

**SYSMAC**  
**C200H-ASC11/ASC21/ASC31**  
**ASCII Units**

**OPERATION MANUAL**

**OMRON**

# **C200H-ASC11/ASC21/ASC31 ASCII Units**


## **Operation Manual**


Revised June 2000


## 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, 1998

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

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

# TABLE OF CONTENTS

<b>PRECAUTIONS</b> .....	<b>xi</b>
1 Intended Audience .....	xii
2 General Precautions .....	xii
3 Safety Precautions .....	xii
4 Operating Environment Precautions .....	xii
5 Application Precautions .....	xiii
<b>SECTION 1</b>	
<b>Introduction</b> .....	<b>1</b>
1-1 Overview .....	2
1-2 System Configuration .....	3
1-3 Internal Configuration .....	5
1-4 Specifications .....	7
1-5 Comparison with Previous ASCII Units .....	10
1-6 Basic Operating Procedures .....	12
<b>SECTION 2</b>	
<b>Installation</b> .....	<b>17</b>
2-1 Nomenclature and Functions .....	18
2-2 Installation .....	22
<b>SECTION 3</b>	
<b>IR/CIO and DM Area Allocations</b> .....	<b>29</b>
3-1 IR/CIO Area Allocations .....	30
3-2 DM Area Allocations .....	38
<b>SECTION 4</b>	
<b>Data Exchange with General-purpose External Devices</b>	<b>43</b>
4-1 Overview .....	44
4-2 Opening a Communications Port .....	44
4-3 Sending Data .....	47
4-4 Receiving Data .....	51
4-5 Closing a Communications Port .....	53
4-6 Communications Errors .....	59
<b>SECTION 5</b>	
<b>Processing</b> .....	<b>61</b>
5-1 Processing Character Strings .....	62
5-2 Processing Bits .....	62
5-3 Processing Receive Buffers .....	62
5-4 Time Processing .....	63
5-5 Interrupt Functions .....	64
5-6 Loop Processing .....	67
<b>SECTION 6</b>	
<b>Data Exchange with the CPU Unit</b> .....	<b>69</b>
6-1 Overview of Data Exchanges .....	70

# TABLE OF CONTENTS

6-2	Selecting the Data Exchange Method .....	74
6-3	Details of the Data Exchange Methods .....	75
6-4	Data Exchange Time Charts .....	87
6-5	IOWR/IORD Instruction Specifications .....	95
6-6	Restrictions for CS1-series PCs.....	108

## SECTION 7

### **Editing BASIC Programs .....** **111**

7-1	Programming Procedure .....	112
7-2	Character Variable Space Allocations .....	118
7-3	Starting/Stopping the BASIC Program .....	120
7-4	Program Configuration .....	121
7-5	List of BASIC Commands.....	129
7-6	BASIC Commands .....	139
7-7	User-defined BASIC Functions.....	231
7-8	Debugging .....	232

## SECTION 8

### **Data Exchange Application Programs .....** **235**

8-1	Asynchronous Processing .....	236
8-2	High-speed Data Exchanges .....	237
8-3	High-volume Data Exchanges .....	239
8-4	Data Transfer: ASCII Unit to CPU Unit .....	239
8-5	Bit Data Exchanges .....	240

## SECTION 9

### **Clearing Error Messages .....** **241**

9-1	List of Error Messages.....	242
9-2	Troubleshooting.....	248
9-3	CPU Unit Error Indicators.....	252
9-4	Reading and Clearing the Error Log .....	254
9-5	Maintenance.....	255

## Appendix

A	Operating Precautions .....	257
B	Comparison with C200H-ASC02 .....	263
C	PC Format .....	265
D	Formats for Storing Variables in Memory .....	275
E	Command Execution Time Samples .....	279
F	Sample Programs .....	283
G	Wiring RS-232C or RS-422A/485 Cable Connectors .....	287
H	Reserved Words .....	289

### **Index .....** **291**

### **Revision History .....** **295**

# About this Manual:

This manual describes the installation and operation of the C200H-ASC11/ASC21/ASC31 ASCII Units and includes the sections described below. These ASCII Units can be used with the C200HX/HG/HE, ZE-version C200HX/HG/HE, C200H, C200HS, and CS1-series PCs as a Special I/O Unit. Unless specified otherwise, references to the “C200HX/HG/HE” include the ZE-version C200HX/HG/HE PCs.

It has been assumed in the writing of this manual that the reader is already familiar with the hardware, programming, and terminology of OMRON PCs. If a review of this information is necessary, the reader should refer to the appropriate OMRON PC manuals for assistance.

Please read this manual carefully and be sure you understand the information provided before attempting to install and operate an ASCII Unit. **Be sure to read the precautions in the following section.**

**Section 1** provides basic information on the features of the ASCII Unit, internal configurations, and operating procedures. It also provides the Unit's specifications, and a comparison of the ASCII Unit models covered in this manual with the previous ASCII Unit.

**Section 2** describes how to install and connect the ASCII Unit, including mounting to the PC Back-plane and connecting to external devices/computers.

**Section 3** describes the methods used to allocate words to the ASCII Unit in the IR and DM Areas and the applications of the bits and words in these areas.

**Section 4** describes the ASCII Unit commands in conjunction with transmission control signals for opening and closing communications ports and for sending and receiving data between the ASCII Unit and external devices.

**Section 5** provides examples of processing character strings, bits, receive buffers, looping, and interrupts.

**Section 6** describes procedures for exchanging data between the ASCII Unit and the CPU Unit, the transfer sequence, instruction/command timing, and the use of various interrupts.

**Section 7** describes the procedures for editing, saving, starting, and stopping BASIC programs. An overview of BASIC program configuration, language definitions, and the use of various BASIC commands, statements, and functions is provided as well.

**Section 8** provides examples of data exchange applications.

**Section 9** provides a list of error messages, probable causes and possible corrections.

The **Appendices** provide important operating precautions, differences with the command language used for the ASC02 ASCII Unit, the PC format, and the formats for storing variables in memory.



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

# PRECAUTIONS

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

**The information contained in this section is important for the safe and reliable application of the 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 .....	xii
2	General Precautions .....	xii
3	Safety Precautions .....	xii
4	Operating Environment Precautions .....	xii
5	Application Precautions .....	xiii

## 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

 **WARNING** Do not attempt to take any Unit apart while the power is being supplied. Doing so may result in electric shock.

 **Caution** Do not turn OFF the power supply to the PC while the BASIC program is being written to the flash ROM of the ASCII Unit by executing ROMSAVE command from the terminal.

 **WARNING** Do not attempt to disassemble, repair, or modify any Units. Any attempt to do so may result in malfunction, fire, or electric shock.


## 4 Operating Environment Precautions

 **Caution** Do not operate the control system in the following places:


- Locations subject to direct sunlight.
- Locations subject to temperatures or humidity outside the range specified in the specifications.



- Locations subject to condensation as the result of severe changes in temperature.
- Locations subject to corrosive or flammable gases.
- Locations subject to dust (especially iron dust) or salts.
- Locations subject to exposure to water, oil, or chemicals.
- Locations subject to shock or vibration.


 **Caution** Take appropriate and sufficient countermeasures when installing systems in the following locations:

- Locations subject to static electricity or other forms of noise.
- Locations subject to strong electromagnetic fields.
- Locations subject to possible exposure to radioactivity.
- Locations close to power supplies.


 **Caution** The operating environment of the PC System can have a large effect on the longevity and reliability of the system. Improper operating environments can lead to malfunction, failure, and other unforeseeable problems with the PC System. Be sure that the operating environment is within the specified conditions at installation and remains within the specified conditions during the life of the system.

## 5 Application Precautions

Observe the following precautions when using the ASCII Unit or PC System.

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

- Always turn OFF the power supply to the PC before attempting any of the following. Not turning OFF the power supply may result in malfunction or electric shock.
  - Mounting or dismounting I/O Units, CPU Units, Memory Cassettes, or any other Units.
  - Assembling the Units.
  - Setting DIP switches or rotary switches.
  - Connecting or wiring the cables.
  - Connecting or disconnecting the connectors.

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

- There are restrictions in the data areas and addresses that can be specified for the PC READ, PC WRITE, PC READ@, and PC WRITE@ commands in the BASIC program when using an ASCII Unit with a CS1-series CPU Unit. Be particularly careful when using existing BASIC programs in ASCII Units for CS1-series PCs.

- 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.
- Install the Unit properly as specified in the operation manual. Improper installation of the Unit may result in malfunction.
- Be sure that all the mounting screws, terminal screws, and cable connector screws are tightened to the torque specified in the relevant manuals. Incorrect tightening torque may result in malfunction.
- Leave the label attached to the Unit when wiring. Removing the label may result in malfunction.
- Remove the label after the completion of wiring to ensure proper heat dissipation. Leaving the label attached may result in malfunction.
- Double-check all the wiring before turning ON the power supply. Incorrect wiring may result in burning.
- Mount the Unit only after checking the connectors and wiring.
- Be sure that the terminal blocks, Memory Units, expansion cables, and other items with locking devices are properly locked into place. Improper locking may result in malfunction.
- 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.
- 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.

# SECTION 1

## Introduction

This section provides basic information on the features of the ASCII Unit, internal configurations, and operating procedures. It also provides the Unit's specifications, and a comparison of the ASCII Unit models covered in this manual with the previous ASCII Unit.

1-1	Overview . . . . .	2
1-2	System Configuration . . . . .	3
1-3	Internal Configuration . . . . .	5
1-4	Specifications . . . . .	7
1-5	Comparison with Previous ASCII Units . . . . .	10
1-6	Basic Operating Procedures . . . . .	12
1-6-1	Editing BASIC Programs . . . . .	12
1-6-2	BASIC Programming Procedure . . . . .	14
1-6-3	Connection and Transfer to External Devices . . . . .	14

## 1-1 Overview

The ASCII Unit is a Special I/O Unit that can exchange ASCII data with personal computers, board computers, bar code readers, ID systems, and non-OMRON Programmable Controllers (PCs) via an RS-232C or RS-422A/485 interface. Because it is programmed in BASIC, the ASCII Unit can independently process read/write data from general-purpose peripheral devices, making it easy to combine sequence control and information processing on the same PC.

### ASCII Communications with Peripheral Devices

The following functions are provided. These functions are not possible with protocol macros and host link non-procedural communications alone.

- Flexible processing of character strings (sampling, comparison, combination, and length determination of portions of character strings).
- Reception processing for only a portion of the data from a receive buffer.
- Processing of interrupt subroutines, such as those for communications, time processing, and PC interrupts.
- Flexible response processing, such as retries and time monitoring.

These functions enable the ASCII Unit to handle all kinds of communications frames and protocols. It can handle essentially all data exchanges with other devices equipped with an RS-232C or RS-422A/485 port.

### Multiport Communications

Up to either 10 or 16 ASCII Units can be mounted on each PC (depending on the model of the CPU Unit), making ASCII Units ideal for monitoring a large number of general-purpose peripheral devices via communications.

### Special Processing

Apart from communicating with general-purpose peripheral devices, ASCII Units can also be used for special operations. Data operations, character string processing, logic flow processing, and other operations that are difficult to carry out in the PC can be handled by using the ASCII Units as “sub-CPU’s” for the CPU Unit. This aids high-speed processing in the system as a whole, and reduces the load on the CPU Unit. In particular, by carrying out special operations and transmitting back the results when an interrupt is received from the CPU Unit, ASCII Units can act as co-processors for the CPU Unit.

## C200H-ASC11/ASC21/ASC31 Features

### High-capacity User Memory

ASCII Units each provide 200 Kbytes of user memory (BASIC program area and variable area), enabling them to handle complicated applications.

### High-speed Processing

A 32-bit MPU (C200H-ASC02 used an 8-bit processor) enables high-speed processing.

### RS-422A/485 Port

The RS-422A/485 port enables long-range communications and 1:N communications.

### Various Formats for CPU Unit Data Transfers

In addition to previous commands (PC READ/WRITE, PC READ@/WRITE@, and PC PUT/PC GET) it is also possible to execute PC QREAD/QWRITE triggered by the ASCII Unit, as well as the PC EPUT/EGET commands, which allow the ASCII Unit to read/write the memory shared between the ASCII Unit and the CPU Unit. For the C200HX/HG/HE or CS1-series PCs, the CPU Unit can immediately read/write the ASCII Unit using the IOWR/IORD instructions. This means that the user can select the data exchange method to match the application, including the read/write trigger, the Unit executing the read/write, the read/write timing, the transmitted data size, and whether or not special programming is required.

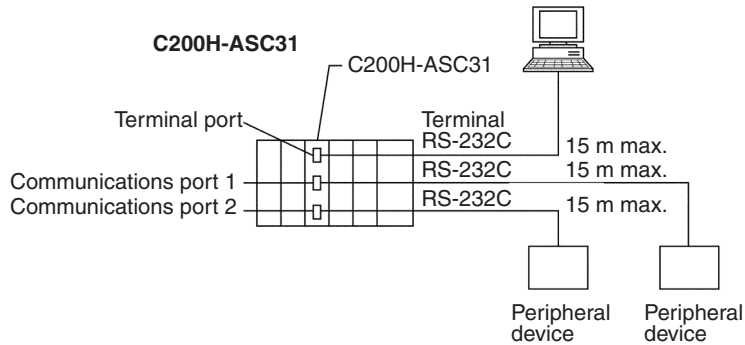
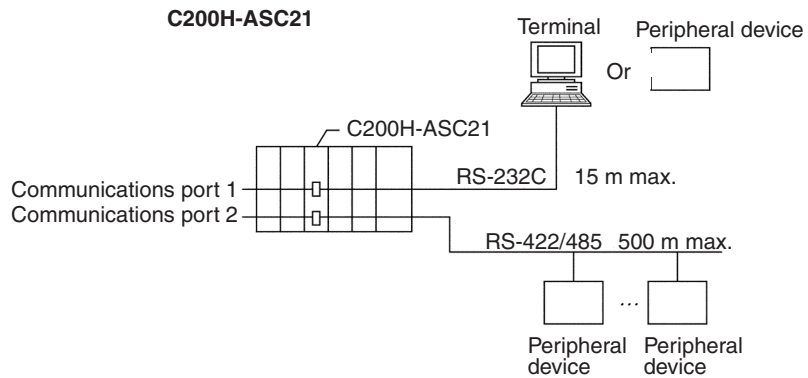
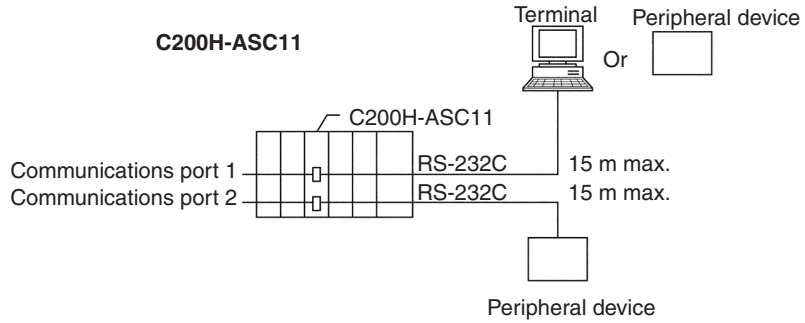
<b>High-speed Data Exchange Using Shared Memory</b>	There is an internal shared memory shared with the CPU Unit, and the ASCII Unit and the CPU Unit can access this shared memory asynchronously. Previously, in data exchange using the I/O refresh period, the CPU Unit or ASCII Unit had to wait until the I/O refresh timing arrived, but now shared memory can be read/written as soon as the command/instruction is executed. High-speed data transfer is also possible by combining an instantaneous interrupt with the IOWR instruction addressed to the ASCII Unit from the CPU Unit.
<b>Bit ON/OFF Command</b>	By treating variables as bit strings, individual bits out of 16 or 32 bits can be turned ON/OFF in integer variables, simplifying bit input processing to the PC.
<b>Extensive Interrupt Processing</b>	There are two ways that the CPU Unit can interrupt the ASCII Unit: by I/O refreshing or by IOWR instruction. There are 5 types of interrupts for receiving data from the communications circuit: General data reception, specified character reception, specified byte size reception, leading and final character reception, leading and specified number of characters reception. Interrupts can also occur for key inputs, when an error occurs, and at specified times. Timers can also generate either one-shot or interval interrupts.
<b>Built-in Transmission Control Signal Control</b>	Commands for transmission control signals (RTS, DTR) and commands to monitor transmission control signals (CTS, DSR) are provided. These commands simplify handling all types of transmission protocols.
<b>Error Check Calculation Commands</b>	The FCS instruction can be used to calculate error check codes, including those for LRC (longitudinal parity), CRC-CCITT (XMODEM), CRC-16 (MODBUS), and SUM (1-byte or 2-byte). This allows you to eliminate the programming previously required to calculate error check codes.
<b>Special Operations</b>	Calculations are possible for exponential function ( $e^x$ ), square root ( $\sqrt{\quad}$ ), trigonometric functions (TAN, SIN, and COS), inverse trigonometric functions ( $\text{TAN}^{-1}$ , $\text{SIN}^{-1}$ , $\text{COS}^{-1}$ ), and logarithmic functions (LOG) in both single and double precision.
<b>More Commands</b>	Support is provided for conditional repeating commands (WHILE/WEND), IF commands enabling structured programming (@IF..ENDIF), and commands that record the interrupted line number when an interrupt occurred (INTRB).
<b>Full BASIC Debugging</b>	<p>The following commands are also supported: BRKPT commands to suspend operation at specified lines, single or multiple line STEP commands, and WATCH commands to display the contents of variables when operation is suspended. This makes it easy to confirm that the BASIC program is running correctly.</p> <p>A TRON command to monitor tracing of the execution lines is also available. The results of this trace are saved in the trace buffer, and can be monitored afterwards by means of the TRACE command. This allows checking the operation of the BASIC program.</p>
<b>Error Log</b>	Up to 30 error records are stored in an error log. This is useful for troubleshooting.

## 1-2 System Configuration

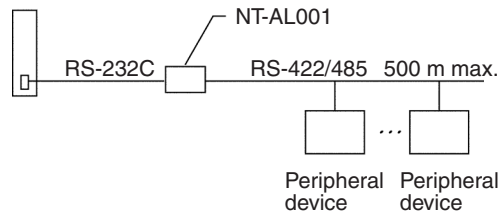
The C200H-ASC11/ASC21/ASC31 ASCII Units are Special I/O Units for the C200H, C200HS, C200HX/HG/HE, and CS1-series Programmable Controllers.

The ASCII Unit's communications port #1 (C200H-ASC11/ASC21) or terminal port (C200H-ASC31) can be connected to a computer terminal using an RS-232C cable to input BASIC commands, compile BASIC programs, debug, and display RUN results.

The ASCII Unit's communications port can be connected to a peripheral device using an RS-232C cable or twisted-pair cable (RS-422A/485) to send/receive communications frames and other data via the BASIC program.



**Note** Use an NT-AL001 Link Adapter (RS-232C to RS-422/485) to connect more than one terminal/device to an RS-232C port (except for the terminal port, port 3, on C200H-ASC31).



## Connectable CPU Units

PC	CPU Unit	Total no. of Units on CPU Rack, Expansion I/O Racks, and SYSMAC BUS Slave Racks	Mounting restrictions
C200HX/HG/HE	C200HE-CPU11-E/32-E/42-E/11-ZE/32-ZE/42-ZE C200HG-CPU33-E/43-E/33-ZE/43-ZE C200HX-CPU34-E/44-E/34-ZE/44-ZE	10	None
	C200HG-CPU53-E/63-E/53-ZE/63-ZE C200HX-CPU54-E/64-E/54-ZE/64-ZE/65-ZE/85-ZE	16	
C200HS	C200HS-CPU01-E/01-EC/21-E/21-EC/31-E/03-E/23-E/33-E	10	
CS1	CS1H-CPU66-E/65-E/64-E/63-E CS1G-CPU45-E/44-E/43-E/42-E	16	None
C200H	C200H-CPU01-E/02-E/03-E/11-E/21-E/23-E/31-E	10	Do not mount Unit to rightmost two slots on CPU Rack.

## Restrictions on SYSMAC BUS Slave Racks

The following restrictions apply when mounting Units to SYSMAC BUS Slave Racks.

Group	A	B	C	D
Unit	ASCII Unit, High-speed Counter Unit, Position Control Unit (NC111/112/113/213) Analog I/O Unit, ID Sensor Unit, Fuzzy Logic Unit	Multi-point I/O Unit	Temperature Sensor Unit, Voice Unit	Position Control Unit (NC211) Motion Control Unit
Number of Units mountable for each group per Remote I/O Master Unit	4 Units	8 Units	6 Units	2 Units
Number of Units mountable for all groups combined per Remote I/O Master Unit	$3A + B + 2C + 6D \leq 12$ , and $A + B + C + D \leq 8$			

**Note** The ASCII Unit can also be mounted to an C200H Expansion Rack connected to a Mini H-type PC (C20H/C28H/C40H/C60H).

## 1-3 Internal Configuration

ASCII Units are each allocated 10 words in the IR/CIO area of the PC. These allow the exchange of data read/write triggers, bit data, status, and interrupts with the CPU Unit during I/O refreshing without the need for special programming.

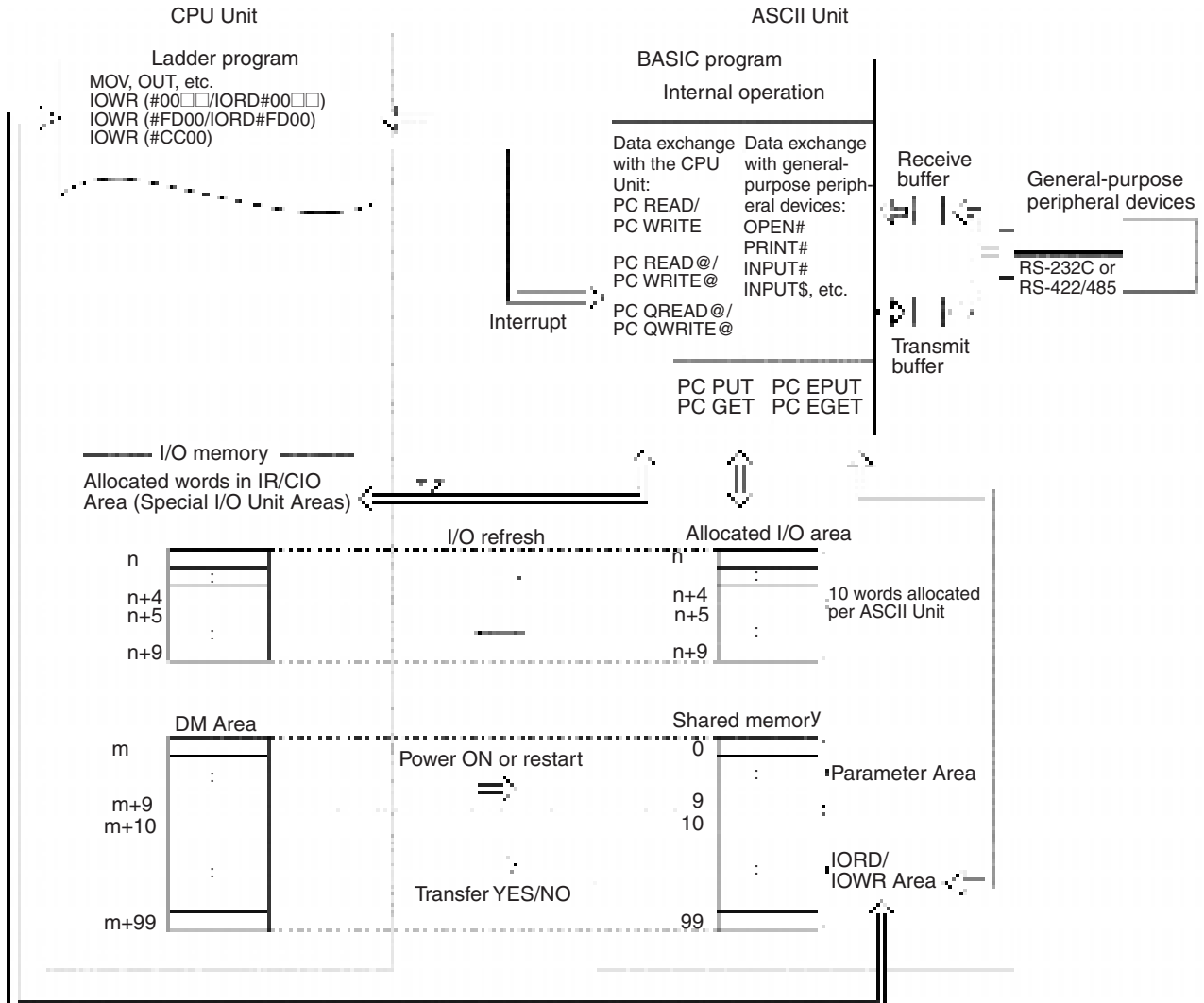
Each Unit is also allocated 10 words as a Setup Area in the data memory area (DM area). The Setup Area contains operating parameters, such as the communications parameters for communications ports and the starting program number. These parameters are automatically transferred from the DM area to the ASCII Unit when the power supply is turned on or the ASCII Unit is restarted.

Internally, the ASCII Unit provides 100 words of shared memory with the CPU Unit (10 Setup Area words and 90 general-purpose words). The ASCII Unit and the CPU Unit can access these areas independently (asynchronously).

Data read/write that does not depend on the cycle time can be executed using the IOWR (SPECIAL I/O UNIT WRITE) or IORD (SPECIAL I/O UNIT READ) instructions from the CPU Unit (C200HX/HG/HE or CS1).

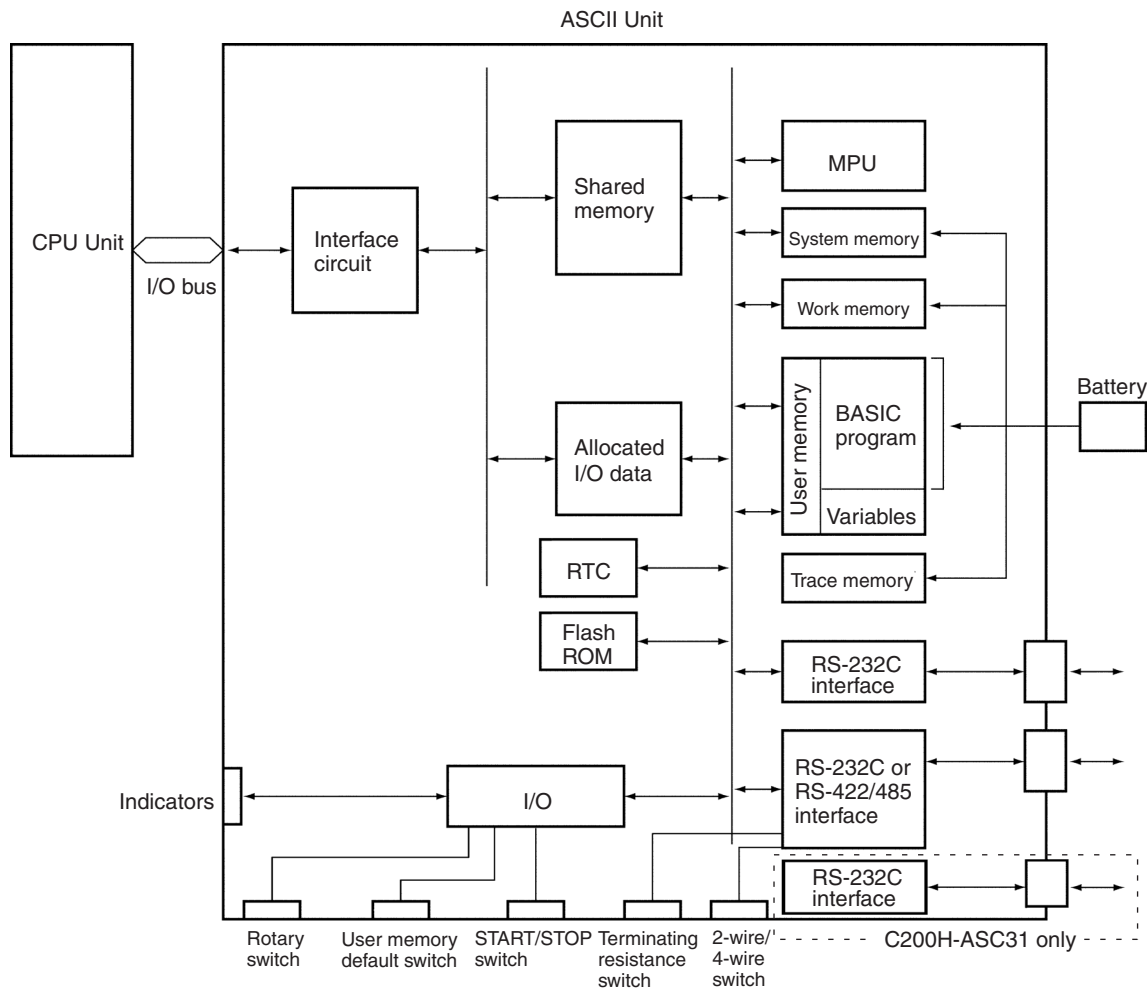
Interrupts that do not depend on the cycle time can be sent from the CPU Unit to the ASCII Unit using IOWR (SPECIAL I/O UNIT WRITE) instructions (C200HX/HG/HE or CS1).

Data Exchange





Internal Layout



- Note**
1. All memory, including RTC, is backed up by battery.
  2. Work memory is part of user memory.
  3. Trace memory is part of system memory.

## 1-4 Specifications

**General Specifications**

Conform to those of the SYSMAC C200H, C200HS, C200HX/HG/HE (-Z), and CS1-series CPU Units.

**Functional and Performance Specifications**

Item		Specifications		
Model		C200H-ASC11	C200H-ASC21	C200H-ASC31
Communications port	Port 1	RS-232C (peripheral device or terminal connection)	RS-232C (peripheral device or terminal connection)	RS-232C (peripheral device connection)
	Port 2	RS-232C (peripheral device connection)	RS-422A/485 (peripheral device connection)	RS-232C (peripheral device connection)
	Terminal port	None	None	RS-232C (terminal connection)

Item		Specifications		
Model		C200H-ASC11	C200H-ASC21	C200H-ASC31
Communications parameters	Communications mode	Half duplex (full duplex available for RS-232C)		
	Synchronization	Start-stop		
	Baud rate	300/600/1,200/2,400/4,800/9,600/19,200/38,400 bps (terminal port does not support 38,400 bps) <b>Note</b> Using interrupts will limit the baud rates that can be used.		
	Transmission mode	RS-232C:	Point-to-point (1:1)	
		RS-422A/485:	Point-to-multipoint (1:N)	
	Transmission distance	RS-232C:	15 m	
		RS-422A/485:	500 m	
Communications buffers	Receive buffer:	512-bytes		
	Transmit buffer:	512-bytes		
Response flow control	RTS/CTS flow (for receive buffer of external device) <b>Note</b> ASCII also asserts RTS when sending data. Xon, Xoff flow (for receive buffers of external device and ASCII Unit)			
Program	Language	BASIC		
	No. of tasks	Single task		
Memory capacity	User program memory area (BASIC program + variable area)	200 Kbytes Contents: BASIC source program (up to 4 programs can be loaded) intermediate code, variables. <b>Note</b> The BASIC program has battery backup. Make sure the battery is installed before operating.		
	Flash ROM	200 Kbytes The BASIC program can be stored in the flash ROM from the user memory area. It can be loaded into the user memory area when the power supply is turned on or the Unit is restarted.		
CPU Unit and data exchange areas	Allocated I/O words	Special I/O Unit Area: 10 words		
	Shared memory area	100 words (no battery backup) as follows: 10 word: Setup Area (can be transferred from the DM area words allocated in the CPU Unit to the ASCII Unit when the power supply is turned on or when the Unit is restarted). 90 word General-purpose area (IORD/IOWR area). <b>Note</b> Read/write is possible using IORD/IOWR instructions from the CPU Unit or PC_EPUT/PC_EGET commands from the ASCII Unit. The position and size of the IORD area/IOWR area can be adjusted. When the power is turned on or the Unit is restarted, the data in the DM words allocated in the CPU Unit can be transferred to the ASCII Unit. (The PC_EPUT and PC_EGET commands are not supported by the C200H and C200HS.)		

Item		Specifications		
Method of data exchange with CPU Unit		Item	CPU Unit instructions	ASCII Unit commands
		ASCII Unit reads/writes I/O memory on trigger from CPU Unit.	MOV, LD, etc.	PC READ/PC WRITE 255 words max. per command.
		ASCII Unit independently reads/writes I/O memory.	Not required	PC READ@/PC WRITE@ 255 words max. per command.
		Read/write allocated IR/CIO area words.	MOV, LD etc.	PC PUT/PC GET 1 word per command for PC PUT; 2 words for PC GET.
		CPU Unit executes IOWR/IORD(#FD00) on trigger from ASCII Unit.	IOWR (#FD00) /IORD (#FD00) <b>Note</b> C200HX/HG/HE or CS1. 128 words max. per instruction.	PC QREAD/PC QWRITE 128 words max. per command.
		CPU Unit and ASCII Unit read/write to shared memory asynchronously.	IOWR (#00□□) /IORD (#00□□) <b>Note</b> C200HX/HG/HE or CS1. 90 words max. per instruction.	PC EPUT/PC EGET 90 words max. per command.
RTC		Year, month, day, date, hour, minute, second (leap year adjusted for) Accuracy: At 25°C: +70 s/-45 s per month At 0°C: +5 s/-110 s per month At 55°C: +0/-140 s per month Battery backup. Can be synchronized with the CPU Unit time.		
Battery	Model	C200H-BAT09		
	Life	5 years at 25°C (The life of the battery is shortened if the ASCII Unit is used at higher temperatures.)		
Diagnostic functions		CPU watchdog timer, battery voltage drops		
Accessories		Connector plug: XM2A-0901 or equivalent (C200H-ASC11/21: 2, C200H-ASC31: 3) Hood: XM2S-0911 or equivalent (C200H-ASC11/21: 2, C200H-ASC31: 3)		
Internal current consumption (See note.)		250 mA max. at 5 VDC	300 mA max. at 5 VDC	300 mA max. at 5 VDC
Dimensions		130 x 35 x 100.5 mm (H x W x D)		
Weight		400 grams max.		

- Note**
1. If an NT-AL001 RS-232C/RS-422 Link Adapter is used at the RS-232C port, an additional 150 mA of current will be consumed.
  2. Always connect the battery to the ASCII Unit before using it.

## 1-5 Comparison with Previous ASCII Units

The following table shows a comparison between the C200H-ASC11/ASC21/ASC31 and the C200H-ASC02 ASCII Units. Before using the ASCII Unit, refer to Appendix A Operating Precautions, and for further details, refer to Appendix B Comparison with C200H-ASC02.

Item	C200H-ASC02	C200H-ASC11 /ASC21/ASC31	Comments
<b>Applicable PCs</b>	C200H, C200HS, CS1-series PCs	C200HX/HG/HE,	---
<b>Unit No.</b>	0 to 9	0 to 15	---
<b>MPU</b>	8 bits	32 bits	---
<b>Interfaces (ports)</b>	Two RS-232C	Two RS-232C OR One RS-232C and one RS-422A/485 OR One terminal port and two RS-232C	---
<b>Baud rate</b>	19,200 bps max.	38,400 bps max.	---
<b>User program memory area (BASIC program + variable area)</b>	24 Kbytes	200 Kbytes	---
<b>BASIC program storage</b>	EEPROM: 24 Kbytes	Flash ROM: 200 Kbytes	---
<b>Number of BASIC programs in user area of Unit</b>	Up to 3	Up to 4	---
<b>Number of BASIC program lines</b>	0 to 63,999	1 to 65,535	---
<b>ASCII Unit system settings</b>	DIP switch on back of Unit	Setup Area (10 words) (transferred from DM area at power ON or restart)	---
<b>Allocated IR/CIO area words</b>	4 words	10 words	---
<b>Shared memory</b>	None	Yes (general-purpose words: 90 words)	---

Item		C200H-ASC02	C200H-ASC11 /ASC21/ASC31	Comments	
Com- mands/ Instruc- tions	ASCII Unit BASIC com- mands for data exchange with CPU Unit	PC READ/ PC WRITE	Yes	Yes (multiple format strings supported)	On trigger from CPU Unit, ASCII Unit reads/writes I/O memory.
		PC READ@/ PC WRITE@	Yes	Yes (multiple format strings supported)	ASCII Unit reads/ writes I/O memory independently.
		PC QREAD/ PC QWRITE	None	Yes	On trigger from ASCII Unit, CPU Unit reads/ writes via IOWR/ IORD instructions.
		PC PUT/ PC GET	Yes	Yes	Reads/writes allo- cated IR/CIO area words.
		PC EPUT/ PC EGET	None	Yes	Reads/writes its own shared memory.
	Exchanging data with ASCII Unit from CPU Unit	IOWR/IORD instruc- tions (read/write to ASCII Unit's shared memory, IR/CIO area transfer, and interrupts)	None	Yes (C200HX/HG/HE or CS1)	---
	Error check code calculation instruction		None	Yes (FCS)	---
	Bit ON/OFF command		None	Yes (BITON, BITOFF)	---
	Conditional repeat command		None	Yes (WHILE/WEND)	---
	Label branching		None	Yes	---
	Integer search		None	Yes (SEARCH)	---
	Debugging function	Break point command	None	Yes (BRKPT)	---
		Trace command	None	Yes (TRACE)	---
		Variable display com- mand	None	Yes (WATCH)	---
	Transmission control code: Control command		None	Yes (RTS/DTR con- trol, CTS/DSR moni- tor)	---
Special opera- tions	Exponential function	None	Yes	---	
	Square root	None	Yes	---	
Floating point calculations		Single	Double	---	
Machine language program		Yes	None	---	

Item		C200H-ASC02	C200H-ASC11 /ASC21/ASC31	Comments	
Inter- rupts	PC interrupts	Using allocated IR/CIO area words	Yes	Yes	---
		Using IOWR (CC##) instruction	None	Yes	---
		Maximum number	15	99	---
	Communica- tions interrupts (via received data)	Interrupts when data is received in the receive buffer	Yes	Yes	---
		Interrupt on reception of designated character	None	Yes	---
		Interrupt on reception of designated number of bytes	None	Yes	---
		Interrupt on reception of designated start and end characters	None	Yes	---
		Interrupt on reception of designated start character and designated number of bytes	None	Yes	---
	Key interrupts		Yes	Yes	---
	Timed interrupts		None	Yes	---
	Interval timer interrupts		None	Yes	---
	One-shot timer interrupts		None	Yes	---
	Error interrupts		Yes	Yes	---
Line code storage command on interrupt		None	Yes (INTRB, INTRR, INTRS)	---	
Error log		None	Yes (Can record up to 30 error records. Time and date of occurrence not recorded.)	---	
Internal current consumption		5 VDC 200 mA	5 VDC 250 mA or 300 mA		

## 1-6 Basic Operating Procedures

### 1-6-1 Editing BASIC Programs

The following example uses WINDOWS 95 on a personal computer as a terminal (hyperterminal VT-100 emulation).

- 1,2,3...
1. Mount the ASCII Unit to the Backplane. Refer to 2-2 Installation.
  2. Connect the Unit's terminal port to the personal computer's RS-232C port via an RS-232C cable. Refer to 2-2 Installation.
  3. Set the switches on the front of the Unit.
  4. Turn on the PC power supply.
  5. Perform the following steps if the ASCII Unit's default setup is used.
    - a) Clear all the DM words allocated to the ASCII Unit in the CPU Unit using a PC Peripheral Device (such as a Programming Console).
    - b) Toggle the PC power switch off and then back on, or restart the ASCII Unit.

6. Perform the following steps if the ASCII Unit's default setup is not being used.
  - a) Set the DM words allocated to the ASCII Unit in the CPU Unit using a PC peripheral device (such as Programming Console).
  - b) Select whether to set the terminals ports' communications parameters to the default settings or not, and whether to set the terminal emulation to VT100 or not. Refer to 3-2 DM Area Allocations.
  - c) Toggle the PC power switch off and then back on, or restart the ASCII Unit.
7. In WINDOWS 95, click **Start** and then point to **Programs**.
8. Point to **Accessories** and then click **Hyperterminal**. Refer to 7-1 Programming Procedure.
9. Start the hyperterminal, and specify the connection settings and properties.
10. Press the Ctrl+X Keys on the computer keyboard.

#### Inputting a BASIC Program Directly into the ASCII Unit

- 1,2,3...
1. Input **New** to clear the ASCII Unit's program.
  2. Input **Auto** to automatically display the line numbers.
  3. Input the BASIC program and press the Enter Key.

#### Transferring a BASIC Program from a Text Editor to the ASCII Unit

- 1,2,3...
1. Input a program using a text editor such as Note Pad, and save it in text format.
  2. Input **New** to clear the ASCII Unit's program.
  3. Input the LOAD command.
  4. Point to **Hyperterminal** and then **Transfer**, and click **Transmit text file** to transfer the saved file to the ASCII Unit. Specify a transmission delay after each line when transmitting data (or use XON/XOFF flow control).
  5. Press the Ctrl+C Keys on the computer keyboard.

#### Transferring the ASCII Unit's BASIC program to the Personal Computer

- 1,2,3...
1. Using the SAVE command, read the program. Input SAVE #□, and press the Enter Key.
  2. Point to **Hyperterminal** and then **Transfer**, and click **Capture Text**.
  3. Input the file name to save the program and switch to reception mode.
  4. Press the Ctrl+C Keys to begin the transfer.
  5. Point to **Hyperterminal**, **Transfer**, and then **Capture Text**, and then click **STOP**.
  6. Press the Ctrl+C Keys.

## 1-6-2 BASIC Programming Procedure

- 1,2,3...
1. Study communications procedures with the external devices. For further details on creating flowcharts for communications procedures, refer to SECTION 4 Data Exchange with General-purpose External Devices.
  2. Study the method of data exchange with the CPU Unit, including the following:
    - Data exchange timing
    - Data exchange trigger to be used (i.e., whether to trigger the exchange from the CPU Unit or the ASCII Unit)
    - Interrupts from the CPU Unit
    - Data transfer method based on the volume of dataFor further details, refer to SECTION 6 Data Exchange with the CPU Unit.
  3. Study BASIC internal processing. For further details, refer to SECTION 5 Processing.
  4. Write the program. Refer to 7-4 Program Configuration, 7-5 List of BASIC Commands, and 7-6 BASIC Commands.
  5. Debug the program. Refer to 7-3 Starting/Stopping the BASIC Program and 7-8 Debugging.

- Note**
1. When storing the program in flash ROM, enter the ROMSAVE command from the terminal.
  2. When automatically reading the program to user memory at power ON or when the Unit is restarted, set bits 08 to 15 of the DM word m to 5A<sub>hex</sub> in the DM Setup Area allocated to the Unit.
  3. Set the port number to 3 when using the LOAD or SAVE command for the C200H-ASC31.

## 1-6-3 Connection and Transfer to External Devices

- 1,2,3...
1. Mount the ASCII Unit to the Backplane.
  2. Set the following switches on the front of the Unit.
    - Unit number switch
    - RS-422/485 2-wire/4-wire switch
    - RS-422/485 terminating resistance switchRefer to 2-1 Nomenclature and Functions.
  3. Connect the external devices using an RS-232C or RS-422/485 port depending on the type of device. Refer to 2-2 Installation.
  4. Turn on the PC power supply.
  5. Using a PC peripheral device (such as Programming Console) set the DM area words allocated in the CPU Unit. In particular, set DM word m+1 bits 00 to 07 to specify whether the ASCII Unit will run when power is turned on or the Unit is restarted. Refer to 3-2 DM Area Allocations.
  6. Transfer the ladder program to the CPU Unit.
  7. Toggle the PC power off and then back on, or restart the ASCII Unit. The BASIC program will be executed if the ASCII Unit is set to start operation at power ON or when the Unit is restarted (in the DM Setup Area allocated to the Unit).
  8. If the ASCII Unit is set for manual starting at power ON or when the Unit is restarted (in the DM Setup Area allocated to the Unit), perform one of the following steps.



- a) Set the START/STOP switch on the front panel of the Unit to START.
- b) Input RUN from the terminal to run the BASIC program. (The START/STOP switch on the front panel of the Unit must be set to START first.)

- Note**
1. Use one of the following steps to pause the BASIC program.
    - Press the Ctrl+X Keys from the terminal.
    - Set the START/STOP switch on the front panel of the Unit to STOP.
  2. Use one of the following steps to pause the BASIC program at a particular line.
    - Set a break point from the terminal.
    - Place a STOP command in the program.
    - Input the RUN TO line number command from the terminal.

# SECTION 2

## Installation

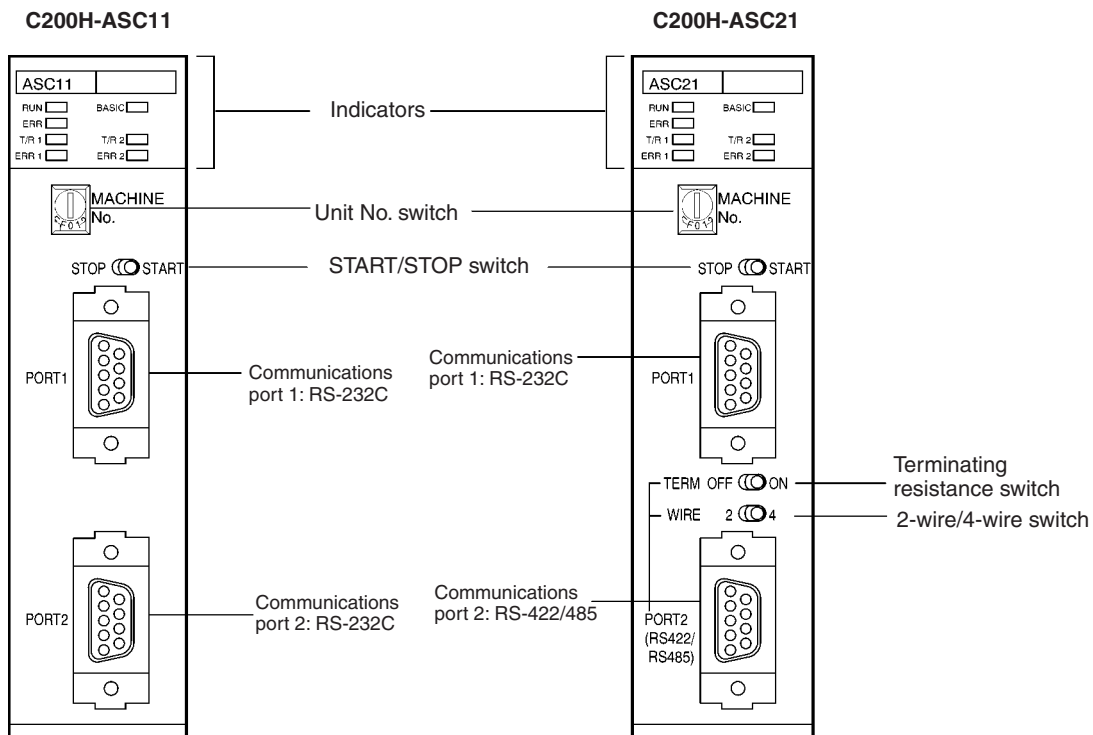
This section describes how to install and connect the ASCII Unit, including mounting to the PC Backplane and connecting to external devices/computers.

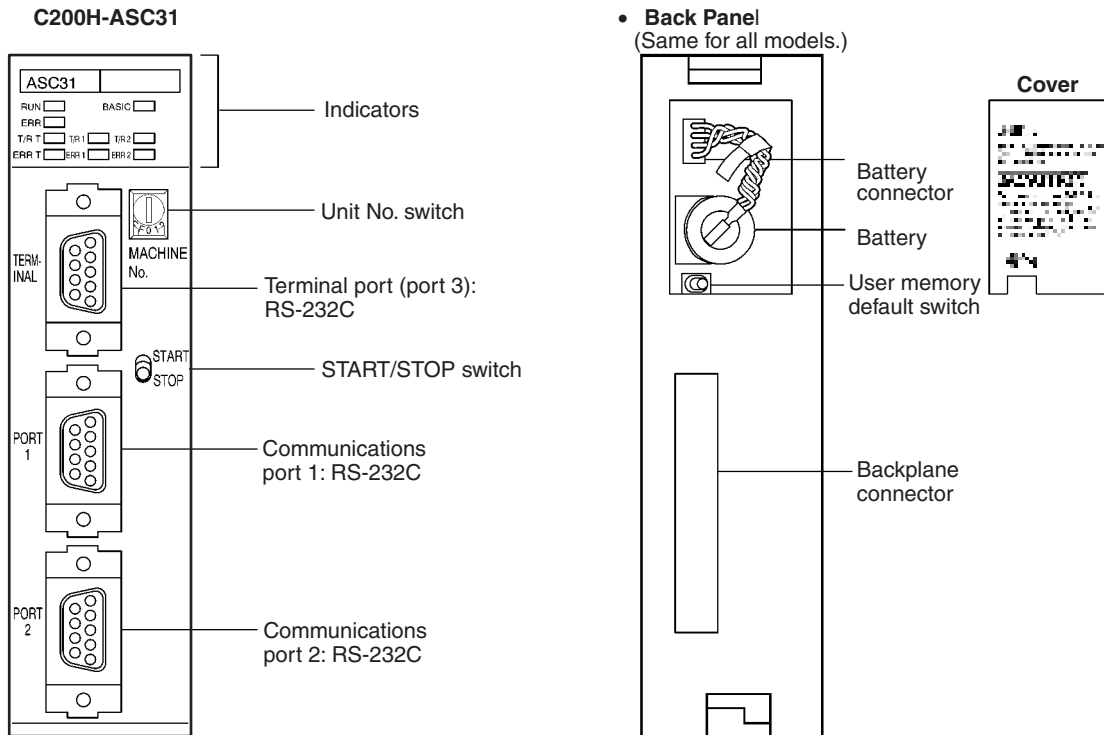
2-1	Nomenclature and Functions .....	18
2-1-1	Names of Parts .....	18
2-1-2	Front Panel .....	19
2-1-3	Rear Panel .....	21
2-2	Installation .....	22
2-2-1	Installation Method .....	22
2-2-2	Connections .....	24
2-2-3	General-purpose Peripheral Device Connections .....	26

## 2-1 Nomenclature and Functions

### 2-1-1 Names of Parts

- Front Panel





**Note** The above diagram shows the Unit with the cover removed. The caution on the cover will be added from Units shipped after November 1998.

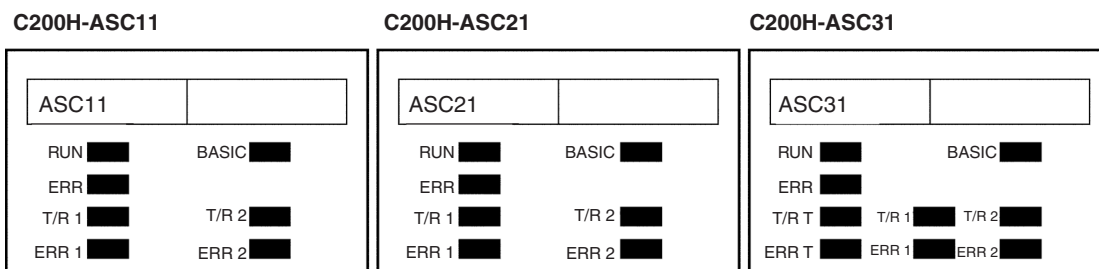
## 2-1-2 Front Panel

### Indicators

Indicator	Color	Name	Display	Meaning	Bit in allocated IR/CIO area words
RUN	Green	Unit operation	Lit	The ASCII Unit is operating normally.	---
			Not lit	An error has occurred. The CPU Unit power is turned OFF, or the MPU in the ASCII Unit is out of control. DM setting is incorrect.	---
BASIC	Green	System mode	Lit	The BASIC program is running.	Bit 7 of n+5
			Slow flashing	INPUT/LINE INPUT/INPUT\$ command standby (awaiting input), (L)PRINT command standby (awaiting transmission conditions), or command input standby.	Bit 7 of n+5 (not for INPUT/LINE INPUT/INPUT\$, and PRINT commands)
			Quick flashing	Transfer standby for the BASIC program. (The flashing cycle will be unstable when transfer of the BASIC program has started.)	---
			Not lit	At power-on waiting for Ctrl+X Key input.	---
ERR	Red	System error/ BASIC error/Battery error	Lit	An error has occurred in the system or the BASIC program.	Bit 2 of n+5
			Flashing	Battery voltage has drop, or battery not connected.	Bit 6 of n+5
T/R 1	Orange	Communications port 1 data transfer	Flashing	Communications port 1 is receiving or transmitting data.	---
T/R 2	Orange	Communications port 2 data transfer	Flashing	Communications port 2 is receiving or transmitting data.	---

Indicator	Color	Name	Display	Meaning	Bit in allocated IR/CIO area words
T/R T	Orange	Terminal port data transfer	Flashing	Terminal port (port 3) is receiving or transmitting data (C200H-ASC31 only).	---
ERR 1	Red	Communications port 1 error	Lit	A reception error (parity error, reception buffer overflow, etc.) has occurred at communications port 1.	Bit 4 of n+5
ERR 2	Red	Communications port 2 error	Lit	A reception error (parity error, reception buffer overflow, etc.) has occurred at communications port 2.	Bit 5 of n+5
ERR T	Red	Terminal port error	Lit	A reception error (parity error, reception buffer overflow, etc.) has occurred at terminal port 1 (C200H-ASC31 only)	Bit 3 of n+5

**Note** If a PROGRAM MEMORY ERROR (BASIC program memory error) is generated, all error indicators (ERR, ERR 1, ERR 2, ERR T) will flash. (Refer to 9-2 Troubleshooting for details.)



**START/STOP Switch**



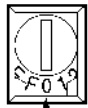
The START/STOP switch starts and stops the BASIC program. If STOP is selected while the program is running, the program will STOP, and if START is selected during a STOP, the program will be restarted from the first line. The effect on operations in relation to initial settings and switch positions are shown in the following table. When START/STOP switch is turned OFF, the program will not run even if the RUN command is sent from the terminal or automatic starting is set in the initial settings.

Operation		Power on or restart		Stopped			Running	
				RUN command from terminal		Switch changed to START	Ctrl+X pressed	Switch changed to STOP
Switch position		START	STOP	START	STOP	STOP to START	START	START to STOP
Startup mode (See note.)	Manual start (00)	No effect	No effect	RUN	No effect	RUN	Break	Break
	Auto start (5A)	RUN	No effect	RUN	No effect	RUN	Break	Break

**Note** Either manual or automatic starting is set in bits 00 to 07 of word m + 1 in the DM area words allocated to the Unit.

**Unit No. Switch**

Set the unit number to between 0 and F. If the same number is set for other Special I/O Units, an I/O Unit Overflow error (fatal) will occur, and the system will not operate.



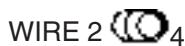
**MACHINE No.**

Indicates the Unit No.

CPU Unit	Unit No. setting range
C200H-CPU01-E/02-E/03-E C200H-CPU21-E/22-E/23-E/31-E, C200HS-CPU01-E/21-E/31-E/03-E/23-E/33-E, C200HE-CPU11-E/32-E/42-E/11-ZE/32-ZE/42-ZE, C200HG-CPU33-E/43-E/33-ZE/43-ZE, C200HX-CPU34-E/44-E/34-ZE/44-ZE	0 to 9
C200HG-CPU53-E/63-E/53-ZE/63-ZE, C200HX-CPU54-E/64-E/54-ZE/64-ZE/65-ZE/85-ZE CS1H-CPU66-E/65-E/64-E/63-E CS1G-CPU45-E/44-E/43-E/42-E	0 to F

**2-wire/4-wire Switch (C200H-ASC21 Only)**

On the C200H-ASC21, the RS-422A/485 port can be set to either 2-wire or 4-wire communications.



2: 2-wire communications  
4: 4-wire communications

**Terminating Resistance Switch (C200H-ASC21 Only)**

On the C200H-ASC21, the RS-422A/485 port's terminating resistance (220 Ω) can be turned ON or OFF.



ON: Terminating resistance connected.  
OFF: Terminating resistance disconnected.

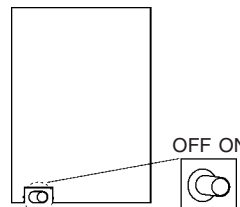
**2-1-3 Rear Panel**

**User Memory Default Switch**

When an error occurs in the ASCII Unit as a result of damage to the user memory, this switch can be used to forcibly format (delete all programs from memory) the user memory and return the ASCII Unit to normal operating status.

**Note** During normal operation, make sure that the user memory switch is turned OFF (set to the left). If the switch is turned ON (set to the right) and the power turned ON, all programs in the ASCII Unit will be deleted. Refer to 9-2 Troubleshooting for details on processing errors.

ON: All user memory will be formatted (returned to default settings)  
OFF: User memory will not be formatted.



**Battery**

The battery backs up the ASCII Unit's BASIC program, trace memory, and real-time clock settings when power is not being supplied. If the battery voltage drops, or the battery is not connected, the Error Indicator on the front panel will flash.

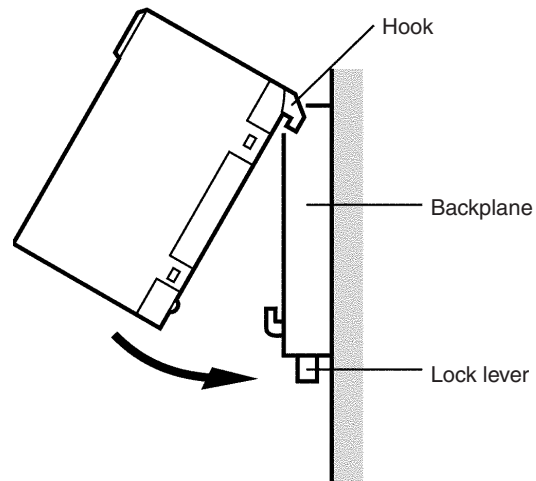
Part name (replacement part)	Model
Battery Set	C200H-BAT09

Refer to 9-5 Maintenance for details on replacing the battery.

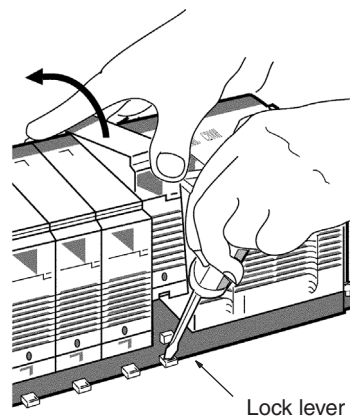
## 2-2 Installation

### 2-2-1 Installation Method

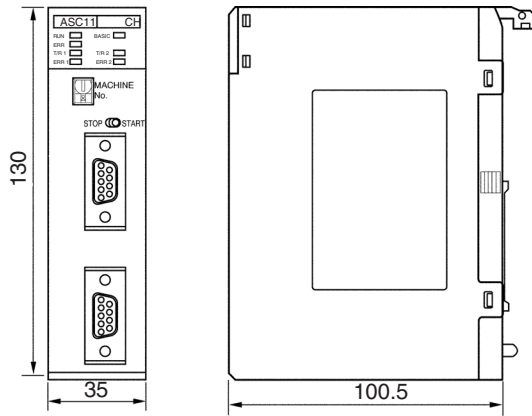
- 1,2,3... 1. Attach the hooks on the upper section of the ASCII Unit onto the Backplane.
2. Insert the ASCII Unit connector into the Backplane connector.



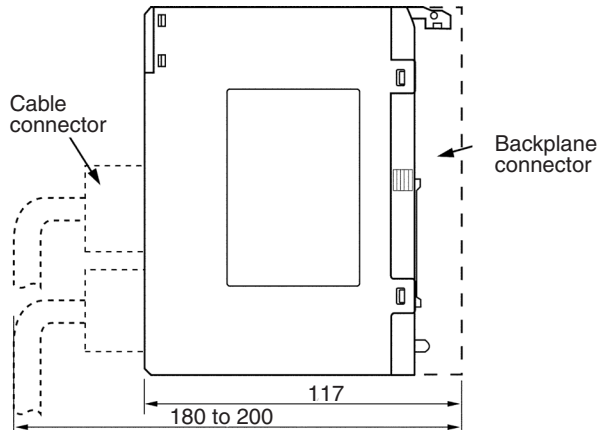
**Note** When removing the ASCII Unit, lift it out while pressing down on the lock lever with a screwdriver, as shown in the following illustration.



Dimensions



Unit Dimensions when Mounted



Mounting Positions and Number of Units

PC	CPU Unit	Total No. of Units on CPU Rack, Expansion I/O Racks, and SYSMAC BUS Slave Racks	Mounting restrictions
C200HX/HG/HE	C200HE-CPU11-E/32-E/42-E/11-ZE/32-ZE/42-ZE C200HG-CPU33-E/43-E/33-ZE/43-ZE C200HX-CPU34-E/44-E/34-ZE/44-ZE	10	None
	C200HG-CPU53-E/63-E/53-ZE/63-ZE C200HX-CPU54-E/64-E/54-ZE/64-ZE/65-ZE/85-ZE	16	
C200HS	C200HS-CPU01-E/01-EC/21-E/21-EC/31-E/03-E/23-E/33-E	10	
CS1	CS1H-CPU66-E/65-E/64-E/63-E CS1G-CPU45-E/44-E/43-E/42-E	16	None
C200H	C200H-CPU01-E/02-E/03-E/11-E/21-E/23-E/31-E	10	Do not mount Unit to rightmost two slots on CPU Rack.

Restrictions on SYSMAC BUS Slave Racks

The following restrictions apply when mounting Units to SYSMAC BUS Slave Racks.

Group	A	B	C	D
Unit	ASCII Unit, High-speed Counter Unit, Position Control Unit (NC111/112/113/213) Analog I/O Unit, ID Sensor Unit, Fuzzy Logic Unit	Multi-point I/O Unit	Temperature Sensor Unit, Voice Unit	Position Control Unit (NC211) Motion Control Unit
Number of Units mountable for each group per Remote I/O Master Unit	4 Units	8 Units	6 Units	2 Units
Number of Units mountable for all groups combined per Remote I/O Master Unit	$3A + B + 2C + 6D \leq 12$ , and $A + B + C + D \leq 8$			



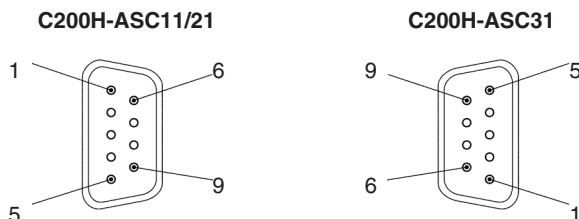
## 2-2-2 Connections

### External Devices

External devices are connected to the RS-232C or RS-422A/485 port. For interface code/timing, refer to SECTION 4 Data Exchange with General-purpose External Devices.

#### RS-232C Connections (C200H-ASC11/21/31)

Using the enclosed connector, connect all pins to match the external device interface.



Pin	Symbol	Name	Direction	Comments
1	FG	Frame ground	-	
2	SD	Send data	Output	
3	RD	Receive data	Input	
4	RTS	Request to send	Output	
5	CTS	Clear to send	Input	
6	-	Not used	-	Only for terminal port of C200H-ASC31.
	5V	5V output	Output	Other ports
7	DSR	Data set ready	Input	
8	DTR	Data terminal ready	Output	
9	SG	Signal ground	-	
Hood	FG	Frame ground	-	

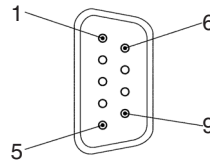
#### RS-232C Interface Specifications

Item	Specifications	
Electrical characteristics	Conform to EIA RS-232C	
Synchronization	Start-stop	
Baud rate	300/600/1,200/2,400/4,800/9,600/19,200/38,400 bps (38,400 bps cannot be used for the terminal port on the ASC31.)	
Transmission mode	Point-to-point (1:1)	
Communications buffers	Transmit: 512 bytes; receive: 512 bytes	
Flow control	RTS/CTS flow, Xon/Xoff flow control supported	
Connector type	D-sub 9-pin for both ports	
Applicable connector (enclosed)	Plug: XM2A-0901 or equivalent Connector hood: XM2S-0911 or equivalent	
Recommended cable	UL2464 AWG28x5P IFS-RVV-SB (UL product) AWG28x5P IFVV-SB (non-UL product)	Fujikura Ltd.
	UL2464-SB(MA) 5Px28AWG (7/0.127) (UL product) CO-MA-VV-SB 5Px28AWG (7/0.127) (non-UL product)	Hitachi Cable, Ltd.
Cable length	15 m max.	

**Note** For details on connecting cables, refer to Appendix F Wiring RS-232C or RS-422A/485 Cable Connectors.

**RS-422A/485 Connections (C200H-ASC21 Only)**

Using the enclosed connector, connect the pins to match the external device interface.



Pin	Symbol	Name	Direction
1	SDA	Send data A (-)	Output
2	SDB	Send data B (+)	Output
3	NC	Not used	-
4	NC	Not used	-
5	NC	Not used	-
6	RDA	Receive data A (-)	Input
7	NC	Not used	-
8	RDB	Receive data B (+)	Input
9	NC	Not used	-
Hood	FG	Frame ground	-

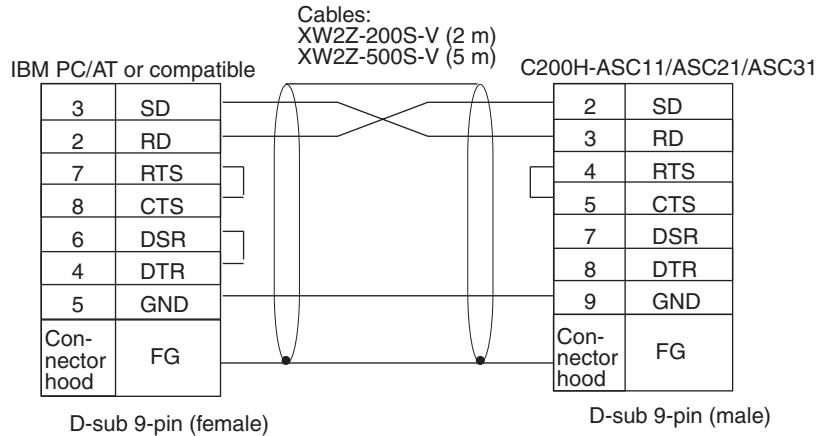
**RS-422A/485 Interface Specifications**

Item	Specifications	
<b>Communications mode</b>	Half duplex (1 ms is required after transmission has been completed until the ASCII Unit is ready to receive.)	
<b>Electrical characteristics</b>	Conform to EIA RS-422A/485	
<b>Synchronization</b>	Start-stop	
<b>Baud rate</b>	300/600/1,200/2,400/4,800/9,600/19,200/38,400 bps	
<b>Transmission mode</b>	Point-to-multipoint (1:N)	
<b>Communications buffer</b>	Transmit: 512 bytes; receive: 512 bytes	
<b>Flow control</b>	None	
<b>Applicable connector (enclosed)</b>	Plug: XM2A-0901 or equivalent Connector hood: XM2S-0911 or equivalent	
<b>Recommended cable</b>	UL2464 AWG28x5P IFS-RVV-SB (UL product) AWG28x5P IFVV-SB (non-UL product)	Fujikura Ltd.
	UL2464-SB(MA) 5Px28AWG (7/0.127) (UL product) CO-MA-VV-SB 5Px28AWG (7/0.127) (non-UL product)	Hitachi Cable, Ltd.
<b>Cable length</b>	500 m max.	

- Note**
1. For details on connecting cables, see Appendix F Wiring RS-232C or RS-422A/485 Cable Connectors.
  2. When the 2/4-wire switch is set for 2-wire operation, use either pins 1 and 2, or pins 6 and 8. (Pins 1 and 6 are short-circuited, and pins 2 and 8 are short-circuited.)

**Personal Computer Used as Terminal**

Check the wiring between the RS-232C connector and the computer, and using the appropriate connectors, prepare the connecting cables using the the pin arrangement shown in the following diagram.

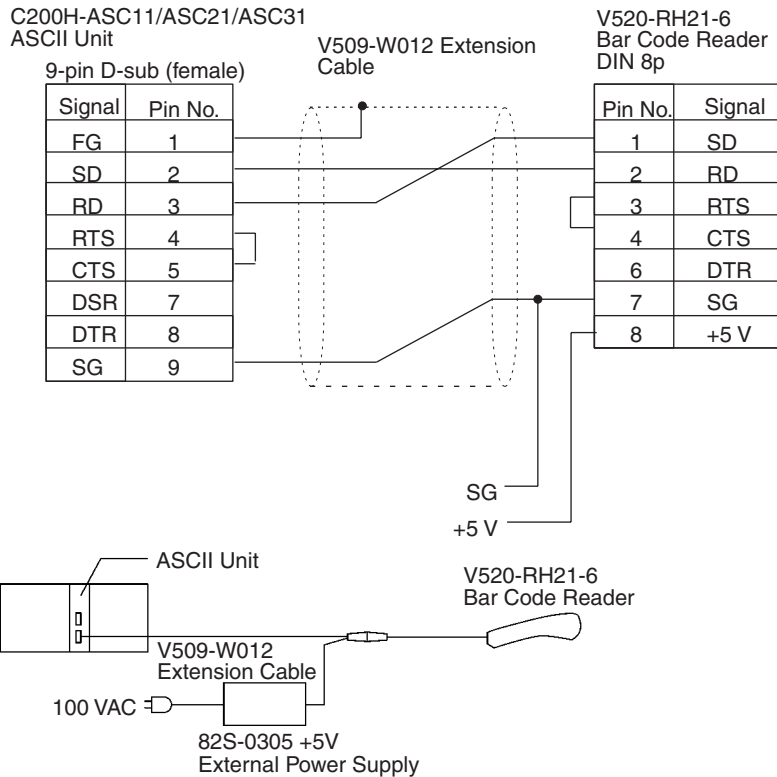


- Note**
1. DSR-DTR signals are not connected back to each other for the cable used above, so do not specify that the DSR signal is checked when the LOAD command is executed. (If the specification is omitted, the DSR signal will not be checked by default.)
  2. CTS-RTS and DSR-DTR are not cross-connected, so if flow control is required, use Xon/Xoff flow control.

### 2-2-3 General-purpose Peripheral Device Connections

#### Bar Code Reader Connected with RS-232C

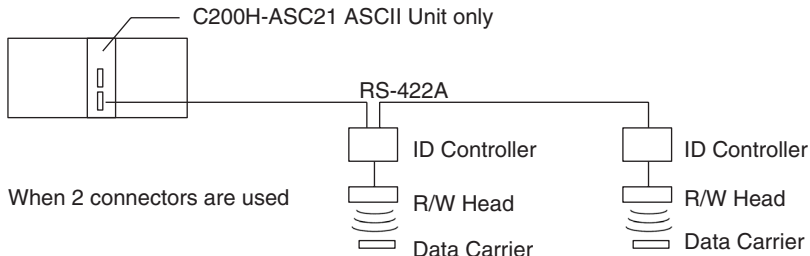
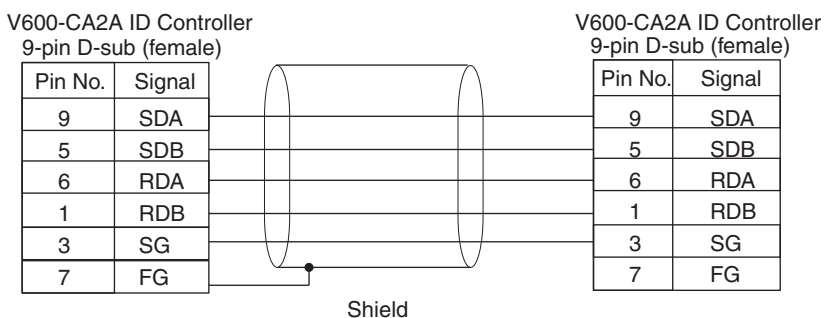
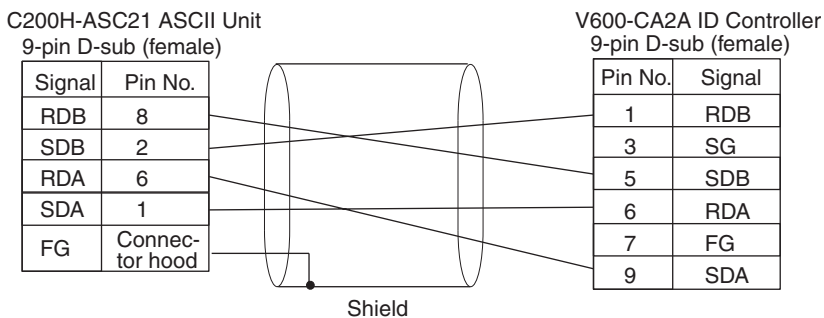
Connecting an OMRON V500 Bar Code Reader



**Note** In order to prevent malfunctions, when external devices have a FG, ground both the external devices and the ASCII Unit with shielded cables.

**ID Controller Connected with RS-422A (4-wire)**

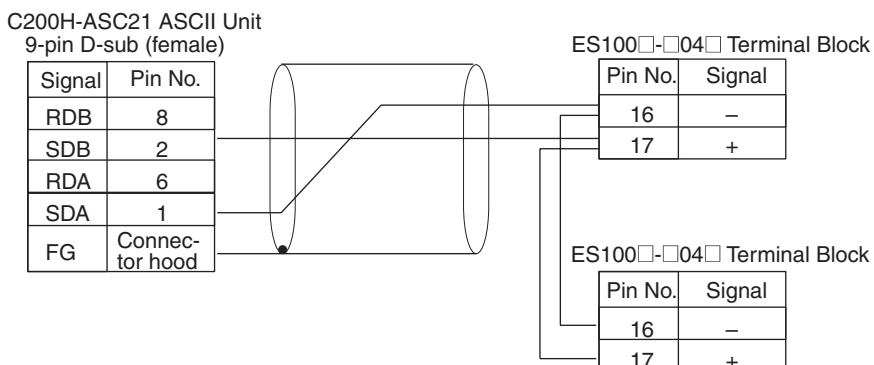
Connecting a V600/620 ID Controller



- Note**
1. Make sure that the shield is grounded at either the ID Controller end or the ASCII Unit end, but not both ends.
  2. Make sure to set the 2/4-wire switch on the C200H-ASC21 Unit to 4 wires.

**Digital Operator Connected with RS-485 (2-wire)**

Connecting an OMRON ES100X Digital Operator



- Note**
1. Make sure that the shield is grounded at either the Digital Operator end or the ASCII Unit end, but not both ends.
  2. Make sure to set the 2/4-wire switch on the C200H-ASC21 Unit to 2 wires.

# SECTION 3

## IR/CIO and DM Area Allocations

This section describes the methods used to allocate words to the ASCII Unit in the IR/CIO and DM Areas and the applications of the bits and words in these areas.

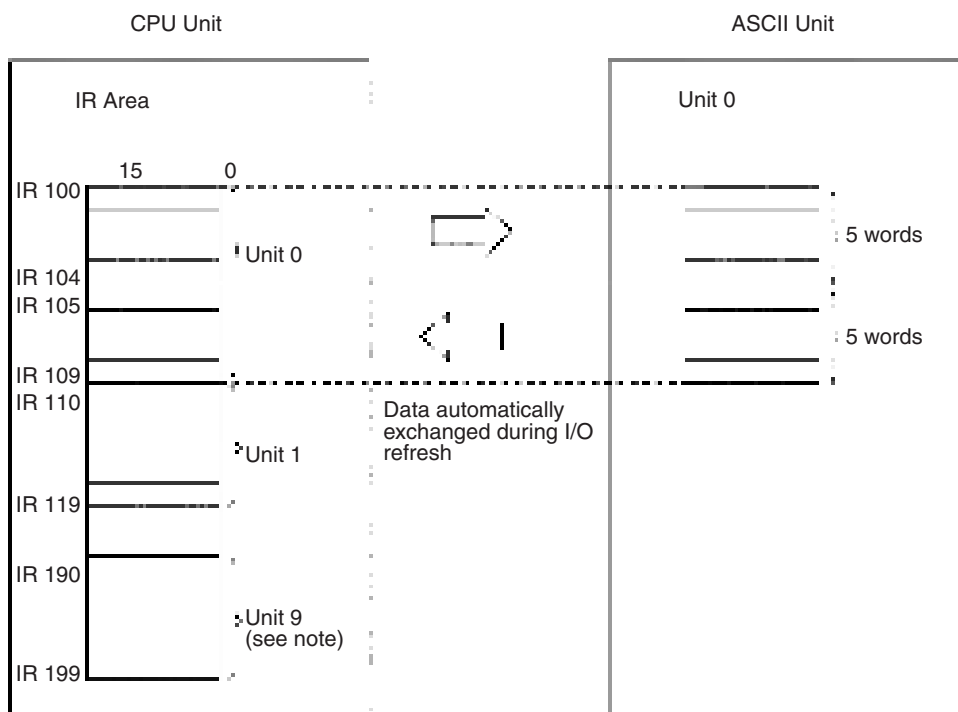
3-1	IR/CIO Area Allocations . . . . .	30
3-1-1	Overview . . . . .	30
3-1-2	Overview of IR/CIO Area Allocations . . . . .	31
3-2	DM Area Allocations . . . . .	38
3-2-1	Overview . . . . .	38
3-2-2	Overview of DM Allocations . . . . .	40

### 3-1 IR/CIO Area Allocations

#### 3-1-1 Overview

Each ASCII Unit is allocated 10 words in the Special I/O Unit Areas of the CPU Unit's IR or CIO Area. The words that are allocated depend on the Unit No. set on the rotary switch on the front panel of the ASCII Unit. The contents of the allocated 10 words is exchanged automatically between the CPU Unit and the ASCII Unit every time the CPU Unit refreshes I/O.

#### Data Exchange for the C200HX/HG/HE, C200H, C200HS



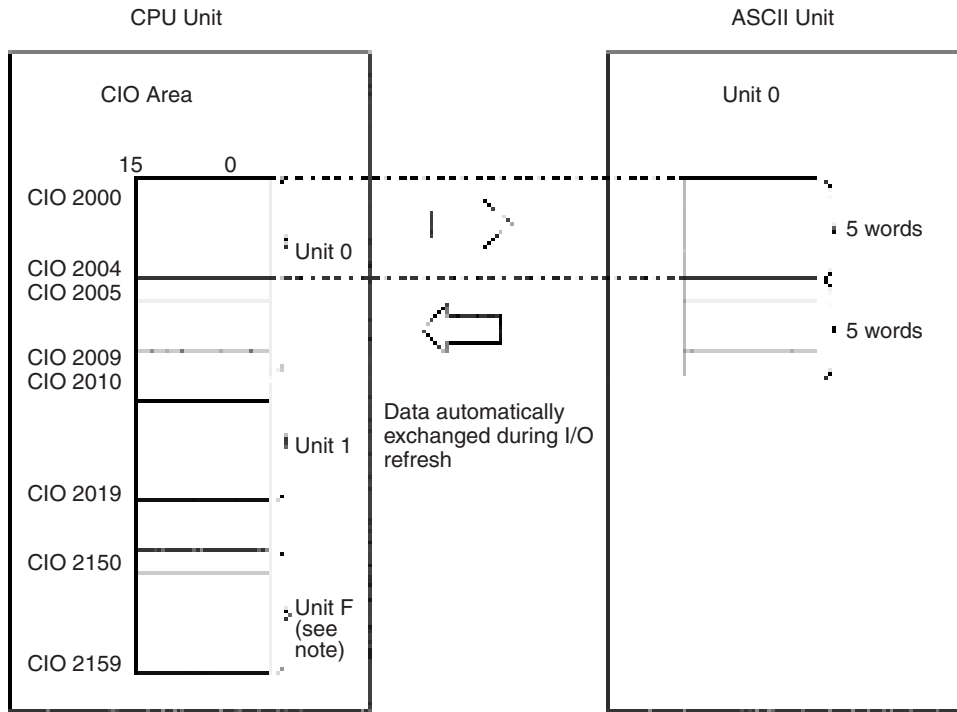
**Note** For CPU Units C200HG-CPU53-E/63-E/53-ZE/63-ZE and C200HX-CPU54-E/64-E/54-ZE/64-ZE/65-ZE/85-ZE, IR 400 to IR 459 are also allocated (maximum of 16 Special I/O Units).

#### Special I/O Area Allocation for the C200HX/HG/HE, C200H, C200HS

Unit No.	Allocated words	Unit No.	Allocated words	Unit No.	Allocated words	Unit No.	Allocated words
0	IR 100 to IR109	4	IR 140 to IR 149	8	IR 180 to IR 189	C	IR 420 to IR 429 (See note.)
1	IR 110 to IR 119	5	IR 150 to IR 159	9	IR 190 to IR 199	D	IR 430 to IR 439 (See note.)
2	IR 120 to IR 129	6	IR 160 to IR 169	A	IR 400 to IR 409 (See note.)	E	IR 440 to IR 449 (See note.)
3	IR 130 to IR 139	7	IR 170 to IR 179	B	IR 410 to IR 419 (See note.)	F	IR 450 to IR 459 (See note.)

**Note** For C200HG-CPU53-E/63-E/53-ZE/63-ZE and C200HX-CPU54-E/64-E/54-ZE/64-ZE/65-ZE/85-ZE CPU Units only.

Data Exchange for the CS1 Series



Special I/O Area Allocation for the CS1 Series

Unit No.	Allocated words	Unit No.	Allocated words	Unit No.	Allocated words	Unit No.	Allocated words
0	CIO 2000 to CIO 2009	4	CIO 2040 to CIO 2049	8	CIO 2080 to CIO 2089	C	CIO 2120 to CIO 2129
1	CIO 2010 to CIO 2019	5	CIO 2050 to CIO 2059	9	CIO 2090 to CIO 2099	D	CIO 2130 to CIO 2139
2	CIO 2020 to CIO 2029	6	CIO 2060 to CIO 2069	A	CIO 2100 to CIO 2109	E	CIO 2140 to CIO 2149
3	CIO 2030 to CIO 2039	7	CIO 2070 to CIO 2079	B	CIO 2110 to CIO 2119	F	CIO 2150 to CIO 2159

3-1-2 Overview of IR/CIO Area Allocations

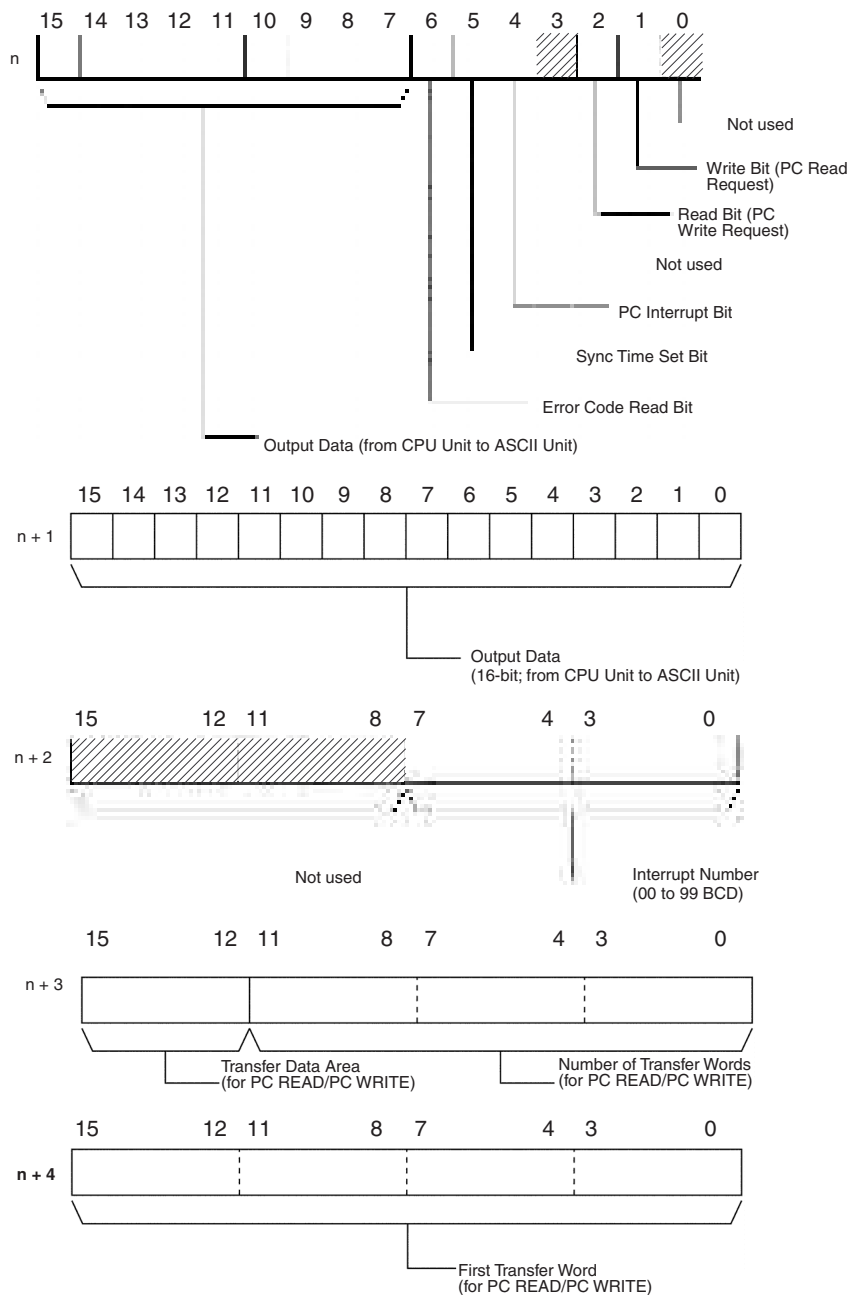
The following data is exchanged automatically during the I/O refresh period. The first word allocated to the ASCII Unit in the IR/CIO Area is “n.” The value of “n” can be calculated from the unit number using the following equation.

- C200HX/HG/HE, C200H, C200HS
  - Unit numbers 0 to 9:  $n = 100 + 10 \times \text{Unit number}$
  - Unit numbers A to F:  $n = 400 + 10 \times (\text{Unit number} - 10)$
- CS1 Series
  - Unit numbers 0 to F:  $n = 2000 + 10 \times \text{Unit number}$



Outputs

The following words are output from the CPU Unit to the ASCII Unit.

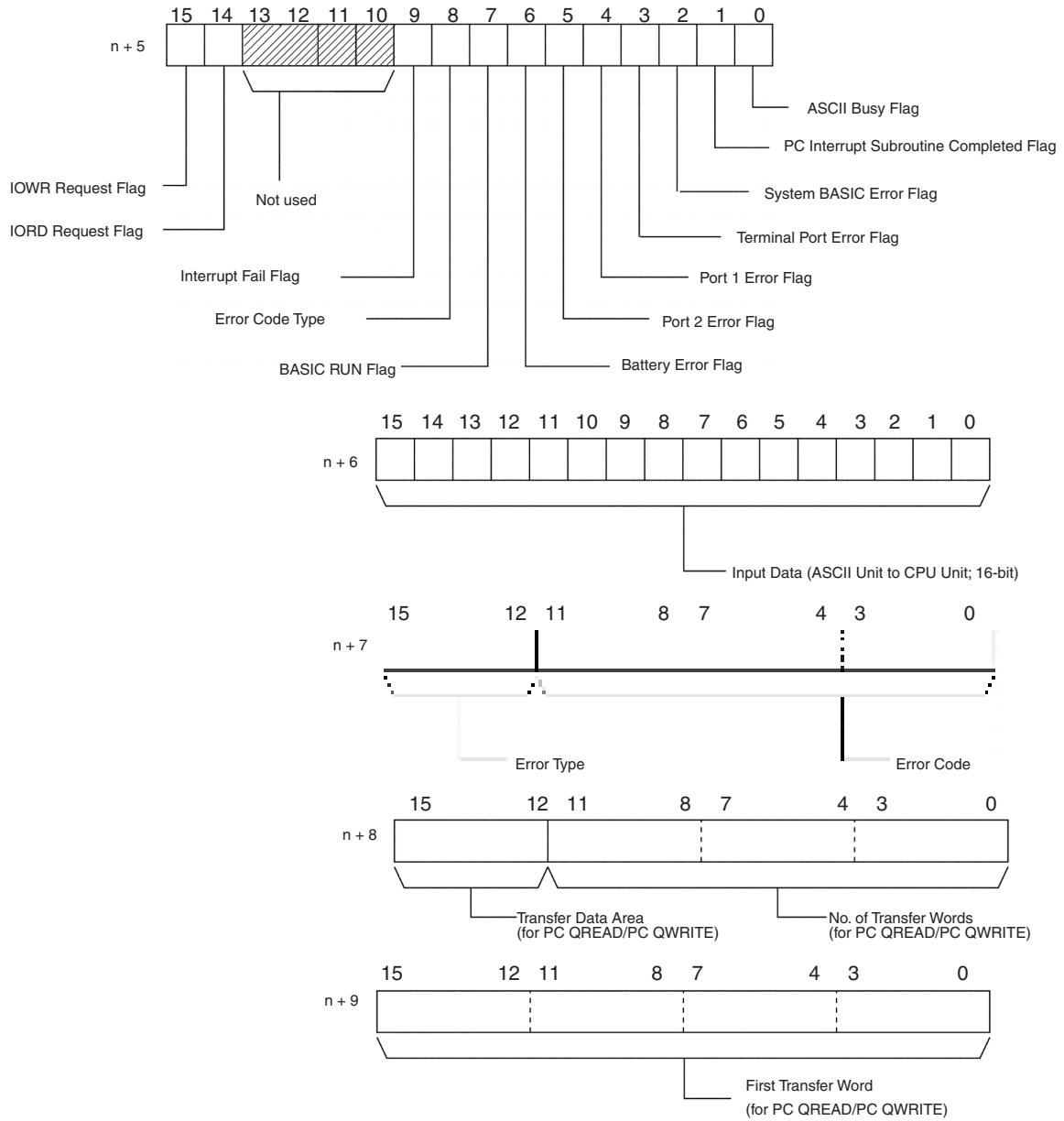


Word	Bit	Name	Function	Use
n	00	Not used	---	---
	01	Write Bit	1: Establishes the execution conditions for PC READ (Data transferred from CPU Unit to ASCII Unit according to the specified First Transfer Word and Number of Transfer Words.) 0: PC READ execution wait.	When this bit is turned ON by the user program in the CPU Unit, data is transferred according to the PC READ command at the ASCII Unit.
	02	Read Bit	1: Establishes the execution conditions for PC WRITE (Data transferred from ASCII Unit to CPU Unit according to the specified First Transfer Word and Number of Transfer Words.) 0: PC WRITE execution wait.	When this bit is turned ON by the user program in the CPU Unit, data is transferred according to the PC WRITE command at the ASCII Unit.
	03	Not used	---	---
	04	PC Interrupt Bit	0 to 1: Sends an interrupt to the ASCII Unit. The interrupt subroutine for the ON PC GOSUB is branched to when this bit changes from 0 to 1.	When this bit is switched from OFF to ON by the user program in the CPU Unit, an interrupt is sent to the ASCII Unit. It is necessary to set the interrupt number in n + 2 bits 0 to 7 when using the PC Interrupt Bit.
	05	Sync Time Set Bit	0 to 1: Sets the ASCII Unit RTC according to the CPU Unit RTC.	The time on the clock in the ASCII Unit is set to the time on the clock in the CPU Unit.
	06	Error Code Read Bit	0 to 1, 1 to 0: Reads the error code to n + 7.	By toggling this bit (0 to 1, 1 to 0) the last error code from the ASCII Unit's internal error log can be read and set in n + 7. The error status in the ASCII Unit will not be cleared when the error code is read using this bit.
	07 to 15	Output Data	Data transferred from the CPU Unit to the ASCII Unit. The ASCII Unit reads this data via the BASIC program's PC GET command.	The CPU Unit writes data here using the MOV instruction in the user program. The ASCII Unit can read this data via the PC GET command.
n + 1	00 to 15	Output Data	Data transferred from the CPU Unit to the ASCII Unit. The ASCII Unit reads this data via the BASIC program's PC GET command.	The CPU Unit writes data here using the MOV instruction in the user program. The ASCII Unit can read this data via the PC GET command.
n + 2	00 to 07	Interrupt Number	00 to 99 BCD (2-digit)	When the ON PC GOSUB command interrupt number and this interrupt number are the same and the PC Interrupt Bit changes from OFF to ON, an interrupt will be sent from the CPU Unit to the ASCII Unit. The interrupt subroutine will be executed in response.
	08 to 15	Not used	---	---

Word	Bit	Name	Function	Use		
n + 3	00 to 11	No. of Transfer Words	Number of words transferred when ASCII Unit PC READ/PC WRITE command is executed (1 to 255).	Data is written via the MOV instruction from the user program in the CPU Unit. If the Write Bit is ON, the data will be transmitted by the ASCII Unit's PC READ/PC WRITE command at the next I/O refresh period.		
	12 to 15	Transfer Data Area	The memory area for ASCII Unit PC READ/PC WRITE commands is set using the following codes.			
			No.		Data area	
					C200HX/HG/HE, C200H, C200HS	CS1 Series
			0		DM Area	
			1		IR Area	CIO Area
			2		LR Area	Data Link Area
			3		HR Area	H000 to H099 in Holding Area
			4		AR Area	H100 to H127 in Holding Area
			5		EM Area bank 0	
6	TC Area	Timer Area				
n + 4	00 to 15	First Transfer Word (address)	First word transferred from the CPU Unit memory address when the ASCII Unit's PC READ/PC WRITE command is executed.			

Input

The following words are input from the ASCII Unit to the CPU Unit.



Word	Bit	Name	Function	Use
n + 5	00	ASCII Busy Flag	1: The ASCII Unit is exchanging data with the CPU Unit. There is an error in the DM settings. The ROMSAVE command is being executed and data is being transferred to the flash ROM. 0: Transfer is not occurring.	When this flag is ON, the next transfer with the ASCII Unit should not be executed in the user program in the CPU Unit. Use this flag's address in an NC input condition for an OUT instruction controlling the Read Flag and Write Flag outputs to the ASCII Unit or as an NC input condition for the IOWR (#00□□) instruction.
	01	PC Interrupt Subroutine Completed Flag	Turns ON when the interrupt subroutine at the PC is completed.	Until this bit toggles, subroutine results are not yet available. (See Note.)
	02	System BASIC Error Flag	1: System error or BASIC error 0: Normal	---
	03	Terminal Port Error Flag	1: Communications error at the terminal port 0: Normal	These bits are cleared by executing an ERC command.
	04	Port 1 Error Flag	1: Port 1 transmission error (parity error, reception buffer overflow, etc.) 0: Normal	
	05	Port 2 Error Flag	1: Port 2 transmission error (parity error, reception buffer overflow, etc.) 0: Normal	
	06	Battery Error Flag	1: Battery voltage drop or no battery connected 0: Normal	---
	07	BASIC RUN Flag	1: BASIC program running 0: BASIC program stopped	---
	08	Error Code Type	1: New 0: Old	This flag shows whether the error for the error code being read was generated before the last RUN.
	09	PC Interrupt Fail Flag	1: PC Interrupt has failed 0: Normal	This flag turns ON when a PC interrupt fails due to the PC OFF (stop interrupt) condition in the ASCII Unit. The PC Interrupt Bit will turn ON for the next cycle and will turn OFF for the following cycle.
	10 to 13	Not used	---	---
	14	IORD Request Flag	1: IORD (#FD00) execution request by the ASCII Unit to the CPU Unit (When a PC QWRITE command is executed, this flag turns ON automatically, and after the data is transferred by the CPU Unit using the IORD (#FD00) instruction, it automatically turns OFF at the next I/O refresh period.) 0: No request	This flag automatically turns ON when a PC QWRITE command is executed to request execution of an IORD (#FD00) instruction from the CPU Unit.
	15	IOWR Request Flag	1: IOWR (#FD00) execution request by the ASCII Unit to the CPU Unit (When a PC QREAD command is executed, this flag turns ON automatically, and after the data is transferred by the CPU Unit using the IOWR (#FD00) instruction, it automatically turns OFF at the next I/O refresh period.) 0: No request	This flag automatically turns ON when a PC QREAD command is executed to request execution of an IOWR (#FD00) instruction from the CPU Unit.

Word	Bit	Name	Function	Use		
n + 6	00 to 15	Input Data	The data (16 bit) transferred from the ASCII Unit to the CPU Unit by a PC PUT command at the ASCII Unit.	The ASCII Unit's PC PUT command writes data to the IR/CIO words allocated to the ASCII Unit. After the next I/O refresh, this 16-bit, 8-bit or 1-bit data is read using the CPU Unit's MOV or LD instruction.		
n + 7	00 to 11	Error Code	Error code	Each time word n bit 06 changes status (ON to OFF or OFF to ON) in the CPU Unit, the next newest error code and error type are read from the ASCII Unit's error code table, and stored in the CPU Unit.		
	12 to 15	Error Type	B: BASIC error 0: System error 1: Port 1 communications error 2: Port 2 communications error 3: Terminal port communications error (C200H-ASC31 only)			
n + 8	00 to 11	No. of Transfer Words	The number of words transferred when the ASCII Unit's PC QREAD/PC QWRITE command is executed. (1 to 128)	The ASCII Unit writes data for these words when the PC QREAD/PC QWRITE commands are executed, and then sets the IORD/IOWR Request Flag in word n + 5 bit 14/15. When the user program in the CPU Unit detects the ON status of the flag using an LD instruction, it can access the words using the IORD/IOWR (#FD00) instructions. (See note 2.)		
	12 to 15	Transfer Data Area	The memory area for the ASCII Unit PC QREAD/PC QWRITE commands.			
			No.		Data area	
					C200HX/HG/HE, C200H, C200HS	CS1 Series
			0		DM Area	
			1		IR Area	CIO Area
			2		LR Area	Data Link Area
			3		HR Area	H000 to H099 in Holding Area
			4		AR Area	H100 to H127 in Holding Area
5			EM Area (bank number 0 only)			
6	TC Area	Timer Area				
n + 9	00 to 15	First Transfer Word (address)	The first word to be transferred (CPU Unit memory address) for the ASCII Unit's PC QREAD/PC QWRITE commands.			

- Note**
1. An OR can be taken of DIFU/DIFD instructions to detect changes in the status of the Interrupt Subroutine Completed Flag. This flag can be used when executing an IORD instruction (#00□□) when a PC EPUT command is received during a subroutine.
  2. The data set in these words by execution of the PC QREAD/PC QWRITE commands is used for monitoring from the CPU Unit. Actual data transfer is performed according to the operands specified for the IOWR/IORD (#FD00) instructions. Care is required for CS1-series PCs because the addressing range for PC QREAD and PC QWRITE is different from that for IOWR/IORD (#FD00). Dummy addresses can be used when the desired addresses cannot be addressed directly with PC QREAD/PC QWRITE. Refer to 6-3 Details of the Data Exchange Methods for details on page 75.

### 3-2 DM Area Allocations

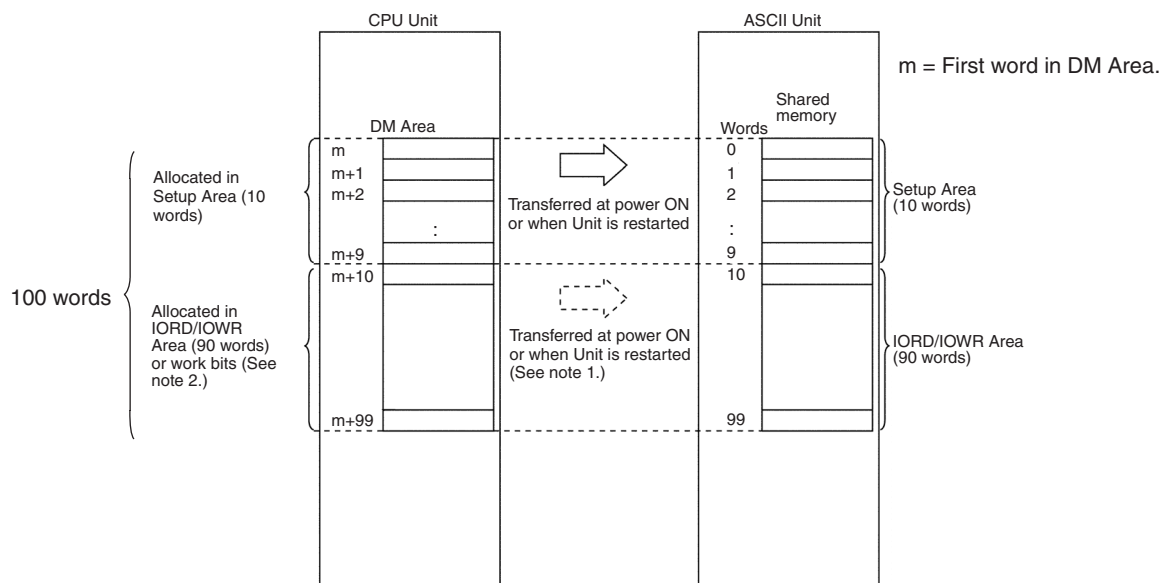
#### 3-2-1 Overview

Each ASCII Unit is also allocated 100 words as shared memory from the CPU Unit's Data Memory Area. The unit number determines which 100 words are allocated. The shared memory (100 words) in the ASCII Unit is divided into two regions as follows:

- Setup Area (10 words): Words 0 to 9
- IORD/IOWR Area (90 words): Words 10 to 99

The ASCII Unit's internal shared memory does not have a battery backup. When the power supply is turned off, it will return to default values. Data from the CPU Unit will be resent every time the power is turned on or the Unit is restarted.

The first word allocated to the ASCII Unit in the DM area and the first word in the shared memory in the ASCII Unit is "m."



- Note**
1. The IORD/IOWR Area can be set to be transferred or not transferred in bits 08 to 11 of word m+1 in shared memory in the Setup Area.
  2. The IORD/IOWR Area (m+10 to m+99) can be used as work bits when not used for the transfer to the ASCII Unit as described in note 1.

#### DM Area Allocation C200HX/HG/HE, C200H, C200HS

Unit No.	Words	Unit No.	Words	Unit No.	Words
0	DM 1000 to DM 1099	6	DM 1600 to DM 1699	A	DM 2000 to DM 2099
1	DM 1100 to DM 1199	7	DM 1700 to DM 1799	B	DM 2100 to DM 2199
2	DM 1200 to DM 1299	8	DM 1800 to DM 1899	C	DM 2200 to DM 2299
3	DM 1300 to DM 1399	9	DM 1900 to DM 1999	D	DM 2300 to DM 2399
4	DM 1400 to DM 1499	---	---	E	DM 2400 to DM 2499
5	DM 1500 to DM 1599	---	---	F	DM 2500 to DM 2599

**Note** Unit numbers A to F are used for the C200HG-CPU53-E/63-E/53-ZE/63-ZE and C200HX-CPU54-E/64-E/54-ZE/64-ZE/65-ZE/85-ZE only.

CS1-series PCs

Unit No.	Words	Unit No.	Words	Unit No.	Words
0	D20000 to D20099	6	D20600 to D20699	A	D21000 to D21099
1	D20100 to D20199	7	D20700 to D20799	B	D21100 to D21199
2	D20200 to D20299	8	D20800 to D20899	C	D21200 to D21299
3	D20300 to D20399	9	D20900 to D20999	D	D21300 to D21399
4	D20400 to D20499	---	---	E	D21400 to D21499
5	D20500 to D20599	---	---	F	D21500 to D21599

**Setup Area  
(10 words)**

The first 10 words (0 to 9) of the allocated 100 words in the DM area are the ASCII Unit's Setup Area. When power is turned ON or the Unit is restarted, it will be possible to transfer the contents of these words from the CPU Unit to the ASCII Unit.

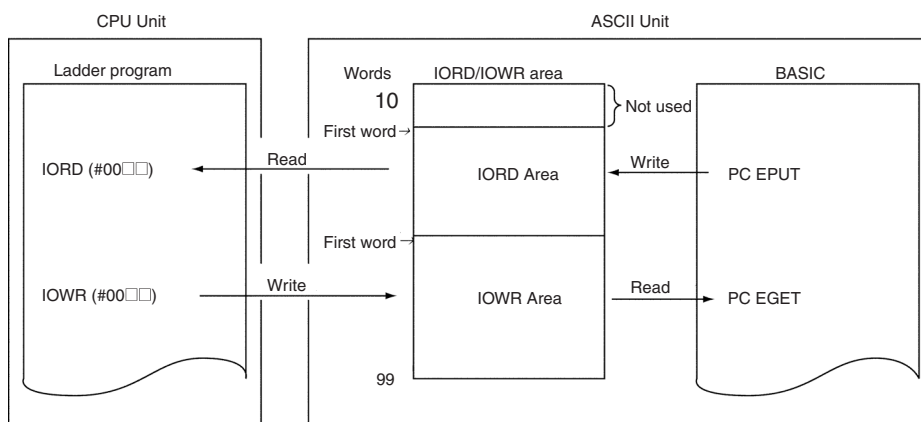
The Setup Area consists of the following parameters. For details, refer to 3-2-2 Overview of DM Allocations.

- Communications parameters for the ASCII Unit ports.
- Number of the program activated at startup.
- Division of the remaining 90 words into an IOWR Area and IORD Area.
- Transfer ON/OFF setting for the remaining 90 words from the CPU Unit to the ASCII Unit at power ON or when the Unit is restarted.

**IORD/IOWR Areas  
(90 Words)**

The 90 words (10 to 99) remaining out of the 100 words are called IORD/IOWR Areas and are used for IOWR/IORD instructions from the CPU Unit. Within the ASCII Unit, these are the areas that can be read/written using the PC EGET/EPUT commands from the ASCII Unit.

Within these 90 words, you can set which words the CPU Unit can read using the IORD instruction (i.e., the area written by the ASCII Unit via the PC EPUT command called the IORD Area) and which words the CPU Unit can write using the IOWR instruction (i.e., the area read by the ASCII Unit using the PC EGET command called the IOWR Area).

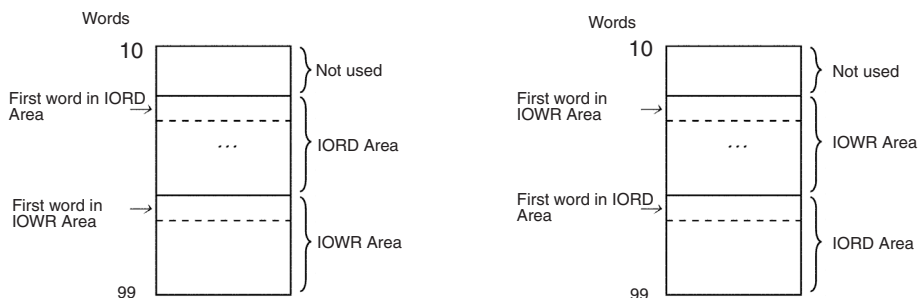


You can also set whether or not the IORD and IOWR Areas in the DM words allocated to the ASCII Unit in the CPU Unit are transferred to the ASCII Unit's IORD/IOWR Area when power is turned ON or the ASCII Unit is restarted. This parameter is set in word  $m + 1$  bits 08 to 11. If not transferred, the DM area words ( $m+10$  to  $m+99$ ) can be used as word bits for other purposes in programming.



**Note** Setting the IORD/IOWR Areas within the following ranges.

- Set the first word in the IORD Area, and the first word in the IOWR Area.
- The IORD Area and IOWR Area can be set in any order and the first part of the allocated DM words can be left unused by either area.



### 3-2-2 Overview of DM Allocations

The first address in the DM words allocated in the CPU Unit is “m.”

C200HX/HG/HE, C200H, C200HS:  $m = DM\ 1000 + 100 \times \text{Unit No.}$

CS1-series PCs:  $m = D20000 + 100 \times \text{Unit No.}$

Shared memory address	DM address	Bit	Name	Function	Settings (hex)	ASCII Unit's default (hex)
0	m	00 to 07	Program No.	Sets the startup program number.	00: No.1 01: No.1 02: No.2 03: No.3 04: No.4	00
		08 to 15	Automatic transfer from flash ROM	Determines whether the program is read from the flash ROM to the user memory area at power ON or when the Unit is restarted.	00: Do not transfer (user memory mode) 5A: Transfer flash ROM to user memory area	00
01	m + 1	00 to 07	Startup mode	Determines whether or not the program runs automatically at power ON or when the Unit is restarted. When manual start is designated, start using the START/STOP switch on the front panel of the ASCII Unit or the RUN command from the terminal. (See page 20 for details on the START/STOP switch.)	00: Manual start 5A: Automatic start	00
		08 to 11	IORD/IOWR Area transfer	Determines whether or not the CPU Unit's 90 words (DM m + 10 to m + 99) are sent to the ASCII Unit's shared memory (IORD/IOWR) area at power on or restart.	0: No transfer 1: Transfer	0
		12 to 15	Not used	---	Set to 0 (OFF)	0

Shared memory address	DM address	Bit	Name	Function	Settings (hex)	ASCII Unit's default (hex)
02	m + 2	00 to 07	Port #1 baud rate	Sets the baud rate for port #1.	00: 9,600 01: 300 02: 600 03: 1,200 04: 2,400 05: 4,800 06: 9,600 07: 19,200 08: 38,400 (bps)	00
		08 to 15	Port #1 DMA transmission (C200H-ASC31 only)	Sets DMA transmission for port #1. (See page 48 for details on DMA.)	00: No DMA sent 5A: DMA sent	00
03	m + 3	00 to 07	Port #2 baud rate	Sets the baud rate for port #2.	00: 9,600 01: 300 02: 600 03: 1,200 04: 2,400 05: 4,800 06: 9,600 07: 19,200 08: 38,400 (bps)	00
		08 to 15	Port #2 DMA transmission	Sets DMA transmission for port #2. (See page 48 for details on DMA.)	00: No DMA sent 5A: DMA sent	00
04	m + 4	00 to 07	Terminal port #3 baud rate (C200H-ASC31 only)	Sets the baud rate of terminal port #3.	00: 9,600 01: 300 02: 600 03: 1,200 04: 2,400 05: 4,800 06: 9,600 07: 19,200 (bps)	00
		08 to 15	Number of transfer words per cycle (C200HX/HG/HE or CS1 only)	Determines the maximum number of transfer words per CPU Unit cycle for PC READ/PC WRITE commands (including @ variants.)	00: 20 words max. 5A: 127 words max. 5A is available only for the C200HX/HG/HE or CS1. (Not available for remote I/O.)	00
05	m + 5	00 to 07	Not used	---	---	00
		08 to 11	Screen size: Width	Determines cursor movement in Edit Mode.	0: 80 characters 1: 40 characters 2: 132 characters	0
		12 to 15	Terminal emulation	Sets the terminal emulation mode for key operation, etc.	0: VT100 mode 1: FIT10 mode	0
06	m + 6	00 to 07	First word in IOWR Area	Sets the first word in the IOWR Area between address 10 and 99.	00: No IOWR Area 10 to 99: IOWR first word	00
		08 to 15	Not used			
07	m + 7	00 to 07	First word in IORD Area	Sets the first word in the IORD Area between address 10 and 99.	00: No IORD Area 10 to 99: IORD first word	00
		08 to 15	Not used			
08	m + 8	00 to 15	Not used			

Shared memory address	DM address	Bit	Name	Function	Settings (hex)	ASCII Unit's default (hex)
09	m + 9	00 to 15	Not used			
10 to 99	m + 10 to m + 99	00 to 15	IORD Area and IOWR Area (set first words at address 6 and 7)			00

- Note**
1. Do not specify settings other than those listed in the Settings column above.
  2. When the start mode is set to start automatically, the message "Please wait compiling..." which is normally output to the terminal port when a program is run, will not be output when the power is turned ON/OFF or when the START/STOP switch is used.
  3. When there is an error in the DM settings, the ERR indicator will be lit. Since terminals cannot be connected when this indicator is lit, use a Programming Console to confirm the error code, reset the DM settings correctly, and either turn the power OFF then ON again, or restart the ASCII Unit.

# SECTION 4

## Data Exchange with General-purpose External Devices

This section describes the ASCII Unit commands in conjunction with transmission control signals for opening and closing communications ports and for sending and receiving data between the ASCII Unit and external devices.

4-1	Overview . . . . .	44
4-2	Opening a Communications Port . . . . .	44
4-2-1	OPEN# Command . . . . .	44
4-2-2	OPEN Command Specifications . . . . .	45
4-2-3	Control/Monitor Commands for Transmission Control Signals . . . . .	46
4-2-4	Device Symbol . . . . .	46
4-3	Sending Data . . . . .	47
4-3-1	PRINT# Command . . . . .	47
4-3-2	DMA Data Transmission . . . . .	48
4-3-3	Transmission Control Signals . . . . .	48
4-4	Receiving Data . . . . .	51
4-4-1	INPUT# Command . . . . .	51
4-4-2	Transmission Control Signals . . . . .	51
4-4-3	INPUT\$ Function . . . . .	52
4-5	Closing a Communications Port . . . . .	53
4-5-1	Transmission Control Signals . . . . .	53
4-5-2	Transmission Control Signal Timing Charts . . . . .	53
4-6	Communications Errors . . . . .	59

## 4-1 Overview

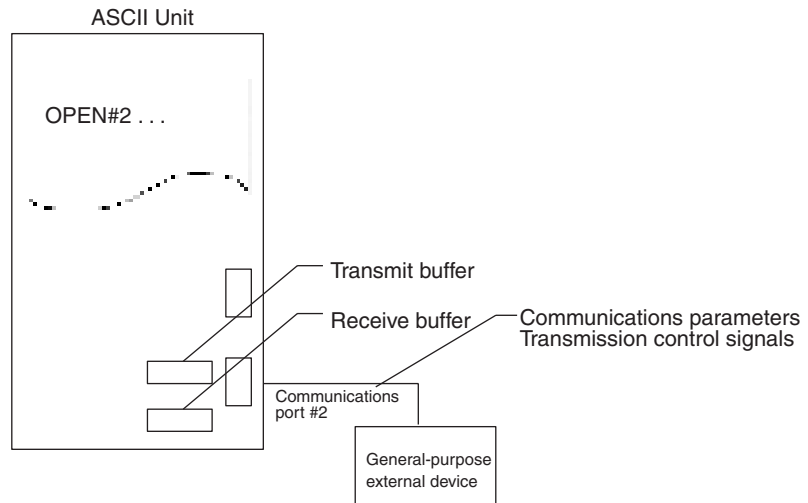
The following operations are covered in this section.

Operation		Command/function
Opening a communications port		OPEN command
Sending data		PRINT command (The LPRINT command executes device: LPRT.)
Receiving data	CR terminator	INPUT command (The LINE INPUT command receives everything in one character variable).
	Specified character length	INPUT\$ function
Closing a communications port		CLOSE command
Communications error processing		ON ERROR command, ERC command, ERR command, and RESUME command

## 4-2 Opening a Communications Port

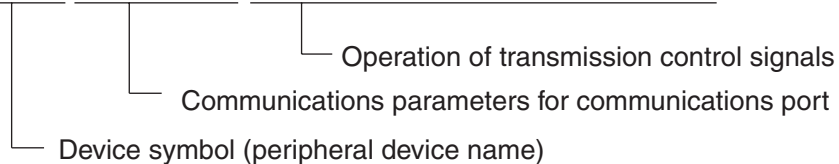
### 4-2-1 OPEN# Command

A communications port is opened using the OPEN# command. The OPEN# command specifies the device (peripheral device name), communications parameters for the communications port, and the use of transmission control signals.



#### Example

OPEN #2, "COMU: 9600, 8, N, 2, CTS\_XX, RTS\_XX, DSR\_XX, XN\_XX"



## 4-2-2 OPEN Command Specifications

The following details are specified in the OPEN command.

Item		Settings	Detail
Device symbol (peripheral device name)		SCRN, LPRT, EKPR, NKPRT, KYBD, COMU, TERM	<ul style="list-style-type: none"> <li>• Use of the transmission/receive buffer.</li> <li>• Control of the transmission control signals.</li> </ul>
Baud rate		300/600/1,200/2,400/4,800/9,600/19,200/38,400 bps (default: 9,600 bps ) (port 3 ≤ 19,200 bps)	
Data length		7 or 8 bits (default: 8 bits)	
Parity		None, odd, or even (default: none)	
Stop bits		1 or 2 bits (default: 2 bits).	
RS-232C port transmission controls	CTS	CTS_ON	Executes RTS/CTS flow control for the receive buffer of the external device. The ASCII Unit monitors the CTS signal during execution of the PRINT command. When the CTS signal is OFF, data transmission is suspended. When the CTS signal is ON, data transmission continues.
		CTS_OFF	RTS/CTS flow control is not executed. The ASCII Unit does not monitor the CTS signal during execution of the PRINT command.
	RTS	RTS_ON	Turns ON the RTS signal from the opening of the communications port until it closes. When receiving data normally from the external device into the receive buffer, RTS must be set to RTS_ON.
		RTS_OFF	Turns ON the RTS signal only during the execution of I/O commands (PRINT and INPUT), turning it OFF at all other times. BASIC's RTS control is disabled.
RS-232C port transmission controls	DSR	DSR_ON	When output commands (such as PRINT or LPRINT) are being executed, the ASCII Unit checks the DSR signal, and sends data if the DSR signal is ON. If the DSR signal is OFF, the ASCII Unit waits to transmit until the DSR signal turns ON. The BASIC indicator will flash slowly while the Unit is waiting to transmit.
		DSR_OFF	The ASCII Unit does not check the DSR signal when output commands are being executed.
RS-422A/485 port transmission controls		CTS, RTS, DSR not used (invalid)	---
Xon/Xoff (Not available for port 2 of ASC21.)		XN_ON	<p>Controls Xon/Xoff flow to both the ASCII Unit's receive buffer and the external device's receive buffer.</p> <ul style="list-style-type: none"> <li>• ASCII Unit's Receive Buffer When the receive buffer of the ASCII Unit reaches 3/4th full (384 bytes), it sends an Xoff signal, requesting the external device to suspend data transmission. Once the Unit's receive buffer drops to 1/4th full (128 bytes), it sends an Xon signal, requesting the external device to resume data transmission.</li> <li>• External Device's Receive Buffer When the ASCII Unit receives an Xoff signal from the external device during execution of a PRINT command, it suspends data transmission. When it receives an Xon signal from the external device, it resumes data transmission.</li> </ul>
		XN_OFF	Xon/Xoff flow control disabled.

### 4-2-3 Control/Monitor Commands for Transmission Control Signals

When control of the transmission control signals is not specified in the OPEN command (i.e., when they are set to OFF), they can be controlled and monitored using the following commands.

Action	Command	Example	Conditions
Turn RTS signal ON/OFF	RTS	RTS #1_SET	Only possible when RTS_ON is specified in the OPEN command.
Turn DTR signal ON/OFF	DTR	DTR #1_SET	Possible when the port is open.
Monitor the DSR signal	DSR	IF DSR(1) THEN ...	None
Monitor the CTS signal	CTS	IF CTS(1) THEN ...	None

### 4-2-4 Device Symbol

The following details are specified by the device symbol (peripheral device name).

- I/O specifications (use of the transmission/receive buffer)
- Control of transmission control signals
- Output code for the PRINT command

**Note** For details on the output codes for the PRINT command, refer to 4-3-1 PRINT# Command.

Device symbol	Peripheral device	I/O (send/receive)		Transmission control signals					
				Communications port open				I/O command execution (PRINT or INPUT)	
		Reception by ASCII Unit (using receive buffer)	Transmission by ASCII Unit (using send buffer)	RTS		DTR		RTS_ON and RTS_OFF in OPEN command	DTR
				RTS_OFF in OPEN command	RTS_ON in OPEN command	Port 1/terminal port	Port 2		
COMU	General-purpose communication device	Yes	Yes	OFF	ON	ON	OFF	ON	No change
TERM	Terminal	Yes	Yes	OFF	ON	ON	OFF	ON	No change
SCRN	Display	None	Yes	OFF	ON	OFF	OFF	ON	No change
LPRT (See note.)	Printer	None	Yes	OFF	ON	OFF	OFF	ON	No change
NKPRT (See note.)	Printer	None	Yes	OFF	ON	OFF	OFF	ON	No change
EKPRT (See note.)	Printer	None	Yes	OFF	ON	OFF	OFF	ON	No change
KYBD	Keyboard	Yes	None	OFF	ON	ON	OFF	ON	No change

**Note** The differences between LPRT, EKPRT and NKPRT are shown in the following table.

Device symbol	External device	Details
LPRT	When not outputting Chinese-derived characters (Kanji)	No conversion from Shift JIS code to JIS code.
NKPRT	When outputting Chinese-derived characters, the printer control code is "PC-PR" printer	Shift JIS code is converted to JIS code and KI (Kanji In): [ESC]K (1B4B Hex) KO (Kanji Out): [ESC]H (1B48 Hex) are added.
EKPRT	When outputting Chinese-derived characters, the printer control code is "ESC/P" printer	Shift JIS code is converted to JIS code and KI (Kanji In): [FS]&(1C26 Hex) KO (Kanji Out): [FS]. (1C2E Hex) are added.

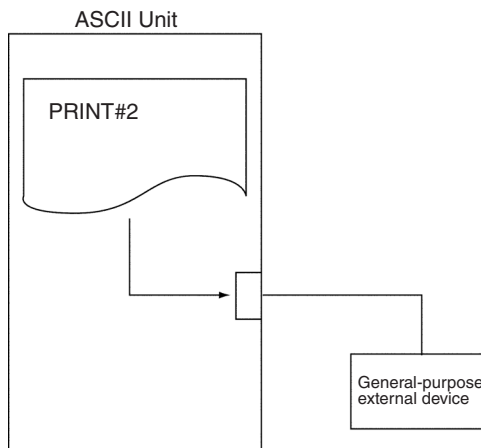
**Note** When the Super ESC/P printer control code is used, both NKPRT and EKPRT are supported.

- PC-PR: Printer control code used by NEC's PC-PR xx Series.
- ESC/P: Printer control code supported by EPSON. Used in most printers.
- Super ESC/P: Printer control code that automatically switches between ESC/P and PC-PR.

## 4-3 Sending Data

### 4-3-1 PRINT# Command

The PRINT# command sends data through the designated communication port. For example, it can send command frames originally stored in variables, print variables to the printer, and display data from variables on the screen.



**Note** The following apply when the device symbol is TERM, LPRT, or SCRN:

- If delimited with commas, output is divided into 9-character fields.
- If delimited with semi-colons or spaces, output is continuous.
- If there is no semi-colon at the end, then a new line is inserted after the transmission has ended.
- In a numerical format, one space is placed at the start and finish. (The first space is for the minus sign.)



### 4-3-2 DMA Data Transmission

It is possible to transmit data using DMA (Direct Memory Access) depending on the model of ASCII Unit and the communications port used.

Model	C200H-ASC11		C200H-ASC21		C200H-ASC31		
Port	Port #1	Port #2	Port #1	Port #2	Port #1	Port #2	Port #3
DMA transmission (See note.)	NO	YES	NO	YES	YES	YES	NO

**Note** At ports that are capable of DMA transmission, DMA transmission is enabled or disabled in the DM Setup Area according to the following table.

Bits	Name	Setting (hex)
m + 2 bits 08 to 15	Port #1 DMA transmission (C200H-ASC31 only)	00: DMA disabled (default) 5A: DMA enabled
m + 3 bits 08 to 15	Port #2 DMA transmission (all models)	00: DMA disabled (default) 5A: DMA enabled

For communications ports that do not transmit DMA data, the data is sent byte by byte from memory to the communications port by the PRINT command. This means that until the transmission is completed, the BASIC program is stopped. For ports that do transmit DMA data, the PRINT command simply provides the trigger, and the DMA controller processes the data transmission. This allows the BASIC program to proceed to execute the next line, speeding up processing. Therefore, use DMA transmission to speed up processing.

If DMA data transmission is enabled and the OPEN command is executed, the RTS/CTS flow control and Xon/Xoff flow control settings will be enabled. If CTS or Xon/Xoff flow control is enabled, DMA will be disabled. When flow control settings are enabled and DMA data transmission is used, the transmission must be completed using PRINT command.

The amount of DMA data that can be sent at one time is 255 bytes max.

To transmit data that is more than 255 bytes, write a program as follows:

```
Example      A$ = .....: 200 bytes
              B$ = .....: 100 bytes
              PRINT A$;B$
```

This example will transmits 300 bytes of data.

**Note** DMA data transmission is a method of sending data directly to a communications port without using the MPU. Because this does not involve the MPU, the ASCII Unit is able to continue executing the BASIC program as data is being transmitted. Transmission of data is carried out by a dedicated circuit (the DMA controller).

### 4-3-3 Transmission Control Signals

If the RTS signal is OFF when the PRINT# command is executed, RTS is turned ON, and the processing will proceed as follows:

**For OPEN Command with CTS\_ON, DSR\_ON**

If CTS and DSR are both ON, data will be transmitted. If DSR is OFF when transmission starts, the BASIC indicator will flash slowly and the Unit will wait to transmit data. If CTS turns OFF during data transmission, data transmission will stop and wait, and the BASIC indicator will flash slowly.

**For OPEN Command with  
CTS\_ON, DSR\_OFF**

If CTS is ON, data will be transmitted. If CTS is OFF, data will not be transmitted and the Unit will wait for CTS to turn ON; the BASIC indicator will flash slowly. If CTS turns OFF during data transmission, data transmission will stop and wait.

**For OPEN Command with  
CTS\_OFF, DSR\_ON**

If DSR is ON at the start of transmission, data will be transmitted. If DSR is OFF, data will not be transmitted, the BASIC indicator will flash slowly, and the Unit will wait for DSR to turn ON.

**For OPEN Command with  
CTS\_OFF, DSR\_OFF**

Data is transmitted irrespective of CTS and DSR.

**Output Status by Device Symbol in the OPEN Command**

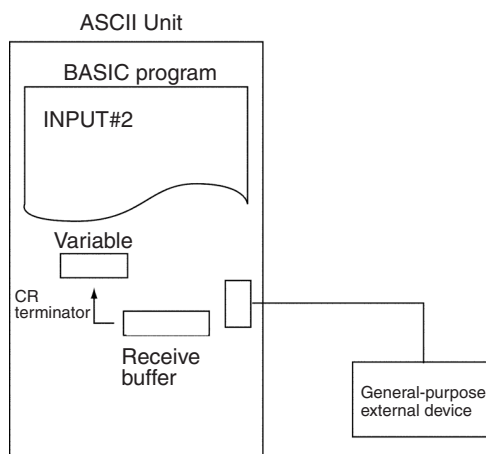
Hex	Code	Meaning	Communications port 1			Communications port 2 (port 1 for ASC31)			
			TERM/SCRN	LPRT/ NKPRT/ EKPRT	COMU	TERM	SCRN	LPRT/ NKPRT/ EKPRT	COMU
00	NUL	Space	Output	Output	Output	Output	Not supported	Output	Output
01	SOH	Start of heading							
02	STX	Start text							
03	ETX	End text							
04	EOT	End of transmission							
05	ENQ	Enquiry							
06	ACK	Acknowledgment							
07	BEL	Bell							
08	BS	Backspace							
09	HT	Horizontal tab							
0A	LF	Return (line field)							
0B	VT	Vertical tab							
0C	FF	Form forward							
0D	CR	Carriage return							
0E	SO	Shift out							
0F	SI	Shift in							
10	DLE	Device line expansion							
11	XON	Device control 1 / Xon							
12	DC2	Device control 2							
13	XOFF	Device control 3 / Xoff							
14	DC4	Device control 4							
15	NAK	Negative acknowledgment							
16	SYN	Sync signal							
17	ETB	End transmission block							
18	CAN	Cancel							
19	EM	End medium							
1A	SUB	Substitute character							
1B	ESC	Expansion							
1C	IS4	File isolation character							
1D	IS3	Group isolation character							
1E	IS2	Record isolation character							
1F	IS1	Unit isolation character							

## 4-4 Receiving Data

Data is received using the INPUT# command or the INPUT\$ (<numeral expression>[,#<port expression>]) function.

### 4-4-1 INPUT# Command

Use the INPUT# command to store data received from the receive buffer in variables using the CR code as a terminator. This allows reception of CR-delimited communications responses and input data from the keyboard. The BASIC indicator on the front panel will flash slowly until the CR code is entered. If data is not received within a certain time, INPUT# can be quit by using this command after the WAIT command. See 5-4 Time Processing.



#### Example

INPUT #2, A\$

Data from communications port #2's receive buffer up until the CR code terminator is stored in character variable A\$.

#### Example

INPUT #2, A\$, B\$

Data from communications port #2's receive buffer delimited by commas is stored in character variables A\$ and B\$.

**Note** CR or LF is not received as data for the INPUT # command.

### 4-4-2 Transmission Control Signals

If the RTS signal is ON, received data will be stored in the receive buffer independent of the INPUT command. When the INPUT command is executed, the data in the receive buffer will be stored in each variable, using the CR code as the terminator.

RTS will turn ON if the INPUT command is executed when the RTS signal is OFF. If data is received, it will be stored in the receive buffer, and then stored in each variable using the CR code as the terminator. (CTS and DSR are not checked.)

If the receive buffer overflows, the allocated I/O area ports (1, 2, and terminal port) will turn ON error flags and indicators.

- Note**
1. Use Xon/Xoff flow control to ensure that the ASCII Unit receive buffer does not overflow.
  2. Choose one of the following flow controls according to the external devices to ensure that the receive buffers of the external devices do not overflow.

- Use CTS\_ON with the OPEN command and monitor the CTS signal for RTS/CTS flow control.
  - Use XN\_ON with the OPEN command and perform Xon/Xoff flow control.
3. When an ON COM interrupt is used, the baud rate of all ports will be as follows:  
 Port 1 + port 2 ≤ 38.4 Kbps  
 If the baud rate is set to exceed this condition, a PORT CONFIGURATION ERROR (error code 0055) will be generated when compiling.

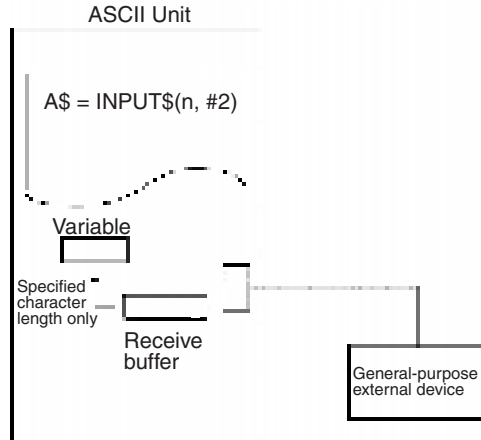
### 4-4-3 INPUT\$ Function

The INPUT\$(character\_length, port) function is used to store a fixed number of characters from the receive buffer in one string variable.

This function enables read communications data with terminators other than CR, specified numbers of bar code characters, a specified number of bytes from the receive buffer, etc. The INPUT\$(character\_length, port) function can be used in conjunction with the ON COM command (communications interrupt) or ON KEY command (key interrupt), so that when a communications interrupt or key interrupt occurs, it is possible to store a specified number of characters from the receive buffer to a variable.

Until the specified number of characters is received, the BASIC indicator on the front panel of the Unit will flash slowly, signifying that it is awaiting input.

If the specified number of characters is not received within a certain time, INPUT\$ can be quit by using this function after the WAIT command. See 5-4 Time Processing.



**Example**

A\$ = INPUT\$(10, #2)  
 Ten characters of data is stored from communications port #2 to A\$.

**Example**

**Reception of Communications Response Frame**

When a terminator other than CR is used, the communication data is received using the INPUT\$ function.

A\$ = INPUT\$(LOC(2), #2)  
 All data in the communications receive buffer is stored in character variable A\$.

- Note** 1. If 256 bytes or more of data is stored in the receive buffer, or the receive buffer is empty, and the data is retrieved using `A$ = INPUT$(LOC(2), #2)`, an `ILLEGAL FUNCTION CALL ERROR` will be generated. Therefore, use the following program to retrieve the data.

```
I% = 0
A% = LOC(2)
WHILE (A% <> 0)
    IF (A%>255) THEN A% = 255
    A$(I%) = INPUT$ (A%, #2)
    I% = I%+1
    A% = LOC(2)
WEND
```

## 4-5 Closing a Communications Port

Communications ports are closed using the `CLOSE` command.

When a transmission device has been specified using the `OPEN` command and the `CLOSE` command is executed, the data remaining in the transmit buffer is sent before the port is closed.

When a reception device has been specified using the `OPEN` command and the `CLOSE` command is executed, the data remaining in the receive buffer is cleared.

All communications ports are also closed by the `END`, `CLEAR`, and `NEW` commands. They are not closed by the `STOP` command.

### 4-5-1 Transmission Control Signals

Model	C200H-ASC11		C200H-ASC21		C200H-ASC31		
Port	Port #1	Port #2	Port #1	Port #2	Port #1	Port #2	Port #3
RTS	OFF		---		OFF		
DTR	ON	No change	ON	---	No change		ON

### 4-5-2 Transmission Control Signal Timing Charts

After turning ON the power or restarting the Unit, and until each port is opened using the `OPEN#` command, the device symbol (peripheral device name) for each port is as follows:

Model	Port	Device symbol (peripheral device name)
C200H-ASC11	Port #1	TERM
C200H-ASC21	Port #2	LPRT
C200H-ASC31	Port #1	LPRT
	Port #2	LPRT
	Port #3	TERM

#### Transmission Control Signal Status

The statuses of the transmission control signals (RTS, DTR, DSR, and CTS) at each point of time are changed as shown in the following table according to the device symbol (peripheral device name) and transmission control signal operation specified in the `OPEN#` command.

## RTS Signal

Device symbol (peripheral device name)	Transmission control signal operation setting	Executing RUN	Ports opened by OPEN# command	Receiving data with INPUT# command or INPUT\$ function	Sending data with PRINT# command	Ports closed
Port #1: TERM, COMU, or KYBD (Port #3 for ASC31)	RTS_ON	ON to OFF	OFF to ON	ON	ON	ON to OFF
	RTS_OFF	ON to OFF	OFF	Switches from OFF to ON, then back OFF again after receiving data.	Switches from OFF to ON. (For CTS_ON and DSR_ON, checks CTS and DSR, and if they are both ON, turns RTS OFF after sending data)	OFF
Port #1: KYBD (Port #3 for ASC31)	RTS_ON	ON to OFF	OFF to ON	ON	---	ON to OFF
	RTS_OFF	ON to OFF	OFF	Switches from OFF to ON, then back OFF again after receiving data.	---	OFF
Port #2: COMU (Ports #1 and #2 for ASC31)	RTS_ON	OFF	OFF to ON	ON	ON	ON to OFF
	RTS_OFF	OFF	OFF	Switches from OFF to ON, then back OFF again after receiving data.	Switches from OFF to ON. (For CTS_ON and DSR_ON, checks CTS and DSR, and if they are both ON, turns RTS OFF after sending data)	OFF
Port #2: KYBD (Ports #1 and #2 for ASC31)	RTS_ON	ON to OFF	OFF to ON	ON	---	ON to OFF
	RTS_OFF	ON to OFF	OFF	Switches from OFF to ON, then back OFF again after receiving data.	---	OFF
Port #1: SCRN and LPRT (Port #3 for ASC31)	RTS_ON	ON to OFF	OFF to ON	---	ON	ON to OFF
	RTS_OFF	ON to OFF	OFF	---	Switches from OFF to ON. (For CTS_ON and DSR_ON, checks CTS and DSR, and if they are both ON, turns RTS OFF after sending data)	OFF
Port #2: SCRN and LPRT (Ports #1 and #2 for ASC31)	RTS_ON	OFF	OFF to ON	---	ON	ON to OFF
	RTS_OFF	OFF	OFF	---	Switches from OFF to ON. (For CTS_ON and DSR_ON, checks CTS and DSR, and if they are both ON, turns RTS OFF after sending data)	OFF

**DTR Signal**

Device symbol (peripheral device name)	Executing RUN	Ports opened by OPEN# command	Receiving data with INPUT# command or INPUT\$ function	Sending data with PRINT# command	Ports closed
Port #1: TERM and COMU (Port #3 for ASC31)	ON	ON	ON	ON	ON
Port #2: COMU (Ports #1 and #2 for ASC31)	OFF	OFF to ON	ON	ON	ON to OFF
Port #1: KYBD (Port #3 for ASC31)	ON	ON	ON	---	ON
Port #2: KYBD (Ports #1 and #2 for ASC31)	OFF	OFF to ON	ON	---	ON to OFF
Port #1: SCRN and LPRT (Port #3 for ASC31)	ON	ON to OFF	---	OFF	OFF to ON
Port #2: SCRN and LPRT (Ports #1 and #2 for ASC31)	OFF	OFF	---	OFF	OFF

**CTS and DSR Signals**

CS is monitored constantly if CTS\_ON is set. DSR is only checked when printing starts.

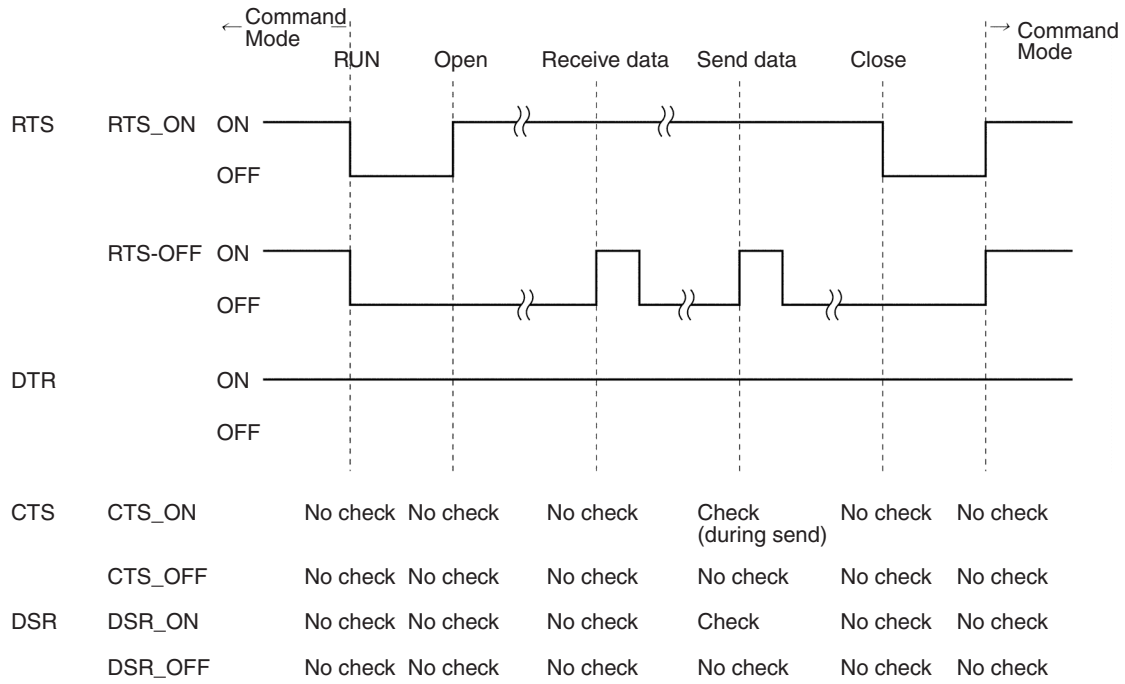
Device symbol (peripheral device name)	Transmission control signal operation setting	Executing RUN	Ports opened by OPEN# command	Receiving data with INPUT# command or INPUT\$ function	Sending data with PRINT# command	Ports closed
Port #1: TERM, COMU, SCRN, and LPRT (Port #3 for ASC31)	CTS_ON	No check	No check	No check	If CTS is OFF, data transmission is stopped. (See note 1.)	No check
Port #2: COMU, SCRN, and LPRT (Ports #1 and #2 for ASC31)	CTS_OFF	No check	No check	No check	No check	No check
	DSR_ON	No check	No check	No check	If DSR is ON, data is transmitted. (See note 2.)	No check
	DSR_OFF	No check	No check	No check	No check	No check

- Note**
1. The CTS signal is checked when data is being transmitted by the PRINT command if CTS\_ON is used. If CTS is OFF, data transmission is stopped. If CTS is ON, transmission will resume (RTS/CTS flow control performed at the receive buffer of the peripheral devices).
  2. The DSR signal is checked when the PRINT command is started if DSR\_ON is used. If DSR is OFF, the Unit will wait to send, and if DSR is ON, data will be transmitted.

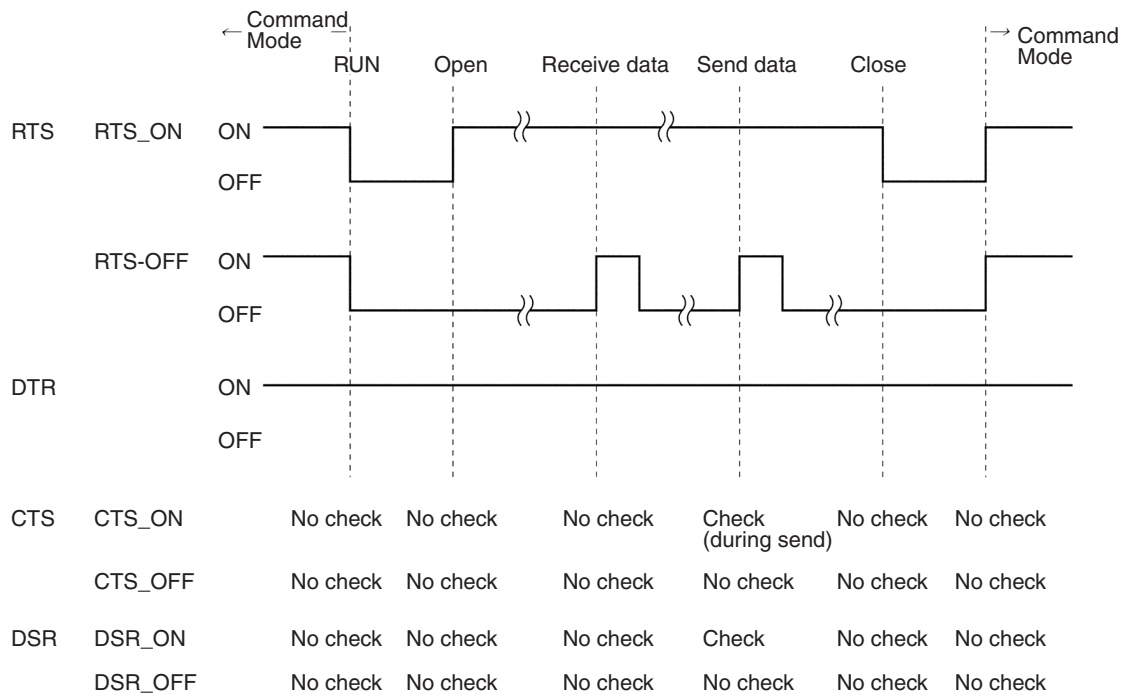


Timing Charts

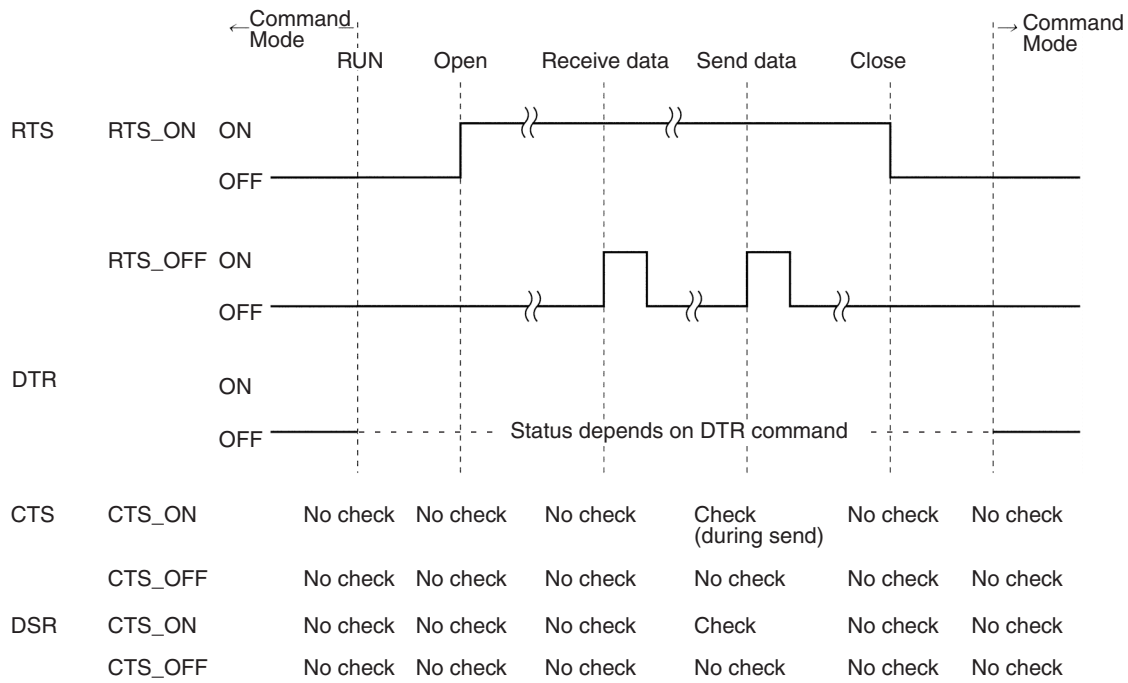
**Device Symbol: TERM**  
**Port #1/Port #3 (ASC31): TERM**



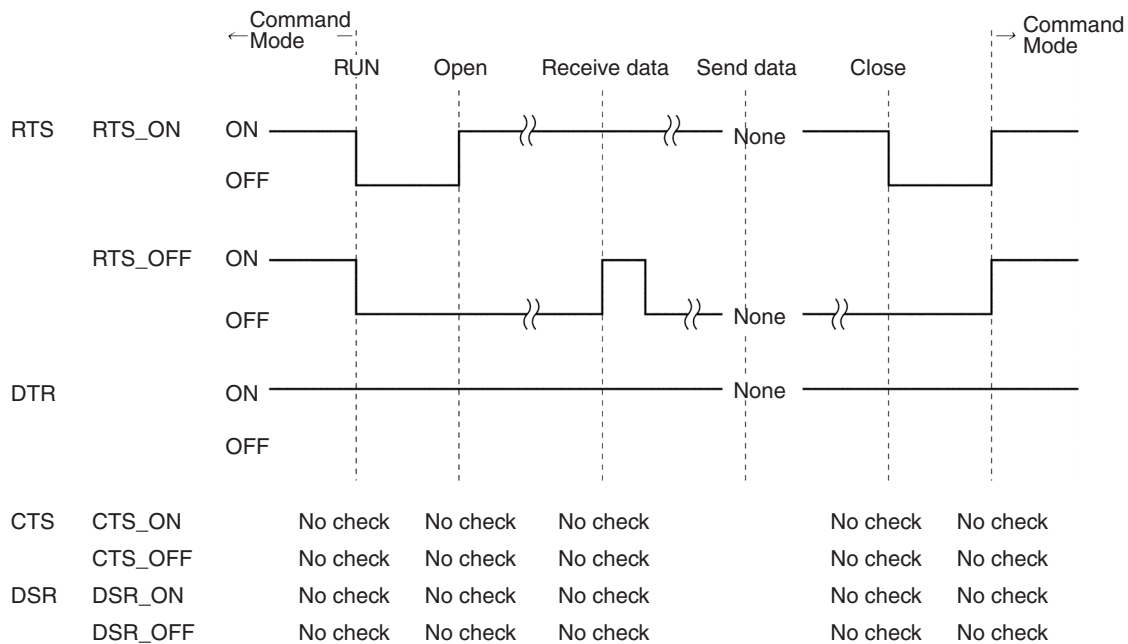
**Device Symbol: COMU**  
**Port #1: COMU (ASC11, ASC21)**



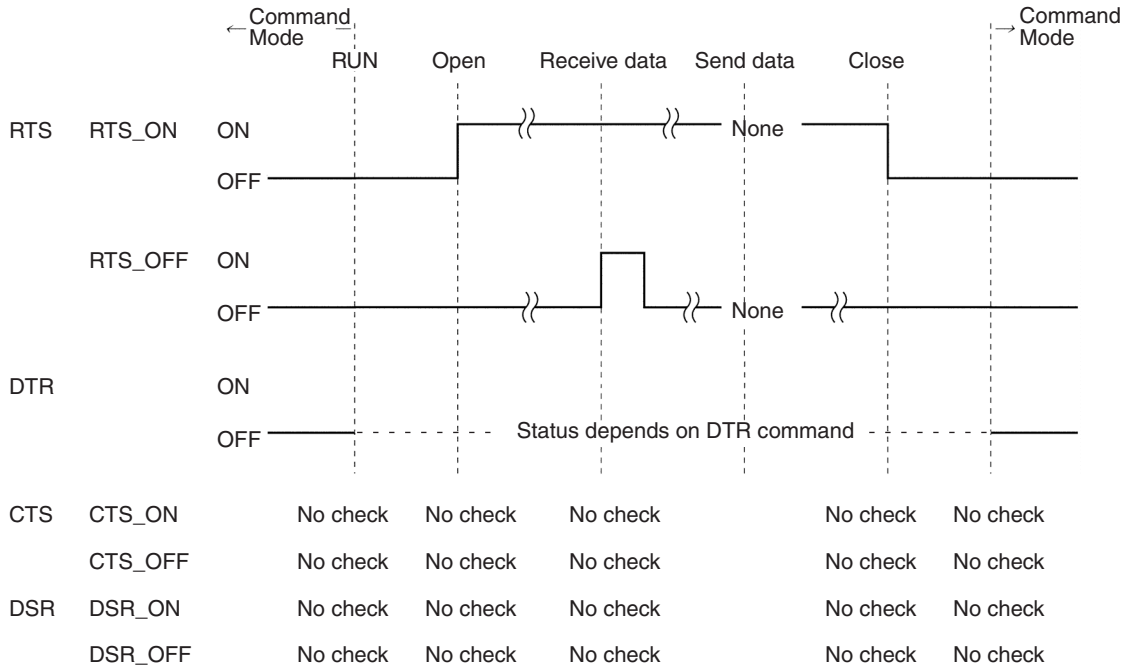
**Port #2: COMU or  
Port #1: COMU for ASC31**



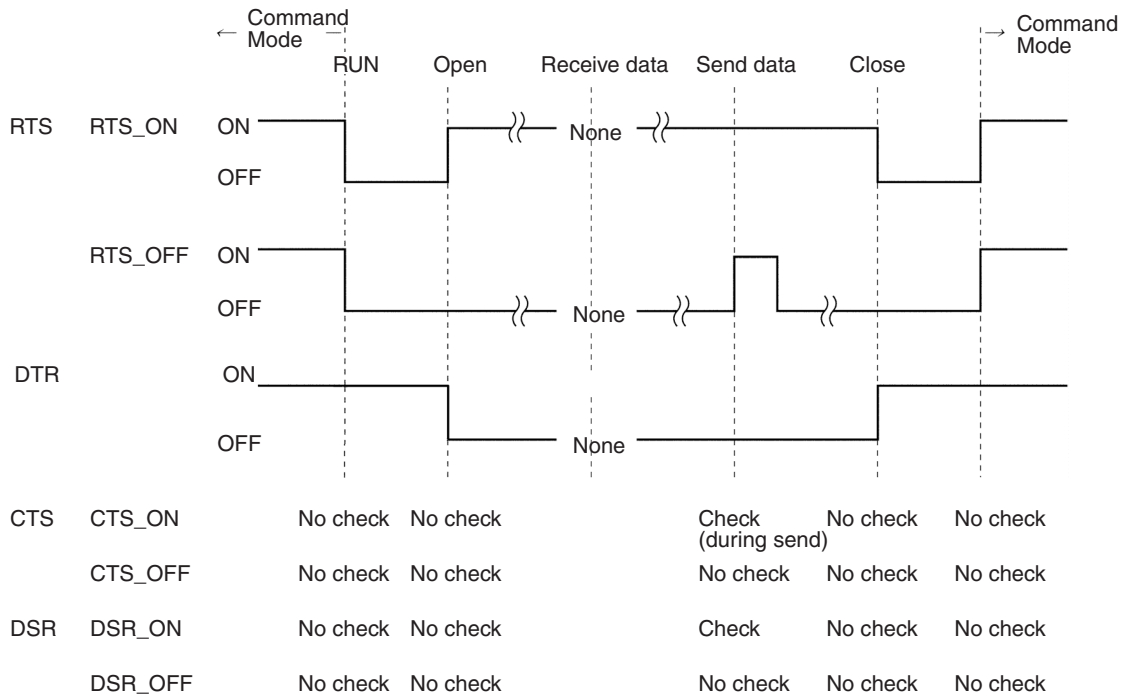
**Device Symbol: KYBD  
Port #1: KYBD (ASC11 or ASC21)**

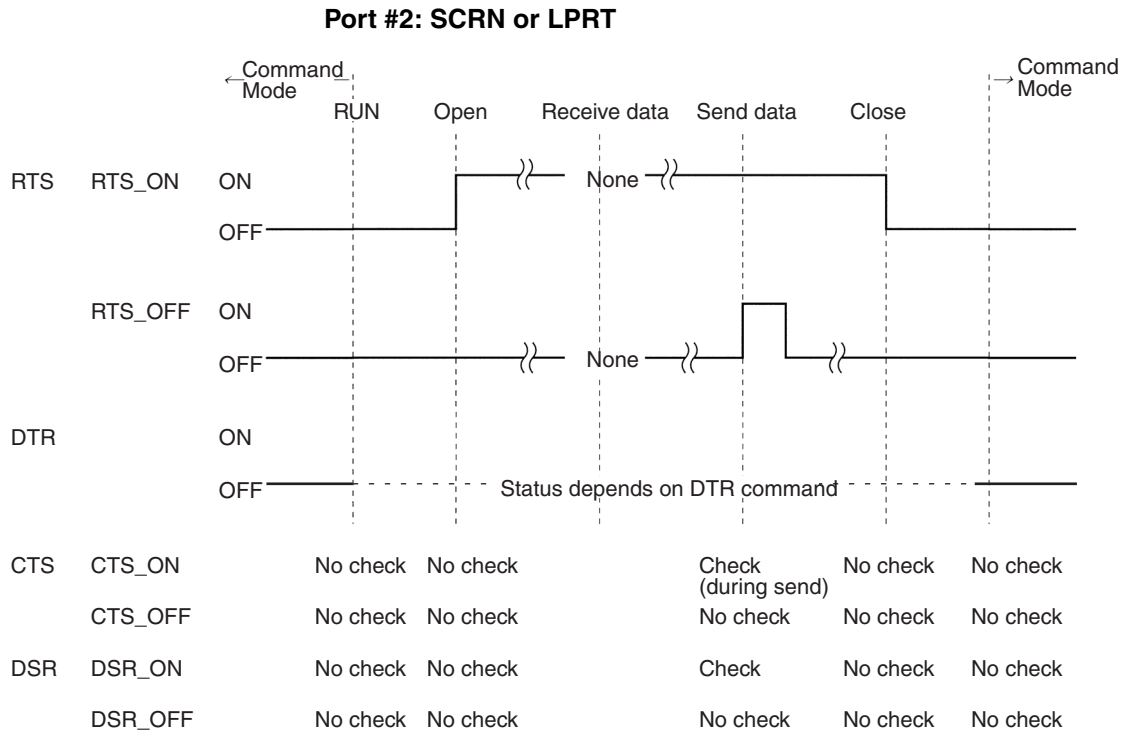


**Port #2: KYBD**



**Device Symbol: SCRN or LPRT**  
**Port #1: SCRN or LPRT**





## 4-6 Communications Errors

When a communications error occurs while a BASIC program is being executed, the ERR1 or ERR2 indicator on the ASCII Unit will be lit and at the same time the CPU Unit's port 1/2 error flag (bit 04/05 of word n+5) will be turned ON (be set to 1). Execution of the BASIC program will continue. If the ON ERROR command has been used in the BASIC program, the program will be diverted to the specified interrupt subroutine when a communications error occurs.

If another error occurs during the execution of the error interrupt subroutine however, the BASIC program will stop executing (regardless of whether the error affects the error subroutine). If the following situation is likely, program execution may be stopped, so be careful when using the ON ERROR command. Alternatively, do not use the ON ERROR command in a situation like the one below, where execution of the BASIC program should be continued even if a communications error occurs.

- Other errors occur during an interrupt subroutine due to communications errors.

### Note ON ERROR Command

The ON ERROR command is used for processing with other errors, so it will be necessary to use the IF-type statements for processing. When error processing is complete, use the RESUME command to return to the origin of the diversion and continue processing. Error status can be cleared after error processing is complete by using the ERC command.

The ERR function can be used to determine the type of error that occurred. Since communications errors can be detected from the BASIC program, retry processing is easily performed from the BASIC program.

ON ERROR GOTO \*EINT

Main program

\*EINT

Error processing

# SECTION 5

## Processing

This section provides examples of processing character strings, bits, receive buffers, looping, and interrupts.

5-1	Processing Character Strings .....	62
5-2	Processing Bits .....	62
5-3	Processing Receive Buffers.....	62
5-4	Time Processing .....	63
5-5	Interrupt Functions .....	64
5-6	Loop Processing .....	67

### 5-1 Processing Character Strings

Process	Programming
Retrieving n characters from beginning of A\$	B\$ = MID\$(A\$, 1, n) or B\$ = LEFT\$(A\$, n)
Retrieving n characters from character number m	B\$ = MID\$(A\$, m, n)
Retrieving n characters from end of A\$	B\$ = RIGHT\$(A\$, n)
Searching for character string from character number m in A\$ and getting position of first character	B = INSTR(m, A\$, character_string)
Getting number of characters in A\$	B = LEN(A\$)
Converting character string A\$ to numeric value	B = VAL(A\$)
Converting numeric value B to character string A\$	A\$ = STR\$(B)

**Example**

The following code will find the position of "STX" in response data R\$, retrieve the characters from there to "ETX" and store them in A\$.

```

100 STX$ = CHR$(&H02)      Generate STX.
110 ETX$ = CHR$(&H03)      Generate ETX.
120 A = INSTR(1, R$, STX$) Find the position of STX$ in R$.
130 B = INSTR(1, R$, ETX$) Find the position of ETX$ in R$.
140 A$ = MID$(R$, A, B-A+1) Retrieve the character string from STX$ to
                             ETX$ in R$ and store it in A$.
    
```

**Example**

If the two characters after the fourth character from the start of response data A\$ are 00, The following line will jump to line 1000 as a correct response.

```

150 IF MID$(A$,4,2)="00" THEN 1000
    
```

### 5-2 Processing Bits

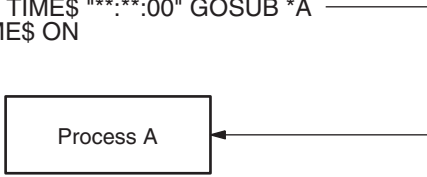
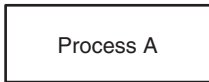

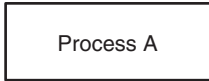
Process	Programming
Turning ON/OFF 1 bit of a variable.	BITON A%, 3 (Turns ON bit 3 of variable A%.)
	BITOFF A%, 3 (Turns OFF bit 3 of variable A%.)

- Note**
1. One bit will be set without changing the other bits.
  2. When transferring bit data to the PC, turn ON/OFF the required bit with these commands and then use the PC PUT command.

### 5-3 Processing Receive Buffers

Process	Programming
Obtaining the number of characters in the receive buffer	A = LOC(1) (Stores the number of characters held in the communications port #1 receive buffer in A.)
Executing a process if the number of characters in a receive buffer is not 0 (i.e.,if it is not empty)	<pre> 100 A = LOC(1) 110 IF A&lt;&gt;0 THEN 200 200                     </pre> <div style="border: 1px solid black; width: 100px; height: 30px; margin: 10px auto; text-align: center;">Process A</div>

## 5-4 Time Processing

Process		Program
Storing a time in T\$ (HH:MM:SS)		T\$ = TIME\$
Storing hours in H\$ (HH)		H\$ = LEFT\$(TIME\$, 2) or H\$ = MID\$(TIME\$, 1, 2)
Storing minutes and seconds in MS\$ (MM:SS)		MS\$ = MID\$(TIME\$, 4, 5)
Executing process A at fixed intervals	Using TIME\$ interrupt, interrupt subroutine is executed every minute (HH:MM:00) when the seconds equal zero.	<pre>100 ON TIME\$ "***:**:00" GOSUB *A 110 TIME\$ ON</pre> 
	Using TIMER interrupt, interrupt subroutine executes process A after every sixty seconds.	<pre>100 ON TIMER 600 GOSUB 500 110 TIMER ON</pre> <p>500</p> 
Standby processing	If response is not received within three seconds after COMMAND (C\$) transmission, process A is executed.	<pre>100 PRINT #1, C\$ 110 WAIT "3.0", 200 120 A\$ = INPUT\$(1, #1)</pre> <p>200</p> 
	If data is not completely transferred from CPU Unit within ten seconds after execution of PC READ, process A is executed.	<pre>100 WAIT "10.0", 200 110 PC READ ...</pre> <p>200</p> 

**Note** If a BASIC interrupt occurs during the wait state caused by the WAIT command, the wait monitor timer will be suspended throughout the time of the interrupt.



## 5-5 Interrupt Functions

Interrupt	Details	Program	Enable/disable/stop commands (See note 2.)
Communications interrupts	The interrupt subroutine is executed when data is received in the receive buffer. All data received is stored in the receive buffer	ON COM n GOSUB n: 1 or 2	COM n (ON/OFF/STOP) n: 1 or 2
	The interrupt subroutine is executed when a specified character is received in the receive buffer. (Data beyond the specified character is also received in the buffer.) (See note 1.)	ON COM n GOSUB, CODE = XX n: 1 or 2	
	The interrupt subroutine is executed when a specified number of bytes is received in the receive buffer. (Data beyond the specified number of bytes is also received in the buffer.) (See note 1.)	ON COM n GOSUB, BYTE = XX n: 1 or 2	
	The interrupt subroutine is executed when a specified final character is received in the receive buffer after a specified start character. (The string of characters between the start character and final character remains in the buffer. Data after the specified final character is also received in the buffer.) (See note 1.)	ON COM n GOSUB, HEAD = XX, TERM = YY n: 1 or 2	
	The interrupt subroutine is executed when a specified number of characters is received in the receive buffer after a specified start character. (The character string from the specified start character to the specified number of characters remains in the receive buffer, and the data after the specified number of characters is also received in the receive buffer.) (See note 1.)	ON COM n GOSUB, HEAD = XX, BYTE = YY n: 1 or 2	
Key interrupts	The interrupt subroutine is executed when a certain key signal is input. (See note 3.)	ON KEY(n) GOSUB n: 0 to 9	KEY n (ON/OFF/STOP) n: 0 to 9
PC interrupts	Specified PC interrupt number interrupt subroutines are executed by interrupts from the IOWR(#CC00) instruction or IR/CIO words allocated to the ASCII Unit in the PC. (See note 4.)  I/O Allocations: PC interrupts are triggered by the IR/CIO words allocated to the ASCII Unit: IR/CIO n bit 04.  PC interrupt number is set in IR/CIO n + 2 bits 00 to 07.  IOWR(#CC00): Triggers a PC interrupt by sending data.	ON PC GOSUB n: 01 to 99	PC n (ON/OFF/STOP) n: 01 to 99
Timer interrupts	The interrupt subroutine is executed repeatedly each time the timer times out (interval time). (Once executed, the subroutine is repeated until TIME OFF is specified.)	ON TIMER 300 GOSUB  A subroutine branch to the GOSUB line number will occur 30 seconds after this command is executed; the subroutine branch is then repeated each 30 seconds. Timer: 0 to 864000 (0.1-s unit)	TIMER (ON/OFF/STOP)
On alarm interrupts	The subroutine interrupt is executed when the timer times out (one-shot timer). (No repetition after the initial execution.)	ON ALARM 300 GOSUB  A subroutine branch to the GOSUB line number will occur once only, 30 seconds after this command is executed. Timer: 0 to 864000 (0.1-s unit)	ALARM (ON/OFF/STOP)
Scheduled interrupts	The interrupt subroutine is executed at a specified time.	ON TIME\$ = "□:□:□" GOSUB	TIME\$ (ON/OFF/STOP)
Error interrupts	The interrupt subroutine is executed when an error occurs.	ON ERROR GOTO	

- Note**
1. When an interrupt is generated upon receiving a specified number of characters or data up to a specified character, all data is received in the receive buffer. If the necessary number of characters is retrieved by the INPUT\$ function, it is possible to use first-in/first-out (FIFO) processing to move that group of characters to the front of the receive buffer.
  2. One of the ON, OFF, or STOP commands must be specified for the interrupt as shown in the column on relevant commands. The STOP command prevents branch processing even if the interrupt has occurred; it simply records the interrupt entry, and branching occurs with the subsequent ON command.
  3. Interrupts set for GOTO statements will wait until the current BASIC command has been executed.
  4. IOWR#CC00 interrupts are not accepted while the ASCII Busy Flag is ON.

**Interrupt Priority**

If more than one interrupt is generated at the same time, the interrupt subroutines will be executed in the following order of priority.

**Interrupt Priority Order Table**

Order of priority	Interrupt
1	ON COM 1
2	ON COM 2
3	ON KEY 0
:	:
12	ON KEY 9
13	ON PC 1
:	:
111	ON PC 99
112	ON ALARM
113	ON TIMER
114	ON TIME\$

If further interrupts are generated during the execution of an interrupt subroutine, they will be processed after execution of the current interrupt subroutine has been completed, according to the interrupt priority order table shown above. ON ERROR interrupts, however, will be executed immediately, even during execution of other interrupt subroutines.

**Example**

If ON COM 1 and ON ALARM interrupts are generated while an ON PC1 interrupt subroutine is being executed, the ON COM1 interrupt subroutine will be executed after the ON PC1 interrupt subroutine has been completed, as ON COM 1 has priority over ON ALARM. After the ONCOM 1 interrupt subroutine has finished executing, the ON ALARM interrupt subroutine will be executed.

If a PC interrupt subroutine is generated while the ON COM 1 interrupt subroutine is being executed, the PC interrupt subroutine will take priority over the ON ALARM interrupt.

If more than one interrupt is generated simultaneously or if another interrupt is generated while a interrupt subroutine is being executed, one interrupt will be recorded if this status is either ON or STOP.

**Application Example**

```

200   ON COM 1 GOSUB 1000
210   COM 1 ON
      [Communications interrupt enabled]
1000  [Interrupt subroutine
      RETURN]
    
```

The ON COM command is often used in conjunction with the INPUT\$ function as shown below. (STX% = &H02, ETX% = &H03, Receive data = 0<LOC(2)<255)

**Example 1**

```

100   ON COM 2 GOSUB 500
110   COM 2 ON
120   GOTO 120
500   A$ = INPUT$(LOC(2),#2)
510   RETURN
    
```

If data is received, all the data in the receive buffer is stored in A\$.

**Example 2**

```

100   ON COM 2 GOSUB 500, CODE = ETX%
110   COM 2 ON
120   GOTO 120
500   A$ = INPUT$(LOC(2), #2)
510   RETURN
    
```

If ETX% is received, all data in the receive buffer is stored in A\$.

**Example 3**

```

100   ON COM 2 GOSUB 500, BYTE = 20
110   COM 2 ON
120   GOTO 120
500   A$ = INPUT$(20, #2)
510   RETURN
    
```

If 20 characters are received, 20 characters from the receive buffer are stored in A\$.

**Example 4**

```

100   ON COM 2 GOSUB 500, HEAD = 02, TERM = 03
110   COM 2 ON
120   GOTO 120
500   A$ = INPUT$(LOC(2), #2)
510   RETURN
    
```

If ETX% is received from STX%, all data in the receive buffer is stored in A\$.

**Example 5**

```

100   ON COM 2 GOSUB 500, HEAD=STX%, BYTE = 20
110   COM 2 ON
120   GOTO 120
500   A$ = INPUT$(LOC (2), #2)
510   A$ = RIGHT$(A$, 20)
520   RETURN
    
```

If 20 characters are received from STX%, 20 characters from the receive buffer are stored in A\$.

Caution is required, because the the next data may already have been placed in the receive buffer.

## 5-6 Loop Processing

Process	Programming
<p>Specified number of repetitions:                      If no response is received within 3 s of transmission of the command (C\$), the command is resent. If there is no response after 3 repetitions, error processing will commence.</p>	<pre>                     10 FOR A = 1 TO 3                     20 PRINT #2, C\$                     30 WAIT "3.0", 50                     40 A\$ = INPUT\$(1, #2): GOTO 1000                     50 NEXT A                     60                     </pre> <div data-bbox="927 421 1131 497" style="border: 1px solid black; padding: 5px; margin: 10px auto; width: fit-content;">                         Error processing                     </div>
<p>Repetition under set conditions:                      A C\$ command is executed and the command/response reception are repeated as long as the 2 characters following the 4th character are 20 or less.</p>	<pre>                     10 WHILE A &lt;= 20                     20 PRINT #2, C\$                     30 B\$ = INPUT\$(LOC(2), #2)                     40 A\$ = MID\$(B\$,4,2)                     50 A = VAL(A\$)                     100 WEND                     </pre>

# SECTION 6

## Data Exchange with the CPU Unit

This section describes procedures for exchanging data between the ASCII Unit and the CPU Unit, the transfer sequence, instruction/command timing, and the use of various interrupts.

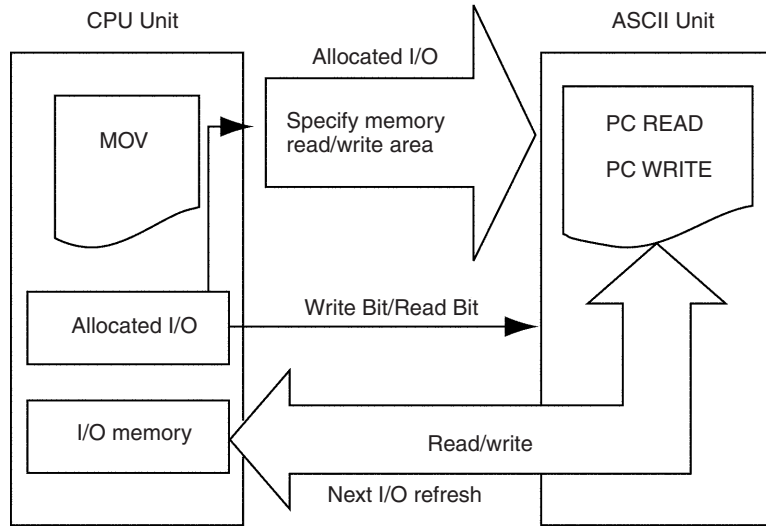
6-1	Overview of Data Exchanges .....	70
6-1-1	Data Exchange Methods.....	70
6-1-2	Sending Interrupts.....	72
6-2	Selecting the Data Exchange Method .....	74
6-3	Details of the Data Exchange Methods.....	75
6-3-1	Data Exchange Method No. 1.....	75
6-3-2	Data Exchange Method No. 2.....	78
6-3-3	Data Exchange Method No. 3.....	79
6-3-4	Data Exchange Method No. 4.....	80
6-3-5	Data Exchange Method No. 5.....	82
6-3-6	Data Exchange Method No. 6.....	85
6-3-7	Data Exchange Method No. 7.....	86
6-4	Data Exchange Time Charts .....	87
6-4-1	PC READ/PC WRITE Timing.....	87
6-4-2	PC READ@/PC WRITE@ Timing .....	87
6-4-3	PC PUT/PC GET Timing .....	88
6-4-4	IOWR (#00□□)/IORD (#00□□) Timing .....	88
6-4-5	PC EPUT/PC EGET Timing.....	88
6-4-6	IOWR (#FD00)/IORD (#FD00) and PC QREAD/PC QWRITE Timing .....	89
6-4-7	Combined IOWR (#00□□), IOWR (#CC00), ON PC GOSUB, and PC EGET Timing .....	89
6-4-8	Combined IOWR (#CC00), IORD (#00□□), ON PC GOSUB, and PC EPUT Timing .....	90
6-4-9	Sending Interrupts from CPU Unit to ASCII Unit: PC Interrupts ..	90
6-5	IOWR/IORD Instruction Specifications .....	95
6-5-1	IOWR Instruction (SPECIAL I/O UNIT WRITE).....	95
6-5-2	IORD Instruction (SPECIAL I/O UNIT READ).....	102
6-5-3	Using the IOWR and IORD Instructions .....	106
6-6	Restrictions for CS1-series PCs .....	108
6-6-1	Using the Sync Time Set Bit for CS1-series PCs.....	109

## 6-1 Overview of Data Exchanges

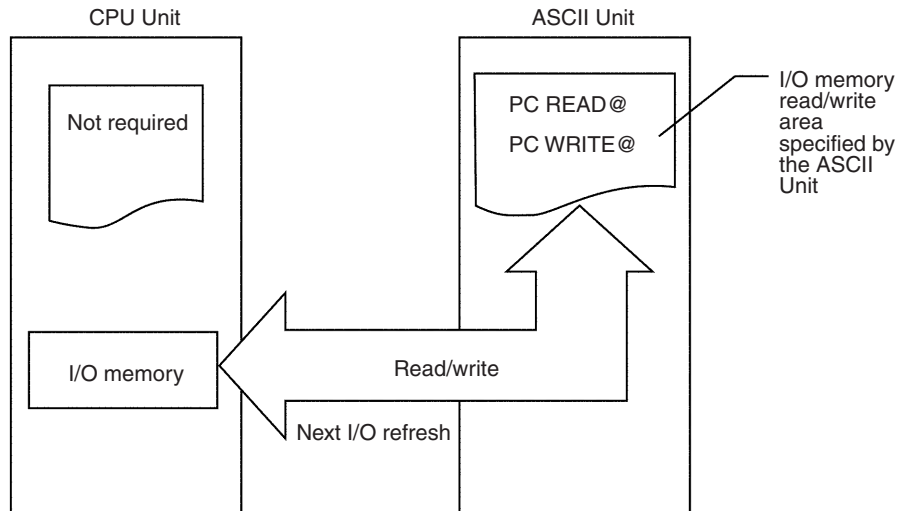
### 6-1-1 Data Exchange Methods

The ASCII Unit can exchange data with the CPU Unit in the following five ways.

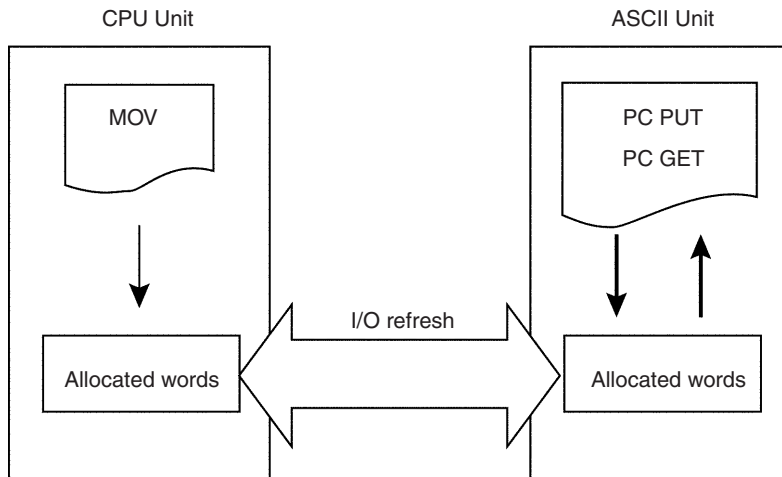
- 1,2,3... 1. The CPU Unit designates its own read/write area in memory. If a PC READ/WRITE command is then specified from the CPU Unit, the ASCII Unit uses the PC READ/WRITE command to access the memory area indicated by the CPU Unit.



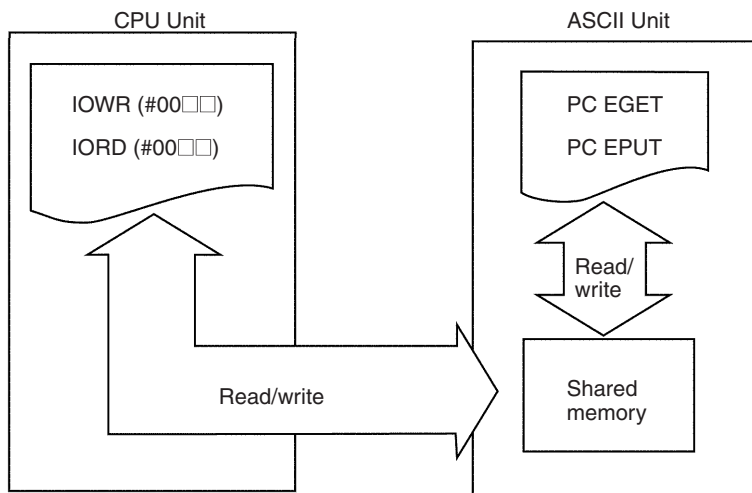
2. The ASCII Unit independently accesses the CPU Unit I/O memory. The CPU Unit's user program is not required at all.



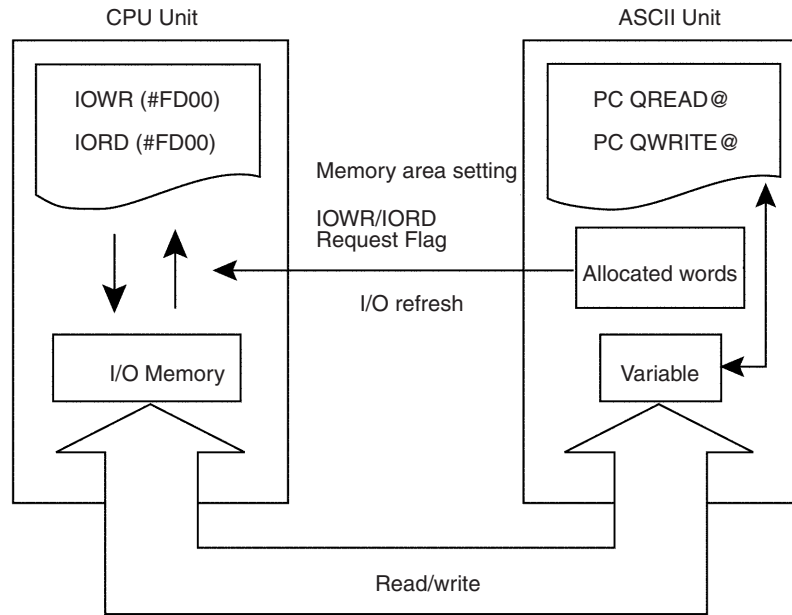
3. The CPU Unit and the ASCII Unit both access their own allocated words in memory. (The ASCII Unit accesses its own data in 16-bit units via the PC PUT/GET command.) The words allocated in the CPU Unit and memory words in the ASCII Unit are exchanged during the I/O refresh period.



4. The ASCII Unit accesses its own shared memory via a PC EGET/EPUT command. Asynchronously, the CPU Unit reads/writes the ASCII Unit's shared memory via an IORD (#00□□)/IOWR (#00□□) instruction. The IORD (#00□□)/IOWR (#00□□) instructions are supported only by the C200HX/HG/HE and CS1-series PCs.



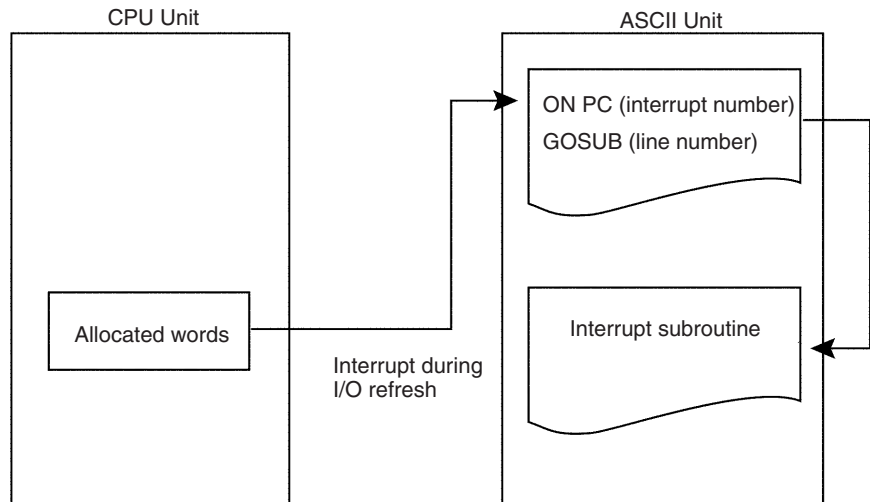
- The ASCII Unit sets an IOWR Request Flag for the CPU Unit. When an IOWR (#FD00) instruction is executed, the CPU Unit writes data. Alternately, the ASCII Unit sets an IORD Request Flag for the CPU Unit. When an IORD (#FD00) instruction is executed, the CPU Unit reads data (supported only by the C200HX/HG/HE and CS1-series PCs).



### 6-1-2 Sending Interrupts

There are two methods for sending interrupts from the CPU Unit to the ASCII Unit. It is not possible to send interrupts to the CPU Unit from the ASCII Unit.

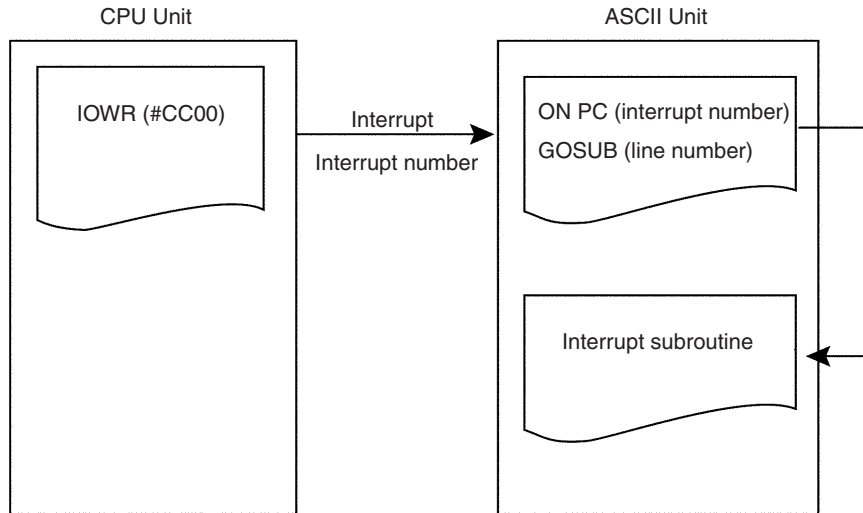
- 1,2,3... 1. Interrupts are sent during the I/O refresh period. An interrupt number is set in the allocated IR/CIO area word, and when the PC Interrupt Bit goes ON, the ON PC command branches to the specified line number.



Note PC interrupts are disabled while the ASCII Busy Flag is ON.

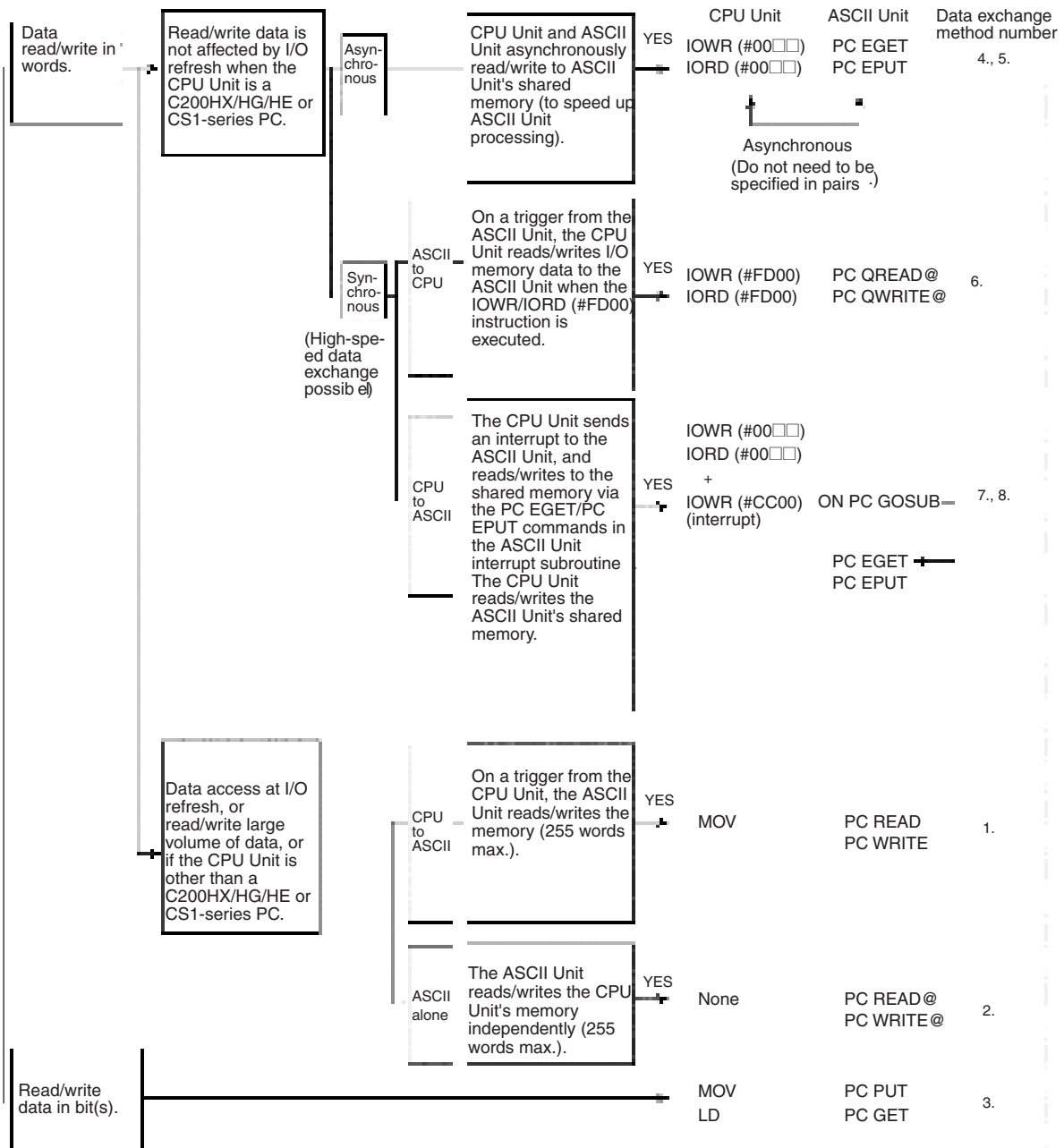


- 2. An interrupt is sent at the execution of the IOWR (#CC00) instruction. The IOWR (#CC00) instruction in the CPU Unit's user program sends an interrupt, the interrupt number is sent to the ASCII Unit, and the program branches to the line number specified in the ON PC command. If the IOWR (#00□□) instruction has previously set input data in the shared memory, it is possible to access the shared memory via the PC EGET/EPUT command in the interrupt subroutine. This is supported only by the C200HX/HG/HE and CS1-series PCs.



Note PC interrupts are disabled while the ASCII Busy Flag is ON.

## 6-2 Selecting the Data Exchange Method



**Comparison Chart of Data Exchange Methods** The differences between the data exchange methods are shown in the table below.

Data exchange method	CPU Unit instructions	ASCII Unit commands	Accessing Unit	Access trigger	Transfer timing	ASCII program	Max. no. of transfer words	
							Per command	Per cycle
1	MOV etc. (specify memory read/write area for execution of PC WRITE/PC READ commands)	PC WRITE PC READ (Reads/writes the CPU Unit's I/O memory)	ASCII Unit	CPU Unit	At I/O refresh	Pauses at execution of PC WRITE/PC READ until completion of transfer.	PC WRITE/ PC READ: 255 words	Using a C200HX/HG/HE or CS1-series PC, and when the number of bytes transferred per cycle set in the Setup Area (Bits 08 to 15 of DM m + 4) is 5A: 127 words 00: 20 words
2	Programming not required	PC WRITE@ PC READ@ (Reads/writes CPU Unit memory)	ASCII Unit	ASCII Unit	At I/O refresh	Pauses at execution of PC WRITE@/PC READ@, until completion of transfer.	PCWRITE@/ PCREAD@: 255 words	
3	MOV, LD, etc.	PC GET PC PUT (Reads/writes allocated IR/CIO area words)	Either	Either	At I/O refresh	Execution does not stop.	PC PUT: 1 word PC GET: 2 words	PC PUT: 1 word PC GET: 2 words
4, 5	IOWR (#00□□) IORD (#00□□) (C200HX/HG/HE or CS1-series PC only)	PC EGET PC EPUT (Reads/writes to ASCII Unit's shared memory)	CPU Unit	Either	At IOWR/IORD (#00□□) execution	Execution does not stop.	PC EGET/ PC EPUT: 90 words IOWR/IORD (#00 □□): 90 words	No restrictions
6	IOWR (#FD00) IORD (#FD00) (C200HX/HG/HE or CS1-series PC only)	PC QREAD@ PC QWRITE@	CPU Unit	ASCII Unit	At IOWR/IORD (#FD00) execution	Pauses at execution of PC QREAD/PC QWRITE until IOWR/IORD (#FD00) has been executed.	PC QREAD/ PC QWRITE: 128 words IOWR, IORD (#FD00): 128 words	No restrictions
7, 8	IOWR (#00□□) IORD (#00□□) + IOWR (#CC00) (C200HX/HG/HE or CS1-series PC only)	ON PC GOSUB PC EGET/ EPUT	CPU Unit	CPU Unit	At IOWR/IORD execution	Execution does not stop.		No restrictions

**Note** There are some restrictions in the memory area addresses that can be used for methods 1 and 2. Refer to 6-6 Restrictions for CS1-series PCs for details. Use methods 3 to 8 whenever possible.

## 6-3 Details of the Data Exchange Methods

### 6-3-1 Data Exchange Method No. 1

The ASCII Unit reads/writes the CPU Unit memory during the I/O refresh period on a trigger from the CPU Unit.

This method allows the exchange of comparatively large volumes of data with a maximum of 255 words, although it will cause the ASCII Unit to be on standby for some time. It is possible with any CPU Unit.

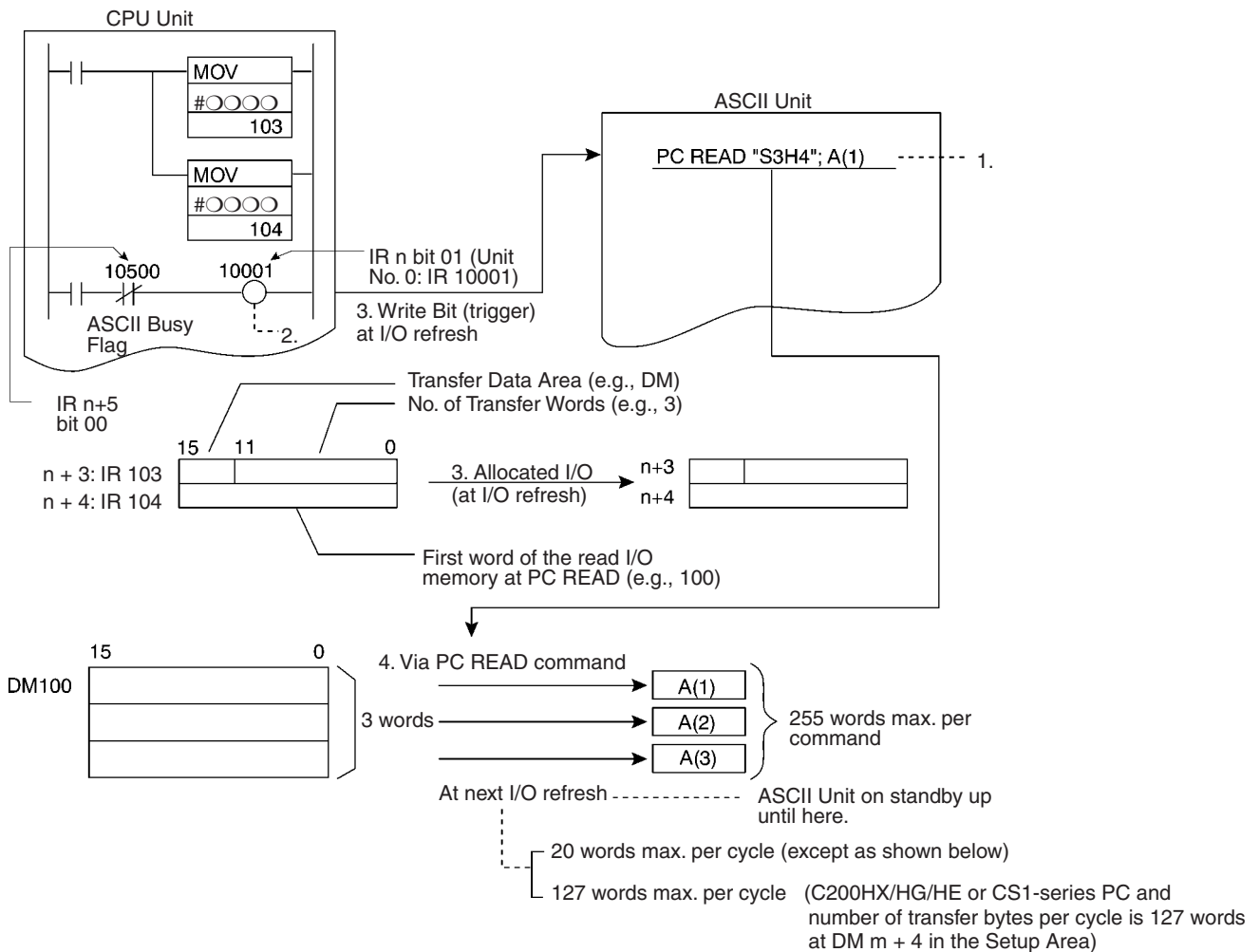
PC READ

When the ASCII Unit receives an ON Write Bit from the CPU Unit, it reads the memory words specified in the allocated IR/CIO words, and stores the contents in the specified variable.

1,2,3...

1. Request using the PC READ command.
2. Write Bit turned ON using the OUT instruction in the ladder diagram.
3. During the I/O refresh period, the Write Bit (IR/CIO n bit 01) from the CPU Unit to the ASCII Unit turns ON.
4. At the next I/O refresh period, the ASCII Unit reads data and the ASCII Busy Flag is turned ON by the system.
5. When the ASCII Busy Flag changes to ON, the Write Bit is turned OFF in the ladder diagram.
6. Once the PC READ command's data transfer has been completed, the ASCII Busy Flag is turned OFF by the system.

The following example is for a C200HX/HG/HE PC.

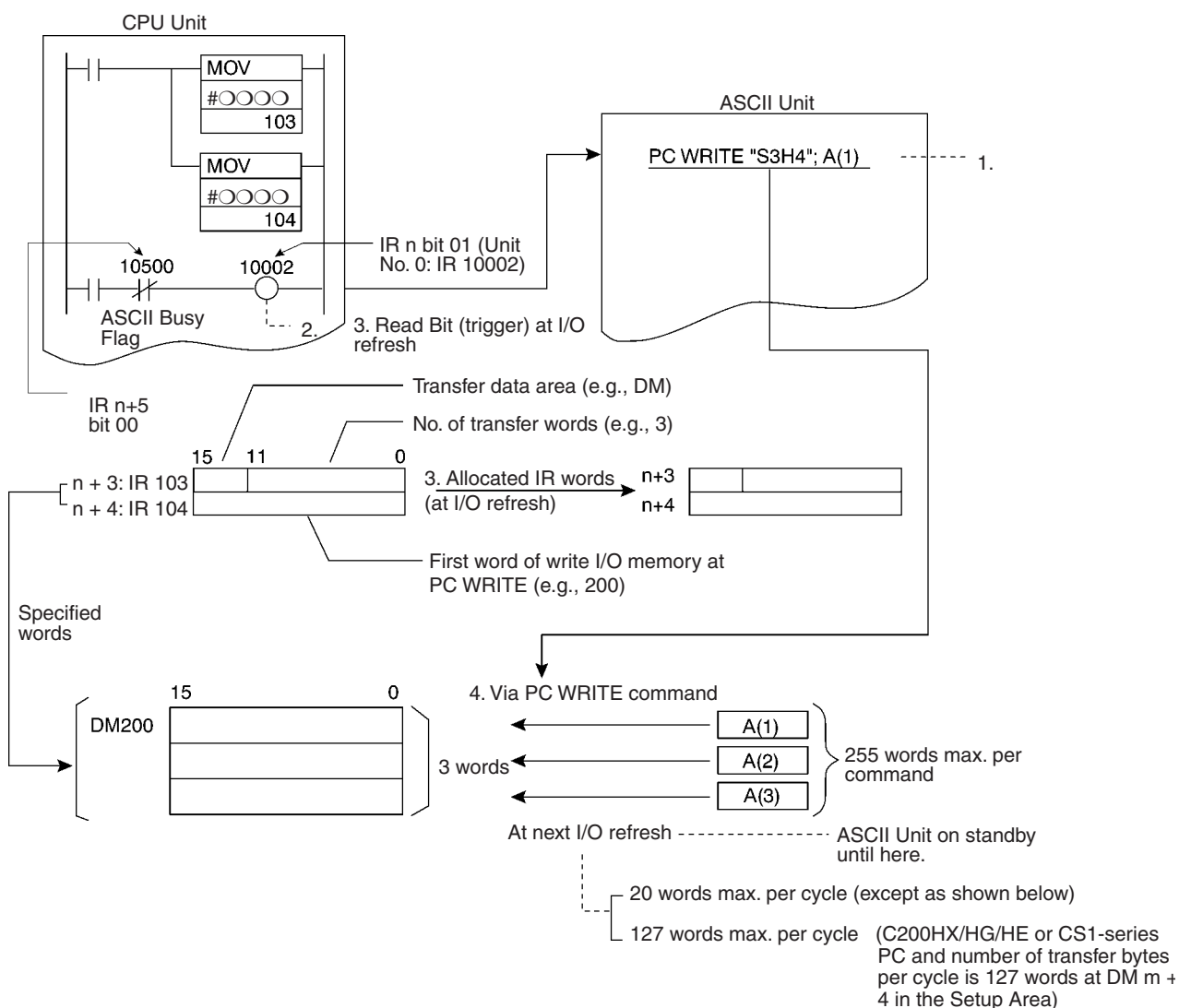


**PC WRITE**

When the ASCII Unit receives an ON Read Bit from the CPU Unit, it writes the contents of the specified variable to the I/O memory area specified in the allocated I/O.

- 1,2,3...**
1. Request by PC WRITE command.
  2. Read Bit turned ON via the OUT command in the ladder diagram
  3. During the I/O refresh period, the Read Bit (n bit 02) from the CPU Unit to the ASCII Unit turns ON.
  4. At the next I/O refresh period, the ASCII Unit writes data (ASCII Busy Flag goes ON).
  5. When the ASCII Busy Flag changes to ON, the Read Bit turned OFF in the ladder diagram.
  6. Once the PC WRITE command data transfer has been completed, the ASCII Busy Flag is turned OFF by the system.

The following example is for a C200HX/HG/HE PC.



### 6-3-2 Data Exchange Method No. 2

The ASCII Unit independently reads/writes the CPU Unit memory during the I/O refresh period.

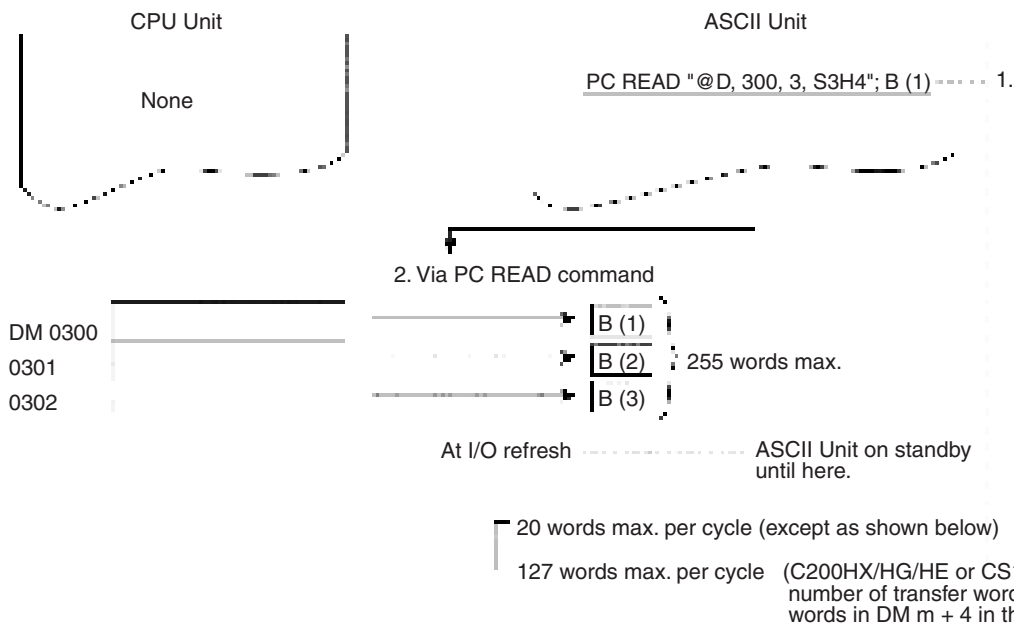
#### PC READ@

The ASCII Unit reads the CPU Unit memory as specified in the PC READ@ command.

1,2,3...

1. PC READ@ command executed.
2. The ASCII Unit reads data during the I/O refresh period (the ASCII Busy Flag goes ON).
3. When the PC READ@ command data transfer is completed, the ASCII Busy Flag goes OFF.

The following example is for a C200HX/HG/HE PC.



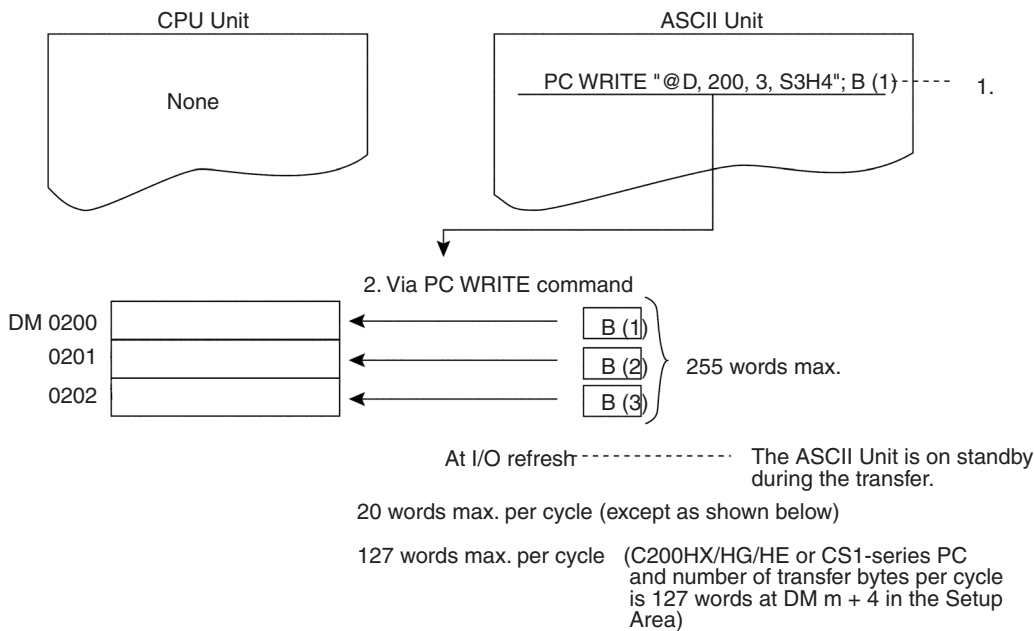
#### PC WRITE@

The ASCII Unit writes to the CPU Unit's internal I/O memory, as specified in the PC WRITE@ command.

1,2,3...

1. PC WRITE@ command executed.
2. The ASCII Unit writes data during the I/O refresh period (the ASCII Busy Flag goes ON).
3. When the PC WRITE@ command data transfer is completed, the ASCII Busy Flag goes OFF.

The following example is for a C200HX/HG/HE PC.



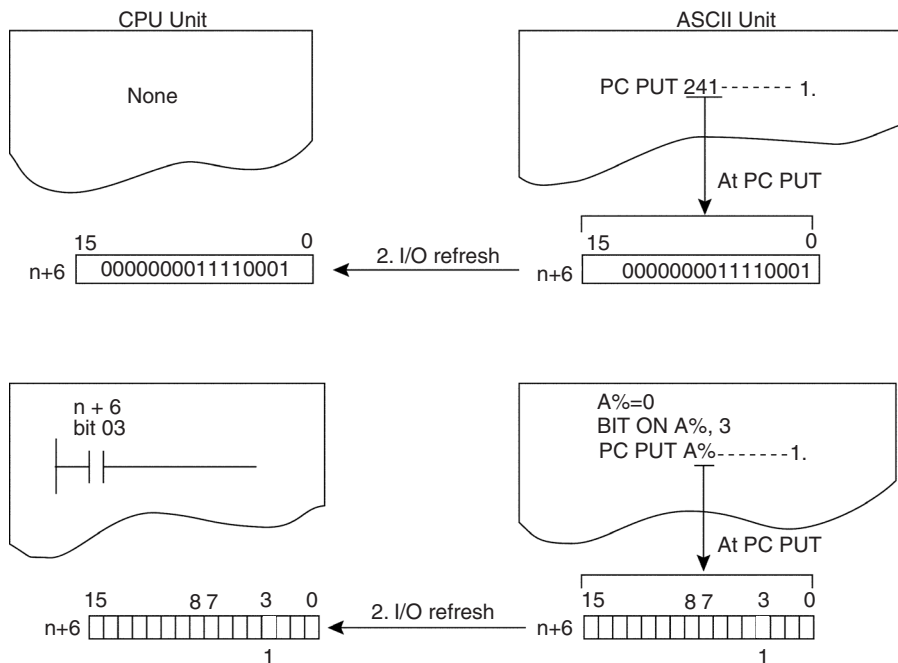
### 6-3-3 Data Exchange Method No. 3

Single-bit or multibit data is read/written during the I/O refresh period from either the ASCII Unit or the CPU Unit.

#### PC PUT

The ASCII Unit converts decimal data to hexadecimal data and stores it at allocated word IR/CIO n + 6. It writes decimal values from -32768 to 32767.

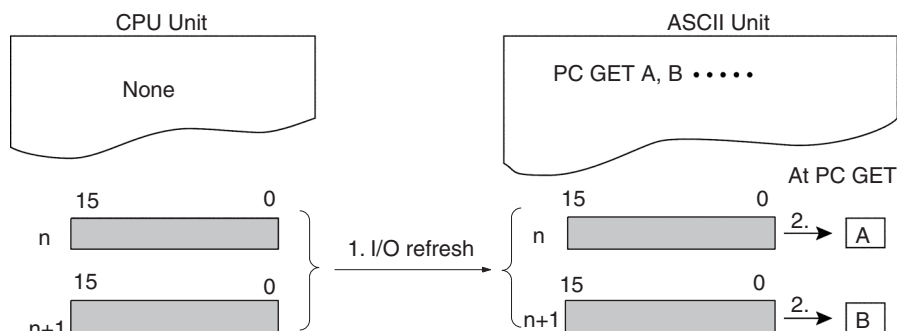
- 1,2,3... 1. PC PUT command executed.
2. The ASCII Unit writes data during the I/O refresh period.



**PC GET**

The ASCII Unit converts the allocated word IR/CIO n bits 0 to 15 from hexadecimal data to decimal data, and stores them in the first variable. It also converts the word IR/CIO n + 1 bits 0 to 15 from hexadecimal to decimal data and stores them in the second variable.

- 1,2,3...**
1. Data set during I/O refresh period.
  2. At the execution of the PC GET command, the ASCII Unit reads the specified data.



**6-3-4 Data Exchange Method No. 4**

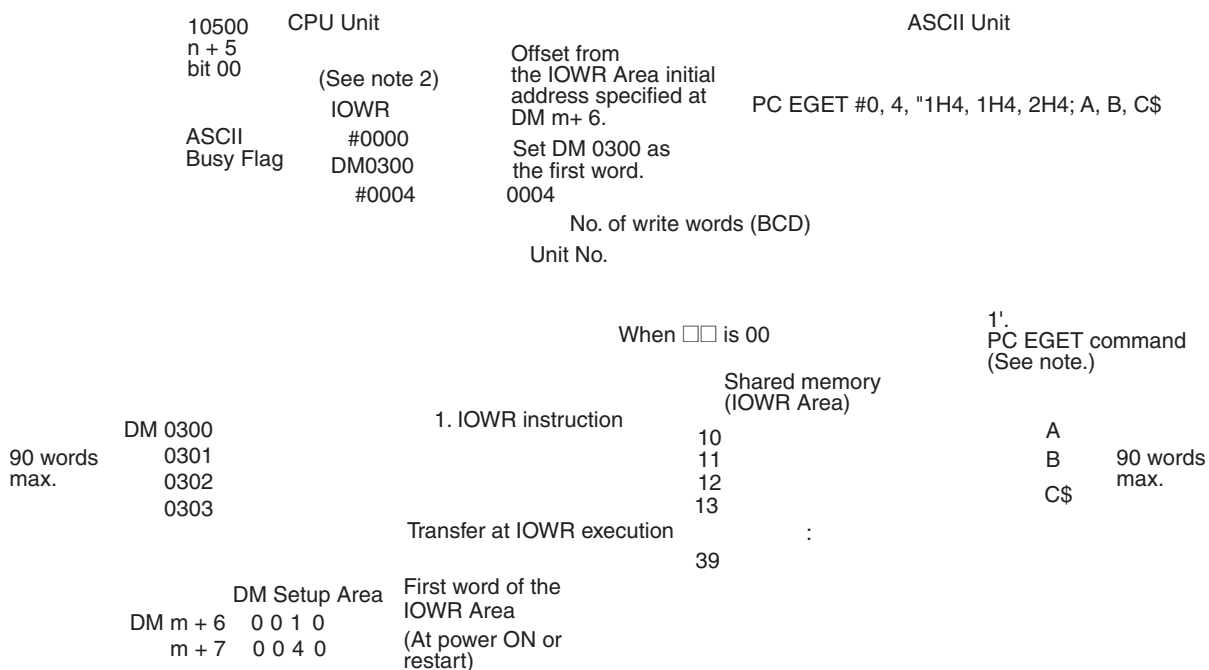
The ASCII Unit and CPU Unit both asynchronously read/write to the ASCII Unit's shared memory.

**PC EGET**

Using the PC EGET command, the ASCII Unit reads the IOWR Area of its shared memory area to a specified variable. Asynchronously, the CPU Unit writes specified memory area data to the ASCII Unit's shared memory areas using the IOWR instruction.

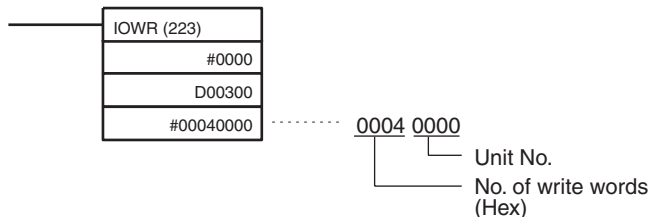
- 1,2,3...**
1. Using the IOWR (#00□□) instruction, write data to the ASCII Unit's shared memory area.
  2. Using the PC EGET command, read data from the shared memory area.

The following example is for a C200HX/HG/HE PC.





- Note**
1. The PC EGET/EPUT commands can synchronize with the PC as they are executed by interrupts from the PC. PC interrupts are possible via IOWR instruction (control code #CC00) and allocated IR/CIO area words (word IR/CIO n bit 04 and word IR/CIO n + 2 bits 00 to 07).
  2. The IOWR instruction is structured as shown below for the CS1-series PCs. The number of write words must be specified as a hexadecimal value.

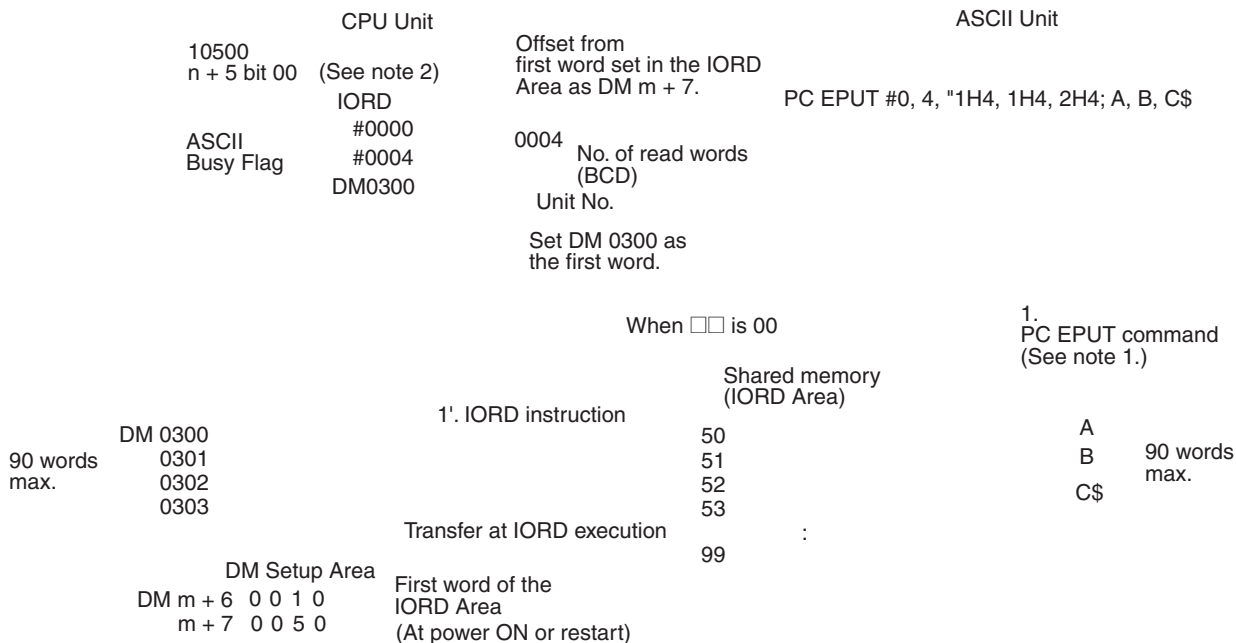


**PC EPUT**

The ASCII Unit writes the contents of specified variables to its shared memory IORD Area using the PC EPUT command. Asynchronously, the CPU Unit reads data from the ASCII Unit's shared memory and writes it to the specified I/O memory area using the IORD instruction.

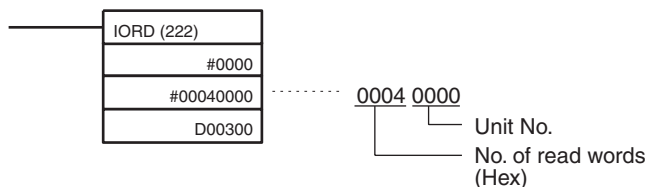
- 1,2,3...**
1. Write data to the shared memory area using the PC EPUT command.
  2. Read data from the ASCII Unit's shared memory area when the IORD (#00□□) instruction is executed in the ladder diagram.

The following example is for a C200HX/HG/HE PC.



- Note**
1. The PC EGET/EPUT commands can synchronize with the PC as they are executed by interrupts from the PC. PC interrupts are possible via IOWR instruction (control code #CC00) and allocated IR/CIO area words (word IR/CIO n bit 04 and word IR/CIO n + 2 bits 00 to 07).

- The IORD instruction is structured as shown below for the CS1-series PCs. The number of read words must be specified as a hexadecimal value.



### 6-3-5 Data Exchange Method No. 5

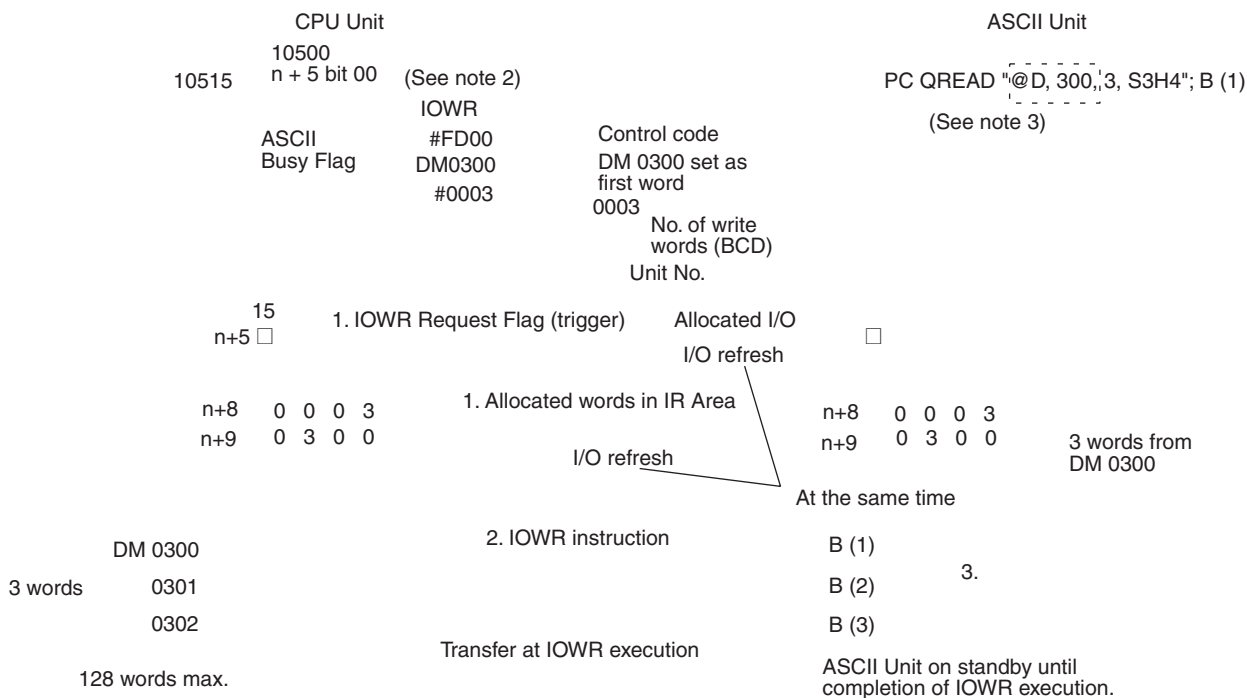
On a trigger from the ASCII Unit, the CPU Unit reads/writes memory data to the ASCII Unit by executing an IOWR/IORD (#FD00) instruction.

#### PC QREAD (Combined with IOWR (#FD00))

The ASCII Unit sets an IOWR Request Flag in the CPU Unit in bit 15 of IR/CIO n + 5. When the CPU Unit detects the ON status of the flag, it will execute the IOWR (#FD00) instruction. Based on IR/CIO n + 8 and n + 9, it writes the specified words to the ASCII Unit.

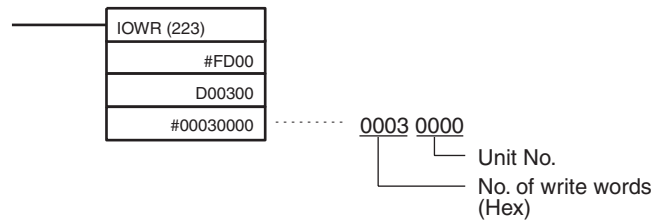
- At I/O refresh, the ASCII Unit sends an IOWR Request Flag to the CPU Unit by the system.
- Data is written to the ASCII Unit when the IOWR (#FD00) instruction is executed in the ladder diagram.
- As a result of the PC QREAD command, the received data is read to a specified variable.

The following example is for a C200HX/HG/HE PC.



**Note** 1. IORD and IOWR instructions cannot be used for ASCII Units on Slave Racks.

- The IOWR instruction is structured as shown below for the CS1-series PCs. The number of write words must be specified as a hexadecimal value.



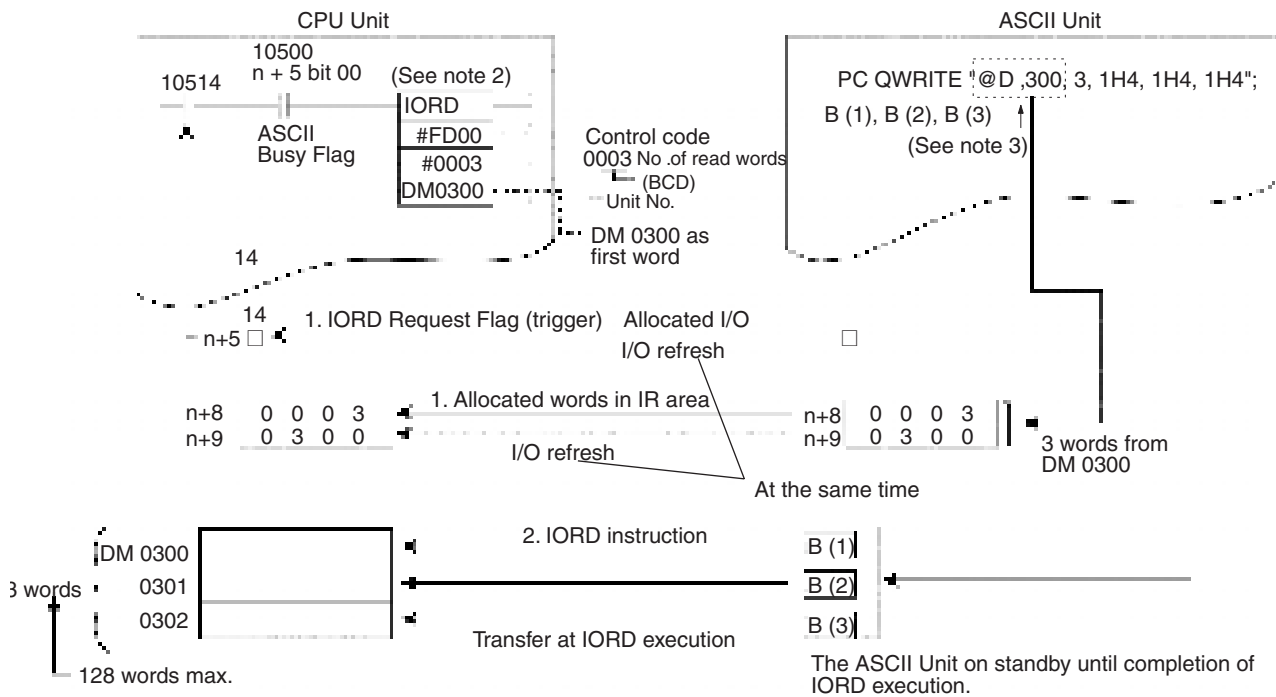
- The data set in words n+8 and n+9 by PC QREAD is used only as monitor data by the CPU Unit. The data that is actually transferred is specified in an operand of the IOWR instruction. This aspect of operation differs from that of the PC READ@ command, which reads data from the operand specified for it. For the CS1 Series, the range that can be addressed for PC QREAD differs from the range for IOWR, making it necessary to use dummy addresses when executing PC QREAD if it cannot address the required words directly. (The CPU Unit will not monitor the words when dummy addresses are used.) In the above example, the address of the first word, D07000 cannot be specified for PC QREAD even though it can be specified for IOWR. Here, a dummy address, like @D,0, can be used for PC QREAD.

**PC QWRITE (Combined with IORD (#FD00))**

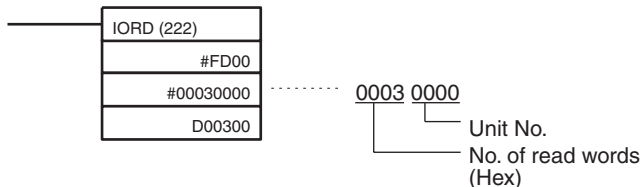
The ASCII Unit sends an IORD Request Flag to the CPU Unit by bit 14 of IR/CIO n + 5. When the CPU Unit detects the ON status of the flag, it will execute the IORD instruction. Based on IR/CIO n + 8 and n + 9, it reads the words from the ASCII Unit.

- 1,2,3...
- At I/O refresh, the ASCII Unit sends an IORD Request Flag to the CPU Unit by the system.
  - Data is prepared for transmission from the specified variable by the PC QWRITE command.
  - Data is read from the ASCII Unit when the IORD (#FD00) instruction is executed in the ladder diagram.

The following example is for a C200HX/HG/HE PC.



- Note**
1. IORD and IOWR instructions cannot be used for ASCII Units on Slave Racks.
  2. The IORD instruction is structured as shown below for the CS1-series PCs. The number of read words must be specified as a hexadecimal value.



3. The data set in words n+8 and n+9 by PC QWRITE is used only as monitor data by the CPU Unit. The data that is actually transferred is specified in an operand of the IORD instruction. This aspect of operation differs from that of the PC WRITE@ command, which reads data from the operand specified for it. For the CS1 Series, the range that can be addressed for PC QWRITE differs from the range for IORD, making it necessary to use dummy addresses when executing PC QWRITE if it cannot address the required words directly. (The CPU Unit will not monitor the words when dummy addresses are used.) In the above example, the address of the first word, D07000 cannot be specified for PC QWRITE even though it can be specified for IORD. Here, a dummy address, like @D,0, can be used for PC QWRITE.

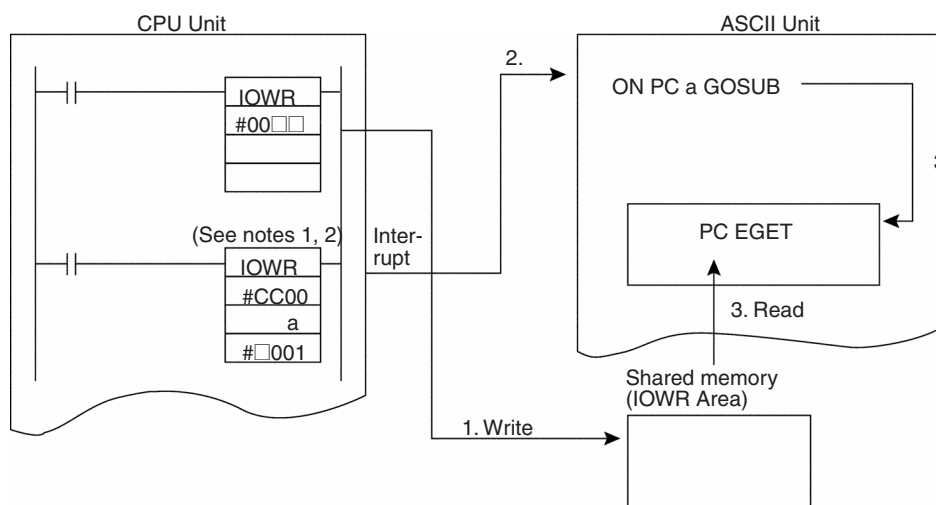
### 6-3-6 Data Exchange Method No. 6

After the CPU Unit has written to the ASCII Unit's shared memory area, it sends an interrupt to the ASCII Unit, and during the interrupt subroutine, the ASCII Unit reads from the shared memory.

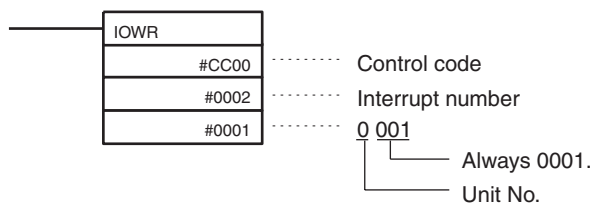
**Note** See 6-4-9 Sending Interrupts from CPU Unit to ASCII Unit: PC Interrupts for details on PC interrupts.

- 1,2,3...**
1. The CPU Unit writes to the ASCII Unit shared memory area (IOWR Area) when the IOWR instruction (#00□□) is executed in the ladder diagram.
  2. The CPU Unit sends an interrupt to the ASCII Unit when the IOWR instruction (#CC00) is executed in the ladder diagram.
  3. Once the interrupt is received by the ASCII Unit, data is read in the ASCII Unit via the PC EGET command.

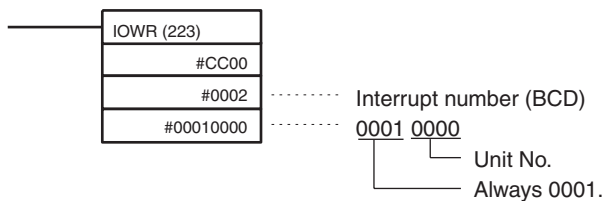
The following example is for a C200HX/HG/HE PC.



**Note** 1. The following IOWR instruction would be used if the ASCII Unit was unit number 0 and the interrupt number was 2.



2. The structure of the IOWR instruction for the CS1-series PCs is shown below.



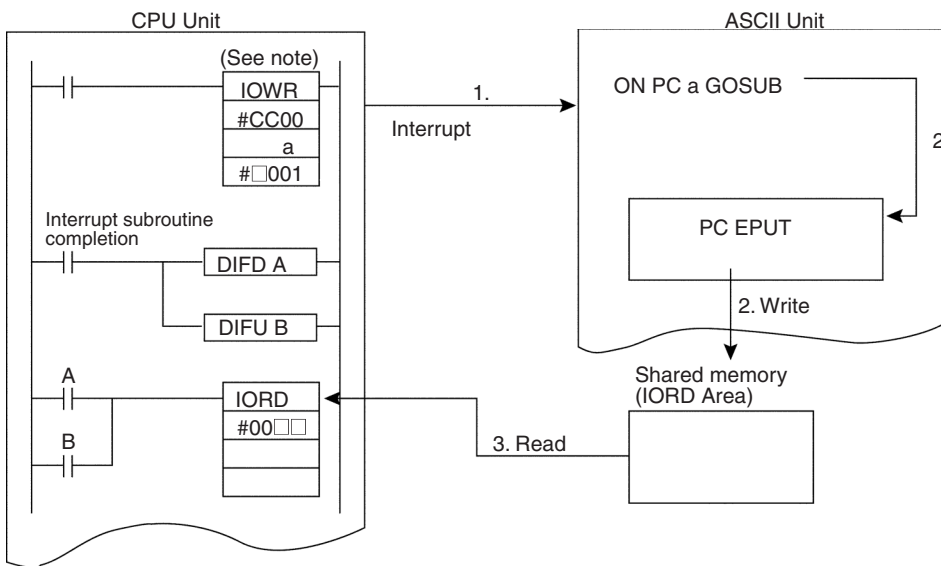
### 6-3-7 Data Exchange Method No. 7

The CPU Unit sends an interrupt to the ASCII Unit, and the ASCII Unit writes data to the shared memory. After the completion of the interrupt subroutine, the CPU Unit reads data from the shared memory.

**Note** See 6-4-9 Sending Interrupts from CPU Unit to ASCII Unit: PC Interrupts for details on PC interrupts.

- 1,2,3...**
1. The CPU Unit sends an interrupt to the ASCII Unit when the IOWR instruction (#CC00) is executed in the ladder diagram.
  2. The ASCII Unit receives the interrupt and writes data to the shared memory using a PC EPUT command.
  3. When a change is detected in the Interrupt Subroutine Completed Flag (1 to 0 or 0 to 1), the shared memory is read when the IORD instruction (#00□□) is executed in the ladder diagram.

The following example is for a C200HX/HG/HE PC.

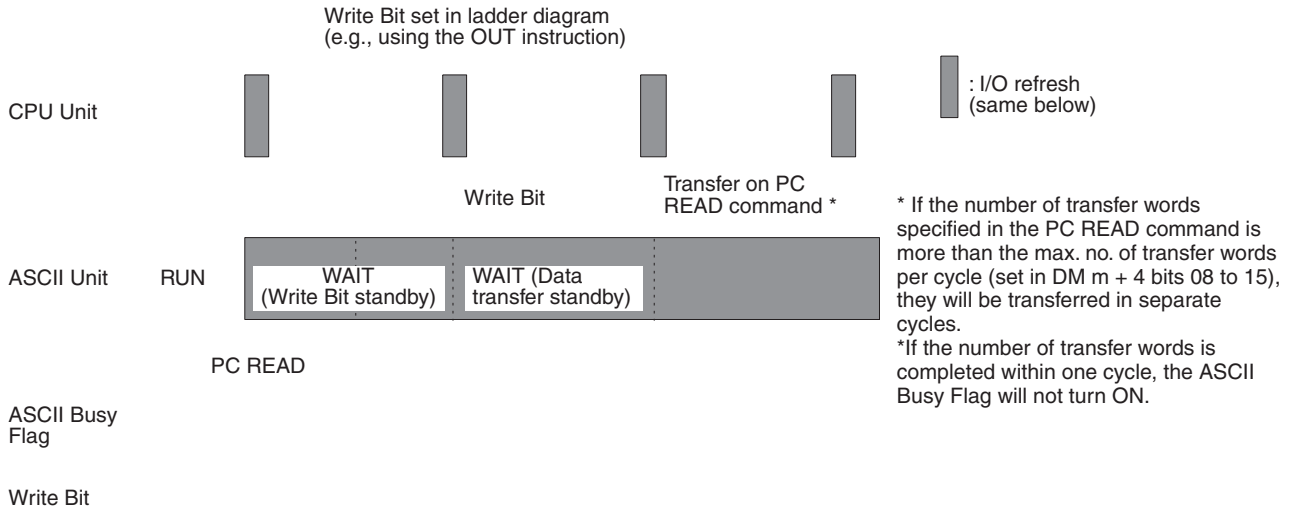


**Note** Refer to the note for method 6 on the previous page.

## 6-4 Data Exchange Time Charts

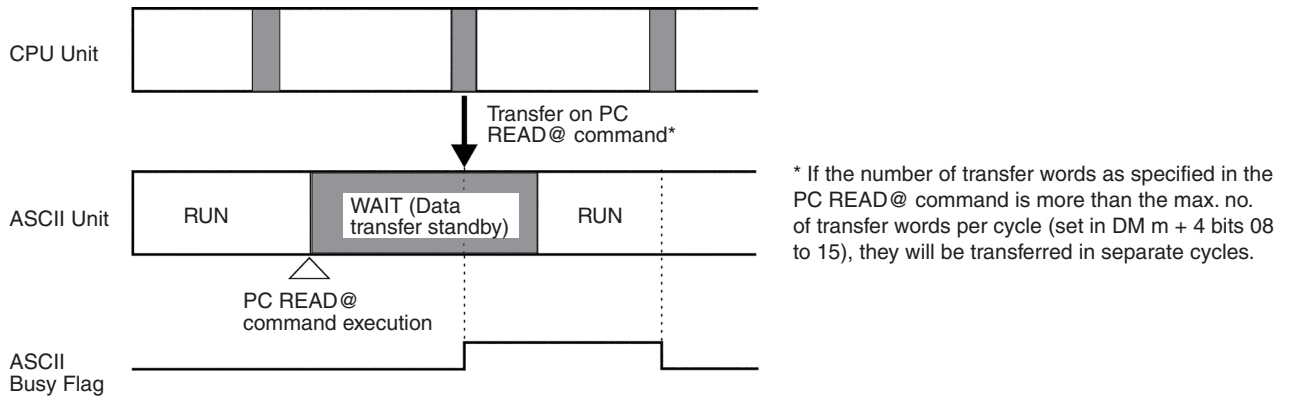
### 6-4-1 PC READ/PC WRITE Timing

After a trigger is sent from the CPU, data is transferred at the next I/O refresh period. After receiving a PC WRITE or PC READ command, the ASCII Unit puts the BASIC program on standby until the data transfer is complete.



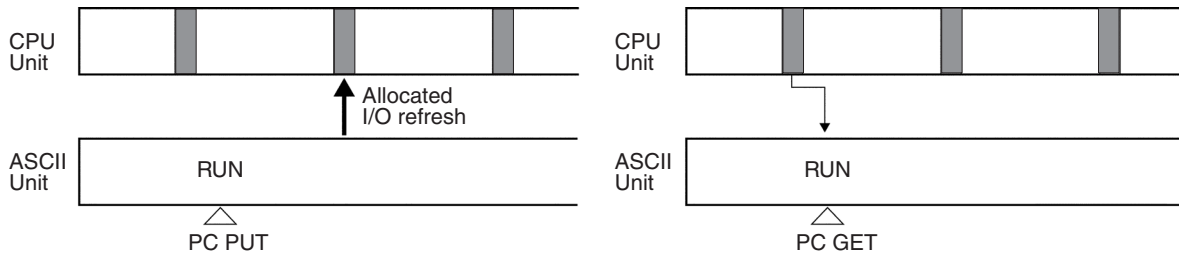
### 6-4-2 PC READ@/PC WRITE@ Timing

Data transfer occurs during the I/O refresh period immediately after the execution of the PC READ@/PC WRITE@ command. The ASCII Unit puts the BASIC program on standby until the completion of the PC READ@/PC WRITE@ command.



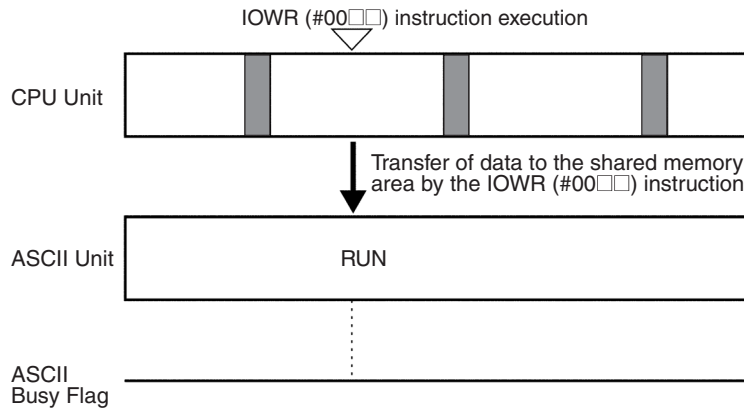
### 6-4-3 PC PUT/PC GET Timing

Data is written during the I/O refresh period immediately after the execution of the PC PUT command. The ASCII Unit BASIC program continues execution. Data is read at the I/O refresh immediately after the execution of the PC GET command.



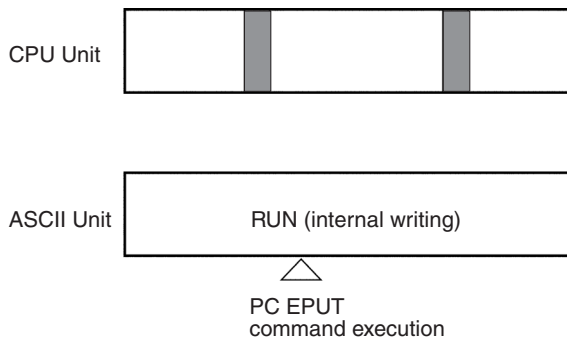
### 6-4-4 IOWR (#00□□)/IORD (#00□□) Timing

Data transfer occurs immediately after the IOWR (#00□□)/IORD (#00□□) instruction is executed. The ASCII Unit BASIC program continues execution.



### 6-4-5 PC EPUT/PC EGET Timing

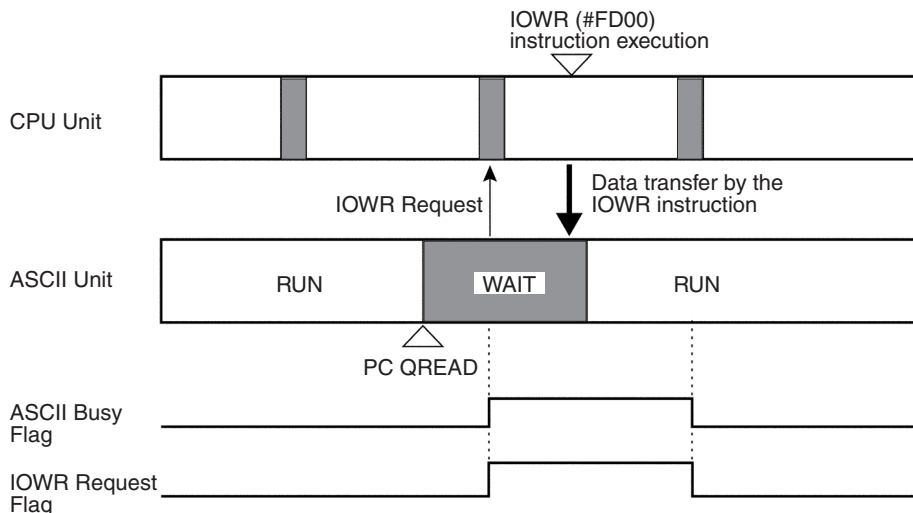
The internal shared memory is read/written to immediately when the execution of the PC EPUT/PC EGET command. The ASCII Unit's BASIC program continues execution.





### 6-4-6 IOWR (#FD00)/IORD (#FD00) and PC QREAD/PC QWRITE Timing

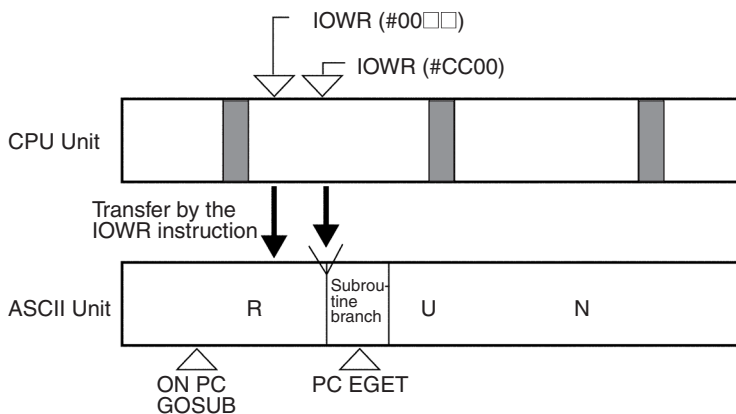
Data transfer occurs as soon as a trigger arrives from the ASCII Unit and the IOWR (#FD00)/IORD (#FD00) command is executed. The ASCII Unit is on standby from the execution of the PC QREAD/PC QWRITE command until the execution of the IOWR (#FD00)/IORD (#FD00) command.



**Note** The ASCII Busy Flag turns ON for PC QREAD only. It does not turn ON for PC QWRITE.

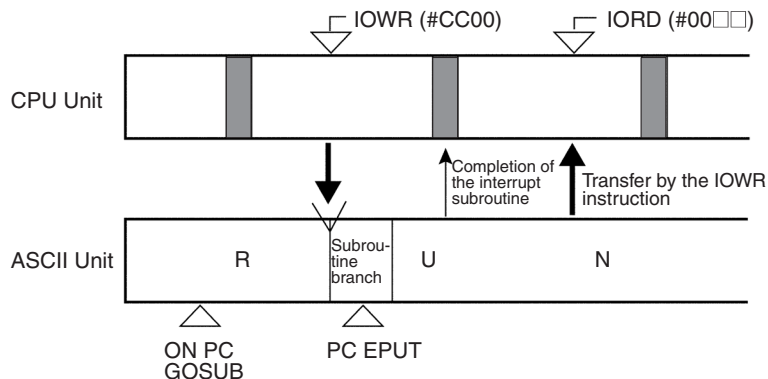
### 6-4-7 Combined IOWR (#00□□), IOWR (#CC00), ON PC GOSUB, and PC EGET Timing

Data is transferred immediately by the IOWR (#00□□) instruction. The IOWR (#CC00) instruction sends an interrupt to the ASCII Unit. The ASCII Unit receives data via the PC EGET command in an interrupt subroutine. Data transfer in the CPU Unit is completed within 1 cycle.



### 6-4-8 Combined IOWR (#CC00), IORD (#00□□), ON PC GOSUB, and PC EPUT Timing

The IOWR (#CC00) instruction sends an interrupt to the ASCII Unit. The interrupt subroutine is executed immediately and writes to the shared memory using the PC EPUT command. At the next I/O refresh period, the completion of the interrupt subroutine is transferred to the CPU Unit, and at the next cycle data is transferred by the IORD (#00□□) instruction.

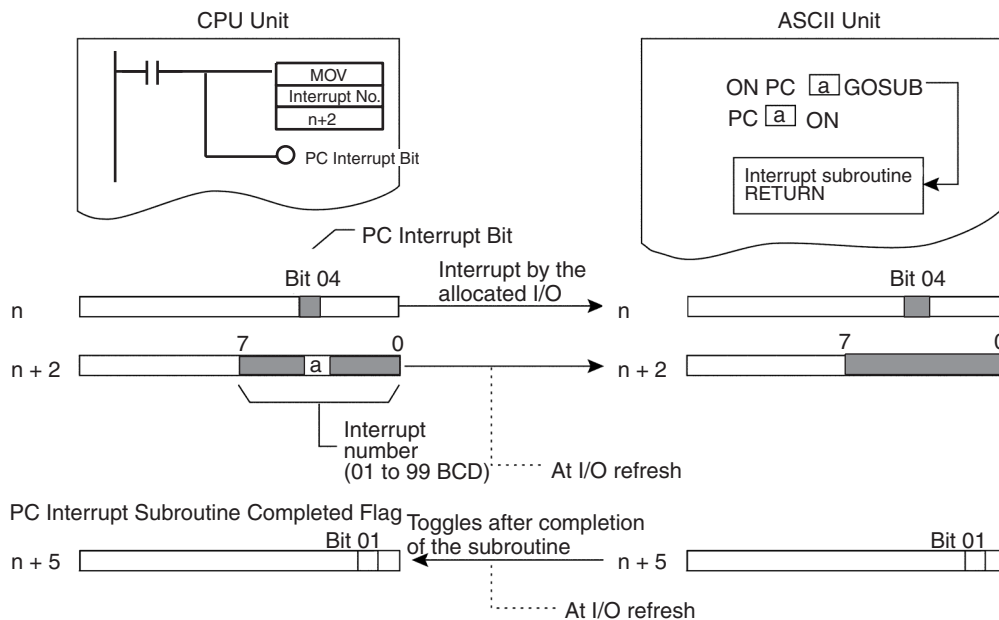


### 6-4-9 Sending Interrupts from CPU Unit to ASCII Unit: PC Interrupts

There are two methods for sending PC Interrupts as described below.

#### Using PC Interrupt Bit

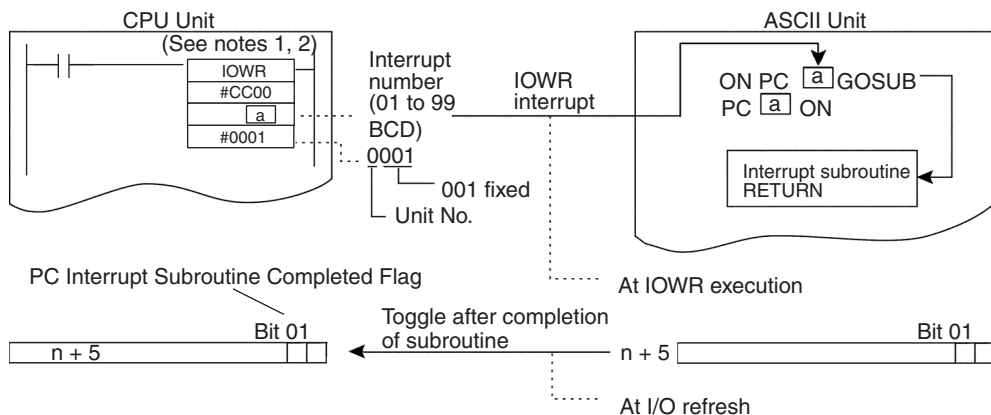
The CPU Unit sets an interrupt number in the ASCII Unit in the allocated words in the IR/CIO area. When the PC Interrupt Bit turns ON, an interrupt is sent. The ASCII Unit executes the interrupt subroutine if the interrupt number is the same as the ON PC command interrupt number. When the interrupt subroutine has been completed, the Interrupt Subroutine Completed Flag is toggled (OFF to ON or ON to OFF).



**Using IOWR Interrupt:  
Control Code: #CC00  
(C200NX/HG/HE and  
CS-1series PCs only)**

The CPU Unit indicates an interrupt number to the ASCII Unit by using the IOWR (#CC00) instruction, and sends an interrupt. The ASCII Unit executes the interrupt subroutine which matches the interrupt number in an ON PC command. When the interrupt subroutine is completed, the Interrupt Subroutine Completed Flag is toggled (OFF to ON or ON to OFF). This method is supported only by the C200HX/HG/HE and CS1-series PCs.

The following example is for a C200HX/HG/HE PC.

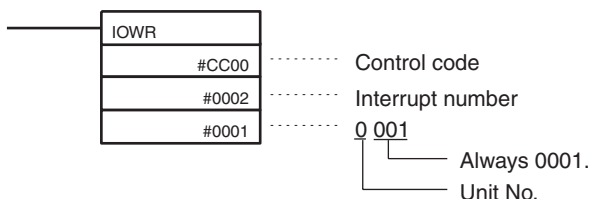


When using the PC Interrupt Bit described on the previous page, the interrupt is sent as soon as the IOWR instruction is executed, in contrast to interrupting using the PC Interrupt Bit (described on previous page), by which the interrupt is sent during the I/O refresh period. For this reason, high-speed interrupts to the ASCII Unit occur in the same cycle, not in the next cycle.

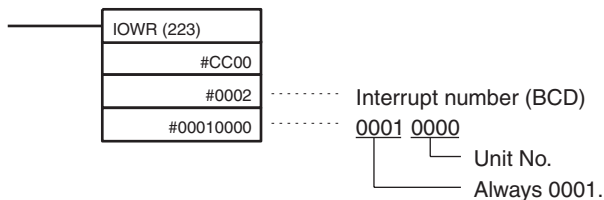
By combining the IOWR/IORD (#00□□) instruction at the CPU Unit with the PC EGET/PC EPUT commands in the ASCII Unit's interrupt subroutine, reads/writes between the CPU Unit and the ASCII Unit can occur within a single cycle.

The following is an example of a program.

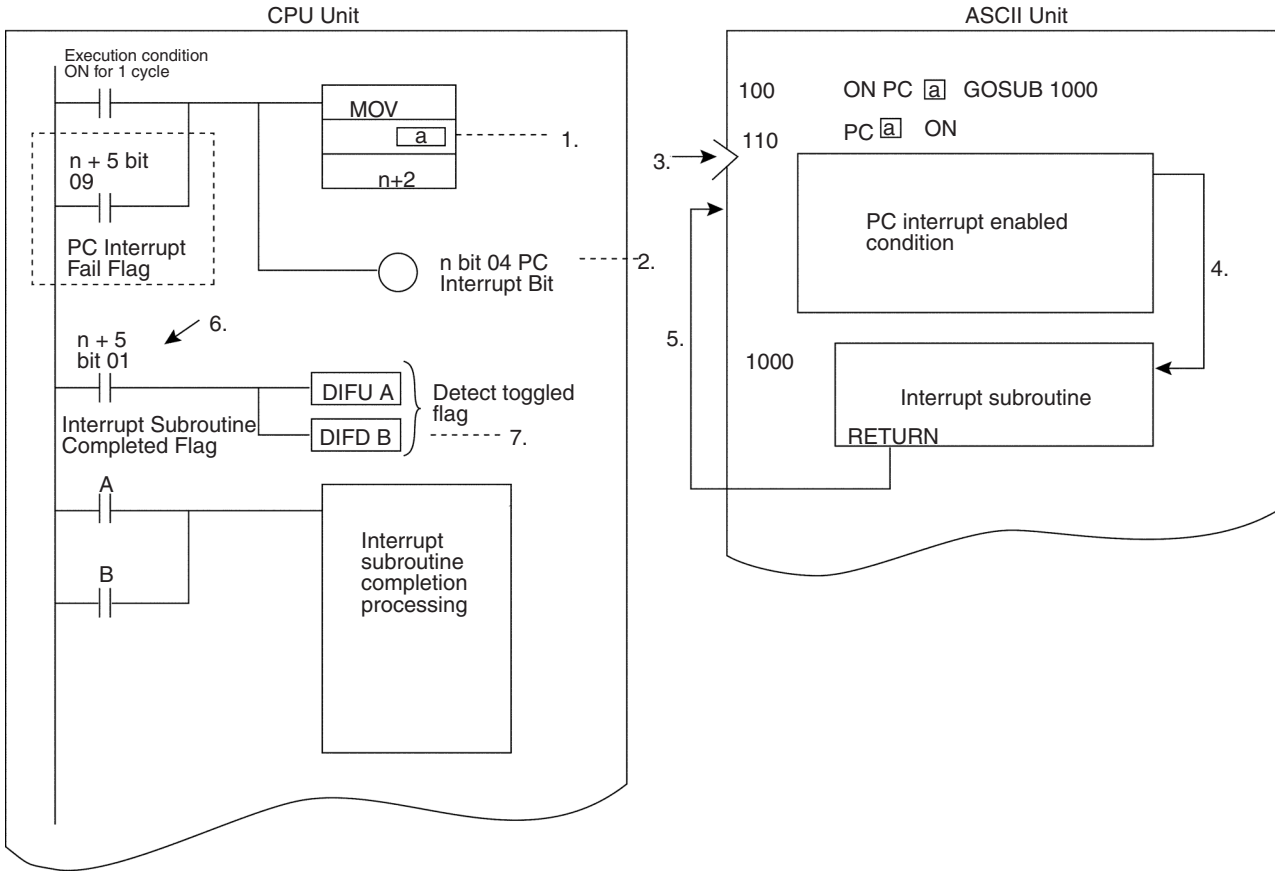
- Note** 1. The following IOWR instruction would be used if the ASCII Unit was unit number 0 and the interrupt number was 2.



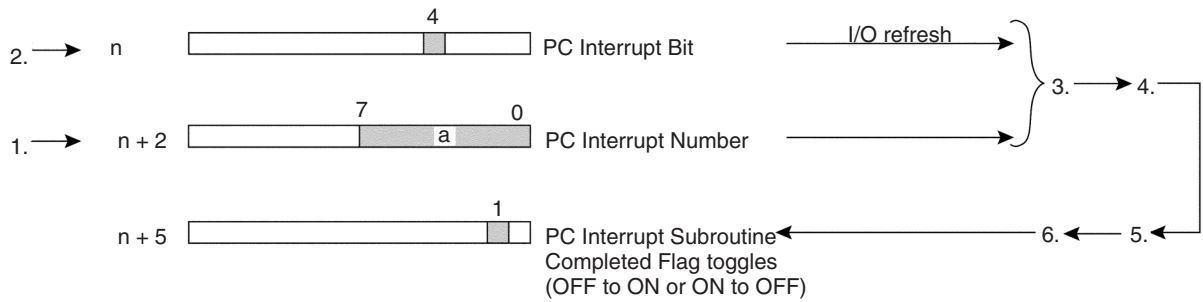
2. The structure of the IOWR instruction for the CS1-series PCs is shown below.



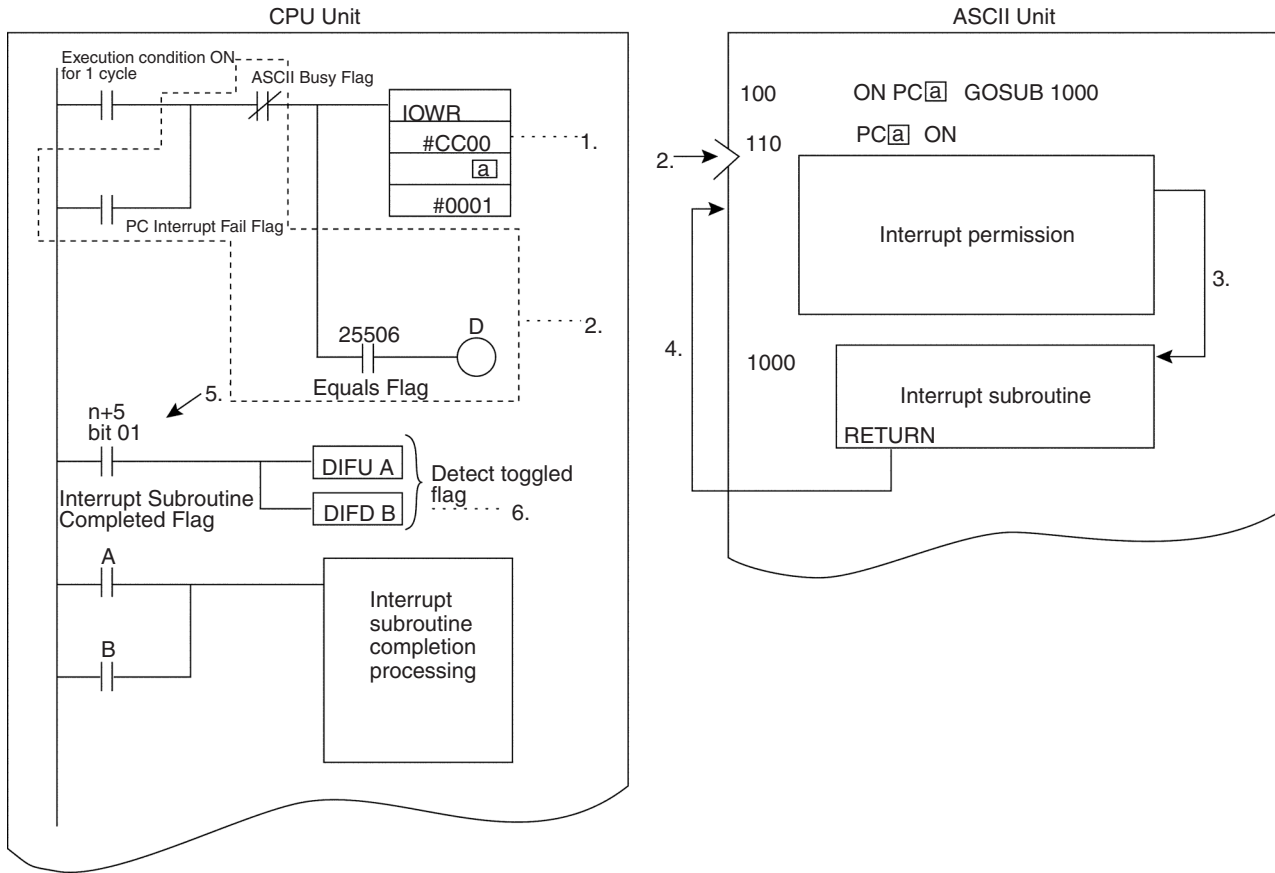
Interrupt by PC Interrupt Bit



**Note** Areas within dotted lines denote retry processing when PC interrupt has failed due to conditions such as a PC OFF (stop interrupt) condition in the ASCII Unit. If an interrupt is not accepted due to PC OFF condition when attempting to execute the interrupt by turning ON the PC Interrupt Bit (word n bit 04), the Interrupt Fail Flag (word n + 5 bit 09) will be set to ON (1) for one scan in the next cycle.



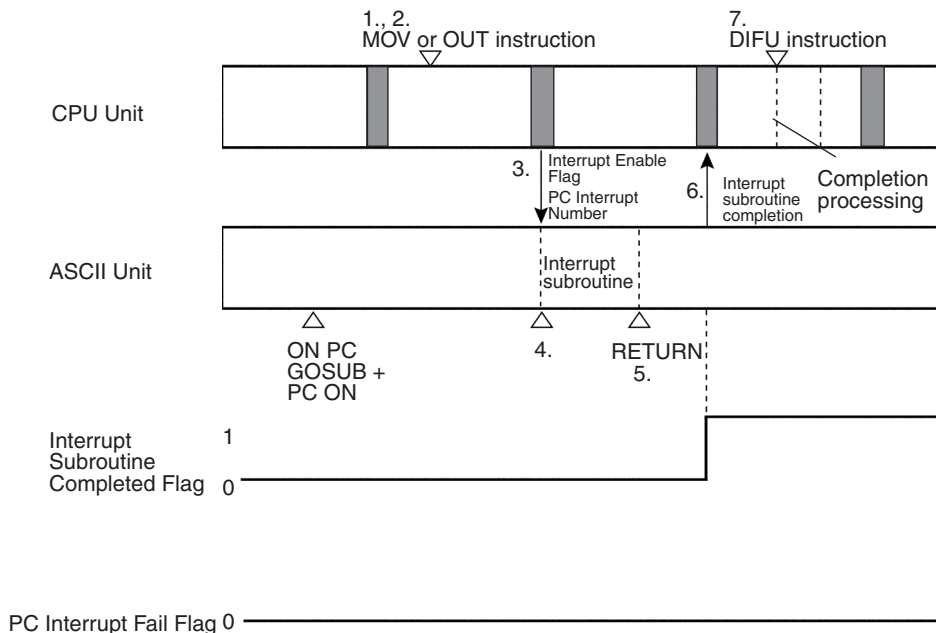
IOWR (#CC00) Interrupt: C200HX/HG/HE or CS1-series PCs Only



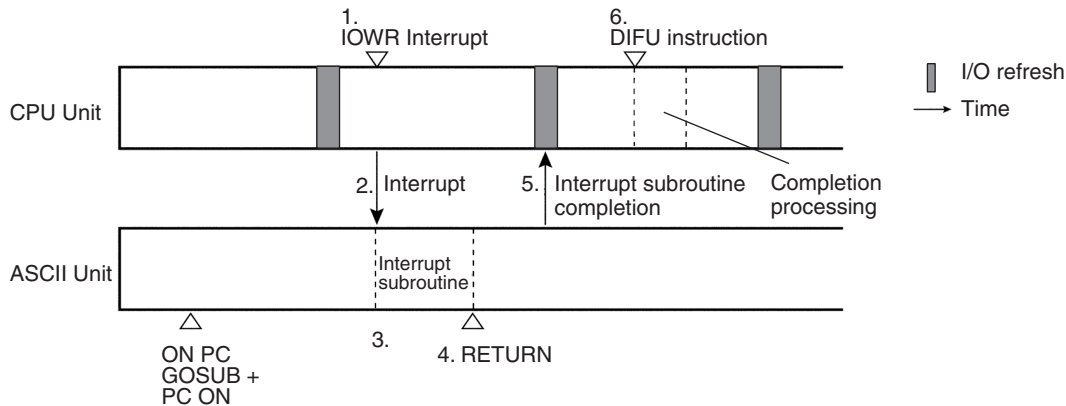
**Note** Areas within dotted boxes denote retry processing when PC interrupt has failed due to conditions such as a PC OFF (stop interrupt) condition in the ASCII Unit. If an IOWR (#CC00) interrupt is sent but not accepted, the PC Interrupt Fail Flag will be set to ON (1) for one scan in the next cycle. The operation of the PC Interrupt Fail Flag when an interrupt is not accepted is explained next.

**Timing Charts**

**Interrupt Execution for PC Interrupt Bit**

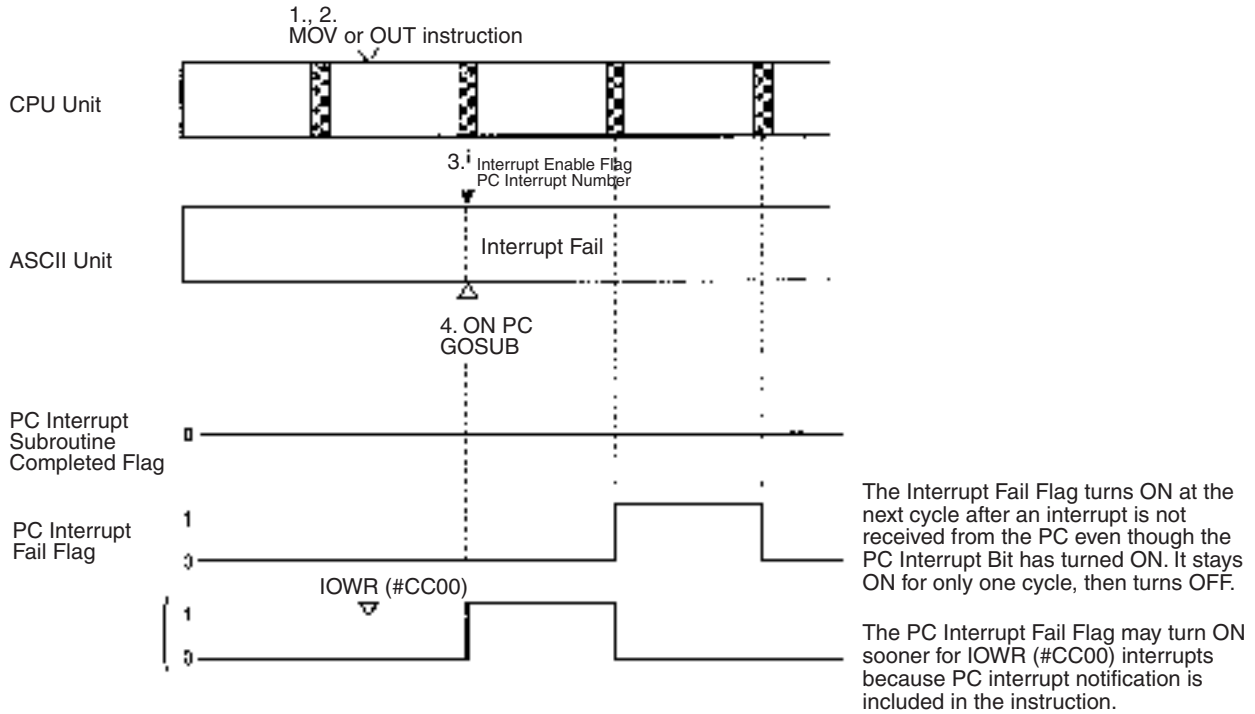


**Interrupt Execution for IOWR (#CC00) Interrupts**

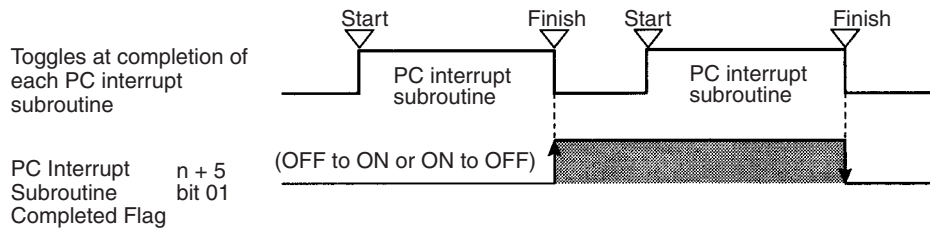


As shown above, interrupts using the IOWR (#CC00) instruction can be processed in one cycle less than interrupts using the PC Interrupt Bit.

**Interrupt Failure (Same for PC Interrupt Bit Interrupts and IOWR (#CC00) Interrupts)**



**PC Interrupt Subroutine Completed Flag Changes**

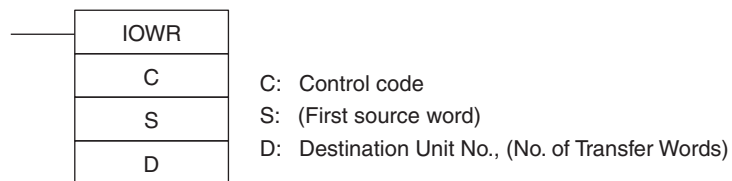


**Note** The time chart shown here indicates the flag operations which are different from the PC flag operations.

**6-5 IOWR/IORD Instruction Specifications**

The IOWR/IORD instructions are available only with the C200HX/HG/HE and CS1-series PCs. They cannot be used for ASCII Units on Slave Racks.

**6-5-1 IOWR Instruction (SPECIAL I/O UNIT WRITE)**



The parts inside parentheses change according to the control code.

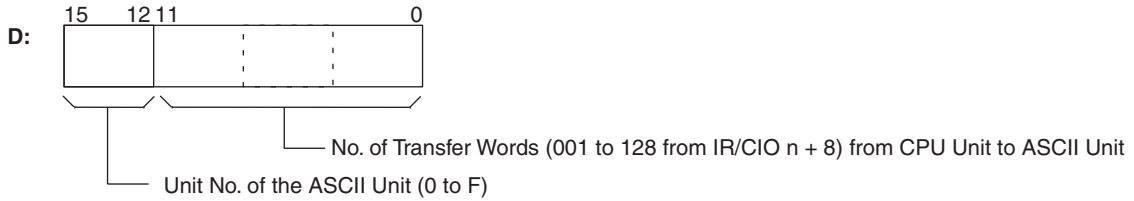
The IOWR instruction can be used with the ASCII Unit in the following four ways.

Writing to the ASCII Unit's Variables

	Value (4-digit hex.)	Details	Combination
C:	#FD00	Allocated IR/CIO area words IR/CIO n + 8 and IR/CIO n + 9 are transferred to the ASCII Unit's variable.	Must be used in combination with the ASCII Unit's PC QREAD@ command.

S: First source word in CPU Unit memory, from IR/CIO n+8 and IR/CIO n+9.

C200HX/HG/HE

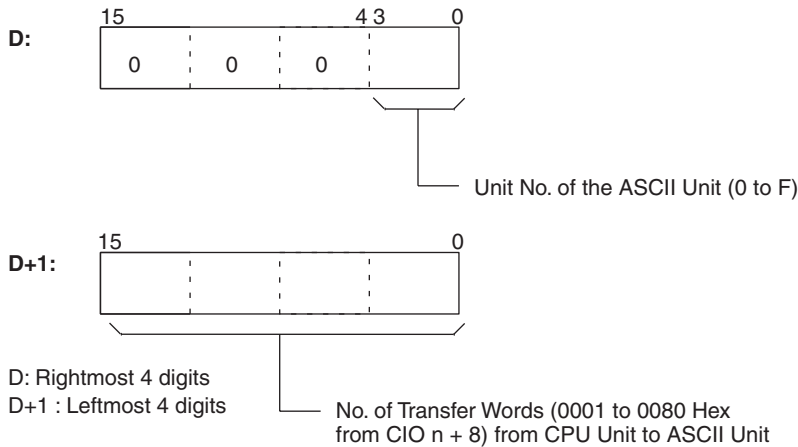


Data

Item	Detail
PC	C200HX/HG/HE
Operand	S
IR Area 1	IR 000 to IR 235
SR Area 1	SR 236 to SR 255
SR Area 2	SR 256 to SR 299
IR Area 2	IR 300 to IR 511
HR Area	HR 00 to HR 99
AR Area	AR 00 to AR 27
LR Area	LR 00 to LR 63
TC Area	TC 000 to TC 511
TR Area	---
DM Area	DM 0000 to DM 6655
EM Area	---
Indirect DM addresses	*DM 0000 to DM 6655
Constants	---



CS1-series PCs



Item	Detail
PC	CS1
Operand	S
CIO Area	CIO 0000 to CIO 6143
Work Area	W000 to W511
Holding Bit Area	H000 to H511
Auxiliary Bit Area	A000 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D00000 to D32767
EM Area without bank	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@D00000 to @D32767 @E00000 to @E32767 @En_00000 to @En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	#0000 to #FFFF (binary)
Data Registers	---
Directly addressing Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 -2048 to +2047 ,IR0 to ,IR15 DR0 to 15, IR0 to IR15 ,IR0 to ,IR15+(++) - (--) IR0 to IR15

Writing to the Shared Memory of the ASCII Unit

C:	Value (4-digit hex.)	Details	Combination
	#00□□	Specified word data of the specified memory area is transferred to the first word* + □□ of the ASCII Unit's shared memory IOWR Area. *The first word in IOWR Area is set in DM m + 6 allocated in the DM Setup Area.	Can be used in combination with the ASCII Unit's PC EGET command. Can also be used on its own.

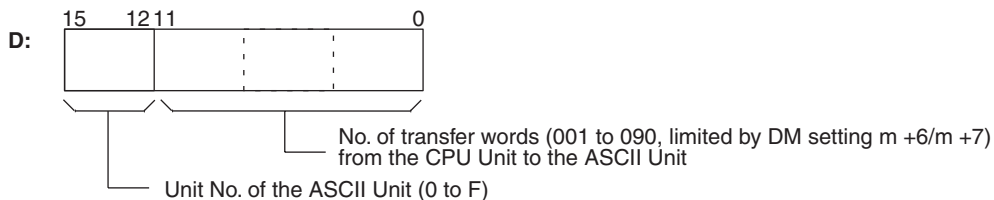
S: First source word in CPU Unit memory.

## Data

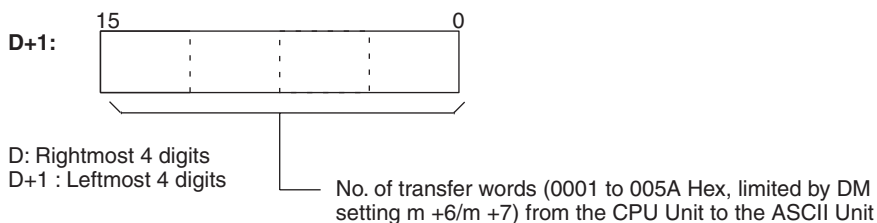
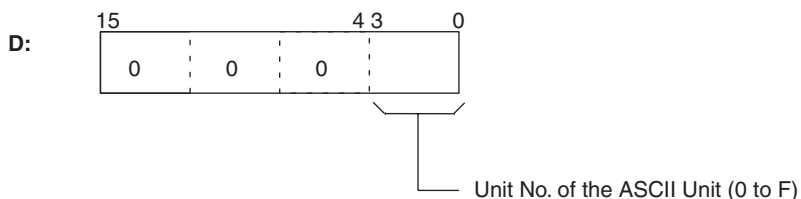
Item	Detail
PC	C200HX/HG/HE
Operand	S
IR Area 1	IR 000 to IR 235
SR Area 1	SR 236 to SR 255
SR Area 2	SR 256 to SR 299
IR Area 2	IR 300 to IR 511
HR Area	HR 00 to HR 99
AR Area	AR 00 to AR 27
LR Area	LR 00 to LR 63
TC Area	TC 000 to TC 511
TR Area	---
DM Area	DM 0000 to DM 6655
EM Area	---
Indirect DM addresses	*DM 0000 to DM 6655
Constants	---

Item	Detail
PC	CS1
Operand	S
CIO Area	CIO 0000 to CIO 6143
Work Area	W000 to W511
Holding Bit Area	H000 to H511
Auxiliary Bit Area	A000 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D00000 to D32767
EM Area without bank	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@D00000 to @D32767 @E00000 to @E32767 @En_00000 to @En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	#0000 to #FFFF (binary)
Data Registers	---
Directly addressing Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 -2048 to +2047 ,IR0 to ,IR15 DR0 to 15, IR0 to IR15 ,IR0 to ,IR15+(++) - (--) IR0 to IR15

**C200HX/HG/HE**



**CS1-series PCs**

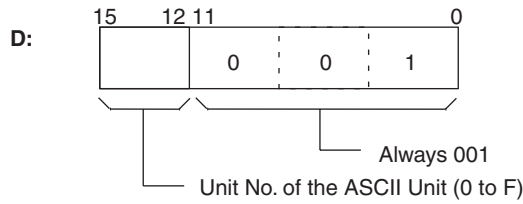


Interrupts to the ASCII Unit

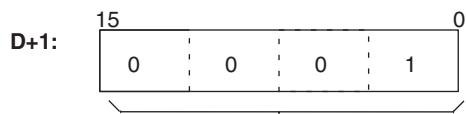
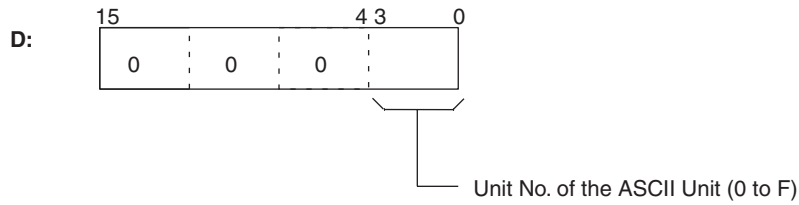
	Value (4-digit hex.)	Details	Combination
C:	#CC00	Sends an interrupt from the CPU Unit to the ASCII Unit.	Must be used in combination with the ASCII Unit's ON PC (Interrupt Number) command.

S: Interrupt Number (corresponds to the ASCII Unit's ON PC (Interrupt Number) command).

C200HX/HG/HE



CS1-series PCs



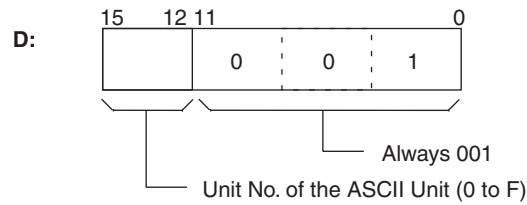
D: Rightmost 4 digits  
D+1 : Leftmost 4 digits

Clearing Errors in the ASCII Unit

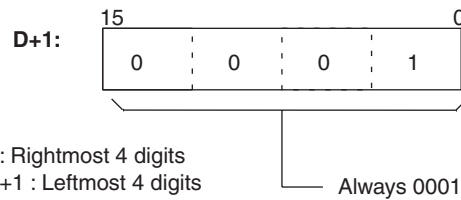
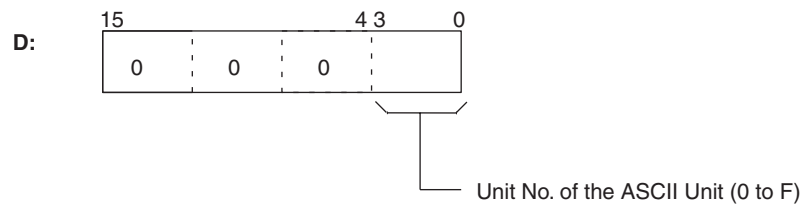
Value (4-digit hex.)	Details
C: #EC00	The current error status is cleared and the current errors are registered in the error history table as old errors.

S: Always #0000

C200HX/HG/HE



CS1-series PCs



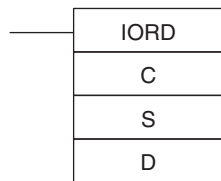
Flags

Value (see note)	ON	OFF
Instruction Execution Error Flag (ER) (SR 25503)	<ul style="list-style-type: none"> <li>The number of transfer words is not in BCD, or it is 0 words or greater than 128 words</li> <li>The indirectly addressed DM address is greater than 6656 or not BCD.</li> <li>The Unit No. of the ASCII Unit is outside the range 0 to F, or is on a SYSMAC BUS Slave Rack.</li> <li>The number of transfer words is not BCD.</li> <li>The instruction was not correctly completed.</li> </ul>	C, S, and D settings are correct.
Carry Flag (CY) (25504)	---	---
Greater Than Flag (GR) (25505)	---	---
Equals Flag (EQ) (25506)	Writing was correctly completed.	Writing was not correctly completed.

Value (see note)	ON	OFF
Less Than Flag (LE) (25507)	---	---
Overflow Flag (OF) (25404)	---	---
Underflow Flag (UF) (25405)	---	---
Negative Flag (N) (25402)	---	---

**Note** SR Area addresses for the C200HX/HG/HE, C200H, and C200HS are given in parentheses.

### 6-5-2 IORD Instruction (SPECIAL I/O UNIT READ)



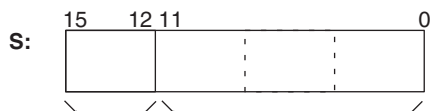
C: Control code  
 S: Source Unit No., No. of transfer words  
 D: First destination word

There are two methods of using the IORD instruction for the ASCII Unit as follows:

#### Reading from ASCII Unit Variables

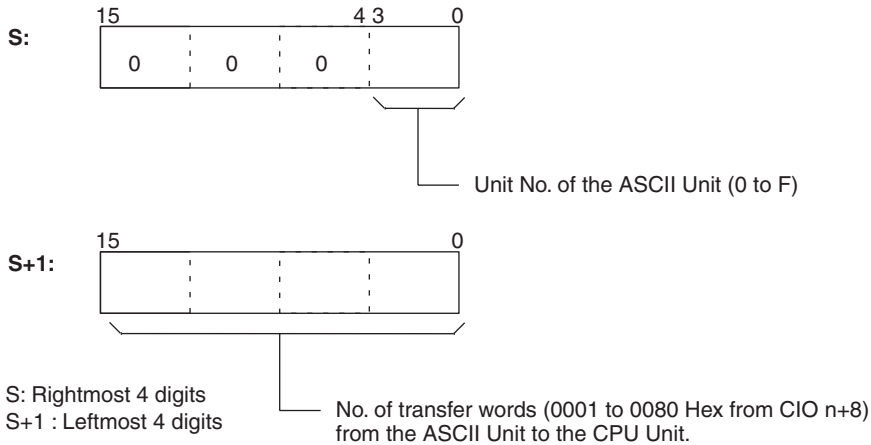
C:	Value (4-digit hex.)	Details	Combination
	#FD00	Transfers from the ASCII Unit's variable to the words specified in the allocated IR/CIO area words IR/CIO n + 8 and IR/CIO n + 9.	Must be used in combination with the ASCII Unit's PC QWRITE@ command.

#### C200HX/HG/HE



No. of transfer words (001 to 128 from IR/CIO n + 8) from the ASCII Unit to the CPU Unit.  
 Unit No. of the ASCII Unit (0 to F).

CS1-series PCs



D: First destination word in CPU Unit's I/O memory from CIO n+8 and CIO n+9.

Data

Item	Detail
PC	C200HX/HG/HE
Operand	D
IR Area 1	IR 000 to IR 235
SR Area 1	SR 236 to SR 252
SR Area 2	SR 256 to SR 299
IR Area 2	IR 300 to IR 511
HR Area	HR 00 to HR 99
AR Area	AR 00 to AR 27
LR Area	LR 00 to LR 63
TC Area	TC 000 to TC 511
TR Area	---
DM Area	DM 0000 to DM 6143
EM Area	---
Indirect DM addresses	*DM 0000 to DM 6655
Constants	---

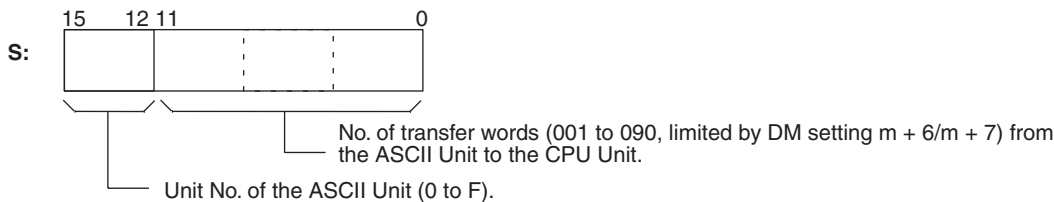
Item	Detail
PC	CS1
Operand	D
CIO Area	0000 to 6143
Work Area	W000 to W511
Holding Bit Area	H000 to H511
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D00000 to D32767
EM Area without bank	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@D00000 to @D32767 @E00000 to @E32767 @En_00000 to @En_32767 (n = 0 to C)

Item	Detail
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---
Data Registers	---
Directly addressing Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 -2048 to +2047 ,IR0 to ,IR15 DR0 to 15, IR0 to IR15 ,IR0 to ,IR15(++) - ( - ) IR0 to IR15

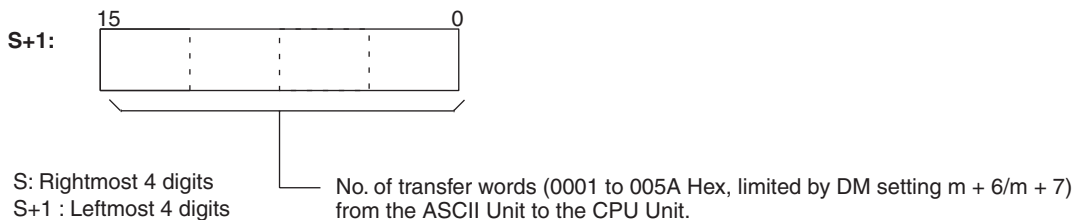
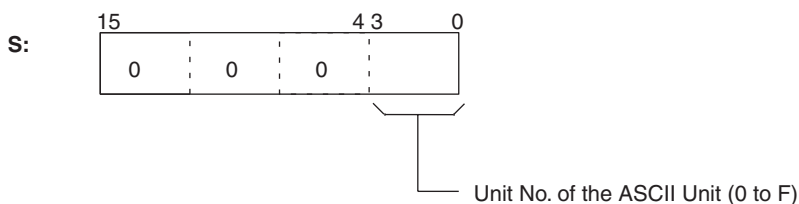
**Reading from the ASCII Unit's Shared Memory**

C:	Value (4-digit hex.)	Details	Combination
	#00□□	Data in the specified number of words is transferred from the first word* + □□ in the IORD Area of the ASCII Unit's shared memory to the specified words in the CPU Unit memory. *Set in DM m + 7 word allocated in the DM Setup area.	May be used in combination with the ASCII Unit's PC EPUT command. May also be used on its own.

**C200HX/HG/HE**



**CS1-series PCs**



D: First destination word in the CPU Unit's I/O memory.



## Data

Item	Detail
PC	C200HX/HG/HE
Operand	D
IR Area 1	IR 000 to IR 235
SR Area 1	SR 236 to SR 252
SR Area 2	SR 256 to SR 299
IR Area 2	IR 300 to IR 511
HR Area	HR 00 to HR 99
AR Area	AR 00 to AR 27
LR Area	LR 00 to LR 63
TC Area	TC 000 to TC 511
TR Area	---
DM Area	DM 0000 to DM 6143
EM Area	---
Indirect DM addresses	*DM 0000 to DM 6655
Constants	---

Item	Detail
PC	CS1
Operand	D
CIO Area	0000 to 6143
Work Area	W000 to W511
Holding Bit Area	H000 to H511
Auxiliary Bit Area	A448 to A959
Timer Area	T0000 to T4095
Counter Area	C0000 to C4095
DM Area	D00000 to D32767
EM Area without bank	E00000 to E32767
EM Area with bank	En_00000 to En_32767 (n = 0 to C)
Indirect DM/EM addresses in binary	@D00000 to @D32767 @E00000 to @E32767 @En_00000 to @En_32767 (n = 0 to C)
Indirect DM/EM addresses in BCD	*D00000 to *D32767 *E00000 to *E32767 *En_00000 to *En_32767 (n = 0 to C)
Constants	---
Data Registers	---
Directly addressing Index Registers	---
Indirect addressing using Index Registers	,IR0 to ,IR15 -2048 to +2047 ,IR0 to ,IR15 DR0 to 15, IR0 to IR15 ,IR0 to ,IR15+(++) - (- -) IR0 to IR15

## Flags

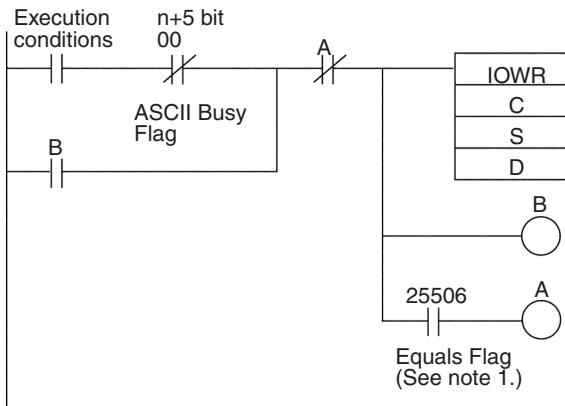
Value (see note)	ON	OFF
Instruction Execution Error Flag (ER) (25503)	<ul style="list-style-type: none"> <li>The number of transfer words is not in BCD, or it is 0 words or is greater than 128 words</li> <li>The indirectly addressed DM address is greater than 6656 or not BCD.</li> <li>The destination Unit No. is outside the range 0 to F, or is on a SYSMAC BUS Slave Rack.</li> <li>The number of transfer words is not BCD</li> </ul>	C., S., and D. settings are correct.
Carry Flag (CY) (25504)	---	---
Greater Than Flag (GR) (25505)	---	---
Equals Flag (EQ) (25506)	Reading was correctly completed.	Reading was not correctly completed.
Less Than Flag (LE) (25507)	---	---
Overflow Flag (OF) (25404)	---	---
Underflow Flag (UF) (25405)	---	---
Negative Flag (N) (25402)	---	---

**Note** SR Area addresses for the C200HX/HG/HE, C200H, and C200HS are given in parentheses.

### 6-5-3 Using the IOWR and IORD Instructions

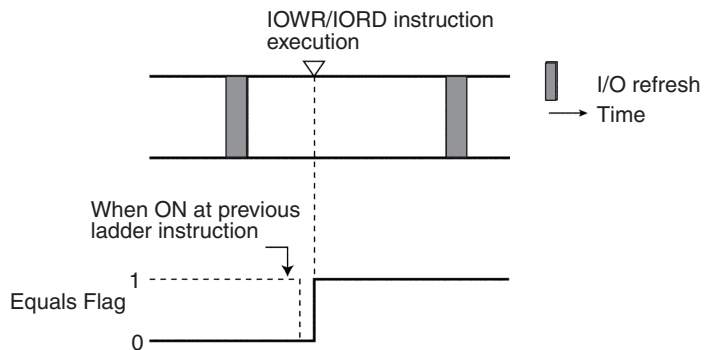
- 1,2,3...
- To allow retry processing if data transfer using the IOWR/IORD instructions fails, be sure to set a self-holding bit in the execution conditions and program the Equals Flag (SR 25506 for the C200HX/HG/HE) to release the bit when the Equals Flag is ON (1).
  - Program an NC condition for the ASCII Busy Flag (IR/CIO n+5 bit 00) as an execution condition so that the ASCII Unit does not execute an IOWR/IORD instruction during data transfer with the CPU Unit.

**Note** Only one IOWR/IORD instruction (control code #FD00) can be executed during a single cycle.

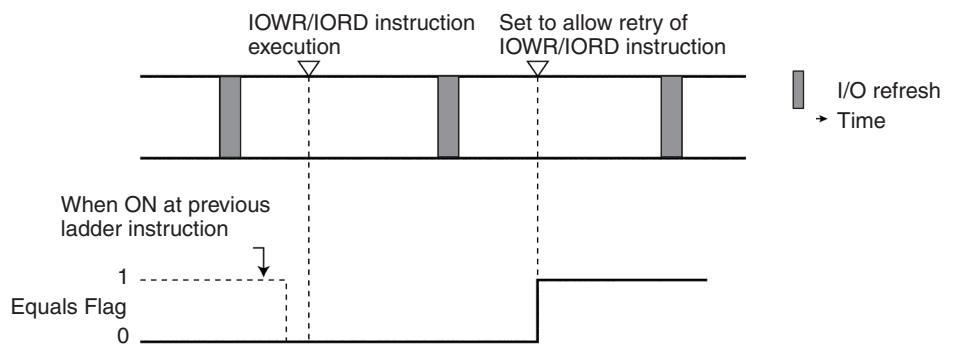


**Note** 1. The Equals Flag (SR 25506 for the C200HX/HG/HE) turns ON when processing to the ASCII Unit by IOWR/IORD instructions has finished normally. It remains OFF when processing to the ASCII Unit does not finish normally. After turning ON, it turns OFF at the execution of the next IOWR/IORD instruction. Since the Equals Flag will change at the execution of other instructions/commands as well (such as comparison instructions) be careful when writing programs.

**Normal Completion**



**Abnormal Completion**



2. The ASCII Busy Flag (IR/CIO n + 5 bit 00) turns ON when the ASCII Unit is exchanging data with the CPU Unit.

## 6-6 Restrictions for CS1-series PCs

There are five methods that can be used to transfer data between the CPU Unit and an ASCII Unit. Refer to 6-2 Selecting the Data Exchange Method for details. When an ASCII Unit is used with a CS1-series PC, however, there are addressing restrictions for some of these methods, as shown in the following table.

Method	CPU Unit	ASCII Unit	Area restrictions
1	MOV and other instructions	PC WRITE or PC READ	Yes
2	No programming required	PC WRITE@ or PC READ@	Yes
3	MOV and other instructions	PC GET or PC PUT	None
4	IOWR (#00□□) or IORD(#00□□)	PC EGET or PC EPUT	None
5	IOWR (#FD00) or IORD(#FD00)	PC QWRITE@ or PC QREAD@	None, but dummy addresses may need to be used for PC QWRITE@ or PC QREAD@

The specific restrictions are as follows:

- 1,2,3...
- Method 1: There are restrictions in the areas and words that can be addressed when using allocated words in memory to transfer data.
  - Method 2: There are restrictions in the areas and words that can be addressed when using PC WRITE@ or PC READ@ in the BASIC program in the ASCII Unit to transfer data.
  - Method 4, 5: IOWR and IORD can be used to transfer data to any memory area words in the CS1-series CPU Unit. Dummy addresses may have to be used in PC QWRITE@ or PC QREAD@.

### Method 1: PC WRITE or PC READ

Any words that can be addressed using the allocated words n+3 and n+4 can be used for data transfer. Possible addresses are shown in the following table.

Area	CS1 address (see note)	Designation method	
		Area (bits 12 to 15 in word n+3, 1 BCD digit)	First word address (bits 00 to 15 in word n+4, 4 BCD digits)
DM Area	D00000 to D06655	0	0000 to 6655
CIO Area	CIO 0000 to CIO 0511 (except CIO 0281)	1	0000 to 0511 (except 0281)
Data Link Area (in CIO Area)	CIO 1000 to CIO 1063	2	0000 to 0063
Holding Bit Area, part 1	H000 to H099	3	0000 to 0099
Holding Bit Area, part 2	H100 (except H 101)	4	0000 (except 0001)
	H102 to H127		0002 to 0027
EM Area, bank 0	E0_00000 to E0_06143	5	0000 to 6143
Timer Area	T0000 to T0511	6	0000 to 0511

**Note** Other addresses cannot be accessed directly in CS1-series CPU Unit memory. Use methods 3 to 5 for other addresses.

**Method 2: PC READ@ or PC WRITE@**

The addresses shown in the following table can be specified for PC READ@ or PC WRITE@.

Area	CS1 address (see note)	Designation method	
		Code	First word address
DM Area	D00000 to D06655	@D	0000 to 6655
CIO Area	CIO 0000 to CIO 0511 (except CIO 0281)	@R	0000 to 0511 (except 0281)
Data Link Area (in CIO Area)	CIO 1000 to CIO 1063	@L	0000 to 0063
Holding Bit Area, part 1	H000 to H099	@H	0000 to 0099
Holding Bit Area, part 2	H100 (except H 101)	@A	0000 (except 0001)
	H102 to H127		0002 to 0027
EM Area, bank 0	E0_00000 to E0_06143	@E	0000 to 6143
Timer Area	T0000 to T0511	@G	0000 to 0511

**Note** Other addresses cannot be accessed directly in CS1-series CPU Unit memory. Use methods 3 to 5 for other addresses.

**Method 3: PC GET or PC PUT**

There are no restrictions.

**Method 4: Reading/Writing Shared Memory in ASCII Unit Using PC EGET/PC EPUT or IOWR/IORD #00**

There are no restrictions.

**Note** The number of read/write words for IOWR/IORD must be specified in hexadecimal for the CS1 Series.

**Method 5: IOWR/IORD #FD00 and PC QREAD@/PC QWRITE@**

There are no restrictions in the words that can be transferred and the words specified for the IOWR/IORD #FD00 instruction will take priority. The addresses specified for the PC QREAD@/PC QWRITE@ commands will be stored in words n+8 and n+9 for monitoring by the CPU Unit. If addresses are specified for IOWR/IORD #FD00 that cannot be specified for PC QREAD@/PC QWRITE@, then use dummy addresses in PC QREAD@/PC QWRITE@.

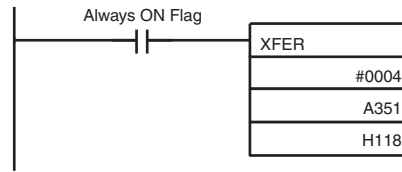
The CPU Unit will not monitor the specified words when dummy addresses are used.

**Note** The number of read/write words for IOWR/IORD must be specified in hexadecimal for the CS1 Series.

**6-6-1 Using the Sync Time Set Bit for CS1-series PCs**

The clock (RTC) in the ASCII Unit can be changed to the time set in the Calendar/Clock Area (AR 18 to AR 21) in C200HX/HG/HE CPU Units by turning ON the Sync Time Set Bit (word n, bit 05). With CS1-series PCs, however, the equivalent area (the Clock Data Area) is in words A351 to A354, so the clock will not be set correctly even if the Sync Time Set Bit is turned ON. To enable using the Sync Time Set Bit with CS1-series PCs, insert the following instructions into the program. (This requires that the ladder program is being exe-

cuted to use the Sync Time Set Bit, i.e., it cannot be used in PROGRAM mode.)



In the physical memory structure, words H118 to H121 in the CS1-series CPU Units correspond to words AR 18 to AR 21 in C200HX/HG/HE CPU Units. The above XFER instruction uses this information to enable using the Sync Time Set Bit with CS1-series PCs by transferring the time data in the normal CS1-series words to those used by the ASCII Unit (the time format is the same for both CS1-series and C200HX/HG/HE CPU Units).

# SECTION 7

## Editing BASIC Programs

This section describes the procedures for editing, saving, starting, and stopping BASIC programs. An overview of BASIC program configuration, language definitions, and the use of various BASIC commands, statements, and functions is provided as well.

7-1	Programming Procedure . . . . .	112
7-1-1	Editing BASIC Programs . . . . .	112
7-1-2	Saving BASIC Programs . . . . .	116
7-1-3	Transferring BASIC Programs Between ASCII Unit and Terminal. . . . .	117
7-2	Character Variable Space Allocations . . . . .	118
7-3	Starting/Stopping the BASIC Program . . . . .	120
7-3-1	Starting the Program . . . . .	120
7-3-2	Stopping the Program . . . . .	120
7-3-3	Stopping the Program at a Specified Line Number . . . . .	120
7-3-4	Continuing the Program after Stopping at a Line. . . . .	120
7-4	Program Configuration . . . . .	121
7-4-1	Command Execution Modes. . . . .	121
7-4-2	Programs, Lines, and Statements . . . . .	121
7-4-3	Character Set. . . . .	123
7-4-4	Constants. . . . .	123
7-4-5	Variables . . . . .	125
7-4-6	Variable Arrays. . . . .	125
7-4-7	Type Conversion . . . . .	126
7-4-8	Expressions. . . . .	127
7-5	List of BASIC Commands . . . . .	129
7-6	BASIC Commands . . . . .	139
7-7	User-defined BASIC Functions. . . . .	231
7-8	Debugging . . . . .	232
7-8-1	Overview. . . . .	232
7-8-2	Breakpoint Function (BRKPT). . . . .	232
7-8-3	Step Function (STEP). . . . .	233
7-8-4	Variable Monitor Function (WATCH) . . . . .	233
7-8-5	Trace Function (TRON and TRACE). . . . .	234

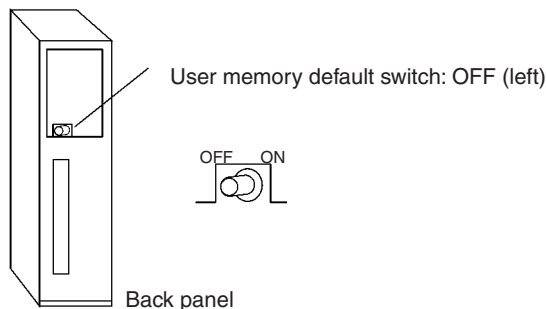
## 7-1 Programming Procedure

### 7-1-1 Editing BASIC Programs

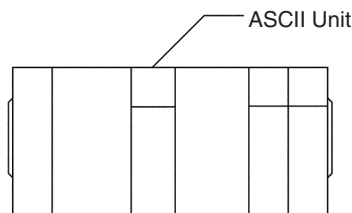
The following procedure shows how to edit BASIC programs using Windows 95 Hyperterminal communications software on a personal computer used as a terminal.

#### Connections and Default Settings

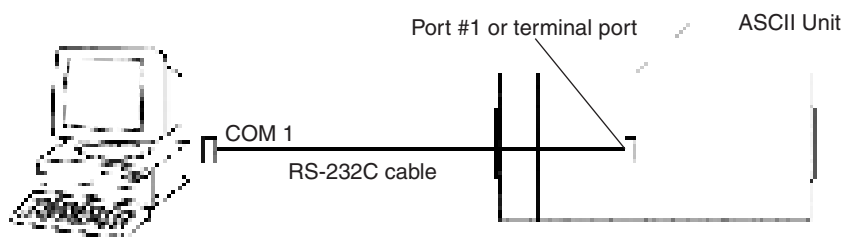
- 1,2,3... 1. Make sure that the user memory default switch is turned OFF (set to the left). If the default switch is turned ON (set to the right) and power is turned ON, the memory will return to the default settings (be formatted). This switch should always be turned OFF, except when processing errors.



2. Mount the ASCII Unit to the Backplane.



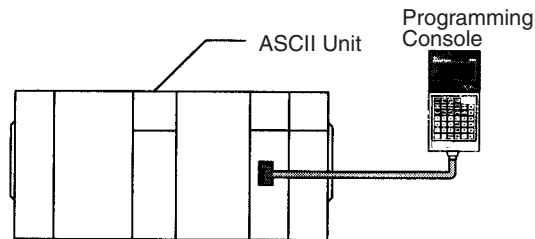
3. Turn OFF the power to the CPU Unit, and connect the RS-232C port (COM 1 in this example) to communications port #1 for the C200H-ASC11/ASC21 or the terminal port for the C200H-ASC31 using an RS-232C cable.



4. Set the switches on the front panel of the ASCII Unit as follows:
- Unit No. switch.
  - Set the START/STOP switch to START.
  - Terminating resistor switch, 2-wire/4-wire switch. (Only set when using communications port #2 of the C200H-ASC21.)
5. Turn ON the power to the CPU Unit.
6. Using a PC peripheral device (such as a Programming Console), perform the following operations.
- a) Create an I/O table
  - b) Set the DM Area of the CPU Unit (DM m to m + 7). (See 3-2 DM Area Allocations for details.)



- c) Once the settings are completed, turn ON the power, or using a PC peripheral device (such as Programming Console), toggle the ASCII Unit's Restart Bit from OFF to ON to OFF again.



**Connecting to the Terminal**

1,2,3...

1. In Windows 95, click **Start** and then point to **Programs**.
2. Point to **Accessories** and then click **Hyperterminal**.
3. Double click on the **Hyperterminal** icon.  
A message asking whether to install a modem will be displayed. Select "No."
4. The **Connection Settings** dialog box will be displayed. Enter your name.
5. The **Telephone No.** dialog box will be displayed. In the connection method field, select the personal computer's communications port number. Point to **Direct to COM 1**, and then click **OK**. The **COM 1 Properties** dialog box will be displayed.
6. In the **Port Settings** field, specify the following settings. (If the settings are already correct, then click **OK**.)  
 Baud rate: 9,600 bps  
 Data bits: 8  
 Parity: None  
 Stop bits: 2 bits  
 Flow control: Xon/Xoff

Note The default communications settings are 9,600 bps, 8-bit data, no parity, 2 stop bits, and no flow control for the ASCII Unit communications port #1 and the terminal port.

7. Point to **File** and then click **Properties**. The **[Your Name] Properties** dialog box will appear.
8. Select **VT100** in the **Emulation** field of the **Settings** sheet, then click **OK**.
9. Press the Ctrl+X Keys on the computer keyboard.  
When the following message is displayed at the terminal, connection is completed.

```

C200H-ASCII UNIT      Version 1.0x  1998/04/17
(c)Copyright OMRON Corporation 1998
READY
>
```

Note The above message is displayed only when the Ctrl+X Keys are pressed for the first time after the CPU Unit power is turned ON. If the terminal is disconnected and then reconnected, only the prompt ">" will be displayed.

10. If the connection has not been successfully made, check the following.
  - Cable connections (whether the computer is connected to the port correctly)

- That the communications parameters are the same for the ASCII Unit and terminal (DM m + 2 or m + 4 contain the terminal communications parameters)
- If the ERR indicator on the front panel of the ASCII Unit is lit, indicating there is an error in word n + 7 (bits 00 to 11: Error Code, bits 12 to 15: Error Type), correct the error, reset the CPU Unit power supply, and then press the Ctrl+X Keys.

Refer to SECTION 3 IR/CIO and DM Area Allocations for specific word allocations.

The ">" prompt may not always be displayed properly after stopping a BASIC program to open a port using the Start/Stop Switch. If the prompt is not displayed, enter Ctrl + X before continuing terminal operations.

## Inputting Programs

Perform the following steps to input a BASIC program directly into the ASCII Unit.

1,2,3...

1. Input **New** and press the Enter Key to clear the ASCII Unit's program.

```
>NEW_
```

Note The ASCII Unit has four program areas (No. 1 to 4) within the user memory area (RAM). Programs input using the default settings will be input in program area 1. To input a program into another area, switch to the desired program area executing the PGEN command as shown in the following example.

### Example:

```
>PGEN 2 .... (Program area 2 will become the active program area.)
```

2. Input **Auto** and press the Enter Key to automatically display line numbers.

```
>NEW
```

```
>AUTO_
```

3. Input the BASIC program.

### Example:

```
>NEW
```

```
>AUTO
```

```
10 A$=CHR$(67)
```

```
20 PRINT A$
```

```
30_
```

To quit AUTO, press the Ctrl+X Keys.

Note a) Execute the LIST command to display the BASIC program that has been written.

- b) Execute the PNAME command to name the program that has been written. See 7-6 BASIC Commands for details on the PNAME command.

**Example:** >PNAME "SAMPLE"

(In this example, **SAMPLE** will be the program name.)

- c) Execute the PINF command to display the information on the ASCII Unit's user memory area (RAM). See 7-6 BASIC Commands for details on the PINF command.

**Example:** >PINF ALL

```
>PINF ALL
```

```
P1      :      [R/W]  76 bytes
```

```
P2      :      [R/W]   0 bytes
```

```
P3      :      [R/W]   0 bytes
```

```
P4      :      [R/W]   0 bytes
```

LIB : 0 bytes  
I-CODE : 41 bytes

### Transferring the BASIC Program Using a Text Editor

Perform the following steps to transfer a BASIC program to the ASCII Unit after compiling it using a text editor:

- 1,2,3...**
1. Edit the program using a text editor such as the Note Pad, and save it in text format.
  2. Set the START/STOP switch to STOP and input **New** to clear the ASCII Unit's program.
  3. Input the LOAD command to put the ASCII Unit into file transmission wait state. The BASIC indicator (green) will flash quickly.
  4. Point to **Hyperterminal** and then **Transfer**, and click **Transmit text file** to transmit the saved file to the ASCII Unit. The file will be stored in the active program No. area and overwrite any existing files.
  5. Input Ctrl + C when the transfer has been completed. The prompt (>) will be displayed again.  
The T/R indicator will go out when the transfer has been completed.

### Precautions

Be sure to use the flow control or send delay setting when transmitting files to the ASCII Unit using **Transmit text file**. Not doing so may cause errors in file reception.

#### Example: Setting Flow Control

At the terminal: In the **Port settings** field, specify **Xon/Xoff** for the flow control.

At the ASCII Unit: Set the Xon/Xoff control to ON when inputting the LOAD command.

```
LOAD #1, "TERM: 9600, 8, N, 2, CTS_OFF, RTS_OFF, DSR_OFF, XN_ON"
```

When executing a program, the ASCII Unit reads the source program and creates intermediate code. Memory is consumed by creating intermediate code. When writing comparatively large programs, occasionally execute RUN and check the memory consumption by executing PINF ALL.

### Closing the Terminal

- 1,2,3...**
1. To save the connection information (session), point to **File** and then **Save as...**
  2. To close the hyperterminal, point to **File** and then **Close Hyperterminal**. A message will be displayed asking whether to disconnect the modem. Click "OK."

### Reconnection Procedures

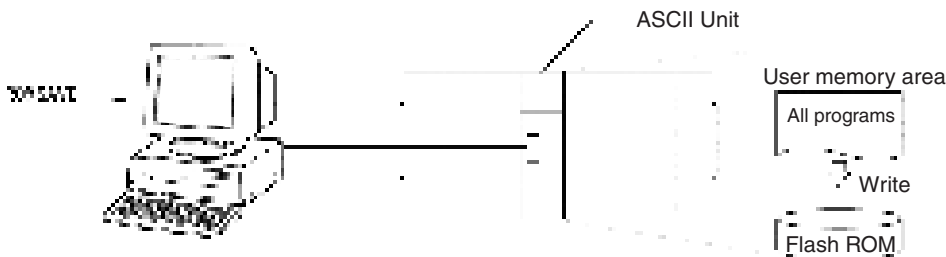
- 1,2,3...**
1. Double-click the icon (session file) that was created.  
**or** Point to **File** and then to **Open**. Select the session file.
  2. A message will be displayed asking whether to install a modem. Select "No."
  3. Enter the Ctrl+X Keys. If the previous message is displayed, the connection has been successfully made.

### 7-1-2 Saving BASIC Programs

#### Writing the BASIC Program to Flash ROM

The BASIC programs in the user memory (RAM) area can be written to the flash ROM in the ASCII Unit by entering the ROMSAVE command from the terminal.

>ROMSAVE



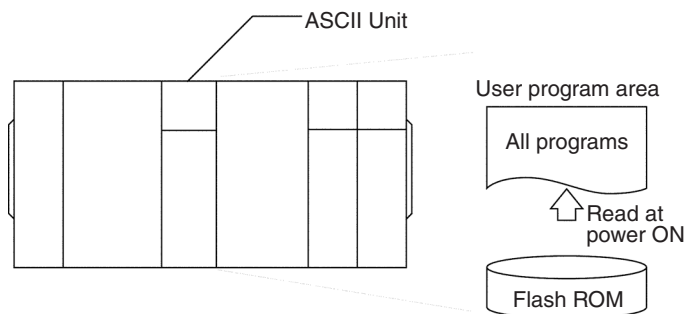
#### Reading the BASIC Program from Flash ROM

The BASIC programs in the flash ROM of the ASCII Unit can be read to the user memory (RAM) area of the ASCII Unit by entering the ROMLOAD command from the terminal.

>ROMLOAD

#### Automatically Reading the Program in Flash ROM at Power ON

The programs in the flash ROM can be automatically read to the user memory (RAM) area when power is turned ON or when the Unit is restarted, by setting bits 08 to 15 of word DM m to 5A<sub>hex</sub> in the DM Setup Area allocated to the Unit.



**⚠ WARNING** Never turn OFF the power supply of the PC while data is being written to flash memory by the ROMSAVE command until the completion display is shown at the terminal. By turning OFF the power supply, there is a possibility that the flash ROM will be electrically destroyed and that it will no longer be capable of storing data.

Do not turn OFF the power supply or restart the PC while data is being read from the flash ROM by the ROMLOAD command or automatically being transferred at startup. ROMLOAD command execution has been completed when the prompt appears on the terminal. Automatic transfer at startup has been completed when the command input prompt appears on the terminal. (Automatic transfer will be performed when power is turned ON or the Unit is restarted if the Startup Mode in the DM Area words allocated to the ASCII Unit is set for automatic transfer.)

Also, if the IOWR or IORD instruction is being used in the ladder program in the CPU Unit to exchange data with the ASCII Unit, always stop ladder program execution before writing to the flash ROM from the BASIC program using ROMSAVE.

**⚠ Caution** Do not turn OFF

## 7-1-3 Transferring BASIC Programs Between ASCII Unit and Terminal

### Transferring BASIC Programs to the Terminal

Use the following procedure to send BASIC programs from the user memory (RAM) of the ASCII Unit to the terminal.

#### Terminal: VT100 Mode (Windows: Hyperterminal)

- 1,2,3... 1. Connect the terminals and input "SAVE #1" or "SAVE #3" (for C200H-ASC31 only) and then press the Enter Key. (See note.)

**Example:**

SAVE #1, "TERM: 9600, 8, n, 2, CTS\_OFF, RTS\_OFF, DSR\_OFF, XN\_ON"

2. Enable the text data receiving mode at the terminal. (See note.)  
3. When the terminal indicates that it is waiting to receive data, press the Ctrl+C Keys. When the ASCII Unit begins to transfer data, the T/R indicator will flash. (See note.)  
4. Once the ASCII Unit has finished transferring all the data, end the text data receiving mode at the terminal.  
5. Press the Ctrl+C Keys to display the prompt again.

**Note** If a password has been set in the BASIC program with the PWORD command, use the following procedure.

- 1,2,3... 1. Input "SAVE #1" or "SAVE #3" (for C200H-ASC31 only) and press the Enter Key. "PASSWORD" will be displayed at the ASCII Unit. Input the password.  
2. Enable the text data receiving mode at the terminal.  
3. When the terminal is ready to receive data, press the Ctrl+C Keys. Steps 4 and 5 are the same as for the above procedure.

#### Terminal: FIT10 Mode

- 1,2,3... 1. Set the START/STOP switch to STOP.  
2. Input "SAVE #1" or "SAVE #3" (for C200H-ASC31 only) and press the Enter Key at the terminal. The ASCII Unit will be in a transfer wait state. This is indicated by the flashing BASIC indicator. (See note.)

**Example:**

SAVE #1, "TERM: 9600, 8, n, 2, CTS\_OFF, RTS\_OFF, DSR\_OFF, XN\_ON"

3. Enable the text data receiving mode at the terminal.  
4. Set the START/STOP switch to START. When the ASCII Unit begins transferring data, the T/R indicator will flash.  
5. Once the ASCII Unit has finished transferring all the data, set the START/STOP switch to STOP, and the prompt will appear again.  
6. End the text data receiving mode at the terminal.

**Note** If a password has been set in the BASIC program, "PASSWORD" will be displayed at the ASCII Unit. Input the password and press the Enter Key. Once the ASCII Unit is in a transfer wait state, continue with the same procedure.

**Writing Data from the Terminal to the ASCII Unit**

To write BASIC programs from the terminal to the user memory (RAM) of the ASCII Unit, execute the LOAD command, and transfer the text file once the ASCII Unit is in a transfer wait state.

**BASIC Program Transfer Commands**

Operation		Command	Example
Transferring BASIC programs between the terminal and ASCII Unit	Terminal-to-ASCII Unit user memory (RAM)	LOAD	>LOAD #1
	ASCII Unit user memory (RAM)-to-terminal	SAVE	>SAVE #1
Transferring BASIC programs between flash ROM and the ASCII Unit	Flash ROM-to-ASCII Unit user memory (RAM)	ROMLOAD	>ROMLOAD
	ASCII Unit user memory (RAM)-to-flash ROM	ROMSAVE	>ROMSAVE

**7-2 Character Variable Space Allocations**

When executing the BASIC program in a C200H-ASC11/ASC21/ASC31 ASCII Unit, the method can be selected for allocating space for character variables to user memory. The following two methods are available.

**Note** For numeric variables, the size of the variable area is fixed based on the type, and is therefore allocated in user memory using the static (fixed) method outlined below.

**Static (Fixed) Method**

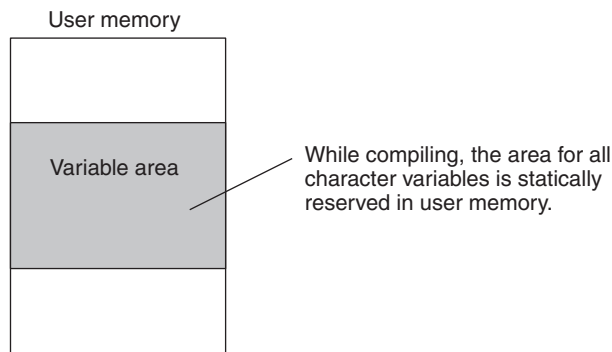
When compiling the BASIC program, space for all character variables in the BASIC program will be statically reserved in user memory.

**Setting**

At the beginning of the BASIC program, the maximum character variable content to be used in the BASIC program is specified using the OPTION LENGTH command. The maximum length can be specified from 1 to 255 characters.

**Content**

The space for character variables is reserved statically and therefore free memory cannot be generated as with the dynamic method. Even if not actually used, the space allocated for all variables will still be used.



Restrictions with this method are outlined below:

- The maximum length specified for the character variable content using the OPTION LENGTH command cannot be exceeded. If the maximum is exceeded, the amount of data over the limit will not be stored in the string variable “truncated.”
- For the static method, once the variable type is specified under a particular variable name, that variable cannot be changed to a different type under the same name during execution of the program.
- For the static method, the DEF (variable type declaration) command cannot be executed more than once for the same variable name during the program.
- For the static method, at compile time, the memory size for the array variable is allocated statically, based on the number of subscripts. Therefore, only a fixed number of DIM command subscripts can be specified. Numeric expressions or numeric variables cannot be specified. (For the dynamic method, it is possible to alter the number of subscripts as required by using numeric expressions or numeric variables during the program to specify the number of subscripts.)

**Dynamic (Free) Method**

While executing the BASIC program, the space for each character variable in the BASIC program is reserved dynamically in user memory.

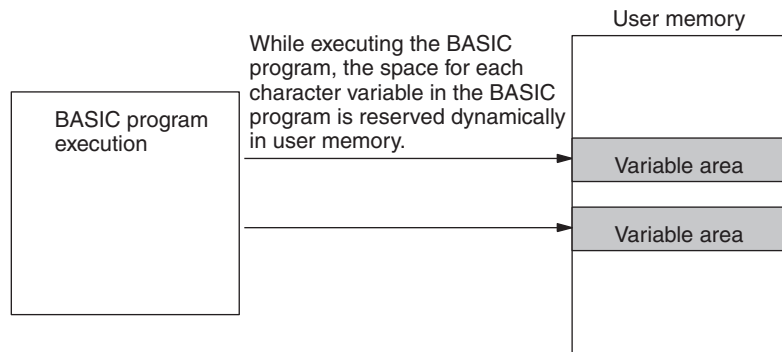
**Setting**

The OPTION LENGTH command is not used.

**Content**

Only the required amount of user memory is used. If the size of the variable becomes larger during the execution of the BASIC program, the space that had been reserved will be released once and then reserved again. (The released area will become free area of that size.)

If the allocation cannot be made when trying to reserve the space again, due to insufficient user memory capacity, an error will be generated and the program will be stopped. In such cases, it is necessary to change the program and the way in which the character variables are used, or use the ON ERROR command to free some memory.



**Note** The ASCII Unit is operated differently, depending on which of the two above methods is used. The differences are as follows:

- The compile time is longer for the static method than for the dynamic method.
- Conversely, when substituting character variables in the dynamic method, user memory release or allocation processing may be generated. Therefore, the execution time for commands is sometimes longer than for the static method.

- Place the PWORD command at the beginning of a statement if it is used with an OPTION LENGTH command.

## 7-3 Starting/Stopping the BASIC Program

### 7-3-1 Starting the Program

#### Selecting the Program Number

When transferring DM data allocated in the CPU Unit to the ASCII Unit at power up or start up, the start program number is specified in bits 00 to 07 of DM m in the DM Setup Area allocated to the Unit. Refer to SECTION 3 IR/CIO and DM Area Allocations for specific word allocations.

00: No. 1	01: No. 1
02: No. 2	03: No. 3
04: No. 4	

Use the PGEN command to change the program number from the terminal.

**Example:** >PGEN 2 (The program area will switch to number 2.)

#### Starting the Program

- 1,2,3...
1. If starting from the terminal, make sure that the START/STOP switch on the front panel of the Unit is in START position, and input the RUN command from the terminal.
  2. If using the START/STOP switch on the front panel of the Unit, toggle the START/STOP switch from STOP to START.
  3. If starting automatically when power is turned ON or when the Unit is re-started, set the START/STOP switch on the front of the Unit to START, and set bits 08 to 11 of DM m + 1 to 5A<sub>hex</sub> (automatic start) in the DM Setup Area allocated to the Unit. Refer to SECTION 3 IR/CIO and DM Area Allocations for specific word allocations.

### 7-3-2 Stopping the Program

Press the Ctrl+X Keys to stop the program from the terminal.

If the port connected to the terminal has been opened as a COMU device or an output device, the program cannot be stopped by pressing the Ctrl+X Keys.

or Turn OFF the START/STOP switch on the front panel of the Unit.

### 7-3-3 Stopping the Program at a Specified Line Number

Perform one of the following steps to stop the program from the terminal.

- Input RUN TO and the line number, and run the program.
- Input the BRKPT command, set the break point, and run the program.
- Input the STEP command and only run the specified lines.

or In the program, set a STOP command at the specified line number before running the program.

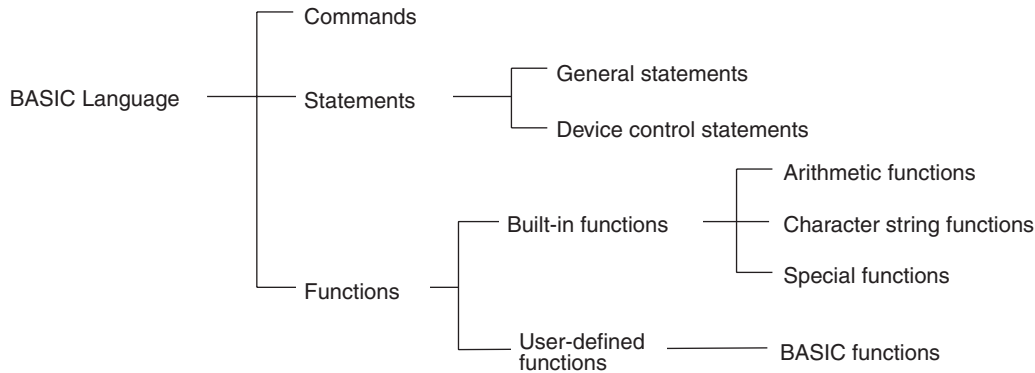
### 7-3-4 Continuing the Program after Stopping at a Line

- 1,2,3...
1. Input the CONT command from the terminal.
  2. Input the STEP command for the desired number of lines.



## 7-4 Program Configuration

A BASIC program consists of commands, statements, and functions.



**Basic Statements** designate and control the flow of programs and are generally used in program lines within a program. Statements are executed by the RUN command and can be executed directly from a terminal without line numbers.

**Basic Commands** are usually entered from the command line and control operations external to the program, such as printing and listing. Commands are executed when the ASCII Unit is in Command Mode. They cannot be used in the program and executed with the RUN command.

**Functions** are self-contained programs which accept one or more arguments, perform predefined calculations, and return a result(s). There are predefined BASIC functions for arithmetic and string operations as well as user-defined functions. Functions cannot be used on their own, but must be combined with statements.

### 7-4-1 Command Execution Modes

There are two modes for executing commands.

#### Command Mode

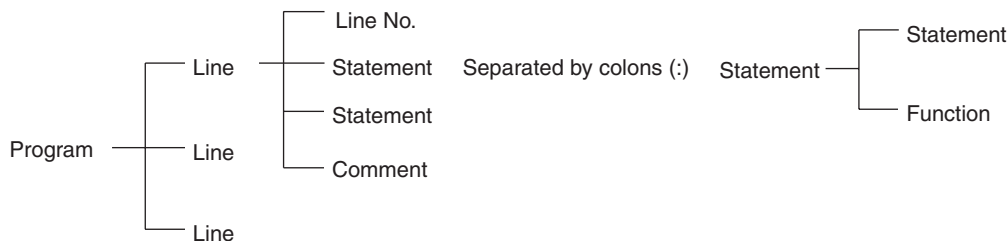
By entering a command and pressing the Enter Key when ">" is displayed on the terminal screen, the command will be immediately executed. The command input standby condition is called Command Mode.

#### Program Mode

A line is created by inputting a BASIC command with a line number at the beginning and pressing the Enter Key. A program consists of a collection of these lines. The execution of commands using these programs is called Program Mode.

### 7-4-2 Programs, Lines, and Statements

A BASIC program consists of a collection of lines. A line is made up of one or more statements, and a statement can contain a function.



#### Line Numbers

- Every BASIC program line begins with a line number. (When there is no line number, the program is executed directly.)

- All inputs up until the Enter Key is pressed are stored in the user memory as a single line, in the order of line numbers. Execution is carried out in the order of line numbers. Line numbers are also used as references for branching destinations such as GOSUB and GOTO, and arguments in LIST and DEL.
- Line numbers must be between 0 and 65535. Each line is written in ascending order of magnitude of the line numbers.
- A line consists of up to 255 characters (including the line number and spaces).



**Note** A period (.) may be used in DELETE, LIST, and EDIT commands to refer to the current line number of a program (the most recently entered or displayed line number).

Examples of the use of the period: LIST.  
EDIT.  
DEL 100-.

**Labels**

Labels are used in place of line numbers in GOTO and GOSUB commands. At the point being labeled, alphabetic characters must follow an asterisk (\*) as the line's first statement. The maximum length of a label is 16 characters, including the asterisk. Text strings that begin with reserved words cannot be used. When characters continue after a label in a line, they are separated by a colon (:). When two identical label names exist, the label in the line with the lowest line number is referred to.

**Example**

```
10 ON COM 1 GOSUB *ABC
                        |
                        | Label
100 *ABC
     |
     | Label
```

**Note** If the line number of the branch destination is specified, the line number will also have to be corrected when editing the program. By using labels, it is not necessary to correct the line number, making program revisions quicker and more effective. If text strings starting with reserved words are used as labels, a SYNTAX ERROR will occur.

**Statements**

- Statements can occur alone or they can contain functions.
- If there is more than one statement on the same line, separate the statements with a colon (:).

**Example:**

```
10 FOR L=1 TO 100: J=L*1: PRINT J: NEXT L
Line No. Statement 1 Statement 2 Statement 3 Statement 4
```

**Comments**

Input comments after single quotations (').

**Note** Comments cannot be used in lines containing the DATA command.

### 7-4-3 Character Set

The BASIC character set comprises alphabetical characters, numeric characters, and special characters.

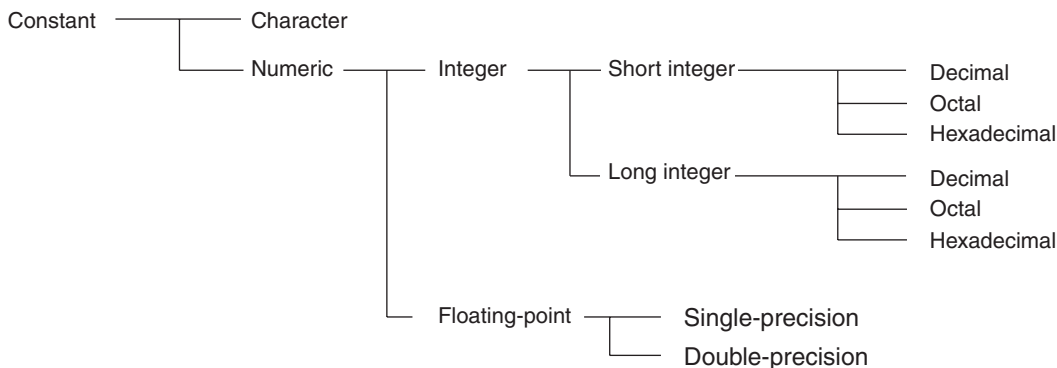
The alphabetic characters in BASIC are the uppercase and lowercase letters of the alphabet. The numeric characters in BASIC are the digits 0 through 9.

The following special characters are recognized by BASIC:

SP (space) ! " # \$ % & ' ( ) \* + , - . / : ; < = > ? [ \ ] ^ \_

### 7-4-4 Constants

The following can be used as constants.



#### Character Constants

A character constant is a character string enclosed by double quotation marks ("). It can be up to 255 characters long. If it contains no characters, it is called an "empty character string" or the null string.

**Example:** "CF-BASIC"

#### Integer Constants

There are two types of integers:

- Short integers (16-bit)
- Long integers (32-bit)

#### Decimal Constants

Whole numbers between -32,768 and 32,767 can be used for short integers, and whole numbers between -2,147,483,648 and 2,147,483,647 can be used for long integers. An optional percent sign (%) for an integer, and an "and" sign (&) for a long integer can be added to specifically indicate an integer constant. Integer constants do not have decimal points.

**Example:** 1234 -1234 12% (integer) 12345678& (long integer)

#### Octal Constants

Octal numbers 0 to 7 beginning with the prefix "&" and between &0 and &177777 can be used for integers. For long integers, between &0 and &3777777777 can be used.

**Example:&** 0127 &7777

#### Hexadecimal Constants

Hexadecimal numbers 0 to F (0 to 9, A to F) beginning with the prefix "&H" and between &H0000 and &HFFFF can be used for integers. Between &H0 and &HFFFFFFFF can be used for long integers.

**Example:** &H5E &HBF4

**Floating Point Constants**

These are positive or negative numbers that are expressed in exponential format. E (single-precision) or D (double-precision) is entered after the mantissa, and then the exponent is entered.

**Example:** For the value 12,300,000, the following equations apply.

$$= 1.23 \times 10,000,000$$

$$= 1.23 \times 10^7$$

$$= 1.23E + 7 \text{ (single-precision)}$$

$$= 1.23D + 7 \text{ (double-precision)}$$

1.23 is the mantissa and E + 7 or D + 7 is the exponent.

**Single Precision**

This type of constant is stored with seven-digit precision and is output as a six-digit constant with the seventh digit rounded off within the range of  $\pm 1.2 \times 10^{-38}$  to  $\pm 3.4 \times 10^{-38}$ . It is represented by one of the following methods:

- As a number with six or fewer digits: 1234.5
- As a number in exponential form using E:  $-1.2E+3$
- As a number with the character "!" at the end: 2.34!

**Double Precision**

This type of constant is stored with 16-digit precision and is output with up to 15 digits or less with the 16th digit rounded off within the range of  $\pm 2.23 \times 10^{-308}$  to  $\pm 1.79 \times 10^{-308}$ . It is represented by one of the following methods:

- As a number with seven or more valid digits: 1.23456789
- As a number in exponential form using D:  $-1.2D-3$
- As a number with the character "#" at the end: 2.34#

**Note** Precautions on Real Numbers:

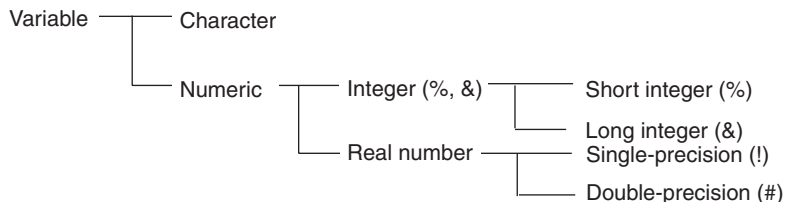
A real number expressed in the floating point format is advantageous in allowing a wide range of numbers to be handled. However, not all numbers are necessarily expressed correctly due to the nature of the binary expression and as a result, some of them are expressed in approximate values. Therefore, most of the values and calculation results stored in memory contain some amount of error. Although this error is normally so small that it cannot cause any problem, it may be interpreted as a large error depending on the application.

For example, when the number is truncated below the decimal point after multiplying "0.1" by 10, the result of "1" will be expected. However, "0.1" is stored in the memory as an approximate value of "0.0999..." because it cannot be expressed in the binary format. When this value of "0.0999..." is multiplied by 10, the result will be "0.999..." When the number is truncated below the decimal point, the result will be "0".

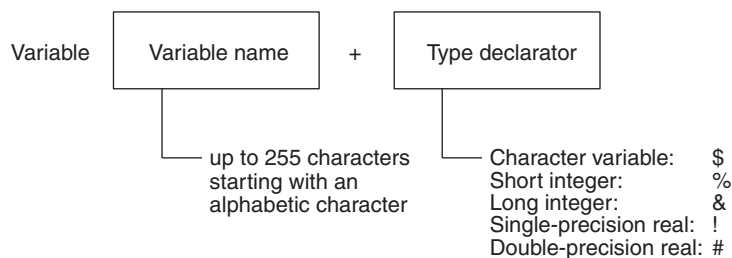
The solutions to this problem vary depending on the application. When reviewing design proposals, check if a desired operation can be achieved by using the integer type or if desired integers can be obtained by rounding the calculation results.

### 7-4-5 Variables

Variables are names used to represent values that are used in a BASIC program. The value of a variable may be assigned as the result of calculations or explicitly by the programmer with an assignment statement. If no value is assigned to a numeric variable, it is assumed to be zero. If no value is assigned to a character variable, it is assumed to be the null string.



Variables are made up of the variable name and the type declarator.



#### Variable Name

A variable may be up to 255 alphanumeric characters long. All characters are valid. No variable can start with “FN” or a BASIC command name.

If a parameter begins with a reserved word, a SYNTAX ERROR will occur. TOTAL and ABSOL, for example, cannot be used because they include reserved words TO and ABS.

#### Type Declarator

The variable TYPE can be declared. This is done using a type declarator that is placed after the variable name. Even if two variables have the same name, they will be treated differently if they are declared as different types of variables.

%	Short Integer	Uses 2 bytes per variable.	Example: A%
&	Long integer	Uses 4 bytes per variable	Example: A&
!	Single-precision real	Uses 4 bytes per variable.	Example: A or A!
#	Double-precision real	Uses 8 bytes per variable.	Example: A#
\$	Character	Uses up to 255 characters.	Example: A\$

**Note** There is a second way to declare variable types. The BASIC statements DEFINT, DEFLNG, DEFSTR, DEFSNG, and DEFDBL may be used to declare the types for certain variable names, starting with a certain letter. By default, all variables are declared as single-precision.

### 7-4-6 Variable Arrays

An array is a group of values of the same TYPE (either numeric or alphabetic) that is stored and referenced as a unit by the same variable name. Each element in an array has a unique position and is referenced by the name of the array subscripted with an integer or integer expression.

There can be many dimensions to an array. The most common types are one-, two-, and three-dimensional arrays. The number of dimensions of an

array and the length of its name are dictated by the amount of available memory. An array has one subscript for each dimension in the array.

For example, T(4) would reference the fourth element in the one-dimensional array T. R(2,3) would reference the value located in the second row and third column of the two-dimensional array R.

The maximum number of dimensions of an array is 255. The maximum number of elements per dimension is 32767. The array size and number of dimensions must be declared with the DIM statement. The subscript value zero is the position of the first element in an array. All elements of an array must be of the same TYPE. When an array is declared, numeric arrays are initialized as zero, and alphabetic arrays are initialized as null strings. When a variable array is used without being declared, it will be declared as a one-dimensional array with elements 0 to 10, or 1 to 10 depending on the option base.

**Note** In this example, the option base = 0

**Example:**

DIM A\$(3) This array is the one-dimensional, alphabetic A\$ variable array. The following 4 variables can be used.

A\$(0)            A\$(1)            A\$(2)            A\$(3)

DIM B(2,3) This array is the two-dimensional, numerical B variable array. The following 12 variables can be used.

B(0,0)	B(1,0)	B(2,0)
B(0,1)	B(1,1)	B(2,1)
B(0,2)	B(1,2)	B(2,2)
B(0,3)	B(1,3)	B(2,3)

### 7-4-7 Type Conversion

When necessary, BASIC will automatically convert a numeric constant from one TYPE to another. The following rules apply.

- 1,2,3...**
1. If the numeric data on the right side of an assignment statement differs from the type of data on the left side, the right side is converted to match the left. Character data, however, cannot be converted to numerical data, or vice versa without using functions (VAL, STR\$)

**Example:**

A = 12.3: if A is an integer, then "12" is assigned to A.

2. Double-precision data is converted to single-precision data when assigned to a single-precision variable.

**Example:**

If "A" is a single-precision variable and the statement:

LET A = 12.3456789# occurs in a program, then 12.3456789# will be converted to a single-precision number and then assigned to "A." (A value rounded to 6 digits will be displayed.)

3. When an arithmetic operation is performed using both single-precision and double-precision values, the single-precision value is converted to double-precision first, and then the operation is performed. Therefore, the result is a double-precision value.

**Example:**

10#/3 (double-precision)

4. In logic operations, all numeric data is first converted into long-integer data unless both inputs are short integers. If any value cannot be converted into

a long integer within the range of -2147483648 to 2147483647, an overflow error will occur.

**Example:**

LET A = NOT 12.34, -13 is assigned as A.

- When a real number is converted into an integer, the value is rounded.

**Example:**

A = 12.3: "12" is assigned to A.

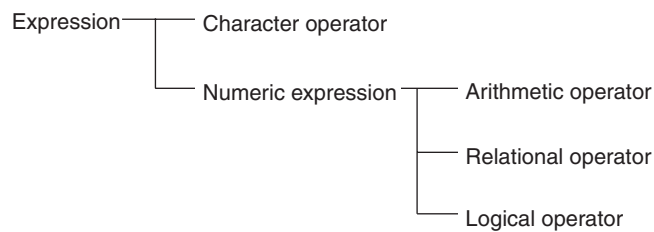
B = 12.6: "13" is assigned to B.

### 7-4-8 Expressions

Expressions refer to constants, variables, and functions that have been combined by operators. Numeric values, variables, or characters alone can also form expressions. There are four types of expressions.

- Arithmetic
- Relational
- Logical
- Character

Of these, the first three produce numeric values as a result and are thus called "numeric expressions." The last type is called a "character expression."



#### Character Operators

A character expression is made up of character constants and variables that are linked with the character operator "+." Instead of adding characters together, the "+" operator links the characters together to form one character string.

Input: A\$ = "CF"  
 B\$ = "BASIC"  
 PRINT A\$ + "-" + B\$

Output: CF-BASIC" is displayed.

#### Arithmetic Operators

An arithmetic expression is made up of constants, variables, and functions combined using arithmetic operators. A list of valid arithmetic operators is shown in the following table.

Arithmetic operator	Example	Operation
+	A + B	Addition
-	A - B, -A	Subtraction or negation
*	A * B	Multiplication
/	A / B	Real number division
\	A \ B	Integer division
MOD	A MOD B	Remainder after integer division
^	A ^ B	Exponentiation

If A or B is a real number in an expression using the \ or MOD operator, the decimal part is first rounded up to convert the real number into an integer, and then the operation is performed.

**Relational Operators**

Relational operators compare two values. The output is “-1” (&HFFFF) if the two values are equal and “0” if they are not.

Relational operator	Example	Operation
=	A = B	Equal
<>, ><	A <> B	Not equal
<	A < B	Less than
>	A > B	Greater than
<=, =<	A <= B	Less than or equal to
>=, =>	A >= B	Greater than or equal to

**Logical Operators**

Logical operators perform tests on multiple relations, bit manipulations, or Boolean operations. The logical operator returns a bitwise result that is either “true” (not 0) or “false” (0). In an expression, logical operations are performed after arithmetic and relational operations. The outcome of a logical operation is determined as shown in the following table. The operators are listed in the order of precedence.

Logical Operator	Description, Example, and Result	
NOT (negation)	A	NOT A
	1	0
	0	1
AND (logical product)	A B	A AND B
	1 1	1
	1 0	0
	0 1	0
	0 0	0
OR (logical sum)	A B	A OR B
	1 1	1
	1 0	1
	0 1	1
	0 0	0
XOR (exclusive-OR)	A B	A XOR B
	1 1	0
	1 0	1
	0 1	1
	0 0	0
EQV (equivalence)	A B	A EQV B
	1 1	1
	1 0	0
	0 1	0
	0 0	1
IMP (implication)	A B	A IMP B
	1 1	1
	1 0	0
	0 1	1
	0 0	1



## 7-5 List of BASIC Commands

The following table lists all BASIC commands alphabetically and gives an outline for each. See 7-6 BASIC Commands for further details on each command.

Square brackets ([ and ]) in the syntax column indicate items that can be omitted. Normal brackets ({ and }) indicate items that can be repeated. (See 7-6 BASIC Commands for further details on syntax.)

Command	Syntax	Details	Command	Statement	Function	Page
@IF	@IF <numeric expression> THEN <program> ELSE <program> ENDIF	Selects either THEN and the following characters, or ELSE and the following characters, based on the results of a numeric expression.		General		138
ABS	ABS (numeric expression)	Determines the absolute value of the value given by the argument.			Arithmetic operation	139
ACOS	ACOS (numeric expression)	Determines the arc cosine of the value given by the argument.			Arithmetic operation	139
ALARM ON/ OFF/STOP	ALARM ON or ALARM OFF or ALARM STOP	Enables, disables, or stops a one-shot timer (ON ALARM) interrupt.		General		140
ASC	ASC (character expression)	Determines the ASCII code of the first character of the character string given by the argument.			Character string	140
ASIN	ASIN (numeric expression)	Determines the arc sine of the value given by the argument.			Arithmetic operation	141
ATN	ATN (numeric expression)	Determines the arc tangent of the value given by the argument.			Arithmetic operation	141
AUTO	AUTO [line number][increment]	Automatically generates numbers at the beginning of lines (if not specified, increment = 10)	Command			141
BITOFF/BITON	BITOFF <integer variable>, <bit position> BITON <numeric variable>, <bit position>	Turns OFF or ON bit position (0 to 31) of integer numeric variables.		General		142
BRKPT	BRKPT [SET <line number> or BRKPT [DEL <line number> or BRKPT [DEL ALL] BRKPT	Sets (SET), deletes (DEL), and displays a list of breakpoints when no arguments are given.	Command			142
CDBL	CDBL (numeric expression)	Converts a value to a double-precision value real number.			Arithmetic operation	143
CHR\$	CHR\$ (numeric expression)	Finds a character that corresponds to the ASCII code (integer) given by the argument.			Character string	143
CINT	CINT (numeric expression)	Rounds off and changes into a short integer			Arithmetic operation	144

Command	Syntax	Details	Command	Statement	Function	Page
CLEAR	CLEAR	Initializes numeric and character variables and closes all ports.		General		144
CLNG	CLNG (numeric expression)	Converts a value to a long integer.			Arithmetic operation	144
CLOSE	CLOSE [#port number]	Closes a port.		Device control		145
CLS	CLS [#port number]	Clears the screen.		Device control		145
COM ON/OFF/STOP	COM [port number] ON or COM [port number] OFF or COM [port number] STOP	Enables, disables, or stops a communication interrupt (ON COM).		General		145
CONT	CONT	Resumes execution of a program that has stopped.	Command			146
COS	COS (numeric expression)	Determines the cosine of the value given by the argument.			Arithmetic operation	146
CSNG	CSNG (numeric expression)	Converts to single-precision real number.			Arithmetic operation	146
CTS	CTS (port number)	Reads CS signal status.			Special	147
DATA	DATA [Constant{,constant}]	Defines the numeric and character constants assigned using the READ command.		General		147
DATE\$	DATE\$ [=“YY/MM/DD”]	Sets or gives date.			Special	147
DAY	DAY [=numeric expression]	Sets or gives day.			Special	148
DEF FN	DEF FN Name [(argument[,argument,...])=expression]	Defines a BASIC function written by the user.		General		148
DEFINT/ DEFLNG/ DEFSNG/ DEFDBL/ DEFSTR	DEFINT character [-character]{character [-character]} or DEFLNG character [-character]{character [-character]} or DEFSNG character [-character]{character [-character]} or DEFDBL character [-character]{character [-character]} or DEFSTR character [-character]{character [-character]}	Declares variable types as integer, long integer, single-precision, double-precision or character.		General		149
DEL	DEL [line number 1] [-[line number 2]]	Deletes part of the program line.	Command			149
DIM	DIM variable name (maximum subscript) {maximum subscript}{variable name(maximum subscript{maximum subscript})}	Specifies the array variable dimensions and the maximum value for the subscript. For character string variables, the maximum number of characters can be specified.		General		150
DMA	DMA (port number)	Determines transmitting/not transmitting for DMA data transmission.			Special	151

Command	Syntax	Details	Command	Statement	Function	Page
DSR	DSR (port number)	Reads status of DR signal.			Special	151
DTR	DTR (port number) SET or DTR (port number) RESET	Sets/resets ER signal.		Device control		151
EDIT	EDIT [Line number]	Edits one line of the program.	Command			152
END	END	Terminates program execution and closes all ports.		General		153
EOF	EOF (port number)	Checks if the reception port buffer for the specified port is empty.			Special	153
ERC	ERC	Deletes the error code in the Error Log Table, or deletes the error status being generated.		General		153
ERL	ERL	Determines the line number being executed at the point that an error is generated.			Special	154
ERR	ERR	Determines the error code generated by the error.			Special	154
ERROR	ERROR <error number>	Simulates the error generation.		General		155
EXP	EXP (numeric expression)	Determine the exponential of a numeric expression.			Arithmetic operation	155
FCS	FCS (character expression, [FCS type])	Determines the error check code for the character expression.			Arithmetic operation	156
FIX	FIX (numeric expression)	Determines the integer part of value given by the argument.			Arithmetic operation	157
FOR ... TO ... STEP TO NEXT	FOR <variable> = <numeric expression> TO <numeric expression> [STEP <numeric expression>]: <program> NEXT variable {,variable} or FOR <variable> = <numeric expression> TO <numeric expression> [STEP <numeric expression>] line end program NEXT variable {,variable}	Repeats the loop from FOR to NEXT a specified number of times.		General		158
FRE	FRE	Assigns the amount of unused user memory.			Special	158
GOSUB TO RETURN	GOSUB line number to RETURN or GOSUB label to RETURN	Calls up and executes subroutines, and returns on RETURN command.		General		159
GOTO	GOTO <line number> or <label>	Branches to specified line number or label.		General		159

Command	Syntax	Details	Command	Statement	Function	Page
HEX\$	HEX\$ (numeric expression)	Changes the decimal value given by the argument to a hexadecimal character string.			Character string	160
IF	IF <numeric expression> THEN[statement{:statement} or <line number> or <label>] GOTO <line number> or <label> [ELSE[statement{:statement} or <line number> or <label>]]	Selects an executable statement or destination based on the results of a numeric expression.		General		161
INKEY\$	INKEY\$ [#port number]	Finds the characters that have been newly received by the reception buffer of a specified port.			Special	161
INPUT	INPUT[;] [#port number,] ["prompt";] variable{,variable} INPUT [#port number,] ["prompt";] variable {,variable}	Reads data separated by carriage returns from the reception buffer of a specified port into a specified variable.		General		162
INPUT\$	INPUT\$ (number of characters [,#port number])	Reads a string of characters of a specified number of characters from reception buffer of specified port.			Special	163
INSTR	INSTR ([search start position,] <character expression 1>, <character expression 2>)	Searches for character expression 2 in character expression 1, and obtains the position of the first character.			Character string	163
INT	INT (numeric expression)	Rounds down a decimal fraction to the nearest integer.			Arithmetic operation	164
INTRB	INTRB	Determines the line number being executed when an interrupt is generated.			Special	164
INTRR	INTRR	Determines the original interrupt type in an interrupt subroutine.			Special	165
INTRS	INTRS	Determines the line number that defined the generated interrupt.			Special	165
KEY ON/OFF/STOP	KEY [key number] ON or KEY [key number] OFF or KEY [key number] STOP	Enables, disables or stops a key input interrupt (ON KEY).		General		166
LEFT\$	LEFT\$ (<character expression>, <character number>)	Finds the specified number of characters beginning from the leftmost character of the character string.			Character string	166
LEN	LEN (character expression)	Determines the number of characters in a character string.			Character string	166
LET	[LET] <variable name>=<expression>	Assigns the value of an expression to a variable name.		General		167

Command	Syntax	Details	Command	Statement	Function	Page
LINE INPUT	LINE INPUT[;] [#port number,] ["prompt";] character variable	Reads one line of data from the reception buffer of a specified port into the character variable.		General		167
LIST	LIST [line number 1] [-line number 2]	Displays the program.	Command			168
LLIST	LLIST [line number 1] [-line number 2]	Outputs the program to the printer.	Command			169
LOAD	LOAD #<port number> [communications conditions character expression]	Transfers a program from the terminal to the ASCII Unit user memory.	Command			170
LOC	LOC (port number)	Provides the number of bytes of data stored in the reception buffer of a specified port.			Special	170
LOG	LOG (numeric expression)	Determines a natural logarithm.			Arithmetic operation	170
LPRINT [USING]	LPRINT [#port number] [USING print format;] expression {[,]expression} or LPRINT [#port number] [USING print format;] expression {[:]expression} or LPRINT [#port number] [USING print format;] expression { [space]expression}	Assigns LPRT as the device symbol (using the specified format), and outputs the expression value to a specified port.		General		171
MID\$	MID\$ (<character expression>, <character position> [number of characters])	Fetches the specified number of characters from a specified character position.			Character string	172
MID\$	MID\$ (<character expression>, <character position> [,number of characters])=<character expression>	Replaces the number of characters from a specified character position with specified character string.		General		173
MODEL	MODEL	Determines ASCII Unit model.			Special	173
NEW	NEW	Deletes the program currently being used and clears all variables.	Command			174
OCT\$	OCT\$ (numeric expression)	Converts decimal value to an octal character string.			Character string	175
ON ALARM	ON ALARM <time expression> GOSUB <line number> or ON ALARM <time expression> GOSUB <label>	Branches to interrupt subroutine after one-shot timer.		General		176
ON COM GOSUB	ON COM [port number] GOSUB <line number> or ON COM [port number] GOSUB <label>	Branches to interrupt subroutine when interrupted by specified port.		General		177

Command	Syntax	Details	Command	Statement	Function	Page
ON ERROR GOTO LINE NUMBER	ON ERROR GOTO <line number> or ON ERROR GOTO <label>	Branches to interrupt subroutine when error is generated.		General		178
ON NUMERIC EXPRESSION GOSUB OR GOTO	ON <numeric expression> GOSUB <line number> {,<line number> or <label>} or ON <numeric expression> GOSUB <label> {,<line number> or <label>} or ON <numeric expression> GOTO <line number> {,<line number> or <label>} or ON <numeric expression> GOTO <label>{,<line number> or <label>}	Branches to specified line number based on value of numeric expression.		General		179
ON KEY GOTO OR GOSUB	ON KEY [key number] GOTO <line number> or <label> or ON KEY [key number] GOSUB <line number> or <label>	Branches to interrupt subroutine on specific key input.		General		180
ON PC GOSUB	ON PC [interrupt number] GOSUB <line number > or ON PC [interrupt number] GOSUB <label>	Branches to interrupt subroutine when interrupted from PC.		General		181
ON TIME\$	ON TIME\$= <time character expression> GOSUB <line number> or <label>	Branches to interrupt subroutine at a certain time.		General		182
ON TIMER	ON TIMER <time expression> GOSUB <line number> or <label>	Branches to interrupt subroutine after interval time has expired, and repeats this process.		General		183
OPEN	OPEN #<port number>; "device symbol :[baud rate[,data length[parity[stop bit[,CS_ON or OFF[,RS_ON or OFF[,DS_ON or OFF[,XN_ON or OFF]]]]]]]"	Opens a specified port under specified conditions (device type, communications conditions, control signal operation)		Device control		184
OPTION BASE	OPTION BASE 0 or OPTION BASE 1	Designates the first number of the array variable subscript as 1 or 0.		General		186
OPTION LENGTH	OPTION LENGTH <numeric expression>	Sets the maximum length (1 to 255) for character string variable content.		General		187
PC EGET	PC EGET #<shared memory address>, <read number of words>,"format{,format}"; variable{,variable}	Reads data from shared memory of ASCII Unit itself to the variables.		General		188

Command	Syntax	Details	Command	Statement	Function	Page
PC EPUT	PC EPUT #<shared memory address>, <write number of words>,"format{,format}"; <numeric expression> {<numeric expression>}	Writes shared memory data of ASCII Unit itself.		General		189
PC GET	PC GET variable name 1 [,variable name 2]	Reads the output data value from the PC's allocated I/O to the variables.		General		190
PC ON/OFF/STOP	PC [interrupt number] ON or PC [interrupt number] OFF or PC [interrupt number] STOP	Enables, disables, or stops a PC interrupt.		General		187
PC PUT	PC PUT <numeric expression>	Writes the numeric expression value to the input data area of the allocated I/O in the PC.		General		190
PC QREAD@	PC QREAD"@area, address, <read number of words>, PC format {,PC format} ,variable {,variable}	Generates a request to execute an IOWR instruction at the PC, and reads the specified I/O memory area.		General		191
PC QWRITE@	PC QWRITE"@area, address, <write number of words>, PC format {,PC format}, numeric expression {,numeric expression}	Generates a request to execute an IORD instruction at the PC, and writes data to the specified I/O memory area.		General		193
PC READ OR PC READ@	PC READ" PC format {,PC format}"; variable {,variable} or PC READ" @ area, address, <read number of words>, PC format {,PC format}"; variable {variable}	Converts the format of data written from the PC and reads it.		General		195
PC WRITE ("@" )	PC WRITE" PC format {,PC format}"; variable {,variable} or PC WRITE" @area, address, <write number of words>, PC format {,PC format}"; variable {variable}	Converts the format of the numeric expression value and transfers it to the PC.		General		200
PEEK	PEEK (address)	Reads the contents of a specified memory address.			Special	205
PGEN	PGEN <numeric expression>	Switches to the program area number to be used.	Command			205
PINF	PINF [ALL] or PINF [Program No.]	Displays information on the user memory area.	Command			206
PMEM	PMEM ON or PMEM OFF	Switches between enabling and disabling write protection for the user memory area.	Command			207
PNAME	PNAME "program name"	Assigns or cancels a program name.	Command			207
POKE	POKE address,value	Writes data to specified address in the user memory.		General		208

Command	Syntax	Details	Command	Statement	Function	Page
PRINT [USING]	PRINT [#port number] [USING print format;] expression {[,]expression} PRINT [#port number] [USING print format;] expression {[:]expression} or PRINT [#port number] [USING print format;] expression {[space] expression}	Outputs expression value to specified port (in the specified format).		General		208
PWORD	PWORD <character expression>	Assigns a password to the BASIC program area.		General		210
RANDOM	RANDOM (expression)	Changes a random sequence based on a random number seed given by the expression.		General		210
READ	READ variable name [,variable name. . .]	Reads data defined using the DATA command into a variable.		General		211
REM	REM (comment statement)	Inserts a comment statement in a program.		General		211
RENUM	RENUM [new line number] [,old line number] [,increment]]	Reallocates line numbers in the program.	Command			212
RESTORE	RESTORE [line number] or RESTORE [label]	Specifies the line number of the DATA command that is starting to be read using the READ command.		General		212
RESUME	RESUME [line number] or RESUME [NEXT]	Resumes program execution after an error handling procedure has been performed.		General		213
RIGHT\$	RIGHT\$ (character expression, number of characters)	Finds the specified number of characters from the rightmost character of the character string.			Character string	213
RND	RND [(numeric expression)]	Determines a random number between 0 and 1.			Arithmetic operation	214
ROMLOAD	ROMLOAD	Reads the BASIC program from flash ROM and stores it in user memory (RAM).	Command			214
ROMSAVE	ROMSAVE	Stores BASIC program from user memory (RAM) in flash ROM.	Command			215
ROMVERIFY	ROMVERIFY	Compares BASIC programs in RAM and flash ROM.	Command			215
RTS	RTS #port number SET or RTS #port number RESET	Sets/resets the RS signal.		Device control		216
RUN	RUN [first line number] [TO final line number]	Starts executing a program, and continues executing until a specified final line number.	Command			216



Command	Syntax	Details	Command	Statement	Function	Page
SAVE	SAVE #port number [communications conditions character expression]	Reads the BASIC program in user memory area to terminal.	Command			217
SEARCH	SEARCH (integer type array variable, numeric expression 1, [numeric expression 2] [,numeric expression 3])	Searches an integer array for the specified value. Returns the array subscript of the first instance found.			Special	217
SGN	SGN (numeric expression)	Determines the sign of an argument.			Arithmetic operation	218
SIN	SIN (numeric expression)	Determines the sine (SIN) given by the argument.			Arithmetic operation	218
SPACE\$	SPACE\$ (number of characters)	Creates a string of spaces of the length of the number of characters.			Character string	218
SPC	SPC (numeric expression)	To output a number of spaces in a PRINT or LPRINT statement.			Character string	219
SQR	SQR (numeric expression)	Calculates the square root.			Arithmetic operation	219
STEP	STEP [numeric expression]	Executes the specified number of lines in BASIC program.	Command			220
STOP	STOP	Terminates program execution.		General		220
STR\$	STR\$ (numeric expression)	Converts the numeric value into a character string.			Character string	221
STRING\$	STRING\$ (numeric expression, numeric expression) or STRING\$ (numeric expression, character expression)	Finds a character string of the specified number of characters.			Character string	221
TAB	TAB (digit position)	Outputs blank space until the specified digit position in the PRINT and LPRINT commands.			Character string	222
TAN	TAN (numeric expression)	Determines the tangent (TAN) of the value given by the argument.			Arithmetic operation	222
TIME\$	TIME\$ ["HH:MM:SS"]	Sets or gives the time.			Special	222
TIME\$ ON/OFF/ STOP	TIME\$ ON or TIME# OFF or TIME\$ STOP	Enables, disables. or stops a time interrupt.		General		223
TIMER ON/ OFF/STOP	TIMER ON or TIMER OFF or TIMER STOP	Enables, disables, or stops a interval timer interrupt.		General		223
TRACE	TRACE	Displays the contents of the trace buffer.	Command			224
TROFF	TROFF	Cancel the tracing of the following line numbers.		General		224
TRON	TRON [M]	Starts tracing the following line numbers. Outputs to the trace buffer when the M option is used.		General		225

Command	Syntax	Details	Command	Statement	Function	Page
VAL	VAL (character expression)	Converts a character string into a numeric value.			Character string	225
VARPTR	VARPTR (variable name)	Determines or returns the memory address where the variable is stored.			Special	226
WAIT	WAIT "waiting time"[,line number]	Sets the monitoring time until execution of following instruction is completed.		General		227
WATCH	WATCH [SET variable name] or WATCH [DEL variable name]	Sets (SET), deletes (DEL), or displays the list (when not displayed) of the variables for the WATCH command.	Command			228
WHILE/WEND	WHILE <numeric expression> <program> WEND	Repeats the program body as long as the expression is not 0.		General		229

## **7-6 BASIC Commands**

# **NTLP**

Please insert MS word document.



## 7-7 User-defined BASIC Functions

User-defined BASIC functions are expressions created by the user in BASIC and saved as user-defined functions. Other BASIC functions can be called up and executed by their specified function name in the same way. Function names, input variables, and expressions are declared using BASIC's DEF FN command.

### User-defined BASIC Function Procedure

- 1,2,3...
1. Declare the following details using the DEF FN command.
    - BASIC function call name
    - Variable name input to the BASIC function
    - Expression
  2. Execute the program using the BASIC function name.

### User-defined BASIC Function Definition (DEF FN)

#### Syntax

DEF FN function\_name (variable\_name[, variable\_name]) = expression

#### Meaning

DEF FN declares a user-defined BASIC function. It sets the function name (same restrictions as for variable names) and the variable names for input to functions (more than one name is possible), and the expression.

Example:

DEF FNABC (A,B) = A+B

Function name  
 Variable  
 Expression

### BASIC Function Calls

As for other functions, user-defined functions are called by specifying the function name and then the variable(s) within parentheses.

**Example:** C = FNABC (A,B)

**Note** Text strings starting with reserved words cannot be used for function names. The following characters cannot be used at the end of function names: \$ % & ! #

## 7-8 Debugging

### 7-8-1 Overview

The debugging function allows you to stop the program at a specified line by various methods so that the contents of variables can be monitored. It is also possible to determine the position of the current line number.

**Note** The breakpoint function can be used to set a breakpoint in an interrupt sub-routine, but it is not possible to continue executing the interrupt subroutine after the breakpoint.

#### Breaks

Break at a specified line	By command input from the terminal	Break at multiple lines (255 settings max.)	Input BRKPT command	Program error
		Break after a single line or specified number of lines	Input STEP command	
		Execute up to the specified line	Input the RUN TO line no.	
	By internal program command	Break at the specified line	STOP command	

Break condition



Break condition



#### Variable Monitor

Variable content monitor (16 variables max.)	By command input from the terminal	Input WATCH command
--	------------------------------------	---------------------

#### Tracing the Current Line Number

By internal program command	TRON command
Save in the trace buffer, then monitor	TRON[M] command to TRACE command input

Each of the following debugging functions are described below.

- Break point function
- STEP function
- Variable monitor function
- Trace function

### 7-8-2 Breakpoint Function (BRKPT)

When a breakpoint is set at a specified line using the BRKPT command, the program will stop at that point. Once it has stopped, a message indicating a break in execution and the current line number will be displayed at the terminal. The ASCII Unit will switch to Command Mode. Up to 255 breakpoints can be set. To restart the program from the breakpoint, enter the CONT command from the terminal.

#### Syntax

BRKPT (SET | DEL) line\_No.

**Meaning**

BRKPT sets and deletes breakpoints. When no line is specified, the current breakpoint settings are displayed.

**Example**

BRKPT SET 200:	Sets breakpoint at line 200.
BRKPT DEL 100:	Deletes breakpoint at line 100.
BRKPT:	Displays the current breakpoint settings.
BRKPT DEL ALL:	Deletes all the current breakpoint settings.

**7-8-3 Step Function (STEP)**

Each time the STEP command is executed, the program executes a single line or a specified number of lines.

**Syntax**

STEP [numeric\_expression]

**Meaning**

The specified number of lines are executed. If no number is specified, 1 line is executed.

**Example**

STEP 5:	5 program lines are executed.
STEP:	1 program line is executed.

**7-8-4 Variable Monitor Function (WATCH)**

It is possible to monitor the contents of specified variables (except array variables) using the WATCH command. If the WATCH command is input, then whenever the BASIC program breaks due to a STOP command, a STEP command, a breakpoint, or a program error, the display will indicate that a break has occurred and the current line number, and then the value of the variable to be monitored will be shown at the terminal. The variables are displayed in order in which the WATCH commands were input. A maximum of 16 variables can be monitored at one time.

**Syntax**

WATCH (SET | DEL) variable\_name

**Meaning**

WATCH sets or deletes a specified variable to be monitored. When no variable is specified, the list of currently monitored variables is displayed. A maximum of 16 variables can be monitored at one time. An array variable cannot be specified for the variable name.

**Example**

WATCH SET A\$:	Sets A\$ to be monitored.
WATCH DEL B:	Stops monitoring B.
WATCH:	Displays the current list of variables being monitored.

**Examples of Variable Monitoring**

WATCH SET B\$:	Sets B\$ to be monitored
WATCH SET A:	Sets A to be monitored

When there is a break at line number xxx after the RUN command,  
BREAK IN LINE xxx

B\$ = "ABCD"

A = 3

**Note** Set the variables using the WATCH command after the program has been compiled for the RUN command.

## 7-8-5 Trace Function (TRON and TRACE)

When the TRON command is executed, the executed program lines are displayed at the terminal in order of execution. This allows you to see execution for programming such as for FOR to NEXT repetitions, and IF GOTO and GOSUB subroutine branching. When using the TRON command with the M option, trace data is stored in the trace buffer, and can be displayed at the terminal later using the TRACE command.

The trace buffer can store a maximum of 255 lines (when 255 lines are exceeded, lines are overwritten starting from the first line executed.)

**Note** Trace details stored in the trace buffer are backed up by a battery even after the power supply is turned off. This means that they can be read afterwards using the TRACE command. The trace buffer is cleared using the NEW or TROFF command.

### TRON

**Syntax** TRON[M]

#### **Meaning**

TRON starts a trace of the lines that follow. After RUN, the executed lines are displayed at the terminal enclosed in brackets [ ]. If the option [M] is used, the trace is output to the trace buffer. A maximum of 255 lines can be traced.

#### **Example**

TRON: Starts a trace of the lines that follow

TRON[M]: Starts a trace of the lines that follow, and stores the trace in the trace buffer.

#### **Examples of the Trace Function**

RUN

[10][20][30][20][40]....: Lines were executed in the order [10][20][30][20][40].

### TRACE

**Syntax** TRACE

#### **Meaning**

TRACE displays the contents of the trace buffer at the terminal. Executed lines are displayed sequentially inside square brackets [ ] in the order in which they were executed.

#### **Examples of the Trace Function**

TRON M

TRACE

[10][20][30][20][40].....: Lines were executed in the order [10][20][30][20][40].

**Note** Only when the execution line number is changed, the line number is added to the trace buffer. Therefore, if a line such as 100 GOTO 100 is executed, TRACE will display [100] one time only.

A DEVICE UNAVAILABLE IN PORT X error will occur if line numbers displayed for traces are duplicated when displaying program trace results on the terminal while executing an interrupt subroutine. If this occurs, use the M option for the TRON command to store trace results in the trace buffer.



# SECTION 8

## Data Exchange Application Programs

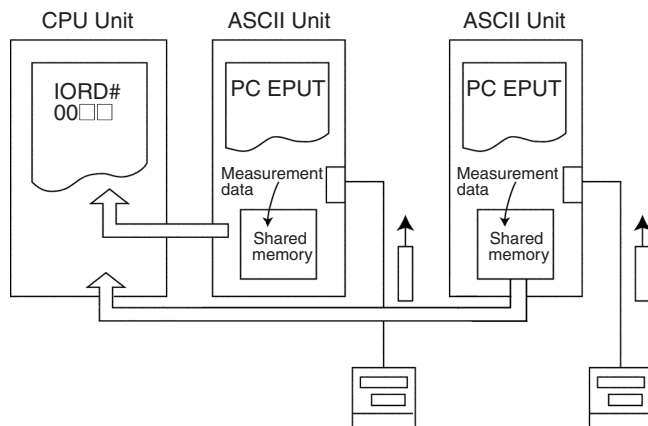
This section provides examples of data exchange applications.

8-1	Asynchronous Processing . . . . .	236
8-2	High-speed Data Exchanges . . . . .	237
8-3	High-volume Data Exchanges . . . . .	239
8-4	Data Transfer: ASCII Unit to CPU Unit . . . . .	239
8-5	Bit Data Exchanges . . . . .	240

# 8-1 Asynchronous Processing

## Example 1

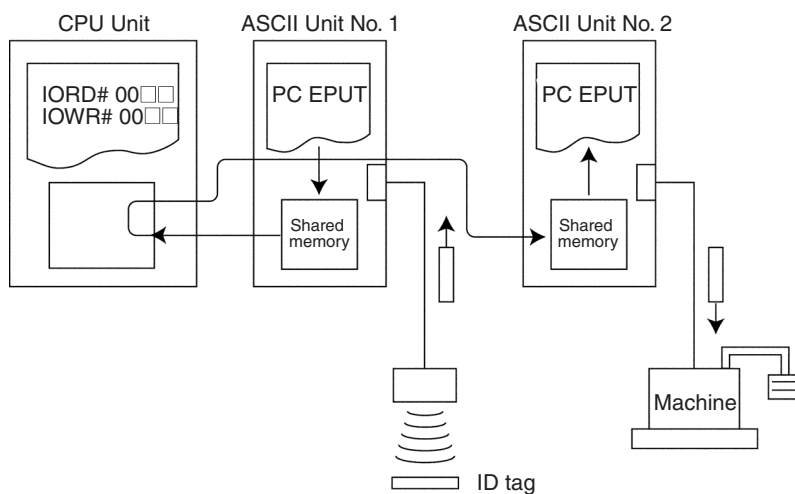
The monitoring system allows measurement data (PV) from a controller to be set whenever required in the shared memory of the ASCII Unit. The CPU Unit reads this data from the ASCII Unit's shared memory as required to monitor it.

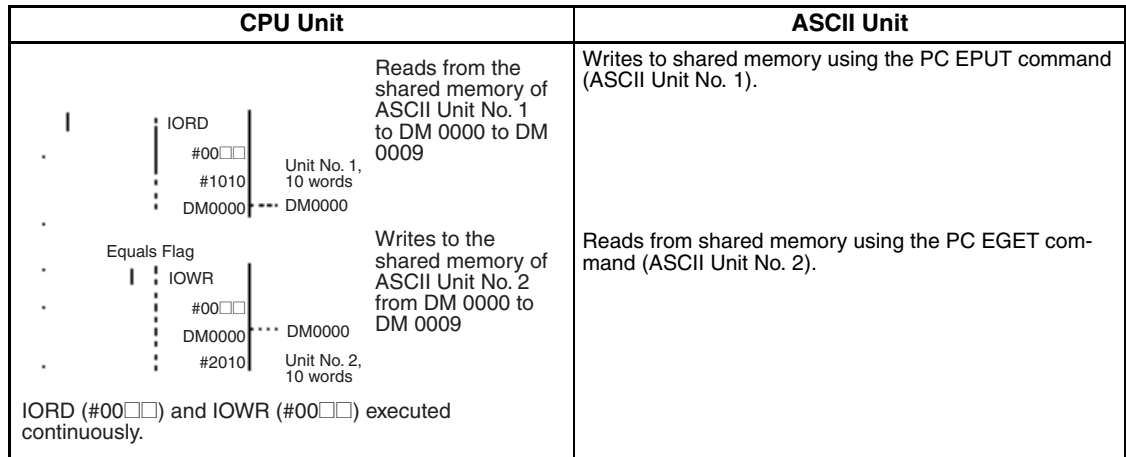


ASCII Unit		CPU Unit
Refreshes at high-speed the measurement data from the controller to the ASCII Unit's shared memory using the PC EPUT command.	→	Reads measurement data from the ASCII Unit as required, using the IORD (#00□□) instruction.

## Example 2

Here, various data is transferred between ASCII Units. The CPU Unit acts as an intermediary.



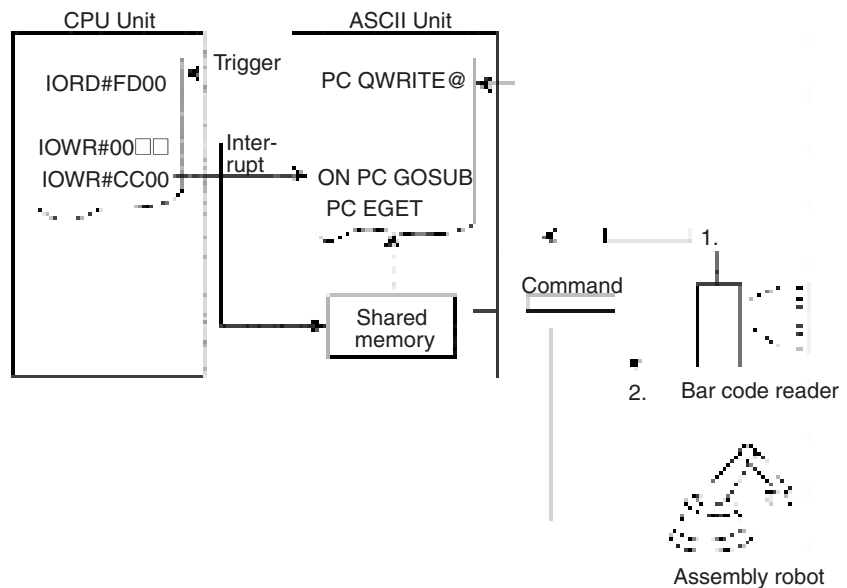


## 8-2 High-speed Data Exchanges

### Example 1

When the ASCII Unit functions as a communications interface, the trigger from the ASCII Unit enables CPU Unit processing to activate a trigger from the CPU Unit.

In response to data from a bar code reader, an assembly robot is controlled via CPU Unit processing.

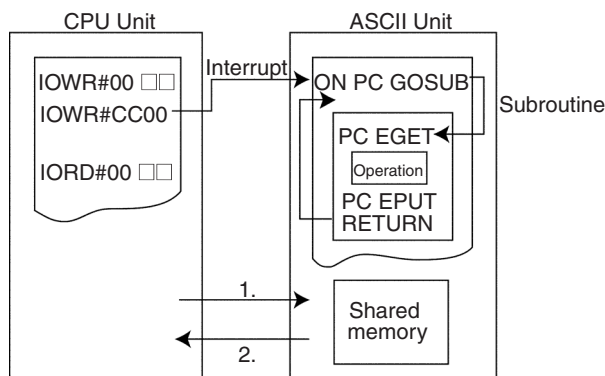


	ASCII Unit		CPU Unit
1	Sends an IORD Request Flag when the bar code reader data is input.	→	Receives the IORD Request Flag, and receives data via the IORD (#FD00) instruction.
			↓
2	After receiving the interrupt, the ASCII Unit reads the data written to the shared memory via the PC EGET command in the interrupt subroutine, and then issues commands to the assembly robot.	←	After processing, writes data to the ASCII Unit shared memory via the IOWR (#00□□) instruction, then sends an interrupt to the ASCII Unit via the IOWR (#CC00) instruction.

**Examples 2**

When the ASCII Unit operates as a special BASIC processing unit, a trigger is sent from the CPU Unit to the ASCII Unit, which after processing, sends a trigger back to the CPU Unit.

The CPU Unit assigns arithmetic processing outside its capability to the ASCII Unit, enabling high-speed processing for both Units by sharing the workload.

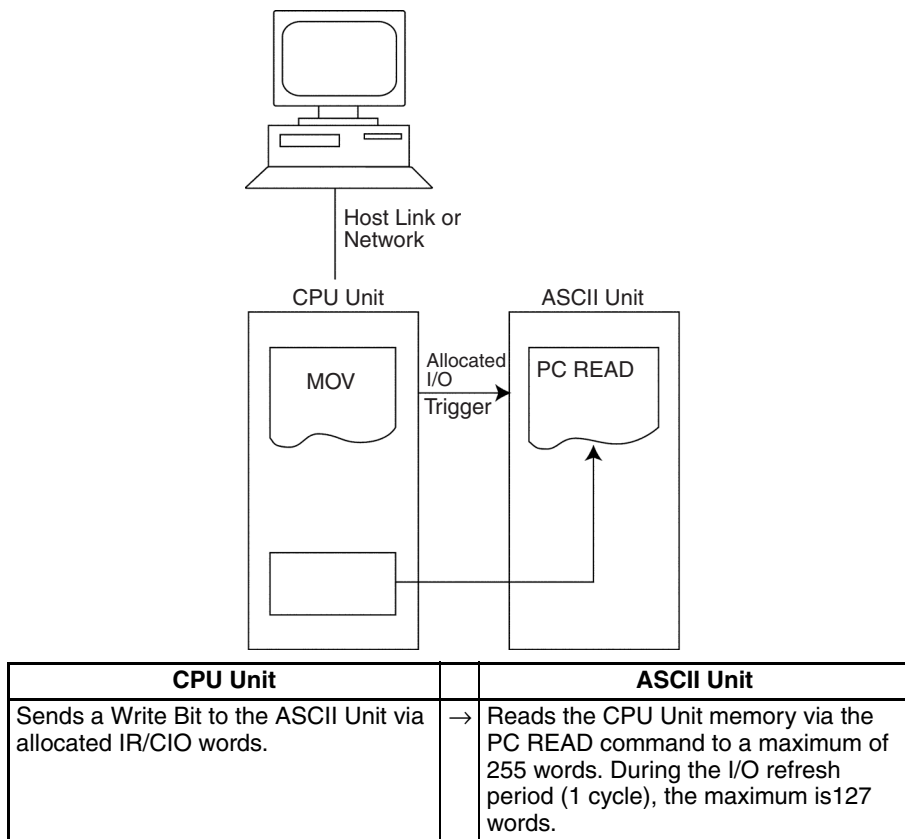


	CPU Unit		ASCII Unit
1	Writes data to the shared memory via the IOWR (#00□□) instruction, then sends an interrupt to the ASCII Unit via the IOWR (#CC00) instruction.	→	Receives the interrupt, reads the data written to the shared memory via the PC EGET command in the interrupt subroutine, then carries out special arithmetic operations.
			↓
2	At the completion of the interrupt subroutine (determined by checking the Interrupt Subroutine Completed Flag), reads from the shared memory via the IORD (#00□□) instruction.	←	Writes the result of the processing to the shared memory via the PC EPUT command; end of the interrupt subroutine.

### 8-3 High-volume Data Exchanges

**Example**

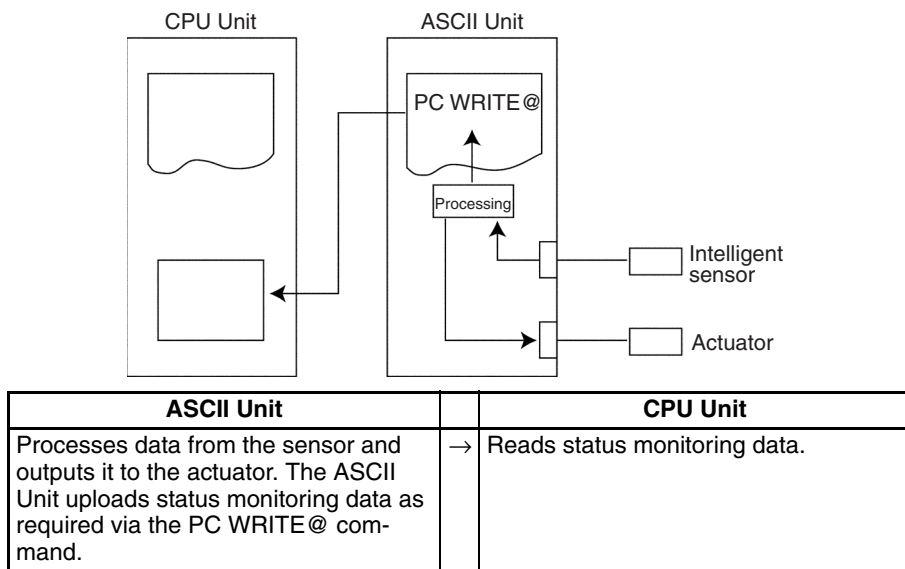
Large volumes of data can be transferred when high speed is not required by sending production directions as recipe data (settings/parameters) to various machines from the host computer.



### 8-4 Data Transfer: ASCII Unit to CPU Unit

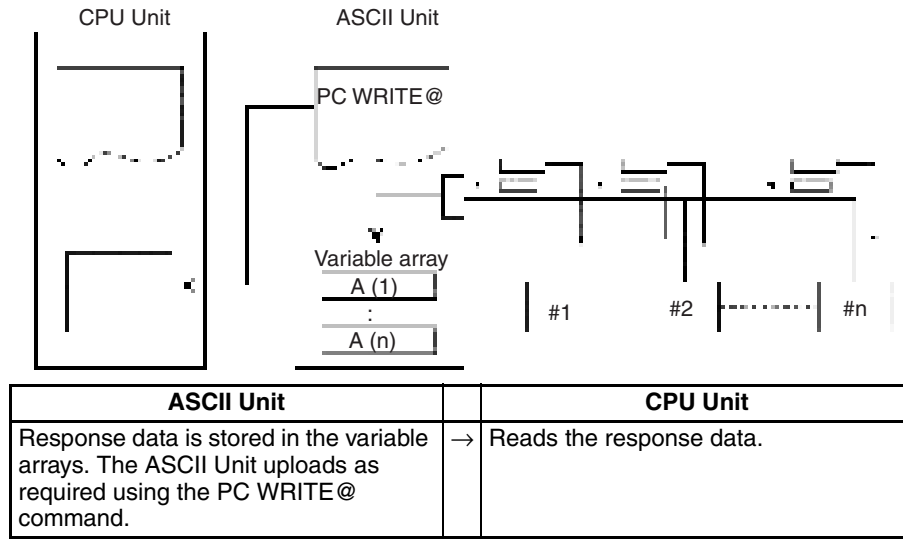
**Example 1**

The ASCII Unit is used as a control unit to upload status monitoring data.



**Example 2**

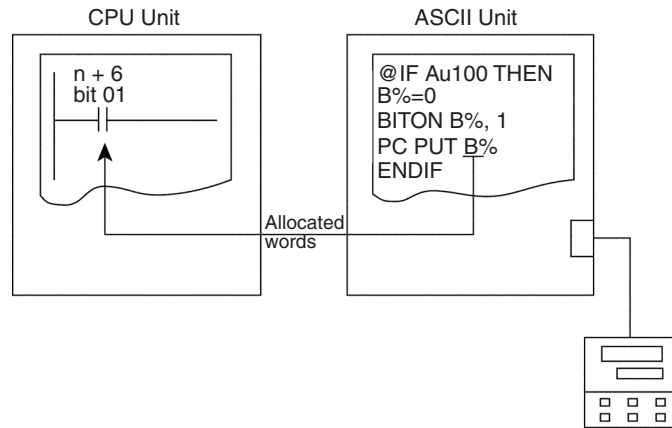
Multiple external devices are connected via the RS-422/485 ports, and polling is used to send commands. The response data is stored in variable arrays and uploaded to the CPU Unit as required.



## 8-5 Bit Data Exchanges

**Example**

A measured temperature from a temperature controller sends an ON signal to the CPU Unit once a certain temperature is reached.



# SECTION 9

## Clearing Error Messages

This section provides a list of error messages, probable causes and possible corrections.

- 9-1 List of Error Messages ..... 242
- 9-2 Troubleshooting ..... 248
  - 9-2-1 Procedure for Dealing with Damage to the Program Memory..... 251
- 9-3 CPU Unit Error Indicators ..... 252
  - 9-3-1 Special I/O Unit Error List ..... 252
  - 9-3-2 Useful Flags and Control Bits..... 253
- 9-4 Reading and Clearing the Error Log ..... 254
  - 9-4-1 Overview of Error Log Table ..... 254
  - 9-4-2 Reading the Error Log Table ..... 255
  - 9-4-3 Clearing Errors ..... 255
- 9-5 Maintenance ..... 255
  - 9-5-1 Mounting Precautions..... 255
  - 9-5-2 Replacing Batteries..... 256

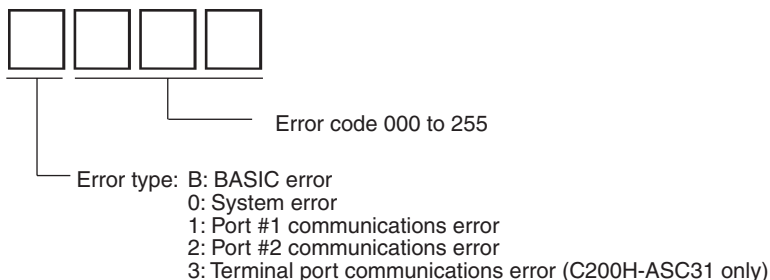
## 9-1 List of Error Messages

If an error (BASIC error) is generated during the execution of a BASIC command, or a system error is generated, the error code is stored in bits 00 to 11, and the error type is stored in bits 12 to 15 of the IR/CIO area (words n + 7) in the CPU Unit. At the same time, the relevant error message is displayed on the terminal. (Only an error code for errors 0080 to 0099 generated by the CPU Unit interface is stored in words n + 7, and no error message is displayed.)

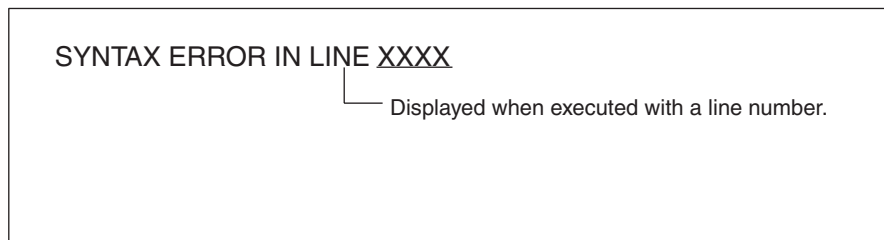
### IR/CIO Area Allocation

Word	Bit	Name	Value
n + 7	0 to 11	Error Code	000 to 255 BCD
	12 to 15	Error Type	B: BASIC error 0: System error 1: Port #1 communications error 2: Port #2 communications error 3: Terminal port communications error (C200H-ASC31 only)

### Error Type and Code



### Example Terminal Display



**Note** If the execution of a BASIC program is stopped when the terminal port is configured as a non-terminal device, the ASCII Unit will wait for a message to be displayed. In this case, connect the terminal to the terminal port and press Ctrl+X Keys, and the message will be displayed.



The various error types, codes, error messages and actions are as follows:

**BASIC Errors**

Error code	Error message	Probable cause	Corrective measures
B001	NEXT WITHOUT FOR	FOR command is missing.	Confirm the FOR....NEXT combination.
B002	SYNTAX ERROR	The program syntax is not correct. A variable name or label name starts with the same characters as a reserved word (command name).	Correct the program line. Change the variable or label name.
B003	RETURN WITHOUT GOSUB	Encountered RETURN command before the GOSUB command was executed.	Confirm the GOSUB....RETURN syntax.
B004	OUT OF DATA	The data to be read by the READ command is missing. The number and type of READ command variables and the number and type of the DATA command constants do not match or RESTORE command is not used.	Confirm the number and type of READ command variables, and the number and type of DATA command constants.
B005	ILLEGAL FUNCTION CALL	Illegal statement or function call.	Confirm the callout method for statements and functions (syntax).
B006	OVERFLOW	The numerical value exceeds the permissible range.	Check the numerical value range and the variable type.
B007	OUT OF MEMORY	Insufficient memory capacity.	There is not enough user memory. Check the status of memory use by using PINF or other command, and review the program, or free some memory.
B008	UNDEFINED LINE NUMBER	Input line number is incorrect.	Check the destination line number.
B009	BAD SUBSCRIPT ERROR	The array variable subscript exceeds the maximum value specified using the DIM command.	Use a value within the maximum subscript value range specified using the DIM command.
B010	DUPLICATE DEFINITION	The definitions of array variables or user-defined BASIC functions have been duplicated.	Change one of the duplicated array variables or user-defined BASIC functions.
B011	DIVISION BY ZERO	A division by zero has occurred.	Check the division and correct the program.
B012	ILLEGAL DIRECT	Execution of a statement has been attempted that is illegal in the Command Mode.	Can only be executed in a BASIC program (Program Mode).
B013	TYPE MISMATCH	The variable type does not match.	Check the variable type.
B014	---	Not used	---
B015	---	Not used	---
B016	---	Not used	---
B017	CANNOT CONTINUE	The program execution cannot be continued.	Restart the program using the RUN command.
B018	UNDEFINED USER FUNCTION	The user function is not defined.	Define the user function using the DEF FN command.
B019	NO RESUME	There is no RESUME command in the error interrupt subroutine within the ON ERROR GOTO command.	Add a RESUME command.
B020	RESUME WITHOUT ERROR	Although there is no error, the RESUME command has been executed.	Check the program.
0021	---	Not used	---
B022	MISSING OPERAND	The required parameters (operands) are missing.	Check the program syntax.

Error code	Error message	Probable cause	Corrective measures
B023	FOR WITHOUT NEXT	The NEXT command or the WHILE command is missing.	Check the FOR...NEXT or WHILE...WEND combinations.
B024	MISMATCH: WHILE/WEND	The WEND or WHILE command is missing.	Add a WEND or WHILE command.
B025	UNDEFINED SYMBOL	The character is outside the permissible range or a label is missing.	Use a character within the permissible range or check the program label.
B026	INVALID EXPRESSION	An invalid expression has been used.	Correct the expression.
B027	INVALID STATEMENT	An invalid statement has been used.	Correct the statement.
B028	---	Not used.	---
0029	MISMATCH: @IF/ENDIF	@IF and ENDIF do not match.	Check that @IF and ENDIF match.

**System Errors**

Error code	Error message	Probable cause	Corrective measures	
0030	FATAL ERROR: BUS ERROR	System error. (The ASCII Unit is not operating properly.)	Turn OFF the power, and then turn ON again. If the error persists, replace the Unit.	
0031	FATAL ERROR: ADDRESS ERROR			
0032	FATAL ERROR: PRIVILEGE VIOLATION			
0033	FATAL ERROR: UNINITIALISED INTERRUPT			
0034	FATAL ERROR: SPURIOUS INTERRUPT			
0035	FATAL ERROR: ILLEGAL INSTRUCTION			
0036	RTC ACCESS ERROR			Check battery and RTC. Otherwise, exchange Unit.
0037	FATAL ERROR: MPU IN LIMP MODE ERROR			Turn OFF the power, and then turn ON again. If the error persists, exchange the Unit.
0038	FATAL DEBUG ERROR: SYSTEM DEBUG ERROR IN PORT XXXX			
0039	---	Not used.	---	

**Operational Errors**

Error code	Error message	Probable cause	Corrective measures
0040	BREAK EXECUTED IN LINE XXXX	---	---
0041	CANNOT RUN WITH SWITCH IN STOP POSITION	Attempted to execute the RUN command with the switch in the STOP position.	Set the switch to START.
0042	USER BREAK	Program stopped by user.	---
0043	FILE TRANSFER INTERRUPTED	File transfer stopped while executing.	---
0044	PROGRAM MEMORY ERROR	Corrupted memory.	Toggle the START/STOP switch to initialize the memory and restart.
0045	---	Not used.	---
0046	---	Not used.	---
0047	---	Not used.	---
0048	---	Not used.	---
0049	---	Not used.	---

Error code	Error message	Probable cause	Corrective measures
0050	BAD PORT NUMBER	The port number is incorrect.	Check the port number.
0051	CANNOT READ FROM WRITE-ONLY DEVICE	Inputting to a write-only external device.	Check the port and external device specifications.
0052	PORT ALREADY OPEN	The port is already open. The same port has been opened twice using the OPEN command.	Delete one of the OPEN commands.
<input type="checkbox"/> 053	PARITY ERROR COMMUNICATING WITH DEVICE	An error has been generated while communicating with an external device.	Check the communications network.
<input type="checkbox"/> 054	---	Not used.	---
0055	PORT CONFIGURATION ERROR	Incorrect port communications parameters. If the ON COM interrupt is being used, the port communications parameters will not be set to proper values.	Check the communications parameters for the port or check that the restrictions on baud rate and communications interrupt conditions for port 1 and 2 have not been exceeded.
<input type="checkbox"/> 056	DIRECT STATEMENT	When the LOAD command was executed, a line with no line number was discovered in the program being transferred to the ASCII Unit.	Check the program being transferred to the ASCII Unit.
0057	PORT IS NOT OPEN	An unopened port number has been used.	Open the port using an appropriate OPEN command for the device symbol.
<input type="checkbox"/> 058	FRAMING ERROR COMMUNICATING WITH DEVICE	Incorrect framing (framing for start bit + data + parity + stop bit) for received data read from the port.	Check whether or not the communications conditions specified using the OPEN command match the framing of the external device.
0059	CANNOT WRITE TO READ ONLY DEVICE	Attempted to write to a read-only external device.	Check whether or not the device symbol specified using the OPEN command matches the external device specifications.
0060	DEVICE UNAVAILABLE	An invalid device symbol has been used in the OPEN command.	Check the OPEN command device symbol.
<input type="checkbox"/> 060	DEVICE UNAVAILABLE IN PORT <input type="checkbox"/>	I/O commands such as a PRINT command may be in conflict with each other due to an interrupt.	Correct the program so that I/O commands do not compete.
0061	---	Not used	---

**Note** The  indicates the port number generating the error.

## Execution Errors

Error code	Error message	Probable cause	Corrective measures
0062	PROTECTED PROGRAM	Program is protected from the NEW command.	When creating a new program using the NEW command, delete the program name using the PNAME" " command. If PMEM ON has been specified, execute the NEW command after executing PMEM OFF.
□063	INCOMING DATA HAS BEEN LOST	The data to be received has been lost.	Reduce the baud rate or use flow control.
0064	NOT SUPPORT	There is no support. Automatic transfer was specified even though the program is not saved in flash ROM	Check the positions of the OPTION LENGTH command and other commands for defining variables. Check the positions of commands.
0065	FLASHROM ERROR	There is an error with the flash ROM, or nothing has been written to the flash ROM.	Try again or replace the Unit.
0066	VERIFY ERROR	Error generated when comparing and verifying flash ROM	If there is no problem with the program in the user memory (RAM), rewrite the correct data to the flash ROM using the ROMSAVE command.
B067	FORMAT ERROR	The format or variable specifications for the PC READ or PC WRITE command are incorrect, or the number of transfer words or the specified first transfer word of the source in the IR/CIO area of the CPU Unit are incorrect.	Check the contents of the PC READ/ PC WRITE command at the ASCII Unit and words n + 3 and n + 4 in the IR/CIO area.
0068	---	Not used.	---
0069	PASSWORD INCORRECT	The password is incorrect.	Input the correct password.
0070	DATA TRANSFER CHECK SUM ERROR	Data sum error generated during data transfer or nothing written in flash ROM.	Perform the data transfer again.
0071	---	Not used.	---
0072	---	Not used.	---
0073	---	Not used.	---
0074	---	Not used.	---
0075	---	Not used.	---
0076	---	Not used.	---
0077	---	Not used.	---
0078	---	Not used.	---
0079	---	Not used.	---

**Note** The □ indicates the port number generating the error.

## PC Interface Errors

Error code	Error message	Probable cause	Corrective measures
0080	No message.	The IOWR/IORD instruction is incorrect.	Check the operands for the IOWR/IORD instructions.
0081	No message.	Incorrect number of transfer (read/write) words in the IOWR/IORD instruction.	Set the IOWR/IORD (#00□□) instruction to a maximum of 90 words and the IOWR/IORD (#FD00) instruction to a maximum of 128 words.
0082	No message.	Interrupt to the ASCII Unit using the IOWR (#CC00) instruction is incorrect.	Check whether or not the IOWR(#CC00) instruction interrupt number matches the ON PC (interrupt number) command.
0083	No message.	When reading and writing to the shared memory using the IOWR/IORD (#00□□) instruction, the first read/write word (offset word □□ from the beginning of the IOWR/IORD Area) is incorrect.	Check the offset word □□ in the IOWR/IORD (#00□□) instruction and the first words of the IOWR Area and IORD Area (m + 6 and m + 7 in the DM Area allocations).
0084	---	Not used.	---
0085	No message.	ON PC interrupt number is out of range.	Use 01 to 99.
0086	No message.	The PC interrupt number is an illegal BCD value.	Use a valid BCD value.
0087	---	Not used.	---
0088	---	Not used.	---
0089	---	Not used.	---

## DM Setting Errors

Error code	Error message	Probable cause	Corrective measures
0090	No message.	Incorrect setting for program number at startup (DM Area allocations).	Set any between 00 and 04 (BCD).
0091	No message.	Incorrect automatic transfer settings from flash ROM (DM Area allocations).	Set either 00 or 5A <sub>hex</sub> .
0092	No message.	Incorrect Start Mode settings (DM Area allocations).	Set either 00 or 5A <sub>hex</sub> .
0093	No message.	Incorrect initial value transfer settings (DM Area allocations)	Set either 0 or 1 (BCD).
0094	No message.	Not used (DM Area allocations).	Set 0 (BCD).
0095	No message.	Incorrect baud rate parameters for port 1, 2, or terminal (DM Area allocations).	Set any of 00 to 08 (BCD).
0096	No message.	Incorrect number of transfer words per cycle (DM Area allocations).	After checking that the PC is a C200HX/HG/HE or CS1-series PC, set either 00 or 5A <sub>hex</sub> .
0097	No message.	Invalid screen size or terminal mode.	Set the specified value within the proper range.
0098	No message.	The first transfer word setting for the IORD or IOWR Area is incorrect (DM Area allocations).	Set either 00, or between 10 and 99 (BCD).
0099	No message.	Invalid DMA option (port 1 or 2)	Set either 00 or 5A <sub>hex</sub> .

When an error is generated, the following commands are used to include an error processing program within a BASIC program.

Situation	Command	Remarks
To branch to an interrupt subroutine when an error has been generated.	ON ERROR GOTO command	Include an error processing program in the interrupt subroutine.
To obtain the error code that has been generated.	ERR function	Use this function in the interrupt subroutine at the ON ERROR GOTO destination.
To obtain the line number where the error occurred.	ERL function	Use this function in the interrupt subroutine at the ON ERROR GOTO destination.
To simulate error generation.	ERROR command	The action that is considered to have generated the error for a specified error code is performed.

## 9-2 Troubleshooting

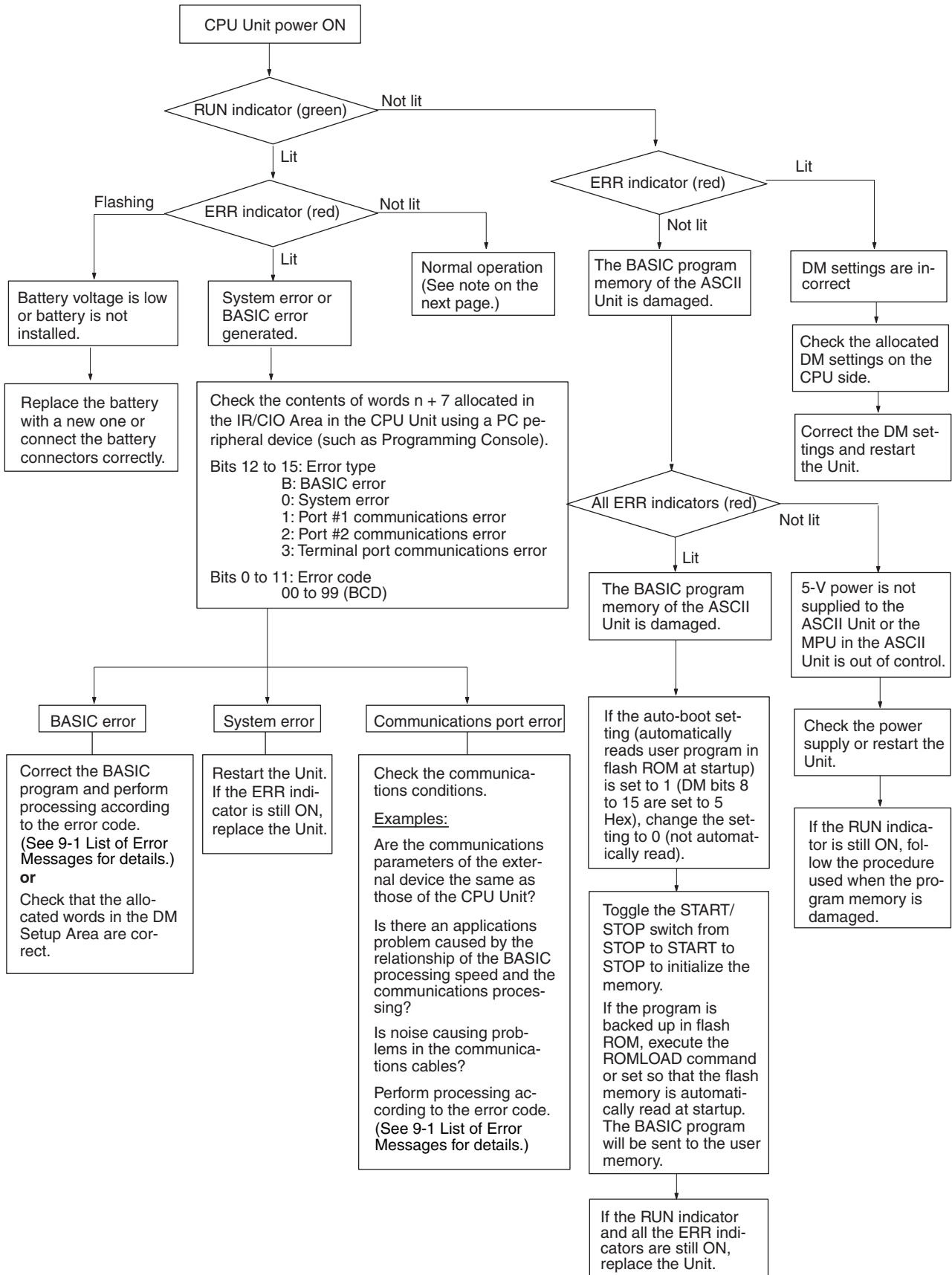
### Error List

Error	Probable cause	Possible correction
None of the indicators are lit.	Power to the PC itself is turned OFF.	Turn ON the PC power.
	The ASCII Unit is not mounted securely.	Mount the ASCII Unit securely.
	One of the Special I/O Units is faulty. (Waiting for the Special I/O Unit to start up) CPU Unit does not start operation.	Exchange the faulty Special I/O Unit. The faulty Unit has "\$" only displayed in the Read I/O Table operation. Before replacing the Unit, perform Corrective Measure 1, given below.
	The Special I/O Unit number occurs twice. (I/O Unit over) In such cases, the CPU Unit will not start operation.	Adjust the settings to ensure that the unit numbers do not occur more than once. The unit number can be viewed using Read I/O Table.
	The refresh between the CPU Unit and the ASCII Unit has not occurred properly. (Special Unit error) In such cases, only the ASCII Unit stops operating.	After eliminating the cause of the fault, Turn ON the corresponding Restart Bit (OFF→ON→OFF). If not restored after restarting, press the Ctrl+X Keys. Perform Corrective Measure 1, given below.
	An error has occurred in the Special I/O Unit or the CPU Unit as a result of damage to the BASIC program memory.	<b>Corrective Measure 1:</b> Set the user memory default switch on the back panel of the ASCII Unit to the right (ON) and turn the power OFF then ON again. Set the default switch back to the left and turn the power OFF then ON again. If the ASCII Unit still does not reset after this process, replace the Unit.
ERR1/2/T indicator is lit. (Port #1/port #2/terminal port communications error.)	The connecting cable is not connected to the device properly.	Connect the connecting cable correctly and secure with screws.
	The connecting cable is burnt out or the connection at the connector is faulty.	Repair or exchange the connecting cable.
	The baud rate or communications parameters of the connected device and the ASCII Unit do not match.	Match the baud rate and communications parameters of the connected device and the ASCII Unit.
ERR (error) indicator is lit. (System/BASIC error)	Incorrect data has been set for the DM Area in the PC. Alternatively, a BASIC program execution problem has been generated.	Check the error code and error message and take the appropriate action for that error.

<b>Error</b>	<b>Probable cause</b>	<b>Possible correction</b>
ERR (error) indicator is flashing. (Battery error)	The battery connector is disconnected.	Connect the battery connector properly.
	The battery voltage has dropped.	Replace with new battery.
All ERR indicators are flashing and PROGRAM MEMORY ERROR appears on the initial screen, and pressing Ctrl+X Keys is ineffective.	The BASIC program memory is destroyed.	<p>Change the DM setting so that the flash ROM is not read automatically at startup and then toggle the START/STOP switch on the front panel of the Unit to START, STOP, and START again. The BASIC program memory will be cleared.</p> <p>If the program is backed up in flash ROM, the program memory can be recovered afterwards using the ROMLOAD command.</p>

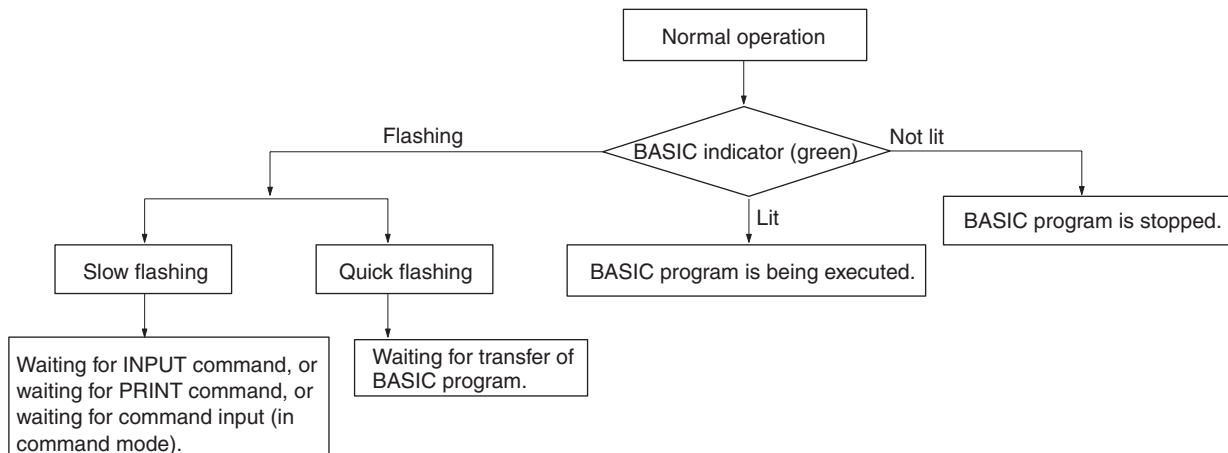
**Note** “Ctrl+X Keys” means to press the X Key while holding down the Ctrl Key.

Troubleshooting Flowchart





**Note** The following flowchart shows the BASIC indicator during normal operation.



### 9-2-1 Procedure for Dealing with Damage to the Program Memory

- 1,2,3...**
1. Turn OFF the power supply to the PC and remove the ASCII Unit from the Backplane.
  2. Set the user memory default switch on the back panel of the ASCII Unit to the right (ON) and attach the ASCII Unit to the Backplane.
  3. Turn ON the power supply to the PC and check that all the indicators on the ASCII Unit are flashing (except T/R). The CPU Unit is now ready for operation. if the DM setting is set to automatically read the flash ROM at startup (DM bits 8 to 15: 5A), set to 0 (flash ROM not automatically read). (If the indicators are not flashing, replace the Unit.)
  4. After checking that all the indicators are flashing, turn OFF the power supply to the PC and remove the ASCII Unit from the Backplane.
  5. Set the user memory default switch on the back panel of the ASCII Unit to the left (OFF), and attach the ASCII Unit to the Backplane.
  6. Turn the power to the PC ON, and check that the RUN indicator is lit. (If the RUN indicator is not lit, replace the Unit.)
  7. Since the program memory is now set to default (all clear), programs need to be transferred once more to RAM (by setting DM bits 8 to 15 to 5A, so that flash ROM is read automatically at startup, etc.).

**Note** When performing the above procedure, make sure that the DM setting for automatically reading the flash ROM at startup is not set to read (bits 8 to 15: 00). If the setting is already set to automatic (bits 8 to 15: 5A), set to 0 and then perform the procedure.

## 9-3 CPU Unit Error Indicators

The CPU Unit of the C200H/HS, C200HX/HG/HE, or CS1-series PC monitors errors that occur in the ASCII Unit as described in this section. The errors are indicated for the ASCII Unit as a Special I/O Unit.

### 9-3-1 Special I/O Unit Error List

Error	Error cause and operation	Remedy
Waiting for Special I/O Unit startup	The program memory of the ASCII Unit is damaged or there is a hardware fault at a Special I/O Unit. The PC will not begin operation in this condition.	Perform the procedure for damage to the program memory. If the Unit still will not restart, replace the Special I/O Unit where the error occurred.  The faulty Unit will be indicated by a "\$*." character in the I/O Table Read operation.
I/O Unit over	Two or more Special I/O Units have the same unit number setting. The PC will not begin operation in this condition. The Special I/O Unit Error Flag (SR 25415) will be ON.	Change the unit number settings to eliminate duplications.  The unit numbers can be listed with the I/O Table Read operation.
Special I/O Unit error	Refreshing between the CPU Unit and Special I/O Unit did not proceed normally. In this case, only the faulty Unit won't operate. The Special I/O Unit Error Flag (SR 25415 for C200H-family PCs; A40206 for CS1-series PCs) will be ON.	With a C200H/HS PC, determine the unit number of the faulty Unit from Error Flags AR 0000 through AR 0009. Restart the Unit by toggling the corresponding Restart Bit (AR 0100 to AR 0109) after eliminating the cause of the error.  With a C200HX/HG/HE PC, determine the unit number of the faulty Unit from Error Flags SR 28200 through SR 28215. Restart the Unit by toggling the corresponding Restart Bit (SR 28100 through SR 28115) after eliminating the cause of the error.  With a CS1-series PC, determine the unit number of the faulty Unit from Error Flags A41800 through A41815. Restart the Unit by toggling the corresponding Restart Bit (A50200 to A50215) after eliminating the cause of the error.  If the Unit doesn't restart after toggling the Restart Bit, perform the procedure for damage to the program memory. If the Unit still will not restart, replace the Unit.

### 9-3-2 Useful Flags and Control Bits

#### Special I/O Unit Error Flags

The PC error flags in the following tables will indicate the following errors.

- Duplicated unit numbers on Special I/O Units
- Refreshing between the CPU Unit and Special I/O Unit didn't proceed normally.

Flag address				Function
C200H/HS	C200HX/HG/HE	CS1		
		Errors	Unit Number Duplication	
AR 0000	SR 28200	A41800	A41100	ON when an error occurred in Unit 0.
AR 0001	SR 28201	A41801	A41101	ON when an error occurred in Unit 1.
AR 0002	SR 28202	A41802	A41102	ON when an error occurred in Unit 2.
AR 0003	SR 28203	A41803	A41103	ON when an error occurred in Unit 3.
AR 0004	SR 28204	A41804	A41104	ON when an error occurred in Unit 4.
AR 0005	SR 28205	A41805	A41105	ON when an error occurred in Unit 5.
AR 0006	SR 28206	A41806	A41106	ON when an error occurred in Unit 6.
AR 0007	SR 28207	A41807	A41107	ON when an error occurred in Unit 7.
AR 0008	SR 28208	A41808	A41108	ON when an error occurred in Unit 8.
AR 0009	SR 28209	A41809	A41109	ON when an error occurred in Unit 9.
---	SR 28210	A41810	A41110	ON when an error occurred in Unit A (10).*
---	SR 28211	A41811	A41111	ON when an error occurred in Unit B (11).*
---	SR 28212	A41812	A41112	ON when an error occurred in Unit C (12).*
---	SR 28213	A41813	A41113	ON when an error occurred in Unit D (13).*
---	SR 28214	A41814	A41114	ON when an error occurred in Unit E (14).*
---	SR 28215	A41815	A41115	ON when an error occurred in Unit F (15).*

**Note** \*Of the C200HX/HG/HE CPU Units, used only in the C200HX/HG-CPU5□-(Z)E C200HX/HG-CPU6□-(Z)E.

Model	Flag address	Function
C200HX/HG/HE(-Z) C200H/HS	SR 25415	ON when an error occurred in a Special I/O Unit.
CS1	A40206	
	A40113	ON with the same unit number has been assigned to more than one Special I/O Unit.

#### Special I/O Unit Restart Bits

To restart a Special I/O Unit, toggle (OFF → ON → OFF) the corresponding Restart Bit shown in the following table. These bits can be used to restart the Unit without turning off the power supply.

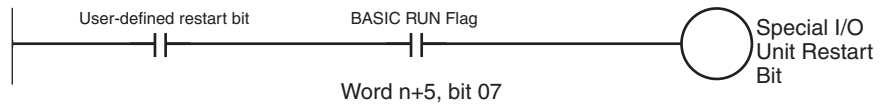
Bit address			Function
C200H/HS	C200HX/HG/HE	CS1	
AR 0100	SR 28100	A50200	Restarts Unit 0.
AR 0101	SR 28101	A50201	Restarts Unit 1.
AR 0102	SR 28102	A50202	Restarts Unit 2.
AR 0103	SR 28103	A50203	Restarts Unit 3.
AR 0104	SR 28104	A50204	Restarts Unit 4.
AR 0105	SR 28105	A50205	Restarts Unit 5.
AR 0106	SR 28106	A50206	Restarts Unit 6.
AR 0107	SR 28107	A50207	Restarts Unit 7.
AR 0108	SR 28108	A50208	Restarts Unit 8.

Bit address			Function
C200H/HS	C200HX/HG/HE	CS1	
AR 0109	SR 28109	A50209	Restarts Unit 9.
---	SR 28110	A50210	Restarts Unit A (10).*
---	SR 28111	A50211	Restarts Unit B (11).*
---	SR 28112	A50212	Restarts Unit C (12).*
---	SR 28113	A50213	Restarts Unit D (13).*
---	SR 28114	A50214	Restarts Unit E (14).*
---	SR 28115	A50215	Restarts Unit F (15).*

**Note** \*Of the C200HX/HG/HE CPU Units, used only in the C200HX/HG-CPU5□-(Z)E C200HX/HG-CPU6□-(Z)E.

Program memory in the ASCII Unit may be destroyed if the ASCII Unit is restarted while it is executing startup processing after the power supply is turned ON. Program restarts so that they will not occur during startup processing.

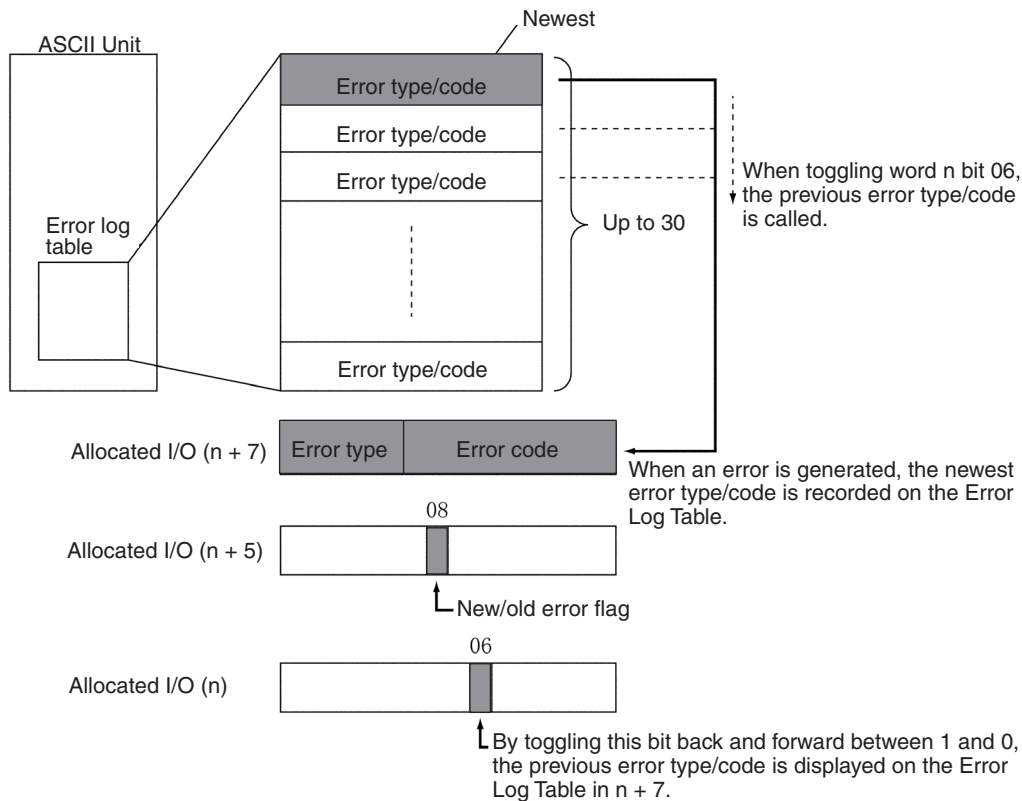
**Example:**



## 9-4 Reading and Clearing the Error Log

### 9-4-1 Overview of Error Log Table

The ASCII Unit has an error log table which contains up to of 30 errors. When an error occurs, the error code and type are stored in the Error Log Table.



### 9-4-2 Reading the Error Log Table

The Error Log Table is reflected in the Allocated I/O as shown below. Therefore, the table can be read from the CPU Unit via the Allocated I/O.

- When an error is generated, the newest error code and error type is set to  $n + 7$  words in the IR/CIO Area Allocations.
- To trace past error codes and error types, refer to the Error Log Table when toggling the Allocated I/O word  $n$  bit 06 from 1 to 0 and 0 to 1. The previous error code and error type will be set to  $n+7$  words. If there are no older errors the  $n + 7$  words will become  $00_{hex}$  once and then return to the newest error.

Note The ERR function is used when an error is generated and that error code is to be read in the BASIC program.

- The error that has been read can be monitored to determine whether or not it originated after the program was run (i.e., old or new error information) by using word  $n + 5$  bit 08 in the IR/CIO Area. This bit matches each error type/code for the error log.

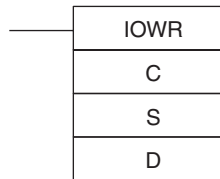
### 9-4-3 Clearing Errors

Errors are cleared using the ERC command as follows:

By executing the ERC command while in Command Mode, the error status at that point will be deleted and the error types and codes in the Error Log Table will also be cleared.

#### Clearing Errors from the ASCII Unit

The IOWR (SPECIAL I/O UNIT WRITE) instruction can be executed with a control code of #EC00 to delete error status (such as with ERC in program mode). The generated error type and code will remain in the Error Log Table.



C: #EC00  
 S: Any (data must be 0000)  
 C200HX/HG/GE  
 D: Bits 12 to 15: Unit number (0 to F)  
 Bits 00 to 11:  $001_{hex}$   
 CS1-series PCs  
 D: Bits 00 to 03: Unit number (0 to F)  
 D+1:  $0001_{hex}$

#### Clearing Error Codes Using BASIC Commands

The ERC command can be executed in Program Mode to delete the current error status. The deleted error type/codes will remain in the Error Log Table. These errors will all be given the status "old."

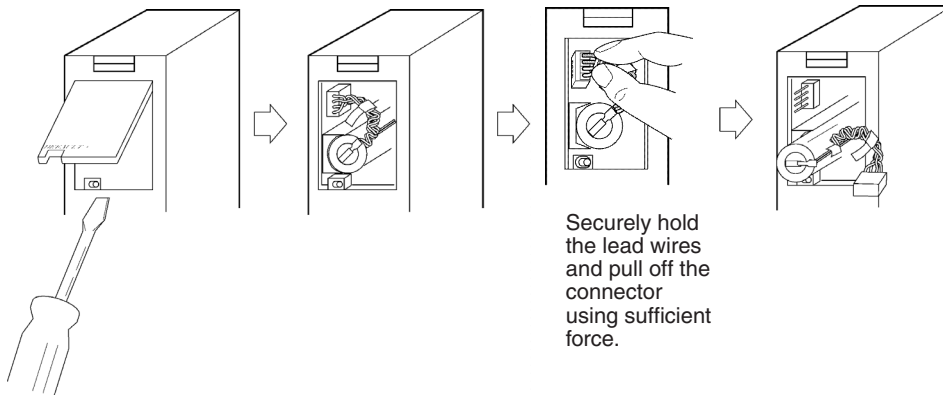
## 9-5 Maintenance

### 9-5-1 Mounting Precautions

- Turn OFF the power before exchanging ASCII Units.
- After exchanging ASCII Units, confirm again that the new Unit is not faulty.
- If the ASCII Unit has a fault, the user memory area (RAM) in the ASCII Unit and the BASIC program or library functions in the flash ROM cannot be read in some cases. Save the compiled BASIC program and library functions on an external device (e.g., personal computer).

## 9-5-2 Replacing Batteries

- 1,2,3...
1. Turn OFF the power. (If the power is already OFF, turn ON the power for at least one minute and then turn it OFF again.)
  2. Press down on the lock lever of the Backplane with a screwdriver, and remove the ASCII Unit.
  3. Remove the cover of the battery case on the back panel of the ASCII Unit using a flat screwdriver.
  4. Unclip the battery connectors, and replace the batteries. This operation must be completed within five minutes.



5. Replace the battery case cover.
6. Mount the ASCII Unit to the Backplane.

### Replacement Batteries

Name: Battery Set  
Model: C200H-BAT 09

### Battery Life and Replacement Time

- The effective period (maximum life) of a battery is five years at an ambient temperature of 25°C, whether or not the ASCII Unit is connected to electricity. If the ambient temperature is too high, the life of the batteries will be shortened.
- Replace the ASCII Unit within one week of the ERROR indicator flashing (red) on the front panel of the unit. Once the ERROR indicator starts to flash (red) on the front panel of the unit, the Battery Error Flag (word n+5 bit 06) will turn ON. Use this flag for battery maintenance.

**Note** Having a spare ASCII Unit and battery available will enable quick and smooth repairs on faulty Units.

# Appendix A

## Operating Precautions

### Changing from C200H-ASC02

#### Execution Time

The time required by the C200H-ASC11/ASC12/ASC31 to execute commands is different from the C200H-ASC02. Therefore, the execution time of BASIC programs will depend on the ASCII Unit being used.

#### Machine Language Programs

Programs created on the C200H-ASC02 using machine language cannot be used with the C200H-ASC11, C200H-ASC21, or C200H-ASC31 ASCII Unit.

#### Integer Format

In the C200H-ASC11/ASC21/ASC31, real numbers (single-precision and double-precision) are in floating-point constant format. Since the standard has changed to conform to IEEE-754, the results of operations using real numbers may change. If precise operations are required, use double-precision variables for the arguments.

Example:      10 A# = 0.9 + 0.1  
              20 A = ACOS (A#)

#### Command Competition in Interrupt Processing

In the C200H-ASC02, an interrupt cannot be generated while a BASIC program command is being executed. This is possible, however, for the C200H-ASC11/ASC21/ASC31. Therefore, if an interrupt is generated while a command is being executed and the same command is included in the interrupt subroutine, there will be competition between the two commands and operations will not run smoothly (e.g., if an interrupt is generated while the PRINT command is executing and the interrupt subroutine contains the PRINT, INPUT, LINE INPUT command for the same port). To stop competition between commands from occurring, exclusively control interrupts by using control commands to enable, disable, and stop execution.

#### Handling Communications Errors

In the C200H-ASC02, the BASIC program will not stop executing due to a communications error. In the same way, if a communications error is generated with the C200H-ASC11/ASC21/ASC31 and the ON ERROR command is not used, the BASIC program will not stop executing. The BASIC program will stop executing, however, if the ON ERROR command is used and a new error (not a communications error) is generated while an error interrupt subroutine is being executed.

#### Internal Bit Allocations

The functions of the words and bits allocated in memory have changed. Make sure to check the allocations when using ladder programs in the CPU Unit. Functions and operations for the ASCII Busy Flag have changed, so exercise caution when using this flag.

#### ASCII Unit System Setup

The method of setting up the ASCII Unit system has changed from using the DIP switch on the back of the Unit to using 10 words at the beginning of the DM Allocation Area. Use a PC peripheral device to set the allocated DM words in the CPU Unit.

## Communications Transmission Code and Transfer Control Signal

Depending on the device code specifications for the OPEN command, the limitations of the codes output when executing the PRINT command have changed (all transmission codes have been changed to outputs except for those specified using the TERM command for port 2). In the same way, depending on the device code specifications for the OPEN command, the operations of the transfer control signals (timing chart) have been changed. Check the interface and external devices (See SECTION 4 Data Exchange with General-purpose External Devices).

## Terminal Emulation Mode Selection

If the same terminal used with the C200H-ASC02 is to be used with the C200H-ASC11/ASC21/ASC31 and it supports VT100 mode, use VT100 mode. If the terminal does not support VT100 mode, use FIT10 mode.

## Using PC READ/PC WRITE (Including @) Character-string Variables

In the C200H-ASC02, when character-string variables are used for the variable with the PC READ/PC WRITE (including @) commands, the first four characters in the character string are written, and only these four characters are read to the variable. For the C200H-ASC11/ASC21/ASC31, all of the characters stored in the variable are written and word data up to 255 characters can be read.

### Example: C200H-ASC02

```
PC READ "@D,0,5,5H4" ; A$,B$,C$,D$,E$
```

### Example: C200H-ASC11/ASC21/ASC31

```
PC READ "@D,0,5,5H4" ; A$(Five words of data will be read from A$.)
```

The following example show the difference in operations between the C200H-ASC02 and the C200H-ASC11/ASC21/ASC31 when the following format is used.

### Example: PC READ "@D,0,5,5H4" ; A\$(0)

C200H-ASC02: The first word will be stored in A\$(0).

C200H-ASC11/ASC21/ASC31: Five words of data will be stored in A\$(0).

If changing from the C200H-ASC02 to the C200H-ASC11/ASC21/ASC31, the program will also need to be changed as the operations are different.

### Example 1

Before changing:

```
10 PC READ "@D,0,10,S10H4" : A$(0)
```

Executing this program with a C200H-ASC11/ASC21/ASC31 will generate a FORMAT ERROR.

After changing:

```
10 PC READ "@D,0,10,S10H4" : A (0)
```

```
20 FOR I=0 TO 9 : A$(I)=RIGHT$ ("0000" + HEX$(A(I)),4) : NEXT I
```

### Example 2

Before changing:

```
10 PC READ "@D,0,5,5H4" ; A0$,A1$,A2$,A3$,A4$
```

After changing:

```
10 PC READ "@D,0,5,5H4" ; A$
```

```
20 A0$=MID$(A$,1,4)
```

```
30 A1$=MID$(A$,5,4)
```

```
40 A2$=MID$(A$,9,4)
```

```
50 A3$=MID$(A$,13,4)
```

```
60 A4$=MID$(A$,17,4)
```

When using a C200H-ASC11/ASC21/ASC31, PC and DM data can be easily sent to and from peripheral devices.



**Example**

DM 0000	1234	→ 100 PC READ "@D,0,3,3H4";A\$ → A\$"123456789ABC"
DM 0001	5678	110 SD\$=HEAD\$+A\$+TERMS
DM 0002	9ABC	120 PRINT SD\$
⋮	⋮	

**Example 3**

Before changing:

```
10 PC READ "@D,0,255,100A3,100A3,55A3" ; A$,B$,C$
```

If the above syntax is used for a C200H-ASC11/21/31, a FORMAT ERROR will occur.

After changing:

```
10 PC READ "@D,0,100,100A3" ; A$
20 PC READ "@D,0,100,100,100A3" ; B$
30 PC READ "@D,0,200,55,55A3" ; C$
```

A maximum of 127 transfer words can be placed into multiple variables by separate the transfer values using the MID\$ function in the same way as shown in Example 2.

Before changing:

```
10 PC READ "@D,0,4,2A3,2A3" ; A$,B$
```

After changing:

```
10 PC READ "@D,0,4,4A3" ; A$
20 A$=MID$(A$,5,4)
30 A$=MID$(A$,1,4)
```

For the C200H-ASC11/21/31, only one character-string variable can be used for each variable. The multiple settings that could be performed for the C200H-ASC02 are not possible. For details on the formats for other PCs, refer to Appendix C PC Format.

## Terminal Operations

### Terminal Software Limitations

VT100 mode can be used as a terminal emulation mode for the C200H-ASC11/ASC21/ASC31. Nevertheless, even if the terminal software used supports VT100 mode, the limitations of the actual terminal software functions may result in some ASCII Unit functions being unavailable. (For example, the Cursor Keys on the Hyper-terminal for the Japanese version of WINDOWS 95 cannot be used.)

### EDIT Command Restrictions

Do not continuously press the Up and Down Cursor Keys when using them to move lines being edited with the EDIT command. Doing so will add the Up or Down Cursor Key code to the edited lines.

### Executing Programs

In the C200H-ASC02, the BASIC source program is executed as it is input. To improve the execution speed of the BASIC program in the C200H-ASC11/ASC21/ASC31, once a BASIC source program is input, it is compiled to an intermediate code (a simpler code that identifies the command contents) and then it is executed. Therefore, if the RUN command is executed using the C200H-ASC11/ASC21/ASC31, there is a delay while compiling is being performed before the actual program is executed. The RUN compiling time is approximately 10 seconds in the 53-KB (approx. 1,300 lines) program. This time will vary according to the commands used in the program. The intermediate code is regenerated the first time the program is executed after the program (area) is changed.

## Saving Programs

Never turn OFF the power to the PC while writing the program to flash ROM using the ROMSAVE command (until the command prompt is displayed at the terminal). Doing so will damage the flash ROM and saving data may be no longer possible. Therefore, be sure not to disconnect power to the PC while the ROMSAVE command is being executed.

## RS-422A/485 Ports

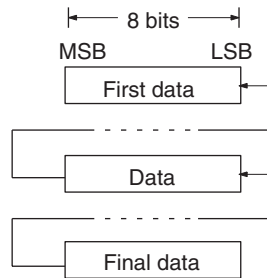
Port 2 (RS-422A/485) of C200H-ASC21 can be used in 1:N communications, however, full-duplex communications are not possible with this port.

## FCS Command Error Check Code Arithmetic Operation

The calculations for CRC-16 and CRC-CCITT used with the FCS command to determine the error check code of the character expressions is described below. Check that the equations are the same as those for the external devices.

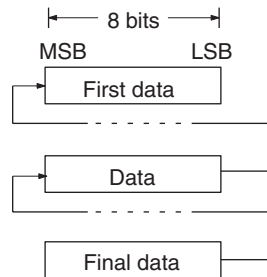
### CRC-CCITT: Using XMODEM

All 8-bit data is placed in series, it is then divided by a specific 17-bit binary value (1, 0001, 0000, 0010, 0001), and the remainder (16-bit) is used as the error check code as shown below. The initial value is 0 (16 bits all 0).



### CRC-16: Using MODBUS

All 8-bit data is placed in series, it is then divided by a specific 17-bit binary value (1, 0001, 0000, 0010, 0001), and the remainder (16-bit) is used as the error check code as shown below. The initial value is -1 (16 bits all 1).



## Baud Rate for Receive Data Interrupt

When an ON COM interrupt is used, the baud rate of all ports will be as follows:

Port 1 + port 2 ≤ 38.4 Kbps

If the baud rate is set to exceed this condition, a PORT CONFIGURATION ERROR will be generated.

## Baud Rate for PC Data Transfer

A maximum of 127 words of data can be transmitted in one cycle. If executing PC READ/PC WRITE with data of a comparatively large size (such as 127 words) and the data is being received using a high baud rate (19.2 kbps or higher), the overlap of data being received and sent will extend the time required to complete the transmission and the cycle time at the PC will also be extended. The effect on the PC cycle time can be minimized by transmitting a maximum of 20 words per cycle.

## Operations after Aborting Program Execution

If a BASIC program that is being executed with an open communications port is aborted by pressing the Ctrl + X Keys, and data has already been received at the communications port, the return of the prompt after the LIST display command is executed may be delayed. To stop the delay, after the program execution has been aborted input the CLOSE command, and then execute the LIST display command.

## Executing BASIC Interrupts during Command Execution

When a BASIC program is stopped due to the wait status of a command such as WAIT, a BASIC interrupt will occur (ON COM, ON PC, etc.) but only one interrupt subroutine will be executed for all BASIC interrupts. All other interrupts will be held and a record made of each.

## Program Startup Time

If a program is saved to ROM using ROMSAVE immediately after being debugged, the program memory may contain garbage. This may cause delays in starting program execution after turning ON the power supply. If this occurs, save the program at a terminal, load the program from the terminal, and then use ROMSAVE.

## Power Interruptions or Restarts during Program Memory Operations

Do not allow PC power to be interrupted or restart the CPU Unit while program memory operations are being performed, including those for LOAD, SAVE, ROMLOAD, ROMSAVE, autoboot, and compiling for RUN. Program memory may be destroyed, preventing further operation.

## Consumption of User Memory through Repetition of Subroutine Branching

Observe the following precautions when using the following two items in combination.

- 1: Subroutine using the GOSUB or interrupt command (ON COM, ON KEY, ON PC, ON TIMER, ON ALARM, ON TIME\$, or ON ERROR)
- 2: GOTO, ON ERROR GOTO/RESUME, ON KEY GOTO, or WAIT command

When the subroutine using the GOSUB or interrupt command (1 above) is forced to branch using the GOTO command (2 above), data will remain in the stack area, consuming the user memory area inside the ASCII Unit. When this branching is repeated, an OUT OF MEMORY error (B007), one of the BASIC errors, will occur or the stack area will overflow, which may result in a malfunction of the system operation.

Be sure to design the program so that it always returns to the subroutine after branching using the GOTO command and exits from the subroutine using the RETURN command. Refer to the following example.

Example:

```
10 ON ERROR GOTO 100
20 PRINT FRE
30 GOSUB 1000
90 END
100 PRINT ERR,ERL;
110 RESUME 20
1000 A=1/0 'Divide by zero
1010 RETURN
```

In the above example, the subroutine will be branched from the 1,000th line to the 100th line using the ON ERROR GOTO command if an error occurs.

In this case, exit from the subroutine by changing the 110th line from RESUME 20 (to resume from the 20th line after error processing) to RESUME NEXT (to resume from the line following the line that caused an error.)

## **ASCII Unit Operation when Program Memory is Damaged**

When program memory is damaged a PROGRAM MEMORY ERROR or a Special Unit error on the PC side will occur and CPU Unit operation will be held. The PC can be restarted by using the default switch on the back panel of the ASCII Unit. In this situation, the program memory will be forcibly initialized, so always be sure to use ROMSAVE to save programs to flash ROM memory after inputting them. Refer to Reading the BASIC Program from Flash ROM in 7-1 Programming Procedure, for details on how to load the saved program to flash ROM.

Be careful when overwriting data in the user memory area using the POKE command, as damage to the program memory may result. When damage to the program memory is caused by the use of the POKE command, a memory error may not occur immediately.

# Appendix B

## Comparison with C200H-ASC02

The following table provides details of the differences between the C200H-ASC11/ASC21/ASC31 and the C200H-ASC02 ASCII Units. For an overview of the comparison, see 1-5 Comparison with Previous ASCII Units.

Item		C200H-ASC02	C200H-ASC11/21/31
Executing programs		Executes the source program as it has been input as soon as RUN is executed.	Generates intermediate code for the source program before executing it, so there is a short delay between when RUN is executed and the program is executed.
Memory	Allocation in the character variable area	Dynamic method.	The user can select between a static (fixed) method and a dynamic (free) method, depending on whether the OPTION LENGTH command is used.
	Defragmentation (reorganization of unused areas)	Executed while program is executing.	Executed when the RUN command is executed.
	Setting memory areas	Memory areas specified using the FRE command.	None
Communications	Communications buffers	Receive buffer: 256 bytes Transmit buffer: 256 bytes	Receive buffer: 512 bytes Transmit buffer: 512 bytes
	ER (DTR) code	Controlled by the system.	Also can be controlled by the DTR command in the program instead of the system.
	Indicator showing that data is waiting to transmit.	The ERR indicator is lit.	BASIC indicator flashes slowly.
	Limiting the output code when executing the PRINT command, depending on the device code specifications for the OPEN command.	Depending on the code, some codes are not output, and some are output with LF added.	The TERM command cannot be used for communications port #2. Otherwise, there is no limitation on outputs.
	DMA (Direct Memory Access) data transmission when sending data with the PRINT command.	Not available. (DMA transmission is not available. Depending on the software, send data and stop the BASIC program execution.)	Available. (DMA transmission can be enabled or disabled. While software data transmission is being performed, parallel hardware data transmission is possible by using a trigger from the software.
Exchanging data with the CPU Unit	Number of data words that can be exchanged using PC READ/PC WRITE (including @) in one cycle of the CPU Unit	20 words max.	20 words or 127 words max. (available for C200HX/HG/HE or CS1-series PCs only)
	Amount of data that can be exchanged using I/O allocation.	CPU-to-ASCII Unit: 8 bits ASCII-to-CPU Unit: 8 bits	CPU-to-ASCII Unit: 32 bits ASCII-to-CPU Unit: 16 bits
	Monitoring errors generated in the ASCII Unit from the CPU Unit.	Not available	Available
	Storing data in character-string variables for such commands as PC READ and PC WRITE.	Only the first 4 characters are read or written for each character variable.	Up to 255 characters can be read or written for each character variable.

Item		C200H-ASC02	C200H-ASC11/21/31
Flags/Control Bits	Software Restart Bit	Yes	No
	Read Bit and PC Interrupt Bit	Uses the Write Bit.	The Read Bit and the PC Interrupt Bit are different.
	ASCII Busy Flag	Changes status when the PC interrupt is generated.	Does not change when the PC interrupt is generated. Instead, the PC Interrupt Fail Flag changes if the PC interrupt fails.
Interrupts	Multiple interrupts	Yes. If a XXX ON command occurs, or another interrupt is generated while executing an interrupt subroutine, the executing interrupt subroutine is stopped, and the next interrupt subroutine is branched to.	No. If another interrupt is generated while executing an interrupt subroutine, the next subroutine is executed after the first interrupt subroutine execution is completed. The interrupt will, however, stop executing immediately when an ON ERROR interrupt is generated.
		Multiple interrupts are executed in the order that they were generated.	The next interrupt subroutine is executed according to order of priority.
	Executing interrupts	Even if a BASIC interrupt occurs while a command is being executed, it will not branch into the interrupt subroutine. Branching will only occur after the command has been executed.	When a BASIC interrupt occurs while a command is being executed, the interrupt subroutine will be executed, but only once.
Processing speed	Execution time depending on destination of GOSUB and GOTO commands.	Changes according to the distance from the current line to the destination.	Same time for all destinations.
Commands	RND command	Random number generator logic is different.	
	WAIT command	Timer does not stop while interrupt subroutine is being executed.	Timer stops while interrupt subroutine is being executed.
	PRINT USING command	Display will change depending on the USING format.	
Number range	Integer range	-32768 to 32767	-2147483648 to 2147483647
	Single-precision real number range	-1.70141E + 38 to 1.70141E + 38	$\pm 1.2 \times 10^{-38}$ to $3.4 \times 10^{38}$
	Double-precision real number range	-1.701411834604692D + 38 to 1.701411834604692D + 38	$\pm 2.2 \times 10^{-308}$ to $1.79 \times 10^{308}$
	Precision of arithmetic function results.	Single-precision	Single or double-precision
Saving to/reading from ROM		Uses the LOAD command to read from EEPROM.	Reads from /writes to the flash ROM and user memory (RAM) using the ROMLOAD and ROMSAVE commands.

# Appendix C

## PC Format

### PC Format

When the ASCII Unit reads from and writes to the I/O memory data of the CPU Unit or the shared memory of the ASCII Unit itself using the following commands, it recognizes the data format of the I/O memory or shared memory. The ASCII Unit uses this data format when reading data from the I/O memory or the shared memory into the ASCII Unit variable or writing data from the ASCII Unit variable to the I/O memory or the shared memory. To allow such read-write operation, the data format must be specified. This specified data format is called the PC format.

#### Target Commands

For CPU Unit I/O Memory Data: PC READ/PC WRITE, PC READ@/ PC WRITE@, PC QREAD@/ PC QWRITE@ commands

For ASCII Unit Shared Memory Data: PC EPUT/ PC EGET commands

- In the A format, multiple PC words can correspond to one ASCII Unit variables, with a maximum of 127 words for one variable.  
For formats other than A, S, and where the variable is a character variable, there must be one variable corresponding to each word.  
In the A format, one variable corresponds to one format.  
In the S format, one array variable corresponds to one format.  
For formats other than A, S, and where the variable is a character variable, multiple variables can be allocated to one format.
- When “m” is omitted, one word is assumed.  
Words from 1 to 64 (1 to 127 for A format) and numerals from 1 to 255 can be handled at one time.  
Formats and subcommands can be specified as character variables.

### Reading Data from CPU Unit I/O Memory or ASCII Unit Shared Memory to ASCII Unit Variable

Name	Format		Meaning	Example
I (BCD) format	mIn	When variables are numerals: m = 1 to 255 n = 1 to 4  When variables are characters: m = 1 to 255 (see note 1) n = 1 to 4  The following must be true when variables are characters: m x n < 256	Data of m words is regarded as BCD data, and the data from the rightmost 1 to n digits of each word is stored in each variable.	213: Data of 2 words is regarded as BCD, and the data from the rightmost 1 to 3 digits of each word (bits 00 to 11) is stored in 2 variables.

Name	Format	Meaning	Example
H (hexadecimal) format	mHn	When variables are numerals: $m = 1$ to 255 $n = 1$ to 4 When variables are characters: $m = 1$ to 255 (see note 1) $n = 1$ to 4 The following must be true when variables are characters: $m \times n < 256$	Data of $m$ words is regarded as hexadecimal data, and the data from the rightmost 1 to $n$ digits of each word is stored in each variable.  2H3: Data of 2 words is regarded as hexadecimal, and the data from the rightmost 1 to 3 digits of each word (bits 00 to 11) is stored in 2 variables.
O (octal) format	mOn	When variables are numerals: $m = 1$ to 255 $n = 1$ to 4 When variables are characters: $m = 1$ to 255 (see note 1) $n = 1$ to 4 The following must be true when variables are characters: $m \times n < 256$	Data of $m$ words is regarded as octal data, and the data from the rightmost 1 to $n$ digits of each word is stored in each variable.  2O3: Data of 2 words is regarded as octal, and the data from the rightmost 1 to 3 digits of each word (bits 00 to 11) is stored in 2 variables.
B (bit) format	mBn	When variables are numerals: $m = 1$ to 255 $n = 0$ to 15 When variables are characters: $m = 1$ to 255 (see note 1) $n = 1$ to 15 The following must be true when variables are characters: No. of characters < 256	Data of $m$ words is regarded as binary data, and the $n$ th bit of data from each word only is converted to decimal data (other bits are regarded as 0) and stored in each numeric variable.  2B3: Data of 2 words is regarded as binary, and the 3rd bit of data from each word only is converted to decimal data (other bits are regarded as 0) and stored in each numeric variable.
A (ASCII) format	mAn	$m = 1$ to 255 (see note 2) $m = 127$ or less if $n = 1$ to 2	Data of $m$ words is regarded as ASCII data, and the data specified as leftmost or rightmost in each $n$ word stored in 1 character variable. $n = 1$ : Rightmost byte $n = 2$ : Leftmost byte $n = 3$ : Leftmost and rightmost bytes



Name	Format		Meaning	Example
S (array variable) format	SmXn Note: X is any of I,H,O or B.	m = 1 to 255 n = same as specified format	Data of m words is regarded as data in the specified format (BCD, hexadecimal, octal, or bit), and the data from each word is stored in n array variables. Character string arrays cannot be used as variables.	S2I2: Data of 2 words is regarded as BCD data and data from each word is stored in 2 array variables.

- Note**
1. For versions up to V1.02, m = 1 to 64. The version is shown on the terminal startup display.
  2. For versions up to V1.02, m = 1 to 127. The version is shown on the terminal startup display.

## Reading Data from ASCII Unit Variable to CPU Unit I/O Memory or ASCII Unit Shared Memory

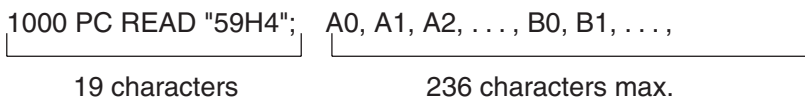
Name	Format		Meaning	Example
I (BCD) format	mIn	When variables are numerals: m = 1 to 255 n = 1 to 4 When variables are characters: m = 1 to 64 n = 1 to 4	Data from the rightmost 1 to n digits of m variables is converted to BCD data, and saved to the rightmost 1 to n digits of each word for m words.	2I3: Data from the rightmost 1 to 3 digits of 2 words is converted to BCD data, and saved to the rightmost 1 to 3 digits of each word for 2 words.
H (hexadecimal) format	mHn	When variables are numerals: m = 1 to 255 n = 1 to 4 When variables are characters: m = 1 to 64 n = 1 to 4	Data from the rightmost 1 to n digits of m variables is converted to hexadecimal data, and saved to rightmost 1 to n digits of each word for m words.	2H3: Data of the rightmost 1 to 3 digits of 2 words is converted to hexadecimal data, and stored in the rightmost 1 to 3 digits of each word for 2 words.
O (octal) format	mOn	When variables are numerals: m = 1 to 255 n = 1 to 4 When variables are characters: m = 1 to 64 n = 1 to 4	Data from the rightmost 1 to n digits of m variables is converted to octal data, and saved to the rightmost 1 to n digits of each word for m words.	2O3: Data of the rightmost 1 to 3 digits of 2 words is converted to octal data, and stored in the rightmost 1 to 3 digits of each word for 2 words.
B (bit) format	mBn	When variables are numerals: m = 1 to 255 n = 0 to 15	Each of m numeric variables is regarded as decimal, and converted to binary data, and the nth bit of data from each word only (other bits are regarded as 0) is stored to each word for m words.	2B3: Each of 2 numeric variables is regarded as decimal and converted to binary data, and the 3rd bit of data from each word (other bits are regarded as 0) only is stored in each of 2 words.

Name	Format		Meaning	Example
A (ASCII) format	mAn	m = 1 to 127 n = 1 to 3	Two $\times m$ characters from the beginning of 1 character variable are converted into ASCII data (2 characters correspond to 1 word), and is stored in the leftmost/rightmost byte position in each word specified by n.  n = 1: Rightmost byte (leftmost byte is 00 <sub>hex</sub> ) n = 2: Leftmost byte (rightmost byte is 00 <sub>hex</sub> ) n = 3: Leftmost and rightmost bytes	2A3: The first 4 characters in 1 character variable is converted to ASCII data, and stored to the leftmost and rightmost bytes of each word for 2 words.
S (array variable) format	SmXn Note: X is any of I,H,O or B.	m = 1 to 255 n = similar to each specified format.	Data of n array variables is converted to a data format specified by X (any of BCD, hexadecimal, octal, or bit) and stored to each of m words. Character string arrays cannot be used as variables.	S2I2: Data from 2 array variables is converted to hexadecimal and stored in each of 2 words.

Up to 255 words can be transferred at one time. For integer variables, however, only a maximum of 50 to 60 words can be transferred at one time. The following restrictions apply:

- A maximum of 255 characters can be written in one line.
- One variable must correspond to each word for integer variables.

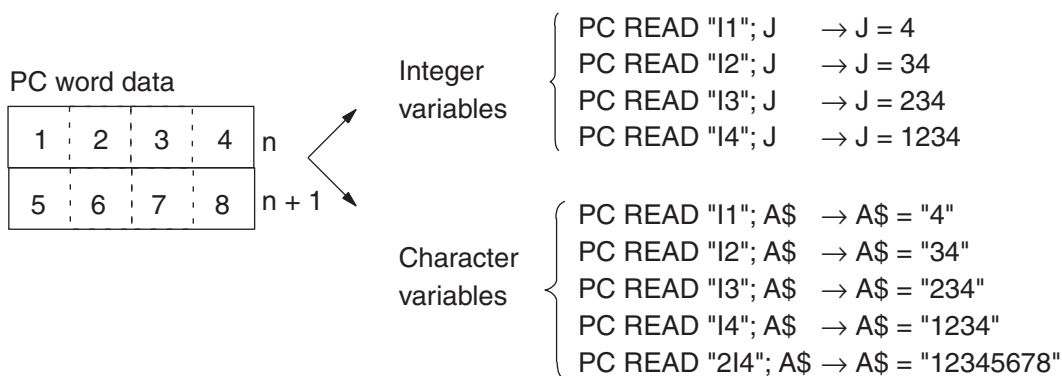
**Example:** H format



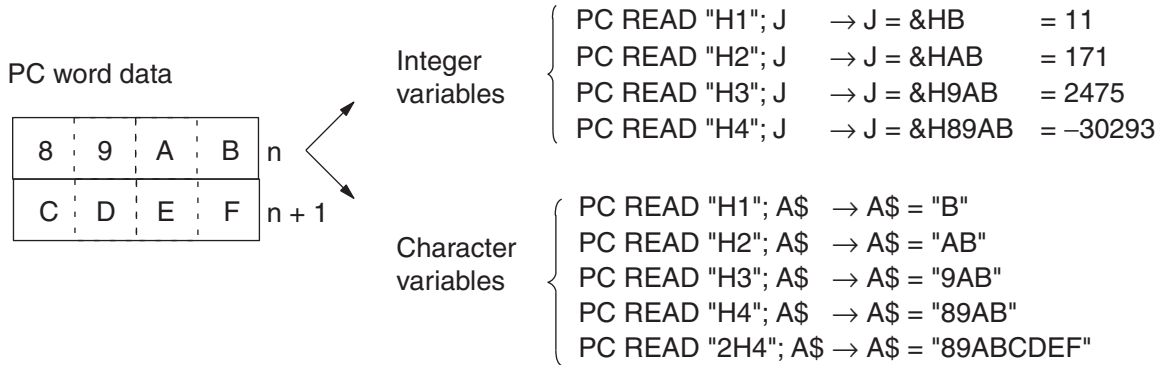
Up to 79 variables using 3 characters per variable ( $236 \div 3 = 79$ ) can be written. Therefore, a maximum of 59 words can be transferred at one time.

## PC READ Format Conversion Examples

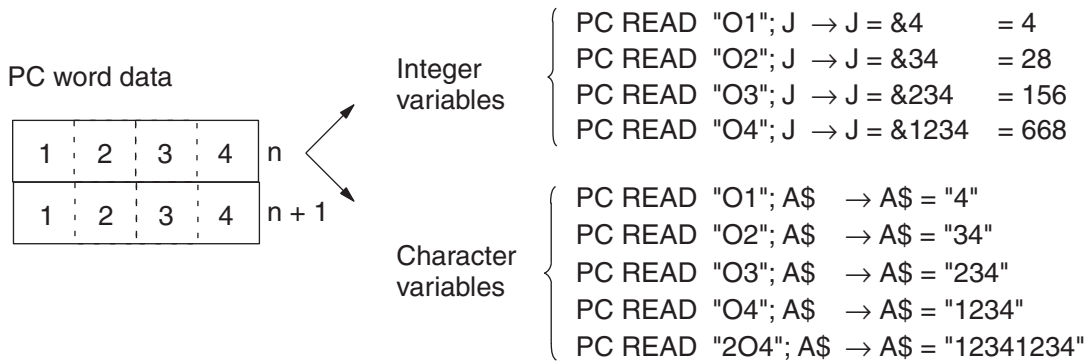
### I Type Format



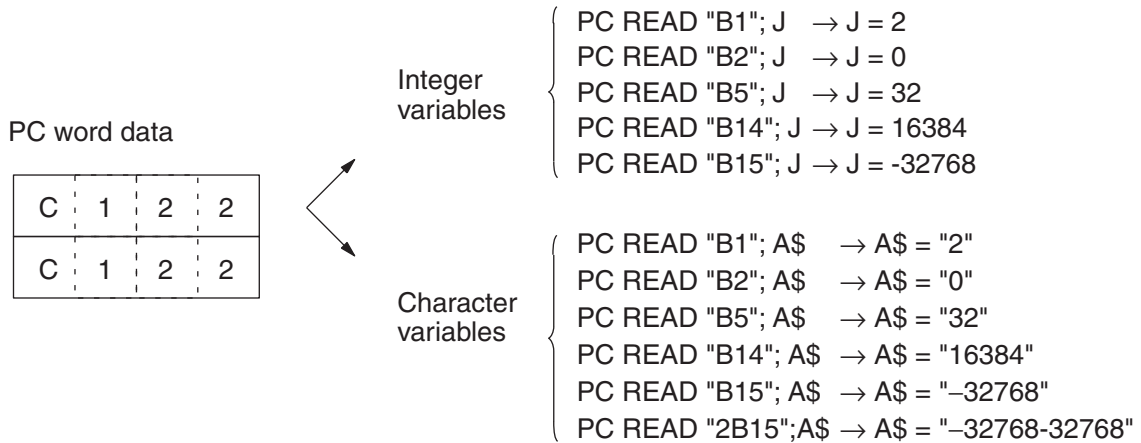
### H Type Format



### O Type Format



### B Type Format



### A Type Format

If the integer variable type does not match, an error will be generated.

PC word data

5	1	5	2
5	3	5	4

Character variables

{ PC READ "2A1"; A\$ → A\$ = "RT"  
 PC READ "2A2"; A\$ → A\$ = "QS"  
 PC READ "2A3"; A\$ → A\$ = "QRST"

{ Q: &H51  
 R: &H52  
 S: &H53  
 T: &H54

### S Type Format

PC word data

0	1	2	3
4	5	6	7
8	9	0	1
2	3	4	5

Integer variables  
(In I type format)

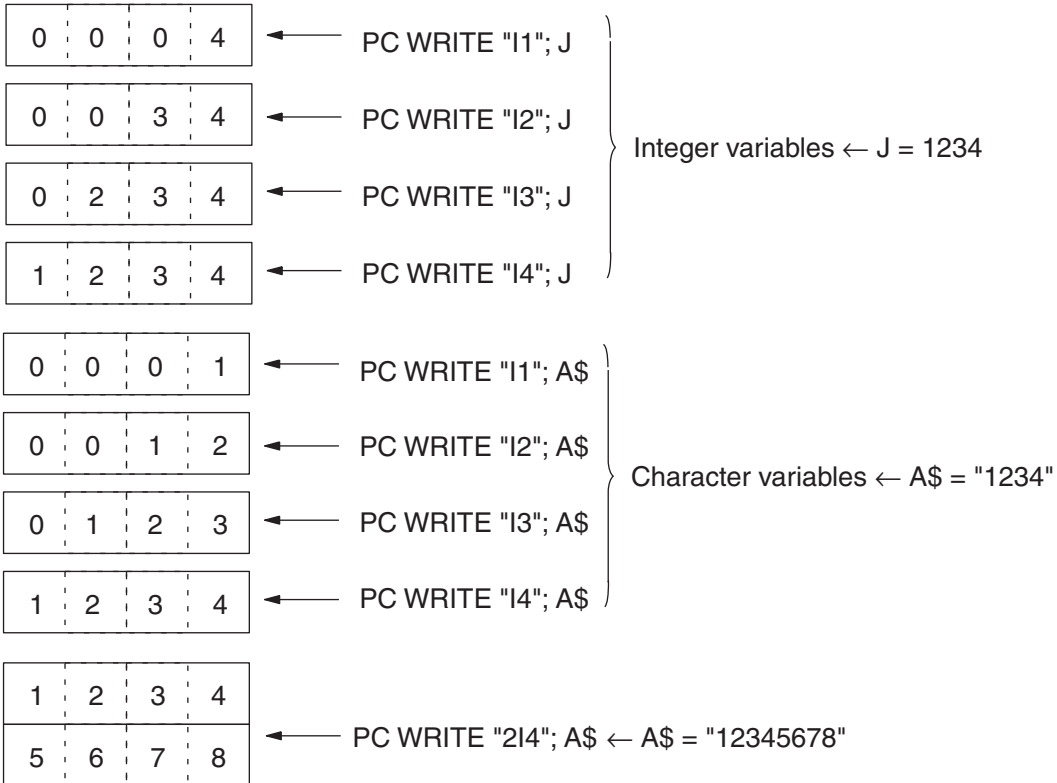
PC READ "S4I4"; A (1) → A (1) = 123  
 → A (2) = 4567  
 → A (3) = 8901  
 → A (4) = 2345

Note: Character-string array variables cannot be used.

# Examples of PC WRITE Format Conversions

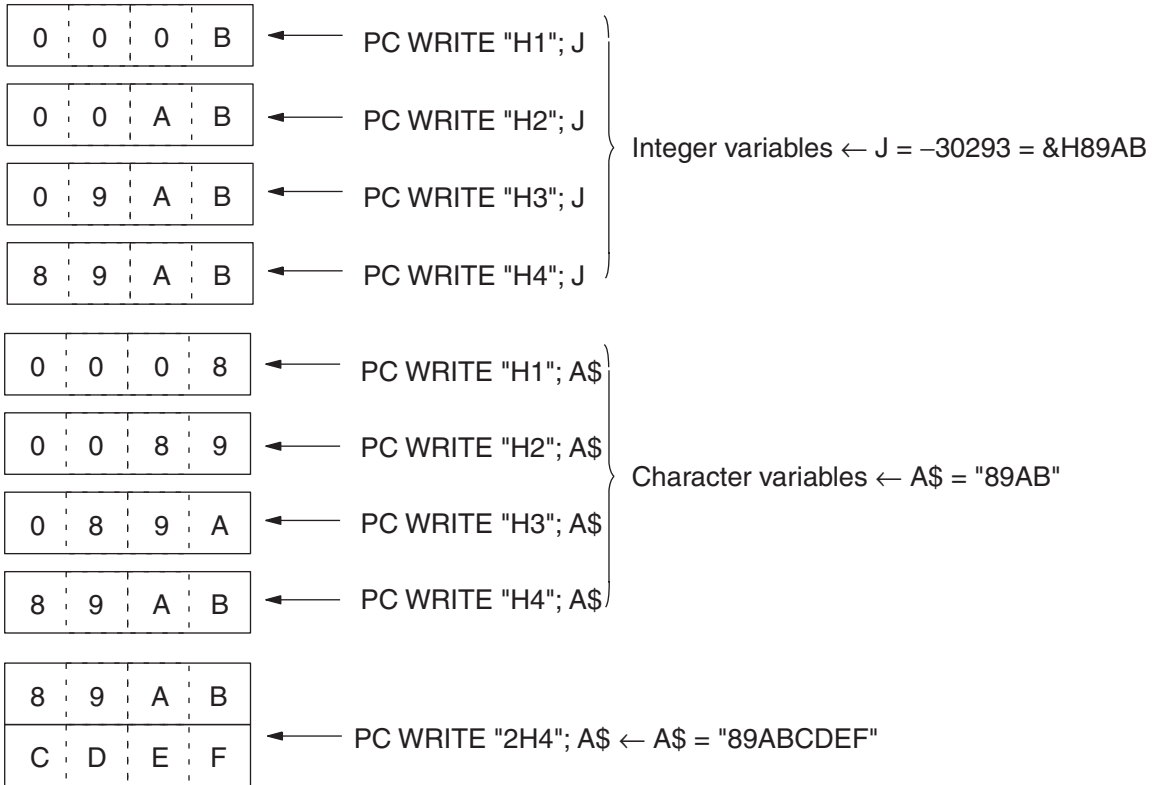
## I Type Format

PC word data



### H Type Format

PC word data

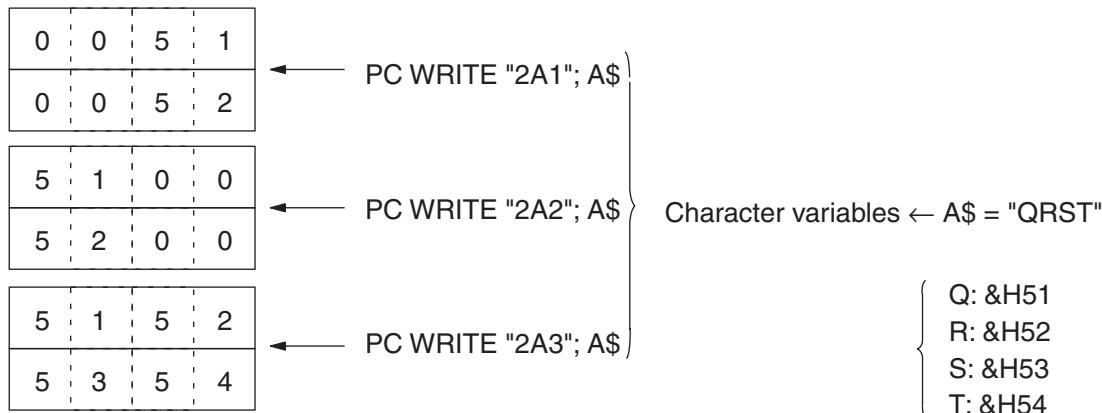




### A Type Format

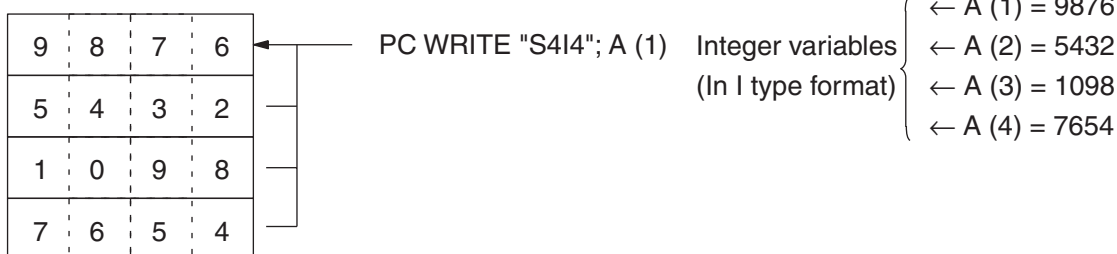
If the integer variable type does not match, an error will be generated.

PC word data



### S Type Format

PC word data



- Note** 1. Numeric variables and character variables can be used together for PC READ and PC WRITE commands if the character variable is at the end of the format as follows:  
 Example: PC READ "@D,0,2,H4,H4"; B, A\$  
 If the command format is written as PC READ "@D,0,2,H4,H4"; A\$, B, a FORMAT ERROR will be generated. The above combination cannot be used if array variables have been used as variables for PC WRITE.
2. Character-string array variables cannot be used with the S-type format. Use the following format to read or write character data.  
 Example: PC READ "@D,0,2,10,10H4"; A\$
3. Only one character-string variable can be used in the variable list, so multiple strings are not possible.
4. If using "mA1, mA2" when writing ASCII variables in A-type format to I/O memory in the CPU Unit, provide 2 x m characters in the character variable.

```
Example 1: PC WRITE "@D,0,4,4A,4A1";("1234"+"1234")
                                     |
                                     Dummy data
Example 2: A$= 1234"
           PC WRITE "@D,0,4A,4A1";(A$+A$)
```

5. Negative values cannot be written using the I-type format for PC WRITE. A format error will occur.

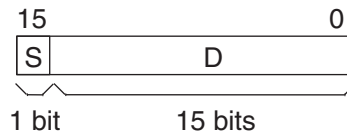


# Appendix D

## Formats for Storing Variables in Memory

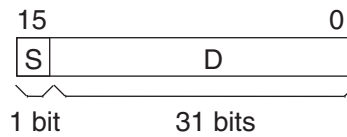
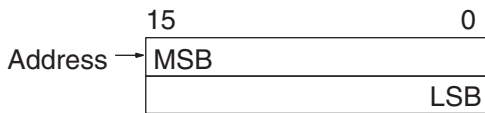
Variables are stored to memory in the following ways, depending on the variable type.

### • Short Integer Variables



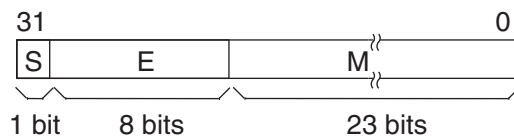
S: Sign (0: positive, 1: negative)  
 D: Numeric value

### • Long Integer Variables



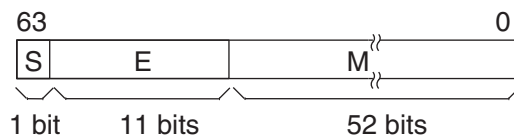
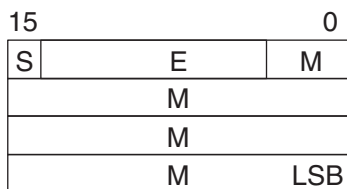
S: Sign (0: positive, 1: negative)  
 D: Numeric value

### • Single-precision Real Number Variables



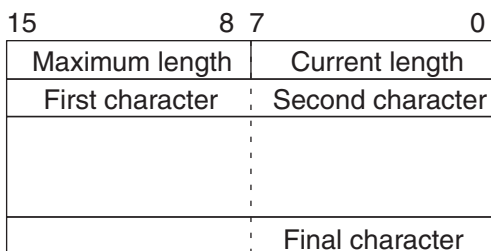
S: Sign (0: positive, 1: negative)  
 E: Exponent (offset: 127)  
 M: Mantissa

### • Double-precision Real Number Variable



S: Sign (0: positive, 1: negative)  
 E: Exponent (offset: 1023)  
 M: Mantissa

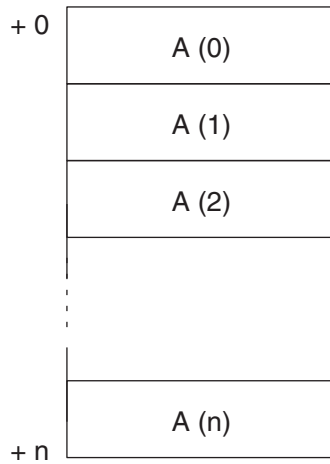
### • Character String Variable



- Add empty bytes and adjust so that there will be an even number of memory bytes for each variable.
- The character data section not being used beyond the current length contains undefined data.

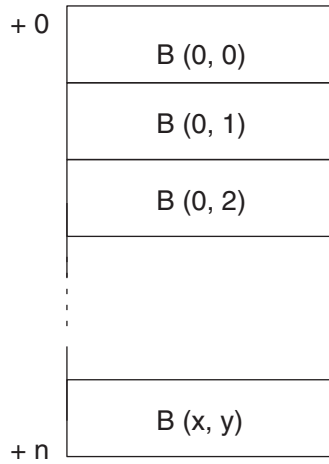
• **Array Variables**

For one-dimensional arrays



- As an element of the array, each variable is stored in memory in order.

For two-dimensional arrays



- For multi-dimensional arrays, the rightmost subscript is stored in memory, in the order of alteration.

## VARPTR (Variable Name)

### Function

Assigns the memory address where the variable value is stored.

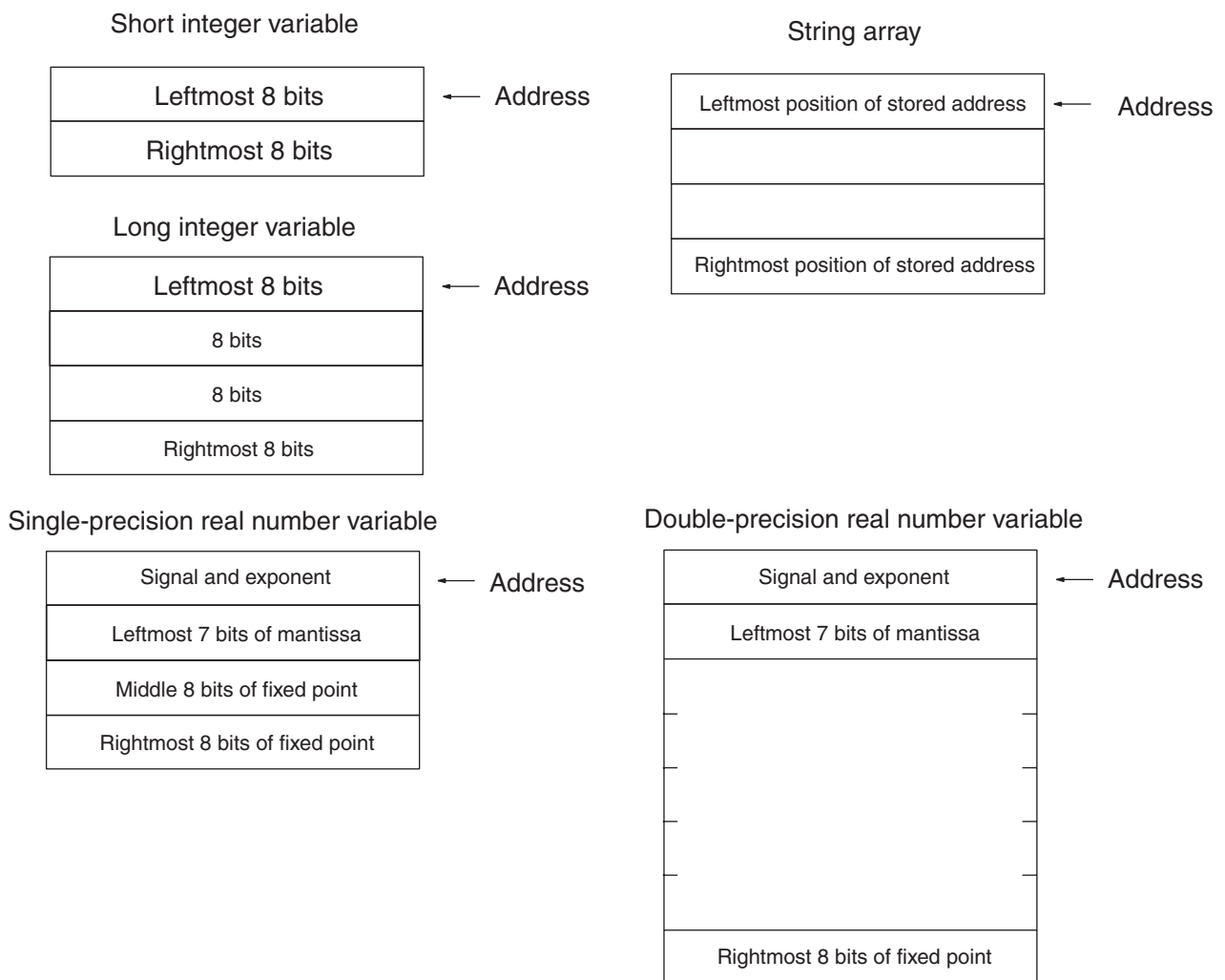
### Notation Example

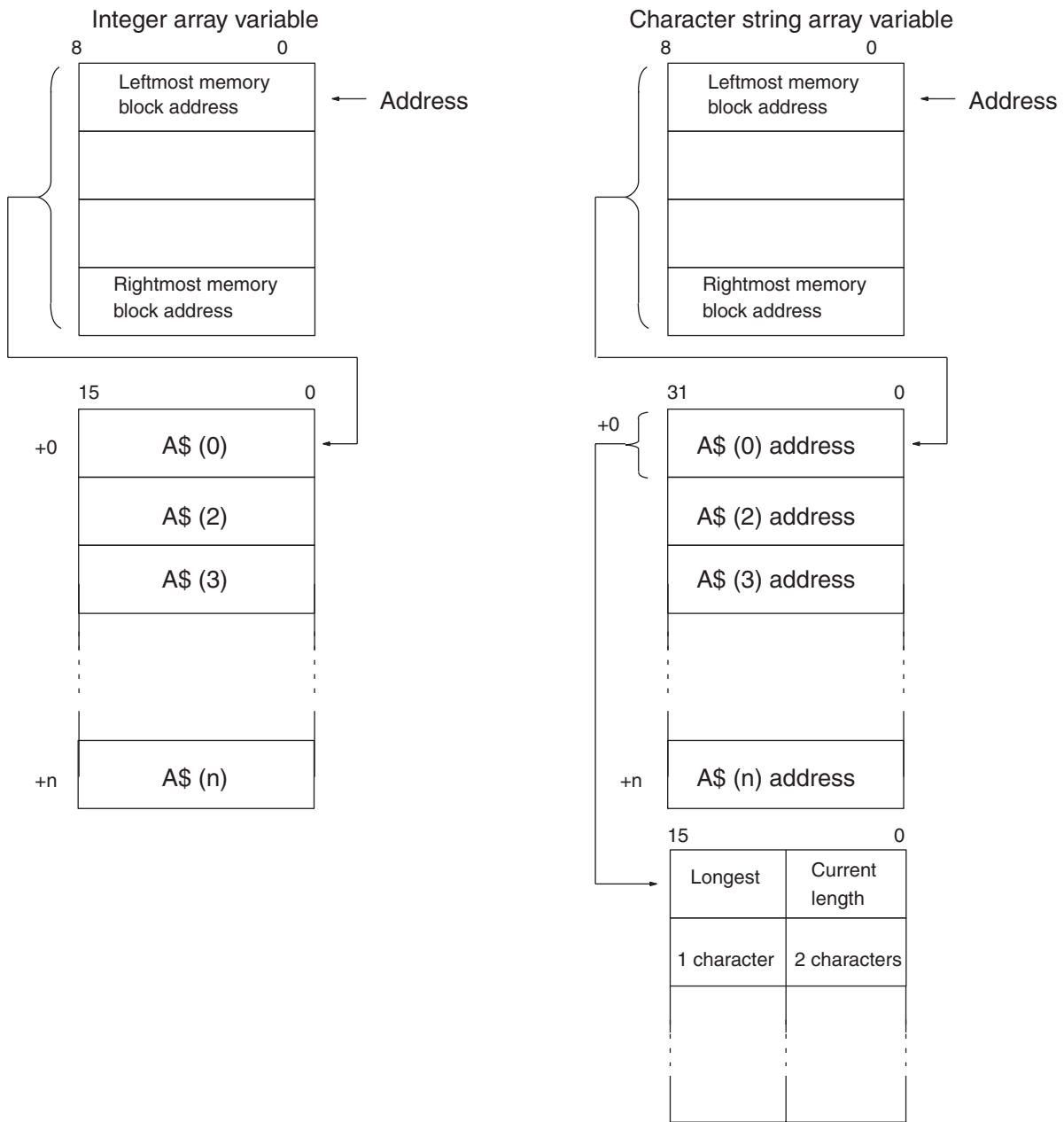
B& = VARPTR (A)

### Details

- Assigns the memory address to the area where variable data specified for the variable name is stored.
- Shows the address of the pointer to where character data is stored for character variables.

### Relationship between Variable Type and VARPTR Address





# Appendix E

## Command Execution Time Samples

### Assignments

Assignment	Execution time (ms)
A=1	0.19
A%=1	0.14
A=A + 1	0.41
A=A - 1	0.41
A=A*1	0.39
A=A/1	0.41
A=A\1	0.36
A=A MOD 2	0.35
A=A^ 2	0.52
A%=NOT A%	0.20
A%=A% AND 1	0.28
A%=A% OR 1	0.28
A%=A% XOR 1	0.28
A%=A% EQV 1	0.28
A%=A% IMP 1	0.28

### Statements

Statement	Execution time (ms)
@IF A=0 THEN B=0 ELSE B=1 ENDIF (true and false)	0.45
BITON C%, 0 / BITOFF C%, 0	0.12
CLOSE	0.60
FOR I=1 TO 10000: NEXT I (for one loop)	0.14
FOR I%=1 TO 10000: NEXT I% (for one loop)	0.07
GOSUB *TEST* TEST: RETURN	0.13
GOTO	0.03
IF A=0 THEN B=0 ELSE B=1 (true and false)	0.45
INPUT #2, A\$ 200 characters	Approx. 20.0
LET	0.19
LINE INPUT #2, A\$ 200 characters	Approx. 30.0
OPEN #2, "COMU:"	1.06
PC EGET #0, 90, "S90I4"; A (0)	7.8
PC EPUT #0, 90, "S90I4"; A (0)	8.4
PC GET I, J	0.34
PC PUT I	0.17
PC QREAD "@D,0,10,10I4"; A,B,C,D,E,F,G,H,I,J (See note 2.)	11.5
PC QWRITE "@D,0,10,10I4"; A,B,C,D,E,F,G,H,I,J (See note 2.)	8.8
PC READ "I4"; A(See note 1.)	7.5
PC READ "5I4"; A,B,C,D,E(See note 1.)	7.6
PC READ "10I4"; A,B,C,D,E,F,G,H,I,J (See note 1.)	7.7
PC READ "63H4"; A\$ (See note 1.)	41.9

Statement	Execution time (ms)
PC WRITE "I4"; A (See note 1.)	3.9
PC WRITE "5I4"; A,B,C,D,E (See note 1.)	4.2
PC WRITE "10I4"; A,B,C,D,E,F,G,H,I,J (See note 1.)	7.7
PC WRITE "63H4"; A\$ (See note 1.)	20.7
PC READ "@D,0,1,I4" A (See note 1.)	5.8
PC READ "@D,0,5,5I4"; A,B,C,D,E (See note 1.)	5.9
PC READ "@D,0,10,10I4"; A,B,C,D,E,F,G,H,I,J (See note 1.)	7.8
PC READ "@D,0,63,63H4"; A\$ (See note 1.)	32.7
PC READ "@D,0,63,63H4"; A\$ ( $\alpha$ mode) (See note 1.)	30.8
PC WRITE "@D,0,1,I4" A (See note 1.)	4.0
PC WRITE "@D,0,5,5I4"; A,B,C,D,E (See note 1.)	4.0
PC WRITE "@D,0,10,10I4"; A,B,C,D,E,F,G,H,I,J (See note 1.)	7.7
PC WRITE "@D,0,63,63H4"; A\$ (See note 1.)	20.7
PC WRITE "@D,0,63,63H4"; A\$ ( $\alpha$ mode) (See note 1.)	9.0
PRINT A\$ (9,600 bps, 255 characters)	Normal mode: Approx. 300 DMA mode: Approx. 5
REM	0.03
WHILE A < 100 : A = A + 1 : WEND (for one loop)	0.67

- Note 1.** The execution time includes the time required for handshaking with the PC. Scan time is usually 2 ms (approx.).
- Transfer of less than 20 words:  
 $2 \times \text{PC scan times max.}$   
 @:  $1 \times \text{PC scan time max.}$
- Transfer of more than 20 words (C200H Mode):  
 $\text{INT} ((\text{No. of transfer words} - 1)/20) + 2 \times \text{PC scan times max.}$   
 @:  $\text{INT} ((\text{No. of transfer words} - 1)/20) + 1 \times \text{PC scan time max.}$
- Transfer of more than 127 words (C200H-HX/HG/HEPC Mode):  
 $\text{INT} ((\text{No. of transfer words} - 1)/127) + 2 \times \text{PC scan times max.}$   
 @:  $\text{INT} ((\text{No. of transfer words} - 1)/127) + 1 \times \text{PC scan time max.}$
2. The execution time includes the time required for handshaking (IORD or IOWR execution) with the PC.

## Functions

Function	Execution time (ms)
A=ABS(-1.5)	0.22
A=ACOS(0.5)	1.72
A%=ASC("A")	0.16
A=ASIN(0.5)	1.7
A=ATN(0.5)	0.74
B#=CDBL(1)	4.7
D%=CHR\$(&H41)	0.45
C%=CINT(0.5)	0.24
A=COS(0.5)	0.92
A=CSNG(0.1#)	0.22
A=EXP(5)	1.05

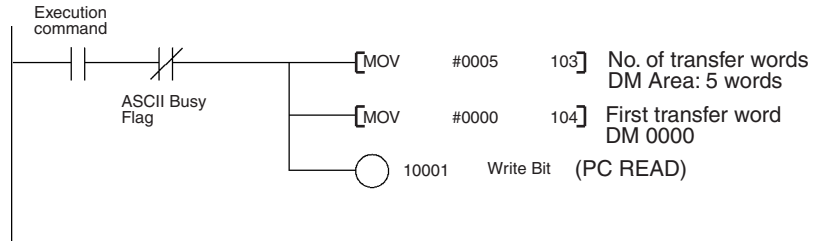
Function	Execution time (ms)
A\$=STRING\$(250,"A") B\$=FCS(A\$,1) Horizontal parity	1.0
C%=FIX(1.1)	0.26
D\$=HEX\$(1)	0.5
INPUT\$(100,#2)	3.0
C%=INT(1.1)	0.25
D\$=LEFT\$("ASCII",2)	0.51
A%=LEN("ASCII")	0.17
A%=LOC(2)	0.22
A=LOG(5)	1.0
D\$=MID\$("ASCII",1,1)	0.62
D\$=OCT\$(&051)	0.54
D\$=RIGHT\$("ASCII",2)	0.5
A=SIN(0.5)	0.73
A=SQR(2)	0.66
D\$=STR\$(1)	0.64
A=TAN(3)	1.01
A%=VAL("A")	0.2

# Appendix F

## Sample Programs

### 1) Transferring PC Data from the ASCII Unit

#### Ladder Program



#### Basic Program

100 PC WRITE "514"; A,B,C,D,E

The 5 words of data starting with the first transfer word DM 0000 are read as BCD data and stored (read) in the 5 variables A, B, C, D, E.

### 2) Transferring PC Data from the PC to the ASCII Unit

#### Ladder Program

A ladder program is not required.

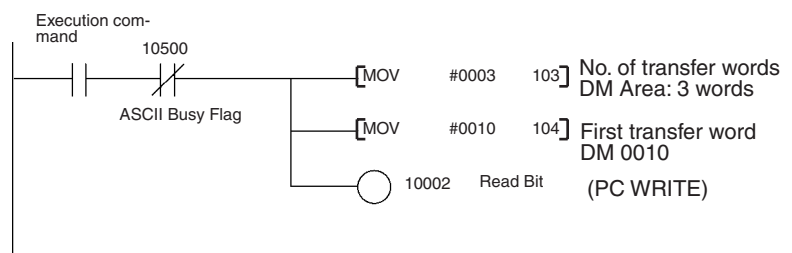
#### Basic Program

100 PC READ "@D,0,5,5I4"; A,B,C,D,E

Five words of data starting with DM 0000 from the CPU Unit are read as BCD data and each fourth digit (16-bit) of data are stored (read) in the 5 variables A, B, C, D, E.

### 3) Transferring ASCII Unit Data from the ASCII Unit to the PC

#### Ladder Program



#### Basic Program

100 PC WRITE "314"; A,B,C

The 3 variables A, B, and C (4-digit data) is converted to BCD data and transferred (written) to the corresponding 4 digits (16 bits) of 3 words in the CPU Unit, beginning with the above first transfer word (DM 0010).



## 4) Transferring ASCII Unit Data from the ASCII Unit to the PC

### Ladder Program

A ladder program is not required.

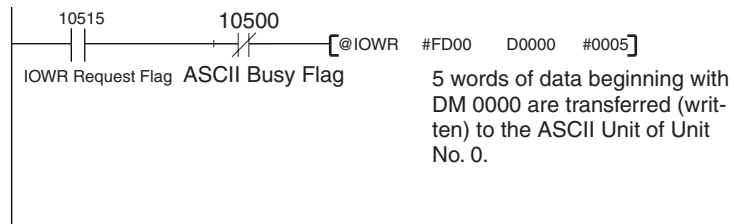
### Basic Program

100 PC WRITE "@D,10,3,3I4"; A,B,C

The 3 variables A, B, and C (4-digit data) is converted to BCD data and transferred (written) to the corresponding 4 digits (16 bits) of 3 words in the CPU Unit, beginning with the first transfer word (DM 0010).

## 5) Transferring Data Asynchronously from PC to ASCII Unit: C200HX/HG/HE PCs Only

### Ladder Program



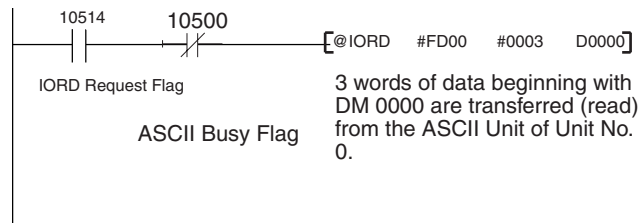
### Basic Program

100 PC QREAD "@D,0,5,5I4"; A,B,C,D,E

The 5 words of data that were transferred using the IOWR(FD00) command are converted to BCD and stored in the 5 variables A, B, C, D, E as 4-digit (16-bit) data.

## 6) Transferring Data Asynchronously from ASCII Unit to PC: C200HX/HG/HE PCs Only

### Ladder Program



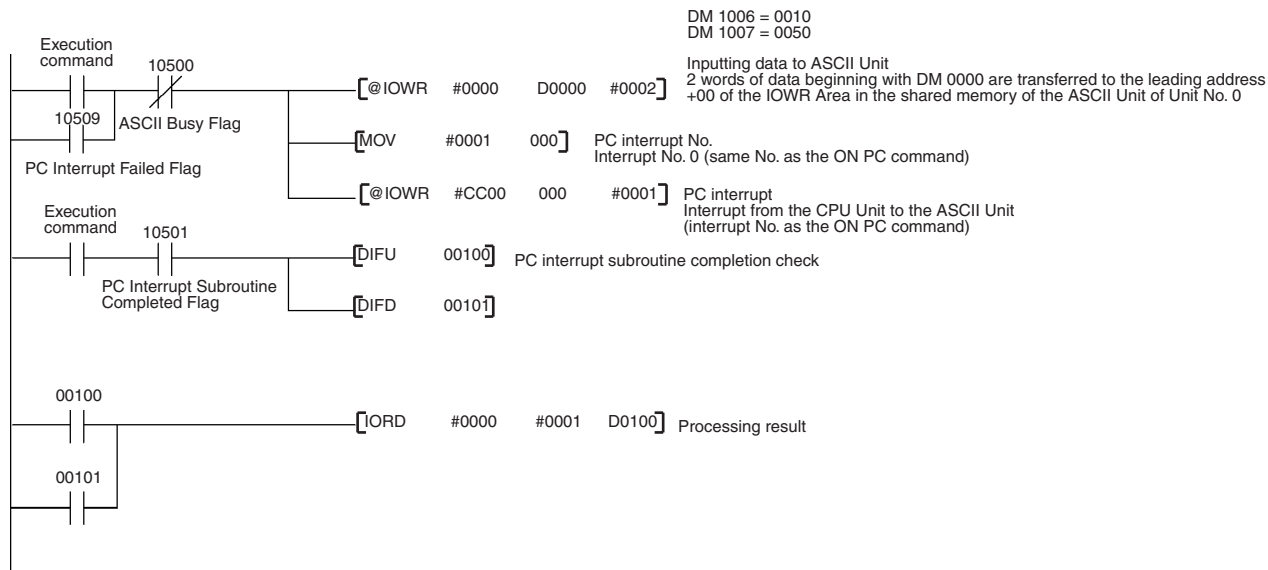
### Basic Program

100 PC QWRITE "@D,0,3,3I4"; A,B,C

The 3 variables A, B, and C (4-digit data) is converted to BCD data and is ready for transferring to the 3 words in the CPU Unit, beginning with DM 0000.

# ASCII Unit Processing as PC Coprocessor

## Ladder Program



## Basic Program

```

100 ON PC 1 GOSUB *PINT
110 PC ON
:
:
200 *PINT
210 PC EGET #0,2,"214"; A,B
220 C%=A+B
230 PC EPUT #0,1,"14"; C%
240 RETURN
    
```

# Appendix G

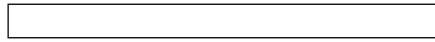
## Wiring RS-232C or RS-422A/485 Cable Connectors

### Cable Processing (End connected to FG)

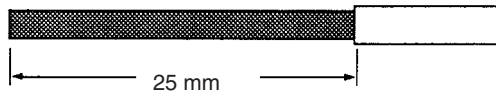
See the diagrams for the lengths required in each step.

1,2,3...

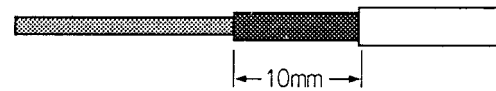
1. Cut the cable to the required length.



2. Peel the sheath using a razor blade without damaging the shield weaving.



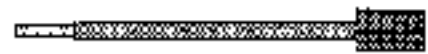
3. Remove the shield using scissors.



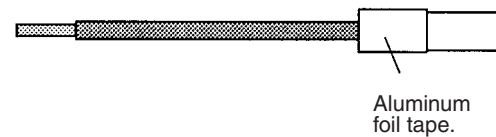
4. Peel the core wire of each wire using a stripper.



5. Fold back the shield wire.



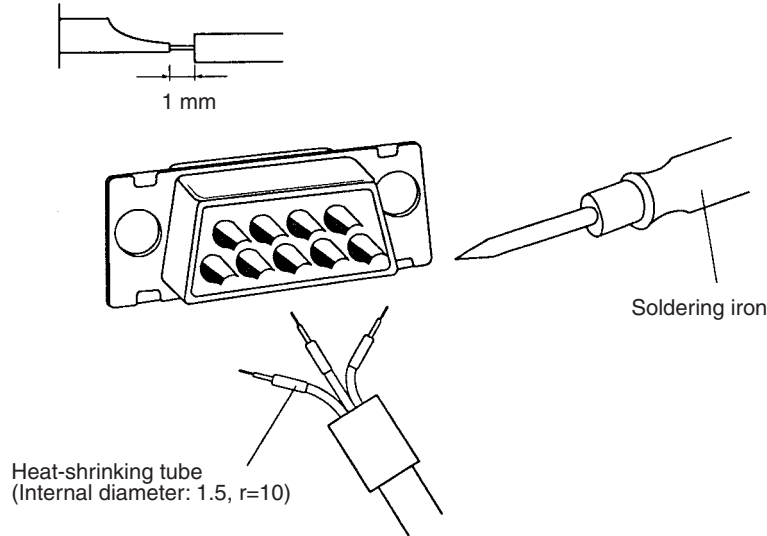
6. Wrap aluminum foil tape on top of the folded shield.



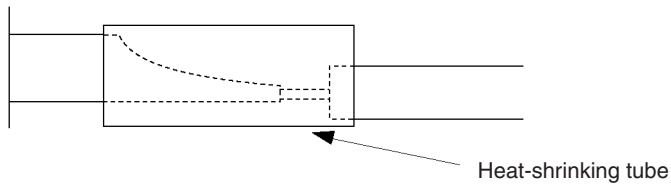
**Soldering**

Solder as described next.

1. Place a heat-shrinking tube around each wire.
2. Presolder each wire and to its connector pin.
3. Solder each wire firmly in place.

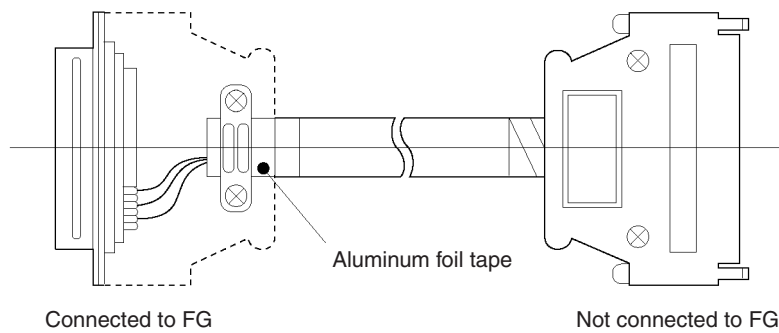


4. Move the heat-shrinking tube to the soldered section and shrink the tube by heating it.



**Assembling Hood**

Assemble the connector hood as shown below.



# Appendix H

## Reserved Words

Reserved word				
@IF	GO	RENUM	EQV	INTRS
ALARM	HELP	RESET	IMP	INTRR
ALL	IF	RESTORE	MOD	LEFT\$
AUTO	INPUT	RESUME	NOT	LEN
BASE	KEY	RETURN	OR	LOC
BITOFF	LENGTH	ROMLOAD	XOR	LOG
BITON	LET		ABS	MID\$
BRKPT	LIB	ROMSAVE	ACOS	MODEL
CLEAR	LINE	ROMVERIFY	ASC	OCT\$
CLOSE	LIST	RTS	ASIN	PEEK
CLS	LLIST	RUN	ATN	RIGHT\$
COM	LOAD	SAVE	CDBL	RND
CONT	LPRINT	SET	CHR\$	SEARCH
DATA	MAP	STEP	CINT	SGN
DEF	NEW	STOP	CLNG	SIN
DEFDBL	NEXT	SUB	COS	SPACE\$
DEFINT	OFF	SYSBACK	CSNG	SPC
DEFLNG	ON	SYSLOAD	CTS	SQR
DEFSNG	OPEN	SYSMOVE	DATE\$	SSTACK
DEFSTR	OPTION	THEN	DAY	STR\$
DEL	PC	TIMER	DMA	STRING\$
DIM	PGEN	TO	DSR	TAB
DTR	PINF	TROFF	EOF	TAN
EDIT	PMEM	TRON	ERL	TIME\$
EGET	PNAME	TRACE	ERR	TIME
ELSE	POKE	USING	EXP	STRING
END	PRG	VAL	FCS	STACK
ENDIF	PRINT	VARPTR	FIX	WDT
EPUT	PUT	WAIT	FRE	BYTE
ERASE	PWORD	WATCH	HEX\$	CODE
ERC	QREAD	WEND	INKEY\$	HEAD
ERROR	QWRITE	WHILE	INPUT\$	TERM
FN	RANDOM	WRITE	INSTR	
FOR	READ		INT	
GET	REM	AND	INTRB	

# Index

## Numerics

2-wire/4-wire switch  
(WIRE), 21

## A

arithmetic expressions, 127  
ASCII communications  
with Peripheral Devices, 2  
asynchronous data exchanges, 236  
auto start (5A)  
and START/STOP switch, 20  
Automatic transfer setting, 40

## B

Bar Code Reader  
connecting, 26  
BASIC  
commands, 121  
functions, 121  
statements, 121  
BASIC commands  
execution times, 279  
listed alphabetically, 129  
BASIC errors, 243  
BASIC functions  
user-defined functions, 231  
BASIC programs  
continuing, 120  
debugging, 232  
editing, 12, 112  
inputting programs, 114  
processing examples, 61  
programming, 14  
reading from flash ROM, 116  
sample programs, 283  
saving to flash ROM, 116  
starting, 120  
starting and stopping, 20  
stopping, 120  
transferring, 115, 117  
battery  
replacement, 256  
bit processing, 62  
breakpoint function (BRKPT), 232  
BRKPT command, 232

## C

C200H-ASC02  
comparison of functions and operation, 263

cable wiring, 287  
cables  
processing, 287  
soldering, 288  
character constant  
definition, 123  
character expressions, 127  
character strings  
processing, 62  
character variables  
allocating space, 118  
clearing error messages, 241  
clearing errors  
with the IOWR instruction, 255  
Command Mode, 121  
commands  
BASIC commands listed alphabetically, 129  
execution times for BASIC commands, 279  
comments  
inputting in BASIC programs, 122  
communications  
communications interrupts, 64  
connections, 24  
multiport communications, 2  
with Peripheral Devices, 2, 4  
communications cables  
wiring, 287  
communications errors, 59  
Communications interrupts, 64  
communications port  
closing, 53  
opening, 44  
connections, 24  
connectors  
hood, 288  
constants  
in BASIC programs, 123  
control bits  
Error Code Read Bit, 33  
PC Interrupt Bit, 33, 90  
Read Bit, 33  
Special I/O Unit Restart Bits, 253  
Sync Time Set Bit, 33  
Write Bit, 33  
CPU Unit  
exchanging data, 69  
CPU Units  
compatible CPU Units, 5

## D

data exchange

- applications, 235
- ASCII Unit to CPU Unit, 239
- asynchronous processing, 236
- bit data exchanges, 240
- from CPU Unit through shared memory, 85
- high-speed, 237
- high-volume, 239
- receiving data, 51
- selecting data exchange method, 74
- sending data, 47
- timing charts, 87
- to CPU Unit through shared memory, 86
- with external devices, 43
- with PC EGET/PC EPUT commands, 80
- with PC PUT/PC GET commands, 79
- with PC QREAD/PC QWRITE commands, 82
- with PC READ/PC WRITE commands, 75
- with PC READ@/PC WRITE@ commands, 78
- with the CPU Unit, 69

debugging BASIC programs, 232

device symbol, 46

Digital Operator

- connecting, 27

DM Area allocations, 38

DM setting errors, 247

DMA data transmission, 48

## E

EDIT command

- precautions, 259

ERC command, 59

ERR function, 60

Error Code and Type, 37

error codes

- errors listed by error code, 243

error indicators

- CPU error indicators, 252

error log

- reading and clearing, 254

errors

- BASIC errors, 243
- clearing error messages, 241
- clearing with IOWR instruction, 101
- clearing with the IOWR instruction, 255
- communications errors, 59
- DM setting errors, 247
- execution errors, 246
- operational errors, 244
- PC Interface errors, 247

execution errors, 246

execution times

- for BASIC commands, 279

- expressions
  - definition, 127
- external devices
  - connecting, 14
  - exchanging data, 43

## F-H

flags

- ASCII Busy Flag, 36
- BASIC RUN Flag, 36
- Battery Error Flag, 36
- Interrupt Fail Flag, 95
- IORD Request Flag, 36
- IOWR Request Flag, 36
- PC Interrupt Fail Flag, 36
- PC Interrupt Subroutine Completed Flag, 36
- Port 1 Error Flag, 36
- Port 2 Error Flag, 36
- Special I/O Unit error flags, 253
- System BASIC Error Flag, 36
- Terminal Port Error Flag, 36

floating point constants

- definition, 124

high-speed data exchanges, 237

high-volume data exchanges, 239

## I

ID Controller

- connecting, 27

indicators, 19

- troubleshooting errors, 248

INPUT# command, 51

INPUT\$ function, 52

inputs

- from the ASCII Unit to the CPU Unit, 35
- Input Data, 37

installation, 22

integer constants

- definition, 123

internal configuration, 5

internal layout, 7

interrupt functions, 64

- interrupt failure, 95

interrupt priority, 65

interrupts

- sending interrupts from the CPU Unit, 72

IORD Area

- first word setting, 41

IORD instruction

- reading ASCII Unit variables, 102
- reading from shared memory, 104

IORD/IOWR Area transfer setting, 40  
IORD/IOWR Areas, 39  
IOWR Area  
  first word setting, 41  
IOWR instruction  
  clearing errors in ASCII Unit, 101  
  sending interrupts, 100  
  writing ASCII Unit variables, 96  
  writing to shared memory, 97  
IOWR/IORD instructions  
  specifications, 95  
IR Area allocations, 30

## L

labels  
  definition, 122  
LED indicators, 19  
  troubleshooting errors, 248  
line number  
  definition, 121  
logical expressions, 128  
loop processing, 67

## M–N

machine language programs, 257  
maintenance, 255  
manual start (00)  
  and START/STOP switch, 20  
modes  
  command execution modes, 121  
multiport communications, 2  
nomenclature, 18

## O

ON ERROR command, 59  
OPEN# command, 44  
operation  
  basic operating procedures, 12  
operational errors, 244  
operators, 127  
OPTION LENGTH command, 118  
outputs  
  from the CPU Unit to the ASCII Unit, 32  
  Output Data, 33

## P

PC format, 265

PC Interface errors, 247  
Peripheral Device  
  general connections, 26  
  specifying with device symbol, 46  
Peripheral Devices  
  ASCII communications with, 2  
personal computer  
  use as a terminal, 26  
Port #1 baud rate setting, 41  
Port #1 DMA transmission setting, 41  
Port #2 baud rate setting, 41  
Port #2 DMA transmission setting, 41  
precautions  
  for Terminal operations, 259  
  operating precautions, 257  
  when changing from C200H-ASC02, 257  
  with PC READ/PC WRITE, 258  
PRINT# command, 47  
program memory  
  dealing with lost/damaged memory, 251  
Program Mode, 121  
program number setting, 40

## R

receive buffer processing, 62  
receiving data, 51  
relational expressions, 128  
Restart Bits  
  Special I/O Unit Restart Bits, 253  
RS-232C connections, 24  
RS-422A/485 connections, 25

## S

sample programs, 283  
sending data, 47  
settings  
  in the Setup Area, 40  
Setup Area, 39  
soldering, 288  
Special I/O Unit Area  
  word allocation, 30, 31  
specifications, 7  
  for IORD and IOWR, 95  
  functional and performance, 7  
  general, 7  
START/STOP switch, 20  
Startup mode setting, 40  
statements  
  definition, 122



STEP command, 233  
step function (STEP), 233  
SYSMAC BUS Slave Racks  
  restrictions on, 5  
system configuration, 3

## **T**

Terminal  
  transferring BASIC programs, 117  
terminal  
  using personal computer as a terminal, 26  
Terminal emulation setting, 41  
Terminal port #3 baud rate setting, 41  
Terminating Resistance switch  
  (TERM), 21  
time processing, 63  
timing charts  
  for data exchanges, 87  
  for transmission control signals, 53  
TRACE command, 234  
trace function (TRON and TRACE), 234  
transmission control signals  
  controlling, 46  
  timing charts, 53  
  when receiving data, 51

  when sending data, 48  
TRON command, 234  
troubleshooting, 248  
troubleshooting BASIC programs, 232  
troubleshooting error messages, 241  
troubleshooting flowchart, 250

## **U**

unit number switch  
  (MACHINE No.), 21  
user memory  
  DM Area allocations, 38  
  formatting, 21  
  IR Area allocations, 30  
User Memory Default Switch, 21

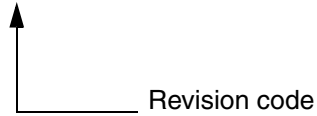
## **V-W**

variable arrays, 125  
variable monitor function (WATCH), 233  
variables, 125  
  storage format, 275  
WATCH command, 233  
wiring communications cables, 287

## Revision History

A manual revision code appears as a suffix to the catalog number on the front cover of the manual.

Cat. No. W306-E1-3



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	September 1998	Original production
2	July 1999	Additions made for CS1 Series.
3	June 2000	<p><b>Pages 5 and 23:</b> C200H-CPU02-E added to the table.</p> <p><b>Page 20:</b> C200H-CPU01-E/02-E/03-E added to the table.</p> <p><b>Page 37:</b> EM Area bank number added to word n+8.</p> <p><b>Page 109:</b> Section 6-6-1 Using the Sync Time Set Bit added.</p> <p><b>Page 114:</b> Description on stopping an open program added to the end of Connecting the Terminal.</p> <p><b>Page 115:</b> Description on T/R indicator added to the end of step 5.</p> <p><b>Page 116:</b> Cautionary information on power supply changed.</p> <p><b>Pages 191 and 193:</b> Table for C200H/HS/HE/HG/HX changed to one for C200HE/HG/HX.</p> <p><b>Page 216:</b> Information added to the end of Remarks.</p> <p><b>Page 233:</b> Description on DEVICE UNAVAILABLE IN PORT X error added to Trace.</p> <p><b>Page 254:</b> Precautionary note on restarting added to Special I/O Unit Restart Bits.</p> <p><b>Page 260:</b> Description on program startup time and power interruptions/restart added to Executing BASIC Interrupts during Command Execution.</p> <p><b>Page 274:</b> Notes 4 and 5 added.</p> <p><b>Page 288:</b> List of reserved words added as Appendix H.</p>