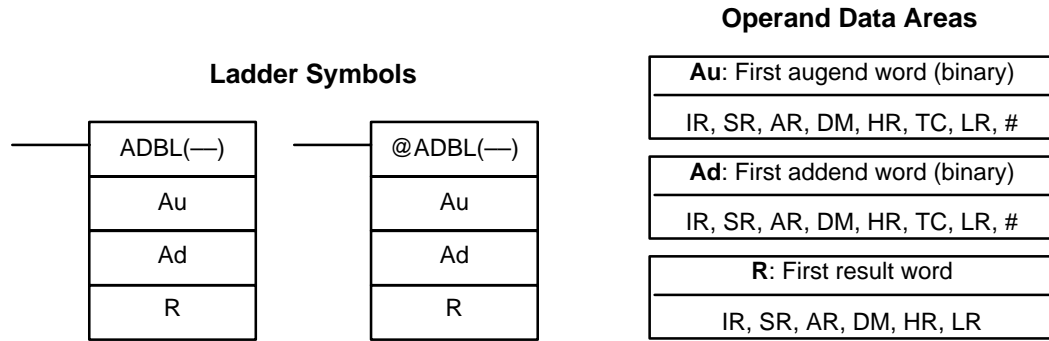
When the execution condition is OFF, DVB(53) is not executed. When the execution condition is ON, DVB(53) divides the content of Dd by the content of Dr and the result is placed in R and R+1: the quotient in R, the remainder in R+1.



**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.

### 5-21-5 DOUBLE BINARY ADD – ADBL(—)

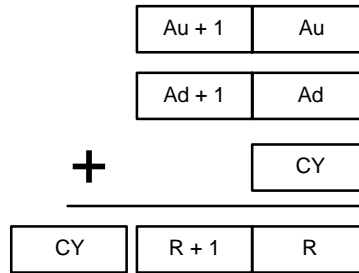


**Limitations**

This instruction is available in the **CQM1-CPU4□-E/-EV1 only**.  
 Au and Au+1 must be in the same data area, as must Ad and Ad+1, and R and R+1.  
 DM 6142 to DM 6655 cannot be used for R.

**Description**

When the execution condition is OFF, ADBL(—) is not executed. When the execution condition is ON, ADBL(—) adds the eight-digit contents of Au+1 and Au, the eight-digit contents of Ad+1 and Ad, and CY, and places the result in R. CY will be set if the result is greater than FFFF FFFF.



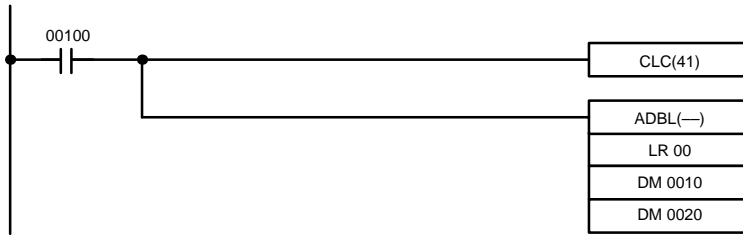
ADBL(—) can also be used to add signed binary data. The underflow and overflow flags (SR 25404 and SR 25405) indicate whether the result has exceeded the lower or upper limits of the 32-bit signed binary data range.

**Flags**

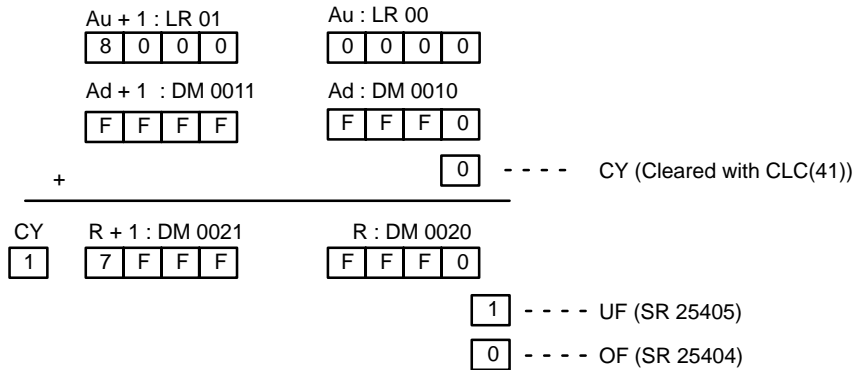
- ER:** Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)
- CY:** ON when the result is greater than FFFF FFFF.
- EQ:** ON when the result is 0.
- OF:** ON when the result exceeds +2,147,483,647 (7FFF FFFF).
- UF:** ON when the result is below -2,147,483,648 (8000 0000).

**Example**

The following example shows an eight-digit addition with CY (SR 25504) used to represent the status of the 9<sup>th</sup> digit. The status of the UF and OF flags indicate whether the result has exceeded the signed binary data range (–2,147,483,648 (8000 0000) to +2,147,483,647 (7FFF FFFF)).

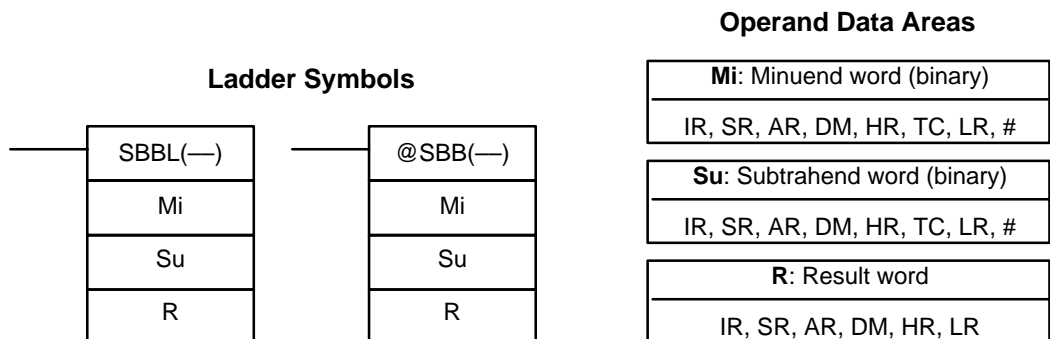


Address	Instruction	Operands
00000	LD	00100
00001	CLC(41)	
00002	ADBL(—)	
		LR 20
		DM 0010
		DM 0020



- Note**
1. For unsigned binary addition, CY indicates that the sum of the two values exceeds FFFF FFFF. (UF and OF can be ignored.)
  2. For signed binary addition, the UF flag indicates that the sum of the two values is below –2,147,483,648 (8000 0000). (CY can be ignored.)

**5-21-6 DOUBLE BINARY SUBTRACT – SBBL(—)**



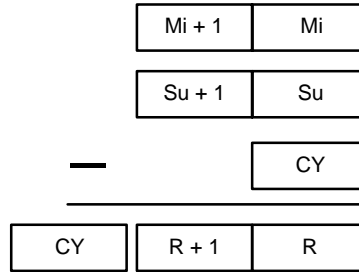
**Limitations**

This instruction is available in the **CQM1-CPU4□-E/-EV1 only**.  
 Mi and Mi+1 must be in the same data area, as must Su and Su+1, and R and R+1.  
 DM 6142 to DM 6655 cannot be used for R.



**Description**

When the execution condition is OFF, SBBL(—) is not executed. When the execution condition is ON, SBBL(—) subtracts CY and the eight-digit value in Su and Su+1 from the eight-digit value in Mi and Mi+1, and places the result in R and R+1. If the result is negative, CY is set and the 2's complement of the actual result is placed in R+1 and R. Use NEGL(—) to convert the 2's complement to the true result.



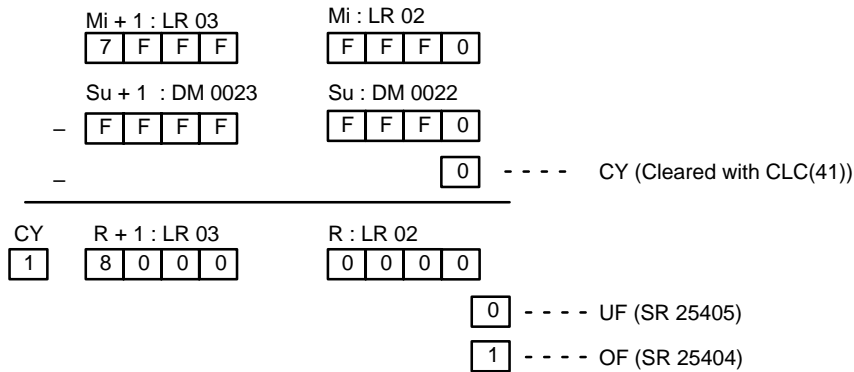
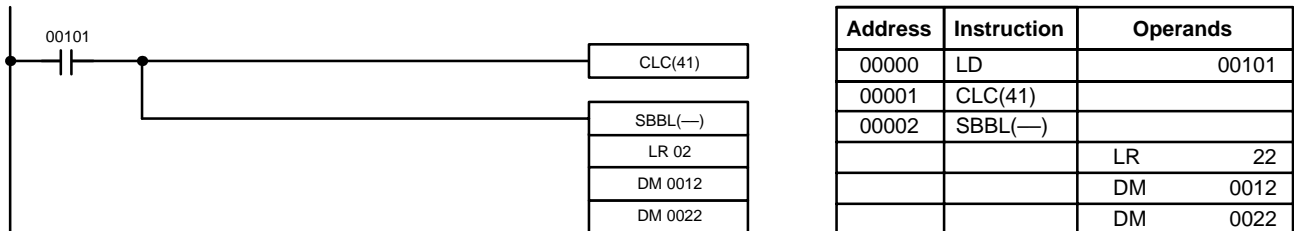
SBBL(—) can also be used to subtract signed binary data. The underflow and overflow flags (SR 25404 and SR 25405) indicate whether the result has exceeded the lower or upper limits of the 32-bit signed binary data range.

**Flags**

- ER:** Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)
- CY:** ON when the result is negative, i.e., when Mi is less than Su plus CY.
- EQ:** ON when the result is 0.
- OF:** ON when the result exceeds +2,147,483,647 (7FFF FFFF).
- UF:** ON when the result is below -2,147,483,648 (8000 0000).

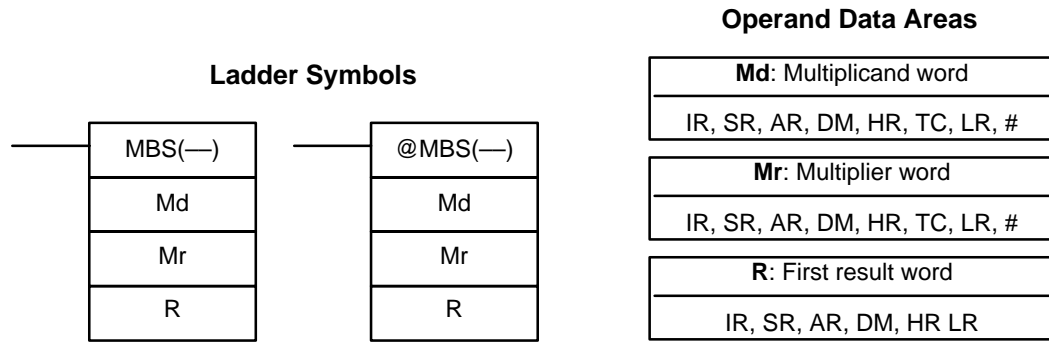
**Example**

The following example shows an eight-digit subtraction with CY (SR 25504) used to indicate a negative result (with unsigned data). The status of the UF and OF flags indicate whether the result has exceeded the signed binary data range (-2,147,483,648 (8000 0000) to +2,147,483,647 (7FFF FFFF)).



- Note**
1. For unsigned binary data, CY indicates that the result is negative. Take the 2's complement using NEGL(—) to obtain the absolute value of the true result. (UF and OF can be ignored.)
  2. For signed binary data, the OF flag indicates that the result exceeds +2,147,483,647 (7FFF FFFF). (CY can be ignored.)

### 5-21-7 SIGNED BINARY MULTIPLY – MBS(—)



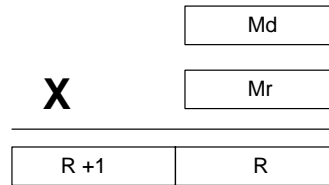
**Limitations**

This instruction is available in the **CQM1-CPU4□-E/-EV1 only**.  
DM 6143 to DM 6655 cannot be used for R.

**Description**

MBS(—) multiplies the signed binary content of two words and outputs the 8-digit signed binary result to R+1 and R. The rightmost four digits of the result are placed in R, and the leftmost four digits are placed in R+1.

**Note** Refer to 1-10 *Calculating with Signed Binary Data* for more details.

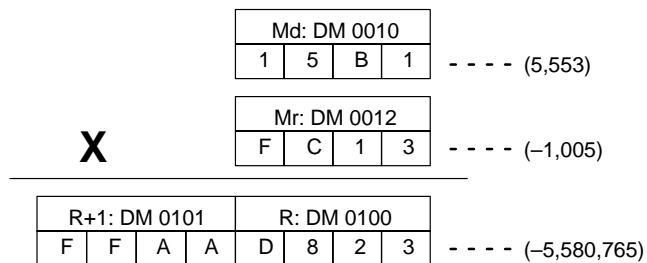
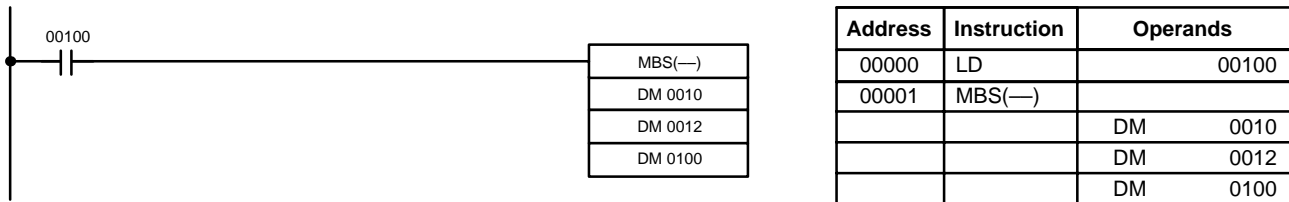


**Flags**

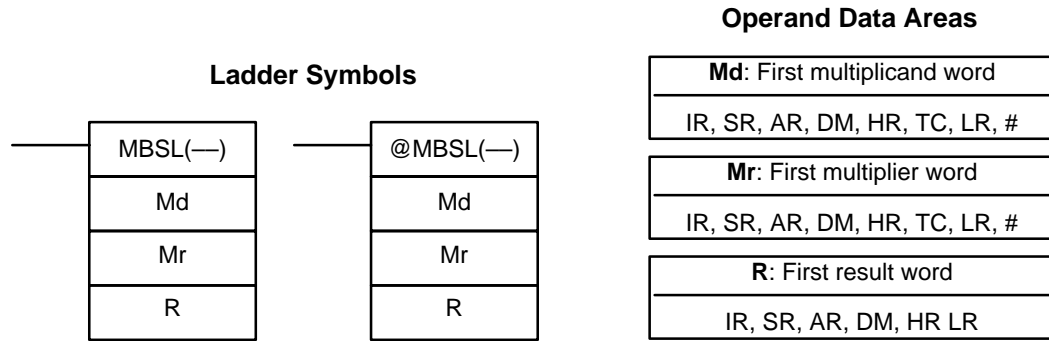
- ER:** Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)
- EQ:** ON when the result is 0000 0000, otherwise OFF.

**Example**

In the following example, MBS(—) is used to multiply the signed binary contents of DM 0010 with the signed binary contents of DM 0012 and output the result to DM 0100 and DM 0101.



### 5-21-8 DOUBLE SIGNED BINARY MULTIPLY – MBSL(—)



**Limitations**

This instruction is available in the **CQM1-CPU4□-E/-EV1 only**.  
 Md and Md+1 must be in the same data area, as must Mr and Mr+1.  
 R and R+3 must be in the same data area.  
 DM 6143 to DM 6655 cannot be used for R.

**Description**

MBSL(—) multiplies the 32-bit (8-digit) signed binary data in Md+1 and Md with the 32-bit signed binary data in Mr+1 and Mr, and outputs the 16-digit signed binary result to R+3 through R.

**Note** Refer to 1-10 *Calculating with Signed Binary Data* for more details.

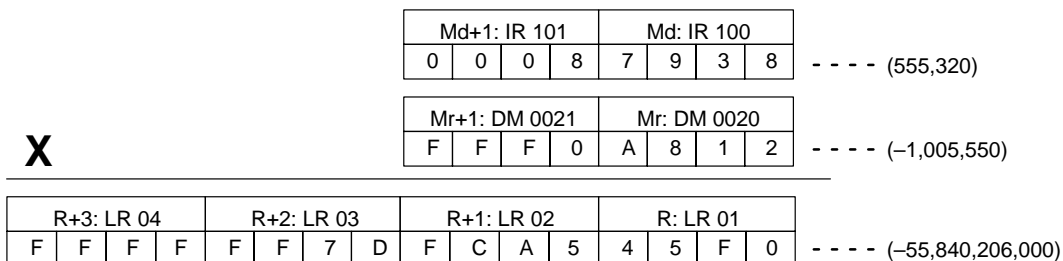
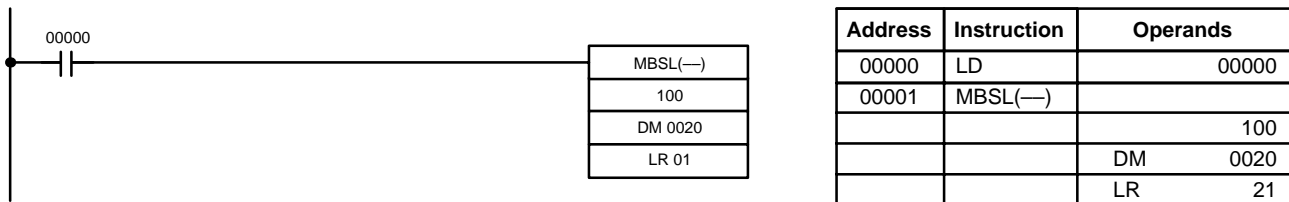


**Flags**

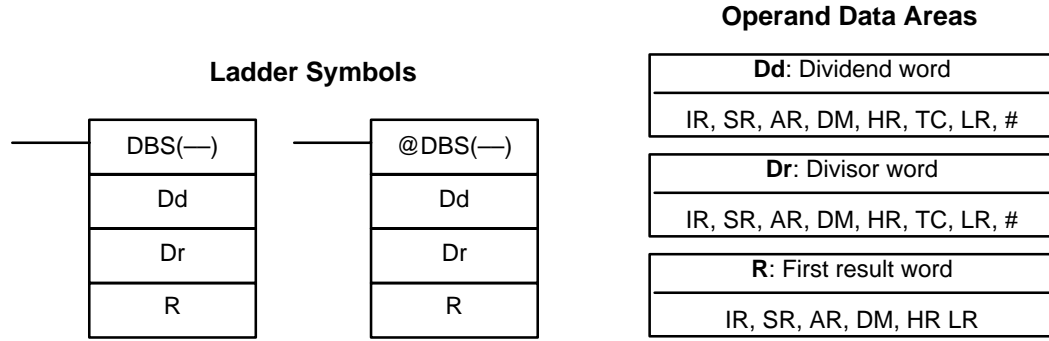
**ER:** Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)  
**EQ:** ON when the result is zero (content of R+3 through R all zeroes), otherwise OFF.

**Example**

In the following example, MBSL(—) is used to multiply the signed binary contents of IR 101 and IR 100 with the signed binary contents of DM 0021 and DM 0020 and output the result to LR 24 through LR 01.



### 5-21-9 SIGNED BINARY DIVIDE – DBS(—)



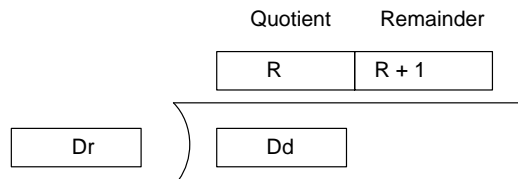
**Limitations**

This instruction is available in the **CQM1-CPU4□-E/-EV1 only**.  
DM 6143 to DM 6655 cannot be used for R.

**Description**

DBS(—) divides the signed binary content of Dd by the signed binary content of Dr, and outputs the 8-digit signed binary result to R+1 and R. The quotient is placed in R, and the remainder is placed in R+1.

**Note** Refer to *1-10 Calculating with Signed Binary Data* for more details.

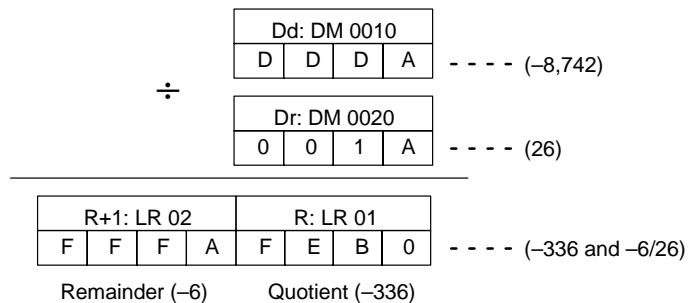
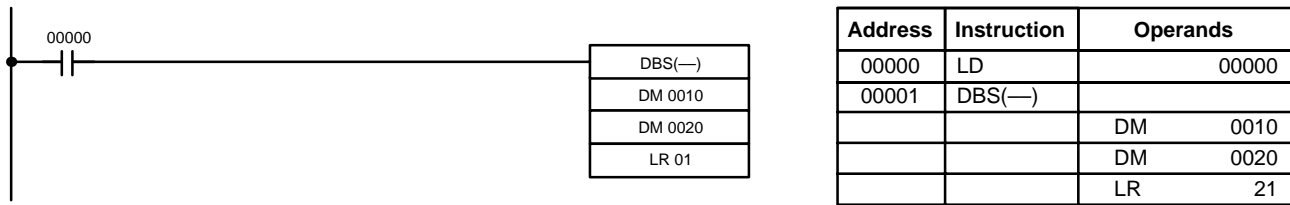


**Flags**

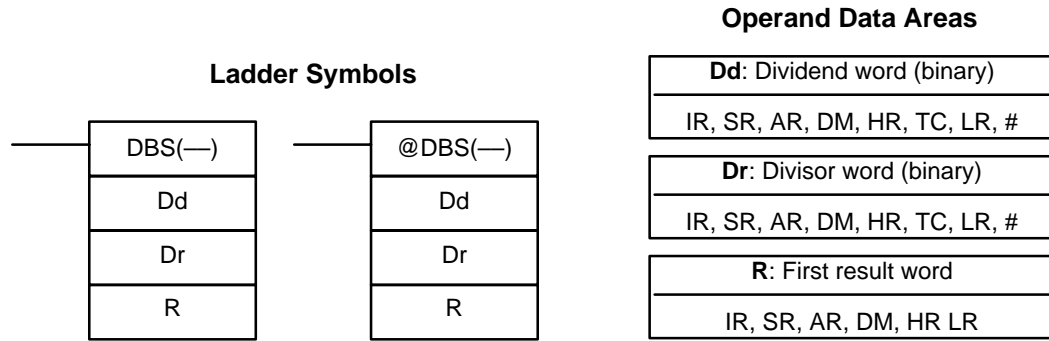
- ER:** Dr contains 0.  
Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)
- EQ:** ON when the content of R (the quotient) is 0000, otherwise OFF.

**Example**

In the following example, DBS(—) is used to divide the signed binary contents of DM 0010 with the signed binary contents of DM 0020 and output the result to LR 21 and LR 02.



### 5-21-10 DOUBLE SIGNED BINARY DIVIDE – DBSL(—)



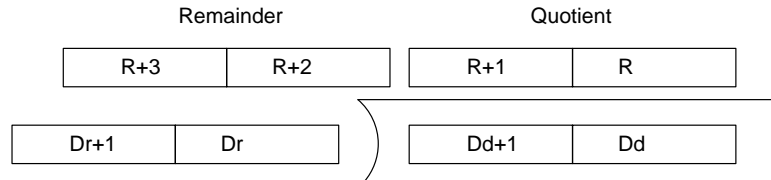
**Limitations**

This instruction is available in the **CQM1-CPU4□-E/-EV1 only**.  
 Dd and Dd+1 must be in the same data area, as must Dr and Dr+1.  
 R and R+3 must be in the same data area.  
 DM 6143 to DM 6655 cannot be used for R.

**Description**

DBS(—) divides the 32-bit (8-digit) signed binary data in Dd+1 and Dd by the 32-bit signed binary data in Dr+1 and Dr, and outputs the 16-digit signed binary result to R+3 through R. The quotient is placed in R+1 and R, and the remainder is placed in R+3 and R+2.

**Note** Refer to *1-10 Calculating with Signed Binary Data* for more details.

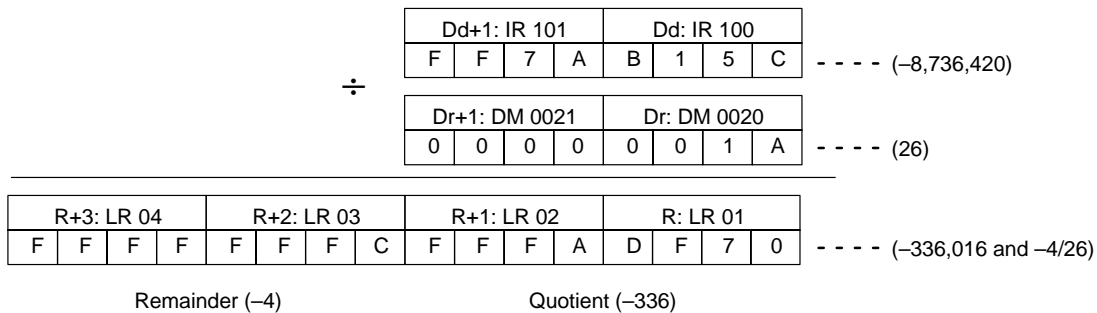
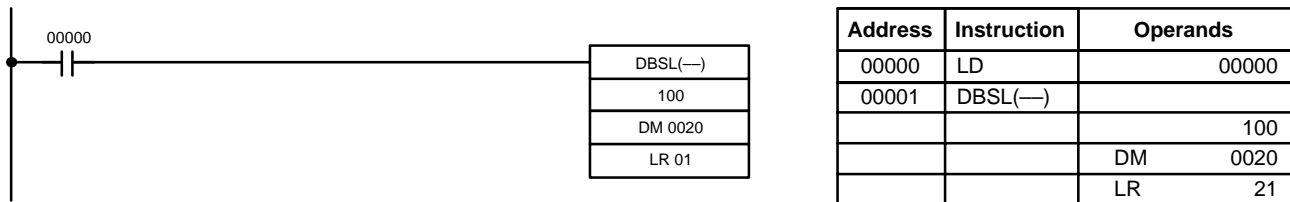


**Flags**

**ER:** Dr+1 and Dr contain 0.  
 Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)  
**EQ:** ON when the content of R+1 and R (the quotient) is 0, otherwise OFF.

**Example**

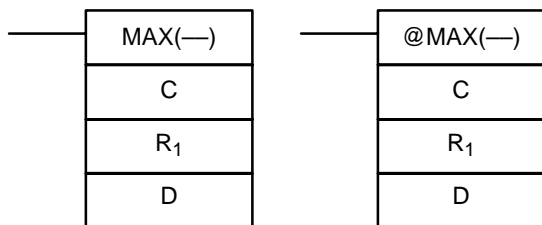
In the following example, DBSL(—) is used to divide the signed binary contents of IR 101 and IR 100 with the signed binary contents of DM 0021 and DM 0020 and output the result to LR 24 through LR 01.



## 5-22 Special Math Instructions

### 5-22-1 FIND MAXIMUM – MAX(—)

#### Ladder Symbols



#### Operand Data Areas

<b>C: Control data</b>
IR, SR, AR, DM, HR, TC, LR, #
<b>R<sub>1</sub>: First word in range</b>
IR, SR, AR, DM, HR, TC, LR
<b>D: Destination word</b>
IR, SR, AR, DM, HR, LR

#### Limitations

This instruction is available in the **CQM1 only**.  
 N must be BCD between 0001 to 9999.  
 R<sub>1</sub> and R<sub>1</sub>+N-1 must be in the same data area.  
 DM 6144 to DM 6655 cannot be used for D.

#### Description

When the execution condition is OFF, MAX(—) is not executed. When the execution condition is ON, MAX(—) searches the range of memory from R<sub>1</sub> to R<sub>1</sub>+N-1 for the address that contains the maximum value and outputs the maximum value to the destination word (D).

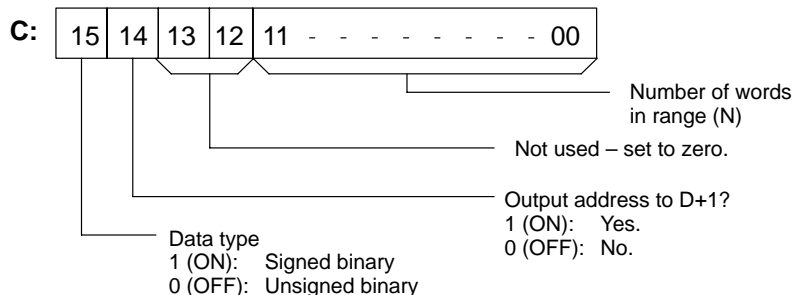
If bit 15 of C is ON, MAX(—) identifies the address of the word containing the maximum value in D+1. The address is identified differently for the DM area:

- 1, 2, 3...
1. For an address in the DM area, the word address is written to C+1. For example, if the address containing the maximum value is DM 0114, then #0114 is written in D+1.
  2. For an address in another data area, the number of addresses from the beginning of the search is written to D+1. For example, if the address containing the maximum value is IR 114 and the first word in the search range is IR 014, then #0100 is written in D+1.

If bit 14 of C is ON and more than one address contains the same maximum value, the position of the lowest of the addresses will be output to D+1. The position will be output as the DM address for the DM area, but as an absolute position relative to the first word in the range for all other areas.

The number of words within the range (N) is contained in the 3 rightmost digits of C, which must be BCD between 001 and 999.

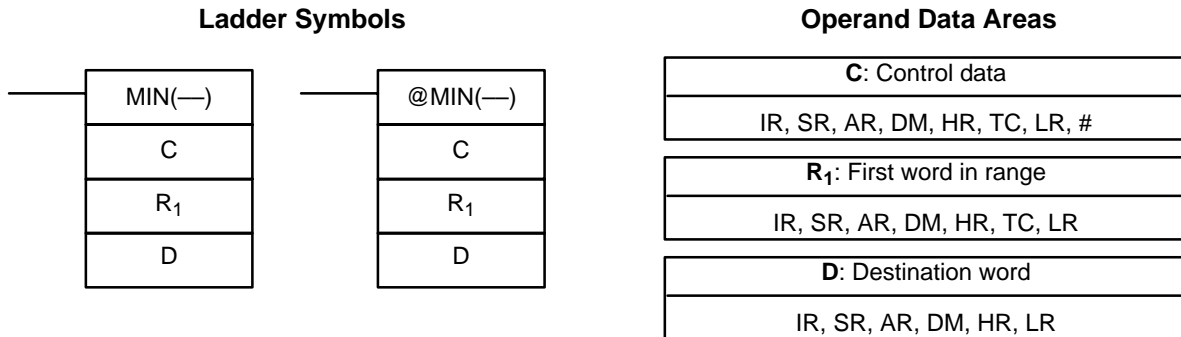
When bit 15 of C is OFF, data within the range is treated as unsigned binary and when it is ON the data is treated as signed binary.



**Caution** If bit 14 of C is ON, values above #8000 are treated as negative numbers, so the results will differ depending on the specified data type. Be sure that the correct data type is specified.

- Flags**
- ER:** Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)  
R<sub>1</sub> and R<sub>1</sub>+N-1 are not in the same data area.
  - EQ:** ON when the maximum value is #0000.

### 5-22-2 FIND MINIMUM – MIN(—)



- Limitations**
- This instruction is available in the **CQM1 only**.
  - N must be BCD between 0001 to 9999.
  - R<sub>1</sub> and R<sub>1</sub>+N-1 must be in the same data area.
  - DM 6144 to DM 6655 cannot be used for D.

**Description**

When the execution condition is OFF, MIN(—) is not executed. When the execution condition is ON, MIN(—) searches the range of memory from R<sub>1</sub> to R<sub>1</sub>+N-1 for the address that contains the minimum value and outputs the minimum value to the destination word (D).

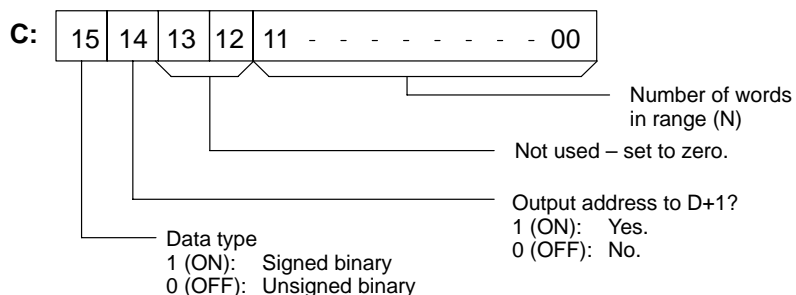
If bit 15 of C is ON, MIN(—) identifies the address of the word containing the minimum value in D+1. The address is identified differently for the DM area:

- 1, 2, 3...**
1. For an address in the DM area, the word address is written to C+1. For example, if the address containing the minimum value is DM 0114, then #0114 is written in D+1.
  2. For an address in another data area, the number of addresses from the beginning of the search is written to D+1. For example, if the address containing the minimum value is IR 114 and the first word in the search range is IR 014, then #0100 is written in D+1.

If bit 14 of C is ON and more than one address contains the same minimum value, the position of the lowest of the addresses will be output to D+1. The position will be output as the DM address for the DM area, but as an absolute position relative to the first word in the range for all other areas.

The number of words within the range (N) is contained in the 3 rightmost digits of C, which must be BCD between 001 and 999.

When bit 15 of C is OFF, data within the range is treated as unsigned binary and when it is ON the data is treated as signed binary.



**Caution** If bit 14 of C is ON, values above #8000 are treated as negative numbers, so the results will differ depending on the specified data type. Be sure that the correct data type is specified.

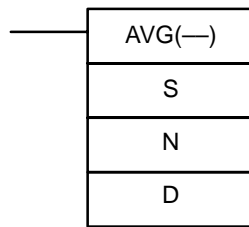
**Flags**

**ER:** Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)  
 R<sub>1</sub> and R<sub>1</sub>+N-1 are not in the same data area.

**EQ:** ON when the minimum value is #0000.

### 5-22-3 AVERAGE VALUE – AVG(—)

#### Ladder Symbols



#### Operand Data Areas

<b>S:</b> Source word
IR, SR, AR, DM, HR, TC, LR
<b>N:</b> Number of cycles
IR, SR, AR, DM, HR, TC, LR, #
<b>D:</b> First destination word
IR, SR, AR, DM, HR, LR

**Limitations**

This instruction is available in the **CQM1 only**.  
 S must be hexadecimal.  
 N must be BCD from #0001 to #0064.  
 D and D+N+1 must be in the same data area.  
 DM 6144 to DM 6655 cannot be used for S, N, or D to D+N+1.

**Description**

AVG(—) is used to calculate the average value of S over N cycles. When the execution condition is OFF, AVG(—) is not executed.

Each time that AVG(—) is executed, the content of S is stored in words D+2 to D+N+1. On the first execution, AVG(—) writes the content of S to D+2; on the second execution it writes the content of S to D+3, etc. On the N<sup>th</sup> execution, AVG(—) writes the content of S is stored in D+N+1, AVG(—) calculates the average value of the values stored in D+2 to D+N+1, and writes the average to D.

The following diagram shows the function of words D to D+N+1.

D	Average value (after N or more executions)
D+1	Used by the system.
D+2	Content of S from the 1 <sup>st</sup> execution of AVG(—)
D+3	Content of S from the 2 <sup>nd</sup> execution of AVG(—)
⋮	⋮
D+N+1	Content of S from the N <sup>th</sup> execution of AVG(—)

**Precautions**

The average value is calculated in binary. Be sure that the content of S is in binary.  
 N must be BCD from #0001 to #0064. If the content of N ≥ #0065, AVG(—) will operate with N=64.  
 The average value will be rounded off to the nearest integer value. (0.5 is rounded up to 1.)  
 Leave the contents of D+1 set to #0000 after the first execution of AVG(—).

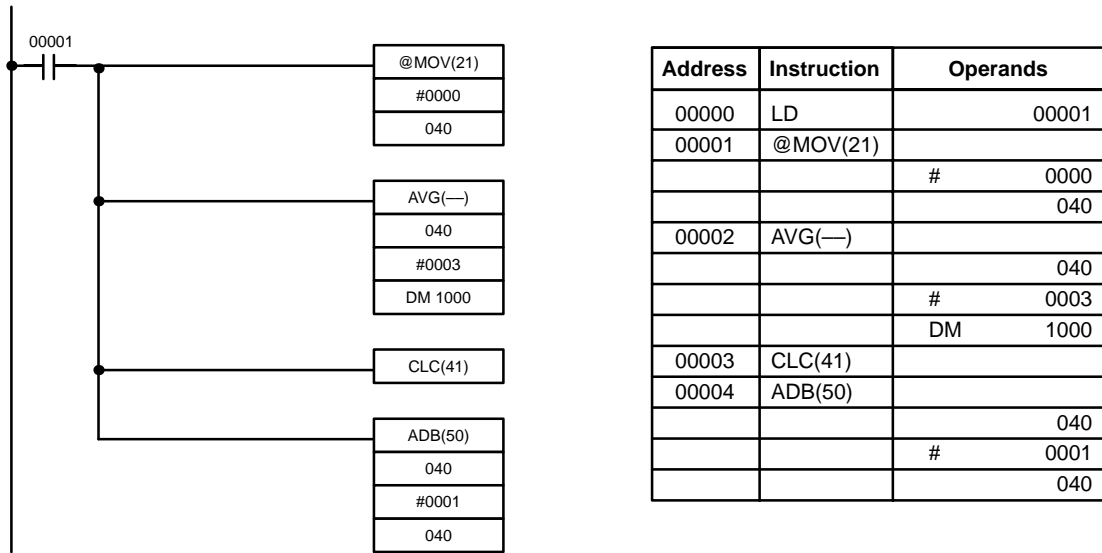


**Flags**

**ER:** Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)  
 One or more operands have been set incorrectly.  
 D and D+N+1 are not in the same data area.

**Example**

In the following example, the content of IR 040 is set to #0000 and then incremented by 1 each cycle. For the first two cycles, AVG(—) moves the content of IR 040 to DM 1002 and DM 1003. On the third and later cycles AVG(—) calculates the average value of the contents of DM 1002 to DM 1004 and writes that average value to DM 1000.



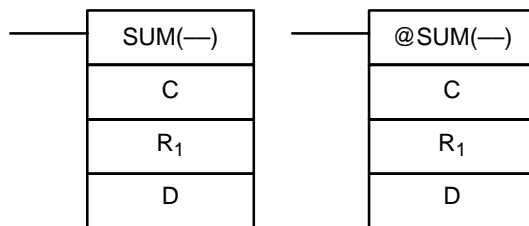
	1 <sup>st</sup> cycle	2 <sup>nd</sup> cycle	3 <sup>rd</sup> cycle	4 <sup>th</sup> cycle
IR 040	0000	0001	0002	0003

	1 <sup>st</sup> cycle	2 <sup>nd</sup> cycle	3 <sup>rd</sup> cycle	4 <sup>th</sup> cycle
DM 1000	0000	0001	0001	0002
DM 1001				
DM 1002	0000	0000	0000	0003
DM 1003	---	0001	0001	0001
DM 1004	---	---	0002	0002

Average Used by the system. Previous values of IR 40

**5-22-4 SUM – SUM(—)**

**Ladder Symbols**



**Operand Data Areas**

<b>C:</b> Control data
IR, SR, AR, DM, HR, LR, #
<b>R<sub>1</sub>:</b> First word in range
IR, SR, AR, DM, HR, TC, LR
<b>D:</b> First destination word
IR, SR, AR, DM, HR, LR

**Limitations**

This instruction is available in the **CQM1** only.  
 The 3 rightmost digits of C must be BCD between 001 and 999.

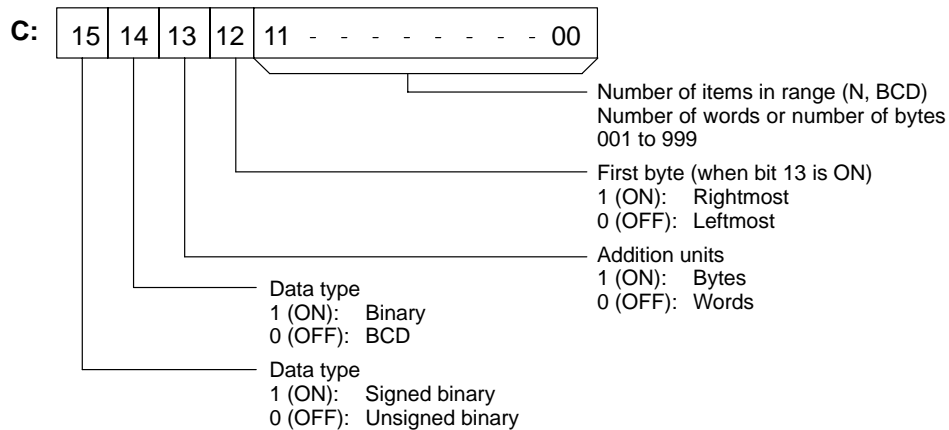
DM 6143 to DM 6655 cannot be used for D.

If bit 14 of C is OFF (setting for BCD data), all data within the range  $R_1$  to  $R_1+N-1$  must be BCD.

**Description**

When the execution condition is OFF, SUM(—) is not executed. When the execution condition is ON, SUM(—) adds either the contents of words  $R_1$  to  $R_1+N-1$  or the bytes in words  $R_1$  to  $R_1+N/2-1$  and outputs that value to the destination words (D and D+1). The data can be summed as binary or BCD and will be output in the same form. Binary data can be either signed or unsigned.

The function of bits in C are shown in the following diagram and explained in more detail below.



**Number of Items in Range**

The number of items within the range (N) is contained in the 3 rightmost digits of C, which must be BCD between 001 and 999. This number will indicate the number of words or the number of bytes depending the items being summed.

**Addition Units**

Words will be added if bit 13 is OFF and bytes will be added if bit 13 is ON. If bytes are specified, the range can begin with the leftmost or rightmost byte of  $R_1$ . The leftmost byte of  $R_1$  will not be added if bit 12 is ON.

	MSB	LSB
$R_1$	1	2
$R_1+1$	3	4
$R_1+2$	5	6
$R_1+3$	7	8
⋮	⋮	⋮
⋮	⋮	⋮

The bytes will be added in this order when bit 12 is OFF: 1+2+3+4....

The bytes will be added in this order when bit 12 is ON: 2+3+4....

**Data Type**

Data within the range is treated as unsigned binary when bit 14 of C is ON and bit 15 is OFF, and it is treated as signed binary when both bits 14 and 15 are ON.

Data within the range is treated as BCD when bit 14 of C is OFF, regardless of the status of bit 15.

**Flags**

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

$R_1$  and  $R_1+N-1$  are not in the same data area.

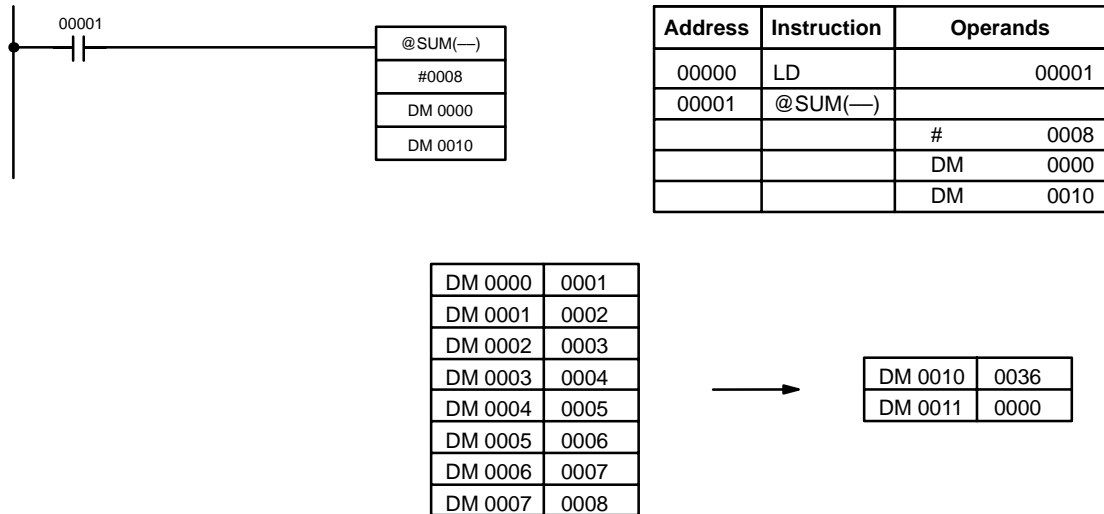
The number of items in C is not BCD between 001 and 999.

The data being summed in not BCD when BCD was designated.

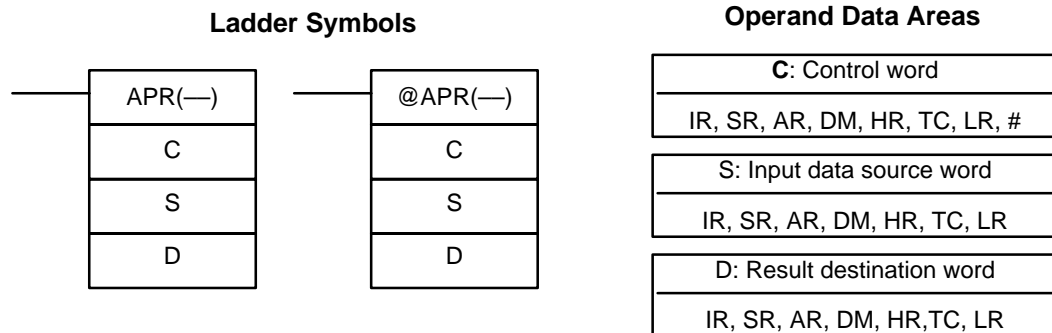
**EQ:** ON when the result is zero.

**Example**

In the following example, the BCD contents of the 8 words from DM 0000 to DM 0007 are added when IR 00001 is ON and the result is written to DM 0010 and DM 0011.



**5-22-5 ARITHMETIC PROCESS – APR(—)**



**Limitations**

This instruction is available in the **CQM1 only**.  
 For trigonometric functions S must be BCD from 0000 to 0900 (0° ≤ ? ≤ 90°).  
 DM 6144 to DM 6655 cannot be used for D.

**Description**

When the execution condition is OFF, APR(—) is not executed. When the execution condition is ON, the operation of APR(—) depends on the control word C.  
 If C is #0000 or #0001, APR(—) computes sin(?) or cos(?)\*. The BCD value of S specifies ? in tenths of degrees.  
 If C is an address, APR(—) computes f(x) of the function entered in advance beginning at word C. The function is a series of line segments (which can approximate a curve) determined by the operator. The BCD or hexadecimal value of S specifies x.

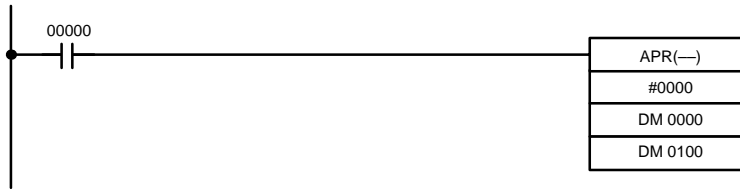
**Flags**

- ER:** Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)  
 For trigonometric functions, x > 0900. (x is the content of S.)  
 A constant other than #0000 or #0001 was designated for C.  
 The linear approximation data is not readable.
- EQ:** The result is 0000.

**Examples**

**Sine Function**

The following example demonstrates the use of the APR(—) sine function to calculate the sine of 30°. The sine function is specified when C is #0000.



Address	Instruction	Operands
00000	LD	00000
00001	APR(—)	
		# 0000
		DM 0000
		DM 0100

Input data, x

S: DM 0000			
0	10 <sup>1</sup>	10 <sup>0</sup>	10 <sup>-1</sup>
0	3	0	0

Enter input data not exceeding #0900 in BCD.

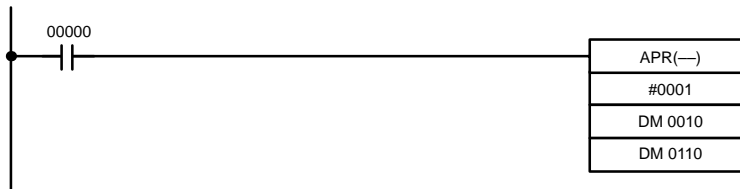
Result data

D: DM 0100			
10 <sup>-1</sup>	10 <sup>-2</sup>	10 <sup>-3</sup>	10 <sup>-4</sup>
5	0	0	0

Result data has four significant digits, fifth and higher digits are ignored. The result for sin(90) will be 0.9999, not 1.

**Cosine Function**

The following example demonstrates the use of the APR(—) cosine function to calculate the cosine of 30°. The cosine function is specified when C is #0001.



Address	Instruction	Operands
00000	LD	00000
00001	APR(—)	
		# 0001
		DM 0010
		DM 0110

Input data, x

S: DM 0010			
0	10 <sup>1</sup>	10 <sup>0</sup>	10 <sup>-1</sup>
0	3	0	0

Enter input data not exceeding #0900 in BCD.

Result data

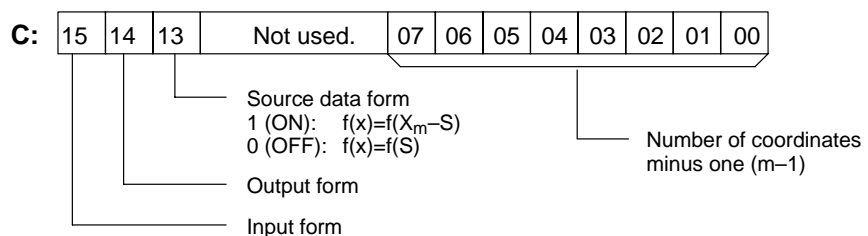
D: DM 0110			
10 <sup>-1</sup>	10 <sup>-2</sup>	10 <sup>-3</sup>	10 <sup>-4</sup>
8	6	6	0

Result data has four significant digits, fifth and higher digits are ignored. The result for cos(0) will be 0.9999, not 1.

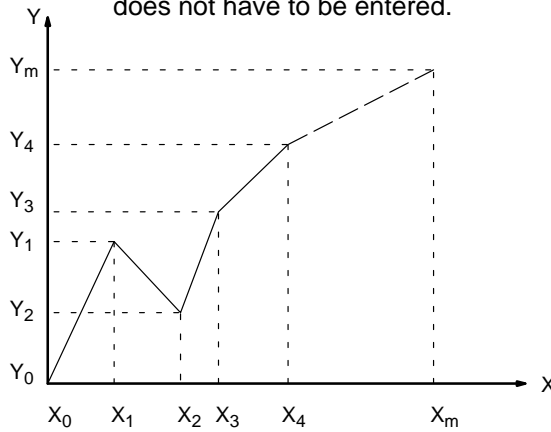
**Linear Approximation**

APR(—) linear approximation is specified when C is a memory address. Word C is the first word of the continuous block of memory containing the linear approximation data.

The content of word C specifies the number of line segments in the approximation, and whether the input and output are in BCD or BIN form. Bits 00 to 07 contain the number of line segments less 1, m-1, as binary data. Bits 14 and 15 determine, respectively, the output and input forms: 0 specifies BCD and 1 specifies BIN.

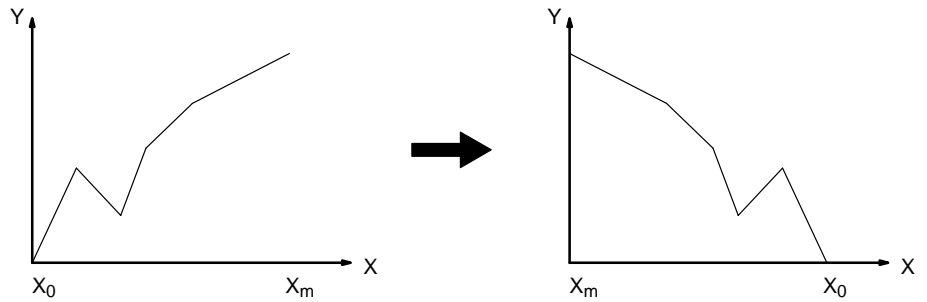


Enter the coordinates of the m+1 end-points, which define the m line segments, as shown in the following table. Enter all coordinates in BIN form. Always enter the coordinates from the lowest X value (X<sub>1</sub>) to the highest (X<sub>m</sub>). X<sub>0</sub> is 0000, and does not have to be entered.

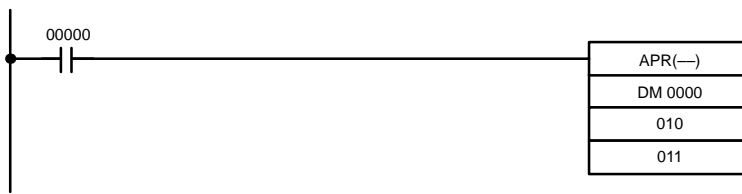


Word	Coordinate
C+1	X <sub>m</sub> (max. X value)
C+2	Y <sub>0</sub>
C+3	X <sub>1</sub>
C+4	Y <sub>1</sub>
C+5	X <sub>2</sub>
C+6	Y <sub>2</sub>
↓	↓
C+(2m+1)	X <sub>m</sub>
C+(2m+2)	Y <sub>m</sub>

If bit 13 of C is set to 1, the graph will be reflected from left to right, as shown in the following diagram.



The following example demonstrates the construction of a linear approximation with 12 line segments. The block of data is continuous, as it must be, from DM 0000 to DM 0026 (C to C + (2 × 12 + 2)). The input data is taken from IR 010, and the result is output to IR 011.

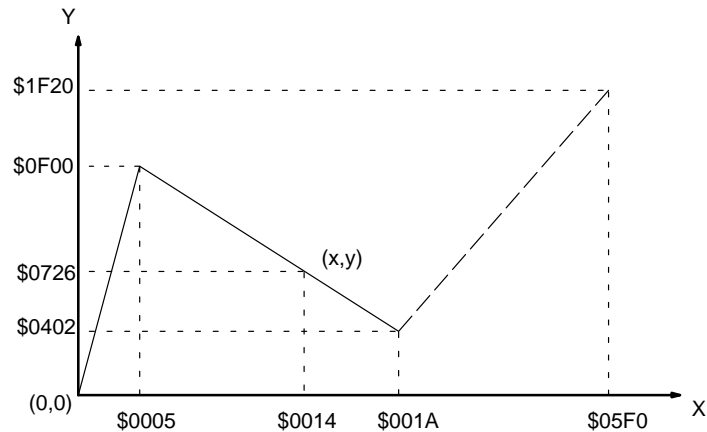


Address	Instruction	Operands
00000	LD	00000
00001	APR(--)	
		DM 0000
		010
		011

Content	Coordinate	Bit 15	Bit 00
DM 0000	\$C00B	1	1
DM 0001	\$05F0 X <sub>12</sub>	0	0
DM 0002	\$0000 Y <sub>0</sub>	0	0
DM 0003	\$0005 X <sub>1</sub>	0	0
DM 0004	\$0F00 Y <sub>1</sub>	0	0
DM 0005	\$001A X <sub>2</sub>	0	0
DM 0006	\$0402 Y <sub>2</sub>	0	0
↓	↓		
DM 0025	\$05F0 X <sub>12</sub>	0	0
DM 0026	\$1F20 Y <sub>12</sub>	0	0

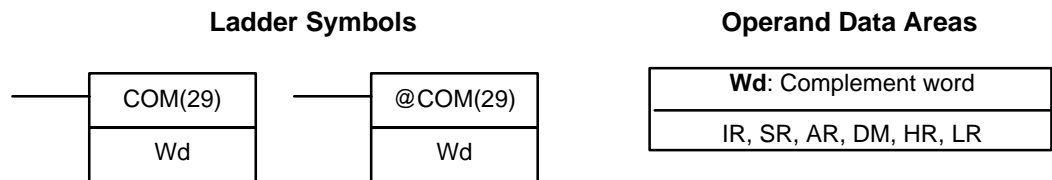
(Output and input both BIN)                      (m-1 = 11: 12 line segments)

In this case, the input data word, IR 010, contains #0014, and f(0014) = #0726 is output to R, IR 011.



## 5-23 Logic Instructions

### 5-23-1 COMPLEMENT – COM(29)



**Limitations**

DM 6144 to DM 6655 cannot be used for Wd.

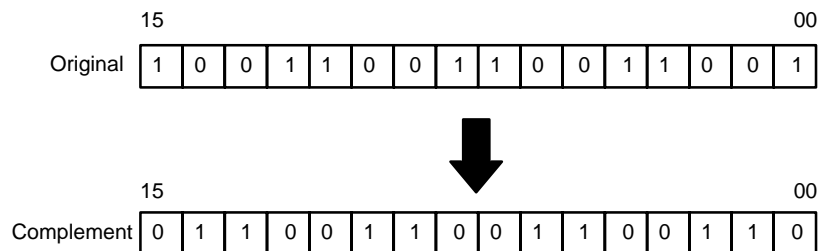
**Description**

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.

**Precautions**

The complement of Wd will be calculated every cycle if the undifferentiated form of COM(29) is used. Use the differentiated form (@COM(29)) or combine COM(29) with DIFU(13) or DIFD(14) to calculate the complement just once.

**Example**

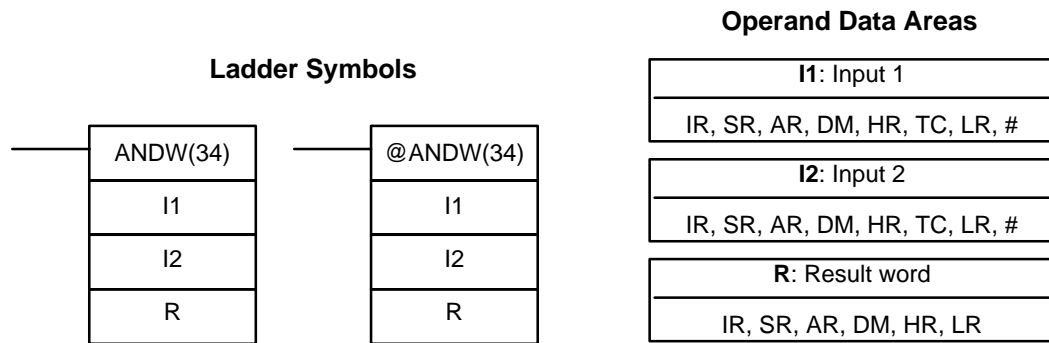


**Flags**

**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.

### 5-23-2 LOGICAL AND – ANDW(34)



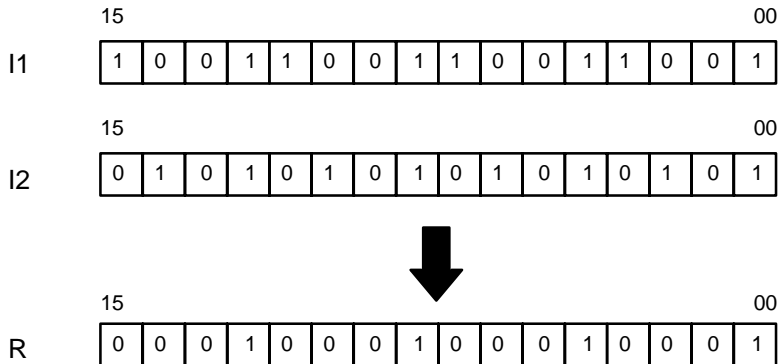
**Limitations**

DM 6144 to DM 6655 cannot be used for R.

**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**

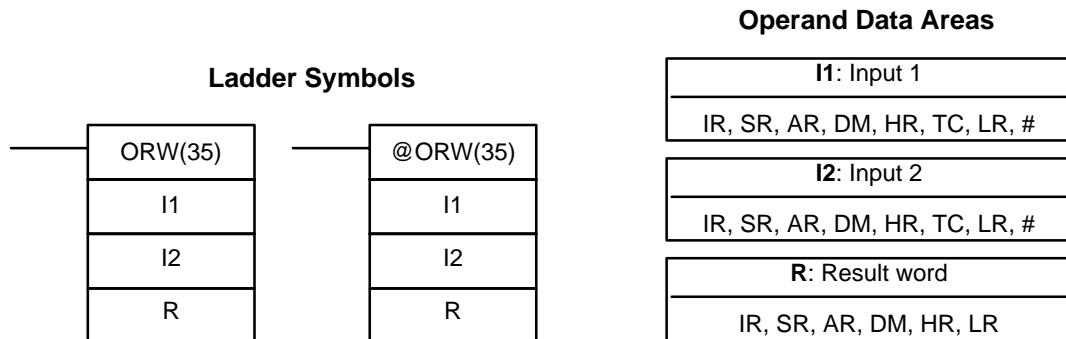


**Flags**

**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.

### 5-23-3 LOGICAL OR – ORW(35)



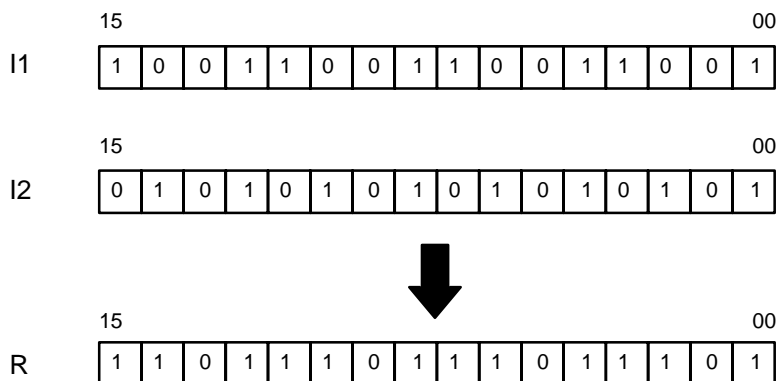
**Limitations**

DM 6144 to DM 6655 cannot be used for R.

**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.

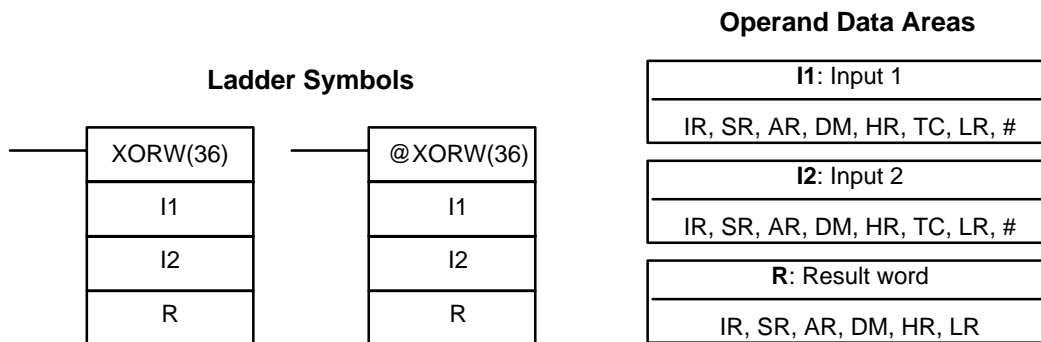
**Example**



**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.

**5-23-4 EXCLUSIVE OR – XORW(36)**



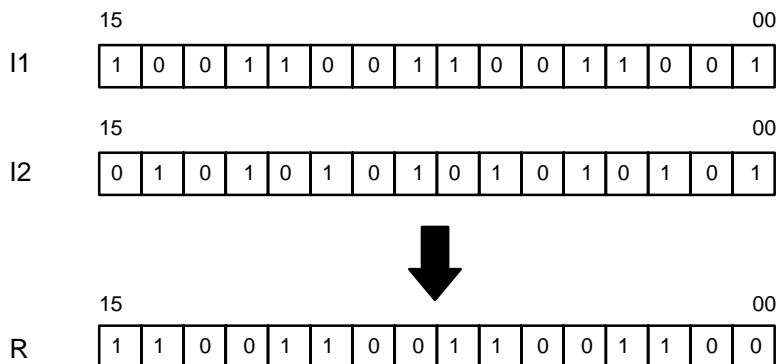
**Limitations**

DM 6144 to DM 6655 cannot be used for R.

**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**

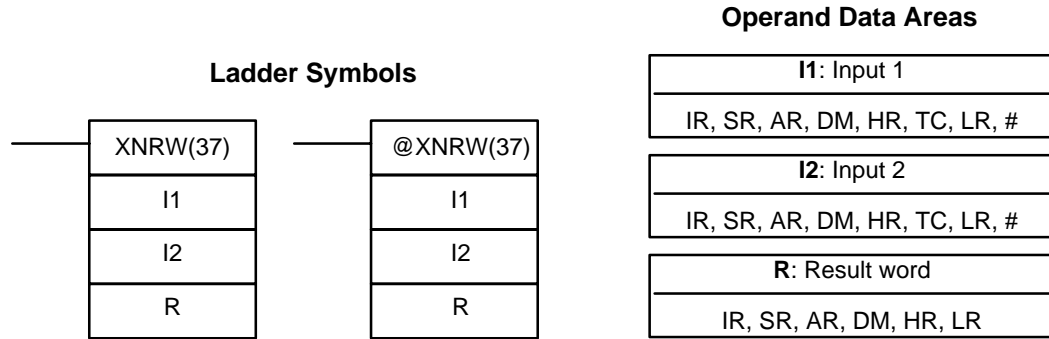


**Flags**

- 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.



### 5-23-5 EXCLUSIVE NOR – XNRW(37)

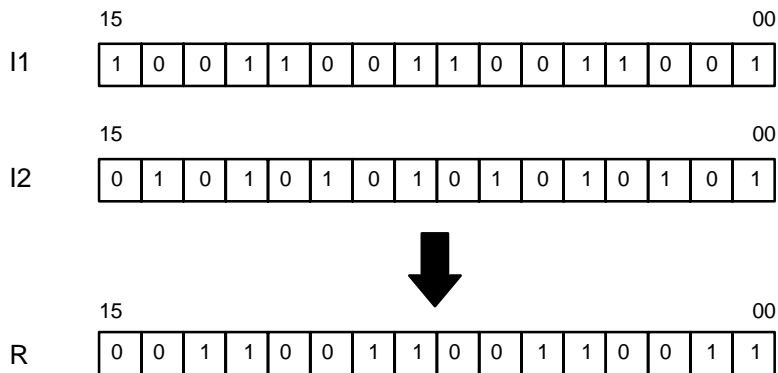


**Limitations**

DM 6144 to DM 6655 cannot be used for R.

**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.



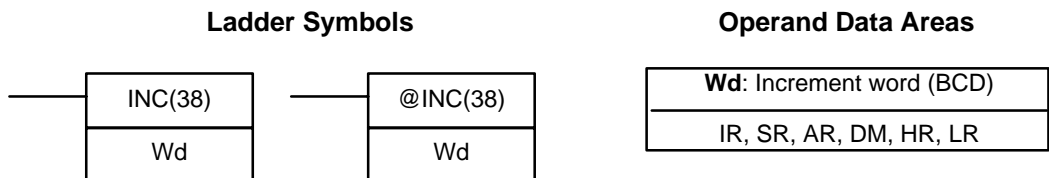
**Flags**

**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.

## 5-24 Increment/Decrement Instructions

### 5-24-1 BCD INCREMENT – INC(38)



**Limitations**

DM 6144 to DM 6655 cannot be used for Wd.

**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).

**Precautions**

The content of Wd will be incremented every cycle if the undifferentiated form of INC(38) is used. Use the differentiated form (@INC(38)) or combine INC(38) with DIFU(13) or DIFD(14) to increment Wd just once.

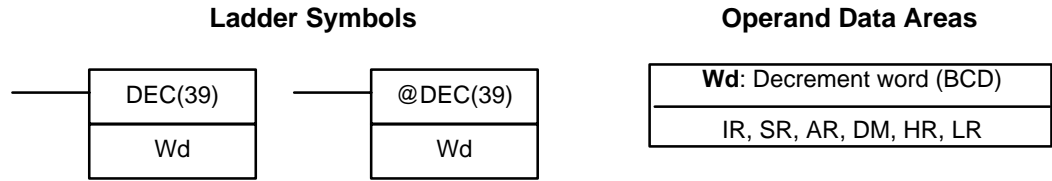
**Flags**

**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.

## 5-24-2 BCD DECREMENT – DEC(39)

**Limitations**

DM 6144 to DM 6655 cannot be used for Wd.

**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.

**Precautions**

The content of Wd will be decremented every cycle if the undifferentiated form of DEC(39) is used. Use the differentiated form (@DEC(39)) or combine DEC(39) with DIFU(13) or DIFD(14) to decrement Wd just once.

**Flags**

**ER:** Wd is not BCD.

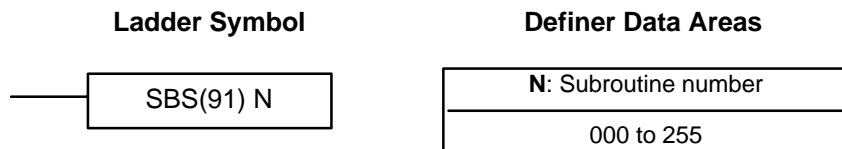
Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)

**EQ:** ON when the decremented result is 0.

## 5-25 Subroutine Instructions

Subroutines break large control tasks into smaller ones and enable you to reuse a given set of instructions. When the main program calls a subroutine, control is transferred to the subroutine and the subroutine instructions are executed. The instructions within a subroutine are written in the same way as main program code. When all the subroutine instructions have been executed, control returns to the main program to the point just after the point from which the subroutine was entered (unless otherwise specified in the subroutine).

### 5-25-1 SUBROUTINE ENTER – SBS(91)

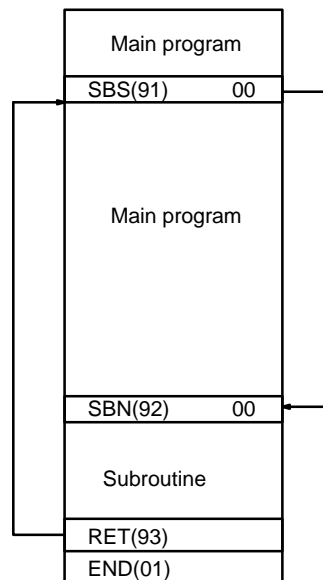


**Limitations**

The CQM1-CPU11/21-E supports only subroutine numbers 000 through 127. The CPM1/CPM1A/SRM1 support only subroutine numbers 000 through 049.

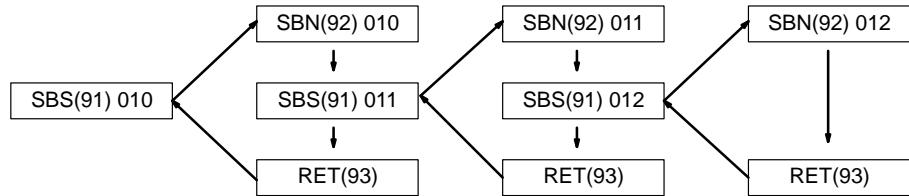
**Description**

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.

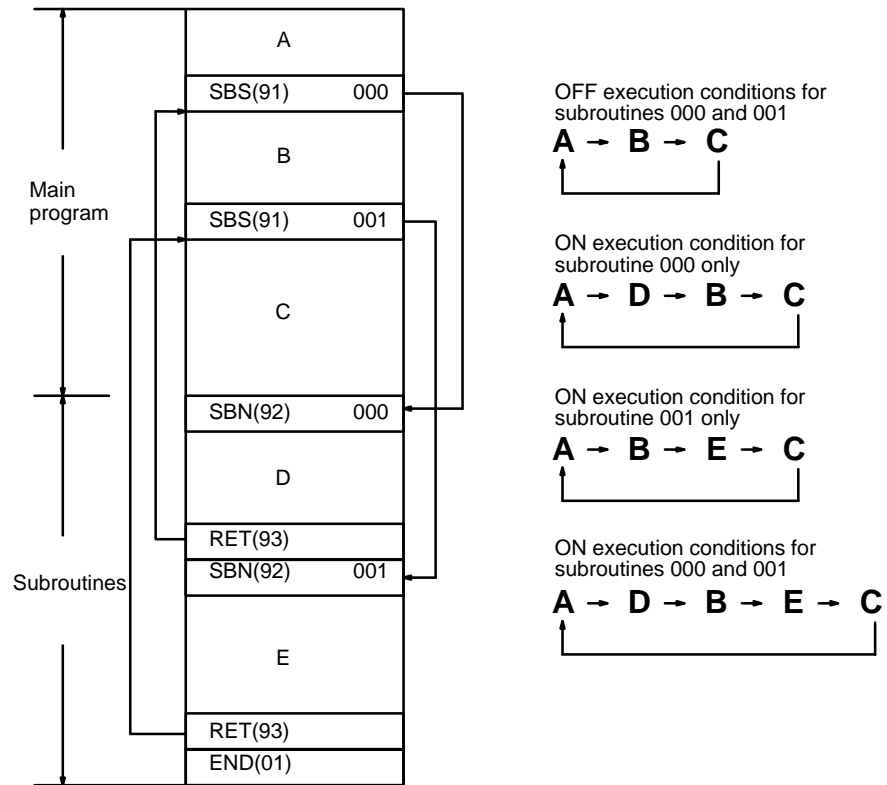


SBS(91) may be used as many times as desired in the program, i.e., the same subroutine may be called from different places in the program).

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) 000 cannot be programmed within the subroutine defined with SBN(92) 000). The following diagram illustrates two levels of nesting.



The following diagram illustrates program execution flow for various execution conditions for two SBS(91).

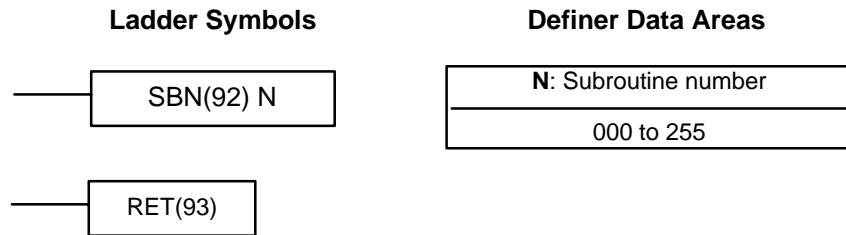


**Flags**

- ER:** A subroutine does not exist for the specified subroutine number.
- A subroutine has called itself.
- An active subroutine has been called.

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

## 5-25-2 SUBROUTINE DEFINE and RETURN – SBN(92)/RET(93)



### Limitations

The CQM1-CPU11/21-E support only subroutine numbers 000 through 127. The CPM1/CPM1A/SRM1 PCs support only subroutine numbers 000 through 049.

Each subroutine number can be used in SBN(92) once only.

### Description

SBN(92) is used to mark the beginning of a subroutine program; RET(93) is used to mark the end. Each subroutine is identified with a subroutine number, N, that is programmed as a definer for SBN(92). This same subroutine number is used in any SBS(91) that calls the subroutine (see 5-25-1 SUBROUTINE ENTER – SBS(91)). No subroutine number is required with RET(93).

All subroutines must be programmed at the end of the main program. When one or more subroutines have been programmed, the main program will be executed up to the first SBN(92) before returning to address 00000 for the next cycle. Subroutines will not be executed unless called by SBS(91).

END(01) must be placed at the end of the last subroutine program, i.e., after the last RET(93). It is not required at any other point in the program.

### Precautions

If SBN(92) is mistakenly placed in the main program, it will inhibit program execution past that point, i.e., program execution will return to the beginning when SBN(92) is encountered.


If either DIFU(13) or DIFU(14) is placed within a subroutine, the operand bit will not be turned OFF until the next time the subroutine is executed, i.e., the operand bit may stay ON longer than one cycle.

### Flags

There are no flags directly affected by these instructions.

## 5-26 Special Instructions

### 5-26-1 TRACE MEMORY SAMPLING – TRSM(45)

 **Caution** This instruction is not supported by CQM1-CPU11-E/21-E, the CPM1/CPM1A/SRM1 PCs.

Data tracing can be used to facilitate debugging programs. To set up and use data tracing it is necessary to have a host computer running SSS; no data tracing is possible from a Programming Console. Data tracing is described in detail in the *SSS Operation Manual: C-series PCs*. This section shows the ladder symbol for TRSM(45) and gives an example program.

#### Ladder Symbol



### Description

TRSM(45) is used in the program to mark locations where specified data is to be stored in Trace Memory. Up to 12 bits and up to 3 words may be designated for tracing. (Refer to the *SSS Operation Manual: C-series PCs* for details.)

TRSM(45) is not controlled by an execution condition, but rather by two bits in the AR area: AR 2515 and AR 2514. AR 2515 is the Sampling Start bit. This bit is turned ON to start the sampling processes for tracing. The Sampling Start bit must not be turned ON from the program, i.e., it must be turned ON only from the peripheral device. AR 2514 is the Trace Start bit. When it is set, the specified data is recorded in Trace Memory. The Trace Start bit can be set either from the program or from the Programming Device. A positive or negative delay can also be set to alter the actual point from which tracing will begin.

Data can be recorded in any of three ways. TRSM(45) can be placed at one or more locations in the program to indicate where the specified data is to be traced. If TRSM(45) is not used, the specified data will be traced when END(01) is executed. The third method involves setting a timer interval from the peripheral devices so that the specified data will be tracing at a regular interval independent of the cycle time. (Refer to the *SSS Operation Manual: C-series PCs*.)

TRSM(45) can be incorporated anywhere in a program, any number of times. The data in the trace memory can then be monitored via a Programming Console, host computer, etc.

### AR Control Bits and Flags

The following control bits and flags are used during data tracing. The Tracing Flag will be ON during tracing operations. The Trace Completed Flag will turn ON when enough data has been traced to fill Trace Memory.

Flag	Function
AR 2515	Sampling Start Bit*
AR 2514	Trace Start Bit
AR 2513	Tracing Flag
AR 2512	Trace Completed Flag

**Note** \*Do not change the status of AR 2515 from the program.

### Precautions

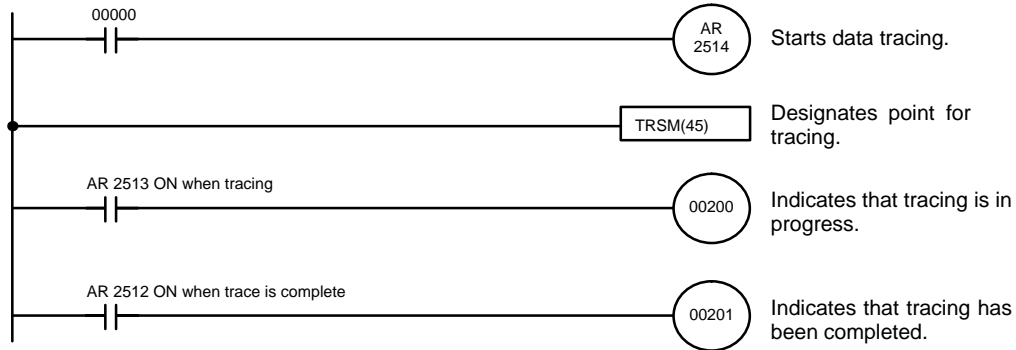
If TRSM(45) occurs TRSM(45) will not be executed within a JMP(08) – JME(09) block when the jump condition is OFF.

### Example

The following example shows the basic program and operation for data tracing. Force set the Sampling Start Bit (AR 2515) to begin sampling. The Sampling Start Bit must not be turned ON from the program. The data is read and stored into trace memory.

When IR 00000 is ON, the Trace Start Bit (AR 2514) is also turned ON, and the CPU Unit looks at the delay and marks the trace memory accordingly. This can mean that some of the samples already made will be recorded as the trace memory (negative delay), or that more samples will be made before they are recorded (positive delay).

The sampled data is written to trace memory, jumping to the beginning of the memory area once the end has been reached and continuing up to the start marker. This might mean that previously recorded data (i.e., data from this sample that falls before the start marker) is overwritten (this is especially true if the delay is positive). The negative delay cannot be such that the required data was executed before sampling was started.

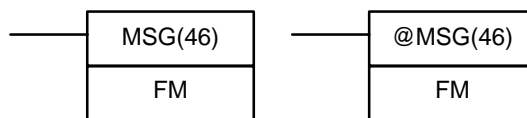


Address	Instruction	Operands
00000	LD	0000
00001	OUT	AR 2514
00002	TRSM(45)	
00003	LD	AR 2513

Address	Instruction	Operands
00004	OUT	00200
00005	LD	AR 2512
00006	OUT	00201

### 5-26-2 MESSAGE DISPLAY – MSG(46)

#### Ladder Symbols



#### Operand Data Areas

<b>FM:</b> First message word
IR, SR, AR, DM, HR, LR

#### Limitations

DM 6649 to DM 6655 cannot be used for FM.

#### Description

When executed with an ON execution condition, MSG(46) reads eight words of extended ASCII code from FM to FM+7 and displays the message on the Programming Console. The displayed message can be up to 16 characters long, i.e., each ASCII character code requires eight bits (two digits). Refer to *Appendix H* for the 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 > HR > AR > TC > DM

In handling messages from the same area, those with the lowest address values have higher priority.

In handling indirectly addressed messages (i.e. \*DM), those with the lowest final DM addresses have higher priority.

**Clearing Messages**

To clear a message, execute FAL(06) 00 or clear it via a Programming Console or the SSS.

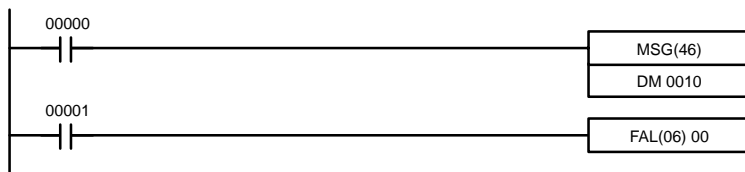
If the message data changes while the message is being displayed, the display will also change.

**Flags**

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

**Example**

The following example shows the display that would be produced for the instruction and data given when 00000 was ON. If 00001 goes ON, a message will be cleared.



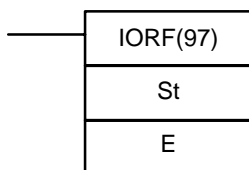
Address	Instruction	Operands
00000	LD	00000
00001	MSG(46)	
		DM 0010
00002	LD	00001
00003	FAL(06)	00

DM contents					ASCII equivalent	
DM 0010	4	1	4	2	A	B
DM 0011	4	3	4	4	C	D
DM 0012	4	5	4	6	E	F
DM 0013	4	7	4	8	G	H
DM 0014	4	9	4	A	I	J
DM 0015	4	B	4	C	K	L
DM 0016	4	D	4	E	M	N
DM 0017	4	F	5	0	O	P

```
MSG
ABCDEFGHIJKLMNOF
```

**5-26-3 I/O REFRESH – IORF(97)**

**Ladder Symbol**



**Operand Data Areas**

<b>St:</b> Starting word
IR 000 to IR 115
<b>E:</b> End word
IR 000 to IR 115

**Note** This instruction is not supported by SRM1 PCs.

**Limitations**

St must be less than or equal to E.

**Description**

To refresh I/O words, specify the first (St) and last (E) I/O words to be refreshed. When the execution condition for IORF(97) is ON, all words between St and E

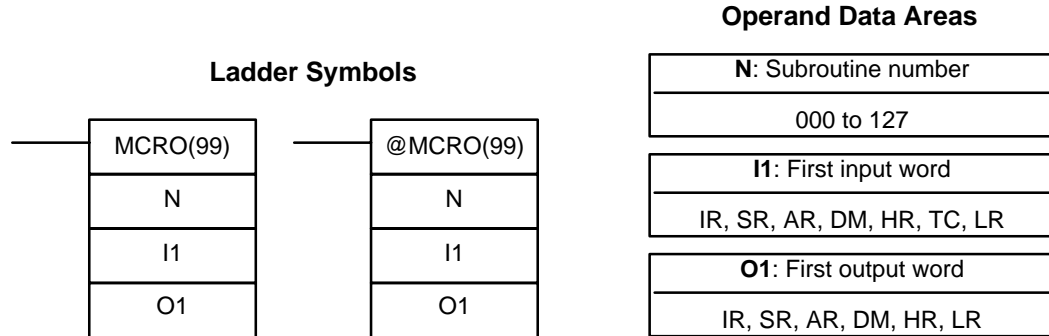


will be refreshed. This will be in addition to the normal I/O refresh performed during the CPU Unit's cycle.

**Note** This instruction will have no effect on words that are not being used for I/O.

**Flags** There are no flags affected by this instruction.

### 5-26-4 MACRO – MCRO(99)



**Limitations**

DM 6144 to DM 6655 cannot be used for O1. The CPM1/CPM1A/SRM1 PCs support only subroutine numbers 000 through 049.

**Description**

The MACRO instruction allows a single subroutine to replace several subroutines that have identical structure but different operands. There are 4 input words, IR 096 to IR 099 (IR 232 to IR 235 in the CPM1/CPM1A/SRM1), and 4 output words, IR 196 to IR 199 (IR 236 to IR 239 in CPM1/CPM1A/SRM1 PCs), allocated to MCRO(99). These 8 words are used in the subroutine and take their contents from I1 to I1+3 and O1 to O1+3 when the subroutine is executed.

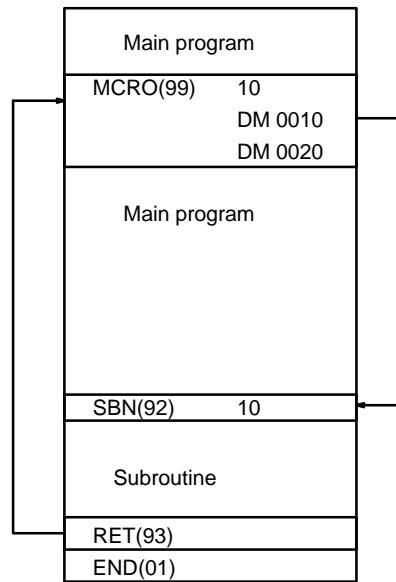
When the execution condition is OFF, MCRO(99) is not executed. When the execution condition is ON, MCRO(99) copies the contents of I1 to I1+3 to IR 096 to IR 099, copies the contents of O1 to O1+3 to IR 196 to IR 199, and then calls and executes the subroutine specified in N. When the subroutine is completed, the contents of IR 196 through IR 199 is then transferred back to O1 to O1+3 before MCRO(99) is completed.

**Note** Refer to page 128 for more details on MCRO(99).

**Example**

In this example, the contents of DM 0010 through DM 0013 are copied to IR 096 through IR 099, the contents of DM 0020 through DM 0023 are copied to IR 196 through IR 199, and subroutine 10 is called and executed. When the subroutine

is completed, the contents of IR 196 to IR 199 are copied back to DM 0020 to DM 0023.

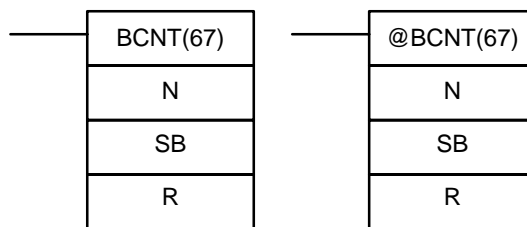


**Flags**

- ER:** A subroutine does not exist for the specified subroutine number.
- An operand has exceeded a data area boundary.
- Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)
- A subroutine has called itself.
- An active subroutine has been called.

**5-26-5 BIT COUNTER – BCNT(67)**

**Ladder Symbols**



**Operand Data Areas**

<b>N:</b> Number of words (BCD)
IR, SR, AR, DM, HR, TC, LR, #
<b>SB:</b> Source beginning word
IR, SR, AR, DM, HR, TC, LR
<b>R:</b> Destination word
IR, SR, AR, DM, HR, TC, LR

**Note** BCNT(67) is an expansion instruction for the SRM1. The function code 67 is the factory setting and can be changed for the SRM1 if desired.

**Limitations**

- N cannot be 0.
- DM 6144 to DM 6655 cannot be used for R.

**Description**

When the execution condition is OFF, BCNT(67) is not executed. When the execution condition is ON, BCNT(67) counts the total number of bits that are ON in all words between SB and SB+(N-1) and places the result in R.

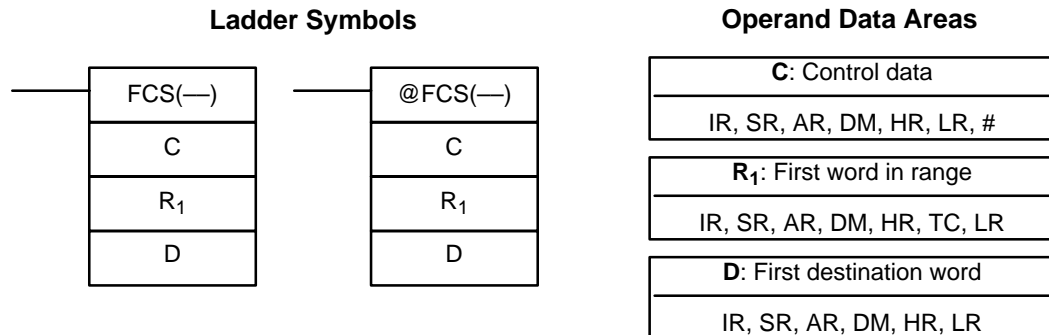
**Flags**

- ER:** N is not BCD, or N is 0; SB and SB+(N-1) are not in the same area. The resulting count value exceeds 9999.

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

**EQ:** ON when the result is 0.

### 5-26-6 FRAME CHECKSUM – FCS(—)

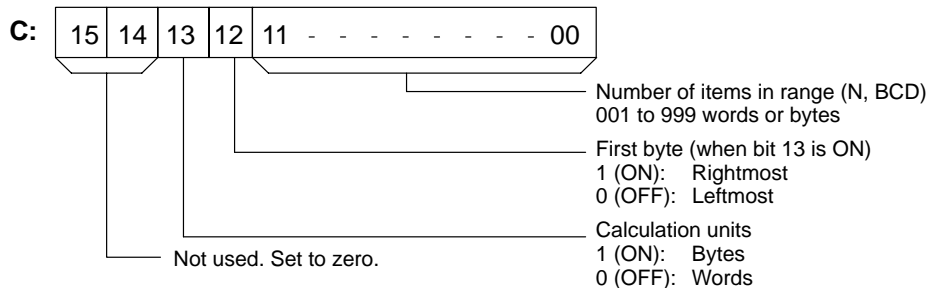


**Limitations**

This instruction is available in the **CQM1/SRM1 only**.  
 The 3 rightmost digits of C must be BCD between 001 and 999.  
 DM 6143 to DM 6655 cannot be used for D.

**Description**

FCS(—) can be used to check for errors when transferring data through communications ports.  
 When the execution condition is OFF, FCS(—) is not executed. When the execution condition is ON, FCS(—) calculates the frame checksum of the specified range by exclusively ORing either the contents of words  $R_1$  to  $R_1+N-1$  or the bytes in words  $R_1$  to  $R_1+N-1$ . The frame checksum value (hexadecimal) is then converted to ASCII and output to the destination words (D and D+1).  
 The function of bits in C are shown in the following diagram and explained in more detail below.



**Number of Items in Range**

The number of items within the range (N) is contained in the 3 rightmost digits of C, which must be BCD between 001 and 999.

**Calculation Units**

The frame checksum of words will be calculated if bit 13 is OFF and the frame checksum of bytes will be calculated if bit 13 is ON.  
 If bytes are specified, the range can begin with the leftmost or rightmost byte of  $R_1$ . The leftmost byte of  $R_1$  will not be included if bit 12 is ON.

	MSB	LSB
$R_1$	1	2
$R_1+1$	3	4
$R_1+2$	5	6
$R_1+3$	7	8
⋮	⋮	⋮
⋮	⋮	⋮
⋮	⋮	⋮

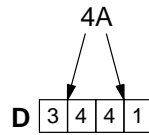
When bit 12 is OFF the bytes will be ORed in this order: 1, 2, 3, 4, ....

When bit 12 is ON the bytes will be ORed in this order: 2, 3, 4, 5, ....

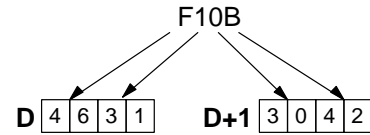
**Conversion to ASCII**

The byte frame checksum calculation yields a 2-digit hexadecimal value which is converted to its 4-digit ASCII equivalent. The word frame checksum calculation yields a 4-digit hexadecimal value which is converted to its 8-digit ASCII equivalent, as shown below.

Byte frame checksum value



Word frame checksum value



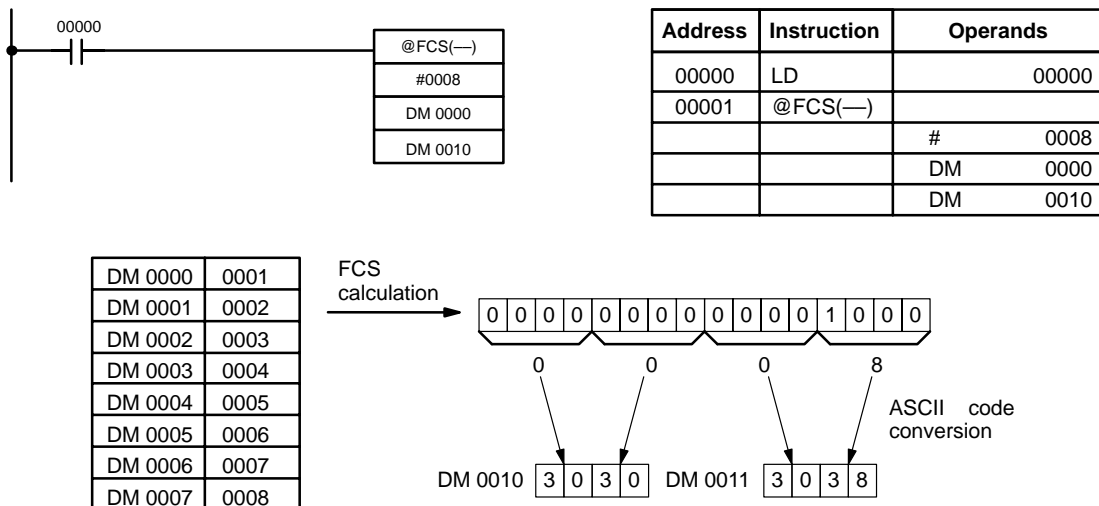
**Flags**

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

The number of items is not 001 to 999 BCD.

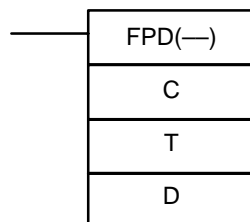
**Example**

When IR 00000 is ON in the following example, the frame checksum (0008) is calculated for the 8 words from DM 0000 to DM 0007 and the ASCII equivalent (30 30 30 38) is written to DM 0010 and DM 0011.



**5-26-7 FAILURE POINT DETECTION – FPD(—)**

**Ladder Symbols**



**Operand Data Areas**

<b>C:</b> Control data
#
<b>T:</b> Monitoring time (BCD)
IR, SR, AR, DM, HR, TC, LR, #
<b>D:</b> First register word
IR, SR, AR, DM, HR, LR

**Limitations**

This instruction is available in the **CQM1 only**.

D and D+8 must be in the same data area when bit 15 of C is ON.

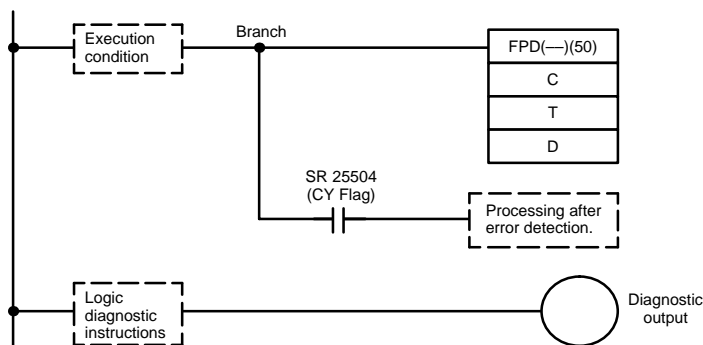
DM 6144 to DM 6655 cannot be used for T or D.

C must be input as a constant.

**Description**

FPD(—) can be used in the program as many times as desired, but each must use a different D. It is used to monitor the time between the execution of FPD(—) and the execution of a diagnostic output. If the time exceeds T, an FAL(06) non-fatal error will be generated with the FAL number specified in C.

The program sections marked by dashed lines in the following diagram can be written according to the needs of the particular program application. The processing programming section triggered by CY is optional and can use any instructions but LD and LD NOT. The logic diagnostic instructions and execution condition can consist of any combination of NC or NO conditions desired.

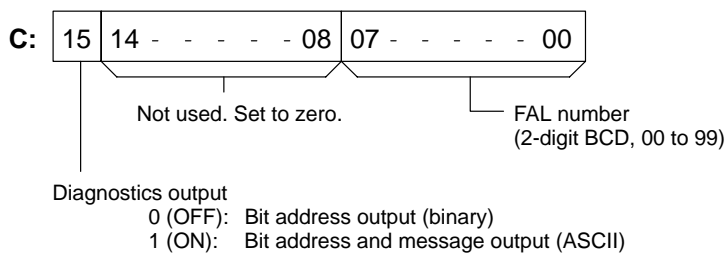


When the execution condition is OFF, FPD(—) is not executed. When the execution condition is ON, FPD(—) monitors the time until the logic diagnostics condition goes ON, turning ON the diagnostic output. If this time exceeds T, the following will occur:

- 1, 2, 3...
1. An FAL(06) error is generated with the FAL number specified in the first two digits of C. If 00 is specified, however, an error will not be generated.
  2. The logic diagnostic instructions are searched for the first OFF input condition and this condition's bit address is output to the destination words beginning at D.
  3. The CY Flag (SR 25504) is turned ON. An error processing program section can be executed using the CY Flag if desired.
  4. If bit 15 of C is ON, a preset message with up to 8 ASCII characters will be displayed on the Peripheral Device along with the bit address mentioned in step 2.

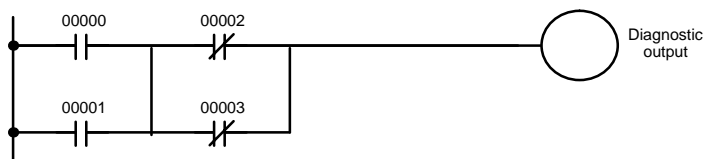
**Control Data**

The function of the control data bits in C are shown in the following diagram.



**Logic Diagnostic Instructions**

If the time until the logic diagnostics condition goes ON exceeds T, the logic diagnostic instructions are searched for the OFF input condition. If more than one input condition is OFF, the input condition on the highest instruction line and nearest the left bus bar is selected.



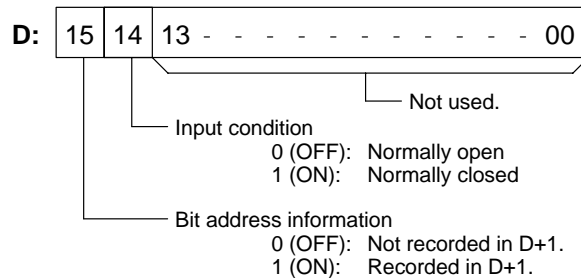
When IR 00000 to IR 00003 are ON, the normally closed condition IR 00002 would be found as the cause of the diagnostic output not turning ON.

**Diagnostics Output**

There are two ways to output the bit address of the OFF condition detected in the logic diagnostics condition.

- 1, 2, 3... 1. Bit address output (used when bit 15 of C is OFF).

Bit 15 of D indicates whether or not bit address information is stored in D+1. If there is, bit 14 of D indicates whether the input condition is normally open or closed.

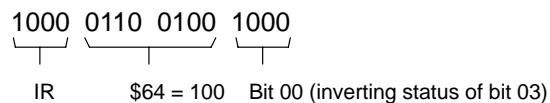


D+1 contains the bit address code of the input condition, as shown below. The word addresses, bit numbers, and TC numbers are in binary.

Data Area	D+1 bit status																	
	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00		
IR, SR	1	0	0	0	Word address								Bit number					
HR	1	0	0	1	1	Word address								Bit number				
LR	1	0	0	1	0	0	Word address								Bit number			
TC*	1	0	0	1	0	1	*	Timer or counter number										

- Note** a) \*For the TC area, bit 09 of D+1 indicates whether the number is a timer or counter. A 0 indicates a timer, and a 1 indicates a counter.  
 b) The status of the leftmost bit of the bit number (bit 03) is reversed.

**Example:** If D + 1 contains 1000 0110 0100 1000, IR 10000 would be indicated as follows:



2. Bit address and message output (selected when bit 15 of C is ON).

Bit 15 of D indicates whether or not there is bit address information stored in D+1 to D+3. If there is, bit 14 of D indicates whether the input condition is normally open or closed. Refer to the following table.

Words D+5 to D+8 contain information in ASCII that are displayed on a Peripheral Device along with the bit address when FPD(—) is executed. Words D+5 to D+8 contain the message preset by the user as shown in the following table.

Word	Bits 15 to 08	Bits 07 to 00
D+1	20 = space	First ASCII character
D+2	Second ASCII character	Third ASCII character
D+3	Fourth ASCII character	Fifth ASCII character
D+4	2D = “-”	“0”=normally open, “1”=normally closed
D+5	First ASCII character	Second ASCII character
D+6	Third ASCII character	Fourth ASCII character
D+7	Fifth ASCII character	Sixth ASCII character
D+8	Seventh ASCII character	Eighth ASCII character

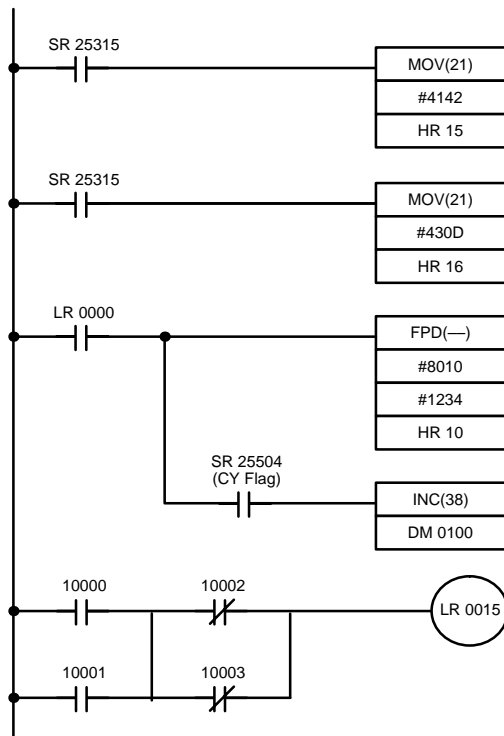
**Note** If 8 characters are not needed in the message, input “0D” after the last character.

**Determining Monitoring Time** The procedure below can be used to automatically set the monitoring time, T, under actual operating conditions when specifying a word operand for T. This operation cannot be used if a constant is set for T.

- 1, 2, 3... 1. Switch the CQM1 to MONITOR Mode operation.
2. Connect a Peripheral Device, such as a Programming Console.
3. Use the Peripheral Device to turn ON control bit AR 2508.
4. Execute the program with AR 2508 turned ON. If the monitoring time currently in T is exceeded, 1.5 times the actual monitoring time will be stored in T. FAL(06) errors will not occur while AR 2508 is ON.
5. Turn OFF AR 2508 when an acceptable value has been stored in T.

**Example**

In the following example, the FPD(—) is set to display the bit address and message (“ABC”) when a monitoring time of 123.4 s is exceeded.



Address	Instruction	Operands
00000	LD	25315
00001	MOV(21)	
		# 4142
		HR 15
00002	LD	25315
00003	MOV(21)	
		# 430D
		HR 16
00004	LD	LR 0000
00005	FPD(—)	
		# 0010
		# 1234
		HR 10
00006	AND	25504
00007	INC(38)	
		DM 0100
00008	LD	10000
00009	OR	10001
00010	LD NOT	10002
00011	OR NOT	10003
00012	AND LD	
00013	OUT	LR 0015

FPD(—) is executed and begins monitoring when LR 0000 goes ON. If LR 0015 does not turn ON within 123.4 s and IR 10000 through IR 10003 are all ON, IR 10002 will be selected as the cause of the error, an FAL(06) error will be generated with an FAL number of 10, and the bit address and preset message (“10002–1ABC”) will be displayed on the Peripheral Device.

HR 10	0000
HR 11	0000
HR 12	0000
HR 13	0000
HR 14	0000
HR 15	4142
HR 16	430D
HR 17	0000
HR 18	0000



HR 10	C000
HR 11	2031
HR 12	3030
HR 13	3032
HR 14	2D31
HR 15	4142
HR 16	430D
HR 17	0000
HR 18	0000

Indicates information, normally closed condition  
 “1”  
 “00”  
 “02”  
 “\_1”  
 “AB”  
 “C”, and CR code  
 The last two words are ignored.  
 (Displayed as spaces.)

**Flags**

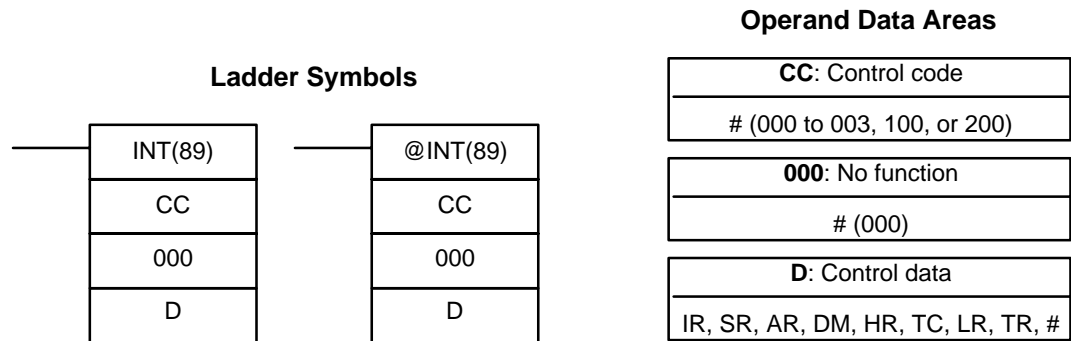
**ER:** T is not BCD.

C is not a constant or is not BCD 00 to 99.

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

**CY:** ON when the time between the execution of FPD(—) and the execution of a diagnostic output exceeds T.

**5-26-8 INTERRUPT CONTROL – INT(89)**



**Note** This instruction is not supported by SRM1 PCs.

**Limitations**

DM 6644 to DM 6655 cannot be used for D when CC=002.

**Description**

When the execution condition is OFF, INT(89) is not executed. When the execution condition is ON, INT(89) is used to control interrupts and performs one of the six functions shown in the following table depending on the value of CC.

**Note** Refer to 1-5 CQM1 Interrupt Functions and 1-6 CPM1/CPM1A Interrupt Functions for more details.

INT(89) function	CC
Mask/unmask input interrupts	000
Clear input interrupts	001
Read current mask status	002
Renew counter SV	003
Mask all interrupts	100
Unmask all interrupts	200

These six functions are described in more detail below. Refer to page 22 for more information on these functions.

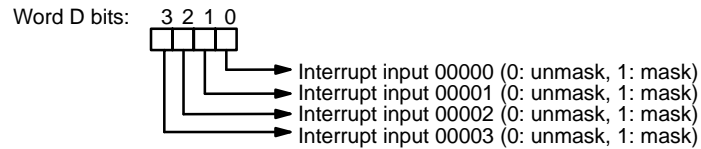
**Mask/Unmask I/O Interrupts (CC=000)**

This function is used to mask and unmask I/O interrupt inputs 00000 to 00003 (00003 to 00006 in CPM1/CPM1A PCs). Masked inputs are recorded, but ignored. When an input is masked, the interrupt program for it will be run as soon as the bit is unmasked (unless it is cleared beforehand by executing INT(89) with CC=001).

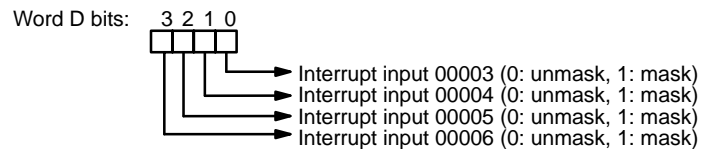


Set the corresponding bit in D to 0 or 1 to unmask or mask an I/O interrupt input. Bits 00 to 03 correspond to 00000 to 00003 (00003 to 00006 in CPM1/CPM1A PCs). Bits 04 to 15 should be set to 0.

**CQM1 PCs**



**CPM1/CPM1A PCs**

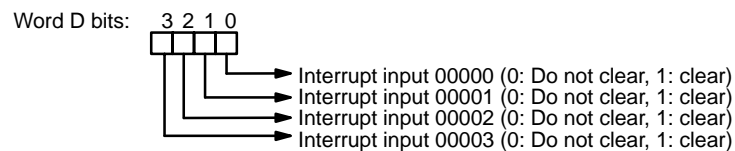


**Clear I/O Interrupts (CC=001)**

This function is used to clear I/O interrupt inputs 00000 to 00003 (00003 to 00006 in CPM1/CPM1A PCs). Since interrupt inputs are recorded, masked interrupts will be serviced after the mask is removed unless they are cleared first.

Set the corresponding bit in D to 1 to clear an I/O interrupt input. Bits 00 to 03 correspond to 00000 to 00003 (00003 to 00006 in CPM1/CPM1A PCs). Bits 04 to 15 should be set to 0.

**CQM1 PCs**



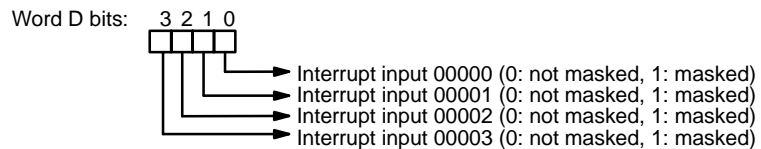
**CPM1/CPM1A PCs**



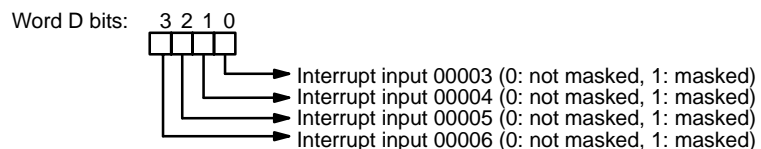
**Read Current Mask Status (CC=002)**

This function is used to write the current mask status for I/O interrupt inputs 00000 to 00003 (00003 to 00006 in CPM1/CPM1A PCs) to word D. The corresponding bit will be ON if the input is masked. (Bits 00 to 03 correspond to 00000 to 00003 in CQM1 PCs, 00003 to 00006 in CPM1/CPM1A PCs.)

**CQM1 PCs**



**CPM1/CPM1A PCs**

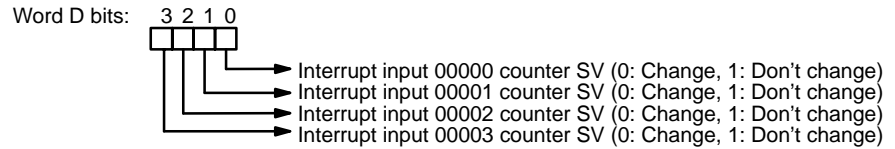


**Renew Counter SV (CC=003)**

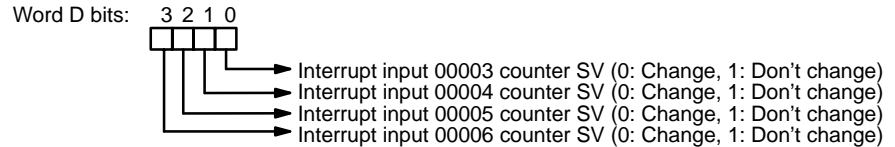
This function is used to renew the counter SV for I/O interrupt inputs 00000 to 00003 (00003 to 00006 in CPM1/CPM1A PCs) to word D. Set the corresponding

bit in D to 1 in order to renew the input's counter SV. (Bits 00 to 03 correspond to 00000 to 00003 in CQM1 PCs, 00003 to 00006 in CPM1/CPM1A PCs.)

**CQM1 PCs**



**CPM1/CPM1A PCs**



**Mask/Unmasking All Interrupts (CC=100/200)**

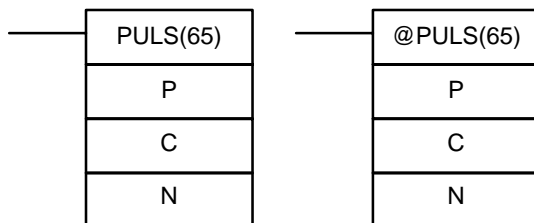
This function is used to mask or unmask all interrupt processing. Masked inputs are recorded, but ignored. Refer to page 44 for details. The control data, D, is not used for this function. Set D to #0000.

**Flags**

- ER:** A counter's SV is incorrect. (CC=003 only)
- Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)
- CC=100 or 200 while an interrupt program was being executed.
- CC=100 when all inputs were already masked.
- CC=200 when all inputs were already unmasked.
- CC and/or D are not within specified values.

**5-26-9 SET PULSES – PULS(65)**

**Ladder Symbols**



**Operand Data Areas**

<b>P:</b> Port specifier
000, 001, or 002
<b>C:</b> Control data
000 to 005
<b>N:</b> Number of pulses
IR, SR, AR, DM, HR, LR

**Limitations**

This instruction is available in the **CPM1A with transistor outputs and CQM1 only**.

N and N+1 must be in the same data area.  
DM 6143 to DM 6655 cannot be used for N.

**Description**

PULS(65) is used to set parameters for pulse outputs that are started later in the program using SPED(64) or ACC(—). The parameters that can be set are the number of pulses that will be output in independent mode, the direction of pulse outputs from ports 1 and 2, and the deceleration point for pulse outputs controlled by ACC(—) mode 0.

Since PULS(65) has a relatively long execution time, the cycle time can be reduced by executing the differentiated version (@PULS(65)) of this instruction only when it is needed.

**Note** Refer to 1-3 Pulse Output Functions for more details.

**Port Specifier (P)**

The port specifier indicates the pulse output location. The parameters set by the in C and N will apply to the next SPED(64) or ACC(—) instruction in which the same port output location is specified.

P	Pulse output location
000	Output bit
001	Port 1
002	Port 2

**Control Data (C)**

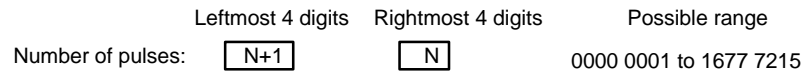
The control data determines the direction of the pulse output to ports 1 and 2 and indicates whether the number of pulses and/or the deceleration point are specified in N to N+3. This operand should be set to 000 when P=000.

C	Direction	Number of pulses	Deceleration point
000	CW	Set in N and N+1	Not set.
001	CCW	Set in N and N+1	Not set.
002	CW	Set in N and N+1	Set in N+2 and N+3
003	CCW	Set in N and N+1	Set in N+2 and N+3
004	CW	Not set.	Not set.
005	CCW	Not set.	Not set.

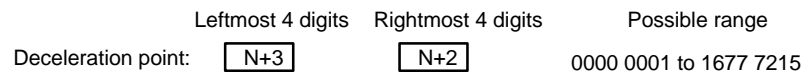
The direction setting is valid until program execution is stopped or PULS(65) is executed again.

**Number of Pulses and Deceleration Point**

When C=000 to 003, N+1, N contains the 8-digit number of pulses setting for independent mode pulse outputs. N+1, N can be from 00000001 to 16777215. The pulse output started by SPED(64) or ACC(—) will stop automatically when this number of pulses has been output.



When C=002 or 003, N+3, N+2 contains the 8-digit number of pulses setting for the deceleration point used in ACC(—) mode 0. N+3, N+2 can be from 00000001 to 16777215. The pulse output started by ACC(—) will begin deceleration when this number of pulses have been output.



When C=004 or 005, neither the number of pulses nor the deceleration point are set. Set N=000 when C=004 or 005.

**Frequency Changes**

The number of pulses set to be output will be used even if SPED(64) is used to change the pulse frequency during operation.

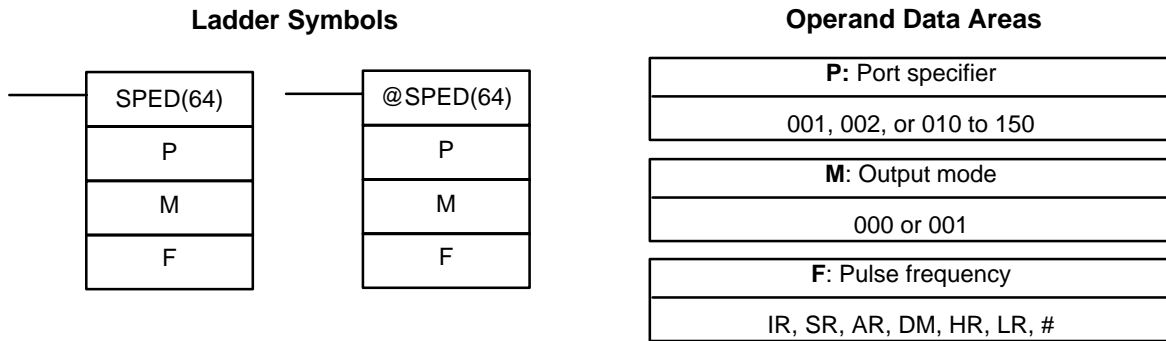
For example, if the number of pulses setting is 2,100 and the frequency is changed from 1 KHz to 100 Hz, pulse output will stop in:

- 12 s if the pulse frequency is changed after 1 s at 1 KHz.
- 3 s if the pulse frequency is changed after 2 s at 1 KHz.

**Flags**

- ER:** There is an error in the instruction settings.  
If a data area boundary is exceeded.  
  
Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)  
  
PULS(65) is executed in an interrupt subroutine while a pulse I/O or high-speed counter instruction is being executed in the main program.

### 5-26-10 SPEED OUTPUT– SPED(64)



**Limitations**

This instruction is available in the **CPM1A with transistor outputs and CQM1 only**.

F must be BCD, #0000 to #5000 when a port is specified, #0000 or #0002 to #0100 when an output bit is specified.

DM 6144 to DM 6655 cannot be used for F.

**Description**

SPED(64) is used to set, change, or stop pulse output from the specified port or output bit. When the execution condition is OFF, SPED(64) is not executed. When the execution condition is ON, SPED(64) sets the pulse frequency F for the port or output bit specified by P. M determines the output mode.

Since SPED(64) has a relatively long execution time, the cycle time can be reduced by executing the differentiated version (@SPED(64)) of this instruction only when it is needed.

**Note** Refer to *1-3 Pulse Output Functions* for more details.

**Port Specifier (P)**

The port specifier specifies the port or output bit where the pulses will be output.

P	Pulse output location
001	Port 1
002	Port 2
000 to 150	Output bits IR 10000 to IR 10015. The first two digits of P specify which bit of IR 100 is the output bit and the third digit of P is always set to 0. For example, P=000 specifies IR 10000, P=010 specifies IR 10001, ... and P=150 specifies bit IR 10015.

**Output Mode (M)**

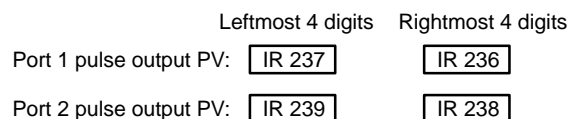
The value of M determines the output mode. A value of 000 indicates independent mode and a value of 001 indicates continuous mode.

In independent mode, the pulse output will continue until one of the following occurs:

- 1, 2, 3...**
1. The number of pulses specified by the PULS(65) instruction is reached. (Execute PULS(65) before SPED(64) when specifying independent mode.)
  2. The INI(61) instruction is executed with C=003.
  3. SPED(64) is executed again with the output frequency, F, set to 000.

When outputting pulses in independent mode, specify the number of pulses beforehand by executing PULS(65). When outputting from port 1 or 2, specify the direction (CW or CCW) as well.

In independent mode, the number of pulses that have been output to ports 1 and 2 are contained in IR 236 and 237 (port 1) and IR 238 and IR 239 (port 2).



In continuous mode, pulses will be output until the INI(61) instruction is executed with C=003 or SPED(64) is executed again with F=0000. If the direction (CW or CCW) is not specified when outputting from port 1 or 2, the pulses will be CW.

**Pulse Frequency (F)**

The value of F sets the pulse frequency in units of 10 Hz, as shown below. Setting F to 0000 will stop the pulse output at the specified location.

Output location	Possible values of F
Port 1 or 2	0000 (stops pulse output) or 0001 to 5000 (10 Hz to 50 kHz)
Output bits	0000 (stops pulse output) or 0002 to 0100 (20 Hz to 1 kHz)

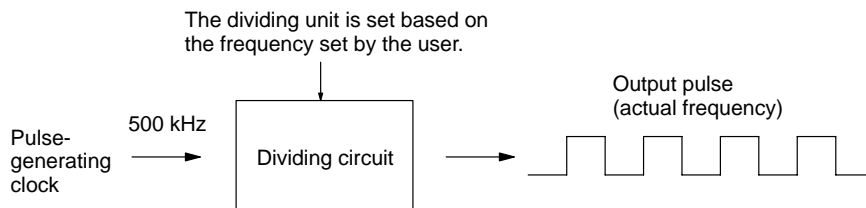
**Precaution Regarding Pulse Output**

The pulse frequency output from the CQM1-CPU43-EV1 is generated by dividing the 500-kHz basic clock by a dividing unit of an integer value, which results in a difference between the set frequency and actual frequency. Refer to the following equation for calculating an actual frequency.

Set Frequency: Output frequency set in the instruction by the user

Dividing Unit: An integer set in the dividing circuit to generate an output pulse of the set frequency

Actual Frequency: Output pulse frequency actually output from the dividing circuit



Equation:

$$\text{Actual frequency (KHz)} = 500 \text{ (KHz)} / \text{INT} (500 \text{ (kHz)} / \text{Set frequency (kHz)})$$

INT: Function for obtaining an integer value

INT (500/Set frequency): Dividing unit

The difference between the set frequency and actual frequency becomes larger as the frequency becomes higher.

Example:

Set frequency (kHz)	Actual frequency (kHz)
45.46 to 50.00	50.00
41.67 to 45.45	45.45
38.47 to 41.66	41.67
:	:
31.26 to 33.33	33.33
29.42 to 31.25	31.25
27.78 to 29.41	29.41
:	:
20.01 to 20.83	20.83
19.24 to 20.00	20.00
18.52 to 19.23	19.23
:	:
10.01 to 10.20	10.20
9.81 to 10.00	10.00
9.62 to 9.80	9.80
:	:
5.01 to 5.05	5.05

Set frequency (kHz)	Actual frequency (kHz)
4.96 to 5.00	5.00
4.90 to 4.95	4.95
:	:
3.02 to 3.03	3.03
3.00 to 3.01	3.01
2.98 to 2.99	2.99
:	:

**Precautions**

With the CQM1-CPU11/21-E, the output refreshing method in DM 6639 (PC Setup) must be set to direct output before initiating pulse output.

The pulse output cannot be used when interval timer 0 is operating.

When a pulse output with a frequency of 500 Hz or higher is output from an output bit, set interrupt processing for the TIMH(15) TC numbers 000 to 003 by setting #0104 in DM 6629 of the PC Setup.

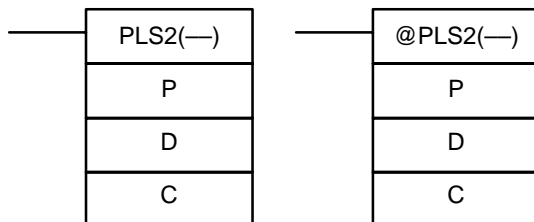
Only one output bit at a time can have a pulse output.

**Flags**

**ER:** SPED(64) is executed while interval timer 0 is operating.  
 Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)  
 There is an error in the instruction settings.  
 SPED(64) is executed in an interrupt subroutine while a pulse I/O or high-speed counter instruction is being executed in the main program.

**5-26-11 PULSE OUTPUT – PLS2(—)**

**Ladder Symbols**



**Operand Data Areas**

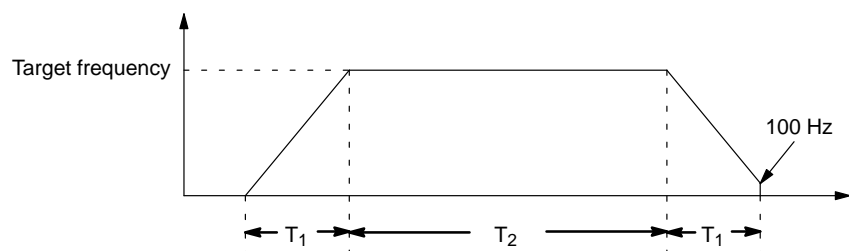
<b>P:</b> Communications port
001 or 002
<b>D:</b> Direction specifier
000 or 001
<b>C:</b> First control word
IR, SR, AR, DM, HR, LR

**Limitations**

This instruction is available in the **CQM1-CPU43-E/-EV1 only**.  
 PLS2(—) cannot be used if the PC Setup (DM 6611) is set to high-speed counter mode.  
 P must be 001 or 002 and D must be 000 or 001.  
 C to C+3 must be in the same data area.

**Description**

PLS2(—) is used to output a specified number of CW or CCW pulses from port 1 or 2. The pulse output accelerates to the target frequency at a specified rate and decelerates at the same rate. (Pulse output stops at 100 Hz.)



The following equations show how to calculate the approximate acceleration/ deceleration time  $T_1$  and running time  $T_2$ . Both times are in seconds.

$$T_1 \approx 0.004 \times \frac{\text{Target frequency}}{\text{Acceleration} \mp \text{deceleration rate}}$$

$$T_2 \approx \frac{\text{Number of pulses} \approx (T_1 \times \text{Target frequency})}{\text{Target frequency}}$$

- Note**
1. Although  $T_1$  and  $T_2$  will vary slightly depending on the operating conditions, the number of pulses output will be accurate.
  2. PLS2(—) will not operate if pulses are already being output from the specified port. Check the pulse output flags (AR 0515 for port 1 and AR 0615 for port 2) before executing PLS2(—).
  3. Refer to 1-3 Pulse Output Functions for more details.

**Operand Settings**

P specifies the port where the pulses will be output. Pulses are output from port 1 when P=001, and pulses are output from port 2 when P=002.

D specifies whether the output signal is clockwise (CW) or counter-clockwise (CCW). The output is CW when D=000 and CCW when D=001.

The content of C determines the acceleration/deceleration rate. During acceleration or deceleration, the output frequency is increased or decreased by the amount set in C every 4.08 ms. C must be BCD from 0001 to 0200 (10 Hz to 2 kHz).

The content of C+1 specifies the target frequency. C+1 must be BCD from 0010 to 5000 (100 Hz to 50 kHz).

The 8-digit content of C+3,C+2 determines the number of pulses that will be output. C+3, C+2 must be BCD between 0000 0001 and 1677 7215.

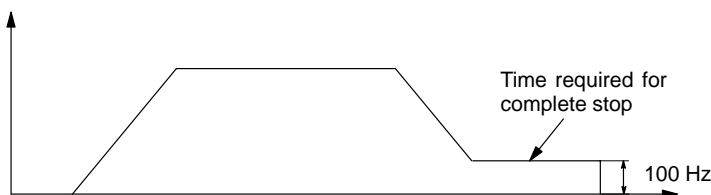
**Flags**

- ER:** There is an error in the operand settings.  
 The CPU Unit is not a CQM1-CPU43-EV1.  
 The PC Setup is not set for pulse output.  
 The target frequency, acceleration/deceleration rate, and number of pulses are incorrect. (Number of pulses <  $T_1 \times$  Target frequency)  
 PLS2(—) is executed in an interrupt subroutine while a pulse I/O or high-speed counter instruction is being executed in the main program.  
 A data area boundary has been exceeded.  
 Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)

- AR 0515:** Port 1 output flag. ON when pulses are being output from port 1.  
**AR 0615:** Port 2 output flag. ON when pulses are being output from port 2.

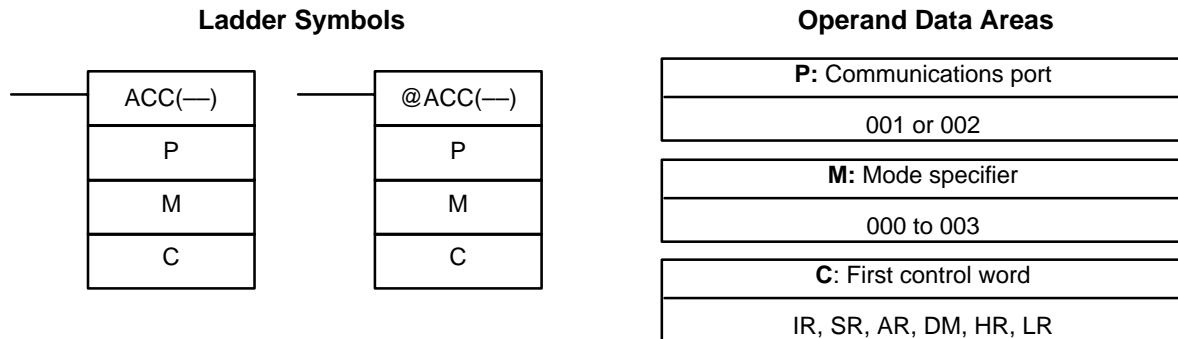
**Caution**

With PLS2(—), conditions such as acceleration/deceleration speed and the target speed can cause low-speed pulse output (100 Hz) to continue for an extended period of time when stopping. Even when this happens, the correct number of pulses will be output.



Correct the system by adjusting the acceleration/deceleration speed and/or the target speed, or by using the ACC(—) instruction (mode 0) to increase the speed (deceleration target frequency) when stopping.

### 5-26-12 ACCELERATION CONTROL – ACC(—)



**Limitations**

This instruction is available in the **CQM1-CPU43-E/-EV1 only**.  
 Mode 0 of ACC(—) cannot be used if the PC Setup (DM 6611) is set to high-speed counter mode.  
 P must be 001 or 002 and M must be 000 to 003.  
 C to C+3 must be in the same data area.

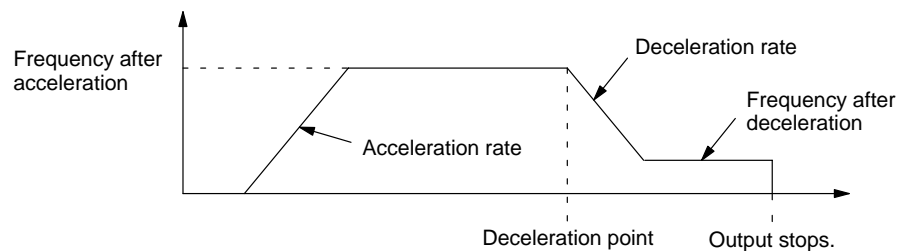
**Description**

ACC(—) is used together with PULS(65) to control the acceleration and/or deceleration of pulses output from port 1 or 2. The 4 available modes are described briefly below.  
 The function of the control words varies in the 4 modes, but P always specifies the port where the pulses will be output and M always specifies the mode. Set P=001 or 002 to indicate port 1 or 2. Set M=000 to 003 to indicate modes 0 to 3.

**Note** Refer to *1-3 Pulse Output Functions* for more details.

**Mode 0 (M=000)**

Mode 0 is used to output a specified number of CW or CCW pulses from port 1 or 2. The acceleration rate, frequency after acceleration, deceleration point, deceleration rate, and frequency after deceleration can all be controlled.



**PULS(65) Operand Settings**

PULS(65) must be executed before ACC(—) in order to specify direction, the total number of pulses to be output, and the deceleration point. The function of PULS(65) operands are described below. Refer to *5-26-9 SET PULSES – PULS(65)* for more details.

- 1, 2, 3...
  1. The first operand of PULS(65) specifies the output port. Pulses are output from port 1 when P=001, and from port 2 when P=002.
  2. The second operand specifies the direction. The output is clockwise (CW) when C=002 and counter-clockwise (CCW) when C=003.
  3. The third operand specifies the first of 4 control words.
    - a) The 8-digit content of N+1,N (0000 0001 to 1677 7215) determines the total number of pulses that will be output.



- b) The 8-digit content of N+3,N+2 (0000 0001 to 1677 7215) determines the deceleration point.

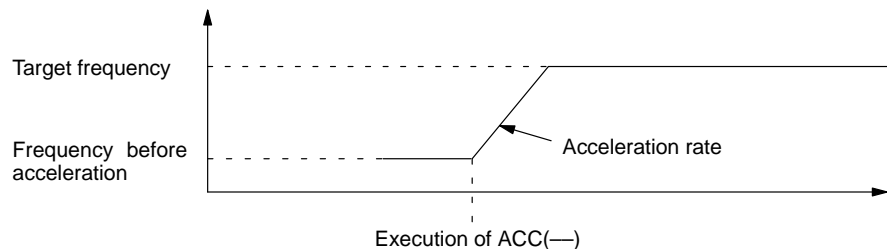
**ACC(—) Control Words**

The 4 control words indicate the acceleration rate, frequency after acceleration, deceleration rate, and frequency after deceleration.

- 1, 2, 3...**
1. The content of C determines the acceleration rate. During acceleration, the output frequency is increased by the amount set in C every 4.08 ms. C must be BCD from 0001 to 0200 (10 Hz to 2 kHz).
  2. The content of C+1 specifies the frequency after acceleration. C+1 must be BCD from 0000 to 5000 (0 Hz to 50 kHz).
  3. The content of C+2 determines the deceleration rate. During deceleration, the output frequency is decreased by the amount set in C+2 every 4.08 ms. C must be BCD from 0001 to 0200 (10 Hz to 2 kHz).
  4. The content of C+3 specifies the frequency after deceleration. C+3 must be BCD from 0000 to 5000 (0 Hz to 50 kHz).

**Mode 1 (M=001)**

Mode 1 is used to increase the frequency being output to a target frequency at the specified rate. Pulse output continues until stopped.

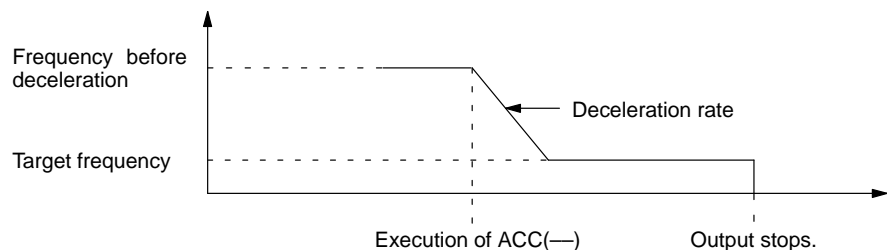


The 2 control words indicate the acceleration rate and target frequency.

- 1, 2, 3...**
1. The content of C determines the acceleration rate. During acceleration, the output frequency is increased by the amount set in C every 4.08 ms. C must be BCD from 0001 to 0200 (10 Hz to 2 kHz).
  2. The content of C+1 specifies the target frequency. C+1 must be BCD from 0000 to 5000 (0 Hz to 50 kHz).

**Mode 2 (M=002)**

Mode 2 is used to decrease the frequency being output to a target frequency at the specified rate. Output stops when the total number of pulses specified in PULS(65) have been output.

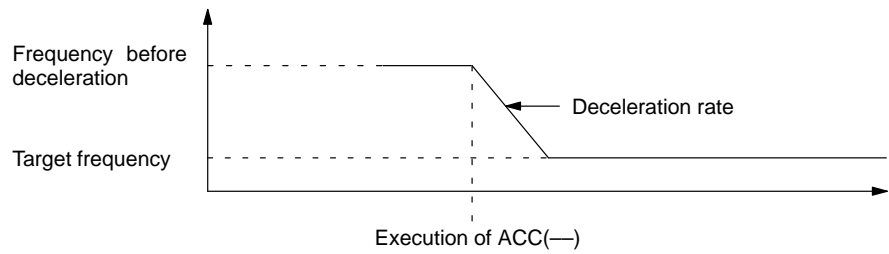


The 2 control words indicate the deceleration rate and target frequency.

- 1, 2, 3...**
1. The content of C determines the deceleration rate. During deceleration, the output frequency is decreased by the amount set in C every 4.08 ms. C must be BCD from 0001 to 0200 (10 Hz to 2 kHz).
  2. The content of C+1 specifies the target frequency. C+1 must be BCD from 0000 to 5000 (0 Hz to 50 kHz).

**Mode 3 (M=003)**

Mode 3 is used to decrease the frequency being output to a target frequency at the specified rate. Pulse output continues until stopped.



The 2 control words indicate the acceleration rate and target frequency.

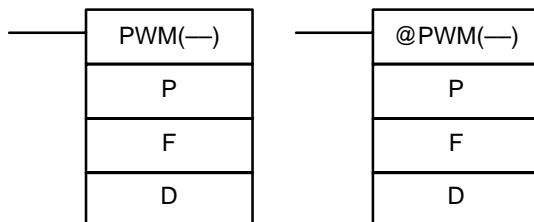
- 1, 2, 3...**
1. The content of C determines the acceleration rate. During acceleration, the output frequency is increased by the amount set in C every 4.08 ms. C must be BCD from 0001 to 0200 (10 Hz to 2 kHz).
  2. The content of C+1 specifies the target frequency. C+1 must be BCD from 0000 to 5000 (0 Hz to 50 kHz).

**Flags**

- ER:** There is an error in the operand settings.  
 The CPU Unit is not a CQM1-CPU43-EV1.  
 The PC Setup is not set for pulse output.  
 ACC(—) is executed with M=000 and the specified output port is in use.  
 ACC(—) is executed in an interrupt subroutine while a pulse I/O or high-speed counter instruction is being executed in the main program.  
 Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)
- AR 0515:** Port 1 output flag. ON when pulses are being output from port 1.  
**AR 0615:** Port 2 output flag. ON when pulses are being output from port 2.

**5-26-13 PULSE WITH VARIABLE DUTY RATIO – PWM(—)**

**Ladder Symbols**



**Operand Data Areas**

<b>P:</b> Communications port
001 or 002
<b>F:</b> Frequency
000, 001, or 002
<b>D:</b> Duty ratio
IR, SR, AR, DM, HR, TC, LR, #

**Limitations**

This instruction is available in the **CQM1-CPU43-E/-EV1 only**.  
 PWM(—) cannot be used unless the PC Setup (DM 6643 or DM 6644) is set for variable duty ratio pulse outputs.  
 P must be 001 or 002 and F must be 000, 001, or 002.  
 D must be BCD between 0001 and 0099.

**Description**

PWM(—) is used to output pulses with the specified duty ratio from port 1 or 2. The output can be set to one of three frequencies: 5.9 kHz, 1.5 kHz, or 91.6 Hz. The pulse output continues until INI(61) is executed to stop it.

In order for PWM(—) to be executed, the specified port must be set for variable duty ratio pulse outputs in the PC Setup. Set the leftmost digit of DM 6643 to 1 to enable variable duty ratio pulse output from port 1, and set the leftmost digit of DM 6644 to 1 to enable variable duty ratio pulse output from port 2. It is not possible to output normal pulses from a port that is set for variable duty ratio output.

**Note** Refer to 1-3 Pulse Output Functions for more details.

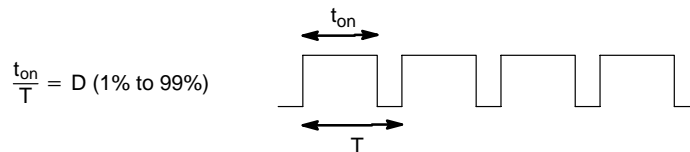
**Operand Settings**

P specifies the port where the pulses will be output. Pulses are output from port 1 when P=001, and pulses are output from port 2 when P=002.

F specifies the frequency of the pulse output, as shown in the following table.

F	Frequency
000	5.9 kHz
001	1.5 kHz
002	91.6 Hz

D specifies the duty ratio of the pulse output, i.e., the percentage of time that the output is ON. D must be BCD from 0001 to 0099 (1% to 99%). The duty ratio is 75% in the following diagram.

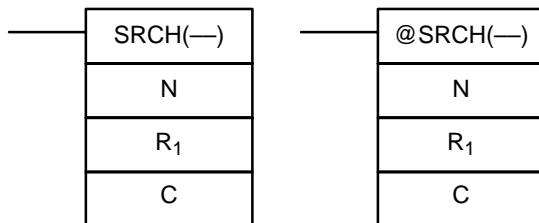


**Flags**

**ER:** There is an error in the operand settings.  
 The CPU Unit is not a CQM1-CPU43-EV1.  
 The PC Setup is not set for variable duty ratio pulse output.  
 PWM(—) is executed in an interrupt subroutine while a pulse I/O or high-speed counter instruction is being executed in the main program.  
 Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)

**5-26-14 DATA SEARCH – SRCH(—)**

**Ladder Symbols**



**Operand Data Areas**

<b>N:</b> Number of words
IR, SR, AR, DM, HR, TC, LR, #
<b>R<sub>1</sub>:</b> First word in range
IR, SR, AR, DM, HR, TC, LR
<b>C:</b> Comparison data, result word
IR, SR, AR, DM, HR, LR

**Limitations**

This instruction is available in the **CQM1 only**.  
 N must be BCD between 0001 to 9999.  
 R<sub>1</sub> and R<sub>1</sub>+N-1 must be in the same data area.  
 DM 6143 to DM 6655 cannot be used for C.

**Description**

When the execution condition is OFF, SRCH(—) is not executed. When the execution condition is ON, SRCH(—) searches the range of memory from R<sub>1</sub> to R<sub>1</sub>+N-1 for addresses that contain the comparison data in C. If one or more addresses contain the comparison data, the EQ Flag (SR 25506) is turned ON and the lowest address containing the comparison data is identified in C+1. The address is identified differently for the DM area:

- 1, 2, 3... 1. For an address in the DM area, the word address is written to C+1. For example, if the lowest address containing the comparison data is DM 0114, then #0114 is written in C+1.
2. For an address in another data area, the number of addresses from the beginning of the search is written to C+1. For example, if the lowest address containing the comparison data is IR 114 and the first word in the search range is IR 014, then #0100 is written in C+1.

If none of addresses in the range contain the comparison data, the EQ Flag (SR 25506) is turned OFF and C+1 is left unchanged.

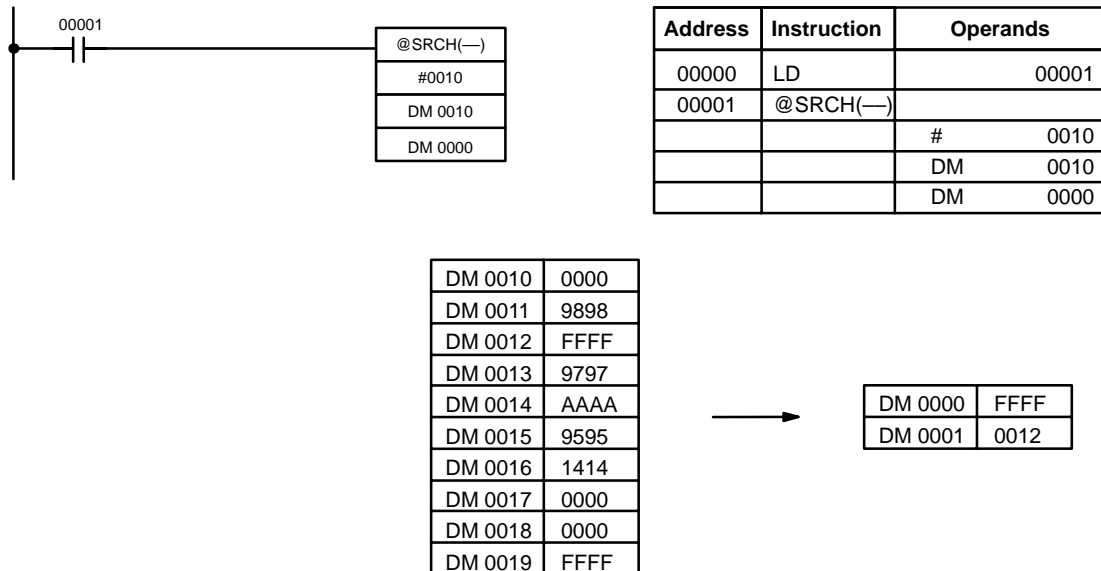
**Flags**

**ER:** Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)  
N is not BCD between 0001 and 9999.

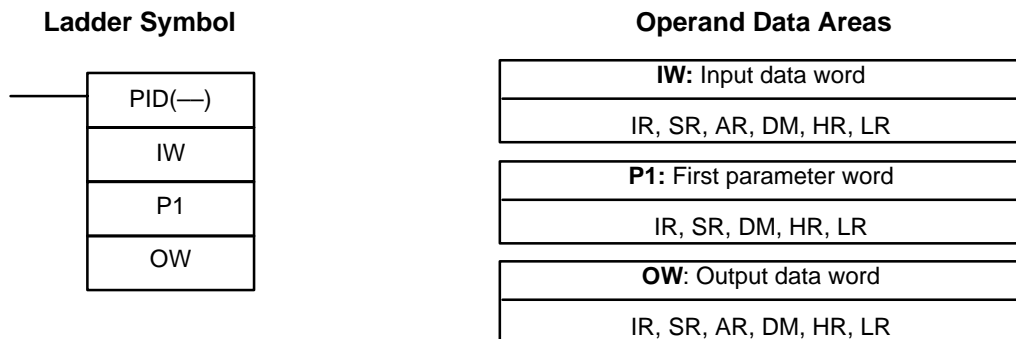
**EQ:** ON when the comparison data has been matched in the search range.

**Example**

In the following example, the 10 word range from DM 0010 to DM 0019 is searched for addresses that contain the same data as DM 0000 (#FFFF). Since DM 0012 contains the same data, the EQ Flag (SR 25506) is turned ON and #0012 is written to DM 0001.




**5-26-15 PID CONTROL – PID(---)**



**Limitations**

This instruction is available in the **CQM1-CPU4□-E/-EV1 only**.  
DM 6144 to DM 6655 cannot be used for IW, P1 to P1+32, or OW.  
P1 to P1+32 must be in the same data area.

 **Caution** A total of 33 continuous words starting with P1 must be provided for PID(—) to operate correctly. Also, PID(—) may not operate dependably in any of the following situations: In interrupt programs, in subroutines, between IL(02) and ILC(03), between JMP(04) and JME(05), and in step programming (STEP(08)/SNXT(09)). Do not program PID(—) in these situations.


**Description**

PID(—) performs PID control based on the parameters specified in P1 through P1+6. The data in IW is used to calculate the output data that is written to OW. The following table shows the function of the parameter words.

Word	Bits	Parameter name	Function/Setting range
P1	00 to 15	Set value (SV).	This is the target value for PID control. It can be set to any binary number with the number of bits set by the input range parameter.
P1+1	00 to 15	Proportional band width.	This parameter specifies the proportional band width/input range ratio from 0.1% to 999.9%. It must be BCD from 0001 to 9999.
P1+2	00 to 15	Integral time	Sets the integral time/sampling period ratio used in integral control. It must be BCD from 0001 to 8191, or 9999. (9999 disables integral control.)
P1+3	00 to 15	Derivative time	Sets the derivative time/sampling period ratio used in derivative control. It must be BCD from 0001 to 8191, or 0000.
P1+4	00 to 15	Sampling period	Sets the interval between samplings of the input data from 0.1 to 102.3 s. It must be BCD from 0001 to 1023.
P1+5	00 to 03	Operation specifier	Sets reverse or normal operation. Set to 0 to specify reverse operation or 1 to specify normal operation.
	04 to 15	Input filter coefficient	Determines the strength of the input filter. The lower the coefficient, the weaker the filter.  This setting must be BCD from 100 to 199, or 000. A setting of 000 sets the default value (0.65) and a setting of 100 to 199 sets the coefficient from 0.00 to 0.99.
P1+6	00 to 07	Output range	Determines the number of bits of output data. This setting must be between 00 and 08, which sets the output range between 8 and 16 bits.
	08 to 15	Input range	Determines the number of bits of input data. This setting must be between 00 and 08, which sets the input range between 8 and 16 bits.
P1+7 to P1+32	00 to 15	Work area	Do not use. (Used by the system.)

When the execution condition is OFF, PID(—) is not executed and the instruction's data is maintained. While the execution condition is OFF, the desired output data can be written directly to OW for manual control.

When the execution condition first goes from OFF to ON, PID(—) reads the parameters and initializes the work area. There is a built-in function to change the output data continuously at startup because sudden changes in the output data might adversely affect the controlled system.

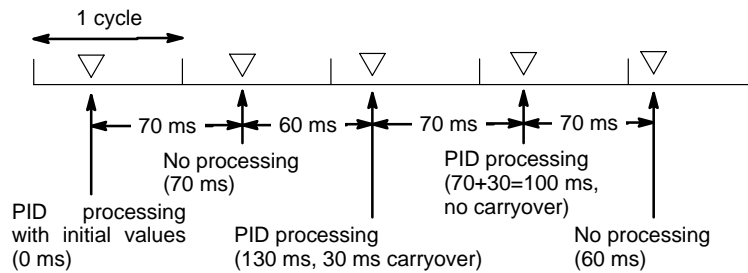
 **Caution** Changes made to the parameters will not be effective until the execution condition for PID(—) goes from OFF to ON.

**Note** Do not use PID(—) in the following situations; it may not be executed properly.

- In interrupt programs
- In subroutine programs
- In interlocked program sections (between IL and ILC)
- In jump program sections (between JMP and JME)
- In step ladder program section (created with STEP)

When the execution condition is ON, PID(—) performs the PID calculation on the input data when the sampling period has elapsed. The sampling period is the time that must pass before input data is read for processing.

The following diagram shows the relationship between the sampling period and PID processing. PID processing is performed only when the sampling period (100 ms in this case) has elapsed.



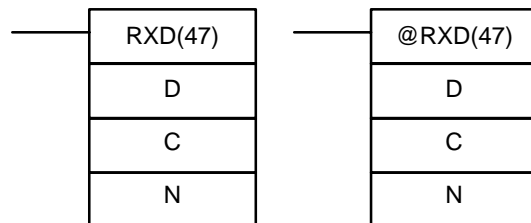
**Flags**

- ER:** There is an error in the parameter settings.  
The cycle time is more than twice as long as the sampling period, so PID(-) cannot be executed accurately. PID(-) will be executed in this case.  
  
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 PID processing has been performed. (OFF when the sampling period has not elapsed.)

## 5-27 Communications Instructions

### 5-27-1 RECEIVE – RXD(47)

**Ladder Symbols**



**Operand Data Areas**

<b>D:</b> First destination word
IR, SR, AR, DM, HR, TC, LR
<b>C:</b> Control word
#
<b>N:</b> Number of bytes
IR, SR, AR, DM, HR, TC, LR, #

**Limitations**

This instruction is available in the **CQM1/SRM1 only**.  
 D and D+(N÷2)-1 must be in the same data area.  
 DM 6144 to DM 6655 cannot be used for D or N.  
 N must be BCD from #0000 to #0256. (#0000 to #0061 in host link mode)

**Description**

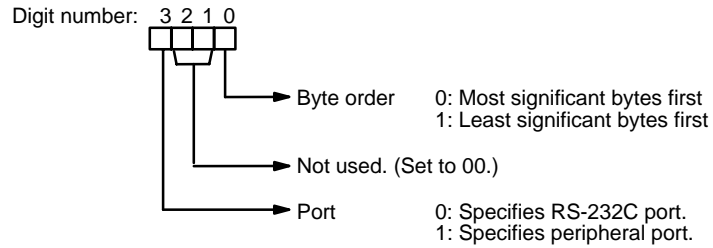
When the execution condition is OFF, RXD(47) is not executed. When the execution condition is ON, RXD(47) reads N bytes of data received at the port specified in the control word, and then writes that data in words D to D+(N÷2)-1. Up to 256 bytes of data can be read at one time.  
 If fewer than N bytes are received, the amount received will be read.

**Note** Refer to *1-9 Communications Functions* for details on using the RXD(47) instruction, setting communications protocol in the PC Setup, etc.

**Caution** The CQM1 or SRM1 will be incapable of receiving more data once 256 bytes have been received if received data is not read using RXD(47). Read data as soon as possible after the Reception Completed Flag is turned ON (AR 0806 for the RS-232C port, AR 0814 for the peripheral port.)

**Control Word**

The value of the control word determines the port from which data will be read and the order in which data will be written to memory.



The order in which data is written to memory depends on the value of digit 0 of C. Eight bytes of data 12345678... will be written in the following manner:

	MSB	LSB
D	1	2
D+1	3	4
D+2	5	6
D+3	7	8
⋮	⋮	⋮
⋮	⋮	⋮

	MSB	LSB
D	2	1
D+1	4	3
D+2	6	5
D+3	8	7
⋮	⋮	⋮
⋮	⋮	⋮

**Flags**

**ER:** The CPU Unit is not equipped with an RS-232C port.  
Another device is not connected to the specified port.

There is an error in the communications settings (PC Setup) or the operation settings.

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

The destination words (D to D+(N+2)-1) exceed the data area.

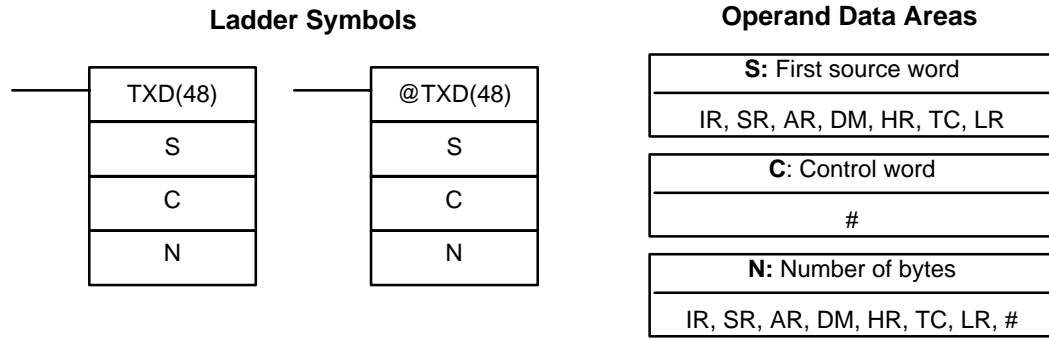
**AR 08:** AR 0806 will be turned ON when data has been received normally at the RS-232C port. Reset when RXD(47) is executed.  
AR 0814 will be turned ON when data has been received normally at the peripheral port. Reset when RXD(47) is executed.

**AR 09:** Contains the number of bytes received at the RS-232C port. Reset to 0000 when RXD(47) is executed.

**AR 10:** Contains the number of bytes received at the peripheral port. Reset to 0000 when RXD(47) is executed.

**Note** Communications flags and counters can be cleared either by specifying 0000 for N or using the Port Reset Bits (SR 25208 for peripheral port and SR 25209 for RS-232C port.)

### 5-27-2 TRANSMIT – TXD(48)



**Limitations**

This instruction is available in the **CQM1/SRM1 only**.  
 S and S+(N÷2)-1 must be in the same data area.  
 DM 6144 to DM 6655 cannot be used for S or N.  
 N must be BCD from #0000 to #0256. (#0000 to #0061 in host link mode)

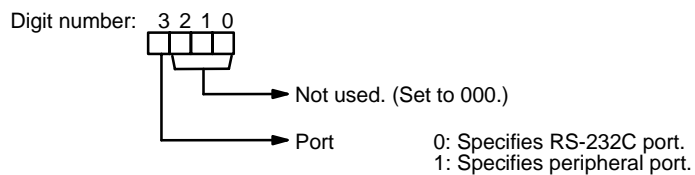
**Description**

When the execution condition is OFF, TXD(48) is not executed. When the execution condition is ON, TXD(48) reads N bytes of data from words S to S+(N÷2)-1, converts it to ASCII, and outputs the data from the specified port. TXD(48) operates differently in host link mode and RS-232C mode, so these modes are described separately.

- Note**
1. Flag AR 0805 will be ON when the CQM1 or SRM1 is capable of transmitting data through the RS-232C port and AR 0813 will be ON when the CQM1 or SRM1 is capable of transmitting data through the peripheral port.
  2. Refer to 1-9 Communications Functions for details on using the TXD(48) instruction, setting communications protocol in the PC Setup, etc.

**Host Link Mode**

N must be BCD from #0000 to #0061 (i.e., up to 122 bytes of ASCII). The value of the control word determines the port from which data will be output, as shown below.

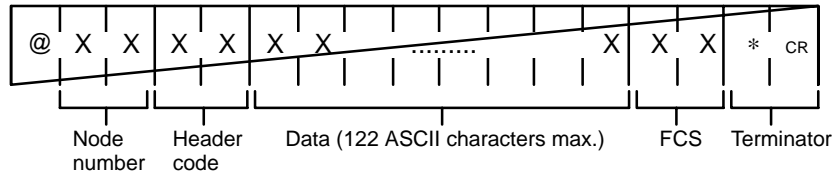


The specified number of bytes will be read from S through S+(N/2)-1, converted to ASCII, and transmitted through the specified port. The bytes of source data shown below will be transmitted in this order: 12345678...

	MSB	LSB
S	1	2
S+1	3	4
S+2	5	6
S+3	7	8
⋮	⋮	⋮
⋮	⋮	⋮



The following diagram shows the format for host link command (TXD) sent from the CQM1. The CQM1 automatically attaches the prefixes and suffixes, such as the node number, header, and FCS.

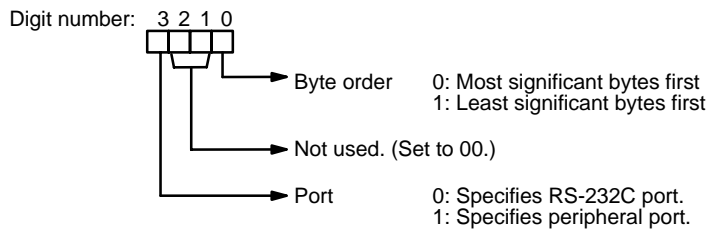


**RS-232C Mode**

N must be BCD from #0000 to #00256. The value of the control word determines the port from which data will be output and the order in which data will be written to memory.

**Control Word**

The value of the control word determines the port from which data will be read and the order in which data will be written to memory.



The specified number of bytes will be read from S through S+(NP2)-1 and transmitted through the specified port.

	MSB	LSB
S	1	2
S+1	3	4
S+2	5	6
S+3	7	8
⋮	⋮	⋮
⋮	⋮	⋮

When digit 0 of C is 0, the bytes of source data shown above will be transmitted in this order: 12345678...

When digit 0 of C is 1, the bytes of source data shown above will be transmitted in this order: 21436587...

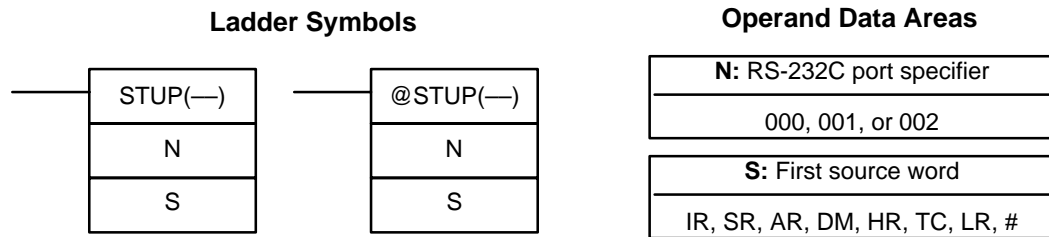
**Note** When start and end codes are specified the total data length should be 256 bytes max., including the start and end codes.

**Flags**

- ER:** The CPU Unit is not equipped with an RS-232C port.  
Another device is not connected to the peripheral port.  
  
There is an error in the communications settings (PC Setup) or the operand settings.  
  
Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)  
  
The source words (S to S+(N÷2)-1) exceed the data area.

**AR 08:** AR 0805 will be turned ON when it is possible to transmit through the RS-232C port. AR 0813 will be turned ON when it is possible to transmit through the peripheral port.

### 5-27-3 CHANGE RS-232C SETUP – STUP(—)



**Limitations**

This instruction is available in the **SRM1 only**.  
 N must be 000, 001, or 002 to specify IR 000, IR 001, or IR 002.  
 S and S+4 must be in the same data area.  
 (S can be set to #0000 to change the RS-232C settings to their defaults.)  
 STUP(—) can't be executed for the internal RS-232C port if pin 2 of the DIP switch is ON.  
 STUP(—) can't be executed within an interrupt subroutine.

**Description**

When the execution condition is OFF, STUP(—) is not executed. When the execution condition is ON, STUP(—) changes the PC Setup settings for the port specified by N.  
 N determines which part of the RS-232C Setup is changed.

N	Specified Port
IR 000	Built-in RS-232C port (PC Setup: DM 6645 to DM 6649)
IR 001	Communications Board port A (PC Setup: DM 6555 to DM 6559)
IR 002	Communications Board port B (PC Setup: DM 6550 to DM 6554)

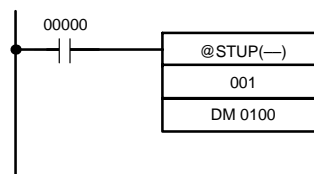
If S is a word address, the contents of S through S+4 are copied to the 5 words in the PC Setup that contain the settings for the port specified by N.

If S is input as the constant #0000, the settings for the specified port are returned to their default values.

S	Function
Word address	The contents of S through S+4 are copied to the part of the PC Setup that contains the settings for the port specified by N.
Constant (#0000)	The settings for the port specified by N are returned to their default values.

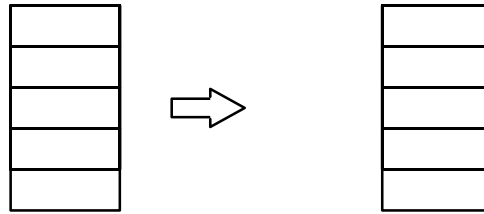
**Application Example**

This example shows a program that transfers the contents of DM 0100 through DM 0104 to the PC Setup area for Communications Board port A (DM 6555 through DM 6569).



Address	Instruction	Operands
00000	LD	00000
00001	@STUP(—)	
		001
		DM 0100

The settings are transferred as shown below. The Changing RS-232C Setup Flag (SR 27504) will be turned OFF when the transfer has been completed.



The following table shows the function of the transferred setup data.

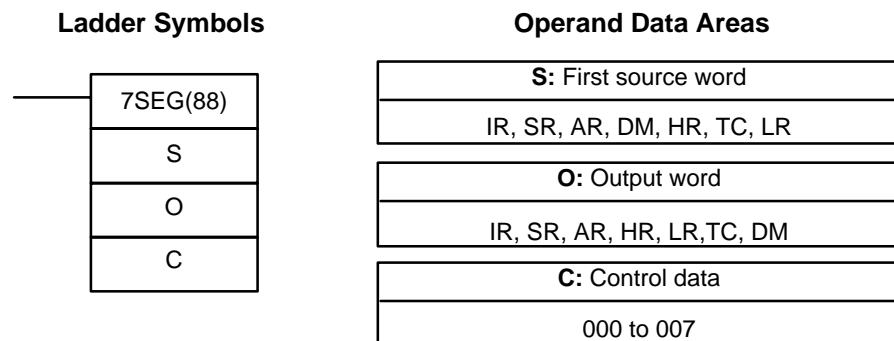
Word	Content	Function
DM 0100	1001	Enables the communications settings in DM 0101 and sets the communications mode to RS-232C.
DM 0101	0803	Sets the following communications settings: 9,600 bps, 1 start bit, 8-bit data, 1 stop bit, no parity
DM 0102	0000	No transmission delay (0 ms)
DM 0103	2000	Enables the end code CR, LF.
DM 0104	0000	---

**Flags**

- ER:** Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)  
The port specifier (N) isn't IR 000, IR 001, or IR 002.  
Port A has been specified, but pin 2 of the DIP switch is ON.  
The PC Setup is write-protected. (Pin 1 of the DIP switch is ON.)  
The specified source words exceed the data area.  
The instruction was executed from an interrupt program.

## 5-28 Advanced I/O Instructions

### 5-28-1 7-SEGMENT DISPLAY OUTPUT – 7SEG(88)



**Limitations**

This instruction is available in the **CQM1 only**.  
Do not use 7SEG(88) more than twice in the program.

**Description**

When the execution condition is OFF, 7SEG(88) is not executed. When the execution condition is ON, 7SEG(88) reads the source data (either 4 or 8-digit), converts it to 7-segment display data, and outputs that data to the 7-segment display connected to the output indicated by O.  
The value of C indicates the number of digits of source data and the logic for the Input and Output Units, as shown in the following table.

Source data	Display's data input logic	Display's latch input logic	C
4 digits (S)	Same as Output Unit	Same as Output Unit	0000
		Different from Output Unit	0001
	Different from Output Unit	Same as Output Unit	0002
		Different from Output Unit	0003
8 digits (S, S+1)	Same as Output Unit	Same as Output Unit	0004
		Different from Output Unit	0005
	Different from Output Unit	Same as Output Unit	0006
		Different from Output Unit	0007

If there are 8 digits of source data, they are placed in S and S+1, with the most significant digits placed in S+1. If there are 4 digits of source data, they are placed in S.

7SEG(88) displays the 4 or 8-digit data in 12 cycles, and then starts over and continues displaying the data.

Refer to page 125 for more information on 7SEG(88) and its applications.

**Flags**

**ER:** S and S+1 are not in the same data area. (When set to display 8-digit data.)

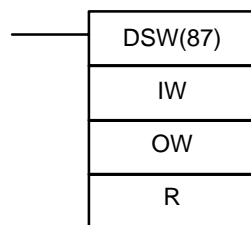
Indirectly addressed DM word is non-existent. (Content of \*DM word is not BCD, or the DM area boundary has been exceeded.)

There is an error in operand settings.

**SR 25409:** ON while 7SEG(88) is being executed.

**5-28-2 DIGITAL SWITCH INPUT – DSW(87)**

**Ladder Symbols**



**Operand Data Areas**

<b>IW:</b> Input word
IR, SR, AR, DM, HR, TC, LR
<b>OW:</b> Output word
IR, SR, AR, DM, HR, TC, LR
<b>R:</b> First result word
IR, SR, AR, DM, HR, TC, LR

**Limitations**

This instruction is available in the **CQM1 only**.

DM 6144 to DM 6655 cannot be used for R.

**Description**

DSW(87) is used to read the value set on a digital switch connected to I/O Units. When the execution condition is OFF, DSW(87) is not executed. When the execution condition is ON, DSW(87) reads the value (either 4 or 8-digit) set on the digital switch from IW and places the result in R.

If the value is an 8-digit number, it is placed in R and R+1, with the most significant digits placed in R+1. The number of digits is set in DM 6639 of the PC Setup.

DSW(87) reads the 4 or 8-digit data in 12 cycles, and then starts over and continues reading the data.

Refer to page 122 for more information on DSW(87) and its applications.

**Flags**

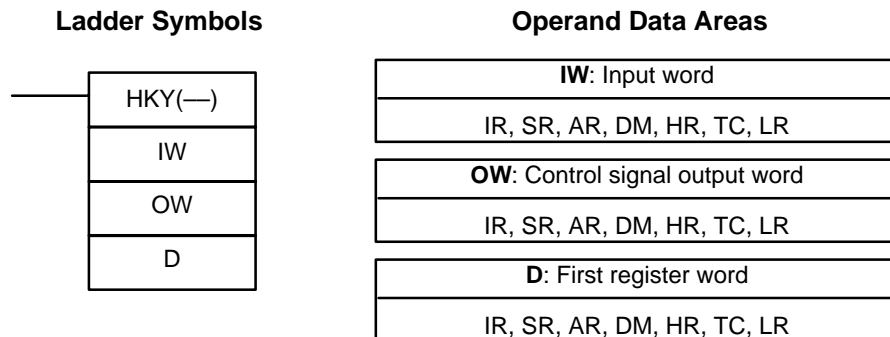
**ER:** IW and/or OW are not allocated to the correct I/O Units.

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

R and R+1 are not in the same data area. (When the CQM1 is set to receive 8-digit data.)

**SR 25410:** ON while DSW(87) is being executed.

### 5-28-3 HEXADECIMAL KEY INPUT – HKY(—)



**Limitations**

This instruction is available in the **CQM1 only**.

D and D+2 must be in the same data area.

Do not use HKY(—) more than twice in the program.

DM 6144 to DM 6655 cannot be used for D.

**Description**

When the execution condition is OFF, HKY(—) is not executed. When the execution condition is ON, HKY(—) inputs data from a hexadecimal keypad connected to the input indicated by IW. The data is input in two ways:

1. An 8-digit shift register is created in D and D+1. When a key is pressed on the hexadecimal keypad, the corresponding hexadecimal digit is shifted into the least significant digit of D. The other digits of D, D+1 are shifted left and the most significant digit of D+1 is lost.
2. The bits of D+2 and bit 4 of OW indicate key input. When one of the keys on the keypad (0 to F) is being pressed, the corresponding bit in D+2 (00 to 15) and bit 4 of OW are turned ON.

**Note** When one of the keypad keys is being pressed, input from the other keys is disabled.

HKY(—) inputs each digit in 3 to 12 cycles, and then starts over and continues inputting. Refer to page 120 for more details on HKY(—).

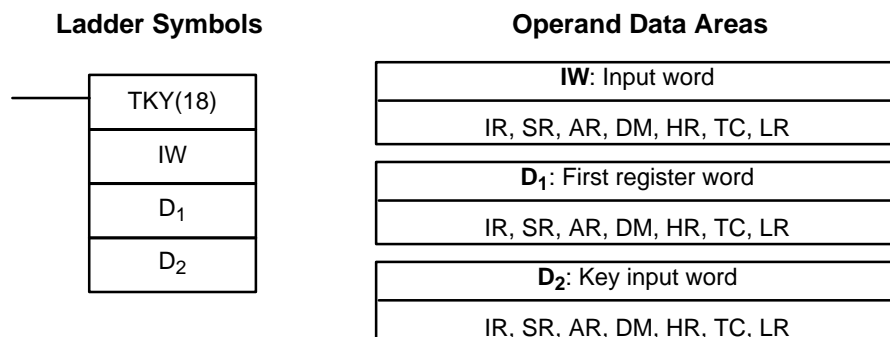
**Flags**

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

D and D+2 are not in the same data area.

**SR 25408:** ON while HKY(—) is being executed.

### 5-28-4 TEN KEY INPUT – TKY(18)



<b>Limitations</b>	This instruction is available in the <b>CQM1 only</b> . $D_1$ and $D_1+1$ must be in the same data area. DM 6143 to DM 6655 cannot be used for $D_1$ .
<b>Description</b>	<p>When the execution condition is OFF, TKY(18) is not executed. When the execution condition is ON, TKY(18) inputs data from a ten-key keypad connected to the input indicated by IW. The data is input in two ways:</p> <p><b>1, 2, 3...</b></p> <ol style="list-style-type: none"><li>1. An 8-digit shift register is created in <math>D_1</math> and <math>D_1+1</math>. When a key is pressed on the ten-key keypad, the corresponding BCD digit is shifted into the least significant digit of <math>D_1</math>. The other digits of <math>D_1</math>, <math>D_1+1</math> are shifted left and the most significant digit of <math>D_1+1</math> is lost.</li><li>2. The first ten bits of <math>D_2</math> indicate key input. When one of the keys on the keypad (0 to 9) is being pressed, the corresponding bit of <math>D_2</math> (00 to 09) is turned ON.</li></ol> <p><b>Note</b> When one of the keypad keys is being pressed, input from the other keys is disabled.</p> <p>TKY(18) can be used in several locations in the program by changing the input word, IW. Refer to page 118 for more details on TKY(18).</p>
<b>Flags</b>	<p><b>ER:</b> Indirectly addressed DM word is non-existent. (Content of *DM word is not BCD, or the DM area boundary has been exceeded.)</p> <p><math>D_1</math> and <math>D_1+1</math> are not in the same data area.</p>

# SECTION 6

## Host Link Commands

This section explains the methods and procedures for using host link commands, which can be used for host link communications via the CQM1/CPM1/CPM1A/SRM1 ports.

6-1	Communications Procedure .....	350
6-2	Command and Response Formats .....	352
6-2-1	Commands from the Host Computer .....	352
6-2-2	Commands from the PC (CQM1/SRM1 Only) .....	355
6-2-3	Response End Codes .....	355
6-3	Host Link Commands .....	356
6-3-1	IR/SR AREA READ — RR .....	356
6-3-2	LR AREA READ — RL .....	356
6-3-3	HR AREA READ — RH .....	357
6-3-4	PV READ — RC .....	357
6-3-5	TC STATUS READ — RG .....	358
6-3-6	DM AREA READ — RD .....	358
6-3-7	AR AREA READ — RJ .....	359
6-3-8	IR/SR AREA WRITE — WR .....	359
6-3-9	LR AREA WRITE — WL .....	360
6-3-10	HR AREA WRITE — WH .....	360
6-3-11	PV WRITE — WC .....	361
6-3-12	TC STATUS WRITE — WG .....	362
6-3-13	DM AREA WRITE — WD .....	362
6-3-14	AR AREA WRITE — WJ .....	363
6-3-15	SV READ 1 — R# .....	364
6-3-16	SV READ 2 — R\$ .....	364
6-3-17	SV READ 3 — R% (CQM1 Only) .....	365
6-3-18	SV CHANGE 1 — W# .....	366
6-3-19	SV CHANGE 2 — W\$ .....	367
6-3-20	SV CHANGE 3 — W% (CQM1 Only) .....	368
6-3-21	STATUS READ — MS .....	369
6-3-22	STATUS WRITE — SC .....	370
6-3-23	ERROR READ — MF .....	370
6-3-24	FORCED SET — KS .....	371
6-3-25	FORCED RESET — KR .....	372
6-3-26	MULTIPLE FORCED SET/RESET — FK .....	373
6-3-27	FORCED SET/RESET CANCEL — KC .....	374
6-3-28	PC MODEL READ — MM .....	375
6-3-29	TEST — TS .....	375
6-3-30	PROGRAM READ — RP .....	376
6-3-31	PROGRAM WRITE — WP .....	376
6-3-32	COMPOUND COMMAND — QQ .....	376
6-3-33	ABORT — XZ .....	378
6-3-34	INITIALIZE — ** .....	378
6-3-35	Undefined Command — IC .....	379

**Command Chart**

The commands listed in the chart below can be used for host link communications with the CQM1/CPM1/CPM1A/SRM1. These commands are all sent from the host computer to the PC.

Header code	Name	PC mode			Applicable PCs	Page
		RUN	MON	PRG		
RR	IR/SR AREA READ	Valid	Valid	Valid	All	356
RL	LR AREA READ	Valid	Valid	Valid	All	356
RH	HR AREA READ	Valid	Valid	Valid	All	357
RC	PV READ	Valid	Valid	Valid	All	357
RG	TC STATUS READ	Valid	Valid	Valid	All	358
RD	DM AREA READ	Valid	Valid	Valid	All	358
RJ	AR AREA READ	Valid	Valid	Valid	All	359
WR	IR/SR AREA WRITE	Not valid	Valid	Valid	All	359
WL	LR AREA WRITE	Not valid	Valid	Valid	All	360
WH	HR AREA WRITE	Not valid	Valid	Valid	All	360
WC	PV WRITE	Not valid	Valid	Valid	All	361
WG	TC STATUS WRITE	Not valid	Valid	Valid	All	362
WD	DM AREA WRITE	Not valid	Valid	Valid	All	362
WJ	AR AREA WRITE	Not valid	Valid	Valid	All	363
R#	SV READ 1	Valid	Valid	Valid	All	364
R\$	SV READ 2	Valid	Valid	Valid	All	364
R%	SV READ 3	Valid	Valid	Valid	CQM1 only	365
W#	SV CHANGE 1	Not valid	Valid	Valid	All	366
W\$	SV CHANGE 2	Not valid	Valid	Valid	All	367
W%	SV CHANGE 3	Not valid	Valid	Valid	CQM1 only	368
MS	STATUS READ	Valid	Valid	Valid	All	369
SC	STATUS WRITE	Valid	Valid	Valid	All	370
MF	ERROR READ	Valid	Valid	Valid	All	370
KS	FORCED SET	Not valid	Valid	Valid	All	371
KR	FORCED RESET	Not valid	Valid	Valid	All	372
FK	MULTIPLE FORCED SET/RESET	Not valid	Valid	Valid	All	373
KC	FORCED SET/RESET CANCEL	Not valid	Valid	Valid	All	374
MM	PC MODEL READ	Valid	Valid	Valid	All	375
TS	TEST	Valid	Valid	Valid	All	375
RP	PROGRAM READ	Valid	Valid	Valid	All	376
WP	PROGRAM WRITE	Not valid	Not valid	Valid	All	376
QQ	COMPOUND COMMAND	Valid	Valid	Valid	All	377
XZ	ABORT (command only)	Valid	Valid	Valid	All	379
**	INITIALIZE (command only)	Valid	Valid	Valid	All	379
IC	Undefined command (response only)	---	---	---	All	379

## 6-1 Communications Procedure

Host link communications are executed by means an exchange of commands and responses between the host computer and the PC.

With the CQM1, there are two communications methods that can be used. One is the normal method, in which commands are issued from the host computer to the PC. The other method allows commands to be issued from the PC to the host computer.



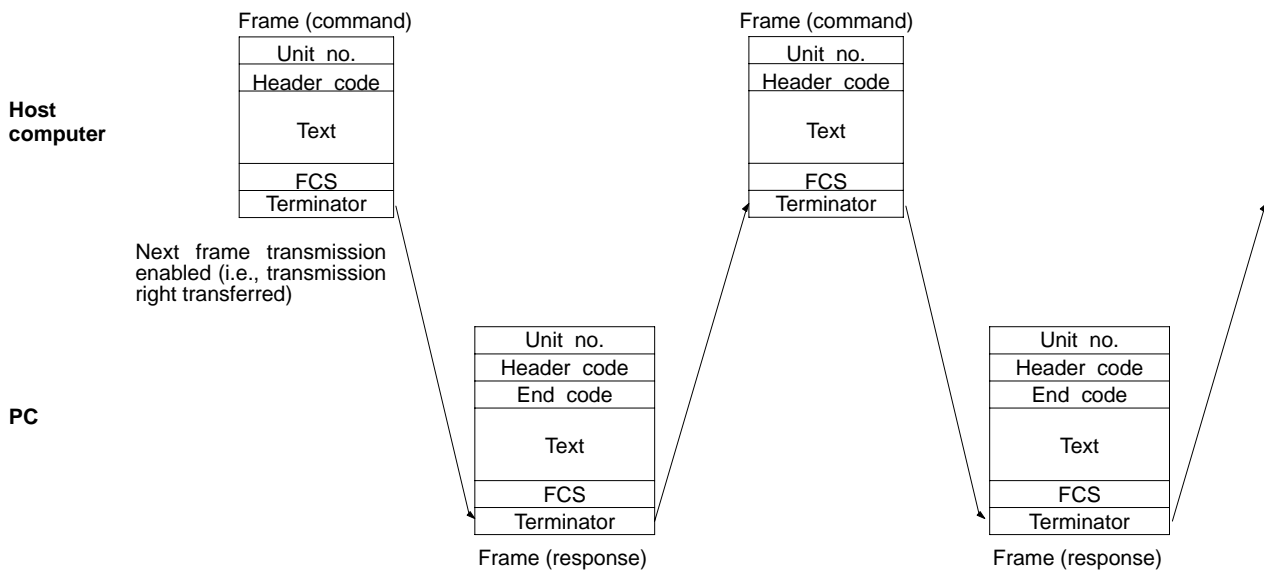
**Frame Transmission and Reception**

Commands and responses are exchanged in the order shown in the illustration below. The block of data transferred in a single transmission is called a "frame." A single frame is configured of a maximum of 131 characters of data.

The right to send a frame is called the "transmission right." The Unit that has the transmission right is the one that can send a frame at any given time. The transmission right is traded back and forth between the host computer and the PC each time a frame is transmitted. The transmission right is passed from the transmitting Unit to the receiving Unit when either a terminator (the code that marks the end of a command or response) or a delimiter (the code that sets frames apart) is received.

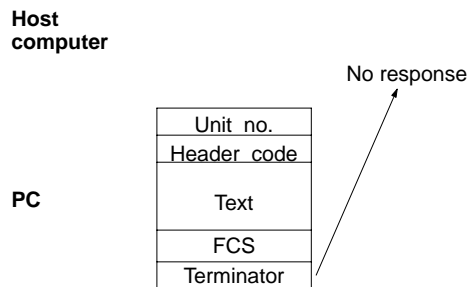
**Commands from Host**

In host link communications, the host computer ordinarily has the transmission right first and initiates the communications. The PC then automatically sends a response.



**Commands from PC (CQM1 Only)**

With CQM1 PCs, it is also possible in host link communications for the PC to send commands to the host computer. In this case it is the PC that has the transmission right and initiates the communications.



When commands are issued to the host computer, the data is transmitted in one direction from the PC to the host computer. If a response to a command is required use a host link communications command to write the response from the host computer to the PC.

## 6-2 Command and Response Formats

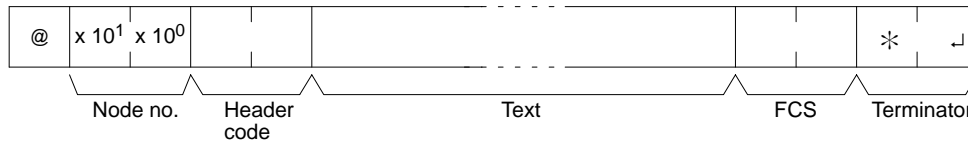
This section explains the formats for the commands and responses that are exchanged in host link communications.

### 6-2-1 Commands from the Host Computer

When a command is issued from the host computer, the command and response formats are as shown below.

#### Command Format

When transmitting a command from the host computer, prepare the command data in the format shown below.



@

An "@" symbol must be placed at the beginning.

#### Node No.

Identifies the PC communicating with the host computer.

Specify the node number set for the PC in the PC Setup (DM 6648, DM 6653).

#### Header Code

Set the 2-character command code.

#### Text

Set the command parameters.

#### FCS

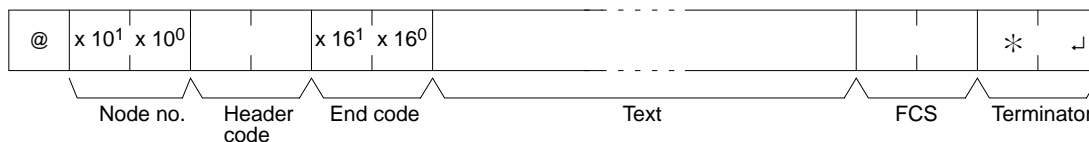
Set a 2-character Frame Check Sequence code. See page 354.

#### Terminator

Set two characters, "\*" and the carriage return (CHR\$(13)) to indicate the end of the command.

#### Response Format

The response from the PC is returned in the format shown below. Prepare a program so that the response data can be interpreted and processed.



#### @, Node No., Header Code

Contents identical to those of the command are returned.

#### End Code

The completion status of the command (e.g., whether or not an error has occurred) is returned.

#### Text

Text is returned only when there is data such as read data.

#### FCS, Terminator

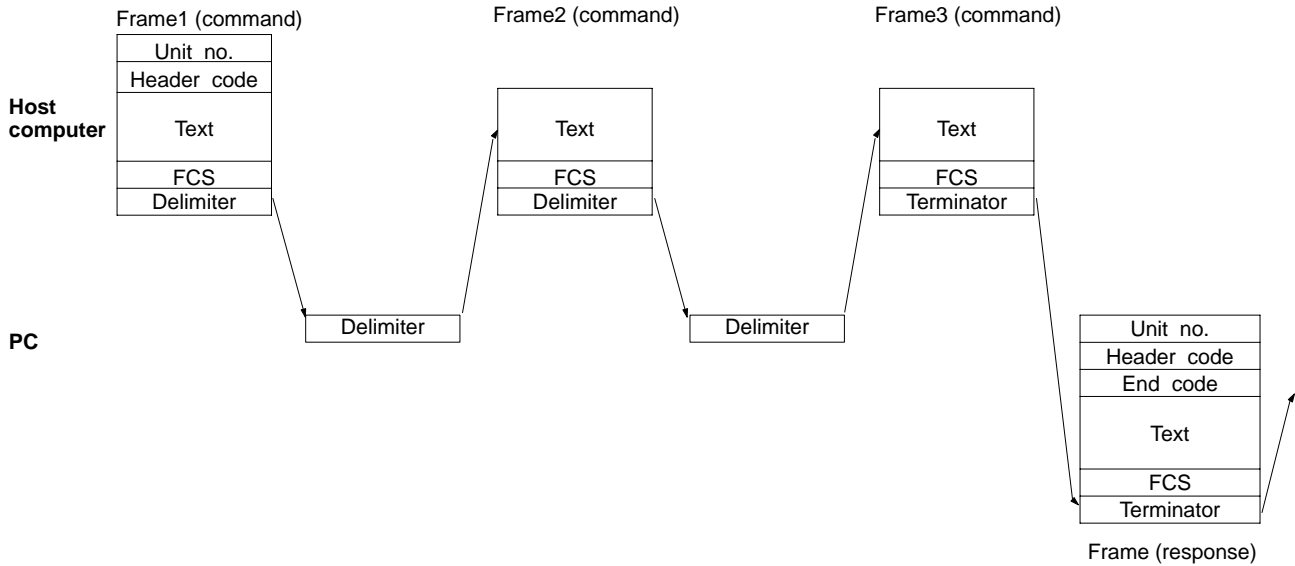
Refer to the corresponding explanations under "Command Format."

#### Long Transmissions

The largest block of data that can be transmitted as a single frame is 131 characters. A command or response of 132 characters or more must therefore be divided into more than one frame before transmission. When a transmission is split, the ends of the first and intermediate frames are marked by a delimiter instead of a terminator.

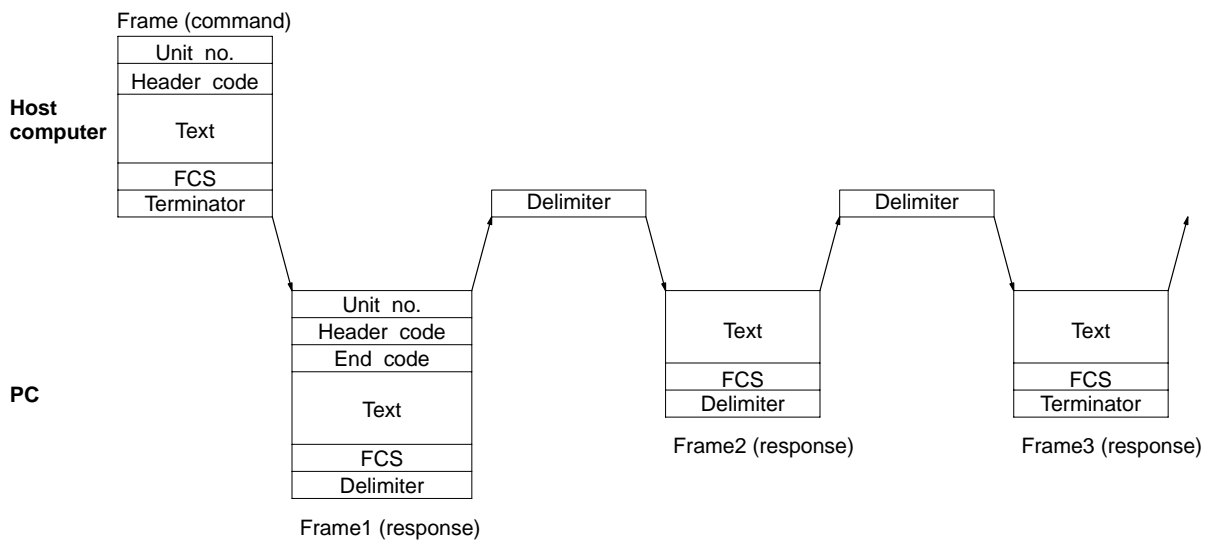
**Dividing Commands (Host Computer to PC)**

As each frame is transmitted by the host computer, the computer waits for the delimiter to be transmitted from the PC. After the delimiter has been transmitted, the next frame will then be sent. This procedure is repeated until the entire command has been transmitted.



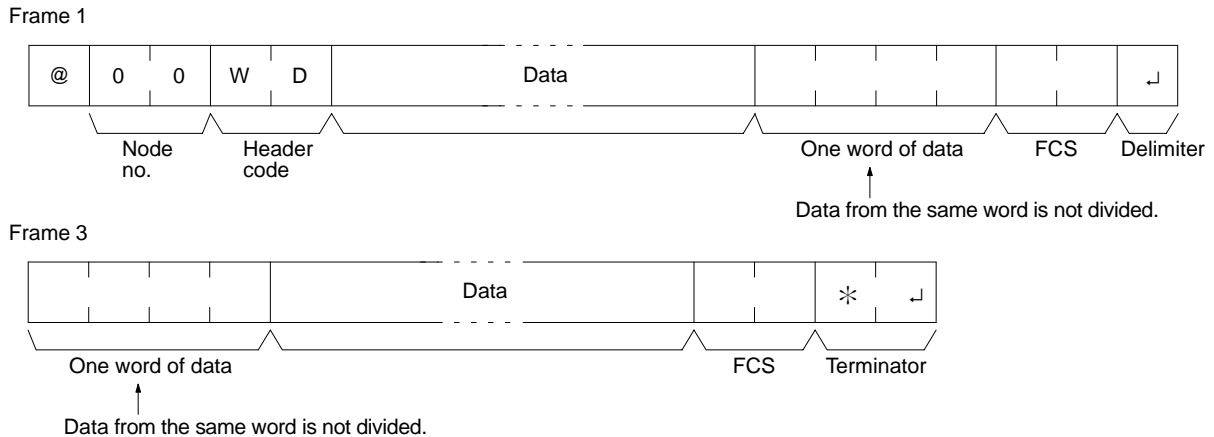
**Dividing Responses (PC to Host Computer)**

As each frame is received by the host computer, a delimiter is transmitted to the PC. After the delimiter has been transmitted, the PC will transmit the next frame. This procedure is repeated until the entire response has been transmitted.

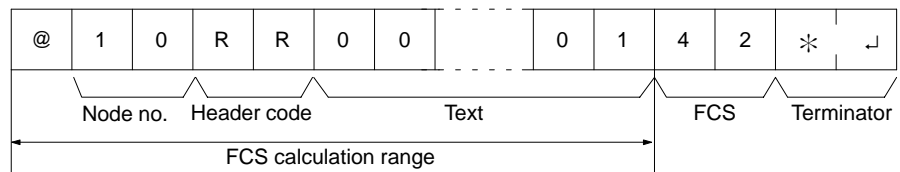


**Precautions for Long Transmissions**

When dividing commands such as WR, WL, WC, or WD that execute write operations, be careful not to divide into separate frames data that is to be written into a single word. As shown in the illustration below, be sure to divide frames so that they coincide with the divisions between words.



**FCS (Frame Check Sequence)** When a frame is transmitted, an FCS is placed just before the delimiter or terminator in order to check whether any data error has been generated. The FCS is 8-bit data converted into two ASCII characters. The 8-bit data is the result of an EXCLUSIVE OR performed on the data from the beginning of the frame until the end of the text in that frame (i.e., just before the FCS). Calculating the FCS each time a frame is received and checking the result against the FCS that is included in the frame makes it possible to check for data errors in the frame.



		ASCII code		
@	40	0100	0000	
			XOR	
1	31	0011	0001	
			XOR	
0	30	0011	0000	
			XOR	
R	52	0101	0010	
			⋮	
1	31	0011	0001	
Calculation result		0100	0010	
		↓	↓	Converted to hexadecimal. Handled as ASCII characters.
		4	2	

**Example Program for FCS**

This example shows a BASIC subroutine program for executing an FCS check on a frame received by the host computer.

```

400 *FCSCHECK
410 L=LEN(RESPONSE$) ' . . . . . Data transmitted and received
420 Q=0:FCCK$=""
430 A$=RIGHT$(RESPONSE$,L)
440 PRINT RESPONSE$,A$,L
450 IF A$="*" THEN LENG$=LEN(RESPONSE$)-3
      ELSE LENG$=LEN(RESPONSE$)-2
460 FCSP$=MID$(RESPONSE$,LENG$+1,2) ' . . . FCS data received
470 FOR I=1 TO LENG$ ' . . . . . Number of characters in FCS
480 Q=ASC(MID$(RESPONSE$,I,1)) XOR Q
490 NEXT I
500 FCSD$=HEX$(Q)
510 IF LEN(FCSD$)=1 THEN FCSD$="0"+FCSD$ ' . . . . FCS result
520 IF FCSD$<>FCSP$ THEN FCCK$="ERR"
530 PRINT"FCSD$=";FCSD$,"FCSP$=";FCSP$,"FCCK$=";FCCK$
540 RETURN
    
```

- Note**
1. Normal reception data includes the FCS, delimiter or terminator, and so on. When an error occurs in transmission, however the FCS or some other data may not be included. Be sure to program the system to cover this possibility.
  2. In this program example, the CR code (CHR\$(13)) is not entered for RESPONSE\$. When including the CR code, make the changes in lines 430 and 450.

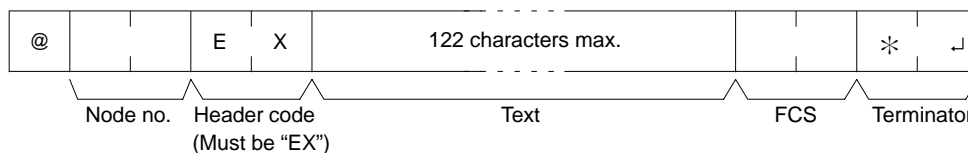
**6-2-2 Commands from the PC (CQM1/SRM1 Only)**

In host link communications, commands are ordinarily sent from the host computer to the PC, but it is also possible for commands to be sent from the PC to the host computer. In Host Link Mode, any data can be transmitted from the PC to the host computer. To send a command to the host computer, use the TRANSMIT instruction (TXD(48)) in the PC program in Host Link Mode.

TXD(48) outputs data from the specified port (the RS-232C port or the peripheral port). Refer to page 342 for details on using TXD(48).

**Reception Format**

When TXD(48) is executed, the data stored in the words beginning with the first send word is converted to ASCII and output to the host computer as a host link command in the format shown below. The "@" symbol, node number, header code, FCS, and delimiter are all added automatically when the transmission is sent. At the host computer, it is necessary to prepare in advance a program for interpreting and processing this format.



One byte of data (2 digits hexadecimal) is converted to two characters in ASCII for transmission, the amount of data in the transmission is twice the amount of words specified for TXD(48). The maximum number of characters for transmission is 122 and the maximum number of bytes that can be designated for TXD(48) is one half of that, or 61.

**6-2-3 Response End Codes**

Refer to 8-7 Host Link Errors for a table listing the response end codes that may be returned in host link communications. An end code of 00 indicates normal completion of a command.

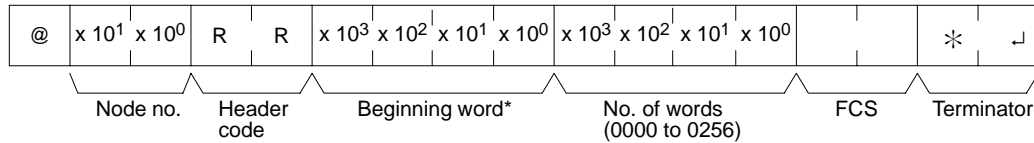
## 6-3 Host Link Commands

This section explains the commands that can be issued from the host computer to the PC.

### 6-3-1 IR/SR AREA READ — RR

Reads the contents of the specified number of IR and SR words, starting from the specified word.

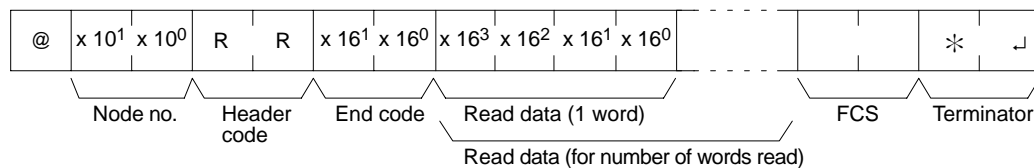
#### Command Format



**Note** Beginning word: 0000 to 0255 in CQM1 PCs, 0000 to 0019 and 0200 to 0255 in CPM1/CPM1A/SRM1 PCs.

#### Response Format

An end code of 00 indicates normal completion.



- Note**
1. Words 0020 to 0199 in CPM1/CPM1A/SRM1 PCs cannot be specified. If an attempt to read any of these words is made, a response of 0000 will be returned.
  2. The response will be divided when reading more than 30 words of data.

#### Parameters

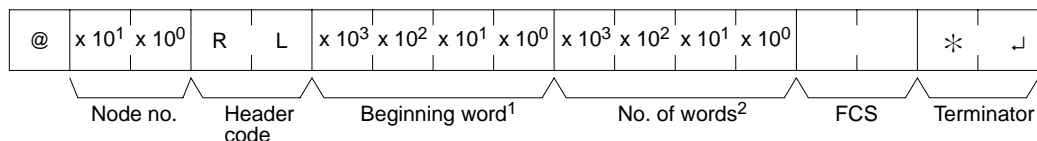
##### Read Data (Response)

The contents of the number of words specified by the command are returned in hexadecimal as a response. The words are returned in order, starting with the specified beginning word.

### 6-3-2 LR AREA READ — RL

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

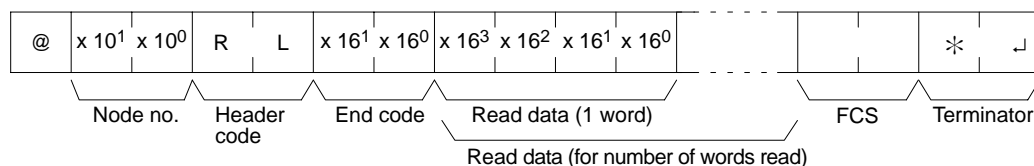
#### Command Format



- Note**
1. Beginning word: 0000 to 0063 in CQM1 PCs, 0000 to 0015 in CPM1/CPM1A/SRM1 PCs
  2. No. of words: 0001 to 0064 in CQM1 PCs, 0001 to 0016 in CPM1/CPM1A/SRM1 PCs

#### Response Format

An end code of 00 indicates normal completion.



Parameters

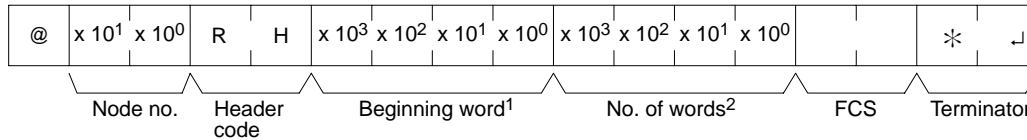
**Read Data (Response)**

The contents of the number of words specified by the command are returned in hexadecimal as a response. The words are returned in order, starting with the specified beginning word.

**6-3-3 HR AREA READ — RH**

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

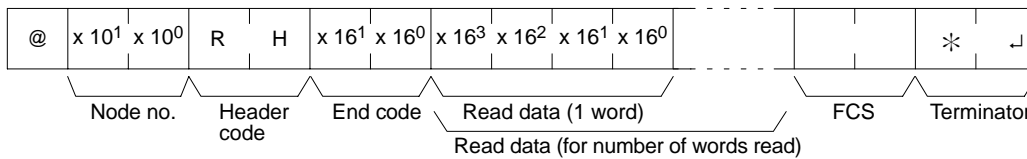
**Command Format**



- Note**
1. Beginning word: 0000 to 0099 in CQM1 PCs, 0000 to 0019 in CPM1/CPM1A/SRM1 PCs
  2. No. of words: 0001 to 0100 in CQM1 PCs, 0001 to 0020 in CPM1/CPM1A/SRM1 PCs

**Response Format**

An end code of 00 indicates normal completion.



Parameters

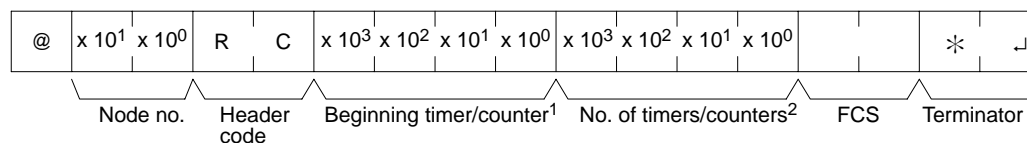
**Read Data (Response)**

The contents of the number of words specified by the command are returned in hexadecimal as a response. The words are returned in order, starting with the specified beginning word.

**6-3-4 PV READ — RC**

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

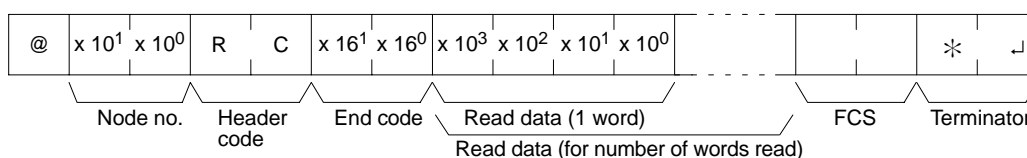
**Command Format**



- Note**
1. Beginning T/C: 0000 to 0511 in CQM1 PCs, 0000 to 0127 in CPM1/CPM1A/SRM1 PCs
  2. No. of T/Cs: 0001 to 0512 in CQM1 PCs, 0001 to 0128 in CPM1/CPM1A/SRM1 PCs

**Response Format**

An end code of 00 indicates normal completion.



The response will be divided when reading more than 30 words of data.

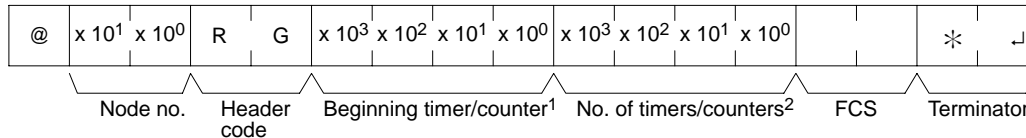
## Parameters

**Read Data (Response)**

The number of present values specified by the command is returned in hexadecimal as a response. The PVs are returned in order, starting with the specified beginning timer/counter.

**6-3-5 TC STATUS READ — RG**

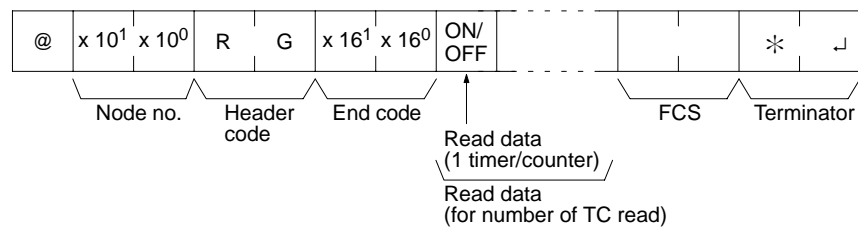
Reads the status of the Completion Flags of the specified number of timers/counters, starting from the specified timer/counter.

**Command Format**

- Note**
1. Beginning T/C: 0000 to 0511 in CQM1 PCs, 0000 to 0127 in CPM1/CPM1A/SRM1 PCs
  2. No. of T/Cs: 0001 to 0512 in CQM1 PCs, 0001 to 0128 in CPM1/CPM1A/SRM1 PCs

**Response Format**

An end code of 00 indicates normal completion.



The response will be divided when reading the status of more than 123 timer/counters.

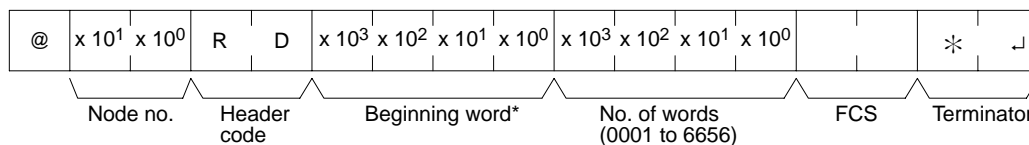
## Parameters

**Read Data (Response)**

The status of the number of Completion Flags specified by the command is returned as a response. "1" indicates that the Completion Flag is ON.

**6-3-6 DM AREA READ — RD**

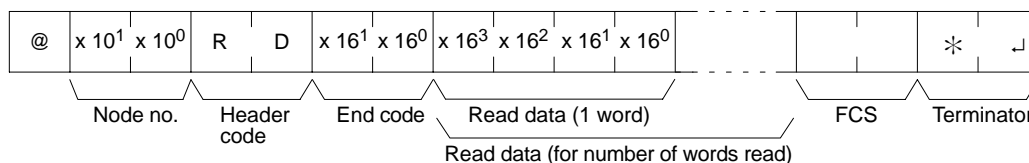
Reads the contents of the specified number of DM words, starting from the specified word.

**Command Format**

- Note** Beginning word: 0000 to 6655 in CQM1 PCs, 0000 to 1023 and 6144 to 6655 in CPM1/CPM1A PCs, and 0000 to 2047 and 6144 to 6655 in SRM1 PCs.

**Response Format**

An end code of 00 indicates normal completion.





- Note**
1. Words 1024 to 6143 in CPM1/CPM1A PCs and words 2048 to 6143 in SRM1 PCs cannot be specified. If an attempt to read any of these words is made, a response of 0000 will be returned.
  2. The response will be divided when reading more than 30 words of data.

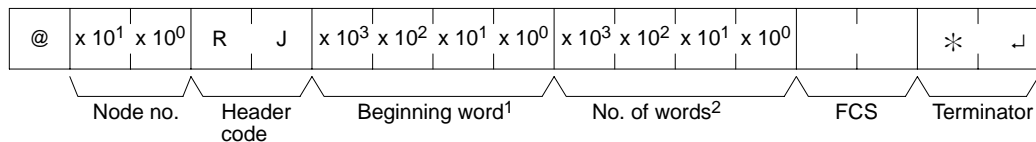
**Parameters****Read Data (Response)**

The contents of the number of words specified by the command are returned in hexadecimal as a response. The words are returned in order, starting with the specified beginning word.

- Note** Be careful about the configuration of the DM area, as it varies depending on the CPU Unit model.

**6-3-7 AR AREA READ — RJ**

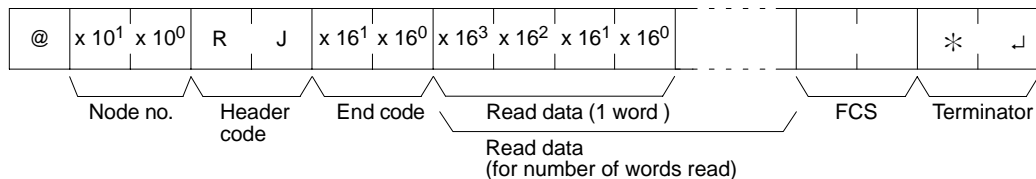
Reads the contents of the specified number of AR words, starting from the specified word.

**Command Format**

- Note**
1. Beginning word: 0000 to 0027 in CQM1 PCs, 0000 to 0015 in CPM1/CPM1A/SRM1 PCs
  2. No. of words: 0001 to 0028 in CQM1 PCs, 0001 to 0016 in CPM1/CPM1A/SRM1 PCs

**Response Format**

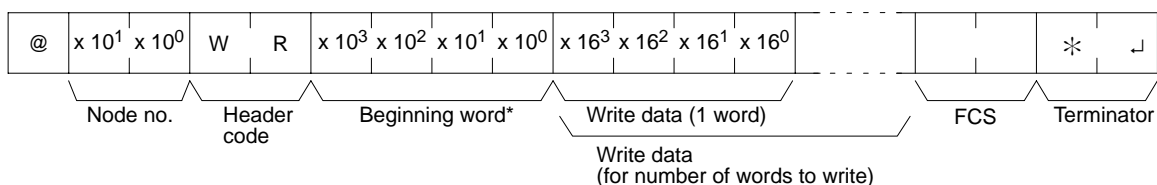
An end code of 00 indicates normal completion.

**Parameters****Read Data (Response)**

The contents of the number of words specified by the command are returned in hexadecimal as a response. The words are returned in order, starting with the specified beginning word.

**6-3-8 IR/SR AREA WRITE — WR**

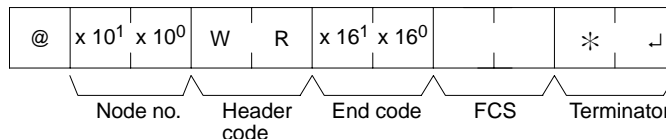
Writes data to the IR and SR areas, starting from the specified word. Writing is done word by word.

**Command Format**

- Note**
1. Beginning word: 0000 to 0252 in CQM1 PCs, 0000 to 0019 and 0200 to 0252 in CPM1/CPM1A/SRM1 PCs.
  2. Divide the command when writing more than 30 words of data.

**Response Format**

An end code of 00 indicates normal completion.



**Note** Words 0020 to 0199 in CPM1/CPM1A/SRM1 PCs cannot be specified. If an attempt to write to any of these words is made, the writing operation will not be executed and normal completion occurs.

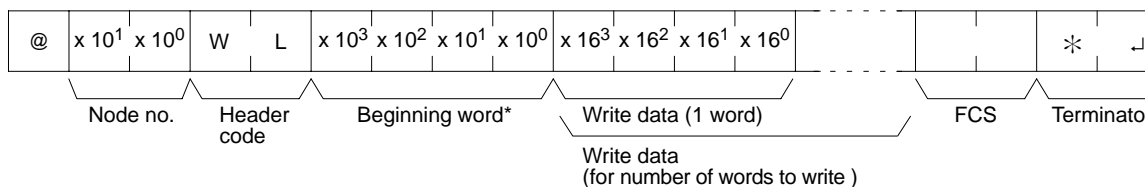
**Parameters****Write Data (Command)**

Specify in order the contents of the number of words to be written to the IR or SR area in hexadecimal, starting with the specified beginning word.

**Note** If data is specified for writing which exceeds the allowable range, an error will be generated and the writing operation will not be executed. If, for example, 252 is specified as the beginning word for writing, and two words of data are specified, then 253 will become the last word for writing data, and the command will not be executed because SR 253 is beyond the writable range.

**6-3-9 LR AREA WRITE — WL**

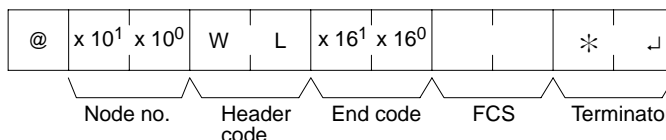
Writes data to the LR area, starting from the specified word. Writing is done word by word.

**Command Format**

**Note** Beginning word: 0000 to 0063 in CQM1 PCs, 0000 to 0015 in CPM1/CPM1A/SRM1 PCs

**Response Format**

An end code of 00 indicates normal completion.

**Parameters****Write Data (Command)**

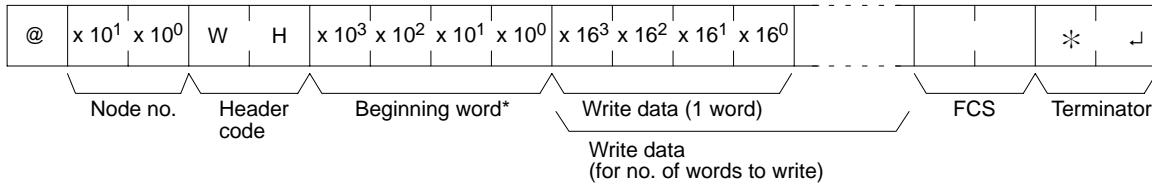
Specify in order the contents of the number of words to be written to the LR area in hexadecimal, starting with the specified beginning word.

**Note** If data is specified for writing which exceeds the allowable range, an error will be generated and the writing operation will not be executed. If, for example, 60 is specified as the beginning word for writing and five words of data are specified, then 64 will become the last word for writing data, and the command will not be executed because LR 64 is beyond area boundary.

**6-3-10 HR AREA WRITE — WH**

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

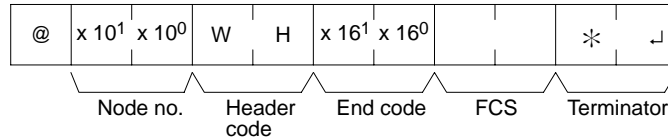
**Command Format**



**Note** Beginning word: 0000 to 0063 in CQM1 PCs, 0000 to 0019 in CPM1/CPM1A/SRM1 PCs

**Response Format**

An end code of 00 indicates normal completion.



**Parameters**

**Write Data (Command)**

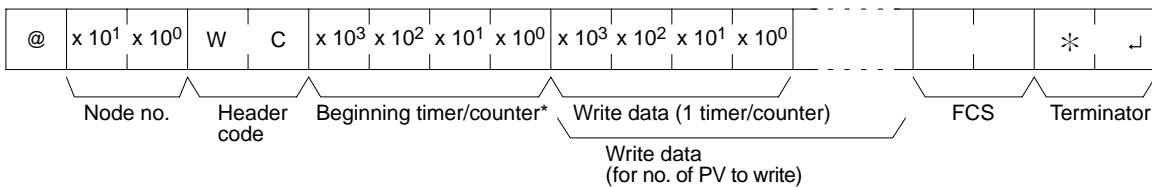
Specify in order the contents of the number of words to be written to the HR area in hexadecimal, starting with the specified beginning word.

**Note** If data is specified for writing which exceeds the allowable range, an error will be generated and the writing operation will not be executed. If, for example, 98 is specified as the beginning word for writing, and three words of data are specified, then 100 will become the last word for writing data, and the command will not be executed because HR 100 is beyond area boundary.

**6-3-11 PV WRITE — WC**

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

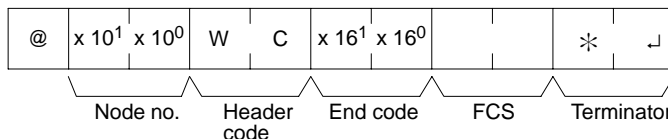
**Command Format**



**Note** 1. Beginning T/C: 0000 to 0511 in CQM1 PCs, 0000 to 0127 in CPM1/CPM1A/SRM1 PCs  
2. Divide the command when writing more than 29 words of data.

**Response Format**

An end code of 00 indicates normal completion.



**Parameters**

**Write Data (Command)**

Specify in decimal numbers (BCD) the present values for the number of timers/counters that are to be written, starting from the beginning timer/counter.

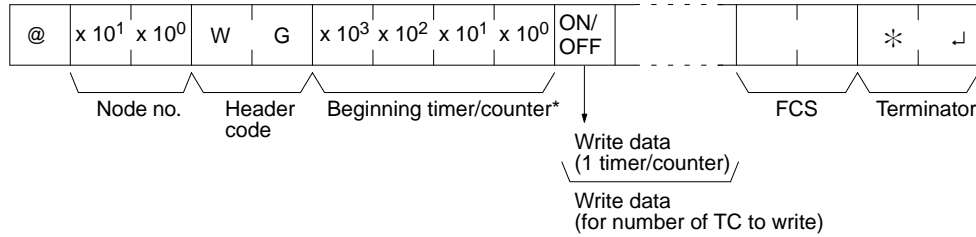
**Note** 1. When this command is used to write data to the PV area, the Completion Flags for the timers/counters that are written will be turned OFF.

- If data is specified for writing which exceeds the allowable range, an error will be generated and the writing operation will not be executed. If, for example, 510 is specified as the beginning word for writing, and three words of data are specified, then 512 will become the last word for writing data, and the command will not be executed because TC 512 is beyond area boundary.

### 6-3-12 TC STATUS WRITE — WG

Writes the status of the Completion Flags for timers and counters in the TC area, starting from the specified timer/counter (number). Writing is done number by number.

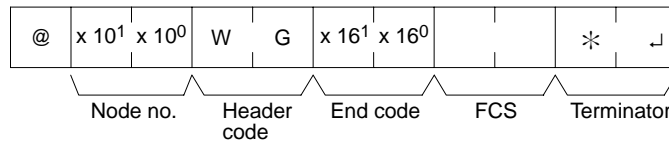
#### Command Format



- Note**
- Beginning T/C: 0000 to 0511 in CQM1 PCs, 0000 to 0127 in CPM1/CPM1A/SRM1 PCs
  - Divide the command when writing the status of more than 118 timer/counters.

#### Response Format

An end code of 00 indicates normal completion.



#### Parameters

##### Write Data (Command)

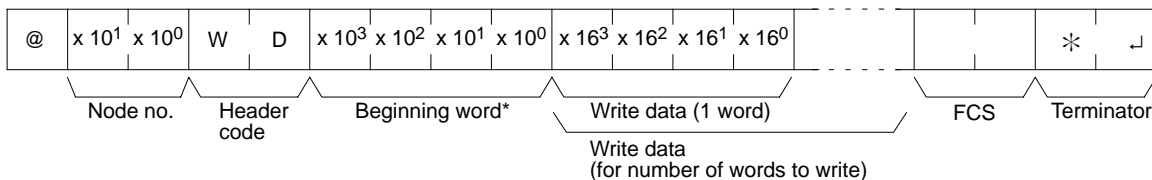
Specify the status of the Completion Flags, for the number of timers/counters to be written, in order (from the beginning word) as ON (i.e., "1") or OFF (i.e., "0"). When a Completion Flag is ON, it indicates that the time or count is up.

- Note** If data is specified for writing which exceeds the allowable range, an error will be generated and the writing operation will not be executed. If, for example, 510 is specified as the beginning word for writing, and three words of data are specified, then 512 will become the last word for writing data, and the command will not be executed because TC 512 is beyond area boundary.

### 6-3-13 DM AREA WRITE — WD

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

#### Command Format

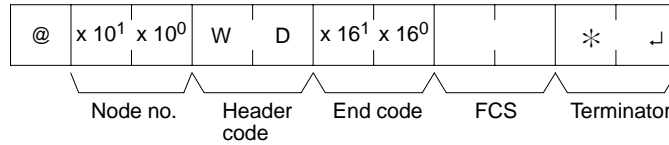


- Note**
- Beginning word: 0000 to 6143 in CQM1 PCs, 0000 to 1023 and 6144 to 6655 in CPM1/CPM1A PCs, and 0000 to 2047 and 6144 to 6655 in SRM1 PCs

2. Divide the command when writing more than 29 words of data.

**Response Format**

An end code of 00 indicates normal completion.



**Note** DM 1024 to DM 6143 in CPM1/CPM1A PCs and DM 2048 to DM 6143 in SRM1 PCs cannot be specified. If an attempt to write to any of these words is made, the writing operation will not be executed for these words and the command will end normally

**Parameters**

**Write Data (Command)**

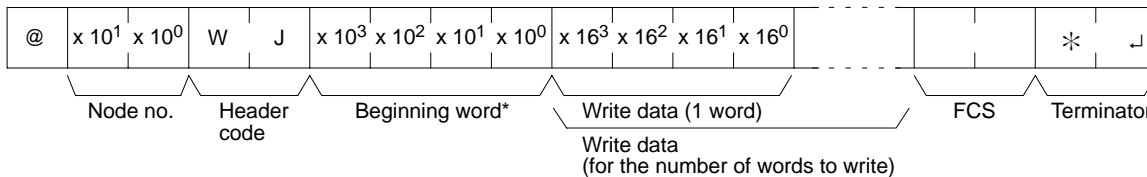
Specify in order the contents of the number of words to be written to the DM area in hexadecimal, starting with the specified beginning word.

- Note**
1. If data is specified for writing which exceeds the allowable range, an error will be generated and the writing operation will not be executed. If, for example, 6142 is specified as the beginning word for writing, and three words of data are specified, then 6144 will become the last word for writing data, and the command will not be executed because DM 6144 is beyond the writable range.
  2. Be careful about the configuration of the DM area, as it varies depending on the CPU Unit model.

**6-3-14 AR AREA WRITE — WJ**

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

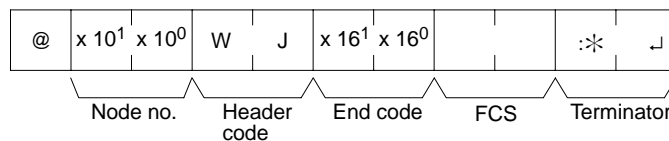
**Command Format**



**Note** Beginning word: 0000 to 0027 in CQM1 PCs, 0000 to 0015 in CPM1/CPM1A and SRM1 PCs

**Response Format**

An end code of 00 indicates normal completion.



**Parameters**

**Write Data (Command)**

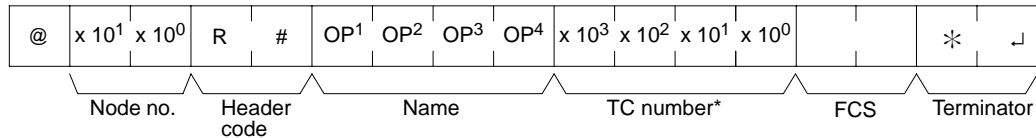
Specify in order the contents of the number of words to be written to the AR area in hexadecimal, starting with the specified beginning word.

**Note** If data is specified for writing which exceeds the allowable range, an error will be generated and the writing operation will not be executed. If, for example, 26 is specified as the beginning word for writing, and three words of data are specified, then 28 will become the last word for writing data, and the command will not be executed because AR 28 is beyond the writable range.

### 6-3-15 SV READ 1 — R#

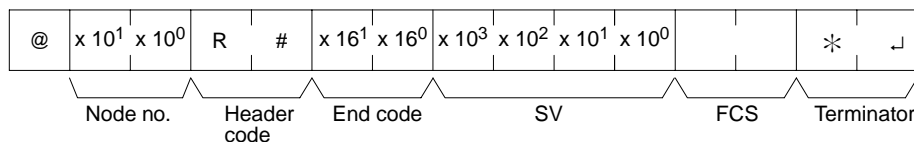
Searches for the first instance of a TIM, TIMH(15), CNT, and CNTR(12) instruction with the specified TC number in the user's program and reads the PV, which assumed to be set as a constant. The SV that is read is a 4-digit decimal number (BCD). The program is searched from the beginning, which may take as much as 10 seconds to produce a response.

#### Command Format



**Note** TC number: 0000 to 0511 in CQM1 PCs, 0000 to 0127 in CPM1/CPM1A/SRM1 PCs

#### Response Format



#### Parameters

##### Name, TC Number (Command)

Specify the instruction for reading the SV in "Name." Make this setting in 4 characters. In "TC number," specify the timer/counter number used for the instruction.

Instruction name				Classification
OP1	OP2	OP3	OP4	
T	I	M	(Space)	TIMER
T	I	M	H	HIGH-SPEED TIMER
C	N	T	(Space)	COUNTER
C	N	T	R	REVERSIBLE COUNTER

##### SV (Response)

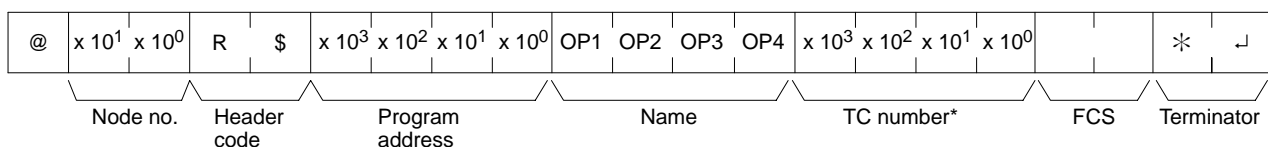
The constant SV is returned.

- Note**
1. The instruction specified under "Name" must be in four characters.
  2. If the same instruction is used more than once in a program, only the first one will be read.
  3. Use this command only when it is definite that a constant SV has been set.
  4. The response end code will indicate an error (16) if the SV wasn't entered as a constant.

### 6-3-16 SV READ 2 — R\$

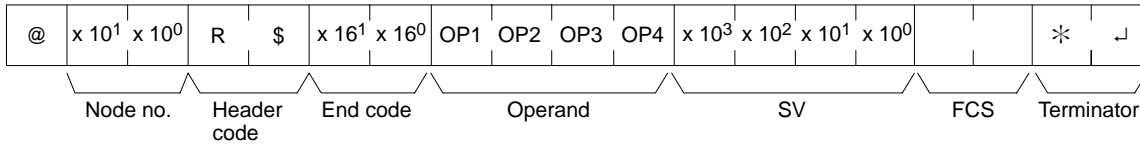
Reads the constant SV or the word address where the SV is stored. The SV that is read is a 4-digit decimal number (BCD) written as the second operand for the TIM, TIMH(15), CNT, or CNTR(12) instruction at the specified program address in the user's program. This can only be done with a program of less than 10K.

#### Command Format



**Note** TC number: 0000 to 0511 in CQM1 PCs, 0000 to 0127 in CPM1/CPM1A/SRM1 PCs

**Response Format** An end code of 00 indicates normal completion.



**Parameters**

**Name, TC Number (Command)**

Specify the name of the instruction for reading the SV in “Name.” Make this setting in 4 characters. In “TC number,” specify the timer/counter number used by the instruction.

Instruction name				Classification
OP1	OP2	OP3	OP4	
T	I	M	(Space)	TIMER
T	I	M	H	HIGH-SPEED TIMER
C	N	T	(Space)	COUNTER
C	N	T	R	REVERSIBLE COUNTER

**Operand, SV (Response)**

The name that indicates the SV classification is returned to “Operand,” and either the word address where the SV is stored or the constant SV is returned to “SV.”

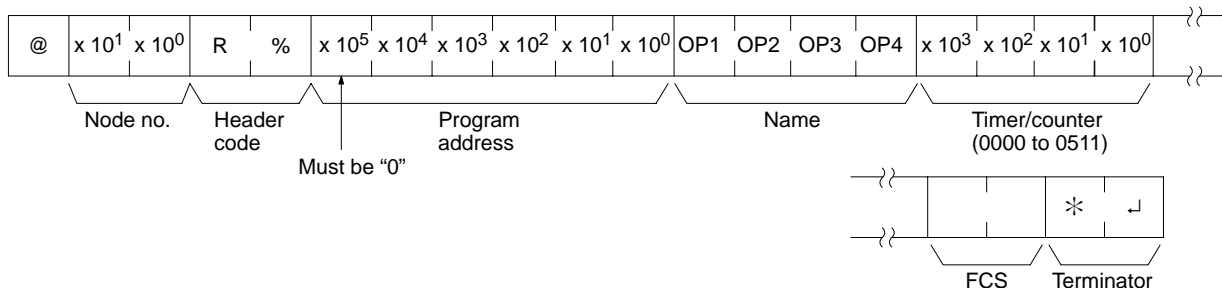
Operand				Classification	Constant or word address	
OP1	OP2	OP3	OP4		CQM1 PCs	CPM1 PCs
C	I	O	(Space)	IR or SR	0000 to 0255	0000 to 0019 0200 to 0255
L	R	(Space)	(Space)	LR	0000 to 0063	0000 to 0015
H	R	(Space)	(Space)	HR	0000 to 0099	0000 to 0019
A	R	(Space)	(Space)	AR	0000 to 0027	0000 to 0015
D	M	(Space)	(Space)	DM	0000 to 6655	0000 to 6655
D	M	*	(Space)	DM (indirect)	0000 to 6655	0000 to 6655
C	O	N	(Space)	Constant	0000 to 9999	0000 to 9999

**Note** The instruction name specified under “Name” must be in four characters. Fill any gaps with spaces to make a total of four characters.

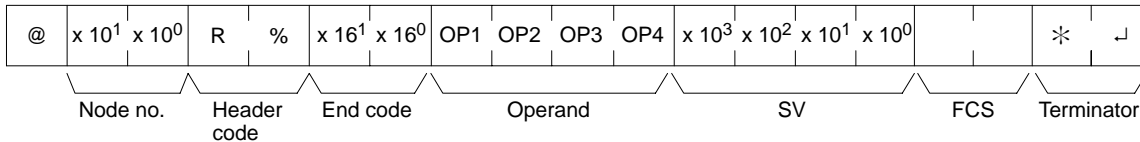
**6-3-17 SV READ 3 — R% (CQM1 Only)**

Reads the constant SV or the word address where the SV is stored. The SV that is read is a 4-digit decimal number (BCD) written in the second word of the TIM, TIMH(15), CNT, or CNTR(12) instruction at the specified program address in the user’s program. With this command, program addresses can be specified for a program of 10K or more.

**Command Format**



**Response Format** An end code of 00 indicates normal completion.



**Parameters**

**Name, TC Number (Command)**

Specify the name of the instruction for reading the SV in "Name." Make this setting in 4 characters. In "TC number," specify the timer/counter number used by the instruction.

Instruction name				Classification	TC number range
OP1	OP2	OP3	OP4		
T	I	M	(Space)	TIMER	0000 to 0511
T	I	M	H	HIGH-SPEED TIMER	
C	N	T	(Space)	COUNTER	
C	N	T	R	REVERSIBLE COUNTER	

**Operand, SV (Response)**

The name that indicates the SV classification is returned to "Operand," and either the word address where the SV is stored or the constant SV is returned to "SV."

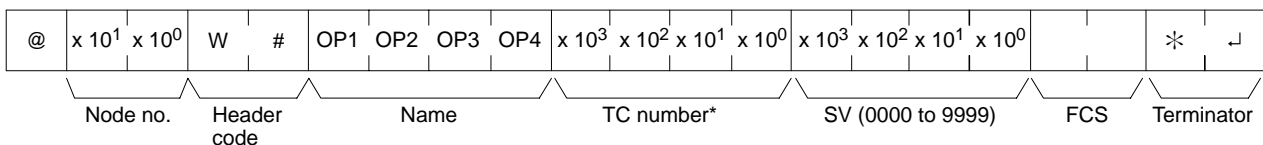
Operand				Classification	Constant or word address
OP1	OP2	OP3	OP4		
C	I	O	(Space)	IR or SR	0000 to 0255
L	R	(Space)	(Space)	LR	0000 to 0063
H	R	(Space)	(Space)	HR	0000 to 0099
A	R	(Space)	(Space)	AR	0000 to 0027
D	M	(Space)	(Space)	DM	0000 to 6655
D	M	*	(Space)	DM (indirect)	0000 to 6655
C	O	N	(Space)	Constant	0000 to 9999

**Note** The instruction name specified under "Name" must be in four characters. Fill any gaps with spaces to make a total of four characters.

**6-3-18 SV CHANGE 1 — W#**

Searches for the first instance of the specified TIM, TIMH(15), CNT, or CNTR(12) instruction in the user's program and changes the SV to new constant SV specified in the second word of the instruction. The program is searched from the beginning, and it may therefore take approximately 10 seconds to produce a response.

**Command Format**

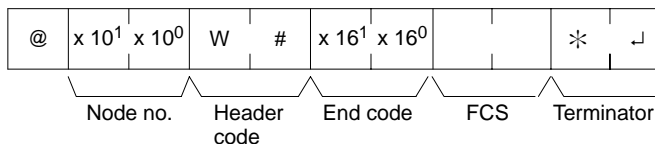


**Note** TC number: 0000 to 0511 in CQM1 PCs, 0000 to 0127 in CPM1/CPM1A/SRM1 PCs



**Response Format**

An end code of 00 indicates normal completion.



**Parameters**

**Name, TC Number (Command)**

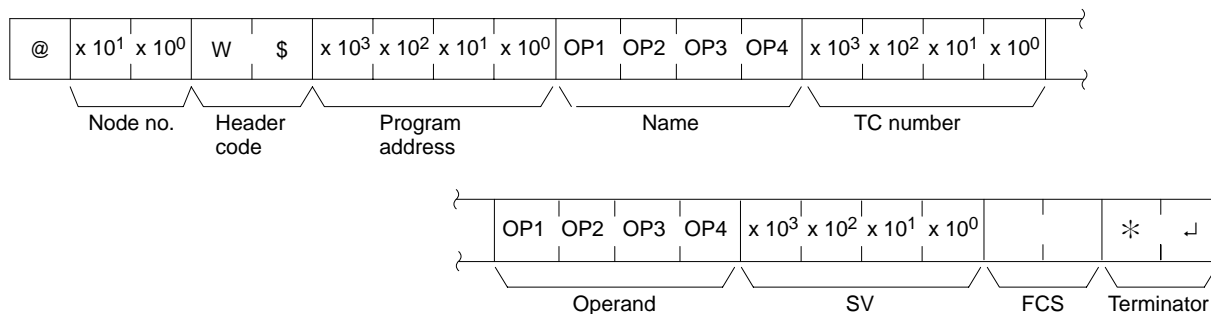
In "Name," specify the name of the instruction, in four characters, for changing the SV. In "TC number," specify the timer/counter number used for the instruction.

Instruction name				Classification
OP1	OP2	OP3	OP4	
T	I	M	(Space)	TIMER
T	I	M	H	HIGH-SPEED TIMER
C	N	T	(Space)	COUNTER
C	N	T	R	REVERSIBLE COUNTER

**6-3-19 SV CHANGE 2 — W\$**

Changes the contents of the second word of the TIM, TIMH(15), CNT, or CNTR(12) at the specified program address in the user's program. This can only be done with a program of less than 10K.

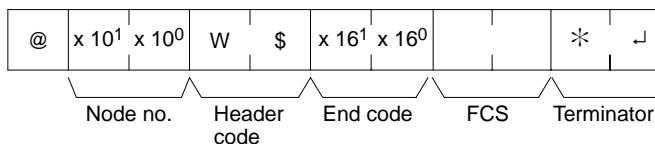
**Command Format**



**Note** TC number: 0000 to 0511 in CQM1 PCs, 0000 to 0127 in CPM1/CPM1A/SRM1 PCs

**Response Format**

An end code of 00 indicates normal completion.



**Parameters**

**Name, TC Number (Command)**

In "Name," specify the name of the instruction, in four characters, for changing the SV. In "TC number," specify the timer/counter number used for the instruction.

Instruction name				Classification
OP1	OP2	OP3	OP4	
T	I	M	(Space)	TIMER
T	I	M	H	HIGH-SPEED TIMER
C	N	T	(Space)	COUNTER
C	N	T	R	REVERSIBLE COUNTER

**Operand, SV (Response)**

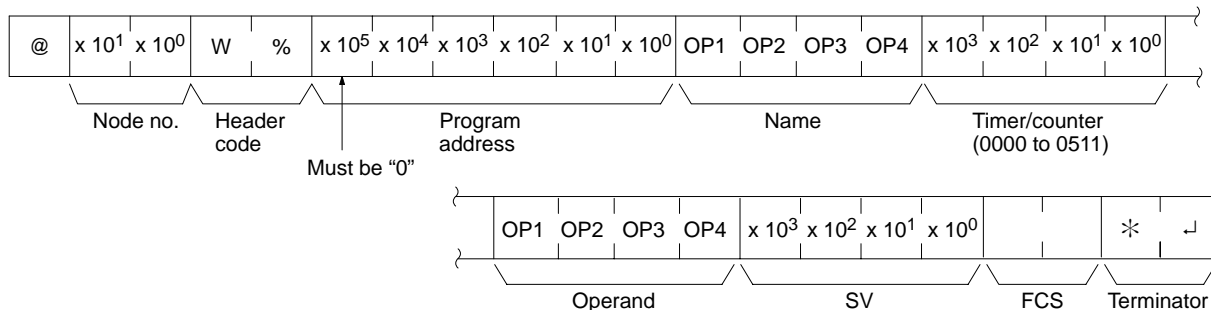
In "Operand," specify the name that indicates the SV classification. Specify the name in four characters. In "SV," specify either the word address where the SV is stored or the constant SV.

Operand				Classification	Constant or word address	
OP1	OP2	OP3	OP4		CQM1 PCs	CPM1 PCs
C	I	O	(Space)	IR or SR	0000 to 0252	0000 to 0019 0200 to 0252
L	R	(Space)	(Space)	LR	0000 to 0063	0000 to 0015
H	R	(Space)	(Space)	HR	0000 to 0099	0000 to 0019
A	R	(Space)	(Space)	AR	0000 to 0027	0000 to 0015
D	M	(Space)	(Space)	DM	0000 to 6655	0000 to 1023 6144 to 6655
D	M	*	(Space)	DM (indirect)	0000 to 6655	0000 to 1023 6144 to 6655
C	O	N	(Space)	Constant	0000 to 9999	0000 to 9999

**6-3-20 SV CHANGE 3 — W% (CQM1 Only)**

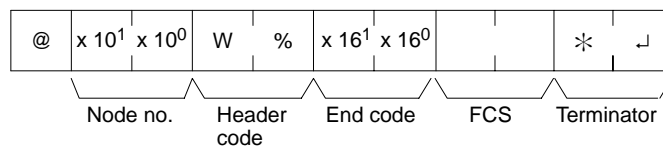
Changes the contents of the second word of the TIM, TIMH(15), CNT, or CNTR(12) at the specified program address in the user's program. With this command, program address can be specified for a program of more than 10K.

**Command Format**



**Response Format**

An end code of 00 indicates normal completion.



**Parameters**

**Name, TC Number (Command)**

In "Name," specify the name of the instruction, in four characters, for changing the SV. In "TC number," specify the timer/counter number used for the instruction.

Instruction name				Classification	TC number range
OP1	OP2	OP3	OP4		
T	I	M	(Space)	TIMER	0000 to 0511
T	I	M	H	HIGH-SPEED TIMER	
C	N	T	(Space)	COUNTER	
C	N	T	R	REVERSIBLE COUNTER	

**Operand, New SV (Response)**

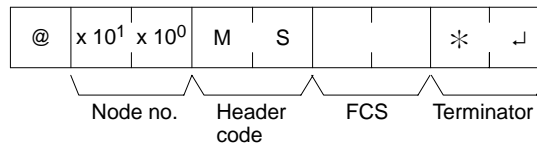
In "Operand," specify the name that indicates the SV classification. Specify the name in four characters. In "SV," specify either the word address where the SV is stored or the constant SV.

Operand				Classification	Constant or word address
OP1	OP2	OP3	OP4		
C	I	O	(Space)	IR or SR	0000 to 0252
L	R	(Space)	(Space)	LR	0000 to 0063
H	R	(Space)	(Space)	HR	0000 to 0099
A	R	(Space)	(Space)	AR	0000 to 0027
D	M	(Space)	(Space)	DM	0000 to 6655
D	M	*	(Space)	DM (indirect)	0000 to 6655
C	O	N	(Space)	Constant	0000 to 9999

### 6-3-21 STATUS READ — MS

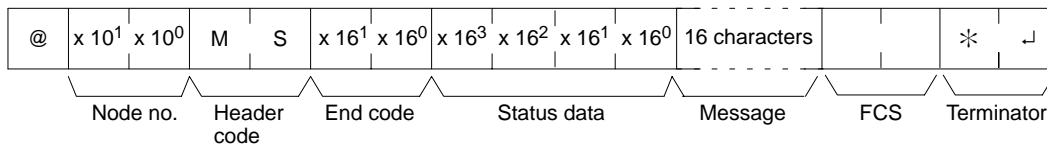
Reads the PC operating conditions.

#### Command Format



#### Response Format

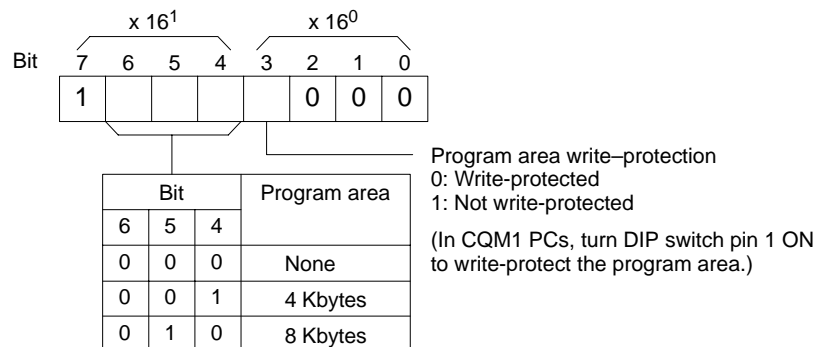
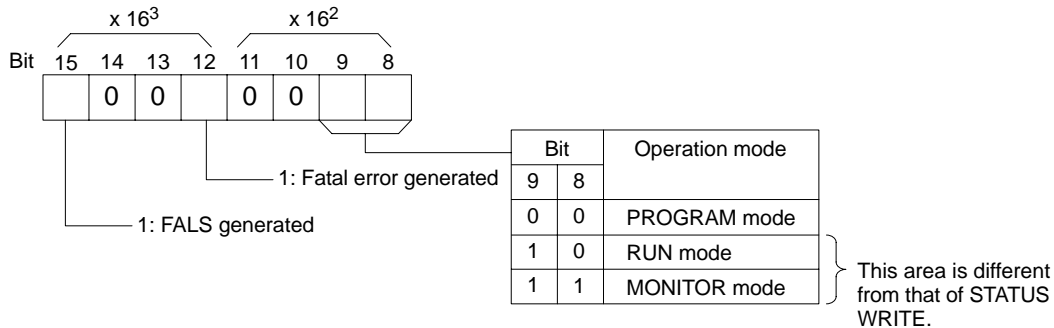
An end code of 00 indicates normal completion.



#### Parameters

#### Status Data, Message (Response)

“Status data” consists of four digits (two bytes) hexadecimal. The leftmost byte indicates CPU Unit operation mode, and the rightmost byte indicates the size of the program area.



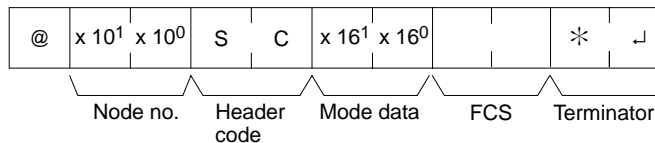
In CQM1 PCs, the “Message” parameter is a FAL/FALS number that exists when the command is executed. When there is no message, this parameter is omitted.

In CPM1/CPM1A/SRM1 PCs, the “Message” parameter is a 16-character message that exists when the command is executed. When there is no message, this parameter is omitted.

### 6-3-22 STATUS WRITE — SC

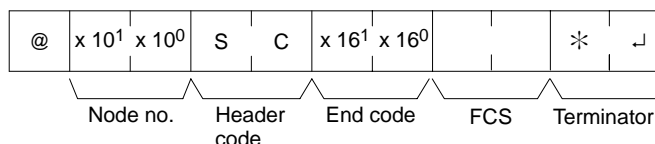
Changes the PC operating mode.

#### Command Format



#### Response Format

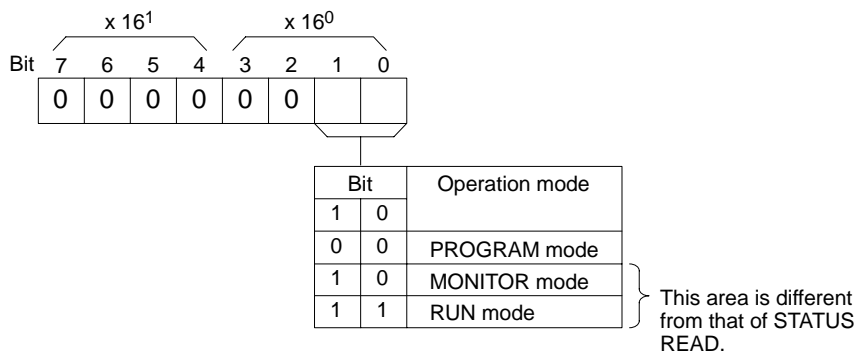
An end code of 00 indicates normal completion.



#### Parameters

##### Mode Data (Command)

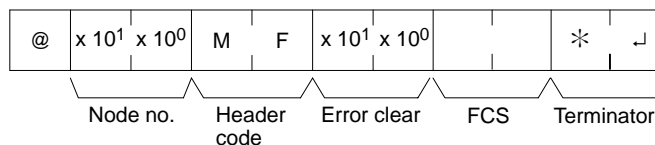
“Mode data” consists of two digits (one byte) hexadecimal. With the leftmost two bits, specify the PC operating mode. Set all of the remaining bits to “0.”



### 6-3-23 ERROR READ — MF

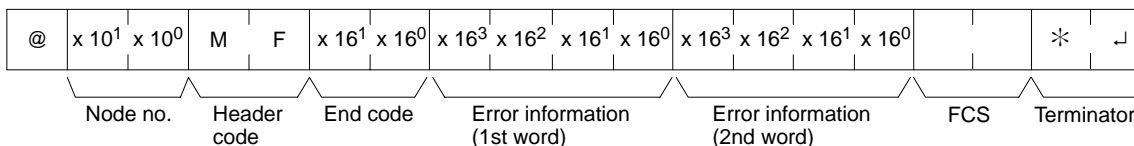
Reads and clears errors in the PC. Also checks whether previous errors have been cleared.

#### Command Format



#### Response Format

An end code of 00 indicates normal completion.



#### Parameters

##### Error Clear (Command)

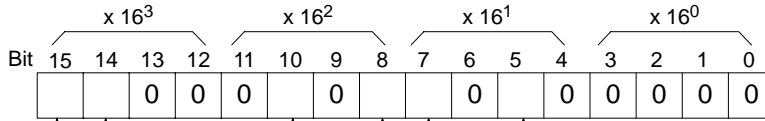
Specify 01 to clear errors and 00 to not clear errors (BCD). Fatal errors can be cleared only when the PC is in PROGRAM mode.

**Error Information (Response)**

The error information comes in two words.

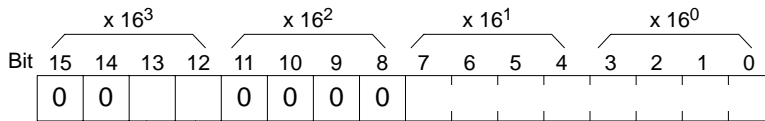
**CQM1/CPM1/CPM1A PCs**

1st word



- - - ON: Battery error (Error code F7, CQM1 only)
- - - ON: System error (FAL)
- - - ON: Memory error (Error code F1)
- - - ON: I/O bus error (Error code C0)
- - - ON: No end instruction error (FALS)
- - - ON: System error (FAL)

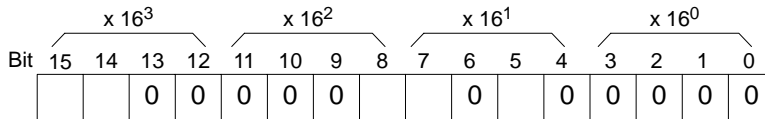
2nd word



- - - FAL, FALS No. (00 to FF)
- - - ON: Cycle time overrun (Error code F8)
- - - ON: I/O Unit overflow (Error code E1)

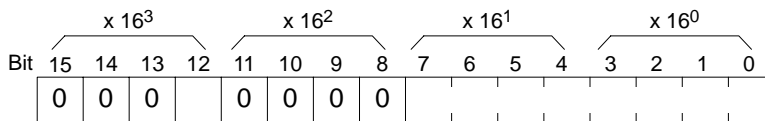
**SRM1 PCs**

1st word



- - - ON: Battery error (Error code F7, CQM1 only)
- - - ON: System error (FAL)
- - - ON: Memory error (Error code F1)
- - - ON: No end instruction error (FALS)
- - - ON: System error (FAL)

2nd word



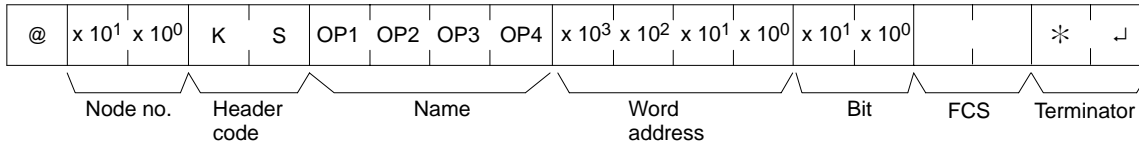
- - - FAL, FALS No. (00 to 99)
- - - ON: Cycle time overrun (Error code F8)

**6-3-24 FORCED SET — KS**

Force sets a bit in the IR, SR, LR, HR, AR, or TC area. Just one bit can be force set at a time.

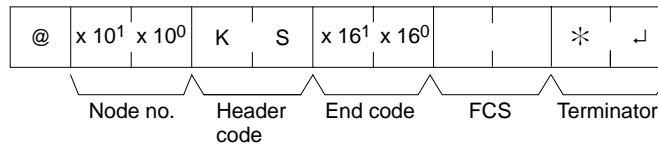
Once a bit has been forced set or reset, that status will be retained until a FORCED SET/RESET CANCEL (KC) command or the next FORCED SET/RESET command is transmitted.

**Command Format**



**Response Format**

An end code of 00 indicates normal completion.



**Parameters**

**Name, Word address, Bit (Command)**

In “Name,” specify the area (i.e., IR, SR, LR, HR, AR, or TC) that is to be forced set. Specify the name in four characters. In “Word address,” specify the address of the word, and in “Bit” the number of the bit that is to be forced set.

Name				Classification	Word address setting range		Bit
OP1	OP2	OP3	OP4		CQM1 PCs	CPM1/ CPM1A/SRM1 PCs	
C	I	O	(Space)	IR or SR	0000 to 0252	0000 to 0019 0200 to 0252	00 to 15 (decimal)
L	R	(Space)	(Space)	LR	0000 to 0063	0000 to 0015	
H	R	(Space)	(Space)	HR	0000 to 0099	0000 to 0019	
A	R	(Space)	(Space)	AR	0000 to 0027	0000 to 0015	
T	I	M	(Space)	Completion Flag (timer)	0000 to 0511	0000 to 0127	Always 00
T	I	M	H	Completion Flag (high-speed timer)			
C	N	T	(Space)	Completion Flag (counter)			
C	N	T	R	Completion Flag (reversible counter)			

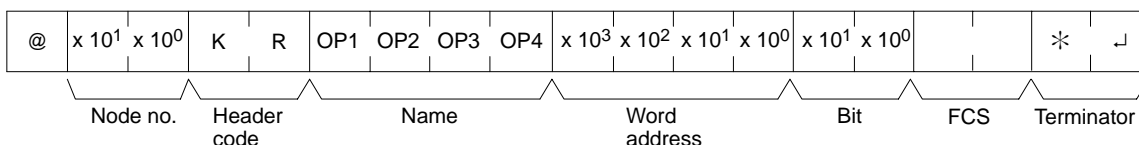
**Note** The area specified under “Name” must be in four characters. Add spaces after the data area name if it is shorter than four characters.

**6-3-25 FORCED RESET — KR**

Force resets a bit in the IR, SR, LR, HR, AR, or TC area. Just one bit can be force reset at a time.

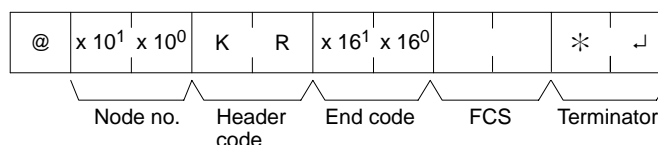
Once a bit has been forced set or reset, that status will be retained until a FORCED SET/RESET CANCEL (KC) command or the next FORCED SET/RESET command is transmitted.

**Command Format**



**Response Format**

An end code of 00 indicates normal completion.



**Parameters**

**Name, Word address, Bit (Command)**

In "Name," specify the area (i.e., IR, SR, LR, HR, AR, or TC) that is to be forced reset. Specify the name in four characters. In "Word address," specify the address of the word, and in "Bit" the number of the bit that is to be forced reset.

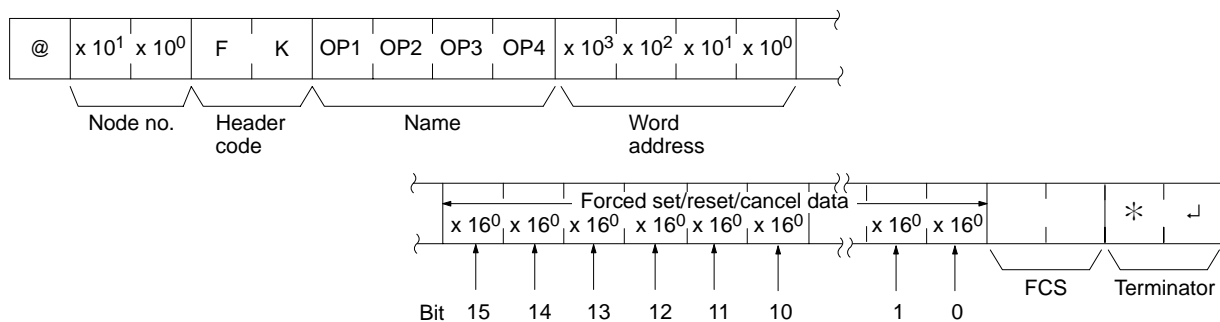
Name				Classification	Word address setting range		Bit
OP1	OP2	OP3	OP4		CQM1 PCs	CPM1/CPM1A/SRM1 PCs	
C	I	O	(Space)	IR or SR	0000 to 0252	0000 to 0019 0200 to 0252	00 to 15 (decimal)
L	R	(Space)	(Space)	LR	0000 to 0063	0000 to 0015	
H	R	(Space)	(Space)	HR	0000 to 0099	0000 to 0019	
A	R	(Space)	(Space)	AR	0000 to 0027	0000 to 0015	
T	I	M	(Space)	Completion Flag (timer)	0000 to 0511	0000 to 0127	Always 00
T	I	M	H	Completion Flag (high-speed timer)			
C	N	T	(Space)	Completion Flag (counter)			
C	N	T	R	Completion Flag (reversible counter)			

**Note** The area specified under "Name" must be in four characters. Add spaces after the data area name if it is shorter than four characters.

**6-3-26 MULTIPLE FORCED SET/RESET — FK**

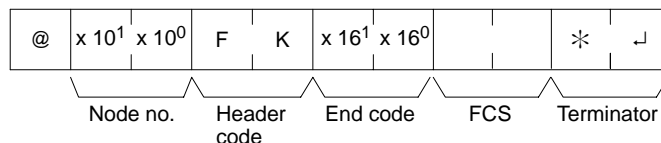
Force sets, force resets, or cancels the status of the bits in one word in the IR, SR, LR, HR, AR, or TC area.

**Command Format**



**Response Format**

An end code of 00 indicates normal completion.



**Parameters**

**Name, Word address (Command)**

In "Name," specify the area (i.e., IR, SR, LR, HR, AR, or TC) that is to be forced set or reset. Specify the name in four characters. In "Word address," specify the address of the word that is to be forced set or reset.

Name				Classification	Word address setting range	
OP1	OP2	OP3	OP4		CQM1 PCs	CPM1/CPM1A/SRM1 PCs
C	I	O	(Space)	IR or SR	0000 to 0252	0000 to 0019 0200 to 0252
L	R	(Space)	(Space)	LR	0000 to 0063	0000 to 0015
H	R	(Space)	(Space)	HR	0000 to 0099	0000 to 0019
A	R	(Space)	(Space)	AR	0000 to 0027	0000 to 0015
T	I	M	(Space)	Completion Flag (timer)	0000 to 0511	0000 to 0127
T	I	M	H	Completion Flag (high-speed timer)		
C	N	T	(Space)	Completion Flag (counter)		
C	N	T	R	Completion Flag (reversible counter)		

**Forced set/Reset/Cancel data (Command)**

If a timer or counter completion flag is specified, only bit 15 is effective and all other bits will be ignored. Only force-setting and force-resetting are possible for timers/counters.

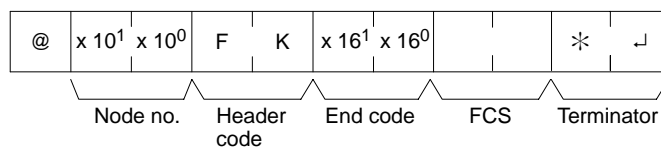
If a word address is specified, the content of the word specifies the desired process for each bit in the specified word, as shown in the following table.

Hexadecimal setting	Process
0000	No action (bit status not changed)
0002	Reset
0003	Set
0004	Forced-reset
0005	Forced-set
0008	Forced set/reset status cancel

The bits that are merely set or reset may change status the next time the program is executed, but bits that are force-set or force-reset will maintain the forced status until it is cleared.

**Response Format**

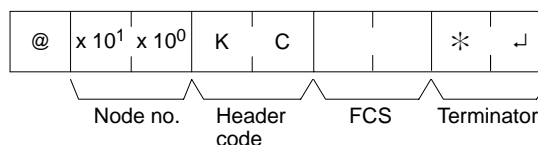
An end code of 00 indicates normal completion.



**6-3-27 FORCED SET/RESET CANCEL — KC**

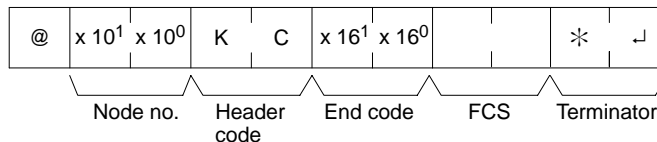
Cancels all forced set and forced reset bits (including those set by FORCED SET, FORCED RESET, and MULTIPLE FORCED SET/RESET). If multiple bits are set, the forced status will be cancelled for all of them. It is not possible to cancel bits one by one using KC.

**Command Format**





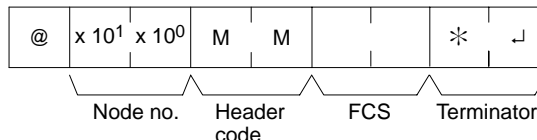
**Response Format** An end code of 00 indicates normal completion.



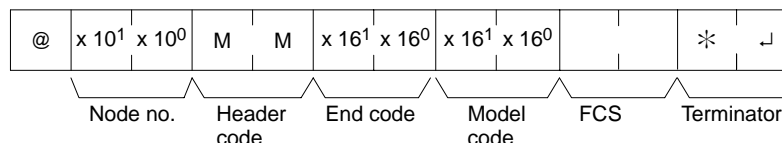
### 6-3-28 PC MODEL READ — MM

Reads the model type of the PC.

**Command Format**



**Response Format** An end code of 00 indicates normal completion.



**Parameters**

#### Model Code

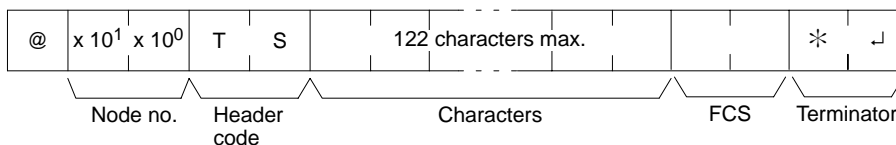
“Model code” indicates the PC model in two digits hexadecimal.

Model code	Model
01	C250
02	C500
03	C120
0E	C2000
10	C1000H
11	C2000H/CQM1/CPM1/CPM1A/SRM1
12	C20H/C28H/C40H/C200H/C200HS
20	CV500
21	CV1000
22	CV2000
40	CVM1-CPU01-E
41	CVM1-CPU11-E
42	CVM1-CPU21-E

### 6-3-29 TEST— TS

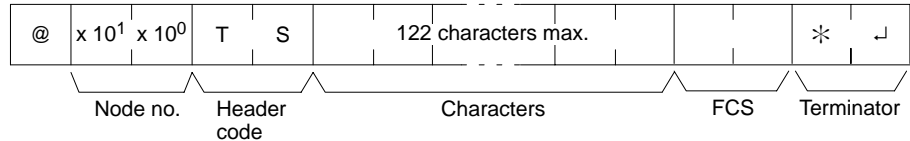
Returns, unaltered, one block of data transmitted from the host computer.

**Command Format**



**Response Format**

An end code of 00 indicates normal completion.



**Parameters**

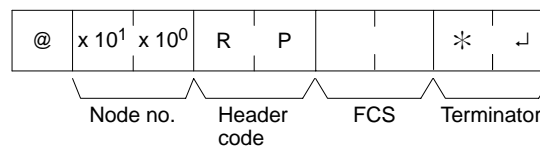
**Characters (Command, Response)**

For the command, this setting specifies any characters other than the carriage return (CHR\$(13)). For the response, the same characters as specified by the command will be returned unaltered if the test is successful.

**6-3-30 PROGRAM READ — RP**

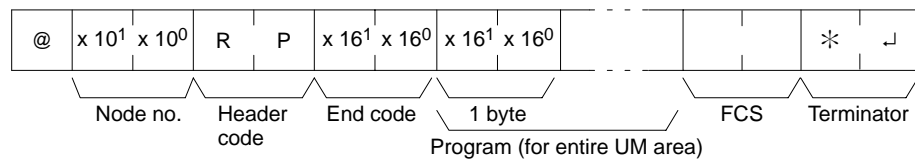
Reads the contents of the PC user's program area in machine language (object code). The contents are read as a block, from the beginning to the end.

**Command Format**



**Response Format**

An end code of 00 indicates normal completion.



**Parameters**

**Program (Response)**

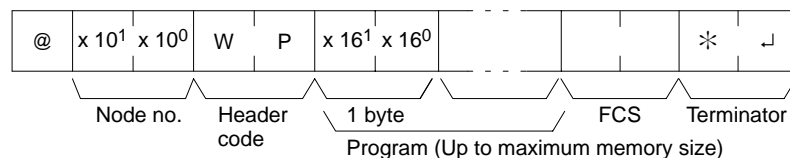
The program is read from the entire program area.

**Note** To stop this operation in progress, execute the ABORT (XZ) command.

**6-3-31 PROGRAM WRITE — WP**

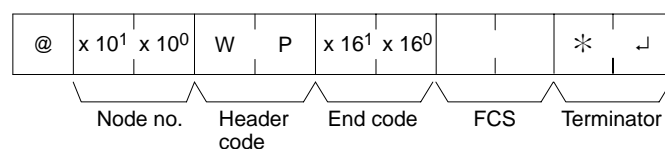
Writes to the PC user's program area the machine language (object code) program transmitted from the host computer. The contents are written as a block, from the beginning.

**Command Format**



**Response Format**

An end code of 00 indicates normal completion.



**Parameters**

**Program (Command)**

Program data up to the maximum memory size.

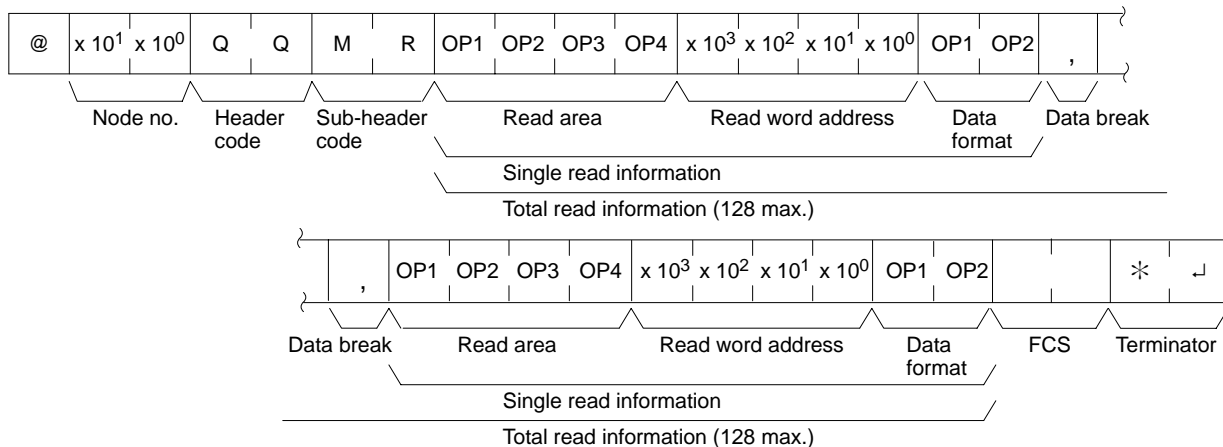
### 6-3-32 COMPOUND COMMAND — QQ

Registers at the PC all of the bits, words, and timers/counters that are to be read, and reads the status of all of them as a batch.

#### Registering Read Information

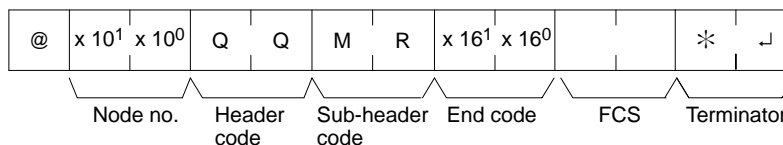
Register the information on all of the bits, words, and timers/counters that are to be read.

#### Command Format



#### Response Format

An end code of 00 indicates normal completion.



#### Parameters

##### Read Area (Command)

Specify in four-character code the area that is to be read. The codes that can be specified are listed in the following table.

**Read Word address, Data Format (Command)**

Depending on the area and type of data that are to be read, the information to be read is as shown in the following table. The “read data” is specified in four digits BCD, and the data format is specified in two digits BCD.

Area classification	Read data	Read area	Read word		Data format
			CQM1 PCs	CPM1/ CPM1A/ SRM1 PCs	
IR or SR	Bit	C I O (S)	0000 to 0255	0000 to 0019 0200 to 0255	00 to 15 (decimal)
	Word				“CH”
LR	Bit	L R (S) (S)	0000 to 0063	0000 to 0015	00 to 15 (decimal)
	Word				“CH”
HR	Bit	H R (S) (S)	0000 to 0099	0000 to 0019	00 to 15 (decimal)
	Word				“CH”
AR	Bit	A R (S) (S)	0000 to 0027	0000 to 0015	00 to 15 (decimal)
	Bit				“CH”
Timer	Completion Flag	T I M (S)	0000 to 0511	0000 to 0127	2 characters other than “CH”
	PV				“CH”
High-speed timer	Completion Flag	T I M H	0000 to 0511	0000 to 0127	2 characters other than “CH”
	PV				“CH”
Counter	Completion Flag	C N T (S)	0000 to 0511	0000 to 0127	2 characters other than “CH”
	PV				“CH”
Reversible counter	Completion Flag	C N T R	0000 to 0511	0000 to 0127	2 characters other than “CH”
	PV				“CH”
DM	Word	D M (S) (S)	0000 to 6655	0000 to 1023* 6144 to 6655	Any 2 characters

**Note** \*For SRM1 PCs, the DM range is from 0000 to 2047.

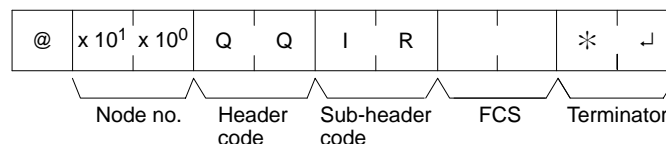
(S): Space

**Data Break (Command)**

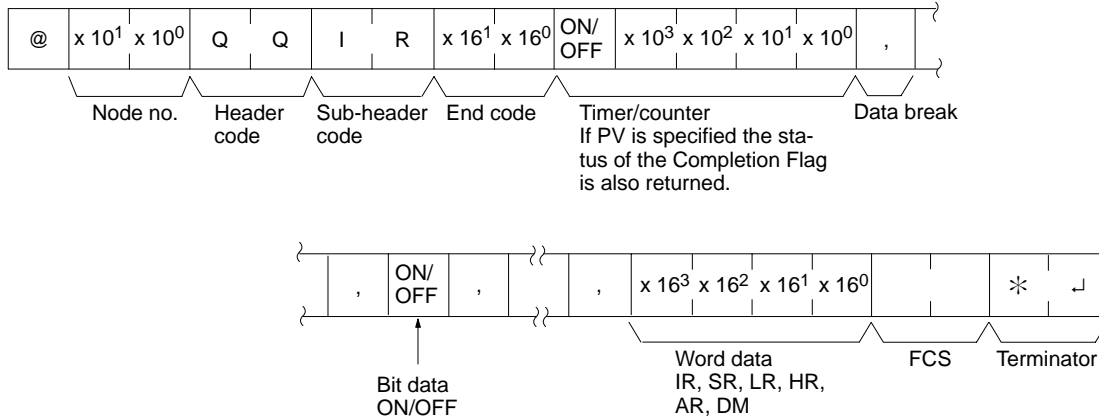
The read information is specified one item at a time separated by a break code (,). The maximum number of items that can be specified is 128. (When the PV of a timer/counter is specified, however, the status of the Completion Flag is also returned, and must therefore be counted as two items.)

**Batch Reading**

The bit, word, and timer/counter status is read as a batch according to the read information that was registered with QQ.

**Command Format**

**Response Format** An end code of 00 indicates normal completion.



**Parameters**

**Read Data (Response)**

Read data is returned according to the data format and the order in which read information was registered using QQ. If "Completion Flag" has been specified, then bit data (ON or OFF) is returned. If "Word" has been specified, then word data is returned. If "PV" has been specified for timers/counters, however, then the PV is returned following the Completion Flag.

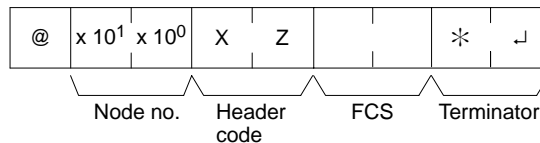
**Data Break (Response)**

The break code ( , ) is returned between sections that are read.

**6-3-33 ABORT — XZ**

Aborts the Host Link operation that is currently being processed, and then enables reception of the next command. The ABORT command does not receive a response.

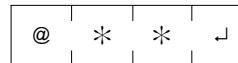
**Command Format**



**6-3-34 INITIALIZE — \*\***

Initializes the transmission control procedure of all the PCs connected to the host computer. The INITIALIZE command does not use node numbers or FCS, and does not receive a response.

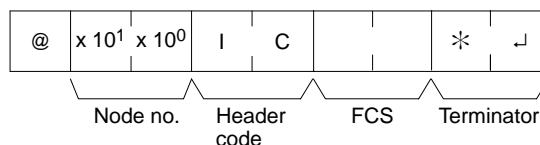
**Command Format**



**6-3-35 Undefined Command — IC**

This response is returned if the header code of a command cannot be decoded. Check the header code.

**Response Format**



# SECTION 7

## PC Operations and Processing Time

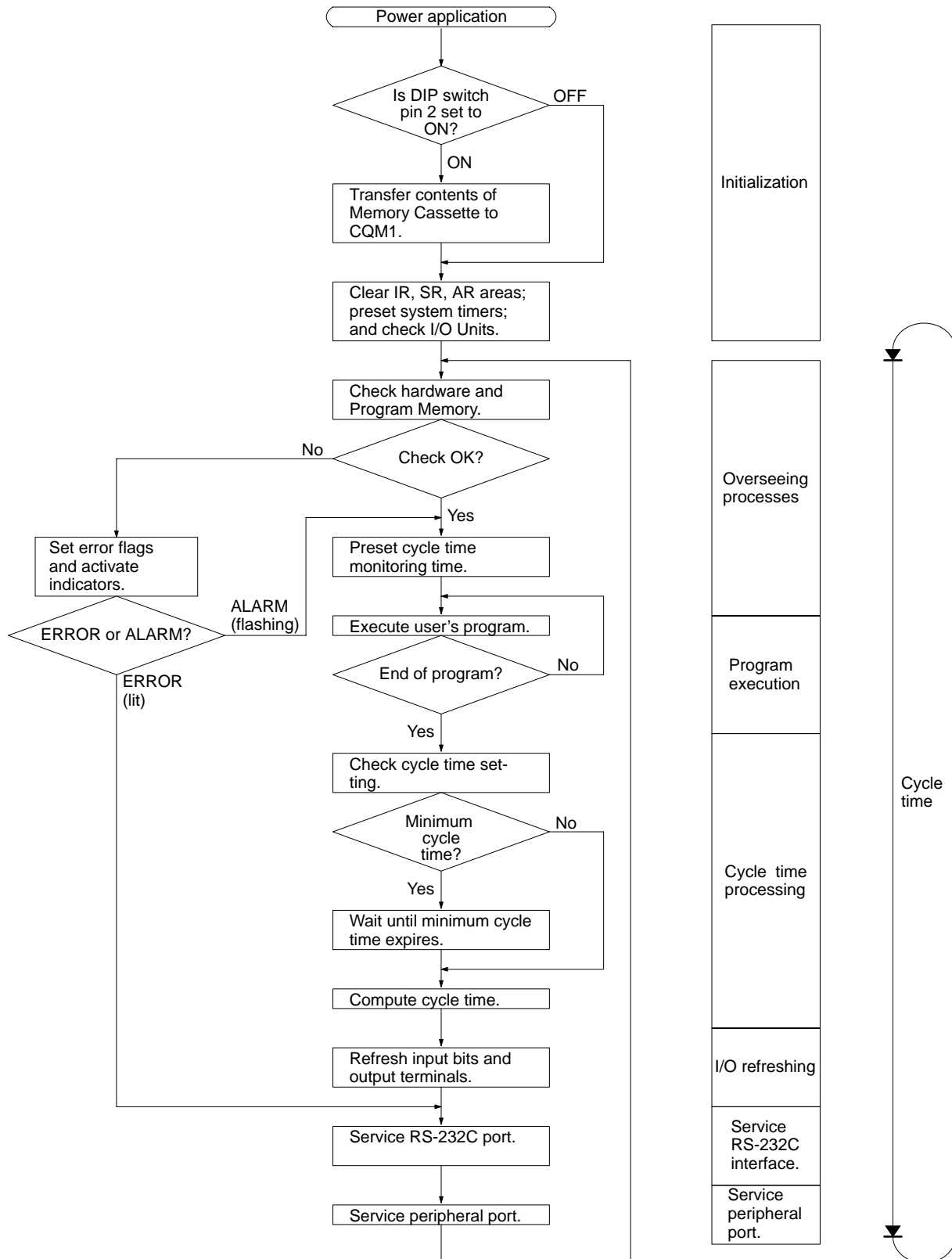
This section explains the internal processing of the CQM1/CPM1/CPM1A/SRM1, and the time required for processing and execution. Refer to this section to gain an understanding of the precise timing of CQM1/CPM1/CPM1A/SRM1 operation.

7-1	CQM1 Cycle Time and I/O Response Time .....	382
7-1-1	The CQM1 Cycle .....	382
7-1-2	CQM1 Cycle Time .....	383
7-1-3	I/O Response Time .....	385
7-1-4	One-to-one Link I/O Response Time .....	386
7-1-5	Interrupt Processing Time .....	388
7-1-6	CQM1 Instruction Execution Times .....	389
7-2	CPM1/CPM1A Cycle Time and I/O Response Time .....	400
7-2-1	The CPM1/CPM1A Cycle .....	400
7-2-2	CPM1/CPM1A Cycle Time .....	401
7-2-3	I/O Response Time .....	402
7-2-4	One-to-one Link I/O Response Time .....	403
7-2-5	Interrupt Processing Time .....	405
7-2-6	CPM1/CPM1A Instruction Execution Times .....	406
7-3	SRM1 Cycle Time and I/O Response Time .....	411
7-3-1	The SRM1 Cycle .....	411
7-3-2	SRM1 Cycle Time .....	412
7-3-3	I/O Response Time .....	414
7-3-4	One-to-one Link I/O Response Time .....	414
7-3-5	Interrupt Processing Time .....	416
7-3-6	SRM1 Instruction Execution Times .....	417

# 7-1 CQM1 Cycle Time and I/O Response Time

## 7-1-1 The CQM1 Cycle

**CQM1 Operation Flowchart** The overall flow of CQM1 operation is as shown in the following flowchart.



One cycle of CPU Unit operation is called a cycle. The time required for one cycle is called the cycle time.

### I/O Refresh Methods

CQM1 I/O refresh operations are broadly divided into two categories. The first of these, input refresh, involves reading the ON/OFF status of input points to the input bits. The second, output refresh, involves writing the ON/OFF status after program execution to the output points. The CQM1 I/O refresh methods are as shown in the following table.

Input/Output	I/O refresh method	Function
Input	Cyclic refresh	Input refresh is executed at a set time once per cycle.
	Interrupt input refresh	Input refresh is executed before execution of the interrupt processing routine whenever an input interrupt, interval timer interrupt, or high-speed counter interrupt occurs. (The cyclic refresh is also executed.)
Output	Cyclic refresh	Output refresh is executed at a set time once per cycle.
	Direct refresh	When there is an output from the user's program, that output point is immediately refreshed. (The cyclic refresh is also executed.)

The initial status of the CQM1 I/O refresh is as follows:

Input: Only cyclic refresh executed.

Output: Only cyclic refresh executed.

Cyclic refresh must be executed for both inputs and outputs. If input refresh is to be executed at the time of interrupts, then set the input refresh range in the PC Setup (DM 6630 to DM 6638). Stopping direct refresh can be set in DM 6639 of the PC Setup.

In addition to the methods described above, it is also possible to execute I/O refreshes in the program by means of IORF(97).

## 7-1-2 CQM1 Cycle Time

The processes involved in a single CQM1 cycle are shown in the following table, and their respective processing times are explained.

Process	Content	Time requirements
Overseeing	Setting cycle watchdog timer, I/O bus check, UM check, refreshing clock, refreshing bits allocated to new functions, etc.	0.8 ms (0.9 ms when a Memory Cassette equipped with a clock is mounted) Add an additional 0.1 ms for the CQM1-CPU4□-EV1 CPU Units.
Program execution	User program is executed.	Total time for executing instructions. (Varies according to content of user's program.)
Cycle time calculation	Standby until set time, when minimum cycle time is set in DM 6619 of PC Setup. Calculation of cycle time.	Almost instantaneous, except for standby processing.
I/O refresh	Input Unit's input information is read to input bits. Output information (results of executing program) is written to Output Unit's output bits.	Number of input words × 0.01 ms
RS-232C port servicing	Devices connected to RS-232C port serviced.	5% or less of cycle time (see note)
Peripheral port servicing	Devices connected to peripheral port serviced.	5% or less of cycle time (see note)

**Note** The percentages can be changed in the PC Setup (DM 6616, DM 6617).



**Cycle Time and Operations** The effects of the cycle time on CQM1 operations are as shown below.

Cycle time	Operation conditions
10 ms or longer	TIMH(15) may be inaccurate when TC 016 through TC 511 are used (operation will be normal for TC 000 through TC 015) (see note 1).
20 ms or longer	Programming using the 0.02-second Clock Bit (SR 25401) may be inaccurate.
100 ms or longer	Programming using the 0.1-second Clock Bit (SR 25500) may be inaccurate. A CYCLE TIME OVER error is generated (SR 25309 will turn ON) (see note 2).
120 ms or longer	The FALS 9F monitoring time SV is exceeded. A system error (FALS 9F) is generated, and operation stops (see note 3).
200 ms or longer	Programming using the 0.2-second Clock Bit (SR 25501) may be inaccurate.

- Note**
1. The number of timers to undergo interrupt processing can be set in DM 6629 of the PC Setup. The default setting is for TC 000 through TC 015.
  2. The PC Setup (DM 6655) can be used to disable detection of CYCLE TIME OVER error.
  3. The FALS 9F cycle monitoring time can be changed by means of the PC Setup (DM 6618).

**Cycle Time Example**

In this example, the cycle time is calculated for a CQM1 with 80 I/O points. The I/O is configured as follows:

DC inputs: 48 points (3 words)  
 Bit outputs: 32 points (2 words)

The rest of the operating conditions are assumed to be as follows:

User's program: 2,000 instructions (configured of LD and OUT instructions)  
 Clock: None  
 RS-232C port: Not used  
 Cycle time: Variable (no minimum set)

- Note** The average processing time for a single instruction in the user's program is assumed to be 0.625  $\mu$ s.

The cycle times are as shown in the following table.

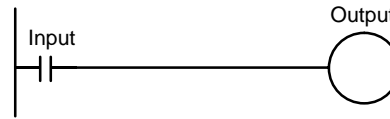
Process	Calculation method	Time with peripheral device	Time without peripheral device
Overseeing	Fixed	0.8 ms	0.8 ms
Program execution	$0.625 \times 2000$ ( $\mu$ s)	1.25 ms	1.25 ms
Cycle time calculation	Negligible	0 ms	0 ms
I/O refresh	$0.01 \times 3 + 0.005 \times 2$ ( $\mu$ s)	0.04 ms	0.04 ms
RS-232C port servicing	Not used.	0 ms	0 ms
Peripheral port servicing	Minimum time	0.34 ms	0 ms
Cycle time	(1) + (2) + (3) + (4) + (5) + (6)	2.43 ms	2.09 ms

- Note**
1. The cycle time can be automatically read from the PC via a Peripheral Device.
  2. The maximum and current cycle time are stored in AR 26 and AR 27.
  3. The cycle time can vary with actual operating conditions and will not necessarily agree precisely with the calculated value.
  4. The RS-232C and peripheral port service time will be 0.34 ms minimum, 87 ms maximum.

### 7-1-3 I/O Response Time

The I/O response time is the time it takes after an input signal has been received (i.e., after an input bit has turned ON) for the PC to check and process the information and to output a control signal (i.e., to output the result of the processing to an output bit). The I/O response time varies according to the timing and processing conditions.

The minimum and maximum I/O response times are shown here, using the following program as an example.

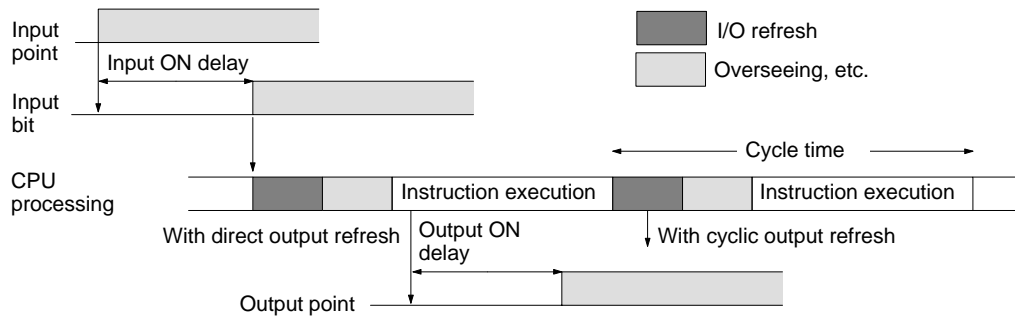


The following conditions are taken as examples for calculating the I/O response times.

- Input ON delay: 8 ms
- Overseeing time: 1 ms
- Instruction execution time: 14 ms
- Output ON delay: 10 ms
- Position of output instruction: Beginning of program
- Communications ports: Not used.

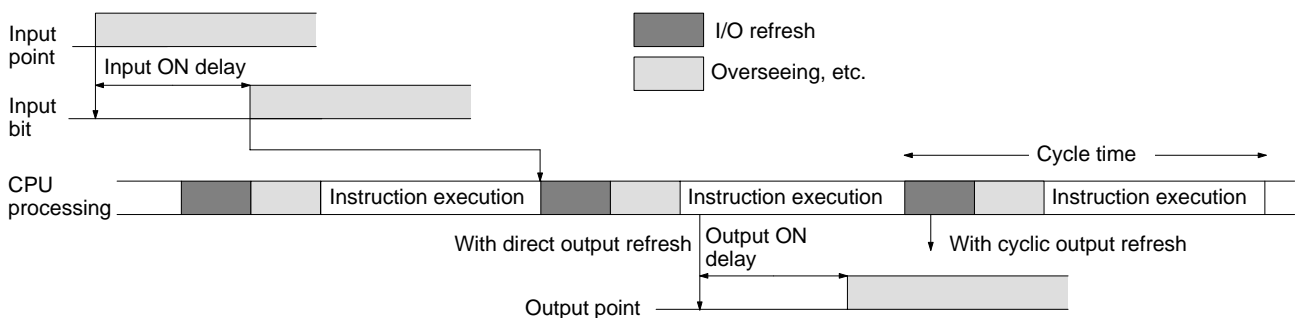
**Note** The input ON delay for DC Input Units can be set in the PC Setup.

**Minimum I/O Response Time** The CQM1 responds most quickly when it receives an input signal just prior to the input refresh phase of the cycle, as shown in the illustration below.



When cyclic output refreshing is used:  
 Minimum I/O response time = 8 + 15 + 10 = 33 ms  
 When direct output refreshing is used:  
 Minimum I/O response time = 8 + 1 + 10 = 19 ms

**Maximum I/O Response Time** The CQM1 takes longest to respond when it receives the input signal just after the input refresh phase of the cycle, as shown in the illustration below. In that case, a delay of approximately one cycle will occur.



When cyclic output refreshing is used:  
 Minimum I/O response time =  $8 + 15 \times 2 + 10 = 48$  ms  
 When direct output refreshing is used:  
 Minimum I/O response time =  $8 + 15 + 10 = 33$  ms

### 7-1-4 One-to-one Link I/O Response Time

When two CQM1s are linked one-to-one, the I/O response time is the time required for an input executed at one of the CQM1s to be output to the other CQM1 by means of one-to-one link communications.

One-to-one link communications are carried out reciprocally between the master and the slave. The respective transmission times are as shown below, depending on the number of LR words used.

Number of words used	Transmission time
64 words (LR 00 to LR 63)	39 ms
32 words (LR 00 to LR 31)	20 ms
16 words (LR 00 to LR 15)	10 ms

The minimum and maximum I/O response times are shown here, using as an example the following instructions executed at the master and the slave. In this example, communications proceed from the master to the slave.



The following conditions are taken as examples for calculating the I/O response times.

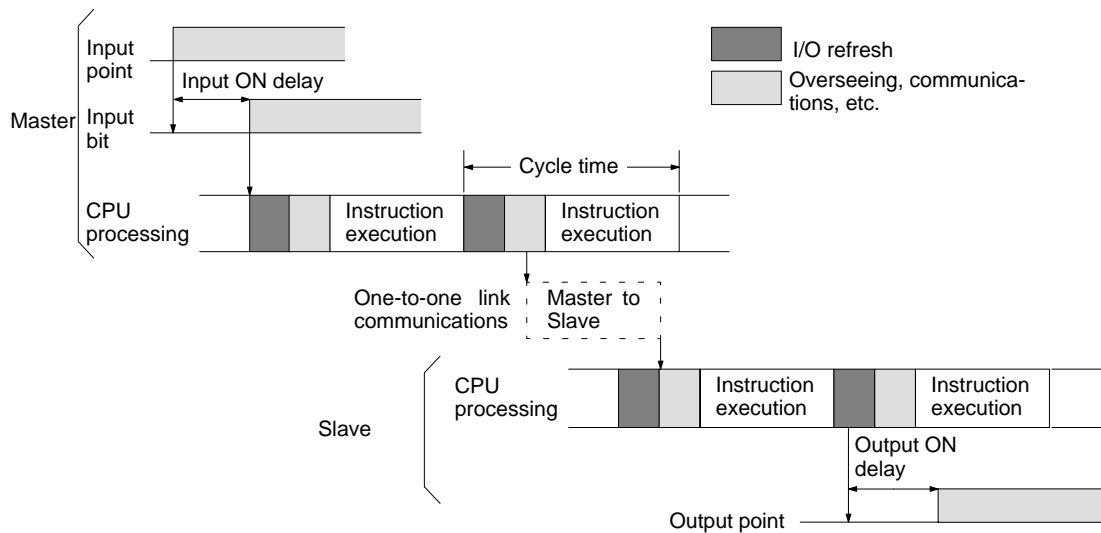
- Input ON delay: 8 ms
- Master cycle time: 10 ms
- Slave cycle time: 14 ms
- Output ON delay: 10 ms
- Direct output: Not used.
- Number of LR words: 64

**Note** The input ON delay for DC Input Units can be set in the PC Setup.

**Minimum I/O Response Time** The CQM1 responds most quickly under the following circumstances:

- 1, 2, 3... 1. The CQM1 receives an input signal just prior to the input refresh phase of the cycle.
2. The master to slave transmission begins immediately.

3. The slave executes communications servicing immediately after completion of communications.

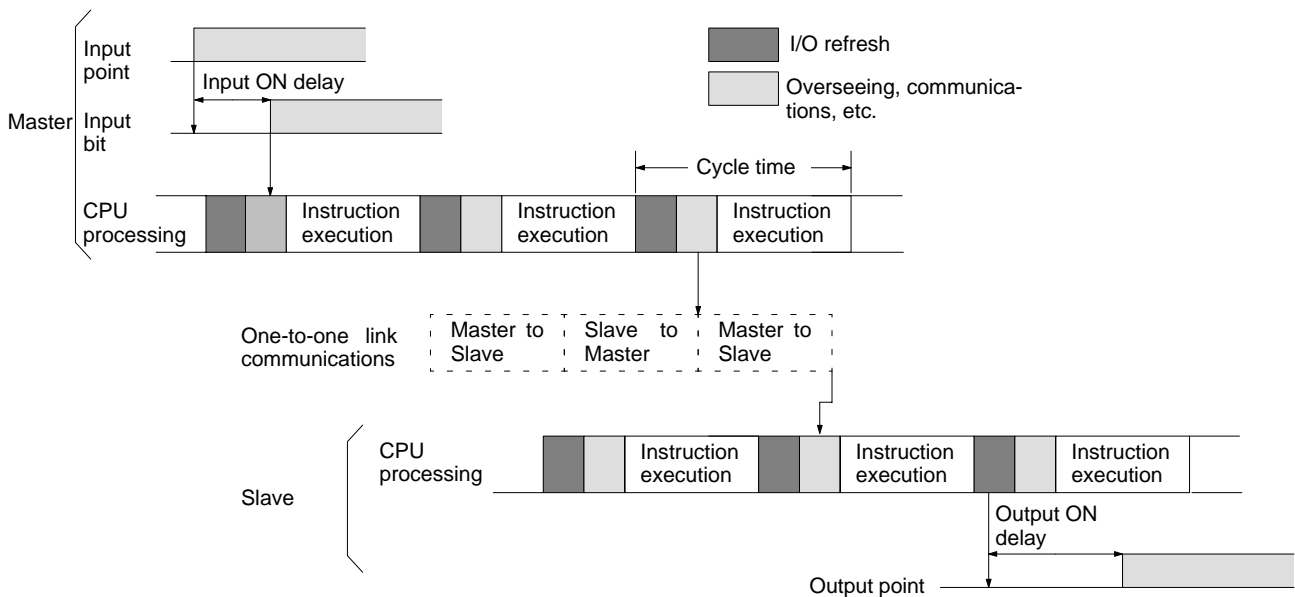


The minimum I/O response time is as follows:

Input ON delay:	8 ms
Master cycle time:	10 ms
Transmission time:	39 ms
Slave cycle time:	15 ms
+ Output ON delay:	10 ms
<hr/>	
Minimum I/O response time:	82 ms

**Maximum I/O Response Time** The CQM1 takes the longest to respond under the following circumstances:

- 1, 2, 3... 1. The CQM1 receives an input signal just after the input refresh phase of the cycle.
2. The master to slave transmission does not begin on time.
3. Communications are completed just after the slave executes communications servicing.



The maximum I/O response time is as follows:

Input ON delay:	8 ms
Master cycle time:	10 ms × 2
Transmission time:	39 ms × 3
Slave cycle time:	15 ms × 2
+ Output ON delay:	10 ms
<hr/>	
Maximum I/O response time:	185 ms

## 7-1-5 Interrupt Processing Time

This section explains the processing times involved from the time an interrupt is executed until the interrupt processing routine is called, and from the time an interrupt processing routine is completed until returning to the original position. The explanation applies to the following three types of interrupts: input interrupts, interval timer interrupts, and high-speed counter interrupts.

### Processing Time

The table below shows the times involved from the generation of an interrupt signal until the interrupt processing routine is called, and from when the interrupt processing routine is completed until returning to the original position.

Item	Contents	Time
Interrupt input ON delay	This is the delay time from the time the interrupt input bit turns ON until the time that the interrupt is executed. This is unrelated to other interrupts.	50 μs
↓ (Interrupt condition realized.) (see note)		
Standby until completion of interrupt-mask processing	This is the time during which interrupts are waiting until processing has been completed. This situation occurs when a mask processes is executed. It is explained below in more detail.	See below.
↓		
Change-to-interrupt processing	This is the time it takes to change processing to an interrupt.	40 μs
↓		
Input refresh at time of interrupt	This is the time required for input refresh when input refresh is set to be executed at the time the interrupt processing routine is called. (Set in PC Setup, DM 6630 to DM 6638.)	10 μs per word
↓ (Interrupt processing routine executed)		
Return	This is the time it takes, from execution of RET(93), to return to the processing that was interrupted.	40 μs

- Note**
1. When high-speed counter 0 is used with a range comparison table, the timing of interrupt processing can be affected by the cycle time.
  2. When high-speed counters 1 and 2 are used with range comparison tables (in CQM1-43/44-EV1 CPU Units), the timing of interrupt processing can be delayed up to 1 ms.

### Mask Processing

Interrupts are masked during processing of the operations described below. Until the processing is completed, any interrupts will remain masked for the indicated times.

High-speed timers: The time shown below is required, depending on (a) the number of timers used with TIMH(15) and (b) the number of high-speed timers active at that time. (The number of high-speed timers is set in the PC Setup, DM 6629. The default setting is 16.)

$$0 \leq \text{Standby time} \leq 50 + 3 \times (a + b) \mu\text{s}$$

Up to 50 μs can be required even when no high-speed timers are used.

Generation and clearing of non-fatal errors:

When a non-fatal error is generated and the error contents are registered at the CQM1, or when an error is being cleared, interrupts will be masked for a maximum of 100  $\mu\text{s}$  until the processing has been completed.

Online editing: Interrupts will be masked for a maximum of 1 second when on-line editing is executed during operation.

Pulse output based on SPED(64) may also be affected by interrupt processing, thus causing output timing to vary.

### Example Calculation

This example shows the interrupt response time (i.e., the time from when the interrupt input turns ON until the start of the interrupt processing routine) when input interrupts are used under the conditions shown below.

Number high-speed timers: 0 (No high-speed timers started)

Online edit: Not used

Input refresh at interrupt: No

#### Minimum Response Time

Interrupt input ON delay:	50 $\mu\text{s}$
Interrupt mask standby time:	0 $\mu\text{s}$
+ Change-to-interrupt processing:	40 $\mu\text{s}$
Minimum response time:	90 $\mu\text{s}$

#### Maximum Response Time

Interrupt input ON delay:	50 $\mu\text{s}$
Interrupt mask standby time:	50 $\mu\text{s}$
+ Change-to-interrupt processing:	40 $\mu\text{s}$
Minimum response time:	140 $\mu\text{s}$

In addition to the response time shown above, the time required for executing the interrupt processing routine itself and a return time of 40  $\mu\text{s}$  must also be accounted for when returning to the process that was interrupted.

Be sure to allow for interrupt processing time when using interrupts in the program.

Outputs from interrupt routines can be output immediately if direct output is used. Direct output will be used for both the main program and the interrupt routines, and cannot be set separately.

## 7-1-6 CQM1 Instruction Execution Times

The following table lists the execution times for CQM1 instructions. The maximum and minimum execution times and the conditions which cause them are given where relevant. When "word" is referred to in the Conditions column, it implies the content of any word except for indirectly addressed DM words. Indirectly addressed DM words, which create longer execution times when used, are indicated by "\*DM."

Execution times for most instructions depend on whether they are executed with an ON or an OFF execution condition. Exceptions are the ladder diagram instructions OUT and OUT NOT, which require the same time regardless of the execution condition. The OFF execution time for an instruction can also vary depending on the circumstances, i.e., whether it is in an interlocked program section and the execution condition for IL is OFF, whether it is between JMP(04) and JME(05) and the execution condition for JMP(04) is OFF, or whether it is reset by an OFF execution condition. "RSET," "IL," and "JMP" are used to indicate these three times.

## Basic Instructions

Code	Mnemonic	ON execution time (μs)	Conditions (Top: min.; bottom: max.)	OFF execution time (μs)		
				RSET	IL	JMP
---	LD	0.5	Any	---		
---	LD NOT			---		
---	AND			---		
---	AND NOT			---		
---	OR			---		
---	OR NOT			---		
---	AND LD OR LD			---		
---	OUT OUT NOT	0.75	Without direct outputs or for operands other than IR 10000 to IR 11515 when direct outputs are used.	---		
---	SET	1.25	Direct outputs	---		
---	RSET	1.25	Direct outputs	---		
---	TIM	1.5	Constant for SV	1.5	1.5	1.5
			*DM for SV	54.1	1.5	1.5
---	CNT	1.5	Constant for SV	1.5	1.5	1.5
			*DM for SV	51.6	1.5	1.5

## Special Instructions

Code	Mnemonic	ON execution time (μs)	Conditions (Top: min.; bottom: max.)	OFF execution time (μs)		
00	NOP	0.5	Any	0.0		
01	END	29		0.0		
02	IL	12.3		10.9		
03	ILC	11.3		11.3		
04	JMP	18.3		11.9		
05	JME	11.0		11.0		
06	FAL	56.8		1.5		
07	FALS	4.0		1.5		
08	STEP	58.2		1.5		
09	SNXT	25.0				
10	SFT			Shift	IL	JMP
		44.2	With 1-word shift register	43.2	15.0	15.0
		77.7	With 10-word shift register	68.5	15.0	15.0
		415.2	With 100-word shift register	322.0	15.0	15.0
11	KEEP	0.75	Without direct outputs or for operands other than IR 10000 to IR 11515 when direct outputs are used.			
		1.25	Direct outputs using IR 10000 to IR 11515			
12	CNTR	53.0	Constant for SV	Shift	IL	JMP
		79.6	*DM for SV	33.1	20.7	20.7
13	DIFU	21.5	Any	Reset	IL	JMP
				21.0	20.8	17.8
14	DIFD	20.8	Any	Reset	IL	JMP
				20.8	20.6	17.6
15	TIMH	36.5	Constant for SV	Shift	IL	JMP
		36.5	*DM for SV	54.7	53.0	27.7
				81.0	79.6	27.7

Code	Mnemonic	ON execution time (μs)	Conditions (Top: min.; bottom: max.)	OFF execution time (μs)
16	WSFT	44.7	With 1-word shift register	2.0
		77.0	With 10-word shift register	
		2.25 ms	With 1,024-word shift register using *DM	
		13.05 ms	With 6,144-word shift register using *DM	
20	CMP	26.7	When comparing a constant to a word	2.0
		29.5	When comparing two words	
		77.3	When comparing two *DM	
21	MOV	23.5	When transferring a constant to a word	2.0
		26.3	When moving from one word to another	
		72.7	When transferring *DM to *DM	
22	MVN	23.7	When transferring a constant to a word	2.0
		26.5	When moving from one word to another	
		72.6	When transferring *DM to *DM	
23	BIN	50.4	When converting a word to a word	2.0
		96.0	When converting *DM to *DM	
24	BCD	47.7	When converting a word to a word	2.0
		93.3	When converting *DM to *DM	
25	ASL	24.0	When shifting a word	1.5
		45.8	When shifting *DM	
26	ASR	24.0	When shifting a word	1.5
		45.8	When shifting *DM	
27	ROL	24.7	When rotating a word	1.5
		46.6	When rotating *DM	
28	ROR	24.7	When rotating a word	1.5
		46.6	When rotating *DM	
29	COM	25.9	When inverting a word	1.5
		48.3	When inverting *DM	
30	ADD	49.9	Constant + word → word	2.5
		53.1	Word + word → word	
		122.1	*DM + *DM → *DM	
31	SUB	49.9	Constant – word → word	2.5
		53.1	Word – word → word	
		122.1	*DM – *DM → *DM	
32	MUL	73.7	Constant × word → word	2.5
		77.0	Word × word → word	
		144.5	*DM × *DM → *DM	
33	DIV	72.2	Word ÷ constant → word	2.5
		75.4	word ÷ word → word	
		143.0	*DM ÷ *DM → *DM	
34	ANDW	41.9	Constant □ word → word	2.5
		45.1	Word □ word → word	
		114.1	*DM □ *DM → *DM	
35	ORW	41.9	Constant V word → word	2.5
		45.1	Word V word → word	
		114.1	*DM V *DM → *DM	
36	XORW	41.9	Constant ∨ word → word	2.5
		45.2	Word ∨ word → word	
		114.1	*DM ∨ *DM → *DM	



Code	Mnemonic	ON execution time (μs)	Conditions (Top: min.; bottom: max.)	OFF execution time (μs)
37	XNRW	41.9	Constant $\nabla$ word $\rightarrow$ word	2.5
		45.1	Word $\nabla$ word $\rightarrow$ word	
		114.1	*DM $\nabla$ *DM $\rightarrow$ *DM	
38	INC	27.8	When incrementing a word	1.5
		50.1	When incrementing *DM	
39	DEC	28.4	When decrementing a word	1.5
		50.8	When decrementing *DM	
40	STC	12.0	Any	1.0
41	CLC	12.0		1.0
45	TRSM	28.8		1.0
46	MSG	24.6	With message in words	1.5
		48.4	With message in *DM	
50	ADB	53.4	Constant + word $\rightarrow$ word	2.5
		56.6	Word + word $\rightarrow$ word	
		125.6	*DM + *DM $\rightarrow$ *DM	
51	SBB	53.4	Constant - word $\rightarrow$ word	2.5
		56.6	Word - word $\rightarrow$ word	
		125.6	*DM - *DM $\rightarrow$ *DM	
52	MLB	45.7	Constant $\times$ word $\rightarrow$ word	2.5
		48.9	Word $\times$ word $\rightarrow$ word	
		116.4	*DM $\times$ *DM $\rightarrow$ *DM	
53	DVB	46.7	Word $\div$ constant $\rightarrow$ word	2.5
		49.9	Word $\div$ word $\rightarrow$ word	
		117.4	*DM $\div$ *DM $\rightarrow$ *DM	
54	ADDL	59.3	Word + word $\rightarrow$ word	2.5
		128.9	*DM + *DM $\rightarrow$ *DM	
55	SUBL	59.3	Word - word $\rightarrow$ word	2.5
		128.9	*DM - *DM $\rightarrow$ *DM	
56	MULL	204.5	Word $\times$ word $\rightarrow$ word	2.5
		271.2	*DM $\times$ *DM $\rightarrow$ *DM	
57	DIVL	205.9	Word $\div$ word $\rightarrow$ word	2.5
		272.6	*DM $\div$ *DM $\rightarrow$ *DM	
58	BINL	76.0	Word $\rightarrow$ word	2.0
		120.6	*DM $\rightarrow$ *DM	
59	BCDL	60.9	Word $\rightarrow$ word	2.0
		105.6	*DM $\rightarrow$ *DM	
70	XFER	72.9	When transferring a constant to a word	2.5
		76.1	When transferring a word to a word	
		2.90 ms	When transferring 1,024 words using *DM	
		16.66 ms	When transferring 6,144 words using *DM	
71	BSET	45.6	When setting a constant to 1 word	2.5
		77.9	When setting word constant to 10 words	
		1.93 ms	When setting *DM to 1,024 words	
		10.95 ms	When setting *DM to 6,144 words	
72	ROOT	63.9	Word calculation $\rightarrow$ word	2.0
		110.8	*DM calculation $\rightarrow$ *DM	
73	XCHG	40.9	Word $\rightarrow$ word	2.0
		85.5	*DM $\rightarrow$ *DM	

Code	Mnemonic	ON execution time (μs)	Conditions (Top: min.; bottom: max.)	OFF execution time (μs)
74	SLD	41.1	Shifting 1 word	2.0
		101.9	Shifting 10 word	
		5.49 ms	Shifting 1024 words using *DM	
		32.58 ms	Shifting 6144 words using *DM	
75	SRD	41.1	Shifting 1 word	2.0
		101.9	Shifting 10 word	
		5.49 ms	Shifting 1,024 words using *DM	
		32.57 ms	Shifting 6,144 words using *DM	
76	MLPX	59.1	When decoding word to word	2.5
		136.4	When decoding *DM to *DM	
77	DMPX	45.1	When encoding word to word	2.5
		120.6	When encoding *DM to *DM	
78	SDEC	60.6	When decoding word to word	2.5
		138.5	When decoding *DM to *DM	
80	DIST	66.0	When setting a constant to a word + a word	2.5
		69.3	When setting a word to a word + a word	
		144.3	When setting *DM to *DM + *DM	
		101.0	When setting a constant to a stack	
		104.3	When setting a word to a stack	
		177.8	When setting *DM to a stack via *DM	
81	COLL	65.1	When setting a constant + a word to a word	2.5
		68.3	When setting a word + a word to a word	
		140.1	When setting *DM + *DM to *DM	
		61.1	When setting a word + constant to FIFO stack	
		64.3	When setting a word + word to FIFO stack	
		137.6	When setting a *DM + *DM to FIFO stack via *DM	
		60.3	When setting a word + constant to LIFO stack	
		63.6	When setting a word + word to LIFO stack	
		136.8	When setting a *DM + *DM to LIFO stack via *DM	
82	MOVB	46.4	When moving constant to word	2.5
		54.9	When moving word to word	
		125.2	When moving *DM to *DM	
83	MOVD	40.7	When moving constant to word	2.5
		49.2	When moving word to word	
		119.4	When moving *DM to *DM	
84	SFTR	57.4	Shifting 1 word	2.5
		98.4	Shifting 10 word	
		2.26 ms	Shifting 1,024 words using *DM	
		12.90 ms	Shifting 6,144 words using *DM	
85	TCMP	95.8	Comparing constant to word-set table	2.5
		98.8	Comparing word to word-set table	
		169.0	Comparing *DM to *DM-set table	
86	ASC	62.5	Word → word	2.5
		144.3	*DM → *DM	
91	SBS	41.4	Any	1.5
92	SBN	---		---
93	RET	39.0		1.5

Code	Mnemonic	ON execution time (μs)	Conditions (Top: min.; bottom: max.)	OFF execution time (μs)
97	IORF	37.7	Refreshing IR 000	2.0
		41.7	Refreshing one input word	
		46.9	Refreshing one output word	
		112.4	Refreshing 8 I/O words	
99	MCRO	140.2	With word-set I/O operands	2.5
		188.1	With *DM-set I/O operands	

## Expansion Instructions

Code	Mnemonic	ON execution time (μs)	Conditions	OFF execution time (μs)
17	ASFT	62.7	Shifting a word	2.5
		96.7	Shifting 10 words	
		2.45 ms	Shifting 1,024 words via *DM	
		16.33 ms	Shifting 6,144 words via *DM	
18	TKY	81.1	Word to word	2.5
		131.8	*DM to *DM	
19	MCMP	123.9	Comparing words	2.5
		195.3	Comparing *DM	
47	RXD	123.1	Inputting 1 byte via word	2.5
		847.3	Inputting 256 bytes via *DM	
48	TXD	105.1	Outputting 1 byte via word (RS-232C)	2.5
		832.3	Outputting 256 bytes via *DM (RS-232C)	
		86.3	Outputting 1 byte via word (host link)	
		141.9	Outputting 256 bytes via *DM (host link)	
60	CMPL	50.9	Comparing words	2.5
		101.0	Comparing *DM	
61	INI	High-speed counter 0 or pulse output from an output bit:		2.5
		90.6	Starting comparison via word	
		114.4	Starting comparison via *DM	
		72.1	Stopping comparison via word	
		83.0	Stopping comparison via *DM	
		163.6	Changing PV via word	
		182.2	Changing PV via *DM	
		56.4	Stopping pulse output via word	
		80.2	Stopping pulse output via *DM	
		High-speed counters 1 and 2 or pulse output from ports 1 and 2:		
		296.8	Starting comparison via word	
		324.3	Starting comparison via *DM	
		207.3	Stopping comparison via word	
		232.8	Stopping comparison via *DM	
		468.3	Changing PV via word	
		487.8	Changing PV via *DM	
		248.8	Stopping pulse output via word	
		269.8	Stopping pulse output via *DM	
		Absolute high-speed counters 1 and 2:		
		296.3	Starting comparison via word	
316.8	Starting comparison via *DM			
202.3	Stopping comparison via word			
226.3	Stopping comparison via *DM			

Code	Mnemonic	ON execution time (μs)	Conditions	OFF execution time (μs)
62	PRV	High-speed counter 0 or pulse output from an output bit:		2.5
		91.5	Designating output via word	
		117.4	Designating output via *DM	
		High-speed counters 1 and 2 or pulse output from ports 1 and 2:		
		229.3	Designating output via word (reading status)	
		249.3	Designating output via *DM (reading status)	
		229.8	Designating output via word (reading range comparison results)	
		256.3	Designating output via *DM (reading range comparison results)	
		Absolute high-speed counters 1 and 2:		
		226.3	Designating output via word (reading status)	
		253.3	Designating output via *DM (reading status)	
		227.8	Designating output via word (reading range comparison results)	
		253.3	Designating output via *DM (reading range comparison results)	

Code	Mnemonic	ON execution time (μs)	Conditions	OFF execution time (μs)
63	CTBL	High-speed counter 0 or pulse output from an output bit:		2.5
		210.3	Target table with 1 target in words and start	
		233.8	Target table with 1 target in *DM and start	
		1.31 ms	Target table with 16 targets in words and start	
		1.33 ms	Target table with 16 targets in *DM and start	
		1.25 ms	Range table in words and start	
		1.27 ms	Range table in *DM and start	
		170.8	Target table with 1 target in words	
		194.3	Target table with 1 target in *DM	
		1.27 ms	Target table with 16 targets in words	
		1.30 ms	Target table with 16 targets in *DM	
		1.09 ms	Range table in words	
		1.11 ms	Range table in *DM	
		High-speed counters 1 and 2 or pulse output from ports 1 and 2:		
		692.8	Target table with 1 target in words and start	
		721.8	Target table with 1 target in *DM and start	
		2.79/7.84 ms	Target table with 16/48 targets in words and start	
		2.81/7.86 ms	Target table with 16/48 targets in *DM and start	
		2.26 ms	Range table in words and start	
		2.27 ms	Range table in *DM and start	
		488.8	Target table with 1 target in words	
		517.8	Target table with 1 target in *DM	
		2.57/7.67 ms	Target table with 16/48 targets in words	
		2.61/7.72 ms	Target table with 16/48 targets in *DM	
		2.19 ms	Range table in words	
		2.21 ms	Range table in *DM	
		Absolute high-speed counters 1 and 2:		
		600.8	Target table with 1 target in words and start	
		624.8	Target table with 1 target in *DM and start	
		6.46 ms	Target table with 48 targets in words and start	
		6.47 ms	Target table with 48 targets in *DM and start	
		1.47 ms	Range table in words and start	
		1.50 ms	Range table in *DM and start	
460.8	Target table with 1 target in words			
484.8	Target table with 1 target in *DM			
6.02 ms	Target table with 48 targets in words			
6.03 ms	Target table with 48 targets in *DM			
1.45 ms	Range table in words			
1.47 ms	Range table in *DM			
64	SPED	Pulse output from an output bit:		2.5
		118.4	Frequency specified by constant	
		123.2	Frequency specified by word	
		146.8	Frequency specified by *DM	
		Pulse output from ports 1 and 2:		
		302.3	Frequency specified by constant	
		310.3	Frequency specified by word	
		320.3	Frequency specified by *DM	

Code	Mnemonic	ON execution time ( $\mu$ s)	Conditions	OFF execution time ( $\mu$ s)
65	PULS	Pulse output from an output bit:		2.5
		109.0	Number of pulses specified by word	
		137.8	Number of pulses specified by *DM	
		Pulse output from ports 1 and 2:		
		337.3	Number of pulses specified by word	
		360.3	Number of pulses specified by *DM	
66	SCL	105.8	Word designation	2.5
		180.5	*DM designation	
67	BCNT	88.4	Counting a word	2.5
		49.32 ms	Counting 6,656 words via *DM	
68	BCMP	140.0	Comparing constant, results to word	2.5
		143.0	Comparing word, results to word	
		194.7	Comparing *DM, results to *DM	
69	STIM	36.8	Word-set one-shot interrupt start	2.5
		73.8	*DM-set one-shot interrupt start	
		37.3	Word-set scheduled interrupt start	
		74.3	*DM-set scheduled interrupt start	
		66.4	Word-set timer read	
		113.6	*DM-set timer read	
		35.3	Word-set timer stop	
		35.6	*DM-set timer stop	
87	DSW	70.3	Word-set 4-digit CS output	2.5
		70.3	Word-set 4-digit RD output	
		89.1	Word-set 4-digit data input	
		93.1	*DM-set 4-digit CS output	
		93.1	*DM-set 4-digit RD output	
		110.3	*DM-set 4-digit data input	
		74.7	Word-set 8-digit CS output	
		75.1	Word-set 8-digit RD output	
		105.5	Word-set 8-digit data input	
		103.5	*DM-set 8-digit CS output	
		103.9	*DM-set 8-digit RD output	
		131.5	*DM-set 8-digit data input	
88	7SEG	78.7	4 digits, word designation	2.5
		102.6	4 digits, *DM designation	
		92.1	8 digits, word designation	
		117.2	8 digits, *DM designation	
89	INT	53.0	Set masks via word	2.5
		80.8	Set masks via *DM	
		49.9	Clear interrupts via word	
		73.2	Clear interrupts via *DM	
		50.7	Read mask status via word	
		71.9	Read mask status via *DM	
		64.8	Change counter SV via word	
		88.1	Change counter SV via *DM	
		27.5	Mask all interrupts via word	
		27.5	Mask all interrupts via *DM	
		28.5	Clear all interrupts via word	
		28.5	Clear all interrupts via *DM	

Code	Mnemonic	ON execution time ( $\mu$ s)	Conditions	OFF execution time ( $\mu$ s)
	HKY	71.5	Output word to word	2.5
		100.3	Output *DM to *DM	
		81.5	Input word to word	
		109.5	Input *DM to *DM	
	FPD	171.6	Word designation, no message, execution	2.5
		279.5	*DM designation, message, execution	
		204.9	Word designation, no message, initial	
		312.0	*DM designation, message, initial	
	SRCH	62.4	Searching word, results to word	2.5
		2.64 ms	Searching 1,024 word via *DM, results to *DM	
		15.11 ms	Searching 6,144 word via *DM, results to *DM	
	MAX	56.1	Searching word, results to word	2.5
		2.56 ms	Searching 999 words via *DM, results to *DM	
	MIN	56.1	Searching word, results to word	2.5
		2.56 ms	Searching 999 words via *DM, results to *DM	
	APR	57.4	Computing sine	2.5
		460.4	Linear approximation with 256-item table via *DM designation	
	LINE	93.4	Word to word	2.5
		166.5	*DM to *DM	
	COLM	115.1	Word to word	2.5
		183.1	*DM to *DM	
	SEC	92.9	Word to word	2.5
		146.2	*DM to *DM	
	HMS	94.9	Word to word	2.5
		148.7	*DM to *DM	
	SUM	72.9	Adding one word, results to word	2.5
		6.86 ms	Adding 999 words via *DM, results to *DM	
	FCS	73.6	Computing one word, results to word	2.5
		2.33 ms	Computing 999 words via *DM, results to *DM	
	HEX	82.3	Word to word	2.5
		154.3	*DM to *DM	
	AVG	73.7	One-cycle average for word	2.5
		282.5	64-cycle average via *DM	
	PWM	266.8	Duty ratio specified by constant	2.5
		272.8	Duty ratio specified by word	
		293.8	Duty ratio specified by *DM	
	PID	2.11 ms	Word to word (initial execution)	2.5
		2.30 ms	*DM to *DM (initial execution)	
		607.7	Word to word (when sampling)	
		893.7	*DM to *DM (when sampling)	
	ADBL	75.4	Word + word $\rightarrow$ word	2.5
		152.0	*DM + *DM $\rightarrow$ *DM	
	SBBL	75.4	Word - word $\rightarrow$ word	2.5
		152.0	*DM - *DM $\rightarrow$ *DM	
	MBS	57.9	Constant $\times$ word $\rightarrow$ word	2.5
		61.1	Word $\times$ word $\rightarrow$ word	
		135.0	*DM $\times$ *DM $\rightarrow$ *DM	

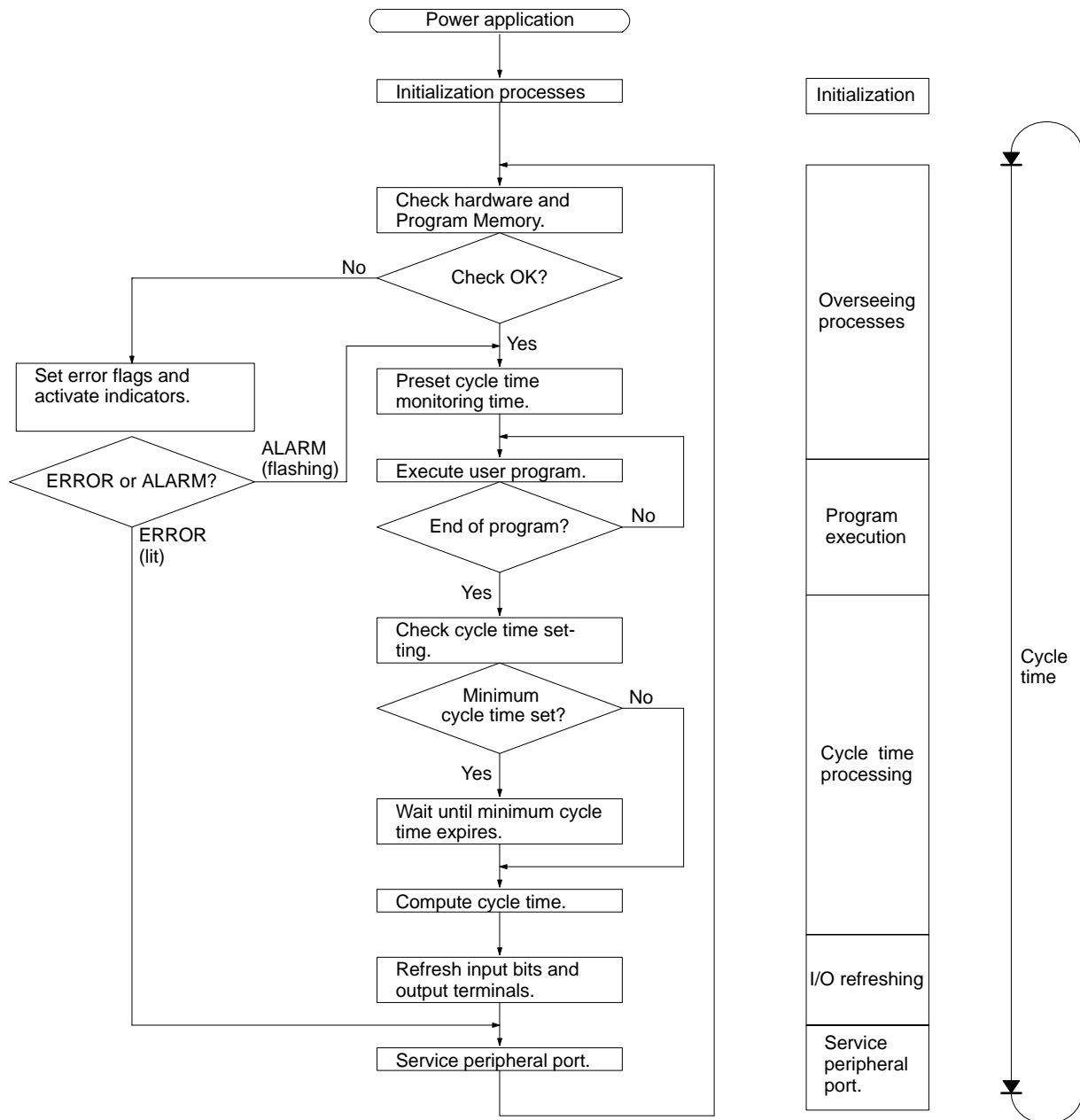
Code	Mnemonic	ON execution time ( $\mu\text{s}$ )	Conditions	OFF execution time ( $\mu\text{s}$ )
	DBS	28.3	Constant $\div$ word $\rightarrow$ word	2.5
		62.4	Word $\div$ word $\rightarrow$ word	
		136.4	*DM $\div$ *DM $\rightarrow$ *DM	
	MBSL	94.0	Word $\times$ word $\rightarrow$ word	2.5
		167.6	*DM $\times$ *DM $\rightarrow$ *DM	
	DBSL	86.3	Word $\div$ word $\rightarrow$ word	2.5
		160.4	*DM $\div$ *DM $\rightarrow$ *DM	
	CPS	31.0	Comparing a constant and word	2.5
		33.7	Comparing words	
		82.4	Comparing *DM	
	CPSL	51.3	Comparing words	2.5
		102.6	Comparing *DM	
	NEG	41.3	Converting a constant $\rightarrow$ word	2.5
		44.5	Converting a word $\rightarrow$ word	
		92.7	Converting *DM $\rightarrow$ *DM	
	NEGL	51.1	Converting a constant $\rightarrow$ words	2.5
		103.2	Converting *DM $\rightarrow$ *DM	
	ZCP	38.2	Comparing a constant to a word range	2.5
		44.7	Comparing a word to a word range	
		114.6	Comparing *DM to a *DM range	
	CPSL	77.7	Comparing words to a word range	2.5
		151.4	Comparing *DM to a *DM range	
	XFRB	35.3	Transferring 1 bit between words with a constant for control data	2.5
		56.8	Transferring 1 bit between words with a word for control data	
		298.3	Transferring 255 bits between *DM with *DM for control data	
	PLS2	821.7	Words for control words	2.5
		849.0	*DM for control words	
	ACC	547.3	Mode 0: Words for control words	2.5
		577.0	Mode 0: *DM for control words	
		392.8	Mode 1: Words for control words	
		424.0	Mode 1: *DM for control words	
		404.8	Mode 2: Words for control words	
		430.3	Mode 2: *DM for control words	
		259.5	Mode 3: Words for control words	
		418.3	Mode 3: *DM for control words	
	SCL2	105.0	Word to word conversion, words for parameter words	2.5
		179.8	*DM to *DM conversion, *DM for parameter words	
	SCL3	112.0	Word to word conversion, words for parameter words	2.5
		186.8	*DM to *DM conversion, *DM for parameter words	



## 7-2 CPM1/CPM1A Cycle Time and I/O Response Time

### 7-2-1 The CPM1/CPM1A Cycle

The overall flow of CPM1/CPM1A operation is as shown in the following flow-chart.



**Note** Initialization processes include clearing the IR, SR, and AR areas, presetting system timers, and checking I/O Units.

## 7-2-2 CPM1/CPM1A Cycle Time

The processes involved in a single CPM1/CPM1A cycle are shown in the following table, and their respective processing times are explained.

Process	Content	Time requirements
Overseeing	Setting cycle watchdog timer, I/O bus check, UM check, clock refreshing, refreshing bits allocated to new functions, etc.	0.6 ms
Program execution	User program is executed.	Total time for executing instructions. (Varies according to content of user's program.)
Cycle time calculation	Standby until set time, when minimum cycle time is set in DM 6619 of PC Setup. Calculation of cycle time.	Almost instantaneous, except for standby processing.
I/O refresh	Input information is read to input bits. Output information (results of executing program) is written to output bits.	10-point CPU : 0.06 ms 20-point CPU: 0.06 ms 30-point CPU: 0.3 ms Expansion I/O Unit 0.3 ms
Peripheral port servicing	Devices connected to peripheral port serviced.	0.26 ms min., 5% or less of cycle time up to 66 ms (see note)

**Note** The percentage of the cycle allocated to peripheral port servicing can be changed in the PC Setup (DM 6617).

**Cycle Time and Operations** The effects of the cycle time on CPM1/CPM1A operations are as shown below. When a long cycle time is affecting operation, either reduce the cycle time or improve responsiveness with interrupt programs.

Cycle time	Operation conditions
10 ms or longer	TIMH(15) may be inaccurate when TC 004 through TC 127 are used (operation will be normal for TC 000 through TC 003).
20 ms or longer	Programming using the 0.02-second Clock Bit (SR 25401) may be inaccurate.
100 ms or longer	TIM may be inaccurate. Programming using the 0.1-second Clock Bit (SR 25500) may be inaccurate. A CYCLE TIME OVER error is generated (SR 25309 will turn ON). See note 1.
120 ms or longer	The FALS 9F monitoring time SV is exceeded. A system error (FALS 9F) is generated, and operation stops. See note 2.
200 ms or longer	Programming using the 0.2-second Clock Bit (SR 25501) may be inaccurate.

**Note**

1. The PC Setup (DM 6655) can be used to disable detection of CYCLE TIME OVER error.
2. The cycle monitoring time can be changed in the PC Setup (DM 6618).

### Cycle Time Example

In this example, the cycle time is calculated for a CPM1/CPM1A CPU Unit with 20 I/O points (12 input points and 8 output points). The I/O is configured as follows:

Inputs: 1 word (00000 to 00011)  
Outputs: 1 word (01000 to 01007)

The rest of the operating conditions are assumed to be as follows:

User's program: 500 instructions (consists of only LD and OUT)  
Cycle time: Variable (no minimum set)

The average processing time for a single instruction in the user's program is assumed to be 2.86 μs. The cycle times are as shown in the following table.

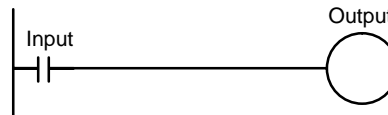
Process	Calculation method	Time with peripheral device	Time without peripheral device
1. Overseeing	Fixed	0.6 ms	0.6 ms
2. Program execution	$2.86 \times 500$ (μs)	1.43 ms	1.43 ms
3. Cycle time calculation	Negligible	0 ms	0 ms
4. I/O refresh	$0.01 \times 1 + 0.005 \times 1$ (μs)	0.06 ms	0.06 ms
5. Peripheral port servicing	Minimum time	0.26 ms	0 ms
Cycle time	(1) + (2) + (3) + (4) + (5)	2.35 ms	2.09 ms

- Note**
1. The cycle time can be read from the PC via a Peripheral Device.
  2. The maximum and current cycle time are stored in AR 14 and AR 15.
  3. The cycle time can vary with actual operating conditions and will not necessarily agree precisely with the calculated value.

### 7-2-3 I/O Response Time

The I/O response time is the time it takes after an input signal has been received (i.e., after an input bit has turned ON) for the PC to check and process the information and to output a control signal (i.e., to output the result of the processing to an output bit). The I/O response time varies according to the timing and processing conditions.

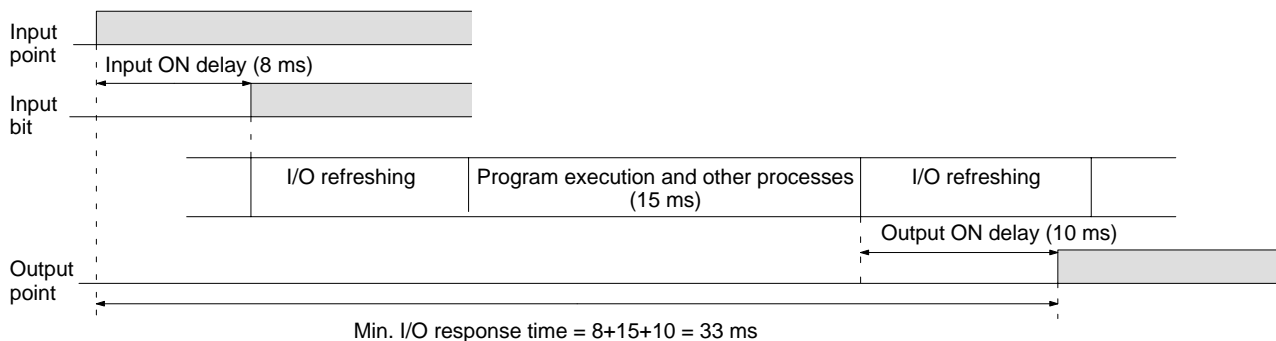
The minimum and maximum I/O response times are shown here, using the following program as an example.



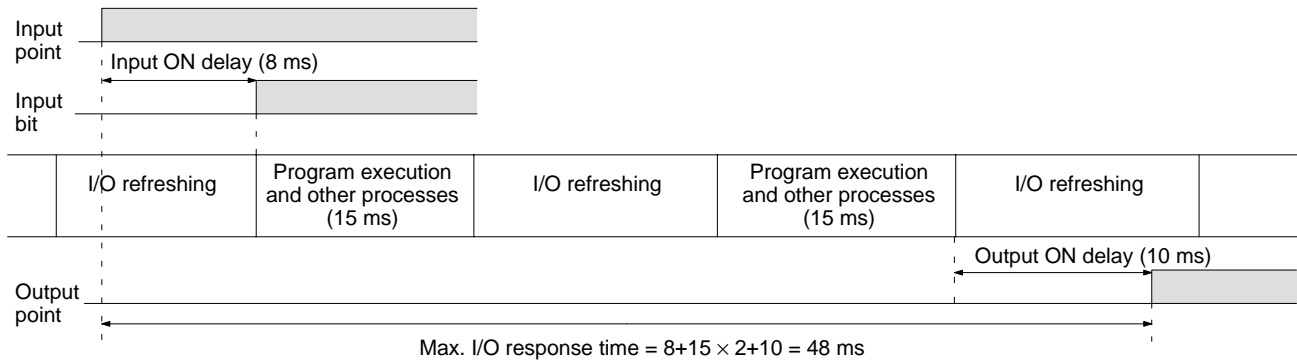
The following conditions are taken as examples for calculating the I/O response times.

- Input ON delay: 8 ms (input time constant: default setting)
- Overseeing time: 1 ms (includes I/O refresh for CPM1A)
- Instruction execution time: 14 ms
- Output ON delay: 10 ms
- Peripheral port: Not used.

**Minimum I/O Response Time** The CPM1/CPM1A responds most quickly when it receives an input signal just prior to I/O refreshing, as shown in the illustration below.



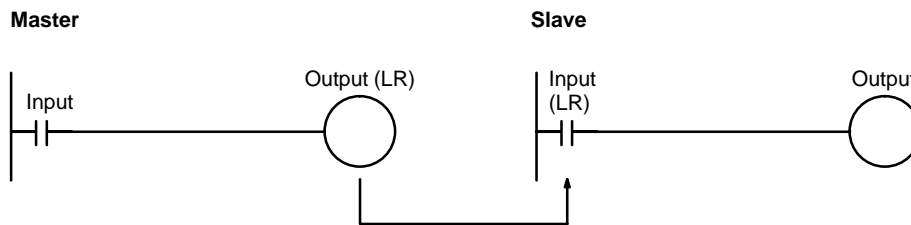
**Maximum I/O Response Time** The CPM1/CPM1A takes longest to respond when it receives the input signal just after the input refresh phase of the cycle, as shown in the illustration below. In that case, a delay of approximately one cycle will occur.



### 7-2-4 One-to-one Link I/O Response Time

When two CPM1/CPM1As are linked one-to-one, the I/O response time is the time required for an input executed at one of the CPM1/CPM1As to be output to the other CPM1/CPM1A by means of one-to-one link communications.

The minimum and maximum I/O response times are shown here, using as an example the following instructions executed at the master and the slave. In this example, communications proceed from the master to the slave.



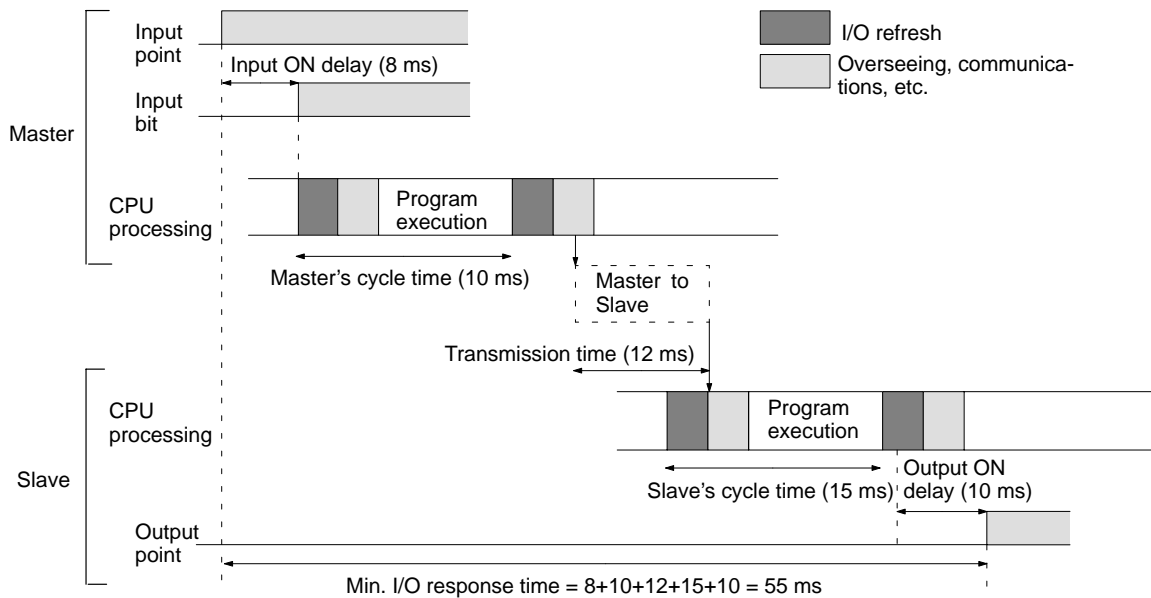
The following conditions are taken as examples for calculating the I/O response times. In CPM1/CPM1A PCs, LR area words LR 00 to LR 15 are used in one-to-one links and the transmission time is fixed at 12 ms.

- Input ON delay: 8 ms (input time constant: default setting)
- Master cycle time: 10 ms
- Slave cycle time: 15 ms
- Output ON delay: 10 ms
- Peripheral port: Not used.

**Minimum I/O Response Time** The CPM1/CPM1A responds most quickly under the following circumstances:

- 1, 2, 3... 1. The CPM1/CPM1A receives an input signal just prior to the input refresh phase of the cycle.
2. The Master's communications servicing occurs just as the master-to-slave transmission begins.

3. The Slave's communications servicing occurs just after the transmission is completed.

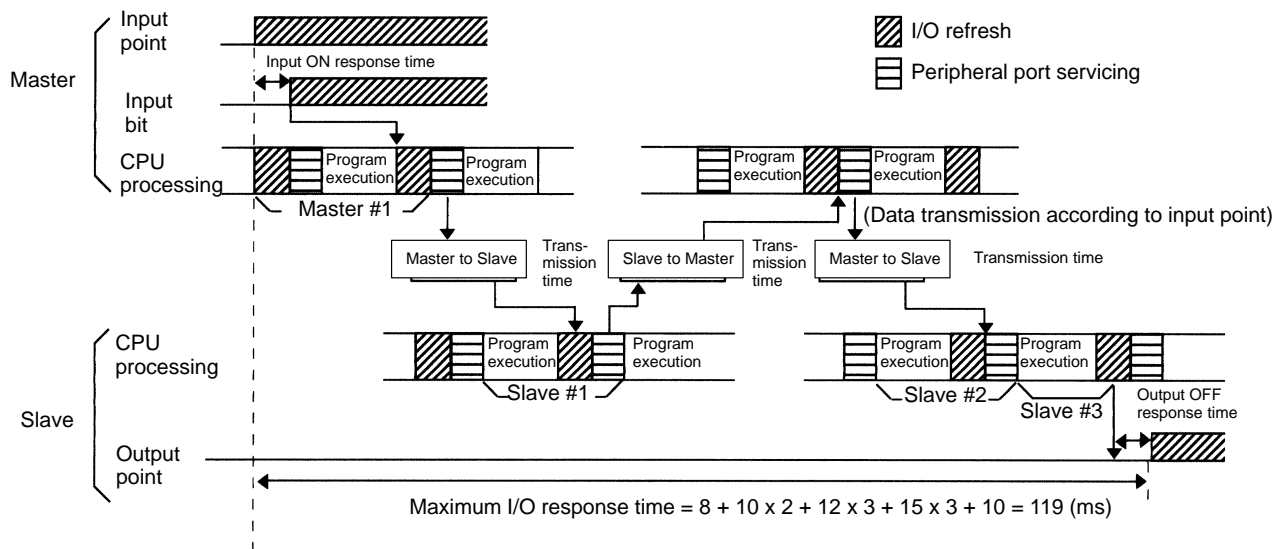


Calculation formula = Input ON response time + Master's cycle time + Slave's cycle time + Output ON response time

**Maximum I/O Response Time** The CPM1/CPM1A takes the longest to respond under the following circumstances:

- 1, 2, 3... 1. The CPM1/CPM1A receives an input signal just after the input refresh phase of the cycle.
2. The Master's communications servicing just misses the master-to-slave transmission.
3. The transmission is completed just after the Slave's communications servicing ends.

**I/O Maximum Response Time** Input ON response time + Master's cycle time x 2 + Transmission time x 3 + Output ON response time



### 7-2-5 Interrupt Processing Time

This section explains the processing times involved from the time an interrupt is executed until the interrupt processing routine is called, and from the time an interrupt processing routine is completed until returning to the initial location. This explanation applies to input interrupts, interval timer interrupts, and high-speed counter interrupts.

- 1, 2, 3... 1. Source of interrupt  
 2. Interrupt ON delay  
 3. Wait for completion of interrupt-mask processing  
 4. Change to interrupt processing  
 5. Interrupt routing (CPM1A only)  
 6. Return to initial location

The table below shows the times involved from the generation of an interrupt signal until the interrupt processing routine is called, and from when the interrupt processing routine is completed until returning to the original position.

Item	Contents	Time
Interrupt ON delay	This is the delay time from the time the interrupt input bit turns ON until the time that the interrupt is executed. This is unrelated to other interrupts.	100 μs
Wait for completion of interrupt-mask processing	This is the time during which interrupts are waiting until processing has been completed. This situation occurs when a mask processes is executed. It is explained below in more detail.	See below.
Change to interrupt processing	This is the time it takes to change processing to an interrupt.	30 μs
Return	This is the time it takes, from execution of RET(93), to return to the processing that was interrupted.	30 μs

#### Mask Processing

Interrupts are masked during processing of the operations described below. Until the processing is completed, any interrupts will remain masked for the indicated times.

Generation and clearing of non-fatal errors:

When a non-fatal error is generated and the error contents are registered at the CPM1, or when an error is being cleared, interrupts will be masked for a maximum of 100 μs until the processing has been completed.

Online editing:

Interrupts will be masked for a maximum of 600 ms (i.e.: editing DM 6144 to DM 6655) when online editing is executed during operation. In addition, the system processing may have to wait for a maximum of 170 μs during this processing.

#### Example Calculation

This example shows the interrupt response time (i.e., the time from when the interrupt input turns ON until the start of the interrupt processing routine) when input interrupts are used under the conditions shown below.

#### Minimum Response Time

Interrupt ON delay:	100 μs
Interrupt mask standby time:	0 μs
+ Change-to-interrupt processing:	30 μs
Minimum response time:	130 μs

#### Maximum Response Time (Except for the Online Editing of DM 6144 to DM6655)

Interrupt ON delay:	100 μs
Interrupt mask standby time:	170 μs
+ Change-to-interrupt processing:	30 μs
Maximum response time:	300 μs

In addition to the response time shown above, the time required for executing the interrupt processing routine itself and a return time of 30 μs must also be accounted for when returning to the process that was interrupted.

### 7-2-6 CPM1/CPM1A Instruction Execution Times

The following table lists the execution times for CPM1/CPM1A instructions.

#### Basic Instructions

Code	Mnemonic	ON execution time (μs)	Conditions (Top: min.; bottom: max.)	OFF execution time (μs)		
				RSET	IL	JMP
---	LD LD NOT	1.72	Any	---		
---	AND AND NOT OR OR NOT	1.32				
---	AND LD OR LD	0.72				
---	OUT OUT NOT	4.0				
---	SET	5.8				
---	RSET	5.9				
---	TIM	10.0		Constant for SV	16.2	16.0
			*DM for SV	31.4	31	6.4
---	CNT	12.5	Constant for SV	14.1	6.2	6.6
			*DM for SV	29.1	6.2	6.6

#### Special Instructions

Code	Mnemonic	ON execution time (μs)	Conditions (Top: min.; bottom: max.)	OFF execution time (μs)		
00	NOP	0.36	Any			
01	END	10.8				
02	IL	4.6		2.6		
03	ILC	3.6		3.6		
04	JMP	4.3		2.4		
05	JME	4.7		4.7		
06	FAL	38.5		5.5		
07	FALS	5.0		5.4		
08	STEP	14.9		11.1		
09	SNXT	14.2		7.6		
10	SFT	21.9	With 1-word shift register	Reset	IL	JMP
		19.7		19.7	2.6	2.6
		34.1	With 10-word shift register	26.5	2.6	2.6
		93.6	With 100-word shift register	60.1	2.6	2.6
11	KEEP	6.2	Any	Reset	IL	JMP
				6.1	3.1	3.1
12	CNTR	25.8	Constant for SV	Reset	IL	JMP
			41.2	*DM for SV	16.8	12.2
13	DIFU	11.8	Any	Shift	IL	JMP
				10.1	12.2	12.2
14	DIFD	11.0	Any	Shift	IL	JMP
				10.0	9.9	2.3

Code	Mnemonic	ON execution time (μs)	Conditions (Top: min.; bottom: max.)	OFF execution time (μs)		
				Reset	IL	JMP
15	TIMH	19.0	Regular execution, constant for SV	25.7	28.4	15.8
		20.2	Interrupt execution, constant for SV			
		19.0	Regular execution, *DM for SV	41.2	43.6	15.8
		20.2	Interrupt execution, *DM for SV			
16	WSFT	29.2	With 1-word shift register	5.6		
		40.7	With 10-word shift register			
		1.42 ms	With 1,024-word shift register using *DM			
17	ASFT	29.6	Shifting a word	5.6		
		50.2	Shifting 10 words			
		1.76 ms	Shifting 1,023 words via *DM			
20	CMP	15.8	When comparing a constant to a word	5.6		
		17.2	When comparing two words			
		46.3	When comparing two *DM			
21	MOV	16.3	When transferring a constant to a word	5.6		
		17.7	When moving from one word to another			
		45.5	When transferring *DM to *DM			
22	MVN	16.4	When transferring a constant to a word	5.6		
		17.5	When moving from one word to another			
		45.7	When transferring *DM to *DM			
23	BIN	31.6	When converting a word to a word	5.6		
		45.7	When converting *DM to *DM			
24	BCD	29.5	When converting a word to a word	5.6		
		57.3	When converting *DM to *DM			
25	ASL	17.3	When shifting a word	5.5		
		31.3	When shifting *DM			
26	ASR	16.9	When shifting a word	5.5		
		31.1	When shifting *DM			
27	ROL	14.5	When rotating a word	5.5		
		28.5	When rotating *DM			
28	ROR	14.5	When rotating a word	5.5		
		28.5	When rotating *DM			
29	COM	18.1	When inverting a word	5.5		
		32.1	When inverting *DM			
30	ADD	29.5	Constant + word → word	5.6		
		30.9	Word + word → word			
		72.7	*DM + *DM → *DM			
31	SUB	29.3	Constant – word → word	5.6		
		30.5	Word – word → word			
		72.5	*DM – *DM → *DM			
32	MUL	49.1	Constant × word → word	5.6		
		50.5	Word × word → word			
		95.1	*DM × *DM → *DM			
33	DIV	47.7	Word ÷ constant → word	5.6		
		50.9	word ÷ word → word			
		94.3	*DM ÷ *DM → *DM			



Code	Mnemonic	ON execution time ( $\mu$ s)	Conditions (Top: min.; bottom: max.)	OFF execution time ( $\mu$ s)
34	ANDW	27.1	Constant $\square$ word $\rightarrow$ word	5.6
		28.7	Word $\square$ word $\rightarrow$ word	
		70.7	*DM $\square$ *DM $\rightarrow$ *DM	
35	ORW	27.1	Constant $\vee$ word $\rightarrow$ word	5.6
		28.7	Word $\vee$ word $\rightarrow$ word	
		70.7	*DM $\vee$ *DM $\rightarrow$ *DM	
36	XORW	27.1	Constant $\nabla$ word $\rightarrow$ word	5.6
		28.7	Word $\nabla$ word $\rightarrow$ word	
		70.5	*DM $\nabla$ *DM $\rightarrow$ *DM	
37	XNRW	27.0	Constant $\overline{\vee}$ word $\rightarrow$ word	5.6
		28.6	Word $\overline{\vee}$ word $\rightarrow$ word	
		70.5	*DM $\overline{\vee}$ *DM $\rightarrow$ *DM	
38	INC	17.9	When incrementing a word	5.5
		31.9	When incrementing *DM	
39	DEC	18.3	When decrementing a word	5.5
		32.3	When decrementing *DM	
40	STC	6.3	Any	5.5
41	CLC	6.3		5.5
46	MSG	21.5	With message in words	5.5
		35.7	With message in *DM	
50	ADB	30.5	Constant + word $\rightarrow$ word	5.6
		32.1	Word + word $\rightarrow$ word	
		73.9	*DM + *DM $\rightarrow$ *DM	
51	SBB	30.9	Constant - word $\rightarrow$ word	5.6
		32.7	Word - word $\rightarrow$ word	
		74.5	*DM - *DM $\rightarrow$ *DM	
52	MLB	34.7	Constant $\times$ word $\rightarrow$ word	5.6
		36.3	Word $\times$ word $\rightarrow$ word	
		80.7	*DM $\times$ *DM $\rightarrow$ *DM	
53	DVB	35.1	Word $\div$ constant $\rightarrow$ word	5.6
		36.7	Word $\div$ word $\rightarrow$ word	
		81.1	*DM $\div$ *DM $\rightarrow$ *DM	
54	ADDL	48.9	Word + word $\rightarrow$ word	5.6
		94.7	*DM + *DM $\rightarrow$ *DM	
55	SUBL	48.9	Word - word $\rightarrow$ word	5.6
		94.7	*DM - *DM $\rightarrow$ *DM	
56	MULL	138.7	Word $\times$ word $\rightarrow$ word	5.6
		184.3	*DM $\times$ *DM $\rightarrow$ *DM	
57	DIVL	136.7	Word $\div$ word $\rightarrow$ word	5.6
		181.3	*DM $\div$ *DM $\rightarrow$ *DM	
60	CMPL	30.4	Comparing words	5.6
		60.8	Comparing *DM	

Code	Mnemonic	ON execution time (μs)	Conditions (Top: min.; bottom: max.)	OFF execution time (μs)
61	INI	112.0	Starting comparison via word	5.6
		126.0	Starting comparison via *DM	
		48.0	Stopping comparison via word	
		48.0	Stopping comparison via *DM	
		120.0	Changing PV via word	
		128.0	Changing PV via *DM	
		46.0	Stopping pulse output via word	
		60.0	Stopping pulse output via *DM	
62	PRV	62.2	Designating output via word	5.6
		78.0	Designating output via *DM	
63	CTBL	106.3	Target table with 1 target in words and start	5.6
		120.3	Target table with 1 target in *DM and start	
		775.5	Target table with 16 targets in words and start	
		799.5	Target table with 16 targets in *DM and start	
		711.5	Range table in words and start	
		722.5	Range table in *DM and start	
		91.9	Target table with 1 target in words	
		106.3	Target table with 1 target in *DM	
		693.5	Target table with 16 targets in words	
		709.5	Target table with 16 targets in *DM	
		607.5	Range table in words	
		621.5	Range table in *DM	
64	SPED	73.6	Specifying a constant	5.6
		75.0	Specifying a word	
		88.8	Specifying *DM	
65	PULS	62.0	Specifying a word	5.6
		78.0	Specifying *DM	
67	BCNT	52.6	Counting a word	5.6
		4.08 ms	Counting 6,656 words via *DM	
68	BCMP	79.6	Comparing constant, results to word	5.6
		80.8	Comparing word, results to word	
		123.2	Comparing *DM, results to *DM	
69	STIM	47.5	Word-set one-shot interrupt start	5.6
		58.7	*DM-set one-shot interrupt start	
		47.9	Word-set scheduled interrupt start	
		59.1	*DM-set scheduled interrupt start	
		33.5	Word-set timer read	
		63.5	*DM-set timer read	
		25.7	Word-set timer stop	
		54.1	*DM-set timer stop	
70	XFER	45.5	When transferring a constant to a word	5.6
		47.1	When transferring a word to a word	
		1.78 ms	When transferring 1,024 words using *DM	
71	BSET	28.1	When setting a constant to 1 word	5.6
		38.3	When setting word constant to 10 words	
		1.12 ms	When setting *DM to 1,024 words	
73	XCHG	30.5	Word → word	5.6
		59.1	*DM → *DM	

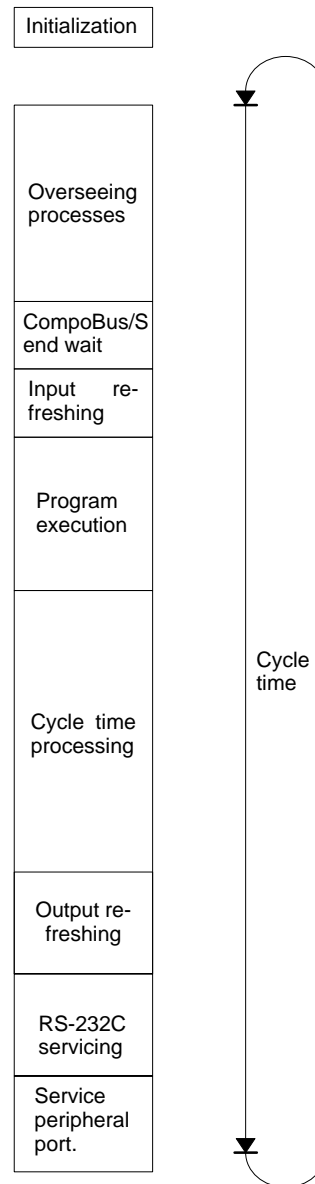
Code	Mnemonic	ON execution time (μs)	Conditions (Top: min.; bottom: max.)	OFF execution time (μs)
74	SLD	25.9	Shifting 1 word	5.6
		51.7	Shifting 10 word	
		3.02 ms	Shifting 1024 words using *DM	
75	SRD	25.9	Shifting 1 word	5.6
		51.7	Shifting 10 word	
		3.02 ms	Shifting 1,024 words using *DM	
76	MLPX	47.7	When decoding word to word	5.6
		92.7	When decoding *DM to *DM	
77	DMPX	59.5	When encoding word to word	5.6
		95.5	When encoding *DM to *DM	
78	SDEC	51.1	When decoding word to word	5.6
		96.3	When decoding *DM to *DM	
80	DIST	39.1	When setting a constant to a word + a word	5.6
		40.9	When setting a word to a word + a word	
		84.7	When setting *DM to *DM + *DM	
		63.4	When setting a constant to a stack	
		65.0	When setting a word to a stack	
		109.6	When setting *DM to a stack via *DM	
81	COLL	42.6	When setting a constant + a word to a word	5.6
		43.6	When setting a word + a word to a word	
		83.4	When setting *DM + *DM to *DM	
		78.0	When setting a word + constant to FIFO stack	
		79.2	When setting a word + word to FIFO stack	
		1.76 ms	When setting a *DM + *DM to FIFO stack via *DM	
		66.8	When setting a word + constant to LIFO stack	
		68.0	When setting a word + word to LIFO stack	
		112.0	When setting a *DM + *DM to LIFO stack via *DM	
82	MOVB	32.5	When moving constant to word	5.6
		37.5	When moving word to word	
		79.1	When moving *DM to *DM	
83	MOVD	28.3	When moving constant to word	5.6
		33.3	When moving word to word	
		75.5	When moving *DM to *DM	
84	SFTR	39.3	Shifting 1 word	5.6
		52.9	Shifting 10 word	
		1.42 ms	Shifting 1,024 words using *DM	
85	TCMP	57.7	Comparing constant to word-set table	5.6
		58.9	Comparing word to word-set table	
		101.9	Comparing *DM to *DM-set table	
86	ASC	56.7	Word → word	5.6
		103.9	*DM → *DM	

Code	Mnemonic	ON execution time ( $\mu$ s)	Conditions (Top: min.; bottom: max.)	OFF execution time ( $\mu$ s)
89	INT	32.3	Set masks via word	5.6
		46.3	Set masks via *DM	
		29.1	Clear interrupts via word	
		43.1	Clear interrupts via *DM	
		27.3	Read mask status via word	
		41.5	Read mask status via *DM	
		29.7	Change counter SV via word	
		43.7	Change counter SV via *DM	
		15.3	Mask all interrupts via word	
		15.3	Mask all interrupts via *DM	
		15.9	Clear all interrupts via word	
		15.9	Clear all interrupts via *DM	
91	SBS	36.6	Any	5.5
92	SBN	1.7		1.7
93	RET	15.0		2.5
97	IORF	40.0	Refreshing IR 000	6.0
		142.6	Refreshing one input word	
		135.4	Refreshing one output word	
99	MCRO	74.0	With word-set I/O operands	5.6
		116.4	With *DM-set I/O operands	

## 7-3 SRM1 Cycle Time and I/O Response Time

### 7-3-1 The SRM1 Cycle

The overall flow of SRM1 operation is as shown in the following flowchart.



- Note**
1. The cycle time can be read using Peripheral Devices.
  2. Cycle time maximum and current cycle time are stored in AR 14 and AR 15.
  3. Change to processing will cause cycle times to change therefore the calculated values and actual values (for cycle time) will no always match.

## 7-3-2 SRM1 Cycle Time

The processes involved in a single SRM1 cycle are shown in the following table, and their respective processing times are explained.

Process	Content	Time requirements
Overseeing	Setting cycle watchdog timer, UM check, refreshing bits allocated to new functions, etc.	0.18 ms
CompoBus/S end wait	Waiting for CompoBus/S processing to finish	CompoBus/S communications response time – Overseeing time – RS-232C port servicing time – peripheral port servicing time
Input refreshing	Input information is read to input bits.	0.02 ms
Program execution	User program is executed. Refer to 7-3-6 SRM1 Instruction Execution Times.	Total time for executing instructions. (Varies according to content of user's program.)
Cycle time calculation	Standby until set time, when minimum cycle time is set in DM 6619 of PC Setup. Calculation of cycle time.	Almost instantaneous, except for standby processing.
Output refreshing	Output information (results of executing program) is written to output bits. CompoBus/S communications are started.	0.05 ms
RS-232C port servicing	Devices connected to RS-232C port serviced.	5% or less of cycle time, but always between 0.55 and 131 ms (Set in DM 6616)
Peripheral port servicing	Devices connected to peripheral port serviced.	5% or less of cycle time, but always between 0.55 and 131 ms (Set in DM 6617)

### Minimum Cycle Time

In SRM1 PCs, CompoBus/S communications are started after the output refresh is completed. As a result, when the overseeing time plus the RS-232C port servicing time plus the peripheral port servicing time is shorter than the CompoBus/S communications response time, processing is placed on stand-by until CompoBus/S communications are completed.

The minimum cycle time therefore is the the CompoBus/S communications response time plus the program execution time plus the input refresh time plus the output refresh time. The CompoBus/S communications response time depends on the maximum number of nodes set, as follows:

Max. no. of nodes set	CompoBus/S response time
32	0.8 ms
16	0.5 ms

### Cycle Time and Operations

The effects of the cycle time on SRM1 operations are as shown below. When a long cycle time is affecting operation, either reduce the cycle time or improve responsiveness with interrupt programs.

Cycle time	Operation conditions
10 ms or longer	TIMH(15) may be inaccurate when TC 004 through TC 127 are used (operation will be normal for TC 000 through TC 003).
20 ms or longer	Programming using the 0.02-second Clock Bit (SR 25401) may be inaccurate.
100 ms or longer	TIM may be inaccurate. Programming using the 0.1-second Clock Bit (SR 25500) may be inaccurate. A CYCLE TIME OVER error is generated (SR 25309 will turn ON). See note 1.
120 ms or longer	The FALS 9F monitoring time SV is exceeded. A system error (FALS 9F) is generated, and operation stops. See note 2.
200 ms or longer	Programming using the 0.2-second Clock Bit (SR 25501) may be inaccurate.

- Note**
1. The PC Setup (DM 6655) can be used to disable detection of CYCLE TIME OVER error.
  2. The cycle monitoring time can be changed in the PC Setup (DM 6618).

**Cycle Time Example**

The following is an example of a cycle time calculation.

The operating conditions are assumed to be as follows:

User's program: 500 instructions (consists of only LD and OUT)

Cycle time: Variable (no minimum set)

RS-232C port: Not used.

Max. nodes: 32 (CompoBus/S communication response time = 0.8 ms)

Peripheral: 0.7 ms

The average processing time for a single instruction in the user's program is assumed to be 1.16  $\mu$ s. The cycle times are as shown in the following table.

Process	Calculation method	Peripheral port used	Peripheral port not used
1. Overseeing	Fixed	0.18 ms	0.18 ms
2. CompoBus/S end wait	See previous page.	0.00 ms	0.62 ms
3. Input refresh	Fixed	0.02 ms	0.02 ms
4. Program execution	$1.16 \times 500 (\mu\text{s})$	0.8 ms	0.8 ms
5. Cycle time calculation	Negligible	0.00 ms	0.00 ms
6. Output refresh	$0.01 \times 1 + 0.005 \times 1 (\mu\text{s})$	0.05 ms	0.05 ms
7. RS-232C port servicing	Not required	0.00 ms	0.00 ms
8. Peripheral port servicing	5% of cycle time	0.7 ms	0.00 ms
Cycle time	(1) + (2) + (3) + ... + (8)	1.75 ms	1.67 ms

- Note**
1. The cycle time can be read from the PC via a Peripheral Device.
  2. The maximum and current cycle time are stored in AR 14 and AR 15.
  3. The cycle time can vary with actual operating conditions and will not necessarily agree precisely with the calculated value.
  4. When the peripheral port is used, there is no CompoBus/S end wait time as it is always 0 or less.
  5. CompoBus/S end wait time =  $0.8 - 0.18 - 0 - 0 = 0.62$  (CompoBus/S communication response time – Overseeing – RS-232C port servicing time – peripheral port servicing time).

### 7-3-3 I/O Response Time

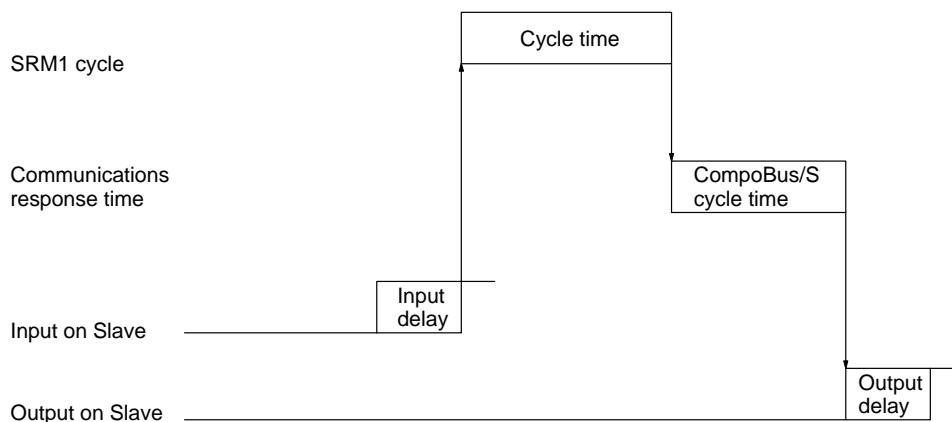
The I/O response time is the time it takes after an input signal has been received (i.e., after an input bit has turned ON) for the PC to check and process the information and to output a control signal (i.e., to output the result of the processing to an output bit).

CompoBus/S communications are started when the SRM1 input refresh finishes. The ON/OFF status is read from the Input Terminals during the input refresh and the ON/OFF status is output to the Output Terminal during the output refresh. Accordingly, the SRM1 I/O response time varies according to the cycle time and CompoBus/S communications cycle status or I/O timing.

Example calculations of the I/O response time are provided next.

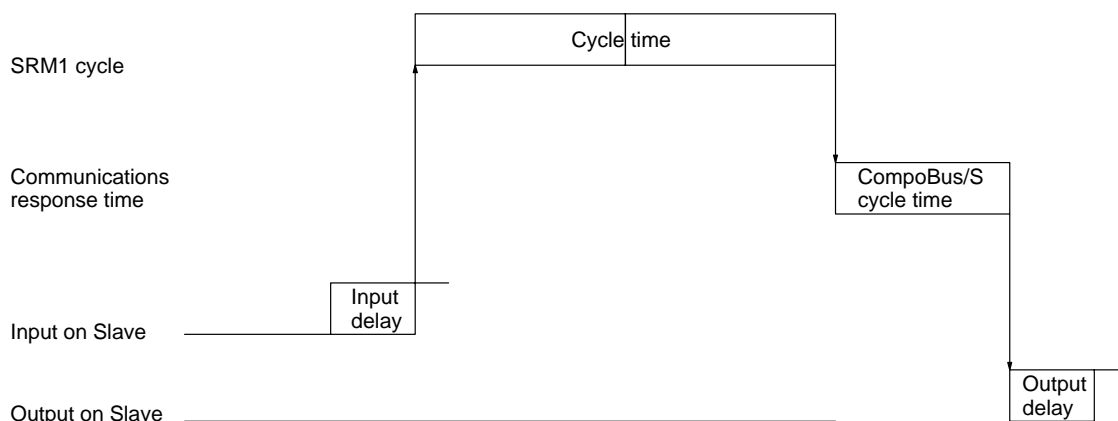
#### Minimum I/O Response Time

Minimum I/O response time =  
 Input ON delay + Output ON delay + CompoBus/S communications cycle time + SRM1 cycle time



#### Maximum I/O Response Time

Maximum I/O response time =  
 Input ON delay + Output ON delay + CompoBus/S communications cycle time + SRM1 cycle time x 2



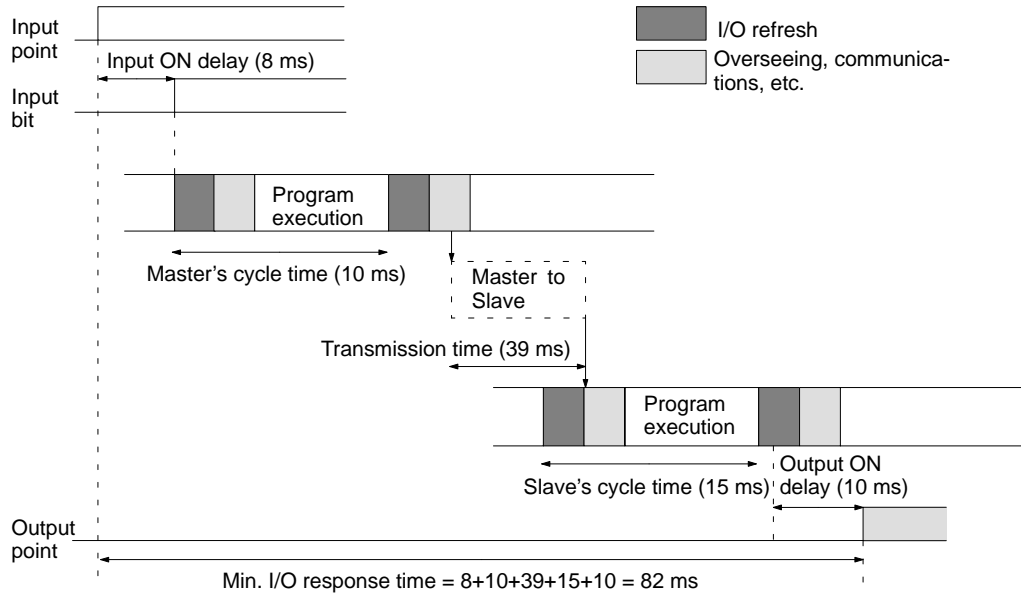
### 7-3-4 One-to-one Link I/O Response Time

When two SRM1s are linked one-to-one, the I/O response time is the time required for an input executed at one of the SRM1s to be output to the other SRM1 by means of one-to-one link communications.

**Minimum I/O Response Time** The SRM1 responds most quickly under the following circumstances:

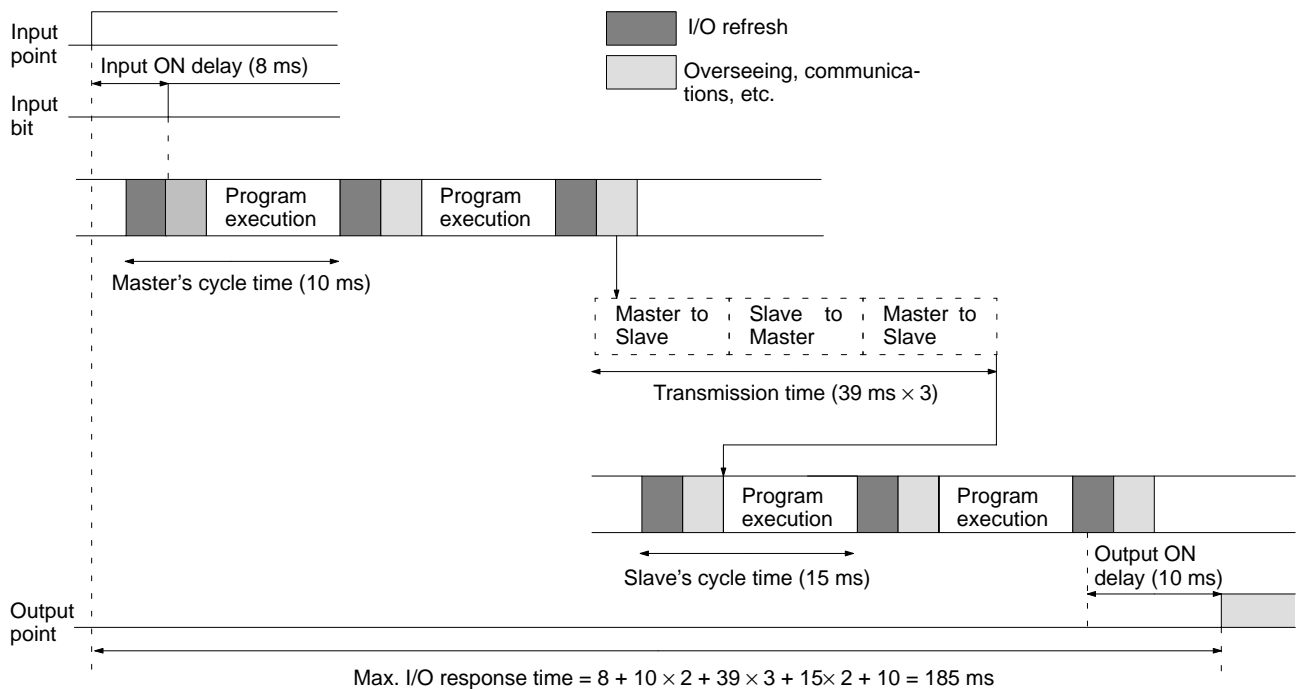


- 1, 2, 3...
1. The SRM1 receives an input signal just prior to the I/O refresh phase of the cycle.
  2. The Master's communications servicing occurs just as the master-to-slave transmission begins.
  3. The Slave's communications servicing occurs just after the transmission is completed.



**Maximum I/O Response Time** The SRM1 takes the longest to respond under the following circumstances:

- 1, 2, 3...
1. The SRM1 receives an input signal just after the I/O refresh phase of the cycle.
  2. The Master's communications servicing just misses the master-to-slave transmission.
  3. The transmission is completed just after the Slave's communications servicing ends.



## 7-3-5 Interrupt Processing Time

This section explains the processing times involved from the time an interrupt is executed until the interrupt processing routine is called, and from the time an interrupt processing routine is completed until returning to the initial location. This explanation applies to input, interval timer interrupts.

- 1, 2, 3...**
1. Source of interrupt
  2. Wait for completion of interrupt-mask processing
  3. Change to interrupt processing
  4. Interrupt routing (CPM1A only)
  5. Return to initial location

The table below shows the times involved from the generation of an interrupt signal until the interrupt processing routine is called, and from when the interrupt processing routine is completed until returning to the original position.

Item	Contents	Time
Wait for completion of interrupt-mask processing	This is the time during which interrupts are waiting until processing has been completed. This situation occurs when a mask processes is executed. It is explained below in more detail.	See below.
Change to interrupt processing	This is the time it takes to change processing to an interrupt.	15 $\mu$ s
Return	This is the time it takes, from execution of RET(93), to return to the processing that was interrupted.	15 $\mu$ s

### Mask Processing

Interrupts are masked during processing of the operations described below. Until the processing is completed, any interrupts will remain masked for the indicated times.

Generation and clearing of non-fatal errors:

When a non-fatal error is generated and the error contents are registered at the SRM1, or when an error is being cleared, interrupts will be masked for a maximum of 100  $\mu$ s until the processing has been completed.

Online editing:

Interrupts will be masked for a maximum of 600 ms (i.e.: editing DM 6144 to DM 6655) when online editing is executed during operation. In addition, the system processing may have to wait for a maximum of 170  $\mu$ s during this processing.

## 7-3-6 SRM1 Instruction Execution Times

The following table lists the execution times for SRM1 instructions.

## Basic Instructions

Code	Mnemonic	ON execution time (μs)	Conditions (Top: min.; bottom: max.)	OFF execution time (μs)		
				RSET	IL	JMP
---	LD	0.97	Any	---		
---	LD NOT	0.97				
---	AND AND NOT	0.77				
---	OR OR NOT	0.78	Any	---		
---	AND LD OR LD	0.39	Any	---		
---	OUT OUT NOT	2.2				
---	SET	2.7				
---	RSET	2.8				
---	TIM	5.7				
			Constant for SV	9.3	9.1	3.5
			*DM for SV	17.4	17.2	3.5
---	CNT	6.6	Constant for SV	8.0	3.6	3.8
			*DM for SV	16.3	3.6	3.8

## Special Instructions and Expansion Instructions

Code	Mnemonic	ON execution time (μs)	Conditions (Top: min.; bottom: max.)	OFF execution time (μs)		
00	NOP	0.20	Any			
01	END	4.8				
02	IL	2.5		1.4		
03	ILC	1.9		1.9		
04	JMP	2.2		1.3		
05	JME	2.5		2.5		
06	FAL	18.4		2.9		
07	FALS	3.6		2.9		
08	STEP	10.7		9.0		
09	SNXT	5.9		4.1		
10	SFT	14.5	With 1-word shift register	Reset	IL	JMP
			With 10-word shift register	11.0	1.4	1.4
		21.0	With 100-word shift register	14.9	1.4	1.4
	49.1		30.8	1.4	1.4	
11	KEEP	3.0	Any	Reset	IL	JMP
				3.4	1.6	1.7
12	CNTR	14.8	Constant for SV	Reset	IL	JMP
			23.2	*DM for SV	9.1	6.6
13	DIFU	6.7	Any	Shift	IL	JMP
				5.8	5.2	1.3
14	DIFD	6.4	Any	Shift	IL	JMP
				5.8	5.7	1.3

Code	Mnemonic	ON execution time (μs)	Conditions (Top: min.; bottom: max.)	OFF execution time (μs)		
				Reset	IL	JMP
15	TIMH	10.3	Regular execution, constant for SV	14.1	13.9	7.0
		10.9	Interrupt execution, constant for SV	15.6	15.4	8.5
		10.3	Regular execution, *DM for SV	22.8	22.1	7.0
		10.9	Interrupt execution, *DM for SV	23.9	23.6	8.5
16	WSFT	16.2	With 1-word shift register	2.9		
		23.0	With 10-word shift register			
		712.3	With 1,024-word shift register using *DM			
17	ASFT*	18.6	Shifting a word	3.0		
		25.9	Shifting 10 words			
		865.7	Shifting 1,023 words via *DM			
20	CMP	9.1	When comparing a constant to a word	3.0		
		9.9	When comparing two words			
		25.6	When comparing two *DM			
21	MOV	9.1	When transferring a constant to a word	3.0		
		9.5	When moving from one word to another			
		24.9	When transferring *DM to *DM			
22	MVN	9.3	When transferring a constant to a word	3.0		
		9.8	When moving from one word to another			
		25.1	When transferring *DM to *DM			
23	BIN	17.2	When converting a word to a word	3.0		
		32.0	When converting *DM to *DM			
24	BCD	15.8	When converting a word to a word	3.0		
		30.6	When converting *DM to *DM			
25	ASL	9.9	When shifting a word	2.9		
		17.3	When shifting *DM			
26	ASR	9.7	When shifting a word	3.0		
		17.2	When shifting *DM			
27	ROL	8.5	When rotating a word	2.9		
		16.1	When rotating *DM			
28	ROR	8.5	When rotating a word	2.9		
		16.1	When rotating *DM			
29	COM	10.5	When inverting a word	3.0		
		17.7	When inverting *DM			
30	ADD	15.9	Constant + word → word	3.1		
		16.4	Word + word → word			
		39.5	*DM + *DM → *DM			
31	SUB	15.6	Constant – word → word	3.0		
		16.3	Word – word → word			
		38.6	*DM – *DM → *DM			
32	MUL	29.7	Constant × word → word	3.0		
		28.5	Word × word → word			
		51.6	*DM × *DM → *DM			
33	DIV	27.2	Word ÷ constant → word	2.9		
		28.5	word ÷ word → word			
		53.1	*DM ÷ *DM → *DM			

Code	Mnemonic	ON execution time (μs)	Conditions (Top: min.; bottom: max.)	OFF execution time (μs)
34	ANDW	14.3	Constant □ word → word	2.9
		15.2	Word □ word → word	
		37.3	*DM □ *DM → *DM	
35	ORW	14.3	Constant V word → word	2.9
		15.2	Word V word → word	
		37.3	*DM V *DM → *DM	
36	XORW	14.3	Constant ∨ word → word	2.9
		15.2	Word ∨ word → word	
		37.3	*DM ∨ *DM → *DM	
37	XNRW	14.3	Constant ∇ word → word	2.9
		15.2	Word ∇ word → word	
		37.3	*DM ∇ *DM → *DM	
38	INC	9.9	When incrementing a word	2.9
		17.3	When incrementing *DM	
39	DEC	10.2	When decrementing a word	2.9
		17.4	When decrementing *DM	
40	STC	3.5	Any	2.9
41	CLC	3.0		2.9
46	MSG	11.3	With message in words	2.9
		19.4	With message in *DM	
47	RXD*	39.1	Word specification, 1 byte input	2.9
		116.8	*DM specification, 256 bytes input	
48	TXD*	31.3	Word specification, 1 byte input (RS-232C)	2.9
		266.5	*DM specification, 256 bytes input (RS-232C)	
		26.7	Word specification, 1 byte input (Host Link)	
		34.0	*DM specification, 256 bytes input (Host Link)	
50	ADB	16.8	Constant + word → word	3.0
		17.6	Word + word → word	
		39.9	*DM + *DM → *DM	
51	SBB	17.0	Constant – word → word	3.0
		17.8	Word – word → word	
		40.2	*DM – *DM → *DM	
52	MLB	19.1	Constant × word → word	3.0
		20.1	Word × word → word	
		43.5	*DM × *DM → *DM	
53	DVB	19.5	Word ÷ constant → word	3.0
		20.4	Word ÷ word → word	
		43.7	*DM ÷ *DM → *DM	
54	ADDL	26.7	Word + word → word	3.0
		49.9	*DM + *DM → *DM	
55	SUBL	26.8	Word – word → word	3.0
		49.9	*DM – *DM → *DM	
56	MULL	81.4	Word × word → word	3.0
		106.2	*DM × *DM → *DM	
57	DIVL	76.9	Word ÷ word → word	3.0
		101.8	*DM ÷ *DM → *DM	
60	Cmpl	16.9	Comparing words	2.9
		32.9	Comparing *DM	

Code	Mnemonic	ON execution time (μs)	Conditions (Top: min.; bottom: max.)	OFF execution time (μs)
67	BCNT*	26.9	Counting a word	3.0
		2.29 ms	Counting 6,656 words via *DM	
68	BCMP*	41.4	Comparing constant, results to word	3.0
		41.9	Comparing word, results to word	
		64.5	Comparing *DM, results to *DM	
69	STIM*	34.7	Word specification, one-shot timer start	3.0
		49.5	*DM specification, one-shot timer start	
		35.3	Word specification, scheduled interrupt start	
		50.0	*DM specification, scheduled interrupt start	
		33.9	Words specification, timer read	
		49.5	*DM specification timer read	
		11.4	Word specification, timer stop	
70	XFER	22.9	When transferring a constant to a word	3.0
		24.0	When transferring a word to a word	
		902.0	When transferring 1,024 words using *DM	
71	BSET	15.2	When setting a constant to 1 word	3.0
		15.7	When setting word constant to 10 words	
		565.2	When setting *DM to 1,024 words	
73	XCHG	16.2	Word → word	3.1
		31.5	*DM → *DM	
74	SLD	13.6	Shifting 1 word	3.0
		26.7	Shifting 10 word	
		1.54 ms	Shifting 1024 words using *DM	
75	SRD	13.6	Shifting 1 word	3.0
		26.6	Shifting 10 word	
		1.54 ms	Shifting 1,024 words using *DM	
76	MLPX	25.5	When decoding word to word	3.0
		48.9	When decoding *DM to *DM	
77	DMPX	35.1	When encoding word to word	3.0
		58.1	When encoding *DM to *DM	
78	SDEC	26.8	When decoding word to word	2.9
		49.9	When decoding *DM to *DM	
80	DIST	21.3	When setting a constant to a word + a word	3.0
		21.9	When setting a word to a word + a word	
		45.7	When setting *DM to *DM + *DM	
		34.3	When setting a constant to a stack	
		35.3	When setting a word to a stack	
		59.3	When setting *DM to a stack via *DM	
81	COLL	21.4	When setting a constant + a word to a word	3.0
		21.8	When setting a word + a word to a word	
		44.9	When setting *DM + *DM to *DM	
		34.0	When setting a word + constant to FIFO stack	
		33.9	When setting a word + word to FIFO stack	
		892.0	When setting a *DM + *DM to FIFO stack via *DM	
		35.4	When setting a word + constant to LIFO stack	
		36.1	When setting a word + word to LIFO stack	
		60.5	When setting a *DM + *DM to LIFO stack via *DM	

Code	Mnemonic	ON execution time (μs)	Conditions (Top: min.; bottom: max.)	OFF execution time (μs)
82	MOVB	18.2	When moving constant to word	3.0
		19.0	When moving word to word	
		42.1	When moving *DM to *DM	
83	MOVD	16.3	When moving constant to word	2.9
		17.6	When moving word to word	
		39.9	When moving *DM to *DM	
84	SFTR	21.0	Shifting 1 word	3.0
		26.9	Shifting 10 word	
		718.5	Shifting 1,024 words using *DM	
85	TCMP	30.0	Comparing constant to word-set table	3.0
		30.7	Comparing word to word-set table	
		53.1	Comparing *DM to *DM-set table	
86	ASC	30.0	Word → word	3.0
		53.7	*DM → *DM	
91	SBS	13.2	Any	3.0
92	SBN	---		1.3
93	RET	7.8		1.3
99	MCRO	26.8	With word-set I/O operands	3.0
		43.5	With *DM-set I/O operands	

**Note** Those instructions marked with an asterisk are expansion instructions.

#### Exchangeable Expansion Instructions

Code	Mnemonic	ON execution time (μs)	Conditions (Top: min.; bottom: max.)	OFF execution time (μs)
---	HEX	43.6	DM → DM	3.0
		73.5	*DM → *DM	
---	FCS	23.4	Adding one word and outputting to word	3.0
		643.7	Adding 999 words and outputting to *DM	
---	STUP	51.2	Transferring constant to word	3.0
		58.2	Transferring word to word	

# SECTION 8

## Troubleshooting

This section describes how to diagnose and correct the hardware and software errors that can occur during PC operation.

8-1	Introduction .....	424
8-2	Programming Console Operation Errors .....	424
8-3	Programming Errors .....	425
8-4	User-defined Errors .....	426
8-5	Operating Errors .....	427
8-5-1	Non-fatal Errors .....	427
8-5-2	Fatal Errors .....	429
8-5-3	Other Errors .....	430
8-6	Error Log .....	430
8-7	Host Link Errors .....	432
8-8	Troubleshooting Flowcharts .....	434
8-8-1	CPM1 Flowcharts .....	434
8-8-2	CPM1A Flowcharts .....	434
8-8-3	SRM1 Flowcharts .....	434
8-8-4	CQM1 Flowcharts .....	434



## 8-1 Introduction

PC errors can be divided broadly into the following four categories:

1, 2, 3...

### 1. Program Input Errors

These errors occur when inputting a program or attempting an operation used to prepare the PC for operation.

### 2. Programming Errors

These errors will occur when the program is checked using the Program Check operation.

### 3. User-defined Errors

There are three instructions that the user can use to define his own errors or messages. The instructions will be executed when a particular condition (defined by the user) has occurred during operation.

### 4. Operating Errors

These errors occur after program execution has been started.

#### a) Non-fatal Operating Errors

PC operation and program execution will continue after one or more of these errors have occurred.

#### b) Fatal Operating Errors

PC operation and program execution will stop and all outputs from the PC will be turned OFF when any of these errors have occurred.

The PC's indicators will indicate when a PC error has occurred and an error message or code will be displayed on the Programming Console or host computer if one is connected. The error code is also contained in SR 25300 to SR 25307.

For the most recent errors, both the type of error and time of occurrence will be recorded in the PC's error log area. Details are provided starting on page 430.

There are flags and other information provided in the SR and AR areas that can be used in troubleshooting. Refer to *Section 3 Memory Areas* for lists of these.

**Note** In addition to the errors described above, communications errors can occur when the PC is part of a Host Link System. Refer to page 432 for details.

## 8-2 Programming Console Operation Errors

The following error messages may appear when performing operations on the Programming Console. Correct the error as indicated and continue with the operation. The asterisks in the displays shown below will be replaced with numeric data, normally an address, in the actual display. Refer to the *Ladder Support Software Operation Manual*, *SYSMAC Support Software Operation Manual: C-series PCs*, or *Data Access Console Operation Manual* for errors that may appear when operating the SSS or a Data Access Console.

Message	Meaning and appropriate response
REPL ROM	An attempt was made to write to write-protected memory. In CQM1 PCs, set the write-protect switch (pin 1 of the CPU Unit's DIP switch) to OFF. In CPM1/CPM1A/SRM1 PCs, set bits 00 to 03 of DM 6602 to "0."
PROG OVER	The instruction at the last address in memory is not NOP(00). Erase all unnecessary instructions at the end of the program.
ADDR OVER	An address was set that is larger than the highest memory address in Program Memory. Input a smaller address.
SET DATA ERR	FALS 00 has been input, and "00" cannot be input. Reinput the data.
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.

## 8-3 Programming Errors

These errors in program syntax will be detected when the program is checked using the Program Check operation.

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.

### Level A Errors

Message	Meaning and appropriate response
?????	The program has been damaged, creating a non-existent function code. Re-enter 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.
OPERAND ERR	A constant entered for the instruction is not within defined values. Change the constant so that it lies within the proper range.
NO END INSTR	There is no END(01) in the program. Write END(01) at the final address in the program.
LOCN ERR	An instruction is in the wrong place in the program. Check instruction requirements and correct the program.
JME UNDEFD	A JME(04) instruction is missing for a JMP(05) instruction. Correct the jump number or insert the proper JME(04) instruction.
DUPL	The same jump number or subroutine number has been used twice. Correct the program so that the same number is only used once for each.
SBN UNDEFD	The SBS(91) instruction has been programmed for a subroutine number that does not exist. Correct the subroutine number or program the required subroutine.
STEP ERR	STEP(08) with a section number and STEP(08) without a section number have been used incorrectly. Check STEP(08) programming requirements and correct the program.

### Level B Errors

Message	Meaning and appropriate response
IL-ILC ERR	IL(02) and ILC(03) are not used in pairs. Correct the program so that each IL(02) has a unique ILC(03). Although this error message will appear if more than one IL(02) is used with the same ILC(03), the program will be executed as written. Make sure your program is written as desired before proceeding.
JMP-JME ERR	JMP(04) and JME(05) are not used in pairs. 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.

## Level C Errors

Message	Meaning and appropriate response
COIL DUPL	The same bit is being controlled (i.e., turned ON and/or OFF) by more than one instruction (e.g., OUT, OUT NOT, DIFU(13), DIFD(14), KEEP(11), SFT(10)). Although this is allowed for certain instructions, check instruction requirements to confirm that the program is correct or rewrite the program so that each bit is controlled by only one instruction.
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.

**Caution**

Expansion instructions (those assigned to function codes 17, 18, 19, 47, 48, 60 to 69, 87, 88, and 89) are not subject to program checks. Program checks also do not cover DM 1024 to DM 6143 for PCs that do not support this part of the DM area (e.g., CQM1-CPU11-E and CQM1-CPU21-E). Data will not be written even if these areas are specified and data read from these areas will always be "0000."

## 8-4 User-defined Errors

There are four instructions that the user can use to define his own errors or messages. These instructions are used to send messages to the Programming Console connected to the PC, cause a non-fatal or a fatal error.

**MESSAGE – MSG(46)**

MSG(46) is used to display a message on the Programming Console. The message, which can be up to 16 characters long, is displayed when the instruction's execution condition is ON. Refer to page 317 for details.

**FAILURE ALARM – FAL(06)**

FAL(06) is an instruction that causes a non-fatal error. Refer to page 205 for details. The following will occur when an FAL(06) instruction is executed:

- 1, 2, 3... 1. The ERR/ALM indicator on the CPU Unit will flash. PC operation will continue.
2. The instruction's 2-digit BCD FAL number (01 to 99) will be written to SR 25300 to SR 25307.
3. The FAL number will be recorded in the PC's error log area. In CQM1 PCs, the time of occurrence will also be recorded if a Memory Cassette with a clock (RTC) is used.

The FAL numbers can be set arbitrarily to indicate particular conditions. The same number cannot be used as both an FAL number and an FALS number.

To clear an FAL error, correct the cause of the error, execute FAL 00, and then clear the error using the Programming Console.

**SEVERE FAILURE ALARM – FALS(07)**

FALS(07) is an instruction that causes a fatal error. Refer to page 205 for details. The following will occur when an FALS(07) instruction is executed:

- 1, 2, 3... 1. Program execution will be stopped and outputs will be turned OFF.
2. The ERR/ALM indicator on the CPU Unit will be lit.
3. The instruction's 2-digit BCD FALS number (01 to 99) will be written to SR 25300 to SR 25307.
4. The FALS number will be recorded in the PC's error log area. In CQM1 PCs, the time of occurrence will also be recorded if a Memory Cassette with a clock (RTC) is used.


The FALS numbers can be set arbitrarily to indicate particular conditions. The same number cannot be used as both an FAL number and an FALS number.

To clear an FALS error, switch the PC to PROGRAM Mode, correct the cause of the error, and then clear the error using the Programming Console.

**FAILURE POINT DETECT – FPD(—)** In CQM1 PCs, non-fatal errors and error messages can also be generated using FPD(—). Refer to page 322 for details.

## 8-5 Operating Errors

There are two kinds of operating errors, non-fatal and fatal. PC operation will continue after a non-fatal error occurs, but operation will be stopped if a fatal error occurs.

 **Caution** Investigate all errors, whether fatal or not. Remove the cause of the error as soon as possible and restart the PC. Refer to the *CQM1 Operation Manual* or *CPM1 Operation Manual* for hardware information and Programming Console operations related to errors. Refer to the *SSS Operation Manual* for SSS operations related to errors.

### 8-5-1 Non-fatal Errors

PC operation and program execution will continue after one or more of these errors have occurred. Although PC operation will continue, the cause of the error should be corrected and the error cleared as soon as possible.

When one of these errors occurs, the POWER and RUN indicators will remain lit and the ERR/ALM indicator will flash.

#### CQM1 Non-fatal Errors

Message	FAL No.	Meaning and appropriate response
SYS FAIL FAL** (see note)	01 to 99	An FAL(06) instruction has been executed in the program. Check the FAL number to determine conditions that would cause execution, correct the cause, and clear the error.
	9D	An error has occurred during data transmission between the CPU Unit and Memory Cassette. Check the status of flags AR 1412 to AR 1415, and correct as directed. AR 1412 ON: Switch to PROGRAM Mode, clear the error, and transfer again. AR 1413 ON: The transfer destination is write-protected.  If the PC is the destination, turn off the power to the PC, be sure that pin 1 of the CPU Unit's DIP switch is OFF, clear the error, and transfer again.  If an EEPROM Memory Cassette is the destination, check whether the power supply is on, clear the error, and transfer again.  If an EPROM Memory Cassette is the destination, change to a writeable Memory Cassette.  AR 1414 ON: The destination has insufficient capacity. Check the source's program size in AR 15 and consider using a different CPU Unit or Memory Cassette.  AR 1415 ON: There is no program in the Memory Cassette or the program contains errors. Check the Memory Cassette.
	9C	An error has occurred in the pulse I/O function or in the absolute-type encoder interface function. Check the contents of AR 0408 to AR 0415 (two digits BCD), and correct as directed. (This error code applies only to CQM1-CPU43-EV1 and CQM1-CPU44-EV1 CPU Units.)  01, 02: An error has occurred in the hardware. Turn the power off, and then power up again. If the error persists, replace the CPU Unit.  03: The PC Setup (DM 6611, DM 6612, DM 6643, DM 6644) settings are incorrect. Correct the settings.  04: CQM1 operation was interrupted during pulse output. Check to see whether the Unit receiving the pulse output was affected.

Message	FAL No.	Meaning and appropriate response
SYS FAIL FAL** (see note)	9B	An error has been detected in the PC Setup. Check flags AR 2400 to AR 2402, and correct as directed.  AR 2400 ON: An incorrect setting was detected in the PC Setup (DM 6600 to DM 6614) when power was turned on. Correct the settings in PROGRAM Mode and turn on the power again.  AR 2401 ON: An incorrect setting was detected in the PC Setup (DM 6615 to DM 6644) when switching to RUN Mode. Correct the settings in PROGRAM Mode and switch to RUN Mode again.  AR 2402 ON: An incorrect setting was detected in the PC Setup (DM 6645 to DM 6655) during operation. Correct the settings and clear the error.
SCAN TIME OVER	F8	Watchdog timer has exceeded 100 ms. (SR 25309 will be ON.)  This indicates that the program cycle time is longer than recommended. Reduce cycle time if possible.
BATT LOW	F7	Backup battery is missing or its voltage has dropped. (SR 25308 will be ON.)  Check the battery and replace if necessary. Check the PC Setup (DM 6655) to see whether a low battery will be detected.

**Note** \*\* is 01 to 99, 9D, 9C, or 9B.

**Communication Errors**

If an error occurs in communications through the peripheral port or RS-232C port the corresponding indicator (COM1 or COM2) will stop flashing. Check the connecting cables as well as the programs in the PC and host computer.

Reset the communications ports with the Port Reset Bits, SR 25208 and SR 25209.

**Output Inhibit**

When the OUT INH indicator is lit, the Output Inhibit Bit (SR 25215) is ON and all outputs from the CPU Unit will be turned off. If it is not necessary to have all outputs off, turn OFF SR 25215.

**CPM1/CPM1A/SRM1 Non-fatal Errors**

Message	FAL No.	Meaning and appropriate response
SYS FAIL FAL** (see note)	01 to 99	An FAL(06) instruction has been executed in the program. Check the FAL number to determine conditions that would cause execution, correct the cause, and clear the error.
	9B	An error has been detected in the PC Setup. Check flags AR 1300 to AR 1302, and correct as directed.  AR 1300 ON: An incorrect setting was detected in the PC Setup (DM 6600 to DM 6614) when power was turned on. Correct the settings in PROGRAM Mode and turn on the power again.  AR 1301 ON: An incorrect setting was detected in the PC Setup (DM 6615 to DM 6644) when switching to RUN Mode. Correct the settings in PROGRAM Mode and switch to RUN Mode again.  AR 1302 ON: An incorrect setting was detected in the PC Setup (DM 6645 to DM 6655) during operation. Correct the settings and clear the error.
SCAN TIME OVER	F8	Watchdog timer has exceeded 100 ms. (SR 25309 will be ON.)  This indicates that the program cycle time is longer than recommended. Reduce cycle time if possible. (The CPM1/CPM1A/SRM1 can be set so that this error won't be detected.)
Communication Errors (no message)	None	If an error occurs in communications through the peripheral port or RS-232C port, the COMM indicator will be off. The error flag in AR0812 will be ON. Check the connecting cables and restart.

**Note** \*\* is 01 to 99 or 9B.

### 8-5-2 Fatal Errors

PC operation and program execution will stop and all outputs from the PC will be turned OFF when any of these errors have occurred.

All CPU Unit indicators will be OFF for the power interruption error. For all other fatal operating errors, the POWER and ERR/ALM indicators will be lit. The RUN indicator will be OFF.

#### CQM1 Fatal Errors

Message	FALS No.	Meaning and appropriate response
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.
MEMORY ERR	F1	AR 1611 ON: A checksum error has occurred in the PC Setup (DM 6600 to DM 6655). Initialize all of the PC Setup and reinput.
		AR 1612 ON: A checksum error has occurred in the program, indicating an incorrect instruction. Check the program and correct any errors detected.
		AR 1613 ON: A checksum error has occurred in an expansion instruction's data. Initialize all of the expansion instruction settings and reinput.
		AR 1614 ON: Memory Cassette was installed or removed with the power on. Turn the power off, install the Memory Cassette, and turn the power on again.
		AR 1615 ON: The Memory Cassette contents could not be read at start-up. Check flags AR 1412 to AR 1415 to determine the problem, correct it, and turn on the power again.
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 ERR	C0	An error has occurred during data transfer between the CPU Unit and an I/O Unit. Determine the location of the problem using flags AR 2408 to AR 2415, turn the power off, check for loose I/O Units or end covers, and turn on the power again.
I/O UNIT OVER	E1	The number of I/O words on the installed I/O Units exceeds the maximum. Turn off the power, rearrange the system to reduce the number of I/O words, and turn on the power again.
SYS FAIL FALS** (see note)	01 to 99	An FALS(07) instruction has been executed in the program. Check the FALS number to determine the conditions that would cause execution, correct the cause, and clear the error.
	9F	The cycle time has exceeded the FALS 9F Cycle Time Monitoring Time (DM 6618). Check the cycle time and adjust the Cycle Time Monitoring Time if necessary.

**Note** \*\* is 01 to 99 or 9F.

#### CPM1/CPM1A/SRM1 Fatal Errors

Message	FALS No.	Meaning and appropriate response
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.
MEMORY ERR	F1	AR 1308 ON: An unspecified bit area exists in the user program. Check the program and correct errors.
		AR 1309 ON: An error has occurred in the flash memory. Since the number of writings to the flash memory has exceeded the specified level, replace the CPU Unit.
		AR 1310 ON: A checksum error has occurred in read-only DM (DM 6144 to DM 6599). Check and correct the settings in the read-only DM area.
		AR 1311 ON: A checksum error has occurred in the PC Setup. Initialize all of the PC Setup and reinput.
		AR 1312 ON: A checksum error has occurred in the program. Check the program and correct any errors detected.
		AR 1314 ON: Power interruption hold area was not held. Clear the error and reset the settings of the power interruption hold area.
		AR 1315 ON: An error has occurred in CompoBus/S communications. If the error cannot be corrected, replace the CPU Unit (SRM! only).
NO END INST	F0	END(01) is not written in the program. Write END(01) at the end of the program.

Message	FALS No.	Meaning and appropriate response
I/O BUS ERR (see note 1)	C0	An error has occurred during data transfer between the CPU Unit and I/O Unit. Check the I/O Unit's connecting cable.
I/O UNIT OVER (see note 1)	E1	Too many I/O Units have been connected. Check the I/O Unit configuration.
SYS FAIL FALS** (see note 2)	01 to 99	A FALS(07) instruction has been executed in the program. Check the FALS number to determine the conditions that caused execution, correct the cause, and clear the error.
	9F	The cycle time has exceeded the FALS 9F Cycle Time Monitoring Time (DM 6618). Check the cycle time and adjust the Cycle Time Monitoring Time if necessary.

- Note**
1. CPM1/CPM1A only.
  2. \*\* is 01 to 99 or 9F.

### 8-5-3 Other Errors

The PWR indicator will be ON for the following errors. Ignore the status of other indicators unless a specific status is given in the following table.

#### CPM1/CPM1A/SRM1 Fatal Errors

Error status	FALS No.	Meaning and appropriate response
CompoBus/S communications error	None	The ERC indicator will light to indicate an error in CompoBus/S communications. Check the slaves and the transmission path and restart the system.
Peripheral/RS-232C port communications error	None	The COMM indicator will light and AR 0812 will turn ON to indicate an error between the peripheral or RS-232C port and the Peripheral Device.

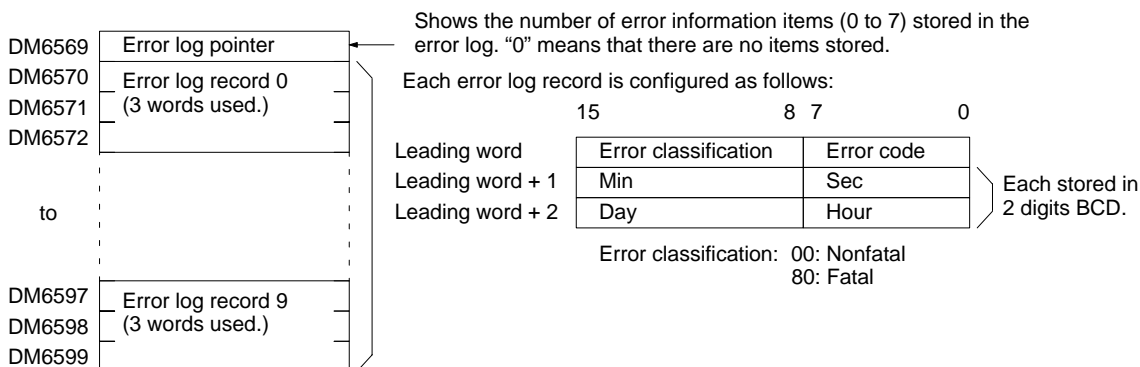
## 8-6 Error Log

The error log function registers the error code of any fatal or non-fatal error that occurs in the PC. The date and time at which the error occurred are registered along with the error code. Refer to page 427 for error codes.

#### CQM1 Error Log Area

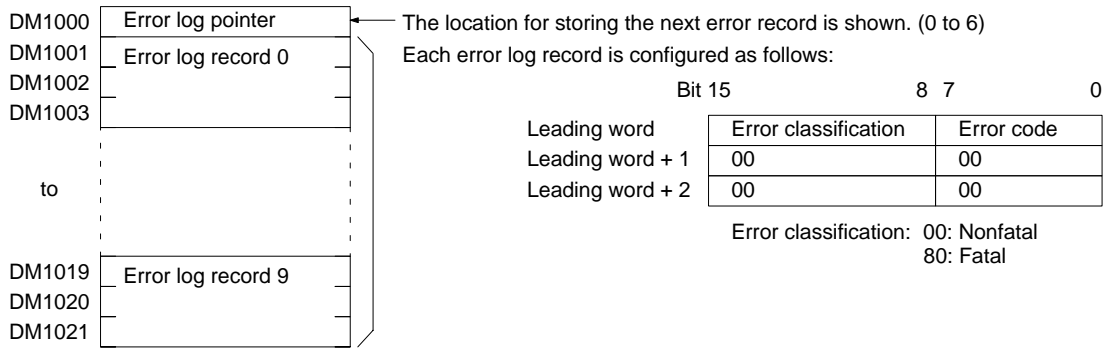
In CQM1 PCs, the error log is stored in DM 6569 through DM 6599.

If a Memory Cassette without a clock is mounted, the date and time will be all zeros.



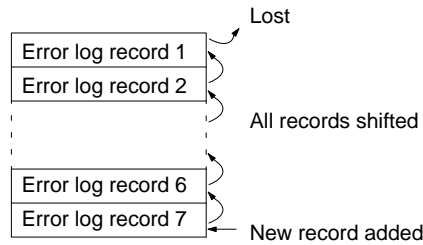
Error records will be stored even if pin 1 on the CQM1 DIP switch is turned ON to protect DM 6144 to DM 6655.

**CPM1/CPM1A Error Log Area** In CPM1/CPM1A PCs, the error log is stored in DM 1000 through DM 1021.



**Error Log Storage Methods** The error log storage method is set in the PC Setup (DM 6655). Set any of the following methods.

- 1, 2, 3...**
1. You can store the most recent 10 error log records and discard older records. This is achieved by shifting the records as shown below so that the oldest record (record 0) is lost whenever a new record is generated.



2. You can store only the first 10 error log records, and ignore any subsequent errors beyond those 10.
3. You can disable the log so that no records are stored.

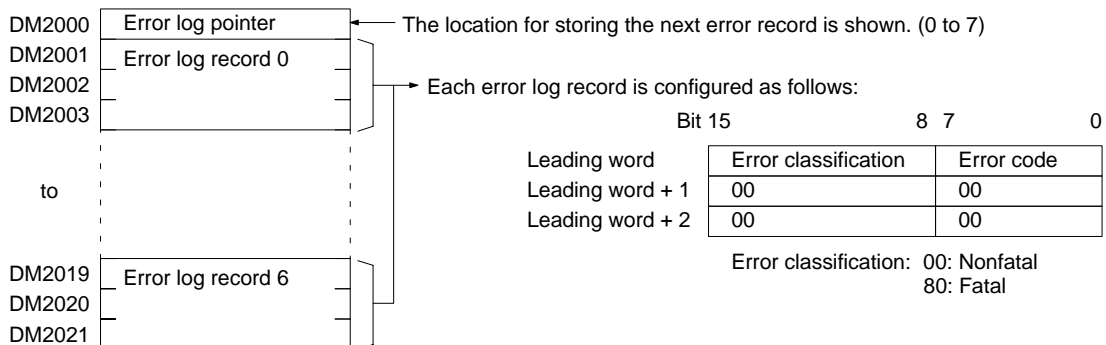
The default setting is the first method. Refer to *Error Log Settings* on page 21 for details on the PC Setup for the error log.

**Clearing the Error Log**

To clear the entire error log, turn ON SR 25214 from a peripheral device. (After the error log has been cleared, SR 25214 will turn OFF again automatically.)

**SRM1 Error Log Area**

In SRM1 PCs, the error log is stored in DM 2000 through DM 2021.





**Error Log Storage Methods** The error log storage method is set in the PC Setup (DM 6655). Set any of the following methods.

- 1, 2, 3...**
1. You can store the most recent 7 error log records and discard older records. This is achieved by shifting the records as shown below so that the oldest record (record 0) is lost whenever a new record is generated.



2. You can store only the first 7 error log records, and ignore any subsequent errors beyond those 7.
3. You can disable the log so that no records are stored.

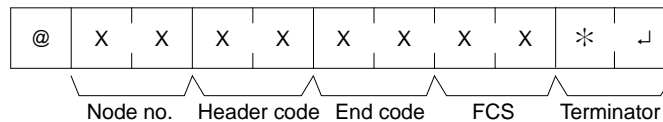
The default setting is the first method. Refer to *Error Log Settings* on page 21 for details on the PC Setup for the error log.

**Clearing the Error Log**

To clear the entire error log, turn ON SR 25214 from a peripheral device. (After the error log has been cleared, SR 25214 will turn OFF again automatically.)

## 8-7 Host Link Errors

These error codes are received as the response code (end code) when a command received by the PC from a host computer cannot be processed. The error code format is as shown below.



The header code will vary according to the command. Some commands (composite commands) contain a subcode.

End code	Contents	Probable cause	Corrective measures
00	Normal completion	---	---
01	Not executable in RUN mode	The command that was sent cannot be executed when the PC is in RUN mode.	Check the relation between the command and the PC mode.
02	Not executable in MONITOR mode	The command that was sent cannot be executed when the PC is in MONITOR mode.	
04	Address over (CPM1/CPM1A/SRM1 PCs)	The user program area's highest address was exceeded.	Check the program.
0B	Not executable in PROGRAM mode	The command that was sent cannot be executed when the PC is in PROGRAM mode.	This code is not presently being used.
13	FCS error	The FCS is wrong. Either the FCS calculation is mistaken or there is adverse influence from noise.	Check the FCS calculation method. If there was influence from noise, transfer the command again.
14	Format error	The command format is wrong.	Check the format and transfer the command again.
15	Entry number data error	The areas for reading and writing are wrong.	Correct the areas and transfer the command again.
16	Command not supported	The specified command does not exist in the specified address. (Reading the SV, etc.)	Check the address and instruction.
18	Frame length error	The maximum frame length was exceeded.	Divide the command into multiple frames.
19	Not executable	Items to read not registered for composite command (QQ).	Execute QQ to register items to read before attempting batch read.
23	User memory write-protected	CQM1 PCs: Pin 1 on the CQM1 DIP switch is ON. CPM1/CPM1A/SRM1 PCs: The memory is write-protected in the PC Setup.	CQM1 PCs: Turn OFF pin 1. CPM1/CPM1A/SRM1 PCs: Change the setting in the PC Setup (DM 6602).
A3	Aborted due to FCS error in transmit data	The error was generated while a command extending over more than one frame was being executed. <b>Note:</b> The data up to that point has already been written to the appropriate area of the CPU Unit.	Check for corrupted frames, correct if necessary, and try the transfer again.
A4	Aborted due to format error in transmit data		
A5	Aborted due to entry number data error in transmit data		
A8	Aborted due to frame length error in transmit data		
Other	---	Influence from noise was received.	Transfer the command again.

### Power Interruptions

The following responses may be received from the CQM1 if a power interruption occurs, including momentary interruptions. If any of these responses are received during or after a power interruption, repeat the command.

#### Undefined Command Response

@00IC4A\*.,┘

#### No Response

If no response is received, abort the last command and resend.

## 8-8 Troubleshooting Flowcharts

### 8-8-1 CPM1 Flowcharts

Refer to 5-6 Troubleshooting Flowcharts in the CPM1 Operation Manual.

### 8-8-2 CPM1A Flowcharts

Refer to 5-6 Troubleshooting Flowcharts in the CPM1A Operation Manual.

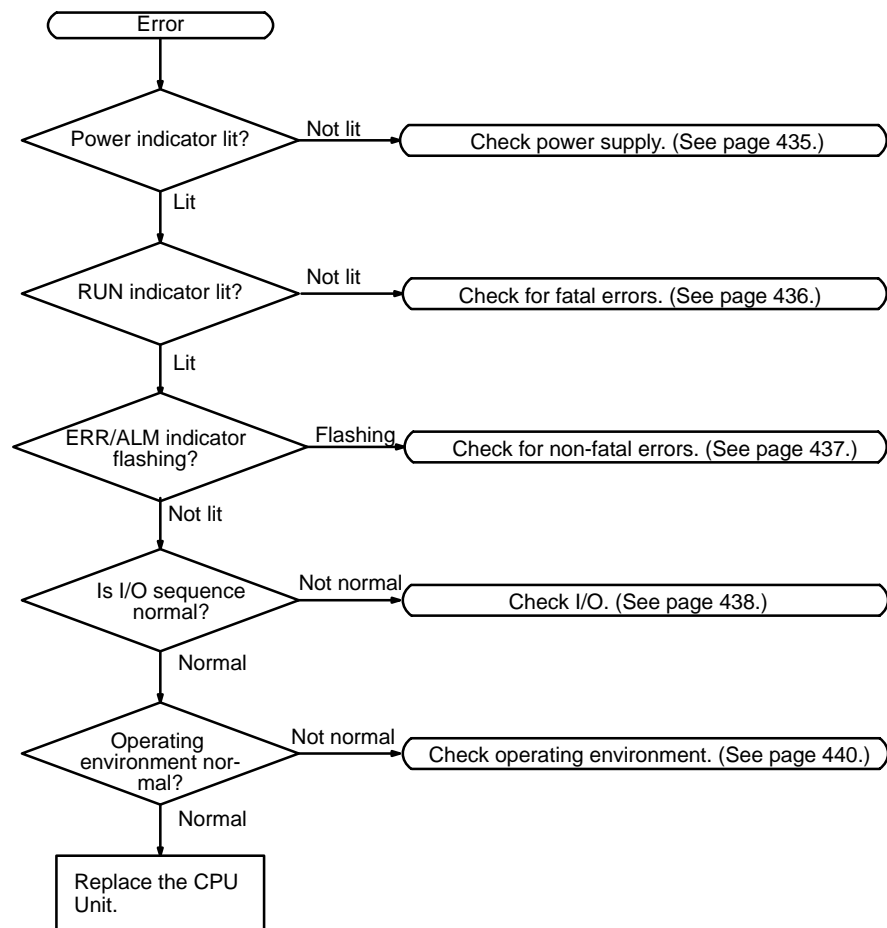
### 8-8-3 SRM1 Flowcharts

Refer to 5-6 Troubleshooting Flowcharts in the SRM1 Operation Manual.

### 8-8-4 CQM1 Flowcharts

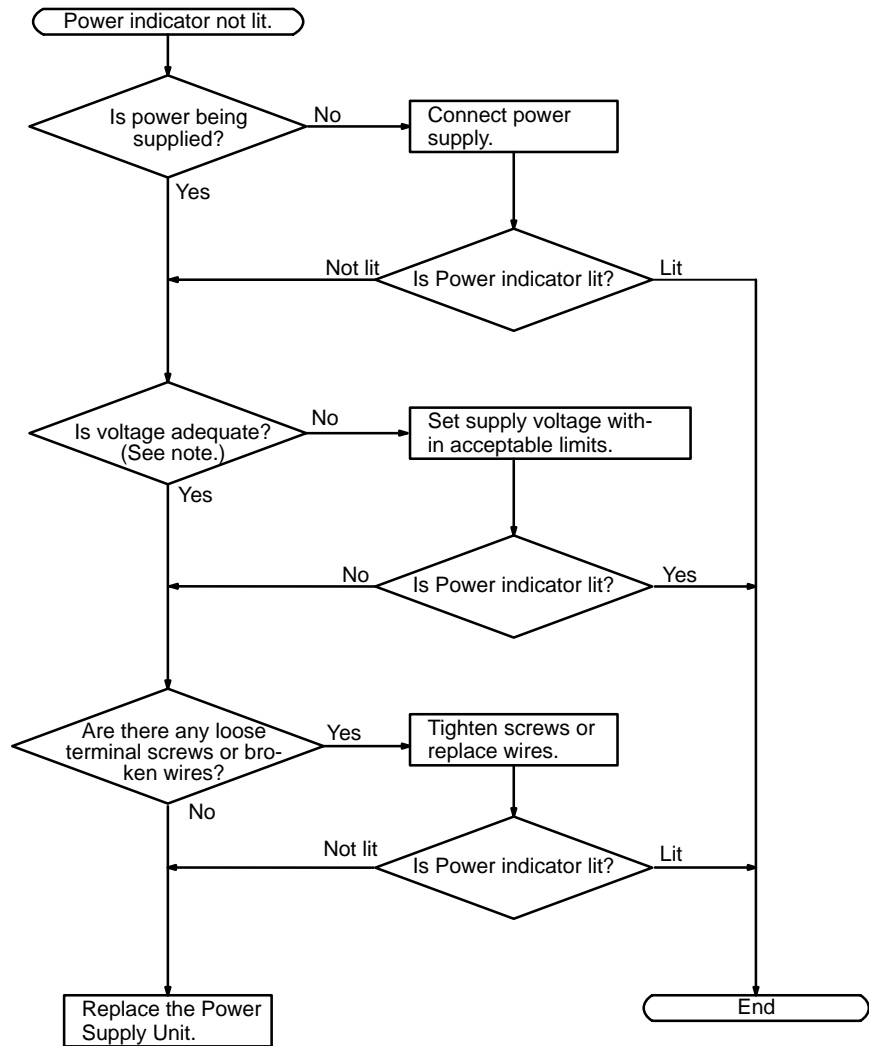
Use the following flowcharts to troubleshoot errors that occur during CQM1 operation.

#### Main Check



**Note** Always turn off the power to the PC before replacing Units, batteries, wiring, or cables.

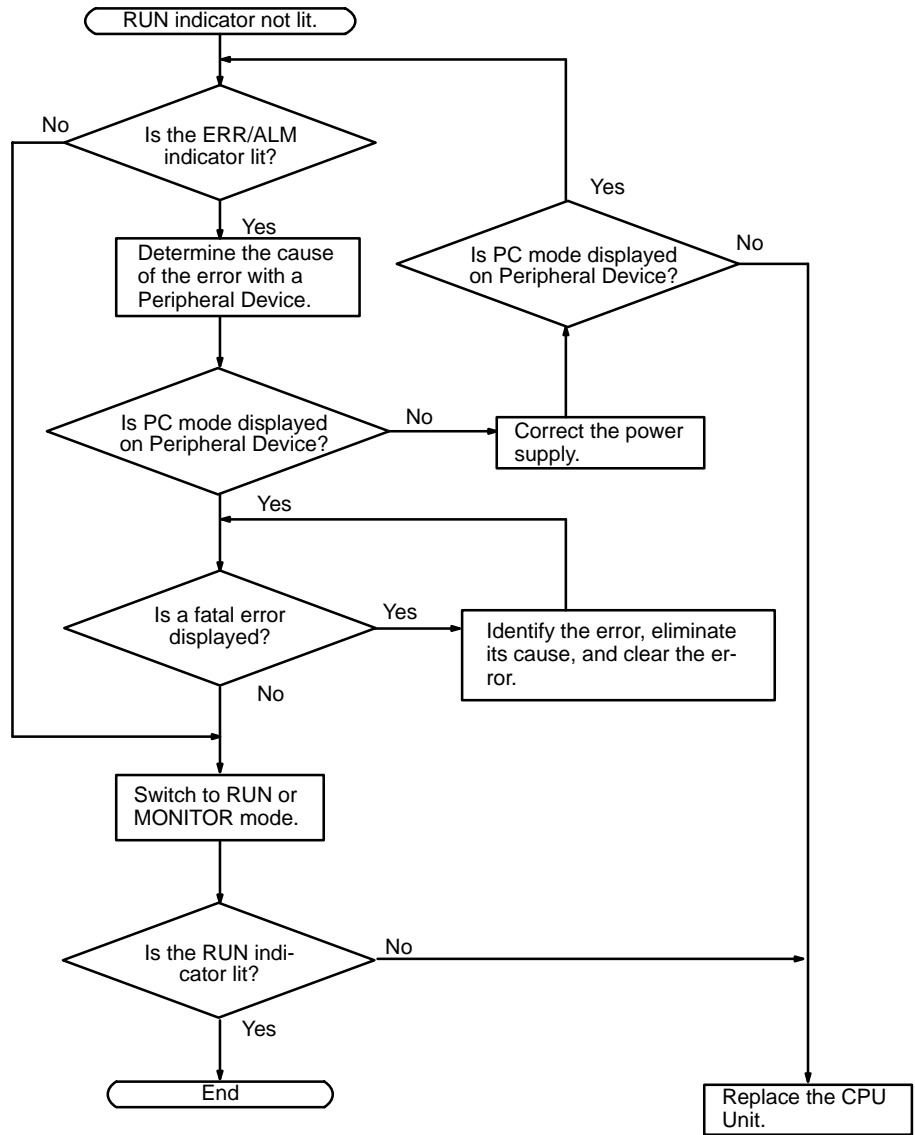
Power Supply Check



**Note** The allowable voltage ranges for the CQM1 are as shown below.  
 CQM1-PA203/PA206: 85 to 264 VAC  
 CQM1-PD026: 20 to 28 VDC

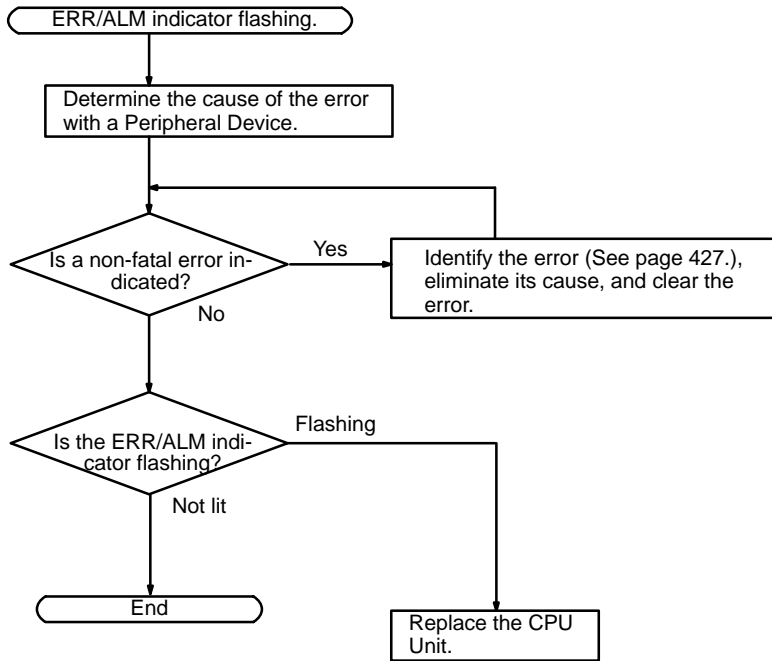
Fatal Error Check

The following flowchart can be used to troubleshoot fatal errors that occur while the Power indicator is lit.



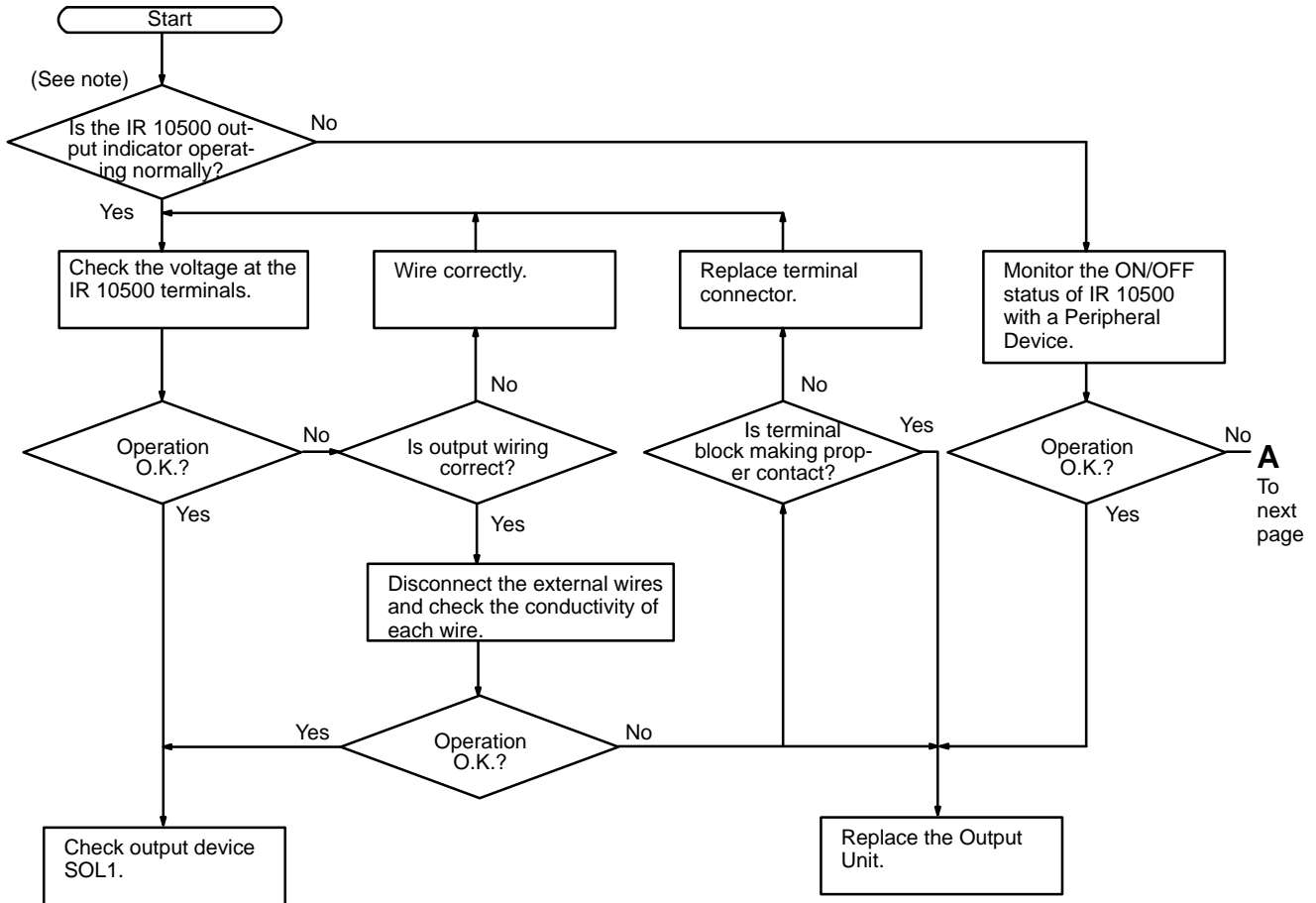
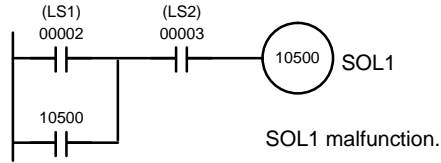
**Non-fatal Error Check**

Although the PC will continue operating during non-fatal errors, the cause of the error should be determined and removed as quickly as possible to ensure proper operation. It may be necessary to stop PC operation to remove certain non-fatal errors.

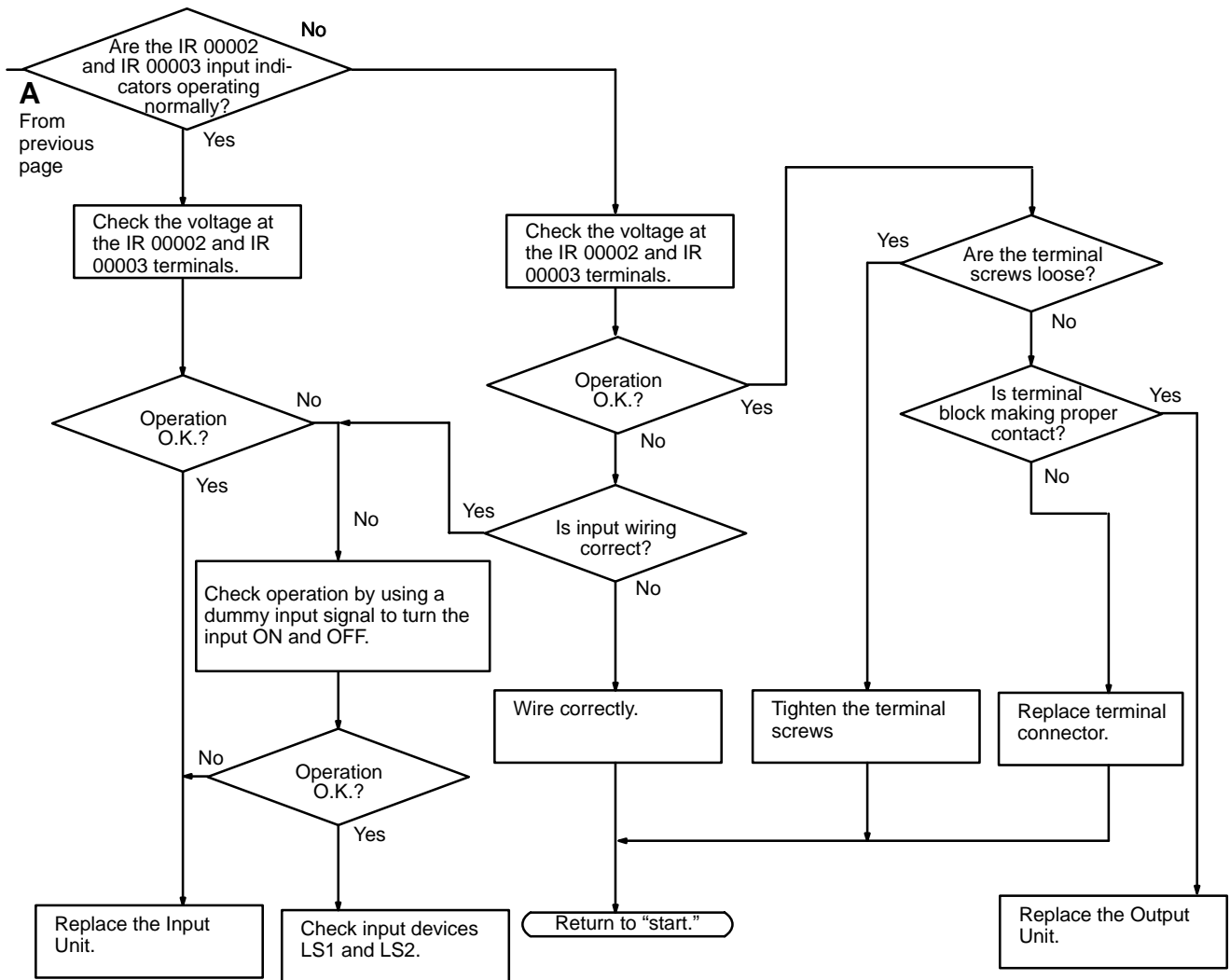


I/O Check

The I/O check flowchart is based on the following ladder diagram section.

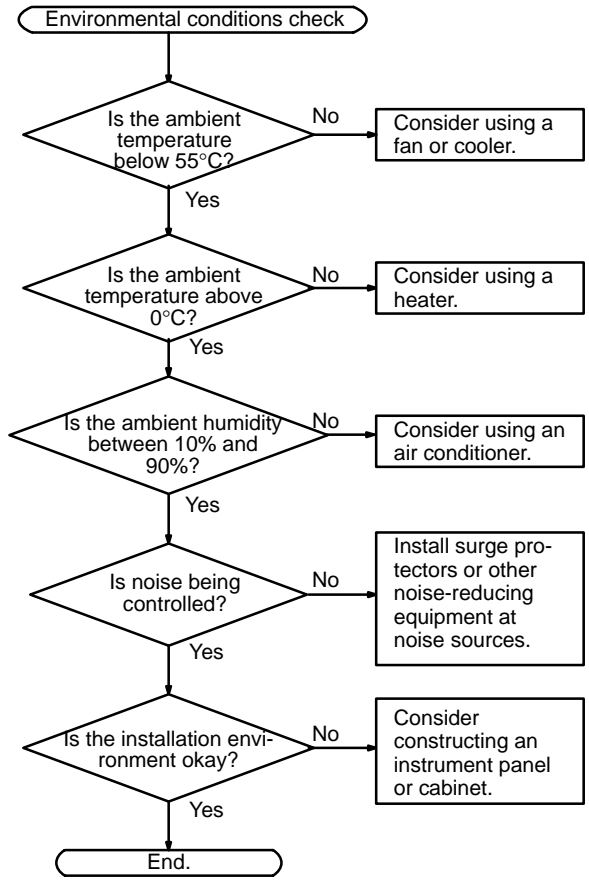


**Note** The CPM1 PC doesn't have the IR 10500 output indicator. Substitute with one from IR 01000 to IR 01915.





Environmental Conditions Check



# Appendix A

## Programming Instructions

A PC instruction is input either by pressing the corresponding Programming Console key(s) (e.g., LD, AND, OR, NOT) or by using function codes. To input an instruction with its function code, press FUN, the function code, and then WRITE. Refer to the pages listed programming and instruction details.

Code	Mnemonic	Name	Function	Page
—	AND	AND	Logically ANDs status of designated bit with execution condition.	196
—	AND LD	AND LOAD	Logically ANDs results of preceding blocks.	197
—	AND NOT	AND NOT	Logically ANDs inverse of designated bit with execution condition.	196
—	CNT	COUNTER	A decrementing counter.	210
—	LD	LOAD	Used to start instruction line with the status of the designated bit or to define a logic block for use with AND LD and OR LD.	196
—	LD NOT	LOAD NOT	Used to start instruction line with inverse of designated bit.	196
—	OR	OR	Logically ORs status of designated bit with execution condition.	196
—	OR LD	OR LOAD	Logically ORs results of preceding blocks.	197
—	OR NOT	OR NOT	Logically ORs inverse of designated bit with execution condition.	196
—	OUT	OUTPUT	Turns ON operand bit for ON execution condition; turns OFF operand bit for OFF execution condition.	197
—	OUT NOT	OUTPUT NOT	Turns operand bit OFF for ON execution condition; turns operand bit ON for OFF execution condition (i.e., inverts operation).	197
—	RSET	RESET	Turns the operand bit OFF when the execution condition is ON, and does not affect the status of the operand bit when the execution condition is OFF.	198
—	SET	SET	Turns the operand bit ON when the execution condition is ON, and does not affect the status of the operand bit when the execution condition is OFF.	198
—	TIM	TIMER	ON-delay (decrementing) timer operation.	209
00	NOP	NO OPERATION	Nothing is executed and program moves to next instruction.	201
01	END	END	Required at the end of the program.	201
02	IL	INTERLOCK	If interlock condition is OFF, all outputs are turned OFF and all timer PVs reset between this IL(02) and the next ILC(03). Other instructions are treated as NOP; counter PVs are maintained.	201
03	ILC	INTERLOCK CLEAR		201
04	JMP	JUMP	If jump condition is OFF, all instructions between JMP(04) and the corresponding JME(05) are ignored.	203
05	JME	JUMP END		203
06	(@)FAL	FAILURE ALARM AND RESET	Generates a non-fatal error and outputs the designated FAL number to the Programming Console.	205
07	FALS	SEVERE FAILURE ALARM	Generates a fatal error and outputs the designated FALS number to the Programming Console.	205
08	STEP	STEP DEFINE	When used with a control bit, defines the start of a new step and resets the previous step. When used without N, defines the end of step execution.	206
09	SNXT	STEP START	Used with a control bit to indicate the end of the step, reset the step, and start the next step.	206
10	SFT	SHIFT REGISTER	Creates a bit shift register.	224
11	KEEP	KEEP	Defines a bit as a latch controlled by set and reset inputs.	199
12	CNTR	REVERSIBLE COUNTER	Increases or decreases PV by one whenever the increment input or decrement input signals, respectively, go from OFF to ON.	211
13	DIFU	DIFFERENTIATE UP	Turns ON the designated bit for one cycle on the rising edge of the input signal.	200

Code	Mnemonic	Name	Function	Page
14	DIFD	DIFFERENTIATE DOWN	Turns ON the bit for one cycle on the trailing edge.	200
15	TIMH	HIGH-SPEED TIMER	A high-speed, ON-delay (decrementing) timer.	212
16	(@)WSFT	WORD SHIFT	Shifts data between starting and ending words in word units, writing zeros into starting word.	225
17 to 19	For expansion instructions.			116
20	CMP	COMPARE	Compares the contents of two words and outputs result to GR, EQ, and LE Flags.	242
21	(@)MOV	MOVE	Copies source data (word or constant) to destination word.	232
22	(@)MVN	MOVE NOT	Inverts source data (word or constant) and then copies it to destination word.	232
23	(@)BIN	BCD TO BINARY	Converts four-digit, BCD data in source word into 16-bit binary data, and outputs converted data to result word.	252
24	(@)BCD	BINARY TO BCD	Converts binary data in source word into BCD, and outputs converted data to result word.	253
25	(@)ASL	ARITHMETIC SHIFT LEFT	Shifts each bit in single word of data one bit to left, with CY.	225
26	(@)ASR	ARITHMETIC SHIFT RIGHT	Shifts each bit in single word of data one bit to right, with CY.	226
27	(@)ROL	ROTATE LEFT	Rotates bits in single word of data one bit to left, with CY.	226
28	(@)ROR	ROTATE RIGHT	Rotates bits in single word of data one bit to right, with CY.	227
29	(@)COM	COMPLEMENT	Inverts bit status of one word of data.	308
30	(@)ADD	BCD ADD	Adds two four-digit BCD values and content of CY, and outputs result to specified result word.	278
31	(@)SUB	BCD SUBTRACT	Subtracts a four-digit BCD value and CY from another four-digit BCD value and outputs result to the result word.	279
32	(@)MUL	BCD MULTIPLY	Multiplies two four-digit BCD values and outputs result to specified result words.	281
33	(@)DIV	BCD DIVIDE	Divides four-digit BCD dividend by four-digit BCD divisor and outputs result to specified result words.	282
34	(@)ANDW	LOGICAL AND	Logically ANDs two 16-bit input words and sets corresponding bit in result word if corresponding bits in input words are both ON.	309
35	(@)ORW	LOGICAL OR	Logically ORs two 16-bit input words and sets corresponding bit in result word if one or both of corresponding bits in input data are ON.	309
36	(@)XORW	EXCLUSIVE OR	Exclusively ORs two 16-bit input words and sets bit in result word when corresponding bits in input words differ in status.	310
37	(@)XNRW	EXCLUSIVE NOR	Exclusively NORs two 16-bit input words and sets bit in result word when corresponding bits in input words are same in status.	311
38	(@)INC	BCD INCREMENT	Increments four-digit BCD word by one.	311
39	(@)DEC	BCD DECREMENT	Decrements four-digit BCD word by one.	312
40	(@)STC	SET CARRY	Sets carry flag (i.e., turns CY ON).	278
41	(@)CLC	CLEAR CARRY	Clears carry flag (i.e., turns CY OFF).	278
45	TRSM	TRACE MEMORY SAMPLE	Initiates data tracing. Cannot be used with CQM1-CPU11/21-E/CPM1/CPM1A /SRM1.	315
46	(@)MSG	MESSAGE	Displays a 16-character message on the Programming Console display.	317
47 & 48	For expansion instructions.			116
50	(@)ADB	BINARY ADD	Adds two four-digit hexadecimal values and content of CY, and outputs result to specified result word.	289
51	(@)SBB	BINARY SUBTRACT	Subtracts a four-digit hexadecimal value and CY from another four-digit hexadecimal value and outputs result to the result word.	290
52	(@)MLB	BINARY MULTIPLY	Multiplies two four-digit hexadecimal values and outputs result to specified result words.	291

Code	Mnemonic	Name	Function	Page
53	(@)DVB	BINARY DIVIDE	Divides four-digit hexadecimal dividend by four-digit hexadecimal divisor and outputs result to specified result words.	292
54	(@)ADDL	DOUBLE BCD ADD	Adds two eight-digit values (2 words each) and content of CY, and outputs result to specified result words.	284
55	(@)SUBL	DOUBLE BCD SUBTRACT	Subtracts an eight-digit BCD value and CY from another eight-digit BCD value and outputs result to the result words.	285
56	(@)MULL	DOUBLE BCD MULTIPLY	Multiplies two eight-digit BCD values and outputs result to specified result words.	286
57	(@)DIVL	DOUBLE BCD DIVIDE	Divides eight-digit BCD dividend by eight-digit BCD divisor and outputs result to specified result words.	287
58	(@)BINL	DOUBLE BCD TO DOUBLE BINARY	Converts BCD value in two consecutive source words into binary and outputs converted data to two consecutive result words. (CQM1 only)	254
59	(@)BCDL	DOUBLE BINARY TO DOUBLE BCD	Converts binary value in two consecutive source words into BCD and outputs converted data to two consecutive result words. (CQM1 only)	254
60 to 69	For expansion instructions.			116
70	(@)XFER	BLOCK TRANSFER	Moves content of several consecutive source words to consecutive destination words.	233
71	(@)BSET	BLOCK SET	Copies content of one word or constant to several consecutive words.	234
72	(@)ROOT	SQUARE ROOT	Computes square root of eight-digit BCD value and outputs truncated four-digit integer result to specified result word. (CQM1 only)	288
73	(@)XCHG	DATA EXCHANGE	Exchanges contents of two different words.	235
74	(@)SLD	ONE DIGIT SHIFT LEFT	Left shifts data between starting and ending words by one digit (four bits).	228
75	(@)SRD	ONE DIGIT SHIFT RIGHT	Right shifts data between starting and ending words by one digit (four bits).	228
76	(@)MLPX	4-TO-16 DECODER	Converts up to four hexadecimal digits in source word into decimal values from 0 to 15 and turns ON, in result word(s), bit(s) whose position corresponds to converted value.	255
77	(@)DMPX	16-TO-4 ENCODER	Determines position of highest ON bit in source word(s) and turns ON corresponding bit(s) in result word.	257
78	(@)SDEC	7-SEGMENT DECODER	Converts hexadecimal values from source word to data for seven-segment display.	259
80	(@)DIST	SINGLE WORD DISTRIBUTE	Moves one word of source data to destination word whose address is given by destination base word plus offset.	235
81	(@)COLL	DATA COLLECT	Extracts data from source word and writes it to destination word.	237
82	(@)MOVB	MOVE BIT	Transfers designated bit of source word or constant to designated bit of destination word.	239
83	(@)MOVD	MOVE DIGIT	Moves hexadecimal content of specified four-bit source digit(s) to specified destination digit(s) for up to four digits.	240
84	(@)SFTR	REVERSIBLE SHIFT REGISTER	Shifts data in specified word or series of words to either left or right.	229
85	(@)TCMP	TABLE COMPARE	Compares four-digit hexadecimal value with values in table consisting of 16 words.	243
86	(@)ASC	ASCII CONVERT	Converts hexadecimal values from the source word to eight-bit ASCII code starting at leftmost or rightmost half of starting destination word.	262
87 to 89	For expansion instructions.			116
91	(@)SBS	SUBROUTINE ENTRY	Calls and executes subroutine N.	313
92	SBN	SUBROUTINE DEFINE	Marks start of subroutine N.	315
93	RET	RETURN	Marks the end of a subroutine and returns control to main program.	315

Code	Mnemonic	Name	Function	Page
97	(@)IORF	I/O REFRESH	Refreshes all I/O words between the start and end words. Cannot be used with the SRM1.	318
99	(@)MCRO	MACRO	Calls and executes a subroutine replacing I/O words.	319

## Expansion Instructions

The following table shows the instructions that can be treated as expansion instructions. The default function codes are given for instructions that have codes assigned by default.

Code	Mnemonic	Name	Function	CPU Units	Page
17	(@)ASFT	ASYNCHRONOUS SHIFT REGISTER	Creates a shift register that exchanges the contents of adjacent words when one of the words is zero and the other is not.	All	230
18	TKY	TEN KEY INPUT	Inputs 8 digits of BCD data from a 10-key keypad.	CQM1	347
19	(@)MCMP	MULTI-WORD COMPARE	Compares a block of 16 consecutive words to another block of 16 consecutive words.	CQM1	247
47	(@)RXD	RECEIVE	Receives data via a communications port.	CQM1/ SRM1	340
48	(@)TXD	TRANSMIT	Sends data via a communications port.	CQM1/ SRM1	342
60	CMPL	DOUBLE COMPARE	Compares two eight-digit hexadecimal values.	All	246
61	(@)INI	MODE CONTROL	Starts and stops counter operation, compares and changes counter PVs, and stops pulse output.	All except SRM1	220
62	(@)PRV	HIGH-SPEED COUNTER PV READ	Reads counter PVs and status data for the high-speed counter.	All except SRM1	221
63	(@)CTBL	COMPARISON TABLE LOAD	Compares counter PVs and generates a direct table or starts operation.	All except SRM1	215
64	(@)SPED	SPEED OUTPUT	Outputs pulses at the specified frequency (10 Hz to 50 KHz in 10 Hz units). The output frequency can be changed while pulses are being output.	CQM1	330
65	(@)PULS	SET PULSES	Outputs the specified number of pulses at the specified frequency. The pulse output cannot be stopped until the specified number of pulses have been output.	CQM1	328
66	(@)SCL	SCALE	Performs a scaling conversion on the calculated value.	CQM1	266
67	(@)BCNT	BIT COUNTER	Counts the total number of bits that are ON in the specified block of words.	All	320
68	(@)BCMP	BLOCK COMPARE	Judges whether the value of a word is within 16 ranges (defined by lower and upper limits).	All	244
69	(@)STIM	INTERVAL TIMER	Controls interval timers used to perform scheduled interrupts.	All	213
87	DSW	DIGITAL SWITCH INPUT	Inputs 4- or 8-digit BCD data from a digital switch.	CQM1	346
88	7SEG	7-SEGMENT DISPLAY OUTPUT	Converts 4- or 8-digit data to 7-segment display format and then outputs the converted data.	CQM1	345
89	(@)INT	INTERRUPT CONTROL	Performs interrupt control, such as masking and unmasking the interrupt bits for I/O interrupts.	All except SRM1	326
---	(@)ACC	ACCELERATION CONTROL	Together with PULS(—), ACC(—) controls the acceleration and/or deceleration of pulses output from port 1 or 2.	CQM1- CPU43-E V1	334
---	(@)ADBL	DOUBLE BINARY ADD	Adds two 8-digit binary values (normal or signed data) and outputs the result to R and R+1.	CQM1	293
---	(@)APR	ARITHMETIC PROCESS	Performs sine, cosine, or linear approximation calculations.	CQM1	305

Code	Mnemonic	Name	Function	CPU Units	Page
---	AVG	AVERAGE VALUE	Adds the specified number of hexadecimal words and computes the mean value. Rounds off to 4 digits past the decimal point.	CQM1	302
---	(@)COLM	LINE TO COLUMN	Copies the 16 bits from the specified word to a bit column of 16 consecutive words.	CQM1	274
---	CPS	SIGNED BINARY COMPARE	Compares two 16-bit (4-digit) signed binary values and outputs the result to the GR, EQ, and LE flags.	CQM1	248
---	CPSL	DOUBLE SIGNED BINARY COMPARE	Compares two 32-bit (8-digit) signed binary values and outputs the result to the GR, EQ, and LE flags.	CQM1	249
---	(@)DBS	SIGNED BINARY DIVIDE	Divides one 16-bit signed binary value by another and outputs the 32-bit signed binary result to R+1 and R.	CQM1	298
---	(@)DBSL	DOUBLE SIGNED BINARY DIVIDE	Divides one 32-bit signed binary value by another and outputs the 64-bit signed binary result to R+3 to R.	CQM1	299
---	(@)FCS	FCS CALCULATE	Checks for errors in data transmitted by a Host Link command.	CQM1/ SRM1	321
---	FPD	FAILURE POINT DETECT	Finds errors within an instruction block.	CQM1	322
---	(@)HEX	ASCII-TO-HEXADECIMAL	Converts ASCII data to hexadecimal data.	CQM1/ SRM1	263
---	HKY	HEXADECIMAL KEY INPUT	Inputs up to 8 digits of hexadecimal data from a 16-key keypad.	CQM1	347
---	(@)HMS	SECONDS TO HOURS	Converts second data to hour and minute data.	CQM1	272
---	(@)LINE	LINE	Copies a bit column from 16 consecutive words to the specified word.	CQM1	273
---	(@)MAX	FIND MAXIMUM	Finds the maximum value in specified data area and outputs that value to another word.	CQM1	300
---	(@)MBS	SIGNED BINARY MULTIPLY	Multiplies the signed binary content of two words and outputs the 8-digit signed binary result to R+1 and R.	CQM1	296
---	(@)MBSL	DOUBLE SIGNED BINARY MULTIPLY	Multiplies two 32-bit (8-digit) signed binary values and outputs the 16-digit signed binary result to R+3 through R.	CQM1	297
---	(@)MIN	FIND MINIMUM	Finds the minimum value in specified data area and outputs that value to another word.	CQM1	301
---	(@)NEG	2'S COMPLEMENT	Converts the four-digit hexadecimal content of the source word to its 2's complement and outputs the result to R.	CQM1	275
---	(@)NEGL	DOUBLE 2'S COMPLEMENT	Converts the eight-digit hexadecimal content of the source words to its 2's complement and outputs the result to R and R+1.	CQM1	276
---	PID	PID CONTROL	Performs PID control based on the specified parameters.	CQM1- CPU43-E V1	338
---	(@)PLS2	PULSE OUTPUT	Accelerates pulse output from 0 to the target frequency at a specified rate and decelerates at the same rate.	CQM1- CPU4□- EV1	332
---	(@)PWM	PULSE WITH VARIABLE DUTY RATIO	Outputs pulses with the specified duty ratio (0% to 99%) from port 1 or 2.	CQM1- CPU43-E V1	336
---	(@)SBBL	DOUBLE BINARY SUBTRACT	Subtracts an 8-digit binary value (normal or signed data) from another and outputs the result to R and R+1.	CQM1	294

Code	Mnemonic	Name	Function	CPU Units	Page
---	(@)SCL2	SIGNED BINARY TO BCD SCALING	Linearly converts a 4-digit signed hexadecimal value to a 4-digit BCD value.	CQM1 CPU4□- EV1	268
---	(@)SCL3	BCD TO SIGNED BINARY SCALING	Linearly converts a 4-digit BCD value to a 4-digit signed hexadecimal value.	CQM1 CPU4□- EV1	269
---	(@)SEC	HOURS TO SECONDS	Converts hour and minute data to second data.	CQM1	271
---	(@)SRCH	DATA SEARCH	Searches the specified range of memory for the specified data. Outputs the word address(es) of words in the range that contain the data.	CQM1	337
---	(@)STUP	CHANGE RS-232C SETUP	Changes the communications parameters in the PC Setup for a specified port.	SRM1	344
---	(@)SUM	SUM CALCULATE	Computes the sum of the contents of the words in the specified range of memory.	CQM1	303
---	(@)XFRB	TRANSFER BITS	Copies the status of up to 255 specified source bits to the specified destination bits.	CQM1	241
---	ZCP	AREA RANGE COMPARE	Compares a word to a range defined by lower and upper limits and outputs the result to the GR, EQ, and LE flags.	CQM1	250
---	ZCPL	DOUBLE AREA RANGE COMPARE	Compares an 8-digit value to a range defined by lower and upper limits and outputs the result to the GR, EQ, and LE flags.	CQM1	252

## Appendix B

### Error and Arithmetic Flag Operation

The following table shows the instructions that affect the ER, CY, GT, LT and EQ flags. In general, ER indicates that operand data is not within requirements. CY indicates arithmetic or data shift results. GT indicates that a compared value is larger than some standard, LT that it is smaller, and EQ, that it is the same. EQ also indicates a result of zero for arithmetic operations. Refer to *Section 5 Instruction Set* for details.

Vertical arrows in the table indicate the flags that are turned ON and OFF according to the result of the instruction. Although ladder diagram instructions, TIM, and CNT are executed when ER is ON, other instructions with a vertical arrow under the ER column are not executed if ER is ON. All of the other flags in the following table will also not operate when ER is ON.

Instructions not shown do not affect any of the flags in the table. Although only the non-differentiated form of each instruction is shown, differentiated instructions affect flags in exactly the same way.

The ER, CY, GT, LT and EQ Flags are turned OFF when END(01) is executed, so their status cannot be monitored with a Programming Device.

The status of the ER, CY, GT, LT and EQ Flags is affected by instruction execution and will change each time an instruction that affects them is executed. Differentiated instructions are executed only once when their execution condition changes (ON to OFF or OFF to ON) and are not executed again until the next specified change in their execution condition. The status of the ER, CY, GT, LT and EQ Flags is thus affected by a differentiated instruction only when the execution condition changes and is not affected during scans when the instruction is not executed, i.e., when the specified change does not occur in the execution condition. When a differentiated instruction is not executed, the status of the ER, CY, GT, LT and EQ Flags will not change and will maintain the status produced by the last instruction that was executed.

Instructions	25503 (ER)	25504 (CY)	25505 (GR)	25506 (EQ)	25507 (LE)	25402 (N)	Page
TIM	↓	Unaffected	Unaffected	Unaffected	Unaffected	Unaffected	209
CNT							210
END(01)	OFF	OFF	OFF	OFF	OFF	OFF	201
STEP(08)	Unaffected	Unaffected	Unaffected	Unaffected	Unaffected	Unaffected	206
SNXT(09)							206
CNTR(12)	↓	Unaffected	Unaffected	Unaffected	Unaffected	Unaffected	211
TIMH(15)							212
WSFT(16)							225
CMP(20)	↓	Unaffected	↓	↓	↓	Unaffected	242
MOV(21)	↓	Unaffected	Unaffected	↓	Unaffected	↓	232
MVN(22)						↓	232
BIN(23)						OFF	252
BCD(24)						Unaffected	253
ASL(25)	↓	↓	Unaffected	↓	Unaffected	↓	225
ASR(26)						OFF	226
ROL(27)						↓	226
ROR(28)						↓	227
COM(29)	↓	Unaffected	Unaffected	↓	Unaffected	↓	308
ADD(30)	↓	↓	Unaffected	↓	Unaffected	Unaffected	278
SUB(31)							279



Instructions	25503 (ER)	25504 (CY)	25505 (GR)	25506 (EQ)	25507 (LE)	25402 (N)	Page
MUL(32)	↕	Unaffected	Unaffected	↕	Unaffected	Unaffected	281
DIV(33)							282
ANDW(34)						↕	309
ORW(35)						↕	309
XORW(36)						↕	310
XNRW(37)						↕	311
INC(38)						Unaffected	311
DEC(39)							312
STC(40)	Unaffected	ON	Unaffected	Unaffected	Unaffected	Unaffected	278
CLC(41)	Unaffected	OFF	Unaffected	Unaffected	Unaffected	Unaffected	278
MSG(46)	↕	Unaffected	Unaffected	Unaffected	Unaffected	Unaffected	317
ADB(50)	↕	↕	Unaffected	↕	Unaffected	↕	289
SBB(51)						↕	290
MLB(52)	Unaffected	↕	Unaffected	Unaffected	↕	↕	291
DVB(53)	↕	Unaffected	Unaffected	↕	Unaffected	↕	292
ADDL(54)	↕	↕	Unaffected	↕	Unaffected	Unaffected	284
SUBL(55)							285
MULL(56)	↕	Unaffected	Unaffected	↕	Unaffected	Unaffected	286
DIVL(57)							287
BINL(58)						OFF	254
BCDL(59)						Unaffected	254
XFER(70)	↕	Unaffected	Unaffected	Unaffected	Unaffected	Unaffected	233
BSET(71)							234
ROOT(72)	↕	Unaffected	Unaffected	↕	Unaffected	Unaffected	288
XCHG(73)	↕	Unaffected	Unaffected	Unaffected	Unaffected	Unaffected	235
SLD(74)							228
SRD(75)							228
MLPX(76)							255
DMPX(77)							257
SDEC(78)							259
DIST(80)	↕	Unaffected	Unaffected	↕	Unaffected	↕	235
COLL(81)						↕	237
MOVB(82)	↕	Unaffected	Unaffected	Unaffected	Unaffected	Unaffected	239
MOVD(83)							240
SFTR(84)	↕	↕	Unaffected	Unaffected	Unaffected	Unaffected	229
TCMP(85)	↕	Unaffected	Unaffected	↕	Unaffected	Unaffected	243
ASC(86)	↕	Unaffected	Unaffected	Unaffected	Unaffected	Unaffected	262
SBS(91)							313
MCRO(99)	↕	Unaffected	Unaffected	Unaffected	Unaffected	Unaffected	319

### Expansion Instructions (All CQM1/SRM1)

Instructions	25503 (ER)	25504 (CY)	25505 (GR)	25506 (EQ)	25507 (LE)	25402 (N)	Page						
ASFT(17)	↕	Unaffected	Unaffected	Unaffected	Unaffected	Unaffected	230						
TKY(18)							347						
MCMP(19)	↕	Unaffected	Unaffected	↕	Unaffected	Unaffected	247						
RXD(47)	↕	Unaffected	Unaffected	Unaffected	Unaffected	Unaffected	340						
TXD(48)							342						
CMPL(60)	↕	Unaffected	↕	↕	↕	↕	246						
INI(61)	↕	↕	Unaffected	Unaffected	Unaffected	Unaffected	220						
PRV(62)							221						
CTBL(63)							215						
SPED(64)							330						
PULS(65)							328						
SCL(66)							↕	Unaffected	Unaffected	↕	Unaffected	Unaffected	266
BCNT(67)							320						
BCMP(68)	↕	Unaffected	Unaffected	Unaffected	Unaffected	Unaffected	244						
STIM(69)							213						
DSW(87) <sup>1</sup>							346						
7SEG(88) <sup>2</sup>							345						
INT(89)							326						
HKY(—) <sup>3</sup>							347						
FPD(—)							↕	↕	Unaffected	Unaffected	Unaffected	Unaffected	322
SRCH(—)	↕	Unaffected	Unaffected	↕	Unaffected	Unaffected	337						
MAX(—)							↕	300					
MIN(—)							↕	301					
APR(—)							↕	305					
COLM(—)								274					
LINE(—)								273					
HMS(—)								272					
SEC(—)								271					
SUM(—)							↕	303					
FCS(—)							↕	Unaffected	Unaffected	Unaffected	Unaffected	Unaffected	321
HEX(—)													263
AVG(—)													302

- Note**
1. Only expansion instructions with default function numbers are applicable to the SRM1 PCs.
  2. SR 25410 will be ON when DSW(87) is being executed.
  3. SR 25409 will be ON when 7SEG(88) is being executed.
  4. SR 25408 will be ON when HKY(—) is being executed.

## Expansion Instructions (CQM1-CPU4□-E/-EV1 Only)

Instructions	25503 (ER)	25504 (CY)	25505 (GR)	25506 (EQ)	25507 (LE)	Page
PWM(—)	↕	Unaffected	Unaffected	Unaffected	Unaffected	336
PID(—)						338
ADBL(—) <sup>1</sup>	↕	↕	Unaffected	↕	Unaffected	293
SBBL(—) <sup>1</sup>						294
MBS(—)	↕	Unaffected	Unaffected	↕	Unaffected	296
DBS(—)						298
MBSL(—)						297
DBSL(—)						299
CPS(—)	↕	Unaffected	↕	↕	↕	248
CPSL(—)						249
NEG(—) <sup>2</sup>	↕	Unaffected	Unaffected	↕	Unaffected	275
NEGL(—) <sup>2</sup>						276
ZCP(—)	↕	Unaffected	↕	↕	↕	250
ZCPL(—)						252
XFRB(—)	↕	Unaffected	Unaffected	Unaffected	Unaffected	241
PLS2(—)						332
ACC(—)						334
SCL2(—)	↕	↕	Unaffected	↕	Unaffected	268
SCL3(—)	↕	Unaffected	Unaffected	↕	Unaffected	269

- Note**
1. Depending on the calculation results, ADBL(—) and SBBL(—) may also affect the status of the overflow and underflow flags (SR 25404 and SR 25405).
  2. Depending on the conversion results, NEG(—) and NEGL(—) may also affect the status of the underflow flag (SR 25405).

# Appendix C

## Memory Areas

### CQM1 Memory Areas

#### Memory Area Structure

The following memory areas can be used with the CQM1.

Data area		Size	Words	Bits	Function
IR area <sup>1</sup>	Input area	128 or 256 bits	IR 000 to IR 015	IR 00000 to IR 01515	CQM1-CPU11/21-E: Up to 8 words (128 bits) can be used for I/O bits. Up to 7 Units can be connected. CQM1-CPU4□-EV1: Up to 16 words (256 bits) can be used for I/O bits. Up to 11 Units can be connected.
	Output area		IR 100 to IR 115	IR 10000 to IR 11515	
	Work areas	2,720 bits min. <sup>2</sup>	IR 012 to IR 095	IR 01200 to IR 09515	Work bits do not have any specific function, and they can be freely used within the program.
			IR 112 to IR 195	IR 11200 to IR 19515	
IR 216 to IR 219			IR 21600 to IR 21915		
IR 224 to IR 229	IR 22400 to IR 22915				
MACRO operand area <sup>1</sup>	Input area	64 bits	IR 096 to IR 099	IR 09600 to IR 09915	Used when the MACRO instruction, <b>MCRO(99)</b> , is used. When the MACRO instruction is not used, these bits may be used as work bits.
	Output area	64 bits	IR 196 to IR 199	IR 19600 to IR 19915	
Analog SV area <sup>1</sup>		64 bits	IR 220 to IR 223	IR 22000 to IR 22315	CQM1-CPU42-EV1: Used to store the analog set values. (Cannot be used as work bits.) Can be used as work bits in other CPU Units.
High-speed Counter 0 PV <sup>1</sup>		32 bits	IR 230 to IR 231	IR 23000 to IR 23115	Used to store the present values of high-speed counter 0.
Port 1 and 2 Pulse Output PVs <sup>1</sup>		64 bits	IR 236 to IR 239	IR 23600 to IR 23915	CQM1-CPU43-EV1: Used to store the present values of pulse outputs for ports 1 and 2. (Cannot be used as work bits.) CQM1-CPU44-EV1: Used by the system. (Cannot be used as work bits.) Can be used as work bits in other CPU Units.
High-speed Counter 1 and 2 PVs <sup>1</sup>		64 bits	IR 232 to IR 235	IR 23200 to IR 23515	CQM1-CPU43/44-EV1: Used to store the present values of high-speed counters 1 and 2 for ports 1 and 2. (Cannot be used as work bits.) Can be used as work bits in other CPU Units.
Expansion Areas <sup>1</sup>		320 bits	IR 200 to IR 215 IR 240 to IR 243	IR 20000 to IR 21515 IR 24000 to IR 24315	These bits are expected to be used in planned function expansion.
SR area		184 bits	SR 244 to SR 255	SR 24400 to SR 25507	These bits serve specific functions such as flags and control bits. Can be used as work bits.
TR area		8 bits	---	TR 0 to TR 7	These bits are used to temporarily store ON/OFF status at program branches.
HR area		1,600 bits	HR 00 to HR 99	HR 0000 to HR 9915	These bits store data and retain their ON/OFF status when power is turned off.
AR area		448 bits	AR 00 to AR 27	AR 0000 to AR 2715	These bits serve specific functions such as flags and control bits.

Data area		Size	Words	Bits	Function
LR area <sup>1</sup>		1,024 bits	LR 00 to LR 63	LR 0000 to LR 6315	Used for 1:1 data link through the RS-232 port.
Timer/Counter area <sup>3</sup>		512 bits	TC 000 to TC 511 (timer/counter numbers)		The same numbers are used for both timers and counters. TC 000 to TC 002 are used for interval timers.
DM area	Read/write	1,024 words	DM 0000 to DM 1023	---	DM area data can be accessed in word units only. Word values are retained when the power is turned off.
		5,120 words	DM 1024 to DM 6143	---	Available in CQM1-CPU4□-EV1 only. <sup>4</sup>
	Read-only <sup>5</sup>	425 words	DM 6144 to DM 6568	---	Cannot be overwritten from program.
	Error history area <sup>5</sup>	31 words	DM 6569 to DM 6599	---	Used to store the time of occurrence and error code of errors that occur.
	PC Setup <sup>5</sup>	56 words	DM 6600 to DM 6655	---	Used to store various parameters that control PC operation.
User program area (UM area)		3,200 or 7,200 words	---		Used to store the program. Retained when the power is turned off. CQM1-CPU11/21-E: 3,200 words CQM1-CPU4□-EV1: 7,200 words

- Note**
1. IR and LR bits that are not used for their allocated functions can be used as work bits.
  2. At least 2,720 bits can be used as work bits. The total number of bits that can be used depends on the configuration of the PC system.
  3. When accessing a PV, TC numbers are used as word data; when accessing Completion Flags, they are used as bit data.
  4. Although the CQM1-CPU11-E and CQM1-CPU21-E do not support DM 1024 through DM 6143, an error will not occur if they are addressed. Any attempt to write to these words will have no effect and any reads will produce all zeros.
  5. Data in DM 6144 to DM 6655 cannot be overwritten from the program.

## SR Area

These bits mainly serve as flags related to CQM1 operation. The following table provides details on the various bit functions.

Word	Bit(s)	Function	Page
SR 244	00 to 15	<b>Input Interrupt 0 Counter Mode SV</b> SV when input interrupt 0 is used in counter mode (4 digits hexadecimal, 0000 to FFFF). (Can be used as work bits when input interrupt 0 is not used in counter mode.)	42
SR 245	00 to 15	<b>Input Interrupt 1 Counter Mode SV</b> SV when input interrupt 1 is used in counter mode (4 digits hexadecimal, 0000 to FFFF). (Can be used as work bits when input interrupt 1 is not used in counter mode.)	
SR 246	00 to 15	<b>Input Interrupt 2 Counter Mode SV</b> SV when input interrupt 2 is used in counter mode (4 digits hexadecimal, 0000 to FFFF). (Can be used as work bits when input interrupt 2 is not used in counter mode.)	
SR 247	00 to 15	<b>Input Interrupt 3 Counter Mode SV</b> SV when input interrupt 3 is used in counter mode (4 digits hexadecimal, 0000 to FFFF). (Can be used as work bits when input interrupt 3 is not used in counter mode.)	
SR 248	00 to 15	<b>Input Interrupt 0 Counter Mode PV Minus One</b> Counter PV-1 when input interrupt 0 is used in counter mode (4 digits hexadecimal).	42
SR 249	00 to 15	<b>Input Interrupt 1 Counter Mode PV Minus One</b> Counter PV-1 when input interrupt 1 is used in counter mode (4 digits hexadecimal).	
SR 250	00 to 15	<b>Input Interrupt 2 Counter Mode PV Minus One</b> Counter PV-1 when input interrupt 2 is used in counter mode (4 digits hexadecimal).	
SR 251	00 to 15	<b>Input Interrupt 3 Counter Mode PV Minus One</b> Counter PV-1 when input interrupt 3 is used in counter mode (4 digits hexadecimal).	

Word	Bit(s)	Function	Page
SR 252	00	<b>High-speed Counter 0 Reset Bit</b>	48
	01	<b>CQM1-CPU43-EV1: High-speed Counter 1 Reset Bit</b> Turn ON to reset PV of high-speed counter 1 (port 1). <b>CQM1-CPU44-EV1: Absolute High-speed Counter 1 Origin Compensation Bit</b> Turn ON to set origin compensation for absolute high-speed counter 1 (port 1). Automatically turns OFF when compensation value is set in DM 6611.	134
	01	<b>CQM1-CPU43-EV1: High-speed Counter 2 Reset Bit</b> Turn ON to reset PV of high-speed counter 2 (port 2). <b>CQM1-CPU44-EV1: Absolute High-speed Counter 2 Origin Compensation Bit</b> Turn ON to set origin compensation for absolute high-speed counter 2 (port 2). Automatically turns OFF when compensation value is set in DM 6612.	134
	03 to 07	Not used.	
	08	<b>Peripheral Port Reset Bit</b> Turn ON to reset peripheral port. (Not valid when peripheral device is connected.) Automatically turns OFF when reset is complete.	92
	09	<b>RS-232C Port Reset Bit</b> Turn ON to reset RS-232C port. Automatically turns OFF when reset is complete.	
	10	<b>PC Setup Reset Bit</b> Turn ON to initialize PC Setup (DM 6600 through DM 6655). Automatically turns OFF again when reset is complete. Only effective if the PC is in PROGRAM mode.	3
	11	<b>Forced Status Hold Bit</b> OFF: Bits that are forced set/reset are cleared when switching from PROGRAM mode to MONITOR mode. ON: The status of bits that are forced set/reset are maintained when switching from PROGRAM mode to MONITOR mode.	17
	12	<b>I/O Hold Bit</b> OFF: IR and LR bits are reset when starting or stopping operation. ON: IR and LR bit status is maintained when starting or stopping operation.	17
	13	Not used.	
	14	<b>Error Log Reset Bit</b> Turn ON to clear error log. Automatically turns OFF again when operation is complete.	431
	15	<b>Output OFF Bit</b> OFF: Normal output status. ON: All outputs turned OFF.	455
	SR 253	00 to 07	<b>FAL Error Code</b> The error code (a 2-digit number) is stored here when an error occurs. The FAL number is stored here when FAL(06) or FALS(07) is executed. This word is reset (to 00) by executing a FAL 00 instruction or by clearing the error from a Peripheral Device.
08		<b>Low Battery Flag</b> Turns ON when a CPU Unit battery voltage drops.	428
09		<b>Cycle Time Overrun Flag</b> Turns ON when a cycle time overrun occurs (i.e., when the cycle time exceeds 100 ms).	428
10 to 12		Not used.	
13		<b>Always ON Flag</b>	---
14		<b>Always OFF Flag</b>	---
15		<b>First Cycle Flag</b> Turns ON for 1 cycle at the start of operation.	---

Word	Bit(s)	Function	Page
SR 254	00	1-minute clock pulse (30 seconds ON; 30 seconds OFF)	---
	01	0.02-second clock pulse (0.01 second ON; 0.01 second OFF)	---
	02 to 03	Not used.	
	04	<b>CQM1-CPU4□-EV1: Overflow (OF) Flag</b> Turns ON when the result of a calculation is above the upper limit of signed binary data.	289
	05	<b>CQM1-CPU4□-EV1: Underflow (UF) Flag</b> Turns ON when the result of a calculation is below the lower limit of signed binary data.	289
	06	<b>Differential Monitor Complete Flag</b> Turns ON when differential monitoring is complete.	134
	07	<b>STEP(08) Execution Flag</b> Turns ON for 1 cycle only at the start of process based on STEP(08).	206
	08	<b>HKY(—) Execution Flag</b> Turns ON during execution of HKY(—).	347
	09	<b>7SEG(88) Execution Flag</b> Turns ON during execution of 7SEG(88).	345
	10	<b>DSW(87) Execution Flag</b> Turns ON during execution of DSW(87).	346
	11 to 14	Not used.	
	15	<b>CQM1-CPU43-EV1: Pulse I/O Error Flag (FALS: 9C)</b> Turns ON when there is an error in a pulse I/O function using port 1 or 2. <b>CQM1-CPU44-EV1: Absolute High-speed Counter Error Flag (FALS: 9C)</b> Turns ON when there is an error in an absolute high-speed counter using port 1 or 2.	427
	SR 255	00	0.1-second clock pulse (0.05 second ON; 0.05 second OFF)
01		0.2-second clock pulse (0.1 second ON; 0.1 second OFF)	---
02		1.0-second clock pulse (0.5 second ON; 0.5 second OFF)	---
03		<b>Instruction Execution Error (ER) Flag</b> Turns ON when an error occurs during execution of an instruction.	---
04		<b>Carry (CY) Flag</b> Turns ON when there is a carry in the results of an instruction execution.	---
05		<b>Greater Than (GR) Flag</b> Turns ON when the result of a comparison operation is "greater."	---
06		<b>Equals (EQ) Flag</b> Turns ON when the result of a comparison operation is "equal," or when the result of an instruction execution is 0.	---
07		<b>Less Than (LE) Flag</b> Turns ON when the result of a comparison operation is "less."	---
08 to 15		Not used.	

**Note** Writing is not possible for the following words: SR 248 through SR 251, and SR 253 through SR255.

## Explanation of SR Bits

### SR 25211 (Forced Status Hold Bit)

When the forced set/reset status is cleared, the bits that were forced will be turned ON or OFF as follows:

Forced set cleared: Bit turned ON  
Forced reset cleared: Bit turned OFF

All force-set or force-reset bits will be cleared when the PC is switched to RUN mode (see note).

This bit is turned ON and OFF from a peripheral device.

A setting can be made in the PC Setup (DM 6601) to cause the status of this Bit to be retained even when powering up.

**Note** DM 6601 in the PC Setup can be set to maintain the previous status of the Forced Status Hold Bit when power is turned on. This setting can be used to prevent forced status from being cleared even when power is turned on.

**SR 25212 (I/O Hold Bit)**

This bit is turned ON and OFF from a peripheral device.

A setting can be made in the PC Setup (DM 6601) to cause the status of this Bit to be retained even when powering up.

**SR 25215 (Output OFF Bit)**

When this bit is turned ON, all outputs will be turned OFF and the CPU Unit's OUT INH indicator will light. Outputs will remain OFF even if output bits are turned ON by the program until this Bit is turned OFF again.

**SR 25308 (Battery Low Flag) and SR 25309 (Cycle Time Overrun Flag)**

A setting can be made in the PC Setup (DM 6655) so that these errors will not be generated.

**AR Area**

These bits mainly serve as flags related to CQM1 operation. The following table provides details on the various bit functions.

With the exception of AR 23 (Power-off Counter), the status of AR words and bits is refreshed each cycle. (AR 23 is refreshed only for power interruptions.)

Word	Bit(s)	Function	Page
AR 00 to AR 03	---	Not used.	
AR 04	08 to 15	<b>CQM1-CPU43/44-EV1: Pulse I/O or Absolute High-speed Counter Status Code:</b> 00: Normal 01, 02: Hardware error 03: PC Setup error 04: PC stopped during pulse output	59
AR 05	00 to 07	<b>CQM1-CPU43/44-EV1: High-speed Counter 1 Range Comparison Flags</b> 00 ON: Counter PV is within comparison range 1 01 ON: Counter PV is within comparison range 2 02 ON: Counter PV is within comparison range 3 03 ON: Counter PV is within comparison range 4 04 ON: Counter PV is within comparison range 5 05 ON: Counter PV is within comparison range 6 06 ON: Counter PV is within comparison range 7 07 ON: Counter PV is within comparison range 8	59
	08	<b>CQM1-CPU43/44-EV1: High-speed Counter 1 Comparison Flag</b> OFF: Stopped ON: Comparing	59
	09	<b>CQM1-CPU43/44-EV1: High-speed Counter 1 Overflow/Underflow Flag</b> OFF: Normal ON: Overflow or underflow occurred.	59
	10 to 11	Not used.	
	12 to 15	<b>CQM1-CPU43-EV1: Port 1 Pulse Output Flags</b> 12 ON: Deceleration specified. (OFF: Not specified.) 13 ON: Number of pulses specified. (OFF: Not specified.) 14 ON: Pulse output completed. (OFF: Not completed.) 15 ON: Pulse output in progress. (OFF: No pulse output.)	333
AR 06	00 to 15	<b>CQM1-CPU43/44-EV1: High-speed Counter 2/Port 2 Pulse Output Flags</b> Identical to the High-speed Counter 1/Port 1 Pulse Output Flags in AR 05.	59
AR 07	00 to 11	Not used.	
	12	<b>DIP Switch Pin 6 Flag</b> OFF: CPU Unit's DIP switch pin no. 6 is OFF. ON: CPU Unit's DIP switch pin no. 6 is ON.	---
	13 to 15	Not used.	



Word	Bit(s)	Function	Page	
AR 08	00 to 03	<b>RS-232C Communications Error Code</b> (1-digit number) The code will be "F" when a computer running LSS/SSS is connected to the Peripheral Port.	92	
	04	<b>RS-232C Error Flag</b> Turns ON when an RS-232C communications error occurs.		
	05	<b>RS-232C Transmission Enabled Flag</b> Valid only when host link, RS-232C communications are used.		
	06	<b>RS-232C Reception Completed Flag</b> Valid only when RS-232C communications are used.		
	07	<b>RS-232C Reception Overflow Flag</b> Valid only when RS-232C communications are used.)		
	08 to 11	<b>Peripheral Device Error Code</b> (1-digit number) The code will be "F" when a computer running LSS/SSS is connected to the Peripheral Port.		99
		12	<b>Peripheral Device Error Flag</b> Turns ON when a peripheral device communications error occurs.	
		13	<b>Peripheral Device Transmission Enabled Flag</b> Valid only when host link, RS-232C communications are used.	
		14	<b>Peripheral Device Reception Completed Flag</b> Valid only when RS-232C communications are used.	
		15	<b>Peripheral Device Reception Overflow Flag</b> Valid only when RS-232C communications are used.	
AR 09	00 to 15	<b>RS-232C Reception Counter</b> 4 digits BCD; valid only when RS-232C communications are used.	99	
AR 10	00 to 15	<b>Peripheral Device Reception Counter</b> 4 digits BCD; valid only when RS-232C communications are used.	99	
AR 11	00 to 07	<b>High-speed Counter 0 Range Comparison Flags</b> 00 ON: Counter PV is within comparison range 1 01 ON: Counter PV is within comparison range 2 02 ON: Counter PV is within comparison range 3 03 ON: Counter PV is within comparison range 4 04 ON: Counter PV is within comparison range 5 05 ON: Counter PV is within comparison range 6 06 ON: Counter PV is within comparison range 7 07 ON: Counter PV is within comparison range 8	51	
	08 to 14	Not used.		
	15	<b>Pulse Output Status for Pulse Output Bit Specification</b> 0: Stopped; 1: Output (This bit is supported only for CPU Units produced in or after April 1995 (Lot No.: **45)).	---	
AR 12	00 to 15	Not used.		
AR 13	00	<b>Memory Cassette Installed Flag</b> Turns ON if the Memory Cassette is installed at the time of powering up.	---	
	01	<b>Clock Available Flag</b> Turns ON if a Memory Cassette equipped with a clock is installed.	---	
	02	<b>Memory Cassette Write-protected Flag</b> ON when an EEPROM Memory Cassette is mounted and write protected or when an EPROM Memory cassette is mounted.	134	
	03	Not used.		
	04 to 07	<b>Memory Cassette Code</b> (1-digit number) 0: No Memory Cassette installed. 1: EEPROM, 4K-word Memory Cassette installed. 2: EEPROM, 8K-word Memory Cassette installed. 4: EPROM-type Memory Cassette installed.	---	
	08 to 15	Not used.		

Word	Bit(s)	Function	Page
AR 14	00	<b>CPU Unit to Memory Cassette Transfer Bit</b> Turn ON for transfer from the CPU Unit to the Memory Cassette. Automatically turns OFF again when operation is complete. (See note.)	148
	01	<b>Memory Cassette to CPU Unit Transfer Bit</b> Turn ON for transfer from the Memory Cassette to the CPU Unit. Automatically turns OFF again when operation is complete. (See note.)	149
	02	<b>Memory Cassette Compare Flag</b> ON when the contents of the PC and the Memory Cassette are being compared. Turns OFF automatically when comparison has completed. (See note.)	149
	03	<b>Memory Cassette Comparison Results Flag</b> ON: Difference found or comparison not possible OFF: Contents compared and found to be the same.	149
	04 to 11	Not used.	
	12	<b>PROGRAM Mode Transfer Error Flag</b> Turns ON when transfer could not be executed due to being in PROGRAM mode.	148
	13	<b>Write-protect Error Flag</b> Turns ON when transfer could not be executed due to write-protection.	427
	14	<b>Insufficient Capacity Flag</b> Turns ON when transfer could not be executed due to insufficient capacity at the transfer destination.	427
	15	<b>No Program Flag</b> Turns ON when transfer could not be executed due to there being no program in the Memory Cassette.	427
	AR 15	00 to 07	<b>Memory Cassette Program Code</b> Code (2-digit number) indicates the size of the program stored in the Memory Cassette. 00: There is no program, or no Memory Cassette is installed. 04: The program is less than 3.2K words long. 08: The program is less than 7.2K words long.
08 to 15		<b>CPU Unit Program Code</b> Code (2-digit number) indicates the size of the program stored in the CPU Unit. 04: The program is less than 3.2K words long. 08: The program is less than 7.2K words long.	---
AR 16	00 to 10	Not used.	
	11	<b>PC Setup Initialized Flag</b> Turns ON when a checksum error occurs in the PC Setup area and all settings are initialized back to the default settings.	429
	12	<b>Program Invalid Flag</b> Turns ON when a checksum error occurs in the UM area, or when an improper instruction is executed.	429
	13	<b>Instructions Table Initialized Flag</b> Turns ON when a checksum error occurs in the instructions table and all settings are initialized back to the default settings.	429
	14	<b>Memory Cassette Added Flag</b> Turns ON if the Memory Cassette is installed while the power is on.	429
	15	<b>Memory Cassette Transfer Error Flag</b> Turns ON if a transfer cannot be successfully executed when DIP switch pin no. 2 is set to ON (i.e., set to automatically transfer the contents of the Memory Cassette at power-up.)	429

**Note** With the LSS/SSS, clear the forced-set status to turn OFF these bits, which will not change even after the operation is complete. (If the Programming Console is being used, these bits will automatically turn OFF.)

Word	Bit(s)	Function	Page
AR 17	00 to 07	"Minutes" portion of the present time, in 2 digits BCD (Valid only when a Memory Cassette with the clock function is installed.)	134
	08 to 15	"Hour" portion of the present time, in 2 digits BCD (Valid only when a Memory Cassette with the clock function is installed.)	
AR 18	00 to 07	"Seconds" portion of the present time, in 2 digits BCD (Valid only when a Memory Cassette with the clock function is installed.)	
	08 to 15	"Minutes" portion of the present time, in 2 digits BCD (Valid only when a Memory Cassette with the clock function is installed.)	
AR 19	00 to 07	"Hour" portion of the present time, in 2 digits BCD (Valid only when a Memory Cassette with the clock function is installed.)	
	08 to 15	"Date" portion of the present time, in 2 digits BCD (Valid only when a Memory Cassette with the clock function is installed.)	
AR 20	00 to 07	"Month" portion of the present time, in 2 digits BCD (Valid only when a Memory Cassette with the clock function is installed.)	
	08 to 15	"Year" portion of the present time, in 2 digits BCD (Valid only when a Memory Cassette with the clock function is installed.)	
AR 21	00 to 07	"Day of week" portion of the present time, in 2 digits BCD [00: Sunday to 06: Saturday] (Valid only when a Memory Cassette with the clock function is installed.)	134
	08 to 12	Not used.	
	13	<b>30-second Adjustment Bit</b> Valid only when a Memory Cassette with the clock function is installed.	
	14	<b>Clock Stop Bit</b> Valid only when a Memory Cassette with the clock function is installed.	
	15	<b>Clock Set Bit</b> Valid only when a Memory Cassette with the clock function is installed.	
AR 22	00 to 07	<b>Input Words</b> Number of words for input bits (2 digits BCD)	137
	08 to 15	<b>Output Words</b> Number of words for output bits (2 digits BCD)	
AR 23	00 to 15	<b>Power-off Counter</b> (4 digits BCD) This is the count of the number of times that the power has been turned off. To clear the count, write "0000" from a peripheral device.	---
AR 24	00	<b>Power-up PC Setup Error Flag</b> Turns ON when there is an error in DM 6600 to DM 6614 (the part of the PC Setup area that is read at power-up).	3
	01	<b>Start-up PC Setup Error Flag</b> Turns ON when there is an error in DM 6615 to DM 6644 (the part of the PC Setup area that is read at the beginning of operation).	428
	02	<b>RUN PC Setup Error Flag</b> Turns ON when there is an error in DM 6645 to DM 6655 (the part of the PC Setup area that is always read).	3
	03, 04	Not used.	
	05	<b>Long Cycle Time Flag</b> Turns ON if the actual cycle time is longer than the cycle time set in DM 6619.	---
	06, 07	Not used.	
	08 to 15	Code (2 digits hexadecimal) showing the word number of a detected I/O bus error 00 to 07: Correspond to input words 000 to 007. 80 to 87: Correspond to output words 100 to 107. FF: End cover cannot be confirmed.	---
AR 25	00 to 07	Not used.	
	08	<b>FPD(—) Teaching Bit</b>	322
	09 to 15	Not used.	

Word	Bit(s)	Function	Page
AR 26	00 to 15	<b>Maximum Cycle Time</b> (4 digits BCD) The longest cycle time since the beginning of operation is stored. It is cleared at the beginning, and not at the end, of operation.  The unit can be any of the following, depending on the setting of the 9F monitoring time (DM 6618). Default: 0.1 ms; "10 ms" setting: 0.1 ms; "100 ms" setting: 1 ms; "1 s" setting: 10 ms	21
AR 27	00 to 15	<b>Current Cycle Time</b> (4 digits BCD) The most recent cycle time during operation is stored. The Current Cycle Time is not cleared when operation stops.  The unit can be any of the following, depending on the setting of the 9F monitoring time (DM 6618). Default: 0.1 ms; "10 ms" setting: 0.1 ms; "100 ms" setting: 1 ms; "1 s" setting: 10 ms	

## CPM1/CPM1A Memory Areas

### Memory Area Structure

The following memory areas can be used with the CPM1/CPM1A.

Data area		Words	Bits	Function
IR area <sup>1</sup>	Input area	IR 000 to IR 009 (10 words)	IR 00000 to IR 00915 (160 bits)	These bits can be allocated to the external I/O terminals.
	Output area	IR 010 to IR 019 (10 words)	IR 01000 to IR 01915 (160 bits)	
	Work area	IR 200 to IR 231 (32 words)	IR 20000 to IR 23115 (512 bits)	Work bits can be freely used within the program.
SR area		SR 232 to SR 255 (24 words)	SR 23200 to SR 25515 (384 bits)	These bits serve specific functions such as flags and control bits.
TR area		---	TR 0 to TR 7 (8 bits)	These bits are used to temporarily store ON/OFF status at program branches.
HR area <sup>2</sup>		HR 00 to HR 19 (20 words)	HR 0000 to HR 1915 (320 bits)	These bits store data and retain their ON/OFF status when power is turned off.
AR area <sup>2</sup>		AR 00 to AR 15 (16 words)	AR 0000 to AR 1515 (256 bits)	These bits serve specific functions such as flags and control bits.
LR area <sup>1</sup>		LR 00 to LR 15 (16 words)	LR 0000 to LR 1515 (256 bits)	Used for a 1:1 data link with another PC.
Timer/Counter area <sup>2</sup>		TC 000 to TC 127 (timer/counter numbers) <sup>3</sup>		The same numbers are used for both timers and counters.
DM area	Read/write <sup>2</sup>	DM 0000 to DM 0999 DM 1022 to DM 1023 (1,002 words)	---	DM area data can be accessed in word units only. Word values are retained when the power is turned off.
	Error log <sup>2</sup>	DM 1000 to DM 1021 (22 words)	---	Used to store the error code of errors that occur. These words can be used as ordinary read/write DM when the error log function isn't being used.
	Read-only <sup>4</sup>	DM 6144 to DM 6599 (456 words)	---	Cannot be overwritten from program.
	PC Setup <sup>4</sup>	DM 6600 to DM 6655 (56 words)	---	Used to store various parameters that control PC operation.

- Note**
1. IR and LR bits that are not used for their allocated functions can be used as work bits.
  2. The contents of the HR area, Counter area, and read/write DM area are backed up by a capacitor. At 25°C, the capacitor will back up memory for 20 days. Refer to 2-1-2 *Characteristics* in the *CPM1 or CPM1A Operation Manual* for a graph showing the backup time vs. temperature.
  3. When accessing a PV, TC numbers are used as word data; when accessing Completion Flags, they are used as bit data.
  4. Data in DM 6144 to DM 6655 cannot be overwritten from the program, but they can be changed from a Peripheral Device.

5. The contents of the HR area, LR area, Counter area, and read/write DM area are backed up by a capacitor. The backup time varies with the ambient temperature, but at 25°C, the capacitor will back up memory for 20 days. If the power supply is off longer than the backup time, memory contents will be cleared and AR1314 will turn ON. (This flag turns ON when data can no longer be retained by the built-in capacitor.) Refer to 2-1-2 *Characteristics* in the *CPM1 Operation Manual* for a graph showing the back-up time vs. temperature.

## SR Area

These bits mainly serve as flags related to CPM1/CPM1A operation or contain present and set values for various functions. The functions of the SR area are explained in the following table.

Word(s)	Bit(s)	Function	Page
SR 232 to SR 235	00 to 15	<b>Macro Function Input Area</b> Contains the input operands for MCRO(99). (Can be used as work bits when MCRO(99) is not used.)	128
SR 236 to SR 239	00 to 15	<b>Macro Function Output Area</b> Contains the output operands for MCRO(99). (Can be used as work bits when MCRO(99) is not used.)	
SR 240	00 to 15	<b>Input Interrupt 0 Counter Mode SV</b> SV when input interrupt 0 is used in counter mode (4 digits hexadecimal). (Can be used as work bits when input interrupt 0 is not used in counter mode.)	42
SR 241	00 to 15	<b>Input Interrupt 1 Counter Mode SV</b> SV when input interrupt 1 is used in counter mode (4 digits hexadecimal). (Can be used as work bits when input interrupt 1 is not used in counter mode.)	
SR 242	00 to 15	<b>Input Interrupt 2 Counter Mode SV</b> SV when input interrupt 2 is used in counter mode (4 digits hexadecimal). (Can be used as work bits when input interrupt 2 is not used in counter mode.)	
SR 243	00 to 15	<b>Input Interrupt 3 Counter Mode SV</b> SV when input interrupt 3 is used in counter mode (4 digits hexadecimal). (Can be used as work bits when input interrupt 3 is not used in counter mode.)	
SR 244	00 to 15	<b>Input Interrupt 0 Counter Mode PV Minus One</b> Counter PV-1 when input interrupt 0 is used in counter mode (4 digits hexadecimal).	42
SR 245	00 to 15	<b>Input Interrupt 1 Counter Mode PV Minus One</b> Counter PV-1 when input interrupt 1 is used in counter mode (4 digits hexadecimal).	
SR 246	00 to 15	<b>Input Interrupt 2 Counter Mode PV Minus One</b> Counter PV-1 when input interrupt 2 is used in counter mode (4 digits hexadecimal).	
SR 247	00 to 15	<b>Input Interrupt 3 Counter Mode PV Minus One</b> Counter PV-1 when input interrupt 3 is used in counter mode (4 digits hexadecimal).	
SR 248, SR 249	00 to 15	<b>High-speed Counter PV Area</b> (Can be used as work bits when the high-speed counter is not used.)	48
SR 250	00 to 15	<b>Analog Volume Setting 0</b> Used to store the 4-digit BCD set value (0000 to 0200) from analog volume control 0.	129
SR 251	00 to 15	<b>Analog Volume Setting 1</b> Used to store the 4-digit BCD set value (0000 to 0200) from analog volume control 1.	

Word(s)	Bit(s)	Function	Page
SR 252	00	<b>High-speed Counter Reset Bit</b>	48
	01 to 07	Not used.	
	08	<b>Peripheral Port Reset Bit</b> Turn ON to reset peripheral port. (Not valid when peripheral device is connected.) Automatically turns OFF when reset is complete.	92
	09	Not used.	
	10	<b>PC Setup Reset Bit</b> Turn ON to initialize PC Setup (DM 6600 through DM 6655). Automatically turns OFF again when reset is complete. Only effective if the PC is in PROGRAM mode.	3
	11	<b>Forced Status Hold Bit</b> (See note.) OFF: The forced status of bits that are forced set/reset is cleared when switching between PROGRAM mode and MONITOR mode. ON: The status of bits that are forced set/reset are maintained when switching between PROGRAM mode and MONITOR mode. The status of this bit can be maintained when PC power turns off by using the PC Setup.	17
	12	<b>I/O Hold Bit</b> (See note.) OFF: IR and LR bits are reset when starting or stopping operation. ON: IR and LR bit status is maintained when starting or stopping operation. The status of this bit can be maintained when PC power turns off by using the PC Setup.	17
	13	Not used.	
	14	<b>Error Log Reset Bit</b> Turn ON to clear error log. Automatically turns OFF again when operation is complete.	431
	15	Not used.	
SR 253	00 to 07	<b>FAL Error Code</b> The error code (a 2-digit number) is stored here when an error occurs. The FAL number is stored here when FAL(06) or FALS(07) is executed. This word is reset (to 00) by executing a FAL 00 instruction or by clearing the error from a Peripheral Device.	205
	08	Not used.	
	09	<b>Cycle Time Overrun Flag</b> Turns ON when a cycle time overrun occurs (i.e., when the cycle time exceeds 100 ms).	---
	10 to 12	Not used.	
	13	<b>Always ON Flag</b>	---
	14	<b>Always OFF Flag</b>	---
	15	<b>First Cycle Flag</b> Turns ON for 1 cycle at the start of operation.	---
SR 254	00	1-minute clock pulse (30 seconds ON; 30 seconds OFF)	---
	01	0.02-second clock pulse (0.01 second ON; 0.01 second OFF)	---
	02	<b>Negative (N) Flag</b>	---
	03 to 05	Not used.	
	06	<b>Differential Monitor Complete Flag</b> Turns ON when differential monitoring is complete.	134
	07	<b>STEP(08) Execution Flag</b> Turns ON for 1 cycle only at the start of process based on STEP(08).	206
	08 to 15	Not used.	

Word(s)	Bit(s)	Function	Page
SR 255	00	0.1-second clock pulse (0.05 second ON; 0.05 second OFF)	---
	01	0.2-second clock pulse (0.1 second ON; 0.1 second OFF)	---
	02	1.0-second clock pulse (0.5 second ON; 0.5 second OFF)	---
	03	<b>Instruction Execution Error (ER) Flag</b> Turns ON when an error occurs during execution of an instruction.	---
	04	<b>Carry (CY) Flag</b> Turns ON when there is a carry in the results of an instruction execution.	---
	05	<b>Greater Than (GR) Flag</b> Turns ON when the result of a comparison operation is "greater."	---
	06	<b>Equals (EQ) Flag</b> Turns ON when the result of a comparison operation is "equal," or when the result of an instruction execution is 0.	---
	07	<b>Less Than (LE) Flag</b> Turns ON when the result of a comparison operation is "less."	---
	08 to 15	Not used.	

**Note** DM 6601 in the PC Setup can be set to maintain the previous status of the I/O Hold Bit (SR 25212) and the I/O Hold Bit (SR 25212) when power is turned off. If power is left OFF for longer than the backup time, however, status may be cleared. For details regarding the backup time, refer to the *CPM1A or CPM1 Operation Manual*. Refer to 1-1-3 *CPM1/CPM1A PC Setup Settings* for details on the PC Setup.

## AR Area

These bits mainly serve as flags related to CPM1/CPM1A operation. These bits retain their status even after the CPM1/CPM1A power supply has been turned off or when operation begins or stops.

Word(s)	Bit(s)	Function	Page
AR 00, AR 01	00 to 15	Not used.	
AR 02	00 to 07	Not used.	---
	08 to 11	<b>Number of I/O Units Connected</b>	
	12 to 15	Not used.	
AR 03 to AR 07	00 to 15	Not used.	
AR 08	00 to 07	Not used.	
	08 to 11	<b>Peripheral Device Error Code</b> 0: Normal completion 1: Parity error 2: Frame error 3: Overrun error	99
	12	<b>Peripheral Device Error Flag</b>	
	13 to 15	Not used.	
AR 09	00 to 15	Not used.	
AR 10	00 to 15	<b>Power-off Counter</b> (4 digits BCD) This is the count of the number of times that the power has been turned off. To clear the count, write "0000" from a peripheral device.	---
AR 11	00 to 07	<b>High-speed Counter Range Comparison Flags</b> 00 ON: Counter PV is within comparison range 1 01 ON: Counter PV is within comparison range 2 02 ON: Counter PV is within comparison range 3 03 ON: Counter PV is within comparison range 4 04 ON: Counter PV is within comparison range 5 05 ON: Counter PV is within comparison range 6 06 ON: Counter PV is within comparison range 7 07 ON: Counter PV is within comparison range 8	51
	08 to 14	Not used.	
	15	<b>Pulse Output Status</b> ON: Stopped. OFF: Pulses being output.	---

Word(s)	Bit(s)	Function	Page
AR 12	00 to 15	Not used.	
AR 13	00	<b>Power-up PC Setup Error Flag</b> Turns ON when there is an error in DM 6600 to DM 6614 (the part of the PC Setup area that is read at power-up).	3
	01	<b>Start-up PC Setup Error Flag</b> Turns ON when there is an error in DM 6615 to DM 6644 (the part of the PC Setup area that is read at the beginning of operation).	
	02	<b>RUN PC Setup Error Flag</b> Turns ON when there is an error in DM 6645 to DM 6655 (the part of the PC Setup area that is always read).	
	03, 04	Not used.	
	05	<b>Long Cycle Time Flag</b> Turns ON if the actual cycle time is longer than the cycle time set in DM 6619.	---
	06, 07	Not used.	
	08	<b>Memory Area Specification Error Flag</b> Turns ON when a non-existent data area address is specified in the program.	---
	09	<b>Flash Memory Error Flag</b> Turns ON when there is an error in flash memory.	---
	10	<b>Read-only DM Error Flag</b> (See note 3.) Turns ON when a checksum error occurs in the read-only DM (DM 6144 to DM 6599) and that area is initialized.	3
	11	<b>PC Setup Error Flag</b> Turns ON when a checksum error occurs in the PC Setup area.	
	12	<b>Program Error Flag</b> Turns ON when a checksum error occurs in the program memory (UM) area, or when an improper instruction is executed.	---
	13	Not used.	
	14	<b>Data Save Error Flag</b> Turns ON when power is turned on if data could not be saved with the built-in capacitor. Data is saved in the following areas with the built-in capacitor: DM area (Read/write-capable: DM 0000 to 0999 and DM 1022 to 1023) HR area (HR 00 to 19) Counter area (CNT 000 to 127) SR area, word 252, bits 11, 12 (when PC Setup in DM 6601 is set to maintain status) AR area, word 10 (power-off counter) Operation mode (when PC Setup in DM 6600 is set to continue mode last used before power failure)  If data could not be saved in the above areas: The DM, error log, HR, counter, SR (word 252, bits 11 and 12), and AR (word 10) areas will be cleared. The operation mode will go into Program Mode.  (For details regarding the holding time, refer to the <i>CPM1A Operation Manual</i> .)	---
	15	Not used.	
AR 14	00 to 15	<b>Maximum Cycle Time</b> (4 digits BCD) (See note 1.) The longest cycle time since the beginning of operation is stored. It is cleared at the beginning, and not at the end, of operation.  The units can be any of the following, depending on the setting of in DM 6618. Default: 0.1 ms; "10 ms" setting: 0.1 ms; "100 ms" setting: 1 ms; "1 s" setting: 10 ms	21
AR 15	00 to 15	<b>Current Cycle Time</b> (4 digits BCD) (See note 1.) The most recent cycle time during operation is stored. The Current Cycle Time is not cleared when operation stops.  The units can be any of the following, depending on the setting of in DM 6618. Default: 0.1 ms; "10 ms" setting: 0.1 ms; "100 ms" setting: 1 ms; "1 s" setting: 10 ms	

**Note** 1. The units will be as follows, depending on the unit setting for the cycle monitor time (DM 6618):

Initial status:	0.1-ms unit
When 10-ms unit is set:	0.1-ms unit
When 100-ms unit is set:	1-ms unit
When 1-s unit is set:	10-ms unit



2. Areas that cannot be used are cleared when the power is turned on.
3. The contents of AR 10 is backed up by the built-in capacitor. If power is left OFF for longer than the back-up time, however, the contents may be cleared. For details regarding the backup time, refer to the *CPM1A* or *CPM1 Operation Manual*.

## SRM1 Memory Areas

### Memory Area Structure

The following memory areas can be used with the SRM1.

Data area		Words	Bits	Function
IR area <sup>1</sup>	Input area	IR 000 to IR 009 (10 words)	IR 00000 to IR 00915 (160 bits)	These bits can be allocated to the external I/O terminals. The ON/OFF status of the I/O bits will be the same as the ON/OFF status of the I/O terminals  Any I/O bits not used for I/O can be used as work bits.
	Output area	IR 010 to IR 019 (10 words)	IR 01000 to IR 01915 (160 bits)	
	Work area	IR 200 to IR 239 (40 words)	IR 20000 to IR 23315 (640 bits)	Work bits can be freely used within the program. IR 232 to IR 239 however, cannot be used as the the MACRO input area for the MACRO instruction.
SR area		SR 240 to SR 255 (16 words)	SR 24000 to SR 25507 (248 bits)	These bits serve as storage space for flags and function set values/present values for SRM1 operation. Refer to <i>SR Area</i> .
TR area		---	TR 0 to TR 7 (8 bits)	When a complicated ladder diagram cannot be recorded as a mnemonic these bits are used to temporarily store ON/OFF status at program branches. These temporary bits cannot be used within the same block but if the blocks are different several may be used. The ON/OFF status of these bits cannot be monitored using the monitoring function of a peripheral device.
HR area <sup>2</sup>		HR 00 to HR 19 (20 words)	HR 0000 to HR 1915 (320 bits)	These bits store data and retain their ON/OFF status when power is turned off, or operation starts or stops. They are used in the same way as work bits.
AR area <sup>2</sup>		AR 00 to AR 15 (16 words)	AR 0000 to AR 1515 (256 bits)	These bits serve specific functions such as flags and control bits. AR 04 to 07 are used as slaves. Refer to <i>AR Area</i> .
LR area <sup>1</sup>		LR 00 to LR 15 (16 words)	LR 0000 to LR 1515 (256 bits)	Used for a 1:1 data link with another SRM1, CQM1 or C200HS PC.
Timer/Counter area <sup>2</sup>		TC 000 to TC 127 (timer/counter numbers) <sup>3</sup>		Timers and counter use the TIM, TIMH(15), CNT and CNTR(12) instructions. The same numbers are used for both timers and counters.  Timer/counter numbers should be specified as bits when dealing with timer/counter present values. The counter data will be stored even when the SRM1 power is turned off or operation is stopped or started.  When timer/counter are treated as up-flags the number should be specified as relay data.

Data area		Words	Bits	Function
DM area	Read/write <sup>2</sup>	DM 0000 to DM 1999 (2,000 words)	---	DM area data can be accessed in word units only. Word values are retained when the power is turned off, or operation started or stopped.  Read/write areas can be read and written freely within the program.
	Error log <sup>4</sup>	DM 2000 to DM 2021 (22 words)	---	Used to store the time of occurrence and error code of errors that occur. Refer to 5-5 <i>Coding Right-hand Instructions</i> .
	Read-only <sup>4</sup>	DM 6144 to DM 6599 (456 words)	---	Cannot be overwritten from program.
	PC Setup <sup>4</sup>	DM 6600 to DM 6655 (56 words)	---	Used to store various parameters that control PC operation.

- Note**
1. IR and LR bits that are not used for their allocated functions can be used as work bits.
  2. The contents of the HR area, LR area, Counter area, and read/write DM area are backed up by a capacitor. At 25°C, the capacitor will back up memory for 20 days. Refer to 2-1-2 *Characteristics* in the *SRM1 Master Control Unit Operation Manual* for a graph showing the backup time vs. temperature.
  3. When accessing a PV, TC numbers are used as word data; when accessing Completion Flags, they are used as bit data.
  4. Data in DM 6144 to DM 6655 cannot be overwritten from the program, but they can be changed from a Peripheral Device.

## SR Area

These bits mainly serve as flags related to SRM1 operation or contain present and set values for various functions. The functions of the SR area are explained in the following table.

Word(s)	Bit(s)	Function	Page
SR 240 to SR247	00 to 15	Not used. Can be used as work bits.	
SR 248, SR249	00 to 15	Reserved.	
SR 250, SR251	00 to 15	Not used. Can be used as work bits.	
SR 252	00	Not used. (system use)	
	01 to 07	Not used.	
	08	<b>Peripheral Port Reset Bit</b> Turn ON to reset peripheral port. (Not valid when peripheral device is connected.) Automatically turns OFF when reset is complete.	92
	09	<b>RS-232C Port Reset Bit</b> Automatically turns OFF when reset is complete.	
	10	<b>PC Setup Reset Bit</b> Turn ON to initialize PC Setup (DM 6600 through DM 6655). Automatically turns OFF again when reset is complete. Only effective if the PC is in PROGRAM mode.	3
	11	<b>Forced Status Hold Bit</b> OFF: The forced status of bits that are forced set/reset is cleared when switching between PROGRAM mode and MONITOR mode. ON: The status of bits that are forced set/reset are maintained when switching between PROGRAM mode and MONITOR mode.	17
	12	<b>I/O Hold Bit</b> OFF: IR and LR bits are reset when starting or stopping operation. ON: IR and LR bit status is maintained when starting or stopping operation.	17
	13	Not used.	
	14	<b>Error Log Reset Bit</b> Turn ON to clear error log. Automatically turns OFF again when operation is complete.	431
	15	Not used.	

Word(s)	Bit(s)	Function	Page
SR 253	00 to 07	<b>FAL Error Code</b> The error code (a 2-digit number) is stored here when an error occurs. The FAL number is stored here when FAL(06) or FALS(07) is executed. This word is reset (to 00) by executing a FAL 00 instruction or by clearing the error from a Peripheral Device.	205
	08	Not used.	
	09	<b>Cycle Time Overrun Flag</b> Turns ON when a cycle time overrun occurs.	---
	10 to 11	Not used.	
	12	<b>RS-232C Port Set Bit</b> Turn ON to set RS-232C port. Turn OFF when reset is complete.	
	13	<b>Always ON Flag</b>	---
	14	<b>Always OFF Flag</b>	---
	15	<b>First Cycle Flag</b> Turns ON for 1 cycle at the start of operation.	---
SR 254	00	1-minute clock pulse (30 seconds ON; 30 seconds OFF)	---
	01	0.02-second clock pulse (0.01 second ON; 0.01 second OFF)	---
	02	<b>Negative (N) Flag</b>	---
	03	Not used.	
	04	<b>Overflow Flag</b>	---
	05	<b>Underflow Flag</b>	---
	06	<b>Differential Monitor Complete Flag</b> Turns ON when differential monitoring is complete.	134
	07	<b>STEP(08) Execution Flag</b> Turns ON for 1 cycle only at the start of process based on STEP(08).	206
	08 to 15	Not used.	
SR 255	00	0.1-second clock pulse (0.05 second ON; 0.05 second OFF)	---
	01	0.2-second clock pulse (0.1 second ON; 0.1 second OFF)	---
	02	1.0-second clock pulse (0.5 second ON; 0.5 second OFF)	---
	03	<b>Instruction Execution Error (ER) Flag</b> Turns ON when an error occurs during execution of an instruction.	---
	04	<b>Carry (CY) Flag</b> Turns ON when there is a carry in the results of an instruction execution.	---
	05	<b>Greater Than (GR) Flag</b> Turns ON when the result of a comparison operation is "greater."	---
	06	<b>Equals (EQ) Flag</b> Turns ON when the result of a comparison operation is "equal," or when the result of an instruction execution is 0.	---
	07	<b>Less Than (LE) Flag</b> Turns ON when the result of a comparison operation is "less."	---
	08 to 15	Not used.	

## AR Area

These bits mainly serve as flags related to SRM1 operation. These bits retain their status even after the SRM1 power supply has been turned off or when operation begins or stops.

Word(s)	Bit(s)	Function	Page
AR 00, AR 01	00 to 15	Not used.	
AR 02	00 to 07	Not used.	
	08 to 11	Not used. (system use)	
	12 to 15	Not used.	
AR 03	00 to 15	Not used.	
AR 04 to AR 07	00 to 15	<b>Slave Status Flag</b>	---

Word(s)	Bit(s)	Function	Page
AR 08	00 to 03	<b>RS-232C Error Code</b> (1-digit number) 0: Normal completion 1: Parity error 2: Framing error 3: Overrun error	---
	04	<b>RS-232C Communications Error</b>	---
	05	<b>RS-232C Transmission Enabled Flag</b> Valid only when host link, no protocol communications are used.	---
	06	<b>RS-232C Reception Completed Flag</b> Valid only when no protocol communications are used.	---
	07	<b>RS-232C Reception Overflow Flag</b> Valid only when no protocol communications are used.	---
	08 to 11	<b>Peripheral Device Error Code</b> 0: Normal completion 1: Parity error 2: Frame error 3: Overrun error	99
	12	<b>Peripheral Device Error Flag</b>	
	13	<b>Peripheral Device Transmission Enabled Flag</b> Valid only when host link, no protocol communications are used.	
	14	<b>Peripheral Device Reception Completed Flag</b> Valid only when no protocol communications are used.	
	15	<b>Peripheral Device Reception Overflow Flag</b> Valid only when no protocol communications are used.	
AR 09	00 to 15	<b>RS-232C Reception Counter</b> (4 digits BCD) Valid only when no protocol communications are used.	---
AR 10	00 to 15	<b>Peripheral Device Reception Counter</b> (4 digits BCD) Valid only when no protocol communications are used.	---
AR 11	00 to 15	4 digits BCD Power supply cut frequency.	---
AR 12	00 to 15	Not used.	
AR 13	00	<b>Power-up PC Setup Error Flag</b> Turns ON when there is an error in DM 6600 to DM 6614 (the part of the PC Setup area that is read at power-up).	3
	01	<b>Start-up PC Setup Error Flag</b> Turns ON when there is an error in DM 6615 to DM 6644 (the part of the PC Setup area that is read at the beginning of operation).	
	02	<b>RUN PC Setup Error Flag</b> Turns ON when there is an error in DM 6645 to DM 6655 (the part of the PC Setup area that is always read).	
	03, 04	Not used.	
	05	<b>Long Cycle Time Flag</b> Turns ON if the actual cycle time is longer than the cycle time set in DM 6619.	---
	06	Turns ON when the program memory (UM) area is full.	---
	07	Turns ON when instructions other than those in the support software area used.	---
	08	<b>Memory Area Specification Error Flag</b> Turns ON when a non-existent data area address is specified in the program.	---
	09	<b>Flash Memory Error Flag</b> Turns ON when there is an error in flash memory.	---
	10	<b>Read-only DM Error Flag</b> Turns ON when a checksum error occurs in the read-only DM (DM 6144 to DM 6599) and that area is initialized.	3
	11	<b>PC Setup Error Flag</b> Turns ON when a checksum error occurs in the PC Setup area.	
	12	<b>Program Error Flag</b> Turns ON when a checksum error occurs in the program memory (UM) area, or when an improper instruction is executed.	---
	13	Not used. (Cleared when power is turned on.)	

Word(s)	Bit(s)	Function	Page
AR 13	14	<p><b>Data Save Error Flag</b></p> <p>Turns ON when power is turned on if data could not be saved in the following areas: DM area (read/write-capable), HR area, CNT area, SR 252, bits 11, 12 (when PC Setup in DM 6601 is set to maintain status), error log, operation mode (when PC Setup in DM 6600 is set to continue mode last used before power failure).</p> <p>(For details regarding the holding time, refer to the <i>SRM1 Operation Manual</i>.)</p> <p>If data could not be saved in the above areas: The DM (read/write-capable), error log, HR, and CNT areas, and SR 252, bits 11 and 12 will be cleared. The operation mode will go into Program Mode.</p>	
	15	<p><b>SRM1 CompoBus/S Communications Error Flag</b></p>	
AR 14	00 to 15	<p><b>Maximum Cycle Time</b> (4 digits BCD)</p> <p>The longest cycle time since the beginning of operation is stored. It is cleared at the beginning, and not at the end, of operation.</p> <p>The units can be any of the following, depending on the setting of in DM 6618. Default: 0.1 ms; "10 ms" setting: 0.1 ms; "100 ms" setting: 1 ms; "1 s" setting: 10 ms</p>	21
AR 15	00 to 15	<p><b>Current Cycle Time</b> (4 digits BCD)</p> <p>The most recent cycle time during operation is stored. The Current Cycle Time is not cleared when operation stops.</p> <p>The units can be any of the following, depending on the setting of in DM 6618. Default: 0.1 ms; "10 ms" setting: 0.1 ms; "100 ms" setting: 1 ms; "1 s" setting: 10 ms</p>	

# Appendix D

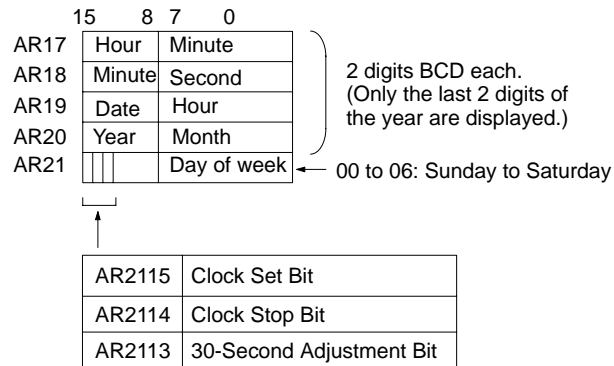
## Using the Clock Function

The CQM1 can be provided with a clock function by installing a Memory Cassette equipped with a clock. This appendix explains how to use the clock.

The following two Memory Cassettes are equipped with a clock:  
 CQM1-ME04R (EPROM) and CQM1-ME08R (EEPROM).

### Data Area Words Used with the Clock

This following illustration shows the configuration of the words (AR 17 through AR 21) that are used with the clock. These words can be read and used as required. AR 17 is provided so that the hour and minute can be accessed quickly.



### Setting the Time

To set the time, use a peripheral device as follows:

#### When Setting Everything

- 1, 2, 3...**
1. Turn ON AR 2114 (Clock Stop Bit) to stop the clock.
  2. Using a peripheral device, set AR 18 through AR 20 (minute/second, date/hour, and year/month) and AR 2100 through AR 2107 (day of week).
  3. Turn ON AR 2115 (Clock Set Bit) when the time set in step 2 is reached. The clock will start operating from the time that is set, and the Clock Stop Bit will turn OFF automatically. When the time setting is complete, AR 2115 will also turn OFF automatically.

#### When Setting Only the Seconds

It is also possible, by using AR 2113, to simply set the seconds to "00" without going through a complicated procedure. When AR 2113 is turned ON, the clock time will change as follows:

If the seconds setting is from 00 to 29, the seconds will be reset to "00" and the minute setting will remain the same. If the seconds setting is from 30 to 59, the seconds will be reset to "00" and the minute setting will advance by one. When the time setting is complete, AR 2113 will turn OFF automatically.

**Note** The time can be set easily using menu operations from a Programming Console or SSS. Refer to the *CQM1 Operation Manual* for the Programming Console procedure or to the *SSS Operation Manual: C-series PCs* for the SSS procedure.

**Caution** The clock will stop and the current date and time information will be lost if the Memory Cassette is removed from the PC.

# Appendix E

## I/O Assignment Sheet

Name of system	Produced by	Verified by	Authorized by
PC model	Sheet no.		

IR _____	Unit no.:	Model:	IR _____	Unit no.:	Model:
00			00		
01			01		
02			02		
03			03		
04			04		
05			05		
06			06		
07			07		
08			08		
09			09		
10			10		
11			11		
12			12		
13			13		
14			14		
15			15		
IR _____	Unit no.:	Model:	IR _____	Unit no.:	Model:
00			00		
01			01		
02			02		
03			03		
04			04		
05			05		
06			06		
07			07		
08			08		
09			09		
10			10		
11			11		
12			12		
13			13		
14			14		
15			15		

# Appendix F

## Program Coding Sheet

Name of system		Produced by	Verified by	Authorized by
PC	Chart no.			

Address					Instruction	Function code	Operands		
			0	0					
			0	1					
			0	2					
			0	3					
			0	4					
			0	5					
			0	6					
			0	7					
			0	8					
			0	9					
			1	0					
			1	1					
			1	2					
			1	3					
			1	4					
			1	5					
			1	6					
			1	7					
			1	8					
			1	9					
			2	0					
			2	1					
			2	2					
			2	3					
			2	4					
			2	5					
			2	6					
			2	7					
			2	8					
			2	9					
			3	0					
			3	1					
			3	2					
			3	3					



Address					Instruction	Function code	Operands		
			3	4					
			3	5					
			3	6					
			3	7					
			3	8					
			3	9					
			4	0					
			4	1					
			4	2					
			4	3					
			4	4					
			4	5					
			4	6					
			4	7					
			4	8					
			4	9					
			5	0					
			5	1					
			5	2					
			5	3					
			5	4					
			5	5					
			5	6					
			5	7					
			5	8					
			5	9					
			6	0					
			6	1					
			6	2					
			6	3					
			6	4					
			6	5					
			6	6					
			6	7					
			6	8					
			6	9					
			7	0					
			7	1					
			7	2					

Address					Instruction	Function code	Operands		
			7	3					
			7	4					
			7	5					
			7	6					
			7	7					
			7	8					
			7	9					
			8	0					
			8	1					
			8	2					
			8	3					
			8	4					
			8	5					
			8	6					
			8	7					
			8	8					
			8	9					
			9	0					
			9	1					
			9	2					
			9	3					
			9	4					
			9	5					
			9	6					
			9	7					
			9	8					
			9	9					

# Appendix G

## List of FAL Numbers

Name of system		Produced by	Verified by	Authorized by
PC model	Chart no.			

FAL No.	FAL contents	Corrective measure	FAL No.	FAL contents	Corrective measure
00			35		
01			36		
02			37		
03			38		
04			39		
05			40		
06			41		
07			42		
08			43		
09			44		
10			45		
11			46		
12			47		
13			48		
14			49		
15			50		
16			51		
17			52		
18			53		
19			54		
20			55		
21			56		
22			57		
23			58		
24			59		
25			60		
26			61		
27			62		
28			63		
29			64		
30			65		
31			66		
32			67		
33			68		
34			69		

FAL No.	FAL contents	Corrective measure	FAL No.	FAL contents	Corrective measure
70			85		
71			86		
72			87		
73			88		
74			89		
75			90		
76			91		
77			92		
78			93		
79			94		
80			95		
81			96		
82			96		
83			97		
84			99		

# Appendix H

## Extended ASCII

The following codes are used to output characters to the Programming Console or Data Access Console using MSG(46) or FPD(—). Refer to pages 317 and 322 for details.

Right digit	Left digit												
	0, 1, 8, 9	2	3	4	5	6	7	A	B	C	D	E	F
0			0	Q	P	`	P		-	Q	P	`	P
1		!	1	A	Q	a	q	!	1	A	Q	a	q
2		"	2	B	R	b	r	"	2	B	R	b	r
3		#	3	C	S	c	s	#	3	C	S	c	s
4		\$	4	D	T	d	t	\$	4	D	T	d	t
5		%	5	E	U	e	u	%	5	E	U	e	u
6		&	6	F	V	f	v	&	6	F	V	f	v
7		'	7	G	W	g	w	'	7	G	W	g	w
8		(	8	H	X	h	x	(	8	H	X	h	x
9		)	9	I	Y	i	y	)	9	I	Y	i	y
A		*	:	J	Z	j	z	*	:	J	Z	j	z
B		+	:	K	[	k	[	+	:	K	[	k	[
C		,	<	L	¥	l	l	,	<	L	¥	l	l
D		-	=	M	]	m	]	-	=	M	]	m	]
E		.	>	N	^	n	+	.	>	N	^	n	
F		/	?	O	_	o	+	/	?	O	_	o	+

# Appendix I

## CPM1/CPM1A and CQM1 Memory Area Comparison

This table shows the differences between the CPM1/CPM1A and CQM1 memory areas.

Data area		CPM1/CPM1A	CQM1
IR area	Input area	IR 000 to IR 009	IR 000 to IR 015
	Output area	IR 010 to IR 019	IR 100 to IR 115
	Work areas and special areas	IR 200 to IR 231 (IR 020 to IR 199 cannot be used.)	IR 016 to IR 099 IR 116 to IR 144
SR area		SR 232 to SR 255	SR 244 to SR 255
HR area		HR 00 to HR 19 (HR 20 to HR 99 cannot be used.)	HR 00 to HR 99
AR area		AR 00 to AR 15 (AR 16 to AR 27 cannot be used.)	AR 00 to AR 27
LR area		LR 00 to LR 15 (LR 16 to LR 63 cannot be used.)	LR 00 to LR 63
Timer/Counter area		TC 000 to TC 127 (TC 128 to TC 511 cannot be used.)	TC 000 to TC 511
DM area	Read/write	DM 0000 to DM 0999 DM 1022 to DM 1023 (DM 1024 to DM 6143 cannot be used.)	DM 0000 to DM 1023 DM 1024 to DM 6143
	Error history	DM 1000 to DM 1021	DM 6569 to DM 6599
	Read-only	DM 6144 to DM 6599	DM 6144 to DM 6568
	PC Setup	DM 6600 to DM 6655	DM 6600 to DM 6655

# Glossary

<b>*DM</b>	Indirectly addressed DM area. See <i>indirect address</i> and <i>DM area</i> .
<b>1:1 link</b>	A link created between two PCs to create <i>common data</i> in their LR areas.
<b>ACP</b>	See <i>add count input</i> .
<b>add count input</b>	An input signal used to increment a counter when the signal changes from OFF to ON.
<b>address</b>	A number used to identify the location of data or programming instructions in memory.
<b>AND</b>	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.
<b>area</b>	See <i>data area</i> and <i>memory area</i> .
<b>area prefix</b>	A one or two letter prefix used to identify a memory area in the PC. All memory areas except the IR and SR areas require prefixes to identify addresses in them.
<b>arithmetic shift</b>	A shift operation wherein the carry flag is included in the shift.
<b>ASCII</b>	Short for American Standard Code for Information Interchange. ASCII is used to code characters for output to printers and other external devices.
<b>AR Area</b>	A PC data area allocated to flags and control bits.
<b>AUTOEXEC.BAT</b>	An MS DOS file containing commands automatically executed at startup.
<b>back-up</b>	A copy made of existing data to ensure that the data will not be lost even if the original data is corrupted or erased.
<b>basic instruction</b>	A fundamental instruction used in a ladder diagram. See <i>advanced instruction</i> .
<b>baud rate</b>	The data transmission speed between two devices in a system measured in bits per second.
<b>BCD</b>	See <i>binary-coded decimal</i> .
<b>BCD calculation</b>	An arithmetic calculation that uses numbers expressed in binary-coded decimal.
<b>binary</b>	A number system where all numbers are expressed in base 2, i.e., numbers are written using only 0's and 1's. Each group of four binary bits is equivalent to one hexadecimal digit. Binary data in memory is thus often expressed in hexadecimal for convenience.
<b>binary calculation</b>	An arithmetic calculation that uses numbers expressed in binary.
<b>binary-coded decimal</b>	A system used to represent numbers so that every four binary bits is numerically equivalent to one decimal digit.
<b>bit</b>	The smallest piece of information that can be represented on a computer. A bit has the value of either zero or one, corresponding to the electrical signals ON and OFF. A bit represents one binary digit. Some bits at particular addresses are allocated to special purposes, such as holding the status of input from external devices, while other bits are available for general use in programming.
<b>bit address</b>	The location in memory where a bit of data is stored. A bit address specifies the data area and word that is being addressed as well as the number of the bit within the word.

<b>bit designator</b>	An operand that is used to designate the bit or bits of a word to be used by an instruction.
<b>bit number</b>	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.
<b>bit-control instruction</b>	An instruction that is used to control the status of an individual bit as opposed to the status of an entire word.
<b>block</b>	See <i>logic block</i> and <i>instruction block</i> .
<b>building-block PC</b>	A PC that is constructed from individual components, or "building blocks." With building-block PCs, there is no one Unit that is independently identifiable as a PC. The PC is rather a functional assembly of Units.
<b>bus</b>	A communications path used to pass data between any of the Units connected to it.
<b>bus bar</b>	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.
<b>byte</b>	A unit of data equivalent to 8 bits, i.e., half a word.
<b>call</b>	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.
<b>Carry Flag</b>	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.
<b>central processing unit</b>	A device that is capable of storing programs and data, and executing the instructions contained in the programs. In a PC System, the central processing unit executes the program, processes I/O signals, communicates with external devices, etc.
<b>CH</b>	See <i>word</i> .
<b>channel</b>	See <i>word</i> .
<b>character code</b>	A numeric (usually binary) code used to represent an alphanumeric character.
<b>checksum</b>	A sum transmitted with a data pack in communications. The checksum can be recalculated from the received data to confirm that the data in the transmission has not been corrupted.
<b>clock pulse</b>	A pulse available at specific bits in memory for use in timing operations. Various clock pulses are available with different pulse widths, and therefore different frequencies.
<b>clock pulse bit</b>	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.
<b>common data</b>	Data that is stored in a memory of a PC and which is shared by other PCs in the same the same system. Each PC has a specified section(s) of the area allocated to it. Each PC writes to the section(s) allocated to it and reads the sections allocated to the other PCs with which it shares the common data.
<b>communications cable</b>	Cable used to transfer data between components of a control system and conforming to the RS-232C or RS-422 standards.
<b>comparison instruction</b>	An instruction used to compare data at different locations in memory to determine the relationship between the data.



<b>Completion Flag</b>	A flag used with a timer or counter that turns ON when the timer has timed out or the counter has reached its set value.
<b>condition</b>	A symbol placed on an instruction line to indicate an instruction that controls the execution condition for the terminal instruction. Each condition is assigned a bit in memory that determines its status. The status of the bit assigned to each condition determines the next execution condition. Conditions correspond to LOAD, LOAD NOT, AND, AND NOT, OR, or OR NOT instructions.
<b>CONFIG.SYS</b>	An MS DOS file containing environment settings for a personal computer.
<b>constant</b>	An input for an operand in which the actual numeric value is specified. Constants can be input for certain operands in place of memory area addresses. Some operands must be input as constants.
<b>control bit</b>	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.
<b>control data</b>	An operand that specifies how an instruction is to be executed. The control data may specify the part of a word is to be used as the operand, it may specify the destination for a data transfer instructions, it may specify the size of a data table used in an instruction, etc.
<b>control signal</b>	A signal sent from the PC to effect the operation of the controlled system.
<b>Control System</b>	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.
<b>controlled system</b>	The devices that are being controlled by a PC System.
<b>count pulse</b>	The signal counted by a counter.
<b>counter</b>	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 TIM/CNT bit and used to count the number of times the status of a bit or an execution condition has changed from OFF to ON.
<b>CPU</b>	See <i>central processing unit</i> .
<b>CTS</b>	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.
<b>CY</b>	See <i>Carry Flag</i> .
<b>cycle</b>	One unit of processing performed by the CPU, including ladder program execution, peripheral servicing, I/O refreshing, etc.
<b>cycle time</b>	The time required to complete one cycle of CPU processing.
<b>cyclic interrupt</b>	See <i>scheduled interrupt</i> .
<b>data area</b>	An area in the PC's memory that is designed to hold a specific type of data.
<b>data area boundary</b>	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.
<b>data disk</b>	A floppy disk used to save user programs, DM area contents, comments, and other user data.
<b>data length</b>	In communications, the number of bits that is to be treated as one unit in data transmissions.

<b>data link</b>	An automatic data transmission operation that allows PCs or Units within PC to pass data back and forth via common data areas.
<b>data link area</b>	A common data area established through a data link.
<b>data movement instruction</b>	An instruction used to move data from one location in memory to another. The data in the original memory location is left unchanged.
<b>data sharing</b>	The process in which common data areas or common data words are created between two or more PCs.
<b>data trace</b>	A process in which changes in the contents of specific memory locations are recorded during program execution.
<b>data transfer</b>	Moving data from one memory location to another, either within the same device or between different devices connected via a communications line or network.
<b>debug</b>	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.
<b>decimal</b>	A number system where numbers are expressed to the base 10. In a PC all data is ultimately stored in binary form, four binary bits are often used to represent one decimal digit, via a system called binary-coded decimal.
<b>decrement</b>	Decreasing a numeric value, usually by 1.
<b>default</b>	A value automatically set by the PC when the user does not specifically set another value. Many devices will assume such default conditions upon the application of power.
<b>definer</b>	A number used as an operand for an instruction but that serves to define the instruction itself, rather than the data on which the instruction is to operate. Definers include jump numbers, subroutine numbers, etc.
<b>destination</b>	The location where an instruction places the data on which it is operating, as opposed to the location from which data is taken for use in the instruction. The location from which data is taken is called the source.
<b>differentiated instruction</b>	An instruction that is executed only once each time its execution condition goes from OFF to ON. Non-differentiated instructions are executed for each scan as long as the execution condition stays ON.
<b>differentiation instruction</b>	An instruction used to ensure that the operand bit is never turned ON for more than one scan after the execution condition goes either from OFF to ON for a Differentiate Up instruction or from ON to OFF for a Differentiate Down instruction.
<b>digit</b>	A unit of storage in memory that consists of four bits.
<b>digit designator</b>	An operand that is used to designate the digit or digits of a word to be used by an instruction.
<b>DIN track</b>	A rail designed to fit into grooves on various devices to allow the devices to be quickly and easily mounted to it.
<b>DIP switch</b>	Dual in-line package switch, an array of pins in a signal package that is mounted to a circuit board and is used to set operating parameters.
<b>direct output</b>	A method in which program execution results are output immediately to eliminate the affects of the cycle time.
<b>distributed control</b>	A automation concept in which control of each portion of an automated system is located near the devices actually being controlled, i.e., control is decentralized

	and 'distributed' over the system. Distributed control is a concept basic to PC Systems.
<b>DM area</b>	A data area used to hold only word data. Words in the DM area cannot be accessed bit by bit.
<b>DM word</b>	A word in the DM area.
<b>downloading</b>	The process of transferring a program or data from a higher-level or host computer to a lower-level or slave computer. If a Programming Device is involved, the Programming Device is considered the host computer.
<b>EEPROM</b>	Electrically erasable programmable read-only memory; a type of ROM in which stored data can be erased and reprogrammed. This is accomplished using a special control lead connected to the EEPROM chip and can be done without having to remove the EEPROM chip from the device in which it is mounted.
<b>electrical noise</b>	Random variations of one or more electrical characteristics such as voltage, current, and data, which might interfere with the normal operation of a device.
<b>EPROM</b>	Erasable programmable read-only memory; a type of ROM in which stored data can be erased, by ultraviolet light or other means, and reprogrammed.
<b>error code</b>	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.
<b>Error Log Area</b>	An area used to store records indicating the time and nature of errors that have occurred in the system.
<b>even parity</b>	A communication setting that adjusts the number of ON bits so that it is always even. See <i>parity</i> .
<b>event processing</b>	Processing that is performed in response to an event, e.g., an interrupt signal.
<b>exclusive NOR</b>	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.
<b>exclusive OR</b>	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.
<b>execution condition</b>	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.
<b>execution cycle</b>	The cycle used to execute all processes required by the CPU, including program execution, I/O refreshing, peripheral servicing, etc.
<b>execution time</b>	The time required for the CPU to execute either an individual instruction or an entire program.
<b>extended counter</b>	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.
<b>extended timer</b>	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.

<b>FA</b>	Factory automation.
<b>factory computer</b>	A general-purpose computer, usually quite similar to a business computer, that is used in automated factory control.
<b>FAL error</b>	An error generated from the user program by execution of an FAL(06) instruction.
<b>FALS error</b>	An error generated from the user program by execution of an FALS(07) instruction or an error generated by the system.
<b>fatal error</b>	An error that stops PC operation and requires correction before operation can continue.
<b>FCS</b>	See <i>frame checksum</i> .
<b>flag</b>	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.
<b>flicker bit</b>	A bit that is programmed to turn ON and OFF at a specific frequency.
<b>floating-point decimal</b>	A decimal number expressed as a number (the mantissa) multiplied by a power of 10, e.g., $0.538 \times 10^{-5}$ .
<b>force reset</b>	The process of forcibly turning OFF a bit via a programming device. Bits are usually turned OFF as a result of program execution.
<b>force set</b>	The process of forcibly turning ON a bit via a programming device. Bits are usually turned ON as a result of program execution.
<b>forced status</b>	The status of bits that have been force reset or force set.
<b>frame checksum</b>	The results of exclusive ORing all data within a specified calculation range. The frame checksum can be calculated on both the sending and receiving end of a data transfer to confirm that data was transmitted correctly.
<b>function code</b>	A two-digit number used to input an instruction into the PC.
<b>hardware error</b>	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).
<b>header code</b>	A code in an instruction that specifies what the instruction is to do.
<b>hexadecimal</b>	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.
<b>host computer</b>	A computer that is used to transfer data to or receive data from a PC in a Host Link system. The host computer is used for data management and overall system control. Host computers are generally small personal or business computers.
<b>host interface</b>	An interface that allows communications with a host computer.
<b>host link</b>	An interface connecting a PC to a host computer to enable monitoring or program control from the host computer.
<b>HR area</b>	A memory area that preserves bit status during power interrupts and used as work bits in programming.
<b>I/O bit</b>	A bit in memory used to hold I/O status. Input bits reflect the status of input terminals; output bits hold the status for output terminals.

<b>I/O capacity</b>	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.
<b>I/O delay</b>	The delay in time from when a signal is sent to an output to when the status of the output is actually in effect or the delay in time from when the status of an input changes until the signal indicating the change in the status is received.
<b>I/O device</b>	A device connected to the I/O terminals on I/O Units. 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.
<b>I/O interrupt</b>	An interrupt generated by a signal from I/O.
<b>I/O point</b>	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.
<b>I/O refreshing</b>	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.
<b>I/O response time</b>	The time required for an output signal to be sent from the PC in response to an input signal received from an external device.
<b>I/O Unit</b>	The Units in a PC that are physically connected to I/O devices to input and output signals. I/O Units include Input Units and Output Units, each of which is available in a range of specifications.
<b>I/O word</b>	A word in the IR area that is allocated to a Unit in the PC System and is used to hold I/O status for that Unit.
<b>IBM PC/AT or compatible</b>	A computer that has similar architecture to, that is logically compatible with, and that can run software designed for an IBM PC/AT computer.
<b>increment</b>	Increasing a numeric value, usually by 1.
<b>indirect address</b>	An address whose contents indicates another address. The contents of the second address will be used as the actual operand.
<b>initialization error</b>	An error that occurs either in hardware or software during the PC System startup, i.e., during initialization.
<b>initialize</b>	Part of the startup process whereby some memory areas are cleared, system setup is checked, and default values are set.
<b>input</b>	The signal coming from an external device into the PC. The term input is often used abstractly or collectively to refer to incoming signals.
<b>input bit</b>	A bit in the IR area that is allocated to hold the status of an input.
<b>input device</b>	An external device that sends signals into the PC System.
<b>input point</b>	The point at which an input enters the PC System. Input points correspond physically to terminals or connector pins.
<b>input signal</b>	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.
<b>install</b>	The preparation necessary to use a program or software package, such as the LSS or SSS, on a computer.

<b>instruction</b>	A direction given in the program that tells the PC of the action to be carried out, and the data to be used in carrying out the action. Instructions can be used to simply turn a bit ON or OFF, or they can perform much more complex actions, such as converting and/or transferring large blocks of data.
<b>instruction block</b>	A group of instructions that is logically related in a ladder-diagram program. A logic block includes all of the instruction lines that interconnect with each other from one or more line connecting to the left bus bar to one or more right-hand instructions connecting to the right bus bar.
<b>instruction execution time</b>	The time required to execute an instruction. The execution time for any one instruction can vary with the execution conditions for the instruction and the operands used in it.
<b>instruction line</b>	A group of conditions that lie together on the same horizontal line of a ladder diagram. Instruction lines can branch apart or join together to form instruction blocks. Also called a rung.
<b>interface</b>	An interface is the conceptual boundary between systems or devices and usually involves changes in the way the communicated data is represented. Interface devices perform operations like changing the coding, format, or speed of the data.
<b>interlock</b>	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.
<b>interrupt (signal)</b>	A signal that stops normal program execution and causes a subroutine to be run or other processing to take place.
<b>interrupt program</b>	A program that is executed in response to an interrupt.
<b>inverse condition</b>	See <i>normally closed condition</i> .
<b>JIS</b>	An acronym for Japanese Industrial Standards.
<b>jump</b>	A type of programming where execution moves directly from one point in a program to another, without sequentially executing any instructions in between.
<b>jump number</b>	A definer used with a jump that defines the points from and to which a jump is to be made.
<b>ladder diagram (program)</b>	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.
<b>ladder diagram symbol</b>	A symbol used in drawing a ladder-diagram program.
<b>ladder instruction</b>	An instruction that represents the conditions on a ladder-diagram program. The other instructions in a ladder diagram fall along the right side of the diagram and are called terminal instructions.
<b>Ladder Support Software</b>	A software package installed on a IBM PC/AT or compatible computer to function as a Programming Device.
<b>least-significant (bit/word)</b>	See <i>rightmost (bit/word)</i> .
<b>LED</b>	Acronym for light-emitting diode; a device used as for indicators or displays.
<b>leftmost (bit/word)</b>	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.

<b>link</b>	A hardware or software connection formed between two Units. "Link" can refer either to a part of the physical connection between two Units or a software connection created to data existing at another location (i.e., data links).
<b>load</b>	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.
<b>logic block</b>	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.
<b>logic block instruction</b>	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.
<b>logic instruction</b>	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.
<b>LR area</b>	A data area that is used in data links.
<b>LSS</b>	See <i>Ladder Support Software</i> .
<b>main program</b>	All of a program except for subroutine and interrupt programs.
<b>mark trace</b>	A process in which changes in the contents of specific memory locations are recorded during program execution.
<b>masked bit</b>	A bit whose status has been temporarily made ineffective.
<b>masking</b>	'Covering' an interrupt signal so that the interrupt is not effective until the mask is removed.
<b>megabyte</b>	A unit of storage equal to one million bytes.
<b>memory area</b>	Any of the areas in the PC used to hold data or programs.
<b>message number</b>	A number assigned to a message generated with the MESSAGE instruction.
<b>mnemonic code</b>	A form of a ladder-diagram program that consists of a sequential list of the instructions without using a ladder diagram.
<b>MONITOR mode</b>	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.
<b>most-significant (bit/word)</b>	See <i>leftmost (bit/word)</i> .
<b>NC input</b>	An input that is normally closed, i.e., the input signal is considered to be present when the circuit connected to the input opens.
<b>negative delay</b>	A delay set for a data trace in which recording data begins before the trace signal by a specified amount.
<b>nesting</b>	Programming one loop within another loop, programming a call to a subroutine within another subroutine, or programming one jump within another.
<b>NO input</b>	An input that is normally open, i.e., the input signal is considered to be present when the circuit connected to the input closes.
<b>noise interference</b>	Disturbances in signals caused by electrical noise.

<b>nonfatal error</b>	A hardware or software error that produces a warning but does not stop the PC from operating.
<b>normal condition</b>	See <i>normally open condition</i> .
<b>normally closed condition</b>	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.
<b>normally open condition</b>	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.
<b>NOT</b>	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.
<b>OFF</b>	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.
<b>OFF delay</b>	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).
<b>offset</b>	A positive or negative value added to a base value such as an address to specify a desired value.
<b>ON</b>	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.
<b>ON delay</b>	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).
<b>one-shot bit</b>	A bit that is turned ON or OFF for a specified interval of time which is longer than one scan.
<b>one-to-one link</b>	See <i>1:1 link</i> .
<b>online edit</b>	The process of changed the program directly in the PC from a Programming Device. Online editing is possible in PROGRAM or MONITOR mode. In MONITOR mode, the program can actually be changed while it is being
<b>operand</b>	The values designated as the data to be used for an instruction. An operand can be input as a constant expressing the actual numeric value to be used or as an address to express the location in memory of the data to be used.
<b>operand bit</b>	A bit designated as an operand for an instruction.
<b>operand word</b>	A word designated as an operand for an instruction.
<b>operating modes</b>	One of three PC modes: <i>PROGRAM mode</i> , <i>MONITOR mode</i> , and <i>RUN mode</i> .
<b>operating error</b>	An error that occurs during actual PC operation as opposed to an initialization error, which occurs before actual operations can begin.
<b>OR</b>	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.
<b>output</b>	The signal sent from the PC to an external device. The term output is often used abstractly or collectively to refer to outgoing signals.



<b>output bit</b>	A bit in the IR area that is allocated to hold the status to be sent to an output device.
<b>output device</b>	An external device that receives signals from the PC System.
<b>output point</b>	The point at which an output leaves the PC System. Output points correspond physically to terminals or connector pins.
<b>output signal</b>	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.
<b>overflow</b>	The state where the capacity of a data storage location has been exceeded.
<b>overseeing</b>	Part of the processing performed by the CPU that includes general tasks required to operate the PC.
<b>overwrite</b>	Changing the content of a memory location so that the previous content is lost.
<b>parity</b>	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.
<b>parity check</b>	Checking parity to ensure that transmitted data has not been corrupted.
<b>PC</b>	See <i>Programmable Controller</i> .
<b>PC configuration</b>	The arrangement and interconnections of the Units that are put together to form a functional PC.
<b>PC System</b>	With building-block PCs, all of the Units connected 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 at the lower end.
<b>PCB</b>	See <i>printed circuit board</i> .
<b>PC Setup</b>	A group of operating parameters set in the PC from a Programming Device to control PC operation.
<b>Peripheral Device</b>	Devices connected to a PC System to aid in system operation. Peripheral devices include printers, programming devices, external storage media, etc.
<b>peripheral servicing</b>	Processing signals to and from peripheral devices, including refreshing, communications processing, interrupts, etc.
<b>port</b>	A connector on a PC or computer that serves as a connection to an external device.
<b>positive delay</b>	A delay set for a data trace in which recording data begins after the trace signal by a specified amount.
<b>Power Supply Unit</b>	A Unit that connected to a PC that provides power at the voltage required by the other Units.
<b>present value</b>	The current value registered in a device at any instant during its operation. Present value is abbreviated as PV. The use of this term is generally restricted to timers and counters.
<b>printed circuit board</b>	A board onto which electrical circuits are printed for mounting into a computer or electrical device.
<b>PROGRAM mode</b>	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.

<b>Programmable Controller</b>	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-unit Programmable Controllers are available, building-block Programmable Controllers are constructed from separate components. Such Programmable Controllers are formed only when enough of these separate components are assembled to form a functional assembly.
<b>programmed alarm</b>	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.
<b>programmed error</b>	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.
<b>programmed message</b>	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.
<b>Programming Console</b>	The portable form of Programming Device for a PC.
<b>Programming Device</b>	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.
<b>PROM</b>	Programmable read-only memory; a type of ROM into which the program or data may be written after manufacture, by a customer, but which is fixed from that time on.
<b>prompt</b>	A message or symbol that appears on a display to request input from the operator.
<b>protocol</b>	The parameters and procedures that are standardized to enable two devices to communicate or to enable a programmer or operator to communicate with a device.
<b>PV</b>	See <i>present value</i> .
<b>RAM</b>	Random access memory; a data storage media. RAM will not retain data when power is disconnected.
<b>RAS</b>	An acronym for reliability, assurance, safety.
<b>read-only area</b>	A memory area from which the user can read status but to which data cannot be written.
<b>refresh</b>	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.
<b>relay-based control</b>	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.
<b>reserved bit</b>	A bit that is not available for user application.
<b>reserved word</b>	A word in memory that is reserved for a special purpose and cannot be accessed by the user.
<b>reset</b>	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.
<b>response code</b>	A code sent with the response to a data transmission that specifies how the transmitted data was processed.

<b>response format</b>	A format specifying the data required in a response to a data transmission.
<b>response monitoring time</b>	The time a device will wait for a response to a data transmission before assuming that an error has occurred.
<b>Restart Bit</b>	A bit used to restart part of a PC.
<b>result word</b>	A word used to hold the results from the execution of an instruction.
<b>retrieve</b>	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.
<b>retry</b>	The process whereby a device will re-transmit data which has resulted in an error message from the receiving device.
<b>return</b>	The process by which instruction execution shifts from a subroutine back to the main program (usually the point from which the subroutine was called).
<b>reversible counter</b>	A counter that can be both incremented and decremented depending on the specified conditions.
<b>reversible shift register</b>	A shift register that can shift data in either direction depending on the specified conditions.
<b>right-hand instruction</b>	See <i>terminal instruction</i> .
<b>rightmost (bit/word)</b>	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.
<b>rising edge</b>	The point where a signal actually changes from an OFF to an ON status.
<b>ROM</b>	Read only memory; a type of digital storage that cannot be written to. A ROM chip is manufactured with its program or data already stored in it and can never be changed. However, the program or data can be read as many times as desired.
<b>rotate register</b>	A shift register in which the data moved out from one end is placed back into the shift register at the other end.
<b>RS-232C interface</b>	An industry standard for serial communications.
<b>RUN mode</b>	The operating mode used by the PC for normal control operations.
<b>rung</b>	See <i>instruction line</i> .
<b>scan</b>	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.
<b>scan time</b>	See <i>cycle time</i> .
<b>scheduled interrupt</b>	An interrupt that is automatically generated by the system at a specific time or program location specified by the operator. Scheduled interrupts result in the execution of specific subroutines that can be used for instructions that must be executed repeatedly at a specified interval of time.
<b>SCP</b>	See <i>subtract count input</i> .
<b>seal</b>	See <i>self-maintaining bit</i> .
<b>self diagnosis</b>	A process whereby the system checks its own operation and generates a warning or error if an abnormality is discovered.

<b>self-maintaining bit</b>	A bit that is programmed to maintain either an OFF or ON status until set or reset by specified conditions.
<b>series</b>	A wiring method in which Units are wired consecutively in a string.
<b>servicing</b>	The process whereby the PC checks a connector or Unit to see if special processing is required.
<b>set</b>	The process of turning a bit or signal ON.
<b>set value</b>	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.
<b>shift input signal</b>	An input signal whose OFF to ON transition causes data to be shifted one bit.
<b>shift register</b>	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.
<b>signed binary</b>	A binary value that is stored in memory along with a bit that indicates whether the value is positive or negative.
<b>software error</b>	An error that originates in a software program.
<b>software protect</b>	A means of protecting data from being changed that uses software as opposed to a physical switch or other hardware setting.
<b>source (word)</b>	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.
<b>special instruction</b>	An instruction input with a function code that handles data processing operations within ladder diagrams, as opposed to a basic instruction, which makes up the fundamental portion of a ladder diagram.
<b>SR area</b>	A memory area containing flags and other bits/words with specific functions.
<b>SSS</b>	See <i>SYSMAC Support Software</i> .
<b>store</b>	The process of recording a program written into a display buffer permanently in memory.
<b>subroutine</b>	A group of instructions placed separate from the main program and executed only when called from the main program or activated by an interrupt.
<b>subroutine number</b>	A definer used to identify the subroutine that a subroutine call or interrupt activates.
<b>subtract count input</b>	An input signal used to decrement a counter when the signal changes from OFF to ON.
<b>SV</b>	See <i>set value</i> .
<b>switching capacity</b>	The maximum voltage/current that a relay can safely switch on and off.
<b>synchronous execution</b>	Execution of programs and servicing operations in which program execution and servicing are synchronized so that all servicing operations are executed each time the programs are executed.
<b>syntax</b>	The form of a program statement (as opposed to its meaning).

<b>syntax error</b>	An error in the way in which a program is written. Syntax errors can include 'spelling' mistakes (i.e., a function code that does not exist), mistakes in specifying operands within acceptable parameters (e.g., specifying read-only bits as a destination), and mistakes in actual application of instructions (e.g., a call to a subroutine that does not exist).
<b>SYSMAC Support Software</b>	A software package installed on a IBM PC/AT or compatible computer to function as a Programming Device.
<b>system configuration</b>	The arrangement in which Units in a System are connected. This term refers to the conceptual arrangement and wiring together of all the devices needed to comprise the System.
<b>system error</b>	An error generated by the system, as opposed to one resulting from execution of an instruction designed to generate an error.
<b>system error message</b>	An error message generated by the system, as opposed to one resulting from execution of an instruction designed to generate a message.
<b>system setup</b>	Operating environment settings for a Programming Device, e.g., the LSS or SSS.
<b>terminal instruction</b>	An instruction placed on the right side of a ladder diagram that uses the final execution conditions of an instruction line.
<b>timer</b>	A location in memory accessed through a TIM/CNT bit and used to time down from the timer's set value. Timers are turned ON and reset according to their execution conditions.
<b>TR area</b>	A data area used to store execution conditions so that they can be reloaded later for use with other instructions.
<b>TR bit</b>	A bit in the TR area.
<b>trace</b>	An operation whereby the program is executed and the resulting data is stored to enable step-by-step analysis and debugging.
<b>trace memory</b>	A memory area used to store the results of trace operations.
<b>transfer</b>	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.
<b>transmission distance</b>	The distance that a signal can be transmitted.
<b>trigger</b>	A signal used to activate some process, e.g., the execution of a trace operation.
<b>trigger address</b>	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.
<b>UM area</b>	The memory area used to hold the active program, i.e., the program that is being currently executed.
<b>Unit</b>	In OMRON PC terminology, the word Unit is capitalized to indicate any product sold for a PC System. Most of the names of these products end with the word Unit.
<b>unit number</b>	A number assigned to some Units to facilitate identification when assigning words or other operating parameters.
<b>unmasked bit</b>	A bit whose status is effective. See <i>masked bit</i> .

<b>unsigned binary</b>	A binary value that is stored in memory without any indication of whether it is positive or negative.
<b>uploading</b>	The process of transferring a program or data from a lower-level or slave computer to a higher-level or host computer. If a Programming Device is involved, the Programming Device is considered the host computer.
<b>watchdog timer</b>	A timer within the system that ensures that the scan time stays within specified limits. When limits are reached, either warnings are given or PC operation is stopped depending on the particular limit that is reached.
<b>WDT</b>	See <i>watchdog timer</i> .
<b>word</b>	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.
<b>word address</b>	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.
<b>work area</b>	A part of memory containing work words/bits.
<b>work bit</b>	A bit in a work word.
<b>work word</b>	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.
<b>write protect switch</b>	A switch used to write-protect the contents of a storage device, e.g., a floppy disk. If the hole on the upper left of a floppy disk is open, the information on this floppy disk cannot be altered.
<b>write-protect</b>	A state in which the contents of a storage device can be read but cannot be altered.

# Index

## A–B

ACC(—), 22, 25, 26, 30  
ADBL(—), 113  
address tracing. *See* tracing, data tracing.  
advanced I/O instructions  
  7-SEGMENT DISPLAY OUTPUT, 125  
  DIGITAL SWITCH INPUT, 122  
  functions, 118  
  HEXADECIMAL KEY INPUT, 120  
  TEN-KEY INPUT, 118  
  using alternate I/O bits, 127  
analog volume controls. *See* analog settings  
arithmetic flags, 113, 185  
ASCII, converting data, 262, 263  
bits, controlling, 197

## C

check levels, program checks, 425  
checksum, calculating frame checksum, 321  
clock, 469  
communication errors, 428  
communications  
  CQM1 PC Setup, 88  
  host link, node number (CQM1), 89  
  host link (CPM1/CPM1A), 93  
  host link (CQM1), 91  
  host link (SRM1), 95  
  link  
    NT Link (CPM1/CPM1A), 103  
    NT Link (SRM1), 106  
    one-to-one (CPM1/CPM1A), 101  
    one-to-one (CQM1), 100  
    one-to-one (SRM1), 104  
  one-to-one (CQM1), 89  
  RS-232C (CQM1/SRM1), 98  
  standard (CQM1). *See* settings  
  types, 87, 88  
  wiring, 91  
communications functions, 87  
constants, operands, 185  
counters  
  conditions when reset, 210, 212  
  creating extended timers, 211  
  reversible counters, 211  
CTBL(63), 22, 27  
cycle monitor time, PC Setup settings, 21

cycle time (CPM1/CPM1A)  
  calculating, 401  
  effects on operations, 401  
  processes, 401  
cycle time (CQM1)  
  calculating, 383  
  effects on operations, 384  
  processes, 383  
cycle time (minimum), PC Setup settings, 18  
cycle time (SRM1)  
  calculating, 412  
  effects on operations, 412  
  processes, 412

## D

data  
  decrementing, 312  
  incrementing, 311  
data tracing, 315–340  
DBS(—), 113  
DBSL(—), 113  
decrementing. *See* data  
definers, definition, 184  
Differential Monitor, function, 129  
differentiated instructions, 186  
  function codes, 184

## E

EEPROM ICs. *See* Memory Cassettes  
end code, host link response end codes, 432  
end codes, host link response codes, 355  
error codes, programming, 205  
error log, PC Setup settings, 21  
error log area, 426  
error messages, programming, 317  
errors  
  1:1 links (CQM1), 101  
  communications, 428  
  fatal, 429  
  general, 424  
  non-fatal, 427  
  other, 430  
  programming, 425  
  Programming Console operations, 424  
  programming messages, 317  
  resetting, 205  
  types, 424  
  user-defined errors, 426  
execution condition, definition, 154

expansion instructions, 444  
function codes, 116

## F

FAL area, 205

FAL(06), 426

FALS(07), 426

flag

arithmetic, programming example, 243, 246

CY

clearing, 278

setting, 278

flags, error and arithmetic, 447

flash memory, SRM1, 145

Frame Check Sequence. *See* frames, FCS

frame checksum, calculating with FCS(—), 321

frames

dividing

*See also* host link

precautions, 354

FCS, 354

function codes, 184

expansion instructions, 116, 117

## H

high-speed counter interrupts, CPM1/CPM1A, 76

high-speed timers, PC Setup settings, 20

hold bit status, PC Setup settings, 17

host link

command and response formats, 352

communications

*See also* host link commands

methods, 350

PC transmission, 355

procedures, 350

procedures (CQM1), 92

data transfer, 351

dividing frames, 353

end codes, 355, 432

frame

definition, 351

maximum size, 351

node number (CQM1), 89

setting parameters, start and end codes (CQM1), 90

setting parameters (CQM1). *See* RS-232C

host link (CPM1/CPM1A), 93

host link (CQM1), 91

host link (SRM1), 95

host link commands

\*\*, 378

FK, 373

IC, 379

KC, 374

KR, 372

KS, 371

MF, 370

MM, 375

MS, 369

QQ, 376

R#, 364

R\$, 364

R%, 365

RC, 357

RD, 358

RG, 358

RH, 357

RJ, 359

RL, 356

RP, 376

RR, 356

SC, 370

TS, 375

W#, 366

W\$, 367

W%, 368

WC, 361

WD, 362

WG, 362

WH, 360

WJ, 363

WL, 360

WP, 376

WR, 359

XZ, 378

## I

I/O bits

CPM1/CPM1A, 140

CQM1, 135

SRM1, 144

I/O points, refreshing, 318

I/O refresh operations, types (CQM1), 383

I/O response time

CPM1/CPM1A. *See* timing

CQM1. *See* timing

one-to-one link communications

CPM1/CPM1A, 403

CQM1, 386

SRM1, 414

SRM1. *See* timing

I/O terminals

IR bit allocation, 141

IR bit allocation (CPM1/CPM1A), 141

I/O words, allocating (CQM1), 136

incrementing, 311

indirect addressing, 185

INI(61), 25, 26, 37



- initialization processes, CPM1/CPM1A, 400
- input interrupts, CPM1/CPM1A, 69
- input time constants, PC Setup settings, 19
- inputs, quick-response inputs (CPM1/CPM1A), 131
- instruction set, 441
  - 7SEG(88), 345
  - ACC(—), 22, 25, 26, 30, 334
  - ADB(50), 289
  - ADBL(—), 113
  - ADD(30), 278
  - ADDL(54), 283
  - AND, 156, 196
    - combining with OR, 157
  - AND LD, 159, 197
    - combining with OR LD, 162
    - use in logic blocks, 161
  - AND NOT, 156, 196
  - ANDW(34), 309
  - ASC(86), 262
  - ASFT(17), 230
  - ASL(25), 225
  - ASR(26), 226
  - AVG(—), 302
  - BCD(24), 253
  - BCDL(59), 254
  - BCMP(68), 244
  - BCNT(67), 320
  - BIN(23), 252
  - BINL(58), 254
  - BSET(71), 234
  - CLC(41), 278
  - CMP(20), 242
  - CMPL(60), 246
  - CNT, 210
  - CNTR(12), 211
  - COLL(81), 237
  - COM(29), 308
  - CPS(—), 248
  - CPSL(—), 249
  - CTBL(63), 22, 27, 215
  - CTW(—), 273
  - DBS(—), 113, 298
  - DBSL(—), 113, 299
  - DEC(39), 312
  - DIFD(14), 174, 200–209
    - using in interlocks, 202
    - using in jumps, 204
  - DIFU(13), 174, 200–209
    - using in interlocks, 202
    - using in jumps, 204
  - DIST(80), 235
  - DIV(33), 282
  - DIVL(57), 287
  - DMPX(77), 257
  - DSW(—), 21
  - DSW(87), 346
  - DVB(53), 292
  - END(01), 159, 201
  - FAL(06), 205
  - FALS(07), 205
  - FCS(—), 321
  - FPD(—), 322
  - HEX(—), 263
  - HKY(—), 347
  - HTS(65), 271
  - IL(02), 170, 201–203
  - ILC(03), 170, 201–203
  - INC(38), 311
  - INI(61), 26, 37, 220
  - INT(89), 44, 73, 326
  - IORF(97), 318
  - JME(05), 203
  - JMP(04), 203
  - JMP(04) and JME(05), 172
  - KEEP(11), 199
    - in controlling bit status, 174
  - ladder instructions, 155
  - LD, 156, 196
  - LD NOT, 156, 196
  - MAX(—), 300
  - MBS(—), 113, 296
  - MBSL(—), 113, 297
  - MCMP(19), 247
  - MCRO(99), 319
  - MIN(—), 301
  - MLB(52), 291
  - MLPX(76), 255
  - MOV(21), 232
  - MOVB(82), 239
  - MOVD(83), 240
  - MSG(46), 317
  - MUL(32), 281
  - MULL(56), 286
  - MVN(22), 232
  - NEG(—), 113
  - NEGL(—), 113
  - NOP(00), 201
  - NOT, 153
  - operands, 152
  - OR, 157, 196
    - combining with AND, 157
  - OR LD, 160, 197
    - combining with AND LD, 162
    - use in logic blocks, 161
  - OR NOT, 157, 196
  - ORW(35), 309
  - OUT, 158, 197
  - OUT NOT, 158, 197
  - PID(—), 338
  - PLS2(—), 22, 25, 26, 29, 332
  - PMW(—), 32
  - PRV(62), 34, 65, 221
  - PULS(65), 23, 26, 27, 36, 328
  - PWM(—), 336
  - RESET, 173
  - RET(93), 38, 315
  - ROL(27), 226
  - ROOT(72), 288
  - ROR(28), 227
  - RSET, 198–199
  - RXD(47), 110, 340
  - SBB(51), 290
  - SBBL(—), 113
  - SBN(92), 67, 315
  - SBS(91), 313
  - SCL(66), 266
  - SCL2(—), 268
  - SCL3(—), 269

- SDEC(78), 259
- SET, 173, 198–199
- SFT(10), 224
- SFTR(84), 229
- SLD(74), 228
- SNXT(09), 206
- SPED(64), 23, 25, 26, 27, 28, 36, 330
- SRCH(—), 337
- SRD(75), 228
- STC(40), 278
- STEP(08), 206
- STH(—), 272
- STIM(69), 45, 213
- STUP(—), 344
- SUB(31), 279
- SUBL(55), 285
- SUM(—), 303
- TCMP(85), 243
- terminology, 152
- TIM, 209
- TIMH(15), 212
- TKY(18), 347
- TRSM(45), 315
- TXD(48), 110, 342
- VCAL(—), 305
- WSFT(16), 225
- WTC(—), 274
- XCHG(73), 235
- XFER(70), 233
- XFRB(—), 241
- XNRW(37), 311
- XORW(36), 310
- ZCP(—), 250
- ZCPL(—), 252
- instruction sets
  - ADBL(—), 293
  - NEG(—), 275
  - NEGL(—), 276
  - SBBL(—), 294
- instructions
  - advanced I/O, 118
  - execution times
    - CPM1/CPM1A, 406
    - CQM1, 389
    - SRM1, 417
  - expansion, 116
  - instruction set lists, 190
  - mnemonics list, ladder, 193
  - right-hand instructions, coding multiple, 168
  - subroutines, 313
  - tables
    - default, 117
    - user-set, 117
- INT(89), 44, 73
- interlocks, 201–203
  - using self-maintaining bits, 175
- interrupt functions
  - CPM1/CPM1A, 67
  - CQM1, 38
  - SRM1, 83
- interrupt processing
  - calculating response time
    - CPM1/CPM1A, 405
    - CQM1, 389
  - masking
    - CPM1/CPM1A, 405
    - CQM1, 388
    - SRM1, 416
  - timing (CPM1/CPM1A), 405
  - timing (CQM1), 388
  - timing (SRM1), 416
- interrupts
  - control, 326
  - high-speed counter, programming, 80
  - input, parameters (CQM1), 39
  - types (CQM1), 38
- interrupts (CPM1/CPM1A)
  - counter mode, 72
  - high-speed counter, 76
    - overflows and overflows, 77
  - input interrupt mode, 70
  - input interrupts, 69
  - interval timers, 74
  - masking, 73
  - setting modes, 70
  - types, 67
  - unmasking, 74
- interrupts (CPM1/CPM1A), interval timers, scheduled interrupt mode, 74
- interrupts (CQM1)
  - absolute high-speed counters
    - CQM1-CPU44-E, 62
    - programming, 63
  - counter mode, 41
  - high-speed counter
    - overflows and overflows, 54
    - programming, 50
  - high-speed counter 0, 47
    - overflows and overflows, 53
  - high-speed counters 1 and 2
    - CQM1-CPU43-E, 55
    - programming, 57
  - input interrupts, 39
  - interval timers, 44
    - scheduled interrupt mode, 45
  - masking, 44
  - setting modes, 40
  - unmasking, 44
- interrupts (SRM1)
  - interval timers, 83
  - scheduled interrupt mode, 84
  - types, 83
- interval timer interrupts
  - CPM1/CPM1A, 74
  - SRM1, 83

## J–L

- jump numbers, 203
- jumps, 203–204

ladder diagram  
 branching, 168  
   IL(02) and ILC(03), 170  
   using TR bits, 169  
 combining logic blocks, 163  
 controlling bit status  
   using DIFU(13) and DIFD(14), 174, 200–209  
   using KEEP(11), 199–200  
   using OUT and OUT NOT, 158  
   using SET and RESET, 173  
   using SET and RSET, 198–199  
 converting to mnemonic code, 154–173  
 display via GPC, FIT, LSS, or SSS, 153  
 instructions  
   combining, AND LD and OR LD, 162  
   controlling bit status  
     using KEEP(11), 174  
     using OUT and OUT NOT, 197  
   format, 184  
 notation, 184  
 structure, 153  
 using logic blocks, 159

ladder diagram instructions, 196–197

lithium battery, backup time vs. temperature, xvi

logic block instructions, converting to mnemonic code, 159–167

logic blocks. *See* ladder diagram

## M

macro function, subroutines. *See* programming

masking  
 CPM1/CPM1A interrupt processes, 405  
 CQM1 interrupt processes, 388  
 SRM1 interrupt processes, 416

MBS(—), 113

MBSL(—), 113

memory areas  
 AR area bits (CPM1/CPM1A), 142, 462  
 AR area bits (CQM1), 138, 455  
 AR area bits (SRM1), 145, 466  
 DM area (CPM1/CPM1A), 142  
 DM area (CQM1), 139  
 DM area (SRM1), 145  
 flag bits (CPM1/CPM1A), 141, 460  
 flag bits (CQM1), 137, 452  
 flag bits (SRM1), 144, 465  
 HR area (CPM1/CPM1A), 142  
 HR area (CQM1), 137  
 HR area (SRM1), 145  
 IR area bits (CPM1/CPM1A), 140  
 IR area bits (CQM1), 135  
 IR area bits (SRM1), 144  
 link bits (CPM1/CPM1A), 142  
 link bits (CQM1), 138  
 link bits (SRM1), 145  
 structure (CPM1/CPM1A), 139, 459  
 structure (CQM1), 134, 451  
 structure (SRM1), 143, 464

timer and counter bits (CPM1/CPM1A), 142  
 timer and counter bits (CQM1), 138  
 timer and counter bits (SRM1), 145  
 TR bits (CPM1/CPM1A), 142  
 TR bits (CQM1), 137  
 TR bits (SRM1), 144  
 user program memory (CQM1), 139  
 work bits (CPM1/CPM1A), 141  
 work bits (CQM1), 135  
 work bits (SRM1), 144

Memory Cassettes, 139  
 and program size, 147  
 automatic reading, 149  
 comparing contents, 149  
 contents, 147  
 reading data, 148  
 reading with peripherals, 149  
 required EEPROMs, 146  
 storing DM and UM data (CQM1 only), 146  
 types, 146  
 writing data, 148

messages, programming, 317

mnemonic code, converting, 154–173

MSG(46), 426

## N

NEG(—), 113

NEGL(—), 113

nesting, subroutines, 314

no protocol communications  
 program example, 111  
 SRM1, 107  
 transmission data configuration, 110  
 transmission flags, 110

normally open/closed condition, definition, 153

NOT, definition, 153

NT Link (CPM1/CPM1A), 103

NT Link (SRM1), 106

## O

one-to-one link communications  
 I/O response timing  
   CPM1/CPM1A, 403  
   CQM1, 386  
   SRM1, 414  
 link errors (CQM1), 101

one-to-one link (CPM1/CPM1A), 101

one-to-one link (CQM1), 100

one-to-one link (SRM1), 104

operand bit, 154

operands, 184  
 allowable designations, 184  
 requirements, 184

operations  
  CPM1/CPM1A internal processing, flowchart, 400  
  CQM1 internal processing, flowchart, 382  
  effects on CPM1/CPM1A cycle time, 401  
  effects on CQM1 cycle time, 384  
  effects on SRM1 cycle time, 412  
  SRM1 internal processing, flowchart, 411

output bit  
  controlling ON/OFF time, 198  
  controlling status, 173, 175

output refresh method, PC Setup settings, 21

outputs, turning OFF, 428

## P

PC Setup. *See* settings

PC Setup settings  
  CPM1/CPM1A, 9  
  CQM1, 4  
  SRM1, 13

peripheral port, servicing time, 18

peripheral port servicing time, PC Setup settings, 18

PLS2(—), 22, 25, 26, 29

PMW(—), 32

precautions, general, xiii

Program Memory, structure, 154

program write protection, PC Setup settings, 17

programming  
  absolute high-speed counters (CQM1), 63  
  errors, 425  
  high-speed counter (CPM1/CPM1A), 80  
  high-speed counter (CQM1), 50  
  instructions, 441  
  interrupts, 80  
  interrupts (CQM1), 50, 63  
  jumps, 172  
  macro function, subroutines, 128  
  precautions, 177  
  preparing data in data areas, 234  
  special features, 116  
  writing, 152

programming (CQM1)  
  high-speed counters 1 and 2, 57  
  interrupts, 57

programs  
  checking, check levels, 425  
  executing, 179

PRV(62), 34, 65

PULS(65), 23, 26, 27, 36

pulse outputs  
  determining status of ports 1 and 2, 34  
  from an output point, 23  
  from ports 1 and 2, 25  
  variable-duty-ratio, 32

PV  
  CNTR(12), 211  
  timers and counters, 209

## Q–R

quick-response inputs (CPM1/CPM1A), 131

RET(93), 38

right-hand instructions, coding. *See* instructions

RS-232C  
  communications  
    one-to-one link (CQM1), 100  
    procedures (CQM1/SRM1), 98  
    receiving (CQM1), 99  
    transmitting (SRM1), 98  
  connecting Units (CQM1), 100  
  control bits (CQM1), 99  
  selecting (CQM1). *See* host link

RS-232C port, servicing time (CQM1/SRM1), 18

RS-232C port servicing time, PC Setup settings, 18

RXD(47), 110

## S

SBBL(—), 113

SBN(92), 67

settings  
  basic operations  
    error log, 21  
    high-speed timers, 20  
    hold bit status, 17  
    input digits number, 21  
    startup mode, 16  
  changing, 3  
  communications, 87, 88  
  conditions (CQM1), 90  
  CQM1 PC Setup, 88  
  host link (CPM1/CPM1A), 93  
  host link (CQM1), 91  
  host link (SRM1), 95  
  RS-232C (CQM1/SRM1), 98  
  defaults, 3  
  expansion instructions, 117  
  I/O operations, 16  
    port servicing scan time, 18  
    pulse output word, 23, 26, 32  
  interrupts, 38  
    external sources (CQM1), 39  
    parameters (CQM1), 40  
  pulse outputs (CQM1 only), 22

seven-segment displays, converting data, 259

signed binary data, 111

SPED(64), 23, 25, 26, 27, 28, 36

startup mode, PC Setup settings, 16

STIM(69), 45

subroutine number, 315

SV

CNTR(12), 211  
timers and counters, 209

## T–W

TC numbers, 208

timers, conditions when reset, 209, 212

timing

basic instructions

CPM1/CPM1A, 406  
CQM1, 390  
SRM1, 417

CPM1/CPM1A cycle time, 401

CQM1 cycle time, 383

I/O response time (CPM1/CPM1A), 402

I/O response time (CQM1), 385

I/O response time (SRM1), 414

instruction execution

CPM1/CPM1A. *See* instruction  
CQM1. *See* instruction  
SRM1. *See* instruction

interrupt processing

CPM1/CPM1A, 405  
CQM1, 388  
SRM1, 416

special instructions

CPM1/CPM1A, 406  
CQM1, 390  
SRM1, 417, 421

SRM1 cycle time, 412

TR bits, use in branching, 169

tracing. *See* See data tracing and address tracing.


troubleshooting, 434

TXD(48), 110

write protecting the program, PC Setup settings, 17

# Revision History

A manual revision code appears as a suffix to the catalog number on the front cover of the manual.

Cat. No. W228-E1-7  


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	August 1993	Original production
2	December 1993	The manual was extensively rewritten to incorporate information related the new CPUs, I/O Units, and Dedicated I/O Units.
2A	March 1994	<b>Page 268:</b> The function/setting range comments for Word P1+3 have been corrected.
2B	March 1995	<p>The following instructions have been corrected throughout the manual: INI(—) corrected to INI(61); PRV(—) corrected to PRV(62); CTBL(—) corrected to CTBL(63); SPED(—) corrected to SPED(64); PULS(—) corrected to PULS(65); INT(—) corrected to INT(89); STIM(—) corrected to STIM(69); BCMP(—) corrected to BCMP(68); TXD(—) corrected to TXD(48); RXD(—) corrected to RXD(47); TKY(—) corrected to TKY(18); DSW(—) corrected to DSW(87); 7SEG(—) corrected to 7SEG(88); ASFT(—) corrected to ASFT(17); CMPL(—) corrected to CMPL(60); MCMP(—) corrected to MCMP(19); SCL(—) corrected to SCL(66); BCNT(—) corrected to BCNT(67)</p> <p><b>Pages 1, 2, 54, 76, 82, 86, 87, 90, 91, 92, 130, 138, 246, 247, 248, 328, 331, 345, 362, 375, 393, 400, 401:</b> SYSMAC Support Software (SSS) added.</p> <p><b>Page 37:</b> Third sentence in "Programming" rewritten.</p> <p><b>Page 47:</b> The "1 kHz" corrected to "4 kHz" in the first sentence of <i>1-4-8 Absolute High-speed Counter Interrupts (CQM1-CPU44-E)</i>.</p> <p><b>Page 61:</b> Areas for reading and writing reversed for the top diagram. <i>1-6 Calculating with Signed Binary Data</i> added.</p> <p><b>Pages 66, 68:</b> Text and diagram corrected for <i>Using the Instruction</i>.</p> <p><b>Page 70:</b> Top diagram corrected.</p> <p><b>Page 71:</b> Top diagram corrected. Application example changed.</p> <p><b>Pages 80, 357:</b> IR area words and bits corrected</p> <p><b>Page 128:</b> SUM added to the table.</p> <p><b>Page 142:</b> First sentence in Example corrected.</p> <p><b>Page 146:</b> Sentence added before the note in Reading Timer PVs.</p> <p><b>Page 148:</b> Second paragraph added to Target Value Comparison.</p> <p><b>Page 150:</b> Note 2 corrected.</p> <p><b>Page 151:</b> Paragraph added above <i>5-15-7 MODE CONTROL - INI(—)</i>.</p> <p><b>Page 176:</b> Bits 12 to 16 corrected to bits 11 to 15 at the top of the page.</p> <p><b>Page 200:</b> Top equation and example diagram corrected.</p> <p><b>Page 262:</b> Second-to-last paragraph of Operand Settings corrected. Caution added.</p> <p><b>Page 268:</b> Note added.</p> <p><b>Page 272:</b> Diagram for the control word corrected.</p> <p><b>Page 281:</b> Delimiter corrected to Terminator in the top diagram and Frame 2 corrected to Frame 3 in the bottom diagram.</p> <p><b>Page 330:</b> First sentence corrected in <i>8-4 User-defined Errors</i>.</p> <p><b>Page 336:</b> Note corrected.</p> <p><b>Pages 343 to 345:</b> <i>Appendix A Standard Models</i> deleted.</p> <p><b>Pages 347 to 352:</b> Programming instructions lists updated.</p> <p><b>Page 364:</b> "01" corrected to "00" for AR21.</p> <p><b>Page 368:</b> Input constants for IR 00000 to IR 00007 of DM 6620 corrected.</p> <p><b>Page 372:</b> Note added.</p>
2C	July 1995	<p>The following additions/corrections were made.</p> <p><b>Page 53:</b> Default communications settings changed and note added.</p> <p><b>Page 60:</b> Paragraph added on link errors.</p> <p><b>Page 63:</b> The following sentence was corrected: Examples are shown below for the Programming Console.</p> <p><b>Page 73:</b> Note added on use of output point 5.</p> <p><b>Page 200:</b> "#" added to the digit designator. Bytes 61 to 66 (a to f) have been deleted from the third sentence in <i>Limitations</i>.</p> <p><b>Page 237:</b> @AVG(—) has been deleted.</p> <p><b>Page 253:</b> Operand data area ranges corrected for IORF(97).</p> <p><b>Page 272:</b> Caution added.</p> <p><b>Page 353:</b> "Programming Console" changed to "Programming Device" and paragraph added following this change.</p> <p><b>Page 373:</b> <i>Appendix E Battery Service Life</i> deleted.</p>
3	December 1995	The manual was extensively revised to include the new CPM1 PCs.
3A	April 1996	<b>Page 296:</b> <i>Precautions Regarding Pulse Output</i> added to <i>Pulse Frequency (F)</i> . <b>Page 415:</b> The bit allocation for the day of the week corrected.
4	June 1997	The manual was revised to include the new CPM1A and SRM1 PCs and the V1 CQM1 CPU Units. The LSS was removed from the manual. <b>Page 54:</b> Note added. <b>Page 105:</b> CLC was added to program.

## Revision History

Revision code	Date	Revised content
5	March 1998	<p>The manual was revised to include the pulse output function for the CPM1A and some CPM1/CPM1A/SRM1 memory specification changes.</p> <p><b>Page xi:</b> Information added.  <b>Page 3:</b> Information added.  <b>Page 3:</b> "Caution" notes deleted.  <b>Page 9:</b> Additions to table.  <b>Pages 11, 12:</b> Notes added.  <b>Pages 11, 20:</b> Error log records corrected from "10" to "7."  <b>Page 16:</b> Note added.  <b>Page 35 to 37:</b> New section added (CPM1A pulse output function).  <b>Page 77 to 78:</b> Information added.  <b>Page 93:</b> Note added.  <b>Page 104:</b> Note added.  <b>Page 138:</b> Changes made to note.  <b>Page 401, 402:</b> Specifications changed.  <b>Page 403:</b> Changes made to text.  <b>Page 404:</b> Changes made to graphic; text added.  <b>Page 405:</b> Graphic changed.  <b>Page 429:</b> Addition to table.  <b>Page 430:</b> Change made to text in graphic.  <b>Page 431:</b> Change made to text in graphic.  <b>Page 463:</b> Changes made to AR area memory.  <b>Page 459, 462, 464:</b> Notes added.  <b>Page 468:</b> Addition to table.  <b>Page 481:</b> Appendix I added.</p>
6	December 1998	<p>The following corrections and revisions were made.</p> <p>A precautions section was added before section 1.  <b>Pages 5 and 19:</b> IR 012 to IR 015 added in DM 6626 and DM 6627.  <b>Page 49 and 78:</b> Descriptions of target and range comparisons added.  <b>Page 317:</b> Ranges corrected for starting and end words.</p>
7	May 1999	<p>The following corrections and revisions were made.</p> <p><b>Page 7:</b> Setting for 1:1 NT Link added for DM 6645, bits 12 to 15.  <b>Page 89:</b> Setting for 1:1 NT Link added for DM 6645 and DM 6650.  <b>Page 89, 148, and 149:</b> Manually resetting bits from LSS/SSS added.  <b>Page 205:</b> "A maximum of three" removed from <i>Resetting Errors</i>.  <b>Page 456:</b> AR 1115 added.  <b>Page 457:</b> Note added for AR 1400, AR 1401 and AR 1402.  <b>Page 469:</b> AR 2115 corrected to AR 2113 at the bottom of the page.</p>

# OMRON

**Authorized Distributor:**