MITSUBISHI

PROGRAMMABLE CONTROLLER

MELSEC-A

User's Manual

Programmable Controller Option Card type A7BDE-A3N-PT32S3

CATALOG
$ 10.00 # JUM-253

MITSUBISHI ELECTRIC

REVISIONS

※The manual number is given on the bottom left of the back cover.

Print Date	※Manual Number	Revision
Jun., 1990	IB (NA) 66253-A	First edition

Thank you for selecting the A7BDE-A3N-PT32S3 A3-CPU Programmable Controller option card. Please read this manual carefully so that the equipment may be used to its optimum. A copy of this manual should be forwarded to the end user.

Users are asked to read the "Software Grant Agreement" before operating the A7BDE-A3N-PT32S3 option card.

IB (NA) 66253-A

CONTENTS

1. INTRODUCTION

This manual explains the functions, handling, and installation procedure of the A7BDE-A3N-PT32S3 A3-CPU Programmable controller option cards, the accompanying driver software, and Access Function Library.

The A7BDE-A3N-PT32S3 system consists of three option cards. Together they enable an A3N PLC CPU and interfaces with the networks MELSECNET AND MELSECNET/MINI-S3, to be installed in an IBM® PC-AT® or compatible personal computer.

Access to the A7BDE-A3N-PT32S3 by the user's application program is made via a system software driver. To aid the programmer, a sample Access Function Library, compatible with the Microsoft C Compiler and Linker, is provided.

We recommend that the Type ACPU Programming Manual, the Type Data Link System User's Manual, and the MELSECNET/MINI-S3 Master Module User's Manual are thoroughly read and understood before attempting to operate the A7BDE-A3N-PT32S3.

1.1 Features

The A7BDE-A3N-PT32S3 Access Function Library enables:

(a) Sequence program device monitoring and control
(b) Sequence program read and write
(c) A7BDE-A3N-PT32S3 SCPU Interrupt sequence program initiation
(d) Remote/local station Special Function Module access
(e) A7BDE-A3N-PT32S3 operating status monitor and control
(f) Master/Slave Free data transmission to A7BDE-J71P21/R21 stations

There are three option cards, one of each may be installed in an IBM® PC-AT® or compatible personal computer.

(a) The A7BDE-A3N-PT32S3A MELSECNET/MINI-S3 Interface Card

This card allows the installed A3N CPU Programmable Controller to be configured as the master station of a MELSECNET/MINI network. Its features and operation are the same as the A-PLC rack mounted MELSECNET/MINI master unit, the AJ71PT32. Though installed in a PC, it is regarded by the A3N CPU as occupying the second slot of a rack system, and communication is made via the sequence program TO/FROM instructions and dedicated control I/O. This card can only operate in conjunction with the A7BDE-A3N-B.C and may not be installed in a PC alone.

For further details, please see section 4.11 and the AJ71PT32 Master Module User's Manual.

(b) The A7BDE-A3N-B.C Programmable Controller Option Card

This card has three main features: The A3N CPU (referred to as the SCPU), the MCPU, and a High-Speed Device Access Memory.

The SCPU has the same features as the A3N Programmable Controller CPU, with a few exceptions. A general comparison is given in the appendix.

- 1968 Remote I/O Points. An additional 80 I/O points (XY00-XY4F) are reserved by the operating system for communications between the PC and the A7BDE-A3N-PT32S3.

- Main and Sub Programs both a maximum of 30K steps. (60K Total)
  261 Programming instructions (sequence, basic, and application)
  Processing speed, averaging 1.0 to 2.3 micro seconds per step.

- Pre-installed RAM, fixed at 64K Bytes (equivalent to A3NMCA-8 Memory Cassette). May be optionally extended by another 64K Bytes of ROM.

- RS422 Serial Port, for programming and monitoring by peripheral devices.

- General operation features include constant scan, latch, remote run/stop/pause, status latch, sampling trace, step run, off-line switch, and real time clock

  For further details, please see section 4.12.

The IFMEM enables general purpose communication between the PC application program and the SCPU. To the SCPU, it is regarded as a special function unit occupying the first slot of a rack system, with a buffer memory (3K words) and general purpose I/O (11 inputs, 7 outputs). Access is by means of the sequence program TO/FROM instructions, input contacts and output coils. The PC application program may access the same buffer memory, read and write data, and control or monitor the general purpose I/O.

For further details, please see section 4.4.

The High Speed Device Access Memory enables device data to be quickly transferred to and from the PC application program, even during the sequence program scan of the SCPU. Device data to and from the high speed memory and the SCPU is refreshed during the END processing of the SCPU sequence program. Direct access to the SCPU device memory would involve long function processing times, due to the delay in waiting for the end of the SCPU sequence program scan.

For further details, please see section 4.8.

(c) The A7LU1EP21/R21 MELSECNET Interface Card

This card allows the installed A7BDE-A3N-PT32S3 CPU Programmable Controller to be configured as the master station or as a slave station of the network MELSECNET. If configured as the master station, the SCPU may directly control the operation of remote I/O stations. General features and operation are the same as those of a standard MELSECNET A-PLC station. This card only operates in conjunction with the A7BDE-A3N-B.C and may not be installed in a PC alone.

1.2 General System Precautions

The following points and precautions must be noted when designing A7BDE-A3N-PT32S3 systems.

1.3 Hardware Restrictions

(a) An extension base unit cannot be connected to an A7BDE-A3N-PT32S3 option card. All I/O control is performed via stations of MELSECNET, or MELSECNET/MINI-S3.

(b) All general purpose I/O Units may be installed in MELSECNET remote stations, with the exception of the dynamic combined I/O unit, the A42XY.

(c) The following special function modules may not be used in MELSECNET remote I/O stations:

*A11VC	*AD51(S3)	*AD57(S1)
*AD58	*AI61	*AJ71P21/R21
*AJ71C21(S1)	*AJ71C22	*AJ71C24(S3)
*AJ71PT32		

(d) The RAM memory capacity is fixed at 64k Bytes (equivalent to the A3NMCA-8 Memory Cassette). RAM memory capacity cannot be increased or decreased. However, another 64K Bytes of ROM containing, for example, the SCPU sequence program, may be added by the user.

1.4 Software Restrictions

(a) The following utility packages, in conjunction with a peripheral programming device, may be used with the A7BDE-A3N-PT32S3:

*SW0C-UTLP-FN0	*SW0GHP-UTLPC-FN0
*SW0GHP-UTLP-FD1	
*SW0-SAPA	

(b) The following utility packages may not be used with the A7BDE-A3N-PT32S3:

*SW0C-UTLP-PID	*SWGHP-UTLPC-PID
*SW0GHP-UTLPC-FN1	
*SW0-AC57P	*SW1GP-AD57P
*SW0GHP-MBASC	

POINT

1. The A7BDE-A3N-PT32S3 option card and system Software Driver are compatible with the following systems:

   Computer: IBM® PC/AT® (or compatible)
   Operating system: MS-DOS® Ver. 3.1 or PC-DOS Ver. 3.2
   Interface Port: 16-Bit PC/AT Standard
   8 MHz Bus Clock
   Support for 4 Wait States

   Operation is not guaranteed when the A7BDE-A3N-PT32S3 is installed in a computer, other than that specified above.

2. In this manual: PC = Personal Computer
   PLC = Programmable Logic Controller

3.

COMPONENT		No.
A7BDE-A3N-PT32S3A MELSECNET/MINI Option Card		1
A7BDE-A3N-B.C A3-CPU and Memory Option Card		1 Pair
ACP2PC A3N-A to A3N-B.C Cable Connector		1
MELSECNET/MINI Twisted-Pair Connector (DDK 17JE-2390-02-D8A)		1
A7LU1EP21/R21 MELSECNET Fiber Optic or Co-Axial interface card.		1
ACP2LU1 A7LU-P21/R21 to A3N-B.C Cable Connector		1
Software driver Access Function Library Assembler Source Code	SW01M-A3N-3.5 (3.5 INCH)	1
	SW01M-A3N-5 (5 INCH)	1

4. Disk contents: The same files are included in the 3.5 and 5 inch disks.

   MA3N.SYS (A3N MAIN + MNET interrupt drivers)
   NYUSERC.H (Driver C Interface Include File)
   MMSCL.LIB (Driver C Interface Library - Large)
   MMSCS.LIB (Driver C Interface Library - Small)
   MMSCL.ASM (Assembler Interface Library - Large)] *1
   MMSCS.ASM (Assembler Interface Library - Small)] *1

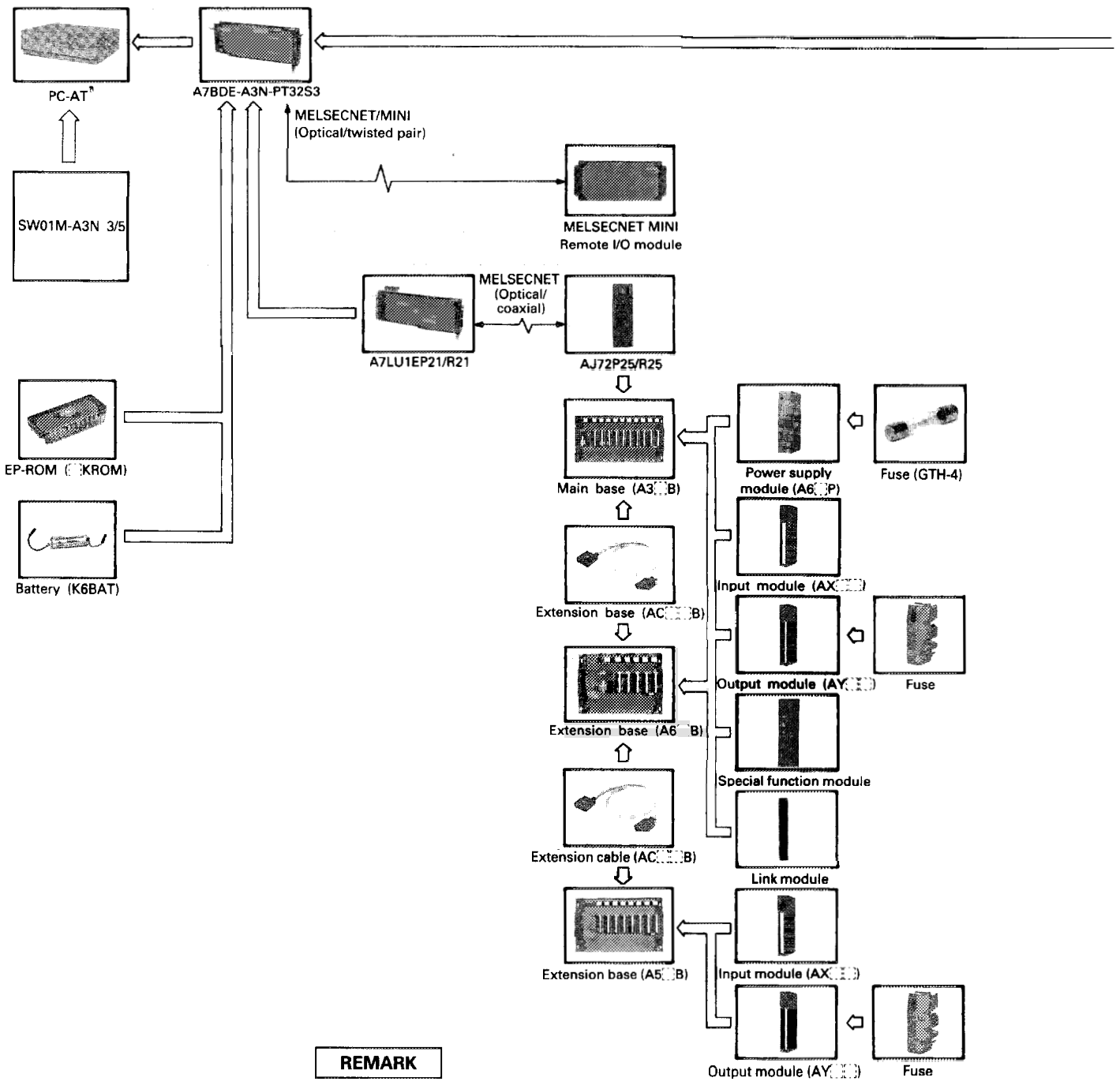
   *1 Source Code of MMSCL.LIB and MMSCS.LIB.

2. SYSTEM CONFIGURATION

The following sections give the general configurations of A7BDE-A3N-PT32S3 systems.

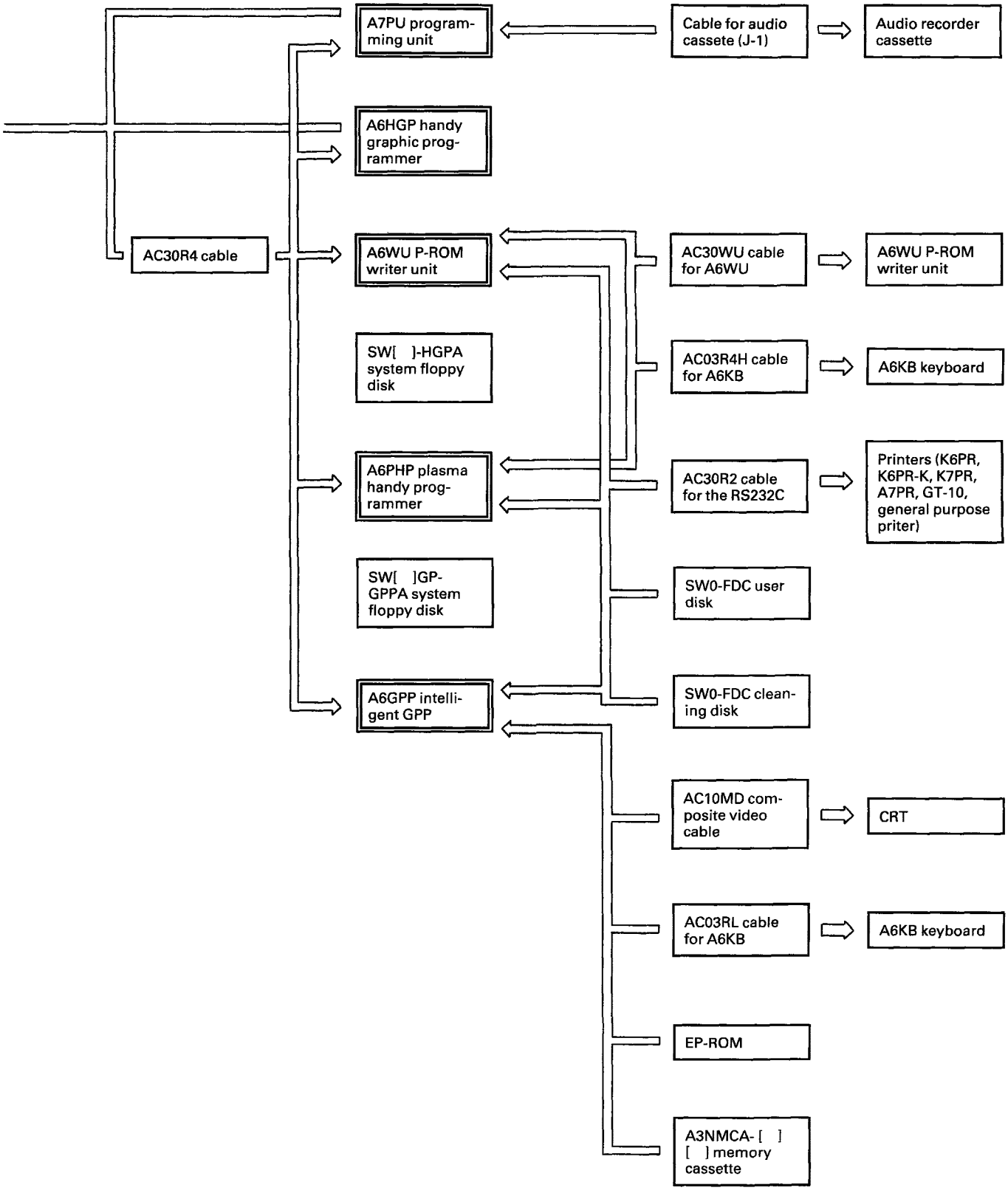
2.1 Overall System Configuration

The following diagram gives the overall system configuration, with the A7BDE-A3N-PT32S3 installed in an IBM® PC/AT® or compatible computer.



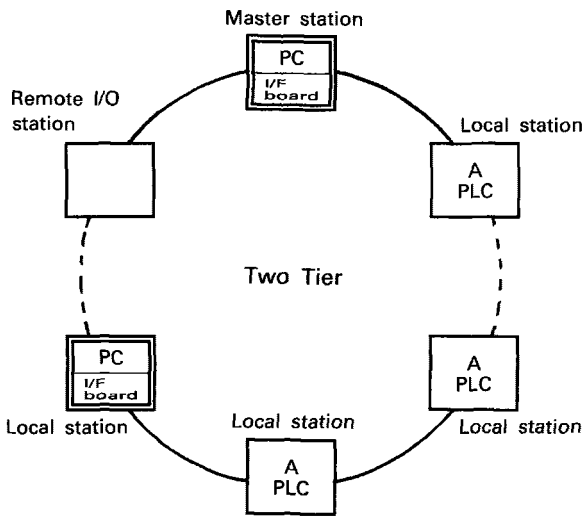
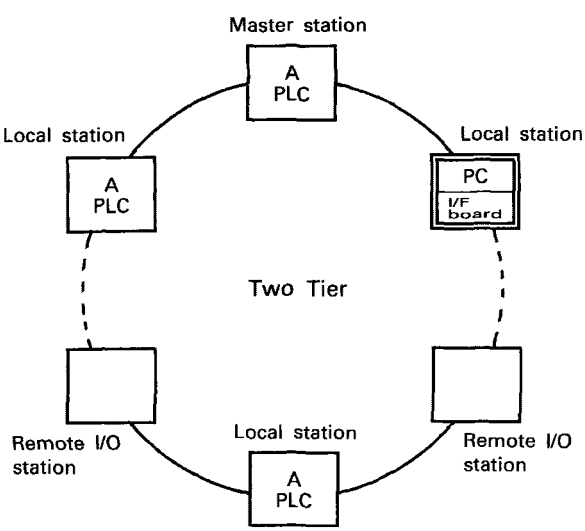
REMARK

(1) Configuring a MELSECNET data link requires the A7LU1EP21/R21.
(2) The A7BDE-A3N-PT32S3 and A7LU1EP21/R21 are connected using the ACP2LU1 cable. The ACP2LU1 is provided with the A7LU1EP21/R1.

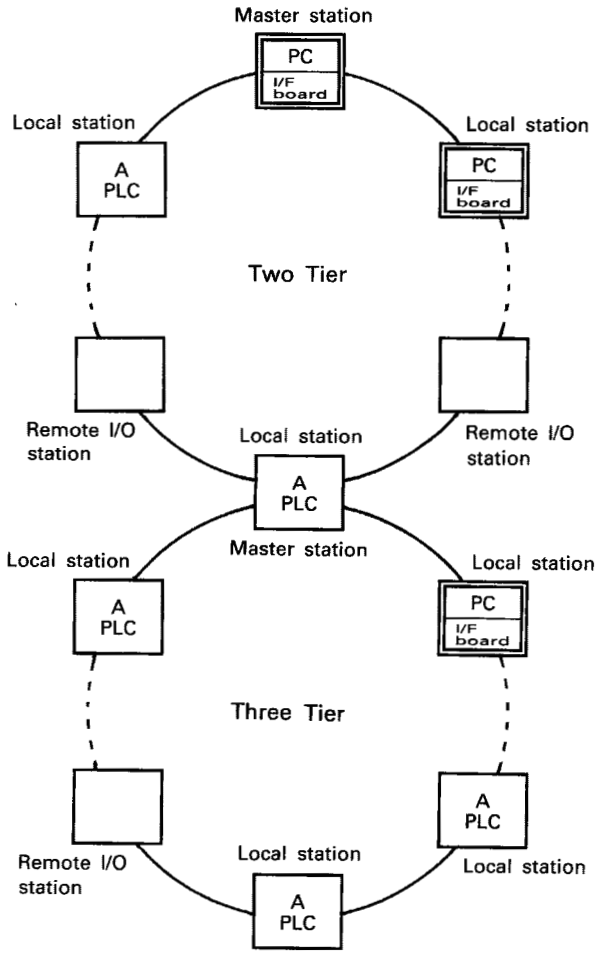

A7PU programming unit
Cable for audio cassete (J-1)
Audio recorder cassette
A6HGP handy graphic programmer
AC30R4 cable
A6WU P-ROM writer unit
AC30WU cable for A6WU
A6WU P-ROM writer unit
SW[ ]-HGPA system floppy disk
AC03R4H cable for A6KB
A6KB keyboard
A6PHP plasma handy programmer
AC30R2 cable for the RS232C
Printers (K6PR, K6PR-K, K7PR, A7PR, GT-10, general purpose priter)
SW[ ]GP-GPPA system floppy disk
SW0-FDC user disk
A6GPP intelligent GPP
SW0-FDC cleaning disk
AC10MD composite video cable
CRT
AC03RL cable for A6KB
A6KB keyboard
EP-ROM
A3NMCA-[ ] [ ] memory cassette

2.2 MELSECNET Configuration

The following diagram shows the A7BDE-A3N-PT32S3 configured as a master or local station of MELSECNET. (Two Tier System)

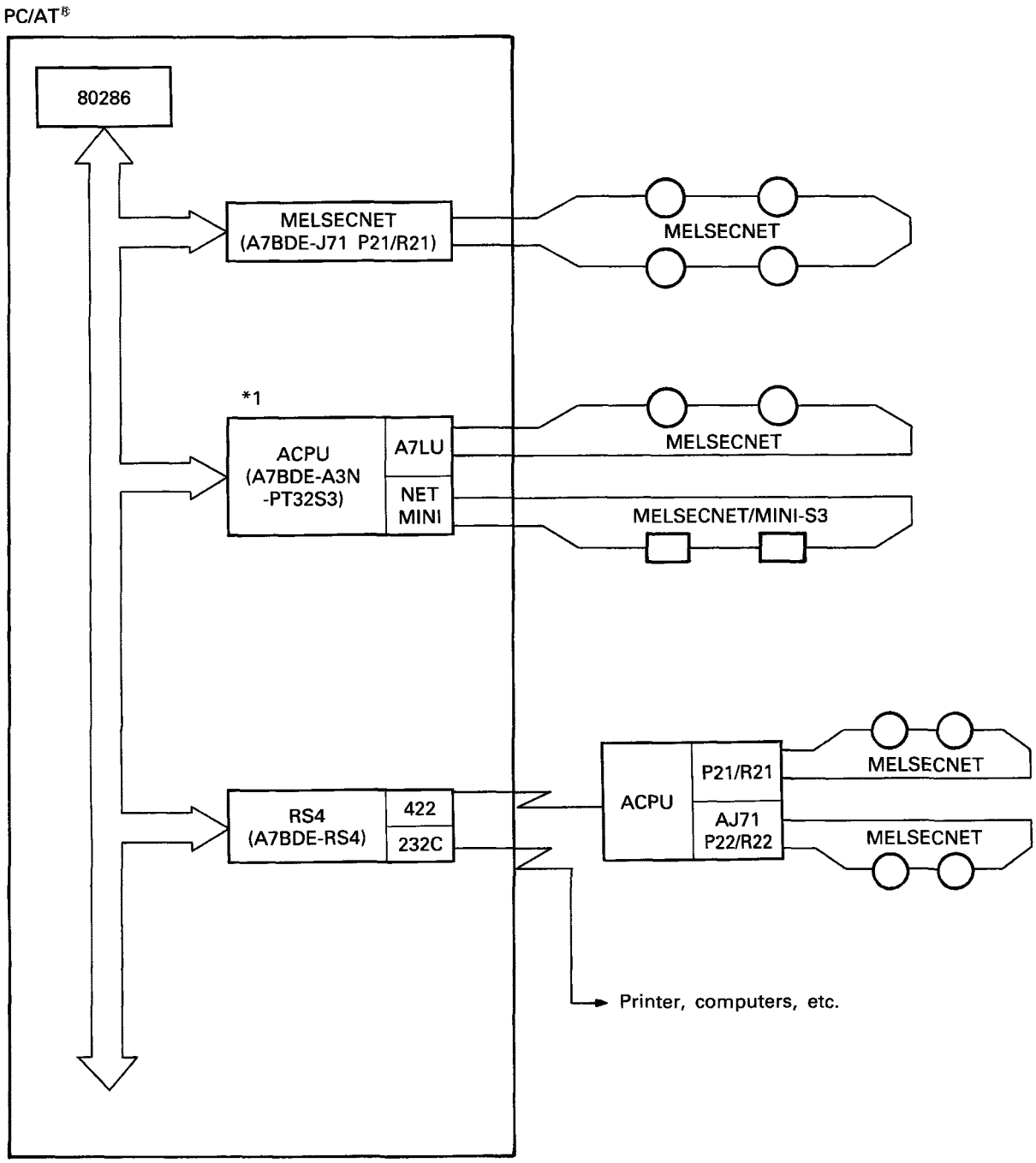
SYSTEM CONFIGURATION	REMARKS
A7BDE-A3N-PT32S3 AS the MASTER STATION	
	(1) When the PC-A7BDE-A3N-PT32S3 is used as a master station it supervises the network, and conducts link parameter settings. (2) A maximum of 64 stations may be connected. (3) Access to remote I/O stations is conducted using the interface driver and Library functions.
PC-A7BDE-A3N-PT32S3 AS a LOCAL STATION	
	(1) A maximum of 64 stations may be connected.

The following diagram shows the A7BDE-A3N-PT32S3 configurated as a local station of MELSECNET. (Three Tier System)

SYSTEM CONFIGURATION	REMARKS
A7BDE-A3N-PT32S3 AS a LOCAL STATION OF the THIRD TIER	
	(1) The A7BDE-A3N-PT32S3 may be configurated as a local station of a third tier MELSECNET network, but not as the master station. (2) Sixty Four stations may be connected to a two tier loop but only sixty three to a third tier loop.

2.2 Installation Configuration

The A7BDE-A3N-PT32S3 Programmable Controller option card is one of a series of three Mitsubishi option cards for use with the IBM® PC/AT® or compatible computer. The other two option cards are the A7BDE-RS4 Serial Interface Card, and the A7BDE-J71P21/R21 MELSECNET Interface Card. Their general configuration, when installed in a PC/AT®, is given below.

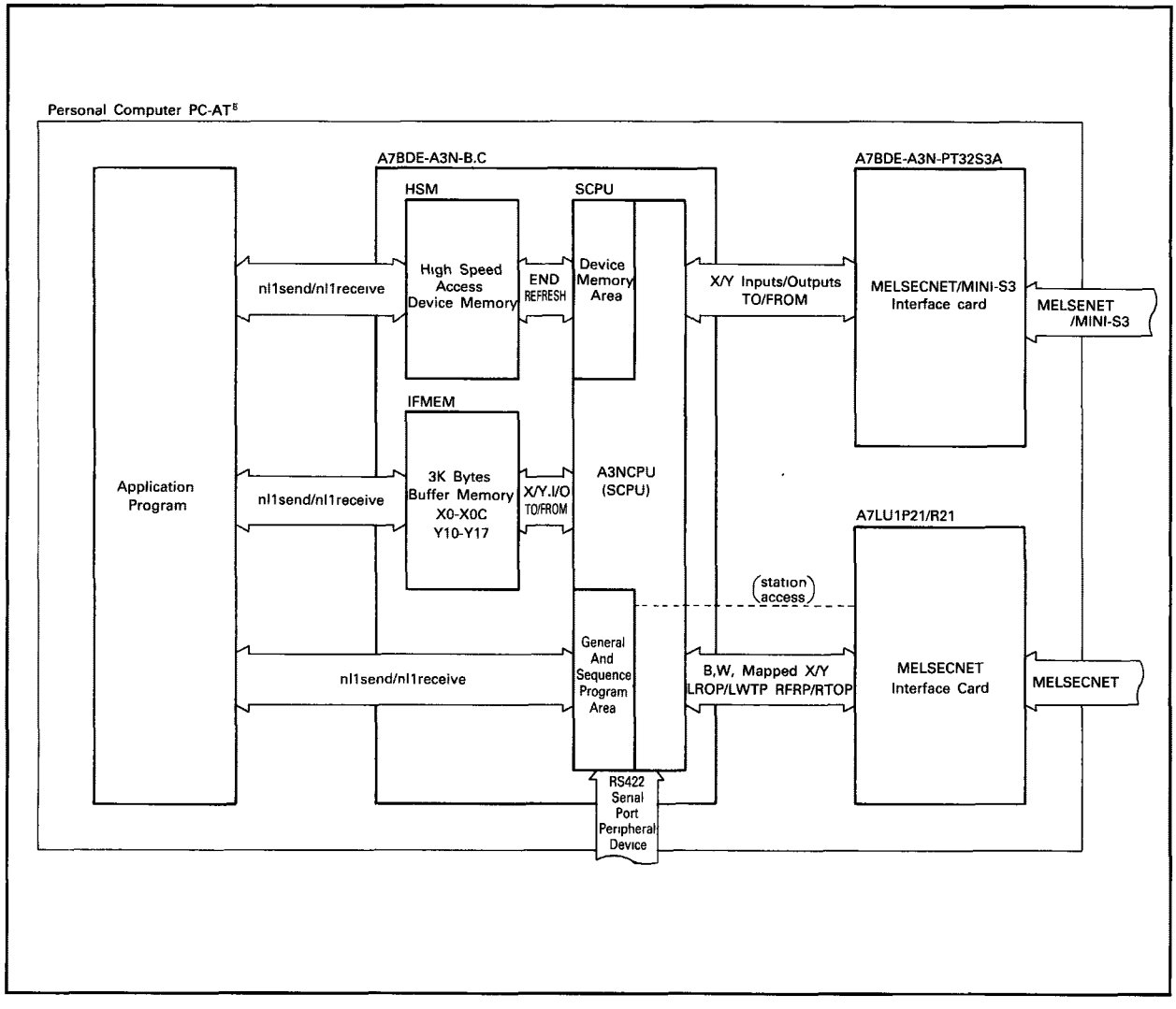


*1 Covered by this document

2.4 Communication Channel Configuration

The diagram below shows the general communication paths between the three option cards (A7BDE-A3N-PT32S3A/B.C A7LU1EP21/R21) and the application program when installed in the personal computer.

For further information, please see section 4.

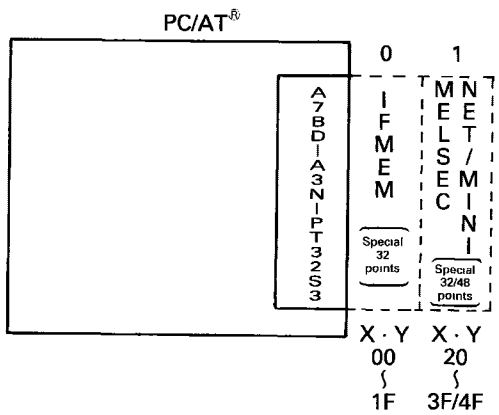


2.5 Input/Output System Configuration

An extension base unit cannot be connected to the SCPU. Therefore to use I/O modules and special function modules requires a remote I/O system to be configured using either MELSECNET or MELSECNET/MINI-S3. As shown below, the I/O numbers of slots 0 and 1 are occupied by the system.

(1) Slot 0 is assigned 32 points for the IFMEM. These devices are used for data transfer between the IFMEM, the PC Application Program, and the SCPU.

(2) Slot 1 is assigned 32/48 points for the MELSECNET/MINI-S3 master unit. (Number of points varies depending on the jumper settings of the number of I/O points occupied)



(1) Extension bases cannot be connected to SCPU. However, the base unit is shown in order to explain I/O numbers.
(2) The I/O numbers available to the user are those after X and Y40/50.

SCPU I/O Number Assignments

3. SPECIFICATIONS

The following sections describe the specifications of the A7BDE-A3N-PT32S3 A3-CPU Programmable Controller option card.

3.1 General Specifications

Item	Specifications
Operating ambient temperature	0 to 40°C
Storage ambient temperature	−20 to 75°C
Operation ambient humidity	20 to 80% RH, non-condensing

POINT

The above specification is for the user's computer and A7BDE-A3N-PT32S3 combined.

3.2 Performance Specifications

Item \ Type		A7BDE-A3N-PT32S3		Refer to:
Control system		Repeated operation (using stored program)		—
I/O control method		Direct mode*		—
Programing language		Language dedicated to sequence control (Combined use of relay symbol type, logic symbolic language, and MELSAP)		—
Number of functions	Sequence instruction	22		
	Basic instruction	132		
	Application instruction	107		
Processing speed (sequence instruction) (sec/step)		1.0 ~ 2.3		—
I/O points		Max. 1968 (The system occupies 80 points = X/Y00 to 4F)		
Watch dog timer (WDT) (ms)		10 ~ 2000		
Memory capacity (bytes)		64K bytes (internal and fixed)		—
Program capacity		(Main sequence program + main microcomputer program) ~ maximum of 30 steps Internal main microcomputer program can be set to a maximum of 58K bytes (29K steps).		For details, refer to the Programming Manual.
		(Sub sequence program + sub microcomputer program) = maximum of 30 steps Internal sub microcomputer program can be set to a maximum of 58K bytes (29 steps).		
Device: Internal relay (M) (points)		1000 (M0 ~ 999)	(The number of M, L and S = 2048 (set in parameters))	
Device: Latch relay (L) (points)		1048 (L 1000 ~ 2047)		
Device: Step relay (S) (points)		0 (Defaults to no value)		
Device: Link relay (B) (points)		1024 (B0 ~ 3FF)		
Device: Timer (T)	Number of points	256		
	Specifications	100 ms timer: setting time 0.1 to 3276.7 sec (T0 to T199) 10 ms timer: setting time 0.01 to 327.67 sec (T200 to T199) 100 ms retentive timer: setting time 0.1 to 3276.7 sec	Set in parameters	
Device: Counter (C)	Number of points	256		
	specifications	Normal counter: setting range 1 to 32767 (C0 to 255) Counter for interrupt program: setting range 1 to 32767 → Counters used in interrupt programs	Set in parameters	
Device: Data register (D) (points)		1024 (D0 ~ 1023)		
Device: Link register (W) (points)		1024 (W0 ~ 3FFF)		

Item \ Type		A7BDE-A3N-PT32S3	Refer to:
Device	Annunciator (F) (points)	256 (F0 to 255)	For details, refer to the Programming Manual.
	File register (R) (points)	Max. 8192 (R0 to 8191)	
	Accumulator (A) (points)	2 (A0, A1)	
	Index register (V, Z) (points)	2 (V, Z)	
	Pointer (P) (points)	256 (P0 to 255)	
	Pointer for interruption (I) (points)	32 (I0 to 31)	
	Special relay (M) (points)	256 (M9000 to 92555)	
	Special register (D) (points)	256 (D9000 to 92555)	
Comment (points)		(specify in batches of 64 points)	——
Self-diagnostic functions		Watch dog error monitor, memory error detection, CPU error detection, I/O error detection, battery error detection, etc.	
Operation mode at the time of error		STOP/CONTINUE	——
STOP RUN Output mode		Output data at time of STOP restored/data output after operation execution	——
Permissible momentary stop time (ms)			——
Maximum number loaded		One per A7LMS-DH/D	——
Number of occupied slots		2 slots (3 slots when A7LU1P21/R21 is loaded)	——
Weight (kg) (lb)		0.75 (1.1 kg when A7LU1P21/R21 is loaded)	——

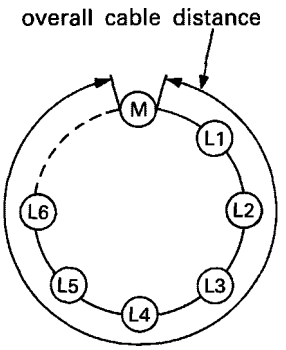
*The SCPU uses the direct method, however, since the I/O modules are installed in either the MELSECNET of MELSECNET/MINI, the delay time of the I/O's are determined by each of data link processing times.

3.3 MELSECNET A7LU1EP21/R21 Communication Specifications

—			A7LU1EP21	A7LU1ER21
Data Link			Optical Data Link	Coaxial Data Link
2/3 tier extended system		Master station	Usable	
		Local station	Usable	
Cyclic transmission	Maximum link points per system	Input (X) Output (Y)	1968 (246 bytes)	However, when master station is A1NCPU21/R21 ····· 256 points (32 bytes) When master station is A2NCPU21/R21 ····· 512 points (64 bytes) When master station is A2NCPU21/R21-S1 ····· 1024 points (128 bytes)
		Link relay (B)	1024 (128 bytes)	
		Link register (W)	1024 (2048 bytes)	
	Maximum link points per station		$\frac{Y\,(\text{points}) + B\,(\text{points})}{8} + 2 \times W\,(\text{points})$ 1024 bytes	
Transient transmission		Master station	All devices and programs of the programmable controller CPU of each local station can be accessed.	
		Local station	All devices and programs of the programmable controller CPU of the master station can be accessed.	
Communication speed			1.25 MBPS	
Communication method			Half duplex, bit serial method	
Synchronous method			Frame synchronous method	
Transmission path			Duplex loop	
Overall loop distance (km/mile)			Maximum 10 km/6.21 miles (1 km/0.621 miles between stations)	Maximum 10 km/6.21 miles (0.5 km/0.31 miles between stations)
Number of stations connected			Maximum 65 stations per loop (1 master station, 64 local/remote I/O stations)	
Modulation method			CMI method	
Transmission format			Conforms to HLDC (frame format)	
Error control method			CRC check and retry after time-out	
RAS function			Loopback function on error detection or cable breakage, diagnostic functions such as link check	
Connector			2-core optical connector plug (CA9003)	BNC-P-5, BNC-P-3 Ni (DDK) or equivalent
Cable			Si-200/250	3C-2V, 5C-2V or equivalent
Transmission loss			Max. 12 dBm/km	——
Send level			−15 to −10 dBm (peak value)	——
Receive level			−30 to −10 dBm (peak value)	——

REMARKS

1. The overall loop distance refers to the distance from the master station sending port to the master station receiving port via local stations.
   For both the fiber optic cables and coaxial cables, the overall loop distance is a maximum of 10 km.
2. Refer to the "MELSECNET Data Link System Reference Manual" for information related to specifications concerning fiber optic and coaxial cables.



3.4 MELSECNET/MINI-S3 A7BDE-A3N-PT32S3A Communication Specifications

		A7BDE-A3N-PT32S3A		Remarks
		Optical Data Link	Twisted Pair Data Link	
For one A7BDE-A3N-PT32S3A	Max. number of link stations	64		No limit to the number of master modules used.
	Input (points)	512		Number of input/output points = 8 per remote I/O station. Total number of input + output points = 512.
	Output (points)	512		
I/O refresh time (ms)		3.2 to 18 *1 (when 64 stations are connected)		
Communication speed (BPS)		1.5M		
Optical transmission level (dB)		−14.4 to −11.6	—	
Optical receive level (dB)		−30 to −14	—	
Optical wave length (mm)		660 (Visible radiation)	—	
Max. interstation transmission distance (m/ft)		1 to 50/3.28 to 164*3	1 to 100/3.28 to 328 (50/164)*2	No limit on overall distance.
Number of I/O points occupied		I/O dedicated mode: 32 Extension mode: 48		Will be changed by the setting of mode switching jumper pins.
5V DC internal current consumption (A)		0.35		
Weight kg (lb)		0.6 (1.32)		

(1) Max. number of link stations per master module

Indicates that the total number of occupied stations assigned to the remote I/O units is up to 64 stations.
For example, up to 8 compact remote I/O units (AJ35PTF-56DT which occupies 8 stations) can be connected.
The allowable maximum number of remote terminal units (occupying 4 stations) is 14.
For the number of stations occupied by each type of the remote terminal units, see the appropriate remote unit user's manual.

(2) Max. number of link points per master module

Depends on the type of remote I/O unit connected.

Example 1: If 8 compact remote I/O units (AJ35PTF-56DT which occupies 8 stations) are used, 256 input and 192 output points can be controlled.

Example 2: If 16 partial refresh type remote I/O units (AJ35PTF-128DT which occupies 4 stations) are used, 1024 input and 1024 output points can be controlled.

REMARK

Use of the partial refresh type remote I/O unit increases the maximum number of link points per master module but makes the I/O response time longer than the batch refresh type remote I/O unit, e.g. the response time of the AJ35PTF-128DT is 107ms max. for input and 21.5ms for output.

POINT

***1: The I/O refresh time is determined by the number of remote units connected in the system, their types, and the setting of the operation mode switch of the master module as indicated below.**

R: Total number of remote stations
B: Number of AJ35PTF-128DT units connected
T: Number of remote terminal units connected

Mode Setting	Operation Mode Switch	I/O Refresh Time (msec)
I/O dedicated mode	Online automatic return (0)	I/O refresh time = 0.48 + (0.042×R) + (0.2×B)
	Online no-automatic return (1)	I/O refresh time = 0.46 + (0.053×R) + (0.2×B)
	Communication stop when error is detected (2)	I/O refresh time = 0.44 + (0.046×R) + (0.2×B)
Extension mode	Online automatic return (0)	I/O refresh time = 0.66 + (0.044×R) + (0.25×B) + (0.95×T)
	Online no-automatic return (1)	I/O refresh time = 0.54 + (0.058×R) + (0.25×B) + (0.95×T)
	Communication stop when error is detected (2)	I/O refresh time = 0.54 + (0.051×R) + (0.25×B) + (0.95×T)

***2: The maximum inter-station transmission distance depends on the twisted-pair cable diameter as follows:**

0.2mm² (0.00031in²) to less than 0.5mm² (0.00077in²) ····· 50m (164ft)
0.5mm² (0.00077in²) or more ····· 100m (328ft)

***3: The inter-station transmission distance of the optical fiber cable is between 1m (3.28ft) and 50m (164ft). Normal communication cannot be guaranteed for distances less than 1m (3.28ft).**
Assembling method of optical fiber cable differs depending on cable length; 1m (3.28ft) to less than 5m (16.4ft), or 5m (16.4ft) or more.

3.5 System software Driver Specifications

The following table gives the available functions of the System Software Driver.

NO	ITEM	FUNCTION	PROCESSING	A3N MASTER STATION			A3N LOCAL STATION			PROCESSING CODE (HEX)	REMARKS
				HOST	SLAVE		HOST	MASTER			
					ACPU	A7BDE		ACPU	A7BDE		

TABLE KEY

NO: Number of the function

ITEM: Function Type

FUNCTION: Function Name

PROCESSING: Function Operation

A3N MASTER STATION: PC-A7BDE-A3N-PT32S3 as the master station

A3N LOCAL STATION: PC-A7BDE-A3N-PT32S3 as a local station

SLAVE: Indicates access to a slave station via the master station

MASTER: Indicates access to the master station via the slave station

HOST: PC to the host A7BDE-A3N-PT32S3 Programmable Controller option card

ACPU: PLC or A7BDE-A3N-PT32S3 station of MELSECNET

A7BDE: PC-A7BDE-J71P21/R21 station of MELSECNET

PROCESSING CODE: Processing code for a particular function operation (hexadecimal)

REMARKS: Page reference of Access Function example

NOTE. (○) = Available
(—) = Unavailable

3.6 Access Function Table

NO	Item	Function	Processing	A3N MASTER STATION			A3N SLAVE STATION			Processing Code (HEX)	Remark
				HOST	SLAVE		HOST	MASTER			
					ACPU	A7BDE		ACPU	A7BDE		
1	ACPU access	ACPU memory access	Batch read	○	○	—	○	○	—	2	Page 6-26
2			Batch write	○	○	—	○	○	—	4	Page 6-28
3			Random read	○	○	—	○	○	—	5	Page 6-30
4			Random write	○	○	—	○	○	—	6	Page 6-32
5		ACPU sequence program access	Batch read	○	○	—	○	○	—	1	Page 6-34
6			Batch write	○	○	—	○	○	—	3	Page 6-36
7			SCPU interrupt program starting	○	—	—	○	—	—	100	Page 6-38
8		ACPU control	Remote RUN/STOP/ PAUSE	○	○	—	○	○	—	18	Page 6-40
9			Requested ACPU Check	○	○	○	○	○	○	8	Page 6-42
10			Parameter analysis request	○	○	—	○	○	—	27	Page 6-44
11	Special module access	Special module access	Shared memory batch read	○	○	—	○	○	—	10	Page 6-46
12			Shared memory batch write	○	○	—	○	○	—	12	Page 6-48
13	A7BDE-A3N-PT32S3 General access	IFMEM access	Batch read	○	—	—	○	—	—	200	Page 6-50
14			Batch write	○	—	—	○	—	—	201	Page 6-52
15			Random read	○	—	—	○	—	—	202	Page 6-54
16			Random write	○	—	—	○	—	—	203	Page 6-56
17			IFMEM input X write	○	—	—	○	—	—	204	Page 6-58
18			IFMEM input Y read	○	—	—	○	—	—	205	Page 6-60
19		High-speed device memory access	Transfer setting for A3N device memory	○	—	—	○	—	—	803	Page 6-62
20			Batch read	○	—	—	○	—	—	206	Page 6-64
21			Batch write	○	—	—	○	—	—	207	Page 6-66
22			Random read	○	—	—	○	—	—	206	Page 6-68
23			Random write	○	—	—	○	—	—	209	Page 6-70
24	A7BDE-A3N-PT32S3 card status monitor and control	A7BDE-A3N-PT32S3 card status monitor and control	Reading LED status	○	—	—	○	—	—	700	Page 6-72
25			Reading switch status	○	—	—	○	—	—	701	Page 6-74
26			A3N board version read	○	—	—	○	—	—	702	Page 6-76
27			Resetting A3N board	○	—	—	○	—	—	800	Page 6-78
28			Resetting A3N indicator	○	—	—	○	—	—	80A	Page 6-80
29	General data	General data	Data free transmission	—	—	○	—	—	○	40	Page 6-82

3.7 System Equipment Specifications

The following tables list the available A-Series system equipment, for use with the A7BDE-A3N-PT32S3, and Remote Stations of MELSECNET.

System Equipment

Module		Type	Description	Occupied Points	Current Consumption		Applicable System								Remarks
							Independent	Coaxial data link			Optical data link			Computer link	
					5 VDC	24 VDC		M station	L station	R station	M station	L station	R station		
Memory	EP-ROM	4KROM	8KB (max.3K steps)	—	—	—	○	○	○	—	○	○	—	○	●Two memories of the same type are used.
		8KROM	16KB (max.7K steps)												
		16KROM	32KB (max.15K steps)												
Input module		AX10	16 points, 100-120 VAC	16	0.06 A	—	—	—	—	○	—	—	○	—	
		AX11	32 points, 100-120 VAC	32	0.11 A	—									
		AX20	16 points, 200-240 VAC	16	0.06 A	—									
		AX21	32 points, 200-240 VAC	32	0.11 A	—									
		AX40	16 points, 12/24 VDC	16	0.06 A	—									
		AX41	32 points, 12/24 VDC	32	0.11 A	—									
		AX42	64 points, 12/24 VDC	64	0.12 A	—									
		AX60	16 points, 100/110/125 VDC	16	0.06 A	—									
		AX70	16 points, for sensor	16	0.06 A	—									
		AX71	32 points, for sensor	32	0.11 A	—									
		AX80	16 points, 12/24 VDC source loading	16	0.06 A	—									
		AX80E	16 points, 12/24 VDC source loading	16	0.06 A	—									
		AX81	32 points, 12/24 VDC source loading	32	0.11 A	---									
		AX81-S2	32 points, 12/24 VDC source loading	32	0.11 A	—									
		AX82	64 points, 12/24 VDC source loading	64	0.12 A	—									

System Equipment

	Module	Type	Description	Occupied Points	Current Consumption		Applicable System									Remarks
							Independent	Coaxial data link			Optical data link					
					5 VDC	24 VDC		M station	L station	R station	M station	L station	R station	Computer link		
Special function module	Single axis positioning	AD70	For single axis positioning control, speed control, speed and positioning control Analog voltage output (0 to ±10 V) Analog input type Permits normal servo operation.	32	0.3 A	—	—	—	—	○	—	—	○	—		
Special function module	Positioning	AD71	For positioning control Pulse chain output, 2 axes (independent, simultaneous, linear interpolation) Use with AD76 for stepping motor control.	32	1.5 A	—	—	—	—	○	—	—	○	—		
Special function module	Positioning	A71S1	For positioning control MELDAS-S1 servo driver. Pulse chain output, 2 axes (independent, simultaneous, linear interpolation)	32	1.5 A	—	—	—	—	○	—	—	○	—		
Special function module	Positioning	AD71S2	For positioning control Pulse chain output, 2 axes (independent, simultaneous, linear interpolation) Use with AD76 for stepping motor control.	32	1.5 A	—	—	—	—	○	—	—	○	—		
Special function module	Positioning	AD72	For positioning control Analog voltage output (0 to ±10 V) 2 axes (independent, simultaneous, linear interpolation)	48 (First 16: vacant, Last 32: special)	0.9 A	—	—	—	—	○	—	—	○	—		
Special function module	Positioning	AD76	Stepping motor driver Use with AD71 or AD71S2	16	—	—	—	—	—	○	—	—	○	—		
Special function module	Position detection	A61LS	Detects absolute positions Resolution: 4096 divisions per resolver revolution Response speed: within 6 ms	48 (First 32: vacant, Last 16: special)	0.8 A	—	—	—	—	○	—	—	○	—		
Special function module	Position detection	A62LS	Detects absolute positions Multi-turn type Resolution: 4096 divisions max. per resolver revolution Response speed: within 2 ms	48 (First 32: special, Last 16: vacant)	1.5 A	—	—	—	—	○	—	—	○	—		
Special function module	High-speed counter	AD61	Binary 24 bits, 1/2 phase input, reversible counter 50KPPS, 2 channels	32	0.3 A	—	—	—	—	○	—	—	○	—		
Special function module	High-speed counter	AD61S1	Binary 24 bits, 1/2 phase input, reversible counter 1 phase·····10KPPS, 2phase ·····7KKPS 2 channels	32	0.3 A	—	—	—	—	○	—	—	○	—		

System Equipment

	Module	Type	Description	Occupied Points	Current Consumption 5 VDC	Current Consumption 24 VDC	Applicable System: Independent	Coaxial data link: M station	Coaxial data link: L station	Coaxial data link: R station	Optical data link: M station	Optical data link: L station	Optical data link: R station	Computer link	Remarks
Special function module	A/D converter	A68AD	4 to 20 mA/0 to +10 V Analog input, 8 channels	32	0.9 A	—	—	—	—	○	—	—	○	—	
		A68ADS2		32	0.9 A	—									
		A616AD	4 to 20 mA/0 to ±10 V Analog input, 16 channels Extensible up to 121 channels by means of the A60MX(R).	32	1.0 A	—	—	—	—	○	—	—	○	—	
		A60MX	Multiplex unit 4 to 20 mA/0 to ±10 V Multiplex devices: IC relay Analog input, 16 channels	18	0.5 A	—	—	—	—	○	—	—	○	—	Use with the A61AD or the A616TD.
		A60MXR	Multiplex unit 4 to 20 mA/0 to ±10 V Multiplex devices: Mercury relay Analog input, 16 channels	16	0.5 A	—	—	—	—	○	—	—	○	—	
	Temperature/digital conversion unit	A616TD	For temperature detection with a thermocouple (with the A60MXT connected 0 to ±10 V/0 to 20 mA (with the A70MX (R) connected)	32	1.0 A	—	—	—	—	○	—	—	○	—	
		A60MXT	Multiplex unit Temperature detection by a thermocouple in conjunction with the A616TD Temperature input, 15 channels	32 (First 16: vacant, Last 16: vacant)	0.8 A	—	—	—	—	○	—	—	○	—	Use with the A616TD.
	D/A converter	A62DA	Analog input, 2 channels	32	0.6 A	0.35 A	—	—	—	○	—	—	○	—	
		A62DAS1	Analog input, 2 channels												
	A/D, D/A converter	A84AD	Analog I/O, 2 channels	48 (First 16: vacant, Last 32: special)	0.24 A	0.53 A	—	—	—	○	—	—	○	—	
	Memory card Centronics interface	AD59	32K bytes memory may be connected to any printer conforming to Centronics standards.	32	0.3 A	—	—	—	—	○	—	—	○	—	
		AD59S1													
Data Link	Coaxial data link unit	AJ72R25	For remote I/O station	—	2.6 A	—	—	—	—	○	—	—	—	—	
		A0J2CPUR25	For remote I/O station	—	0.89 A	—									
	Optical data link unit	AJ72P25	For remote I/O station	—	2.3 A	—	—	—	—	—	—	—	○	—	
		A0J2CPUP25	For remote I/O station	—	0.47 A	—									

System Equipment

Module	Type	Description	Occupied Points	Current Consumption 5 VDC	Current Consumption 24 VDC	Applicable System: Independent	Coaxial data link: M station	Coaxial data link: L station	Coaxial data link: R station	Optical data link: M station	Optical data link: L station	Optical data link: R station	Computer link	Remarks
Dummy module	AG62	16, 32, 48 or 64 points may be selected.	Number of set points	0.07 A	—	—	—	—	○	—	—	○	—	With 16 simulation switches
Blank cover	AG60	Dustproof cover for use in vacant slot	16	—	—	—	—	—	○	—	—	○	—	
Power supply module	A61P	Input: 110/220 VAC; Output: 5 VDC 8 A; For use in power supply slot	—	—	—	—	—	—	○	—	—	○	—	
	A62P	Input: 110/220 VAC; Output: 5 VDC 5 A, 24 VDC 0.8 A; For use in power supply slot												
	A63P	Input: 24 VDC; Output: 5 VDC 8 A; For use in power supply slot												
	A65P	Input: 110/220 VAC; Output: 5 VDC 2 A, 24 VDC 1.5 A; For use in power supply slot												Must not be used on main base unit.
	A66P	Input: 110/220 VAC; Output: 24 VDC 1.2 A; For use in I/O slot	16	—	—									
	A68P	Input: 110/220 VAC; Output: 15 VDC +1.2 A −0.7 A; For use in I/O slot	32 (First 16: vacant Last 16: vacant)	—	—									Power supply for the AD70, A616DAV, and A616DAI.
Base unit: Main base unit	A38B	Can accommodate 8 I/O modules.	—	—	—	—	—	—	○	—	—	○	—	
	A35B	Can accommodate 5 I/O modules.												
	A32B	Can accommodate 2 I/O modules.												Without extension connector.
Base unit: Extension base unit	A68B	Can accommodate 8 I/O modules.												Requires power supply module.
	A65B	Can accommodate 5 I/O modules.												
	A58B	Can accommodate 8 I/O modules.												Must not be used with the A61P, A62P, A63P, A65P.
	A55B	Can accommodate 5 I/O modules.												
Extension cable	AC06B	600 mm (23.6 inch); For use between base units	—	—	—	—	—	—	○	—	—	○	—	
	AC12B	1200 mm (47.2 inch); For use between base units												
	AC30B	3000 mm (118.1 inch); For use between base units												
	LC06AB	600 mm (23.6 inch); For use between base units												
	LC12AB	1200 mm (47.2 inch); For use between base units												
Simulation switch	A6SW16	16 points simulation switch	—	—	—	—	—	—	○	—	—	○	—	Used with an input module.
	A6SW32	32 points simulation switch												
Others: Battery	A6BAT	IC-RAM backup	—	—	—	○	○	○	—	○	○	—	○	
Others: Fuse: For AY11E, AY13E	MF51NM8	Cartridge type 8 A	—	—	—	—	—	—	○	—	—	○	—	
Others: Fuse: For AY22	HP-70K	Plug type 7 A												
Others: Fuse: For AY23	HP-32	Plug type 3.2 A												
Others: Fuse: For AY50, AY80	MP-20	Plug type 2 A												
Others: Fuse: For AY60	MP-32	Plug type 3.2 A												
Others: Fuse: For AY60E	MP-50	Plug type 5 A												
Others: Fuse: For power supply	GTH-4	Cartridge type 250 V 4 A												
Others: Fuse: For A63P	SM6.3A	Cartridge type 6.3 A												

MELSECNET/MINI-S3 Equipment

Name	Type	Description	No. of Occupied Stations/ No. of Occupied Stations	Usable Master Module Modes	
				Extension mode	I/O dedicated mode
Data storage memory	16KROM	Stores initial data when the master module is used in the extension mode. (Installed in master module.)	—	○	—
		Stores message data when the operating box is used. (Installed in the master module.)	—	○	—
		Stores character generation data when the operating box is used. (Installed in the operating box.)	—	○	—
Stand-alone Remote I/O Unit (For optical data link)	AJ35PJ-8A	AC input unit, 100-120V AC, 8 points	1 station	○	○
	AJ35PJ-8D	DC input unit (sink type) 12/24V DC, 8 points			
	AJ35PJ-8R	Contact output unit, 24V DC 2A, 240V AC 2A, 8 points			
	AJ35PJ-8S1	Triac output unit, 100-240V AC, 0.6A/point, 8 points			
	AJ35PJ-8T1	Transistor output unit (sink type), 12/24V DC, 0.1A/point, 8 points			
	AJ35PJ-8T2	Transistor output unit (sink type), 12/24V DC, 0.5A/point, 8 points			
	AJ35PJ-8T3	22Transistor output unit (sink type), 12/24V DC, 2A/point, 8 points			
	AJ35PJ-8S2	Triac output unit, 100-240V AC, 2A/point, 8 points			
Stand-alone Remote I/O Unit (For twisted-pair data link)	AJ35TJ-8A	AC input unit, 100-120V AC, 8 points	1 station	○	○
	AJ35TJ-8D	DC input unit (sink type), 12/24V DC, 8 points			
	AJ35TJ-8R	Contact output unit, 24V DC 2A, 240V AC 2A, 8 points			
	AJ35TJ-8S1	Triac output unit, 100-240V AC, 0.6A/point, 8 points			
	AJ35TJ-8T1	Transistor output unit (sink type), 12/24V DC, 0.1A/point, 8 points			
	AJ35TJ-8T2	Transistor output unit (sink type), 12/24V DC, 0.5A/point, 8 points			
	AJ35TJ-8T3	Transistor output unit (sink type), 12/24V DC, 2A/point, 8 points			
	AJ35TJ-8S2	Triac output unit 100-240V AC, 2A/point, 8 points			
Cable-through fitting	——	For sealing cables into a stand-alone remote I/O station. User prepared.	—	○	○

MELSECNET/MINI-S3 Equipment

Name	Type	Description	No. of Occupied Stations/ No. of Occupied Stations	Usable Master Module Modes	
				Extension mode	I/O dedicated mode
Compact Type Remote I/O unit (for optical data link, twisted-pair data link)	AJ35PTF-32A	AC input unit, 100-120V AC, 32 points	4 stations	○	○
	AJ35PTF-32D	10DC input unit (sink type), 12/24V DC, 32 points			
	AJ35PTF-24R	Contact output unit, 24V DC 2A, 240V AC 2A, 24 points			
	AJ35PTF-24S	Triac output unit, 100-240V AC, 0.6A/point, 24 points			
	AJ35PTF-24T	Transistor output unit, 12/24V DC, 0.5A/point, 24 points			
	AJ35PTF-28AR	I/O unit Input side····· 100-120V AC, 16 points Output side ····· contact output, 24V DC 2A, 240V AC 2A, 12 points			
	AJ35PTF-28AS	I/O unit Input side····· 100-120V AC, 16 points Output side ····· triac output, 100-240V AC, 0.6A/point, 12 points			
	AJ35PTF-28DR	16I/O unit Input side····· sink type, 12/24V DC, 16 points Output side ····· contact output, 24V DC 2A, 240V AC 2A, 12 points			
	AJ35PTF-28DS	I/O unit Input side····· sink type, 12/24V DC, 16 points Output side ····· triac output, 100-240V AC, 0.6A/point, 12 points			
	AJ35PTF-28DT	I/O unit Input side····· sink type, 12/24V DC, 16 points Output side ····· transistor output, sink type, 12/24V DC, 0.5A/point, 12 points			
	AJ35PTF-56AR	I/O unit Input side····· 100-120V AC, 32 points Output side ····· contact output, 24V DC 2A, 24 points	8 stations		
	AJ35PTF-56AS	I/O unit Input side····· 100-120V AC, 32 points Output side ····· triac output, 100-240V AC, 0.6A/point, 24 points			
	AJ35PTF-56DR	I/O unit Input side····· sink type, 12/24V DC, 32 points Output side ····· contact output, 24V DC 2A, 240V AC 2A, 24 points			
	AJ35PTF-56DS	I/O unit Input side····· sink type, 12/24V DC, 32 points Output side ····· triac output, 100-240V AC, 0.6A/point, 24 points			
	AJ35PTF-56DT	I/O unit Input side····· sink type, 12/24V DC, 32 points Output side ····· transistor output, sink type, 12/24V DC, 0.5A/point, 24 points			

MELSECNET/MINI-S3 Equipment

Name	Type	Description	No. of Occupied Stations/ No. of Occupied Stations	Usable Master Module Modes	
				Extension mode	I/O dedicated mode
Data Link Module (for optical data link, twisted-pair data link)	AJ72PT35	Allows the building block type I/O modules to be used as remote I/O units. • Max. number of modules: 8 • I/O points: 128 points • Number of occupied stations: 4, 8, 12, 16 (selected by switch)	See left	○	○
Partial refresh type remote I/O unit (for optical data link, twisted-pair data link)	AJ35PTF-128DT	I/O unit Input side····· sink type, 12/24V DC, 64 points Output side ····· transistor output, 12/24V DC, 100mA/ point, 64 points	4 stations	○	○
RS-232C interface unit (for optical data link, twisted-pair data link)	AJ35PTF-R2	Interface for external equipment conforming to RS-232C interface specifications 1 RS-232C channel General I/O····· each 4 points	4 stations	○	—
Mount type operating box (for optical data link, twisted-pair data link)	AJ35PT-OPB-M1	Character display, key input unit Character display····· 3 lines by 30 columns LCD Sheet keys ····· 8 keys Touch keys····· 24 keys LED display ····· 8	4 stations	○	—
Portable type operating box (for twisted-pair data link)	AJ35T-OPB-P1				
Joint box (for twisted-pair data link)	AJ35T-JB AJ35T-JBR	Connects the portable type operating box to the MINI-S3 link when necessary.	—	○	—
MELSEC-F series PC connection interface unit	F-16NP (for optical data link)	Interface unit for connecting the MELSEC-F series PC to the MINI-S3 link.	2 stations	○	○
	F-16NT (for twisted-pair data link)				
FR-Z200 series transistorized inverter connection interface board	FR-ZDL	Interface board for connecting the Mitsubishi FR-Z200 series transistorized inverter to the MINI-S3 link.	4 stations	○	○
Twisted-pair shield cable	—	Twisted-pair cable for MINI-S3 link User prepared in accordance with Section 4.4.	—	○	○
Optical fiber cable	—	Optical fiber cable for MINI-S3 link User prepared in accordance with Section 4.3.	—	○	○

MELSECNET/MINI-S3 Equipment

<table>
<tr><th rowspan="2">Name</th><th rowspan="2">Type</th><th rowspan="2">Description</th><th rowspan="2">No. of Occupied Stations/ No. of Occupied Stations</th><th colspan="2">Usable Master Module Modes</th></tr>
<tr><th>Extension mode</th><th>I/O dedicated mode</th></tr>
<tr><td>Optical fiber cable connector</td><td>CA9104AP</td><td>1-core connector for use with the optical fiber cable. Consists of the following:
<table>
<tr><th>Equipment</th><th>Quantity</th></tr>
<tr><td>Housing</td><td>1</td></tr>
<tr><td>Ferrule</td><td>1</td></tr>
<tr><td>Sleeve</td><td>1</td></tr>
</table>
The optical fiber cable connector and assembling tool kit are only used with the plastic fiber.</td><td rowspan="2">—</td><td rowspan="2">–</td><td rowspan="2">–</td></tr>
<tr><td>Assembling tool kit</td><td>CT9004P</td><td>For assembling optical fiber cable connectors. Consists of the following:
<table>
<tr><th>Equipment</th><th>Type</th><th>Quantity</th></tr>
<tr><td>Fiber stripper</td><td>ST1000</td><td>1</td></tr>
<tr><td>Fiber cutter</td><td>CV1000</td><td>1</td></tr>
<tr><td>Fiber clamper</td><td>FC1000</td><td>1</td></tr>
<tr><td>Replacement blade for cutter</td><td>—</td><td>1</td></tr>
</table></td></tr>
<tr><td>Optical power tester</td><td>HT-101P</td><td colspan="3">For measuring the luminous energy of the MINI-S3 link.</td></tr>
</table>

Peripheral Equipment

Unit	Description	Type	Corrent Consumption 5 VDC	Corrent Consumption 24 VDC	Remarks
Programming unit with CRT	Intelligent GPP	A6GPP-SET	—	—	○ Consists of the following models:
					Type: A6GPP — Remarks: ○ Programming unit with CRT ○ Equipped with ROM writer, FDD and printer interface functions.
					Type: *3 SW□GP-GPPA — Remarks: A series system disk
					Type: SW□GP-GPPK — Remarks: K series system disk
					Type: SW0-GPPU — Remarks: User disk (3.5 inch, formatted)
					Type: AC30R4 — Remarks: Cable for connection of CPU and A6GPP 3 m/9.84 ft length
	Composite video cable	AC10MD	—	—	Cable for connection of GPP and expanded monitor display. 1m/3.28 ft length.
Programming unit with LCD	Handy graphic programmer	A6HGP-SET	—	—	○ Consists of the following models:
					Type: A6HGP — Remarks: ○ Programming unit with LCD ○ Equipped with FDD, printer interface and memory card interface functions.
					Type: *3 SW□-HGPA — Remarks: A series system disk
					Type: SW□-HGPK — Remarks: K series system disk
					Type: SW0-GPPU — Remarks: User disk (3.5 inch, formatted)
					Type: AC30R4 — Remarks: Cable for connection of CPU and A6HGP 3 m/9.84 ft length
Programming unit with plasma display	Plasma handy programmer	A6PHPE-SET	—	—	○ Consists of the following models:
					Type: A6PHP — Remarks: ○ Programming unit with plasma display ○ Equipped with FDD, printer interface and memory card interface functions.
					Type: *3 SW□GP-GPPA — Remarks: A series system disk
					Type: SW□GP-GPPK — Remarks: K series system disk
					Type: SW0-GPPU — Remarks: User disk (3.5 inch, formatted)
					Type: AC30R4 — Remarks: Cable for connection of CPU and A6PHP 3 m/9.84 ft length
Common to programming units with CRT and LCD	RS-422 cable	AC30R4	—	—	Cable for connection of CPU and A6GPP/A6HGP/A6PHP — 3 m/9.84 ft length
		AC300R4	—	—	Cable for connection of CPU and A6GPP/A6HGP/A6PHP — 30 m/98.4 ft length
	User disk	SW0-GPPU	—	—	User disk (3.5 inch, formatted) for storing programs
	Cleaning disk	SW0-FDC	—	—	Cleaning disk for disk drive

Peripheral Equipment

Unit	Description	Type	Corrent Consumption		Remarks
			5 VDC	24 VDC	
Printer	Printer	K6PRE	—	—	○ For print out of program ladder diagrams and lists.
		K7PRE	—	—	
	RS-232C cable	AC30R2	—	—	Cable for connection of A6GPP/A6PHP/A6HGP and printer. 3 m/9.84 ft length.
	Printer paper	K6PR-Y	—	—	Paper for K6PRE. 9 inch. Available in units of 2000.
Programming unit	Programming unit	*3 A7PU	0.3 A	—	○ Connected to the CPU directly or via cable to read and write programs. Equipped with MT function. ○ The A7PU is supplied with a cable for connection of the A7PU and audio cassette recorder.
	RS-422 cable	AC30R4 AC300R4	—	—	Cable for connection of CPU and A7PU. 3 m(9.84 ft)/30 m(98.4 ft) length.
P-ROM writer unit	P-ROM writer unit	*3 A6WU	0.8 A	—	○ Used to store programs onto ROM and read programs from ROM to the CPU. ○ Connected to the CPU directly or via the AC30R4 cable.
	RS-422 cable	AC30R4 AC300R4	—	—	Cable for connection of CPU and A6WU. 3 m(9.84 ft)/30 m(98.4 ft) length.

4. GENERAL OPERATION

4.1 Overview

The A7BDE-A3N-PT32S3A/B.C and A7LU1EP21/R21 option cards enable a Mitsubishi A3N Programmable Controller, and MELSEC-NET - MELSECNET/MINI-S3 interfaces to be installed directly into an IBM PC-AT® or compatible computer. The addition of the A7BDE-A3N-PT32S3 option cards enables fast access to the installed A3N CPU, and to the stations of MELSECNET or MELSECNET/MINI. The PC may then be configured as the master station of both networks.

To link the A7BDE-A3N-PT32S3 option cards with the PC's operating system and application programs, a device driver program is installed. This supervises interrupts, and the transfer of data to and from the application program. The device driver provides various functions for communication and control of the option cards.

The following sections give information on the software configuration, PC-A7BDE-A3N-PT32S3 configuration, and the A3N CPU (SCPU) operation.

4.2 Software Configuration

The following diagram shows the software configuration, the various components, and their relationship to each other.

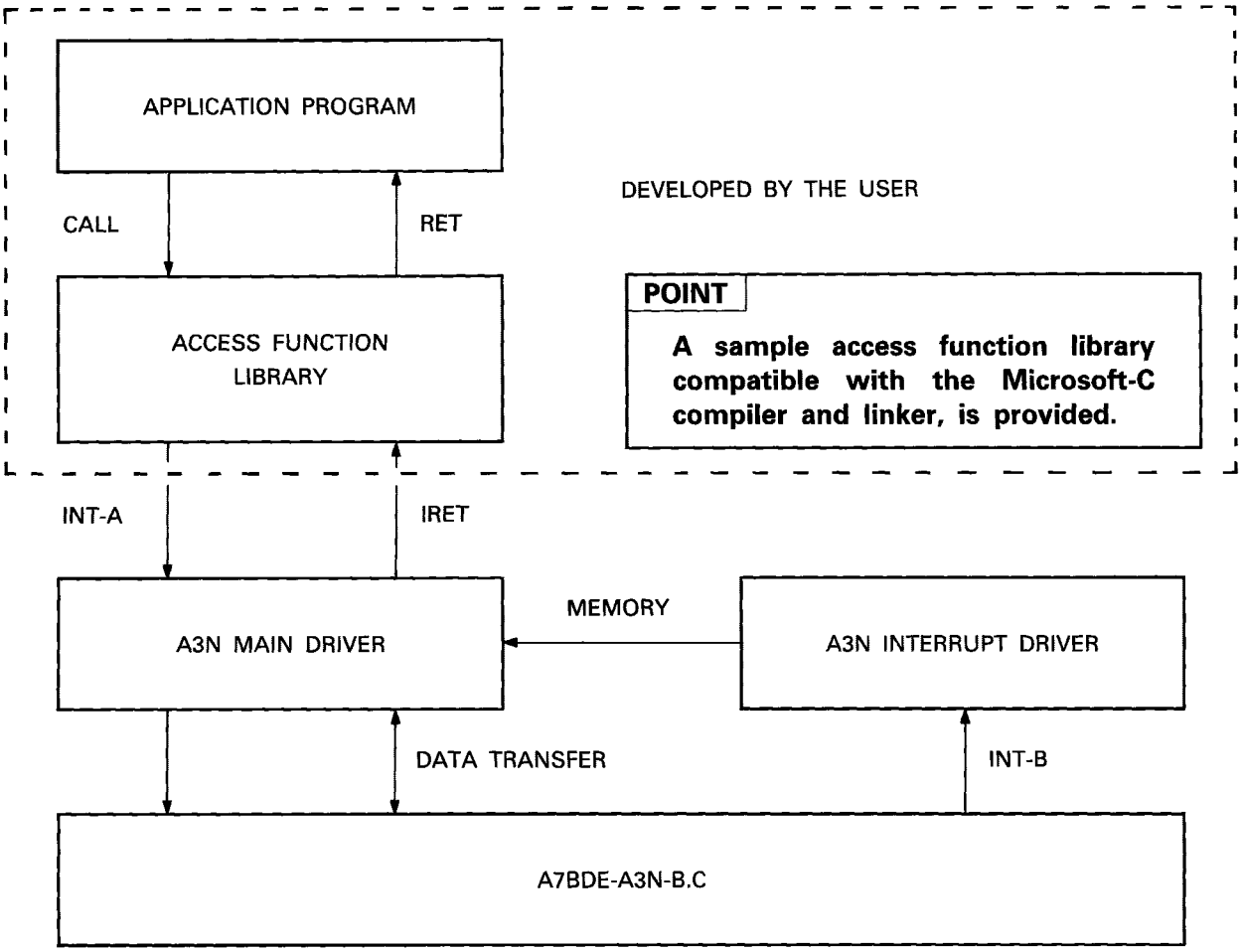
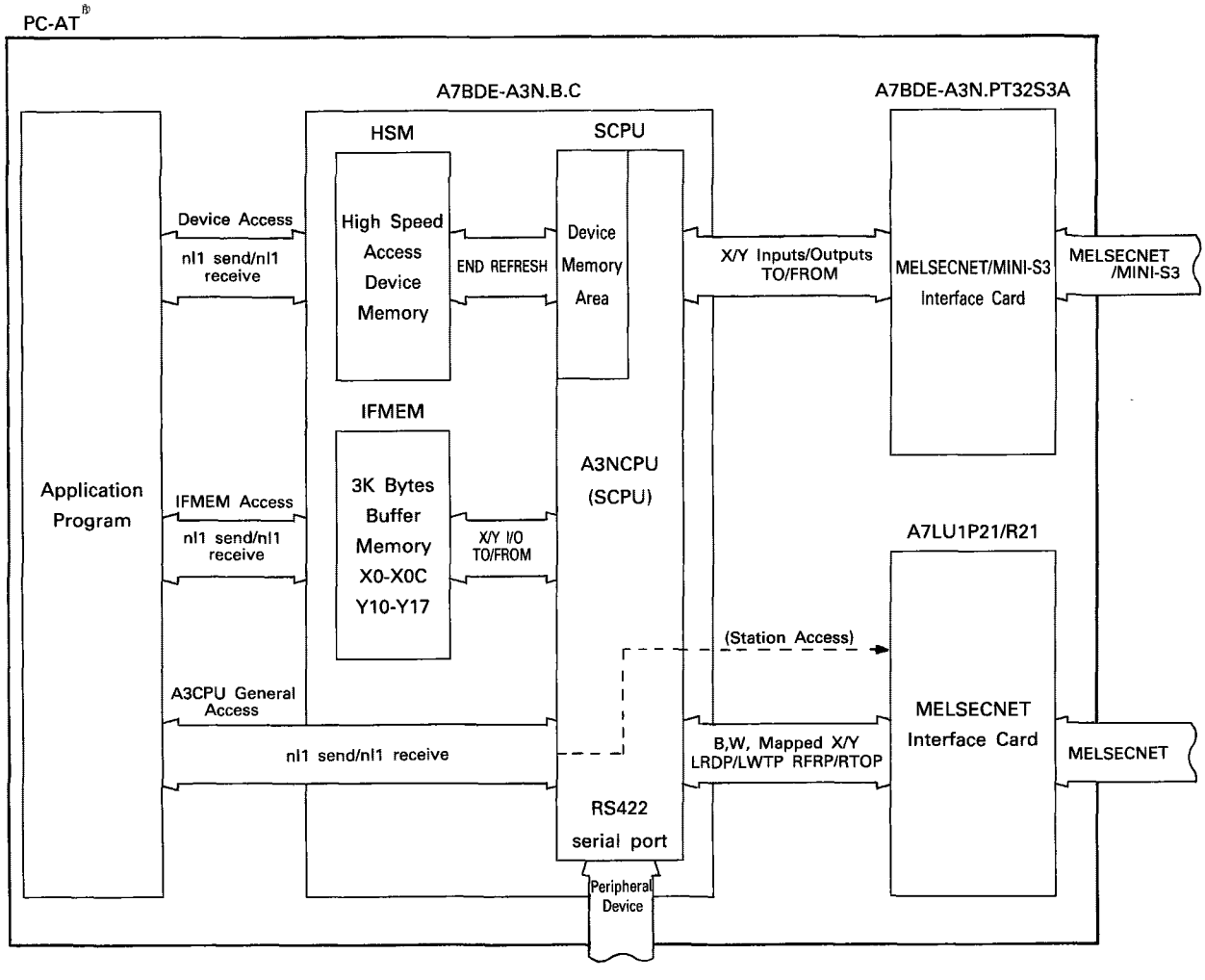


Diagram Key

Application	User-created application program requiring access to the A7BDE-A3N-PT32S3 Programmable Controller.
Access Function Library	User-created function library, providing specific access subroutines.
A3N Main Driver	Accesses/requests A7BDE-A3N memory areas.
A3N INTERRUPT Driver	Receives INTERRUPT (IRQ) reply from the A7BDE-A3N-B.C.
A7BDE-A3N-B.C	A3N CPU Programmable Controller Option card.

4.3 Hardware Configuration and Operation

The diagram below shows the general configuration and communication paths of the three option cards (A7BDE-A3N-PT32S3A/B.C A7LU1EP21/R21), when installed inside a PC.



From the diagram it can be seen that the A7BDE-A3N-B.C has three main components: the high-speed-access device memory, the IFMEM, and the SCPU. Their general operation is covered in the proceding sections.

The application program may directly access the high-speed device memory, the IFMEM, and the general memory areas (e.g. sequence program) of the SCPU, and stations of MELSECNET. Communication with stations of MELSECNET/MINI is by means of the SCPU sequence program, i.e. using FROM/TO instructions. The SCPU may also be accessed by a peripheral programming device, e.g. A6GPP, via the RS422 serial port.

4.4 The IFMEM

4.5 IFMEM I/O

X, Y00 to 1F are assigned for data transmission between the SCPU and IFMEM.

(1) Input signals from the IFMEM to the SCPU are X00 to X1F, —32 points.

Input No.	Content
X00 to X0A	General purpose input Turned ON/OFF by the PC Application program and read by the SCPU.
X0B	PC Ready On·····PC-AT® System Ready. Off·····PC-AT® System Not Ready.
X0C to X1F	Used by operating system. Not to be included in sequence programs.

(2) Output signals from the SCPU to the IFMEM are Y00 to Y1F, 32 points.

Output No.	Content
Y00 to Y0F	May be used in place of internal relay (M).
Y10 to Y15	General purpose output Turned ON/OFF by the SCPU, and read by the PC Application Program.
Y16	High-speed access memory refresh enable signal ON: Start high-speed access memory refresh OFF: Stop high-speed access memory refresh
Y17 to Y1F	Used by operating system. Not to be included in sequence programs.

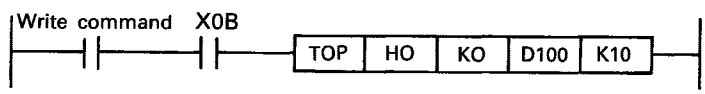
4.6 IFMEM Access by the Sequence Program

The IFMEM may be regarded as a 32 point special function unit that has been loaded into the first slot of a rack system. The IFMEM has a buffer memory of 3K words (H0 to H3FF), accessable by FROM/TO instructions, and also general purpose or dedicated I/O (XY00 TO XY1F).

When accessing the buffer memory with the sequence program, always use the FROM/TO enable signal, input X0B, as an interlock. This prevents simultaneous access by the sequence and application programs. Should the sequence program try to transfer data to or from the IFMEM buffer memory when the interlock input X0B is OFF, an error code and message, "41 - SPECIAL UNIT DOWN" will be indicated by the SCPU self diagnostics.

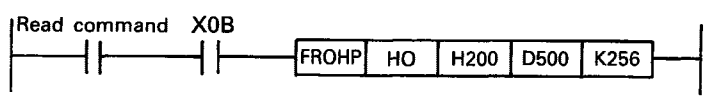
Example 1

The following is an example of D100 to 109 data being written to buffer memory addresses 0 to 9.



Example 2

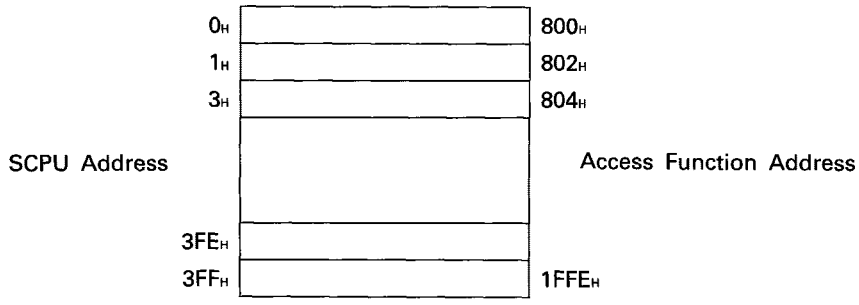
The following is an example of 256 words from the buffer memory address 200H to 2FFH being read to D500 to 755.



4.7 IFMEM Access by the PC Application Program

The IFMEM may be directly accessed by the PC application program. Data may be transferred to and from the buffer memory, and the status of the IFMEM general purpose I/O, six outputs (Y10 to Y15), and ten inputs (X00 to X0A), may be controlled as required. Details of the specific access functions are provided in programming section.

Access to the buffer memory by the sequence program is in units of words, and the memory addresses are HO to H3FF. However, the PC application program may only access the IFMEM buffer in units of bytes, so the corresponding addresses are 0x800 to 0x1FFF (C notation for hexadecimal), i.e.



POINT

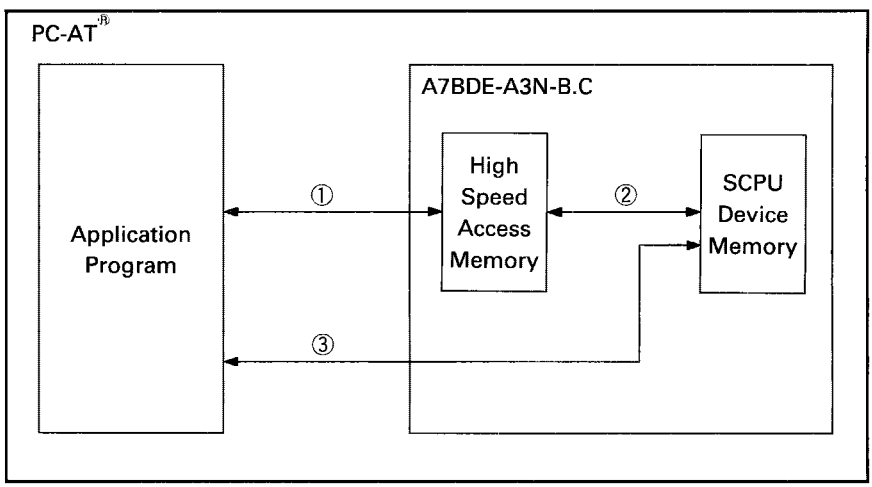
When specifying addresses with personal computer functions, the least significant first byte of the buffer memory becomes the smaller number.
For example, if address 0 of the buffer memory is to be read or written using the personal computer function, specify the least significant byte as 800H and the most significant byte as 801H.

4.8 The High-Speed-Access Device Memory

The high-speed access device memory is used as an interface when transferring data to and from the PC application program and the SCPU device memory area, i.e. monitoring or controlling the status of the devices X, Y, M, L, S, B, F, T/C (contact, coil, and present value), D, and W registers of the SCPU. Details of the specific access functions are provided in the programming section.

4.9 Data Transfer

The diagram below shows the general sequence of communications between the application program, the high-speed access device memory, and the SCPU device memory.



(1) Data is transferred to and from the PC application program to the high-speed access device memory. Since access is to the high-speed access device memory, and not the SCPU device memory, there are no communication delays due to the scan of the SCPU. The SCPU device memory can only be accessed after the END or COM instructions have been processed. The high-speed memory allows data transfer at any time during the SCPU scan. Access is only restricted during device refresh.

(2) Data is transferred to and from the high-speed memory and the SCPU. The devices are refreshed after the SCPU executes the END or COM instruction of the sequence program.

(3) The PC application program may also directly access the SCPU device memory, but only after the END or COM instruction has been executed. This produces a delay, and subsequently longer processing times than when accessing the high-speed memory.

POINT

For device refresh of the high-speed access device memory to occur, the PC application program must have set the transfer parameters, and the sequence program must have switched the output Y16 ON.

4.10 High Speed Device Memory Operation

To minimize the device refresh time of the high-speed memory, the ranges of devices to be updated may be specified by the PC application program. The range parameters are set using one of the access functions. Further details are provided in the programming section. There are two types of device ranges to be specified:

Data ranges to be transferred from the SCPU to the high-speed memory.

Data ranges to be transferred from the high-speed memory to the SCPU.

Please note that all data will be transferred for the devices SpD and SpM (special registers and relays) whatever the range setting.

To start the refresh processing, set the refresh enable signal (Y16) of the high-speed access memory to ON. To stop the refresh processing, set the enable signal for the high-speed access memory to OFF. The data contained in the high speed access memory immediately prior to stopping will be retained.

The time taken to refresh the high-speed memory (Tm) may be calculated from using the formula below. Please note that (Tm) is dependent on the device range settings.

$$Tm = 5610 + T_{M-S} + T_{S-M} \ (\mu s)$$

$$T_{M-S} = 2.6 \times \left(\frac{n_1}{8} + n2\right) \ (\mu s)$$

$$T_{S-M} = 4.9 \times \frac{n_3}{8} + 2.6 \times n_3 \ (\mu s)$$

T_{M-S} : Refresh time from the H.S.M to the SCPU.
T_{S-M} : Refresh time from the SCPU to the H.S.M.
n_1 : Total number of bit devices transmitted from the H.S.M. to the SCPU.
n_2 : Total number of word devices transmitted from the H.S.M. to the SCPU.
n_3 : Total number of bit devices transmitted from the SCPU to the H.S.M.
n_4 : Total number of word devices transmitted from the SCPU to the H.S.M.

When timer (T) and counter (C) device ranges have been specified to be refreshed, please note that contact points, coils and present values of the timer (T) and counter (C) are also refreshed.
Hence, the point numbers of n1, n2, n3 and n4 should be set as 2 points for n1 or n3, and 1 point for n2 or n4 for each point of the timer (T) and counter (C).
For example, when SCPU refreshes T0 through 255 for the H.S.M. 512 and 256 are set in n3 and n4 respectively.

4.11 The A7BDE-A3N-PT32S3A MELSECNET/MINI-S3 Master Station Interface

The A7BDE-A3N-PT32S3A option card provides an interface to the network MELSECNET/MINI-S3 by acting as the master station. The functions of the A7BDE-A3N-PT32S3A are almost the same as those of the AJ71PT32 MELSECNET/MINI-S3 master station module, and are regarded by the SCPU to be loaded in the second slot (head address XY20). For further details, please refer to the MELSECNET/MINI-S3 Master Station User's Manual.

The communications I/O between the ACPU and the A7BDE-A3N-PT32S3A MELSECNET/MINI-S3 master station option card are given in the table below.

(1) I/O Dedicated Mode

Device No.	Signal Name	Device No.	Signal Name
X20	Hardware error	Y20 to Y37	Not used
X21	MINI link communication in progress		
X22	Not used	Y38	MINI link communication start
X23		Y39	Not used
X24		Y3A	FROM/TO instruction response specification
X25	Test mode		
X26	MINI link error detect	Y3B	Error station data link specification
X27	MINI link communication error		
		Y3C	Not used
X28 to X3F	Not used	Y3D	Error reset
		Y3E Y3F	Not used

POINT

1) **The A7BDE-A3N-PT32S3A uses a D sub-connector for the twisted-pair data link, not screw terminals as with the AJ71PT32. Details on the construction are provided in the appendix.**

2) **It is not possible to monitor the I/O status of the remote I/O station with the I/O monitoring LEDs of the remote I/O station and the monitor station number setting switches. Create a sequence program to confirm the I/O status.**

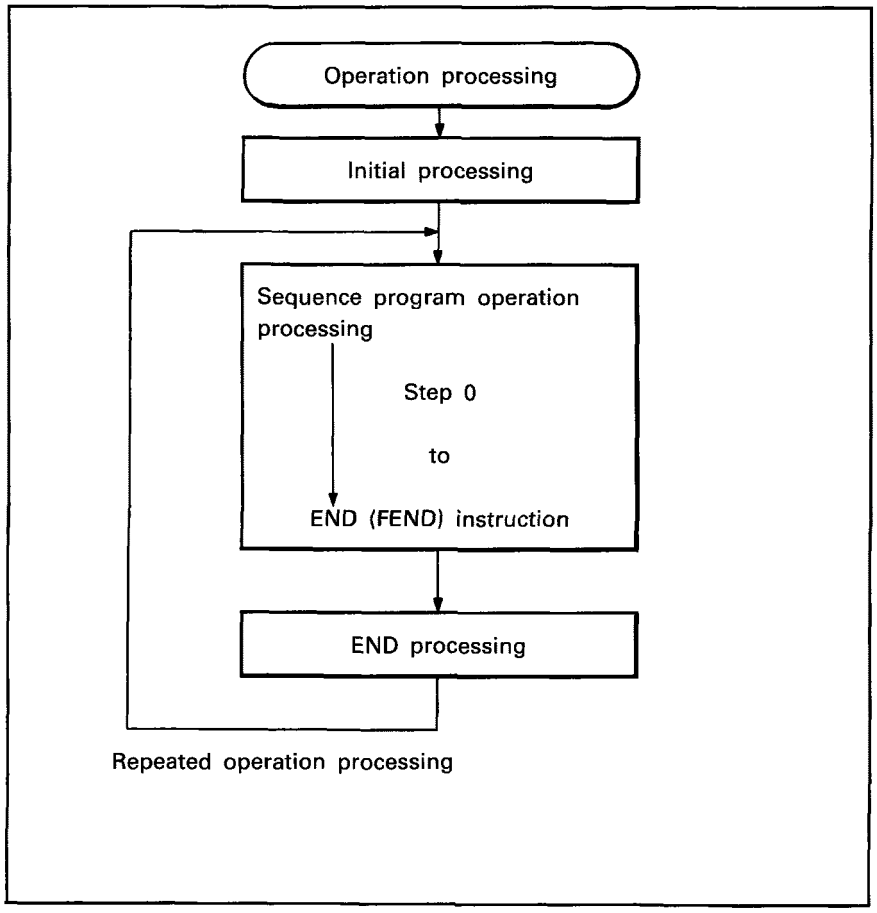
(2) I/O list for the extension mode
A list of I/O signals used when the A7BDE-A3N-PT32S3A is being used in the extension mode is given below.

Device No.	Signal		Device No.	Signal	
X20	Transmit complete signal	For remote terminal unit No. 1	Y20	Transmit request signal	For remote terminal unit No. 1
X21	Read request signal		Y21	Read complete signal	
X22	Transmit complete signal	For remote terminal unit No. 2	Y22	Transmit request signal	For remote terminal unit No. 2
X23	Read request signal		Y23	Read complete signal	
X24	Transmit complete signal	For remote terminal unit No. 3	Y24	Transmit request signal	For remote terminal unit No. 3
X25	Read request signal		Y25	Read complete signal	
X26	Transmit complete signal	For remote terminal unit No. 4	Y26	Transmit request signal	For remote terminal unit No. 4
X27	Read request signal		Y27	Read complete signal	
X28	Transmit complete signal	For remote terminal unit No. 5	Y28	Transmit request signal	For remote terminal unit No. 5
X29	Read request signal		Y29	Read complete signal	
X2A	Transmit complete signal	For remote terminal unit No. 6	Y2A	Transmit request signal	For remote terminal unit No. 6
X2B	Read request signal		Y2B	Read complete signal	
X2C	Transmit complete signal	For remote terminal unit No. 7	Y2C	Transmit request signal	For remote terminal unit No. 7
X2D	Read request signal		Y2D	Read complete signal	
X2E	Transmit complete signal	For remote terminal unit No. 8	Y2E	Transmit request signal	For remote terminal unit No. 8
X2F	Read request signal		Y2F	Read complete signal	
X30	Transmit complete signal	For remote terminal unit No. 9	Y30	Transmit request signal	For remote terminal unit No. 9
X31	Read request signal		Y31	Read complete signal	
X32	Transmit complete signal	For remote terminal unit No. 10	Y32	Transmit request signal	For remote terminal unit No. 10
X33	Read request signal		Y33	Read complete signal	
X34	Transmit complete signal	For remote terminal unit No. 11	Y34	Transmit request signal	For remote terminal unit No. 11
X35	Read request signal		Y35	Read complete signal	
X36	Transmit complete signal	For remote terminal unit No. 12	Y36	Transmit request signal	For remote terminal unit No. 12
X37	Read request signal		Y37	Read complete signal	
X38	Transmit complete signal	For remote terminal unit No. 13	Y38	Transmit request signal	For remote terminal unit No. 13
X39	Read request signal		Y39	Read complete signal	
X3A	Transmit complete signal	For remote terminal unit No. 14	Y3A	Transmit request signal	For remote terminal unit No. 14
X3B	Read request signal		Y3B	Read complete signal	
X3C	Reserved		Y3C	Reserved	
X3D			Y3D		
X3E			Y3E		
X3F			Y3F		
X40	Hardware fault		Y40		
X41	MINI-S3 link communicating		Y41		
X42	Reserved		Y42		
X43	Receive data clear completion (for AJ35PTF-R2)		Y43	Receive data clear request (for AJ35PTF-R2)	
X44	Remote terminal unit error detection		Y44	Remote terminal unit error detection clear	
X45	Test mode		Y45	Reserved	
X46	MINI-S3 link error detection		Y46		
X47	MINI-S3 link communication error		Y47		
X48	ROM error		Y48	MINI-S3 link communication start	
X49	Reserved		Y49	Reserved	
X4A			Y4A	FROM / TO instruction response designation	
X4B			Y4B	Faulty station data clear designation	
X4C			Y4C	Switching buffer memory channel	
X4D			Y4D	Error reset	
X4E			Y4E	Reserved	
X4F			Y4F		

4.12 The SCPU

4.13 SCPU Operation Processing

The general operation processing of the SCPU is given in the flow chart below.



4.14 Initial processing

Initiates the sequence program operation processing, i.e. the following processing is executed when the power is turned on at the PC or the SCPU is reset.
The amount of time required for initial processing varies depending on system configuration, but is normally 2 to 4 seconds.

(1) I/O module initialization
Resets and initializes the Remote I/O modules.

(2) Data memory clear

(a) If unlatched, clears the data memory.
The latch setting is made with a parameter using the peripheral equipment.

(b) Clears Y data content where "Y" is the memory area of non-loaded modules being used as internal relay M.

(3) Link parameter setting
Data link is started when link parameter data is set in the data link module and MELSECNET is the master station.

(4) I/O address assignment
Automatically assigns I/O addresses to the I/O modules.

(5) I/O module data entry
Enters the types of I/O modules loaded in the Remote units. I/O module data is used to verify I/O modules.

(6) Self-diagnosis
The SCPU conducts self-checks when it is powered up or reset.
For further details, see Section 4.21.

4.15 END Processing

Returns the SCPU to step 0 in the repeated operation processing. The following processing is performed after the END (FEND) instruction is executed.

(1) Self-diagnosis
Checks for blown fuse, I/O module verify error, low battery power, etc. For further details, see Section 4.21

(2) Timer/counter processing
Updates timer/counter present values and contract status. For further details, see sections 4.16 and 4.17.

(3) Constant scan processing
Allows the repeated operation processing to be initiated after the specified constant scan time (set to special data register D9020 is reached if the constant scan function is used.)

(4) Data communication processing with IFMEM
Transmits data between the SCPU and the IFMEM when a read/write request is given from the IFMEM.

(5) Refresh processing

(a) Link refresh processing
Executed when a link refresh request is received from the data link module of the MELSECNET.
For details concerning the link refresh timing, refer to the "MELSECNET Data Link System Reference Manual".

(b) High-speed access memory refresh processing
Executed between the SCPU device memory and the high-speed access memory. For details, refer to Section 4.8.

(6) Sampling trace processing
Stores the specified device status to the sampling trace area when the trace point of the sampling trace is "every scan (after the execution of the END instruction)".

(7) RUN/STOP switch position check
Changes the SCPU operating status in accordance with the RUN/STOP switch position.
For information concerning the transition processing of the RUN, STOP, PAUSE, and STEP-RUN operations, refer to section 4.20.

4.16 Timer Processing

The SCPU timers are up-counting timers that increment the present time value based on three timing periods, i.e. a 100 ms timer, a 10 ms timer, and 100 ms retentive timer.

*The 100 ms timer can be set between 0.1 and 3276.7 sec in 100 ms increments.

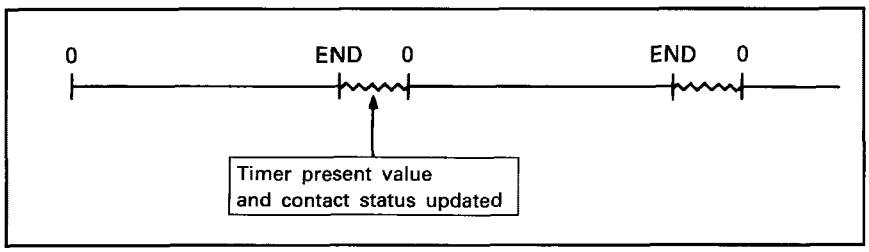
*The 10 ms timer can be set between 0.01 and 327.67 sec in 10ms increments.

*The 100 ms retentive timer retains its present value even if its coil is switched OFF. The timing can be set between 0.1 and 3276.7 sec in 100 ms increments.

(1) Timer present value and contact status update
When the timer coil is set ON by the OUT T[ ] instruction, the present value of the timer is updated after the END(FEND) instruction has been executed. The timer contacts close after the timer has timed out.
(a) 100ms timer, 10ms timer
When the input status is OFF, the timer coil is set to OFF, and after the END(FEND) instruction has been executed, the the present value of the timer is set to 0 and the contacts open.
(b) 100ms retentive timer
When the input status condition is OFF and the timer coil is set to OFF, the updating of the present value is terminated. However, the present value is still retained.

(2) RST T[ ] instruction execution
At the point the timer reset is executed by the RST instruction, the present value is set to 0 and the contacts open.
Even with the coils of the 100ms retentive timer set to OFF, the present value and contact status are maintained.
The RST T[ ] instruction is used to reset the 100ms retentive timer.

(3) OUT T[ ] jumped
If the OUT T[ ] instruction is jumped after the timer begins timing, it continues to time; the contacts are closed when the timer times out.



Timer Processing

POINT

Timer accuracies are as follows. For further details, refer to the ACPU Programming Manual.

Timer	Scan Time T	Accuracy
10 ms	T<10 ms	+2 scan time to −10 ms
10 ms	T≧10 ms	+2 scan time to −1 scan time
100 ms, 100 ms retentive	T<100 ms	+2 scan time to −100 ms
100 ms, 100 ms retentive	T≧100 ms	+2 scan time to −1 scan time

4.17 Counter Processing

The SCPU counter detects the leading edge of the input signal (OFF ON) and adds the present value. Two counters, normal and interrupt, are provided.

- The normal counter is used in main routine programs or subroutine programs.
- The interrupt counter is used in interrupt programs.

(1) Counter present value and contact status update

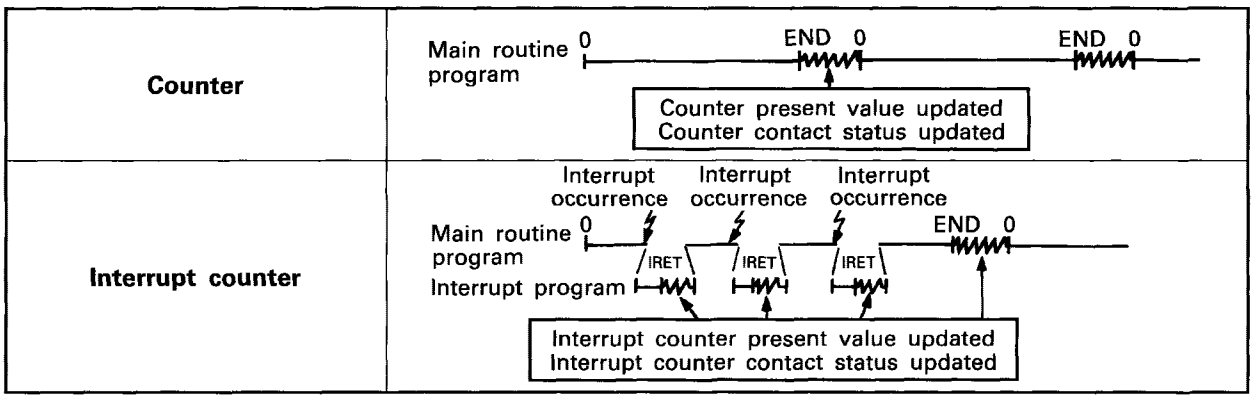
The OUT C [ ] instruction sets the counter coil to either ON/OFF. When the leading edge of the coil signal is detected, the present value is updated and the contacts close after the counter has counted out.

(a) Normal counter
The present value and contact status are updated after the END(FEND) instruction is executed.

(b) Interrupt counter
The present value and contact status are updated after the IRET instruction is executed.

(2) Opening counter contacts

The counter contacts are opened using the RST instruction. The present value is reset to 0 and the contacts are opened at the point the RST C [ ] instruction is executed.



Counter Processing

POINT

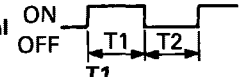
The maximum counting speed of the counter depends on the scan time. Counting is only possible if the input condition is ON/OFF for a period longer than that of one scan time. For further details, refer to the ACPU Programming Manual.

$$\text{Maximum counting speed } Cmax = \frac{n}{100} \times \frac{1}{ts} \text{ [times/sec]}$$

where, n = duty (%)

Duty is the ratio of the input signal's ON time to OFF time as a percentage.

Count input signal



If $T1 \leqq T2$ $\quad n = \frac{T1}{T1 + T2} \times 100$ (%)

If $T1 > T2$ $\quad n = \frac{T2}{T1 + T2} \times 100$ (%)

ts: Program scan time (sec)

4.18 Watch Dog Timer (WDT) Processing

(1) Watch dog timer
The watch dog timer is an internal timer used to detect errors of the SCPU's repeated operation function.
Default value is 200 ms. Timing can be set with parameters in 10ms increments in the range of 20 to 2000 ms.

(2) Operation
During each scan of program execution, the WDT checks for SCPU hardware errors and processing not completed within predefined periods. When either is detected, a WDT error is set, penetrating an alarm and stopping operation.

(3) Reset timing
The WDT is reset by the END instruction when SCPU operations have been completed within predefined periods.

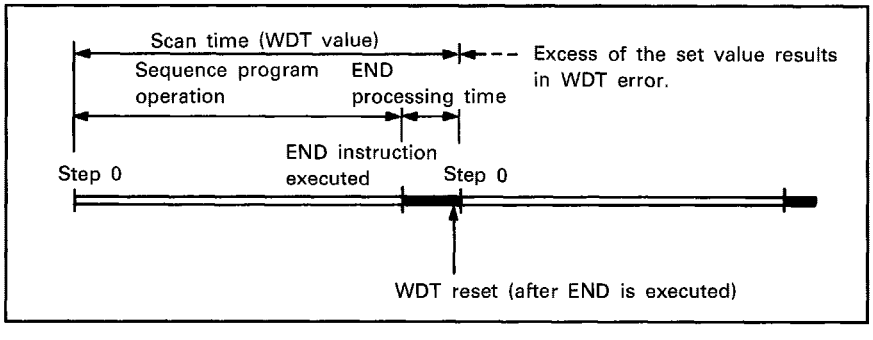
(4) Error
Two types of WDT error codes, 22 and 25, are provided.
Error code 22 indicates that the END instruction was executed outside of the predefined periods.
Error code 23 indicates that the END instruction was not executed due to operations entering an endless loop (such as from a CJ instruction). (For further details, refer to Section 7.7 Error Codes.)

(5) Operation at an occurrence of WDT error
When a WDT error occurs, the operational status of the SCPU becomes as follows:

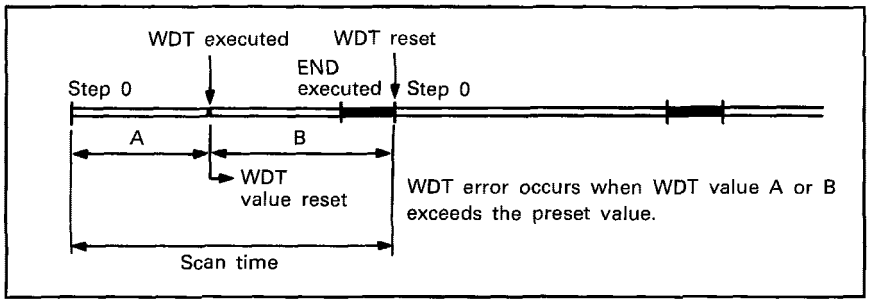
(a) SCPU operation ceases and all outputs are set to OFF.

(b) The RUN LED on the SCPU front panel flickers.

(c) "WDT ERROR" is displayed when the setting of the option board is set to board information.



(6) Resetting method
The WDT present value is reset when the WDT reset (WDT) instruction is executed in the sequence program.
The WDT restarts timing at 0.
The execution of the WDT instruction will not reset any scan time stored in D9017 to 9010.



(7) If the WDT error has occurred, check the error definition according to Section 10, reset, and remove the cause of error.

4.19 Operation Processing at Instantaneous Power Failure Occurrence

The SCPU detects any instantaneous power failure when the input line voltage to the power supply module falls below the defined value.
If the instantaneous power failure time is within the allowable value (10 ms), the SCPU performs instantaneous power failure processing as described below;

(1) Instantaneous power failure within 10 ms

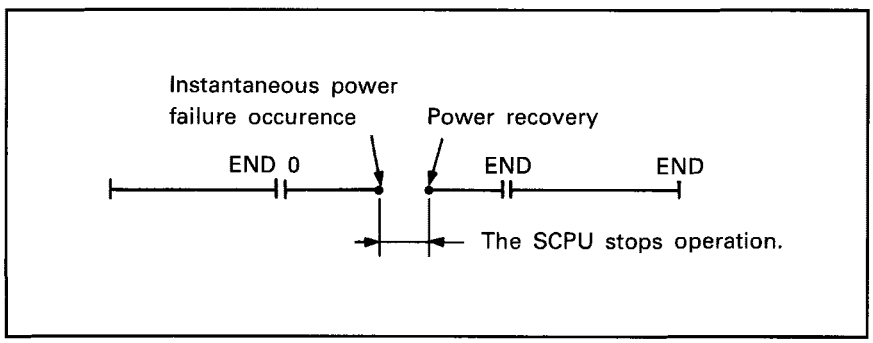
(a) The operation processing is stopped with the output retained.

(b) The operation processing is resumed when normal status is restored.

(c) The watch dog timer (WDT) keeps timing while the operation is at a stop.

For instance, if the WDT and scan time settings are 200 ms and 195 ms respectively, and instantaneous power failure of 10 ms will result in a WDT error.

(2) Instantaneous power failure over 10 ms
The SCPU is initialized and the sam operational process occurs that happen when the power is turned on or reset processing is undertaken.



Operation Processing at Occurrence ofInstantaneous Power Failure

4.20 RUN, STOP, PAUSE, STEP-RUN Operation Processing

The SCPU is operated in either of the RUN, STOP, PAUSE, and STEP-RUN states as described below.

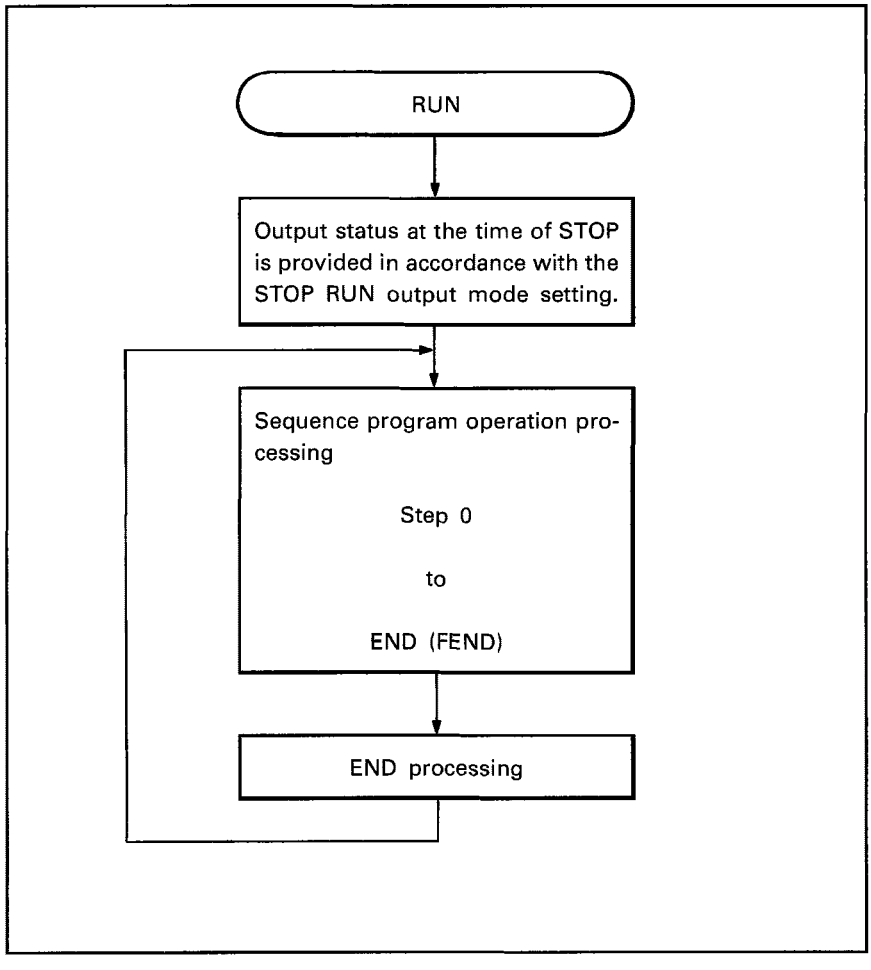
(1) RUN operation processing

RUN indicates repeated operation of the sequence program in order of step 0 to END(FEND) instruction, then back to step 0.

When the SCPU is set to RUN, the output status at the time of STOP is provided in accordance with the STOP RUN output mode setting in the parameter.

After the switching from STOP to RUN, the processing period is usually 1 to 3 seconds until the sequence program operation restarts, depending on system configuration.

The processing shown in the flow chart below is repeated until RUN is switched to another state.



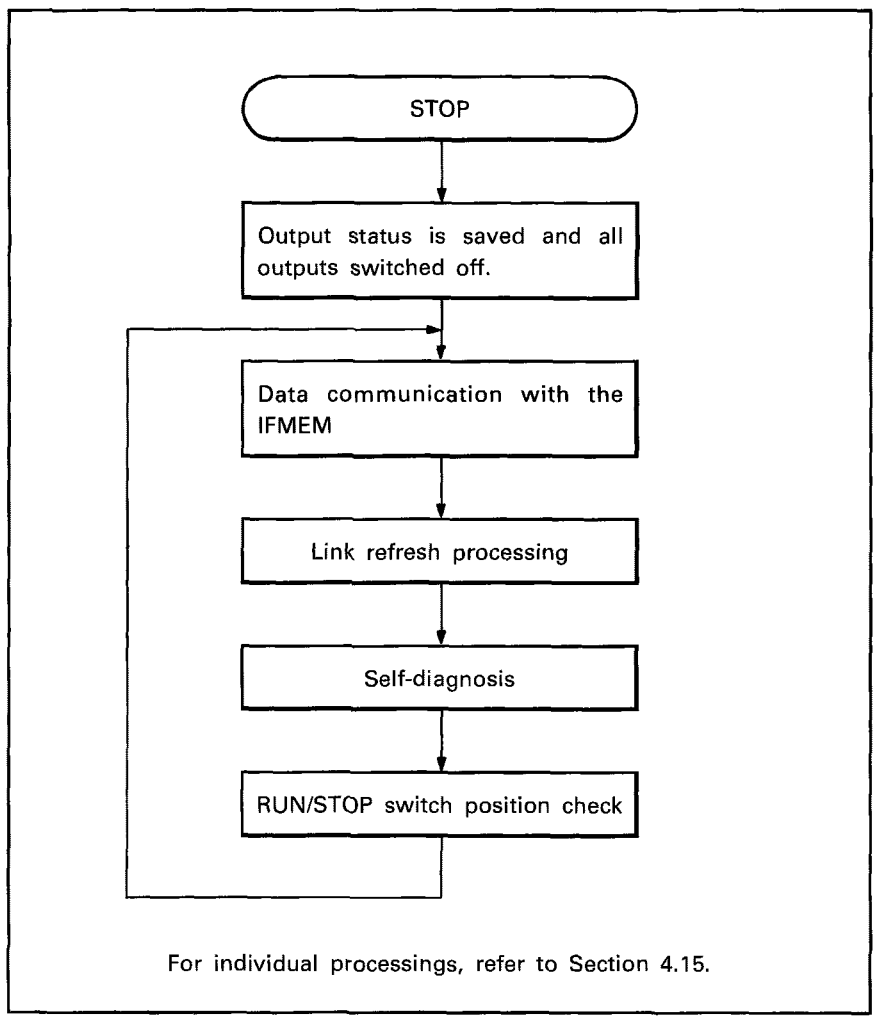
RUN Operation Processing

(2) STOP operation processing

STOP indicates a stop of the sequence program operation by using the RUN/STOP switch or remote STOP (Section).

When the SCPU is set to stop, the output status is saved and all outputs are switched off. Data other than the outputs (Y) is retained.

The processing shown in the flow chart below is repeated until STOP is switched to another state.



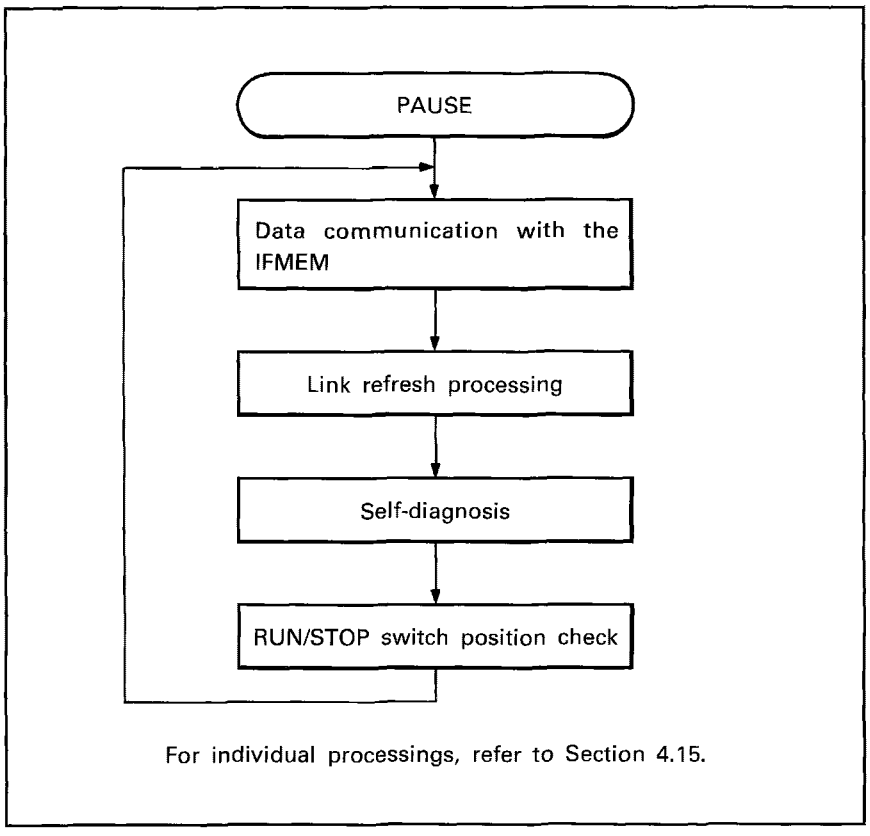
STOP Operation Processing

(3) PAUSE operation processing

PAUSE indicates a stop of the sequence program operation with the output and data memory status retained.

The processing shown in the flow chart below is repeated until PAUSE is switched to another state.

For the procedure to set the SCPU in the PAUSE state, refer to Section.



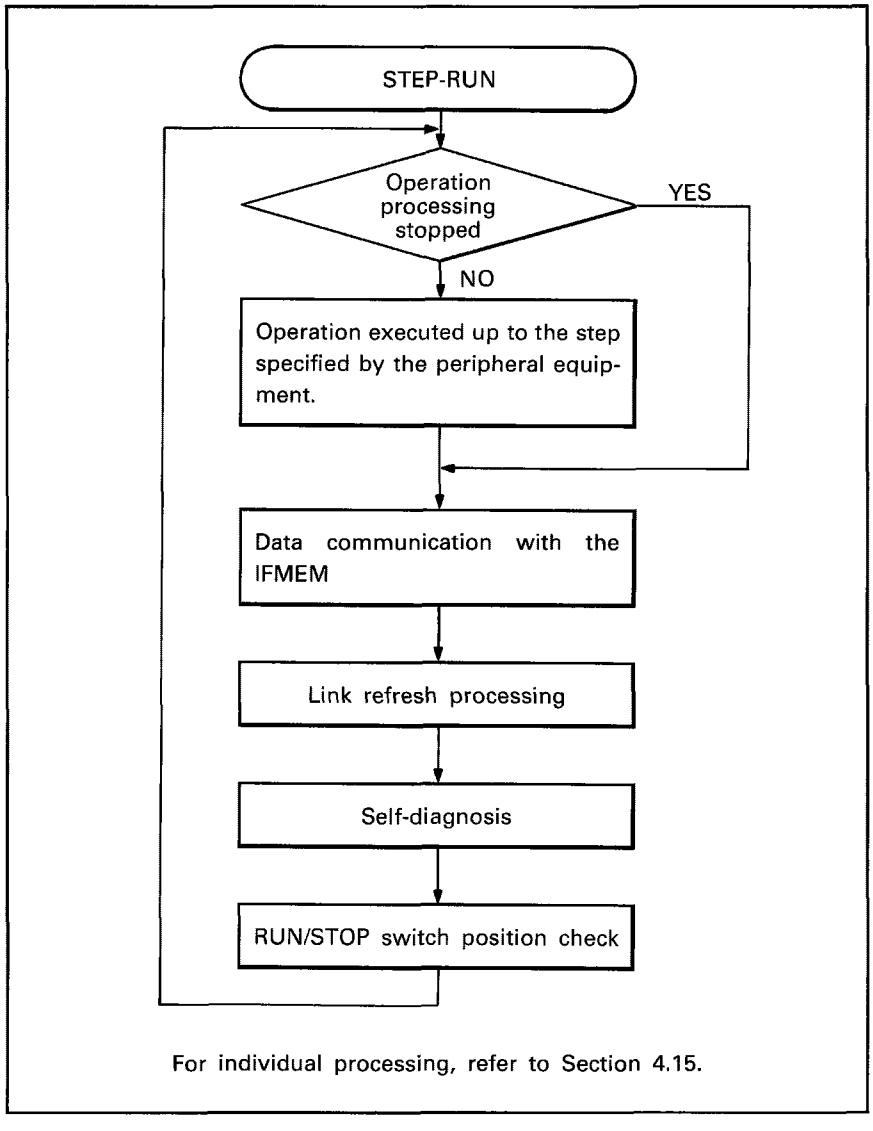
PAUSE Operation Processing

(4) STEP-RUN operation processing

STEP-RUN indicates a run mode which allows the sequence program operation processing to be stopped or continued per instruction using the peripheral equipment.

The execution state can be checked as the operation processing is stopped with the output and data memory status retained.

The processing shown in the flow chart below is repeated until STEP-RUN is switched to another state.



STEP-RUN Operation Processing

(5) Relation between RUN/STOP switch control and SCPU operation processing.

RUN/STOP Switch \ SCPU Operation Processing		Sequence Program Operation Processing	External Output	Data Memory (Y, M, L, S, T, C, D)	Remarks
RUN → STOP STEP-RUN → STOP		Stopped	Output status is saved by the OS and all outputs switched off.	Status at the time of STOP is retained.	
STOP → RUN		Started	Depends on the STOP RUN output mode set in the parameter.	Operation resumes in the status immediately prior to the STOP state.	
RUN → PAUSE (with M9040 on)		Stopped	Output status is retained.	Status immediately prior to the PAUSE status is retained.	When M9040 is off, the operation processing performed is the same as when the RUN/STOP switch is in RUN position. (The PAUSE status is not set.)
STOP → STEP RUN	Operation stopped from the peripheral	Operation stopped at the step specified from the peripheral.		Status immediately prior to operation stop.	
PAUSE → STEP RUN	Operation resumed from the peripheral.	Operation resumed following the operation stopped step.		Operation resumes in the status immediately prior to operation stop.	
PAUSE → RUN		Started	Operation resumes in the PAUSE output status.	Operation resumes in the status immediately prior to the PAUSE status.	

RUN/STOP Switch and SCPU Operation Processing

(6) Processing during stop of the sequence program operation.

RUN /STOP Switch \ Processing	Self-Diagnosis	Timer/ Counter Present Value and Contact Status Update	Constant Scan Processing (with constant scan set)	Communication with IFMEM	Link Refresh Processing	Sampling Trace Processing	RUN/ STOP Switch Position Check	Remarks
RUN (END processing)	Executed	Executed	Executed	Allowed	Allowed	Executed	Executed	
STOP	Executed	—	—	Allowed	Allowed	—	Executed	
PAUSE	Executed	—	—	Allowed	Allowed	—	Executed	
STEP -RUN	Executed	—	—	Allowed	Allowed	—	Executed	END processing is performed when the END (FEND) instruction is executed during STEP-RUN. In this case, the 10ms timer present value is incremented by 1 every scan and the 100ms timer present value is incremented by 1 every 10 scans.

Processing during Program Operation Stop

4.21 SCPU Self-Diagnosis

The self-diagnosis function detects the occurrence of abnormal conditions within the CPU.

The special function modules self-check for error at power on and during run. When any error is detected, the CPU indicates the error and stops operation to prevent faults and ensure reliable operation.

At error detection:

The CPU may operate in either of two modes. These are the processing stop mode and the processing continue mode. In the processing continue mode, the CPU may be able to continue step processing for some types of errors, according to the parameter settings.

The occurrence and content of the error are stored in special relay (M) and special register (D). These should be used in the program, especially when in the continue mode, to prevent malfunction of the programmable controller or machinery.

If the self-diagnosis function is in the processing stop mode, operation is stopped at the point the error is detected and all outputs (Y) are set to OFF.

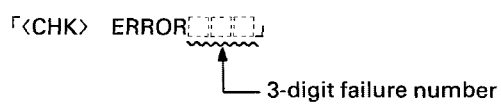
If the self-diagnosis function is in the processing continue mode, the program is executed continuously except for the portion in which the error occurred.

When an I/O module verify error is detected, processing continues with the I/O addresses used prior to the error. For self-diagnosed errors, see the table over page.

POINT

(1) The two conditions listed in columns "CPU Status" and "RUN" LED Status of the RUN/STOP Switch and SCPU Processing Table can be changed by settings of peripheral equipment.

(2) The LED displays the message shown below only when an error has been detected using the "CHK" instruction in the "Processing Check Error". The message is displayed using board information set by the option board.



4.22 Self Diagnosis Function Table

Diagnosis		Diagnosis Timing	CPU Status	"RUN" LED Status	Error Message (Peripheral Device)
Memory error	Instruction code check	When corresponding instruction is executed	Stop	Flicker	INSTRCT. CODE ERR.
	Parameter setting check	When power is switched on or reset performed When switched from STOP/PAUSE to RUN/STEP-RUN			PARAMETER ERROR
	No END instruction	When M9056 or M9057 is switched on When switched from STOP/PAUSE to RUN/STEP-RUN			MISSING END INS.
	Instruction execution disable	When CJ, SCJ, JMP, CALL(P), FOR to NEXT instruction is executed When switched from STOP/PAUSE to RUN/STEP-RUN			CAN'T EXECUTE (P)
	Format (CHK instruction) check	When switched from STOP/PAUSE to RUN/STEP-RUN			CHK FORMAT ERR.
	Instruction execution disable	When interrupt occurs When switched from STOP/PAUSE to RUN/STEP-RUN			CAN'T EXECUTE (I)
	No memory cassette	When power is switched on or reset performed			CASSETTE ERROR
CPU error	RAM check	When power is switched on or reset performed When M9084 is switched on during STOP	Stop	Flicker	RAM ERROR
	Operation circuit check	When power is switched on or reset performed			OPE. CIRCUIT ERR.
	Watch dog error check	When END instruction is executed			WDT ERROR
	END instruction unexecution	When END instruction is executed			END NOT EXECUTE
	Endless loop executed	Always			WDT ERROR
I/O error	I/O unit verify	When END instruction is executed (Not checked when M9084 or M9094 is on)	Stop / Run	Flicker / On	UNIT VERIFY ERR.
	Fuse blow	When END instruction is executed (Not checked when M9084 or M9094 is on)			FUSE BREAK OFF.
Special function module error	Control bus check	When FROM, TO instruction is executed	Stop	Flicker	CONTROL-BUS ERR.
	Special function unit error	When FROM, TO instruction is executed			SP. UNIT DOWN
	Link module error	When power is switched on or reset performed When switched from STOP/PAUSE to RUN/STEP-RUN			LINK UNIT ERROR
	I/O interruption error	When interrupt occurs			I/O INT. ERROR
	Special function unit assignment error	When power is switched on or reset performed When switched from STOP/PAUSE to RUN/STEP-RUN			SP. UNIT LAY. ERR.

Diagnosis		Diagnosis Timing	CPU Status	"RUN" LED Status	Error Message (Peripheral Device)
Special function module error	Special function module error	When FROM, TO instruction is executed	Stop / Run	Flicker / On	SP. UNIT ERROR
	Link parameter error	When power is switched on or reset performed When switched from STOP/PAUSE to RUN/-STEP-RUN	Run	On	LINK PARA. ERROR
	Battery low	Always (Not checked when M9084 is on)	Run	On	BATTERY ERROR
Operation check error		When corresponding instruction is executed	Stop / Run	Flicker / On	OPERATION ERROR

4.23 SCPU Devices

The table below lists the program devices for use with the SCPU. Devices marked with a * are set as required in the system parameters.

Device			Application Range (Number of points)		Explanation
	X	Input	X, Y0 to 7FF (Number of Xs + Ys = 2048)		Provides PC command and data from external device, e.g. pushbutton, select switch, limit switch, digital switch.
	Y	Output			Provides program control result to external device, e.g. solenoid, magnetic switch, signal light, digital display.
	M	Special relay	M9000 to 9255 (256)		Predefined auxiliary relay for special purpose and for use in the PC.
*	M	Internal relay	M0 to 999 (1000)	Number of Ms + Ls + Ss = 2048	Auxiliary relay in the PC which cannot be output directly.
*	L	Latch relay	L1000 to 1024 (1024)		Auxiliary relay in the PC which cannot be output directly. Backed up during power failure.
*	S	Step relay	Can be used by setting the parameter (0)		Used in the same manner as an internal relay (M), e.g. as a relay indicating the stage number of a step-by-step process operation program.
	B	Link relay	B0 to 3FF (1024)		Internal relay for data link which cannot be output. May be used as an internal relay if not set for link initial data.
	F	Annunciator	F0 to 255 (256)		Used to detect a fault. When switched on during RUN by a fault detection program, stores a corresponding number in special register D.
*	T	100ms timer	T0 to 199 (200)		Up timers available in 100ms, 10ms, and 100ms retentive types.
		10ms timer	T200 to 255 (56)		
		100ms retentive timer	Can be used by setting the parameter. (0)		
*	C	Counter	C0 to 255 (256)		Up counters available in normal and interrupt types.
		Interrupt counter	Can be used by setting the parameter. (0)		
	D	Data register	D0 to 1023 (1024)		Memory for storing PC data.
		Special register	D9000 to 9255 (256)		Predefined data memory for special purpose.
	W	Link register	W0 to 3FF (1024)		Data register for use with data link.
*	R	File register	Can be used by setting the parameter. (0)		Extends data register using user memory area.
	A	Accumulator	A0, A1 (2)		Data register for storing the operation results of basic and application instructions.
	Z	Index register	Z (1)		Used to modify devices (X, Y, M, L, B, F, T, C, D, W, R, K, H, P).
	V		V (1)		
	N	Nesting	N0 to 7 (8 levels)		Indicates the nesting of master controls.
	P	Pointer	P0 to 255 (256)		Indicates the destination of the branch instruction (CJ, SCJ, CALL, JMP).
	I	Pointer for interruption	I0 to 31 (32)		Indicates the destination of an interrupt program corresponding to the interrupt factor which has occurred.
	K	Decimal constant	K-32768 to 32767 (16-bit instruction)		Used to specify the timer/counter set value, pointer number, interrupt pointer number, the number of bit device digits, and basic and application instruction values.
			K-2147483648 to 2147483647 (32-bit instruction)		
	H	Hexadecimal constant	H0 to FFFF (16-bit instruction)		Used to specify the basic and application instruction values.
			H0 to FFFFFFFF (32-bit instruction)		

4.24 SCPU Parameters

(1) Parameter setting involves specifying the usable ranges of various functions and the assignment of user memory area within the SCPU unit.
The parameters are stored in the first 3K bytes of the user memory area.

(2) The default values for the parameters are shown in the table below. The defaults may be used without alteration.

(3) the parameter settings may be changed for applications within the given limits. The parameters are set by peripheral equipment. Refer to the operating manuals of the peripheral equipment for information concerning parameter settings.

Parameter Setting Ranges

Item \ Setting		Default Value	Setting Range
Main sequence program capacity		6K steps	1 to 30K steps (in units of 1K step)
Sub-sequence program capacity		Absent	1 to 30K steps (in units of 1K step)
File register capacity		Absent	1 to 8K points (in units of 1K points)
Comment capacity		Absent	0 to 4032 points (in units of 64 points)
Status latch	Memory capacity	Absent	0/8 to 24 KB
	Data memory		Absent/present (0/8 KB)
	File register		Absent/present (2 to 16 KB)
Sampling trace		Absent	0/8 KB
Microcomputer program capacity		Absent	0 to 58KB (in units of 2 KB)
Setting of latch (power failure data retention) range	Link relay (B)	Only for L1000 to 2047. Absent for others.	B0 to 3FF (in units of 1 point)
	Timer (T)		T0 to 255 (in units of 1 point)
	Counter (C)		C0 to 255 (in units of 1 point)
	Data register (D)		D0 to 1023 (in units of 1 point)
	Link register (W)		W0 to 3FF (in units of 1 point)

Parameter Setting Ranges

Item \ Setting		Default Value	Setting Range
Setting of link range	Number of link stations	Absent	1 to 64
	Input (X)		X0 to 7FF (in units of 16 points)
	Output (Y)		Y0 to 7FF (in units of 16 points)
	Link relay (B)		B0 to 3FF (in units of 16 points)
	Link register (W)		W0 to 3FF (in units of 1 point)
Setting of internal relay (M), latch relay (L), step relay (S) setting		M0 to 999 L1000 to 2047 Absent for S	M/L/S0 to 2047 M, L, S are serial numbers
Setting of timer		100ms: T0 to 99 10ms: T200 to 255	256 points of 100ms, 10ms, and retentive timers (in units of 8 points) Timers have serial numbers.
Setting of counter		No interrupt counter	156 points of counters and interrupt counters (in units of 8 points)Timers have serial numbers.
I/O number assignment	Input (X) module	Absent	0 to 64 points (in units of 16 points)
	Output (Y) module		
	Special function module		
	Empty slot		
Setting of remote RUN/PAUSE contact		Absent	X0 to 7FF (1 point for each of RUN and PAUSE contacts. Setting of only PAUSE contact cannot be performed.)
Operation mode at the time of error	Fuse blown	Continuation	Stop/Continuation
	I/O verify error	Stop	
	Operation error	Continuation	
	Special function unit check error	Stop	
Annunciator display mode		F number display	Display of only F number or alternate display of F number and comment (Only alphanumeric characters may be displayed for comment.)
STOP → RUN display mode		Operation status prior to stop is re-output.	Output before stop or after operation execution
Print title entry		Absent	All 128 characters from MELSAP
Keyword entry		Absent	Maximum. 6 digits in hexadecimal (0 to 9, A to F)

4.25 SCPU Memory Operation

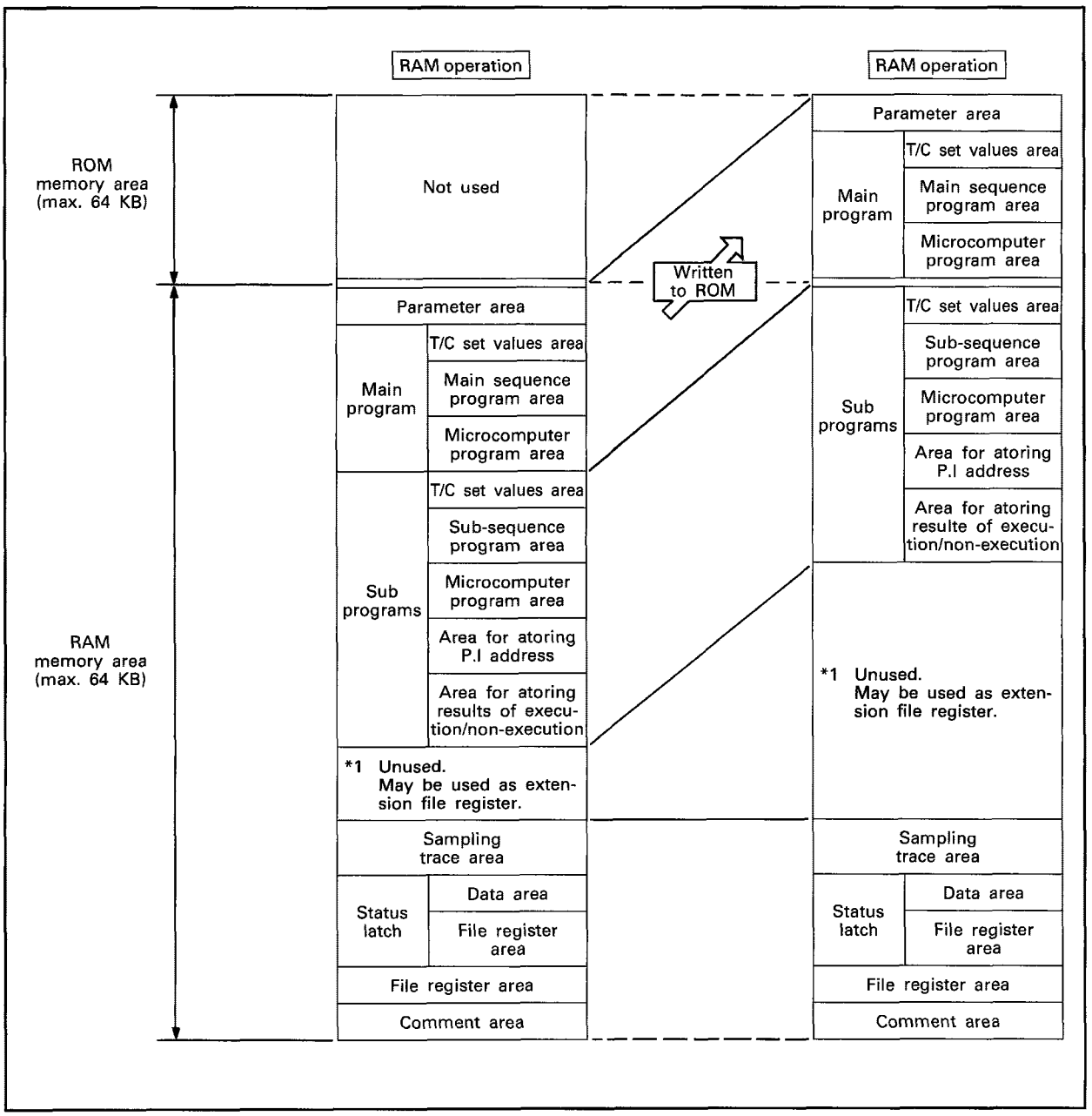
The SCPU has two memory modes, RAM operation and ROM operation.

The memory maps for RAM operation and ROM operation are shown below.

The types of data stored vary depending on the parameter settings.

(a) RAM operation
Beginning with the head address, the mapped RAM memory is, in order: the parameter area, the main program, and sub- program. Beginning with the last address, the mapped RAM memory is, in order: the comment, the file register, status switch, and sampling trace areas.

(b) ROM operation
The parameters and main program are stored in the ROM area. The sub-program is contained from the head address. Beginning with the last address, the mapped RAM memory is, in order: the comment, the file register, status switch, and sampling trace areas.


RAM operation
RAM operation
ROM memory area (max. 64 KB)
Not used
Written to ROM
Parameter area
Main program
T/C set values area
Main sequence program area
Microcomputer program area
Sub programs
T/C set values area
Sub-sequence program area
Microcomputer program area
Area for atoring P.I address
Area for atoring resulte of execu-tion/non-execution
RAM memory area (max. 64 KB)
Parameter area
Main program
T/C set values area
Main sequence program area
Microcomputer program area
Sub programs
T/C set values area
Sub-sequence program area
Microcomputer program area
Area for atoring P.I address
Area for atoring results of execu-tion/non-execution
*1 Unused. May be used as exten-sion file register.
*1 Unused. May be used as exten-sion file register.
Sampling trace area
Status latch
Data area
File register area
File register area
Comment area
Sampling trace area
Status latch
Data area
File register area
File register area
Comment area

Parameter Settings and Memory Capacity

Item		Unit of Settings	Memory Capacity	REMARKS	ROM Capabilities
Main Program	Parameter, T/C values	—	4 KB (fixed)	Yes	Parameters and T/C settings occupy 4 KB
	Sequence program	1K step	[Main sequence program capacity] × 2 KB		
	Microcomputer program	2 KB	[Main microcomputer program capacity] KB		
Sub Program	T/C settings, etc.	—	6 KB (fixed)	None	Values of the T/C settings, and the storage area of the PI addresses occupy 6 KB.
	Sequence program	1K step	[Main sequence program capacity] × 2 KB		
	Microcomputer program	2 KB	[Main microcomputer program capacity] KB		
Sampling trace		Absent/Present	0/8 KB		
Status Latch	Data memory	Absent/Present	0/8 KB		The capacity for the memory of the file register status latch is set by the number of file registers set by the parameters.
	File register	Absent/Present	[File register memory capacity] KB		
File Registers		1K points	[File register points] × 2 KB		
Comments		64 points	$\frac{\text{(Comment points)}}{64} + 1$ KB		1 KB is occupied by the system when the comment capacity is set.

POINT

The amount of usable memory varies depending upon the parameter settings.

4.26 SCPU I/O Assignment

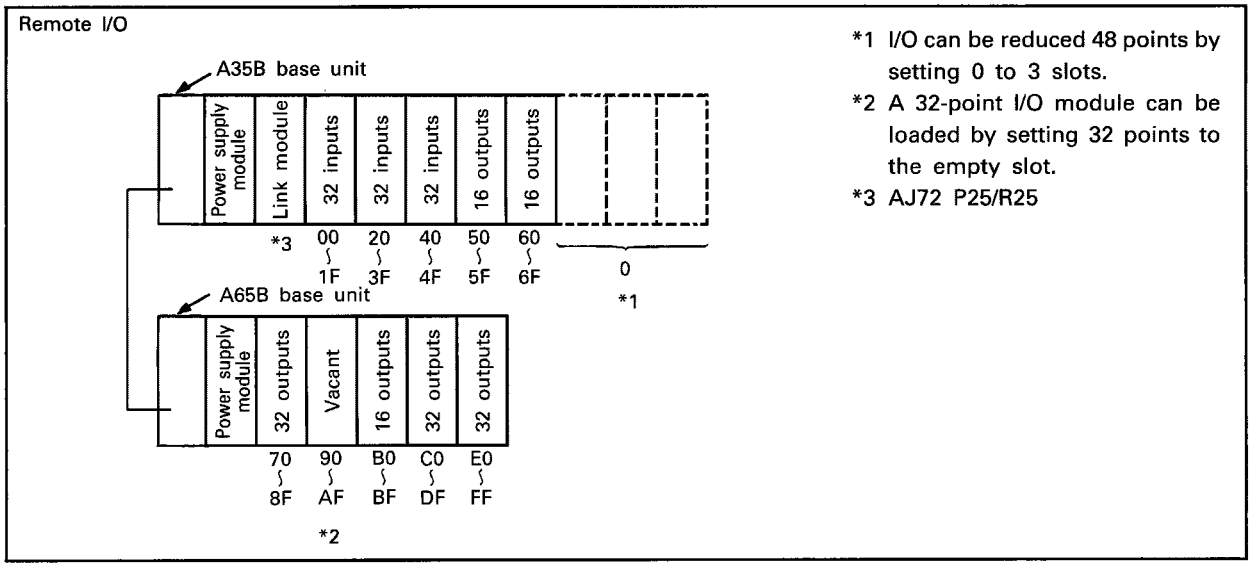
The initial processing of the SCPU automatically assigns the I/O addresses of the I/O modules and special function modules, loaded on Remote Stations of MELSECNET.

It is not necessary to set the I/O assignments using the peripheral equipment.

(1) Advantages of setting I/O assignments in relation to the remote I/O stations:

(a) Conserving the number of I/O points of empty slots
Setting "0" as the number of I/O points for the empty slots will conserve the number of I/O points occupied by empty slots.
For example, empty slots occupy 48 points when the A35B base unit is used. 48 points can be conserved by using the peripheral equipment to set the number of assignment points to 0.

(b) Reserving I/O points
32, 48, and 64 points can be reserved for empty slots in anticipation of future system extension.
Reserving I/O points makes it easy to extend and modify sequence programs since it is not necessary to change the addresses for each of the I/O modules.



(2) Precautions related to I/O assignments

(a) With the SCPU, slots 0 and 1 are used by the system. When setting I/O assignments, assign the special 32 points for slot 0, and the special 32 points and special 48 points for slot 1, which are set by the I/O points setting jumpers. See Section 4.11
Slot 0 is used for the transfer of data between SCPU and IFMEM. Assign to the special function module 32 points. Slot 1 is used for the MELSECNET/MINI-S3 master module. Assign to the special function mode 32/48 points.

(b) When configuring a MELSECNET remote I/O system with the SCPU as the master station, all remote I/O station areas must be assigned when setting I/O assignments.
I/O assignments cannot be made for only some of the slots (remote I/O stations).

4.27 SCPU Functions

The SCPU functions are listed below.

Function	Description	Refer to:
Constant scan	Executes the sequence program at the predetermined intervals independently of the scan time. Setting allowed between 10 and 2000 ms.	Section 4.28
LATCH (power failure data retention)	Retains device data if the PC is switched off or reset, or if instantaneous power failure occurs 20ms or longer. L, B, T, C, D and W can be latched.	Section 4.29
Remote RUN/STOP	Allows remote run/stop from external device (e.g. peripheral, external input, computer) with RUN/STOP switch in RUN position.	Section 4.30
PAUSE	Stops operation with the output (Y) status retained. Pause function may be switched on by any of the following ways: RUN/STOP switch on the front of the CPU. Remote pause contact Peripheral	Section 4.31
Status latch	Stores all device data to the status latch area of the memory cassette when the status latch condition is satisfied. The stored data can be monitored by the peripheral.	Section 4.32
Sampling trace	Samples the specified device operating status at predetermined intervals and stores the sampling result in the sampling trace area of the memory cassette. The stored data can be monitored by the peripheral.	Section 4.33
Step run	Executes the sequence program per instruction. Step run may be executed in either of two ways: a) By specifying the loop count. b) Per instruction.	Section 4.34
Offline switch	Allows the device (Y, M, L, S, F, B) used with the OUT instruction to be disconnected from the sequence program operation processing.	Section 4.35
Real Time Clock	Executes clock operation in the CPU module. Clock data includes the year, month, day, hour, minute, second, and day of the week. Clock data can be read to special registers D9025 to D9028.	Section 4.36

4.28 CONSTANT SCAN

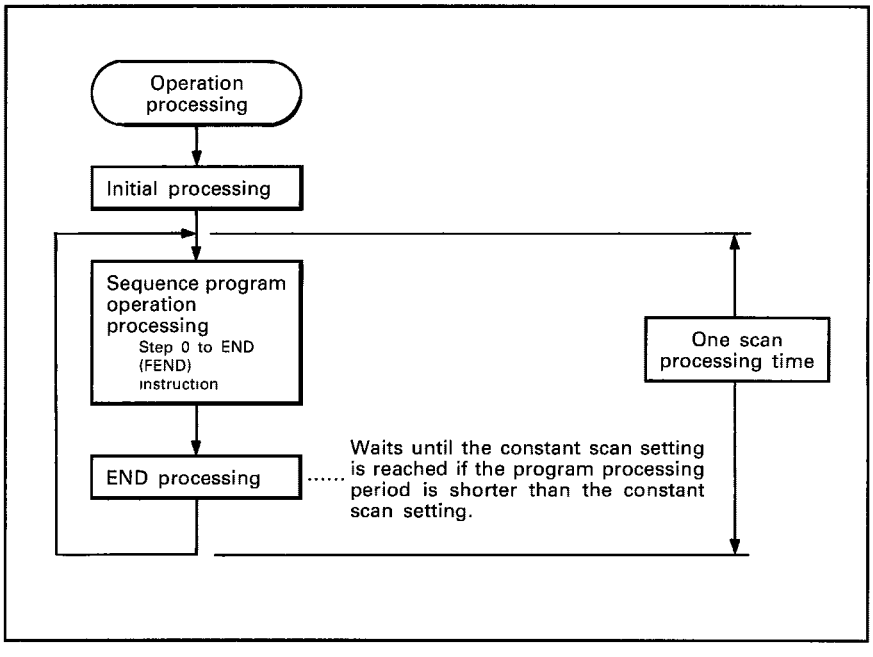
APPLICATION

Variations in positioning may occur due to the execution and non-execution times of instructions in the sequence program. Variations in positioning can be minimized through use of the constant scan function.

FUNCTION

(1) Definition

The constant scan function uniformly sets the processing time for each scan of the sequence program.



(2) Setting range

(a) The constant scan settings can be wirtten to D9020 in 10 ms increments between the value of 1 to 200. When values other than 1 to 200 are written to D9020 the following becomes true.

- −32768 to 0 ···· No constant scan setting
- 1 to 200 ····· Constant scan setting 10 to 2000 ms
- 201 to 32767 ··· Constant scan setting 2000 ms

(b) The following shows the relationship between D9020 and WDT (watch dog timer)

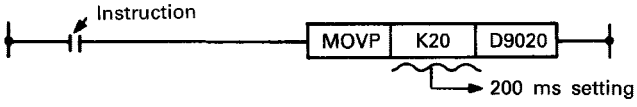
(D9020 value) (WDT value)−1

A WDT error may occur if the value set in D9020 is greater than that given in the above formula.

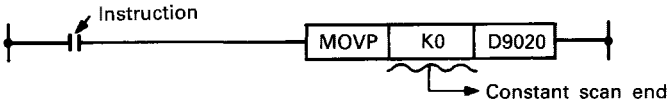
(3) Program example

The following is a program example of a constant scan setting and termination.

(a) To set constant scan to 200 ms.

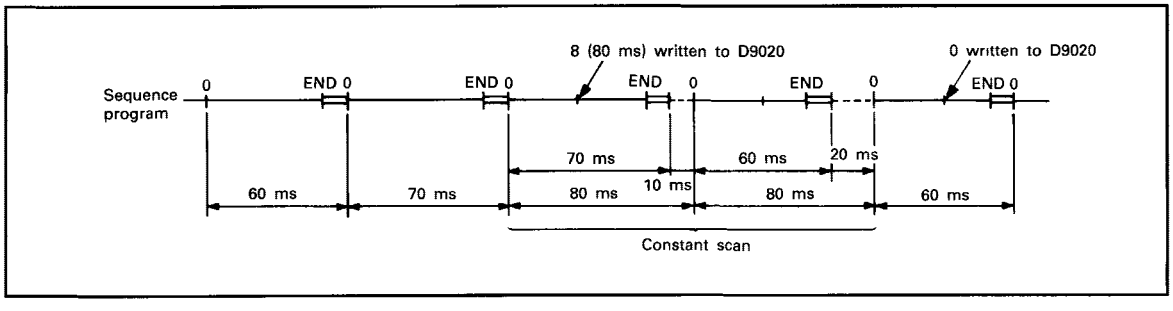


(b) To terminate constant scan



(4) Operation

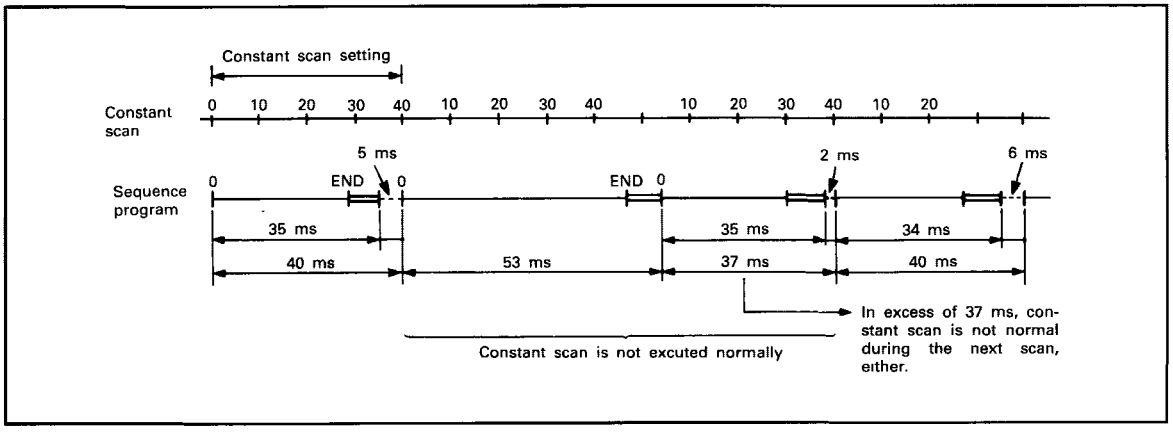
(a) Constant scan is executed for scan beginning with the scan in which the set value is written to D9020.



Constant Scan Execution

(b) The constant scan setting must be greater than the maximum scan time in the sequence program.

The constant scan is not executed normally if its setting is shorter than the program scan time.



Scan Time Longer Than Constant Scan Setting

(5) Accuracy

(a) Any of the following interrupt processings is allowed when there is wait time during END processing. The constant scan accuracy may therefore be deteriorated by the corresponding interrupt processing time.

Interrupt	Processing Time
I/O interrupt	General processing of data from IFMEM and MELSECNET 0.2 to 0.5 ms Interrupt from IFMEM 0.2 ms + (I16 interrupt program execution period)
10 ms interrupt	1.0 ms + I29 to I31 interrupt program execution period
Interrupt from peripheral	0.2 ms

When one or more of the above interrupts have occurred, total processing time is the sum of the individual interrupt processing time.

OPERATION

(a) To execute constant scan

1) Write the set value to D9020 in the sequence program; or
2) Write the set value to D9020 in test mode of the peripheral.

(b) To terminate constant scan

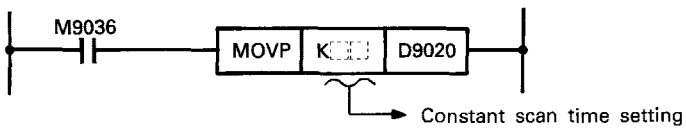
1) Write 0 to D9020 in the sequence program; or
2) Write 0 to D9020 in test mode of the peripheral.

(c) To change the set value during SCPU RUN

1) Modify the program which writes the constant scan set value to D9020 using the peripheral, rewrite it during RON, and switch on the constant setting instruction; or
2) Write a new value to D9020 in the test mode of the peripheral.

CAUTION

(a) D920 is cleared when the PC is switched on or reset. The following program is required to initiate constant scan after power on or reset.



(b) The constant scan is not executed normally if an instantaneous power failure occurs less than 10ms because constant scan period is prolonged by instantaneous power failure period.

4.29 LATCH

APPLICATION

Retains data if an instantaneous power failure occurs for more than 10 ms during continuous control.

FUNCTION

(1) Definition

The latch function retains device data stored in the SCPU if the SCPU is turned OFF or reset, or if an instantaneous power failure has occurred for more than 10 ms.

(2) Devices latched

1) Latch relay (L)
2) Link relay (B)
3) Timer (T)
4) Counter (C)
5) Data register (D)
6) Link register (W)

(3) Clearing latched data

(a) Latched data may be cleared in either of two ways:

1) Set the RUN/STOP switch to STOP and press the L.CLR switch.
2) Clear all devices from the GPP/HGP/PHP.

(b) Clearing latched data clears unlatched data at the same time.

OPERATION

Retains data if an instantaneous power failure occurs for more than 10 ms during continuous control.

CAUTION

(a) Device content stored in the latch range is backed by the battery (K6BAT) located on the A7BD-A3N-B circuit board. The battery is therefore required, since the sequence program is stored in a ROM during normal operations.

(b) Latched/unlatched device data is stored in the SCPU module. The data in the latch range is therefore lost if the battery connector is disconnected while the power is off.

4.30 REMOTE RUN/STOP

APPLICATION

(a) RUN/STOP may be executed at remote locations without controlling the RUN/STOP switch on the SCPU front panel when:

1) The SCPU is out of reach.
2) The SCPU is contained in a control box.

FUNCTION

(1) Definition

The Remote RUN/STOP controls run/stop of the SCPU from an external device (e.g. peripheral, external input, IFMEM) when the RUN/STOP switch is in the RUN position.

(2) Operation

1) Remote stop

The SCPU is set to STOP after the sequence program is executed up to the END (FEND) instruction.

2) Remote run

After remote stop, remote run sets the SCPU back to RUN to execute the sequence program from step 0.

OPERATION

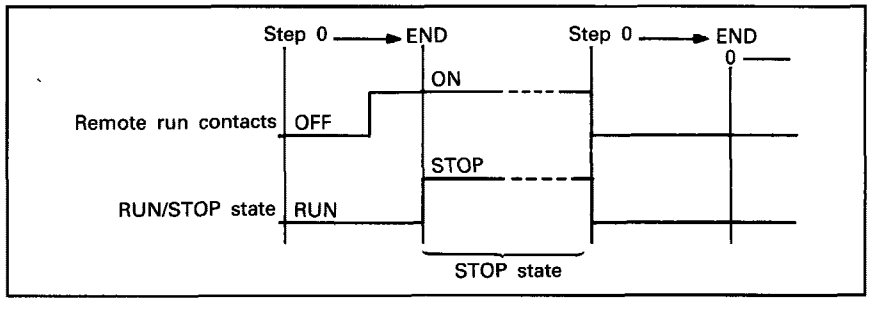
(a) Remote RUN/STOP may be executed using one of the following methods:

1) Remote run contacts (external input to be set by the peripheral);
2) Peripheral;
3) IFMEM

Remote run contacts

RUN/STOP of the SCPU is conducted by setting to ON/OFF the remote RUN contacts specified by the parameter settings, as shown below.

Remote run contacts { OFF ·····RUN state
ON ····· STOP state



RUN/STOP Timing Using Remote Run Contacts

Peripheral, IFMEM

The SCPU is set to RUN/STOP by remote RUN/STOP command from the peripheral or IFMEM.

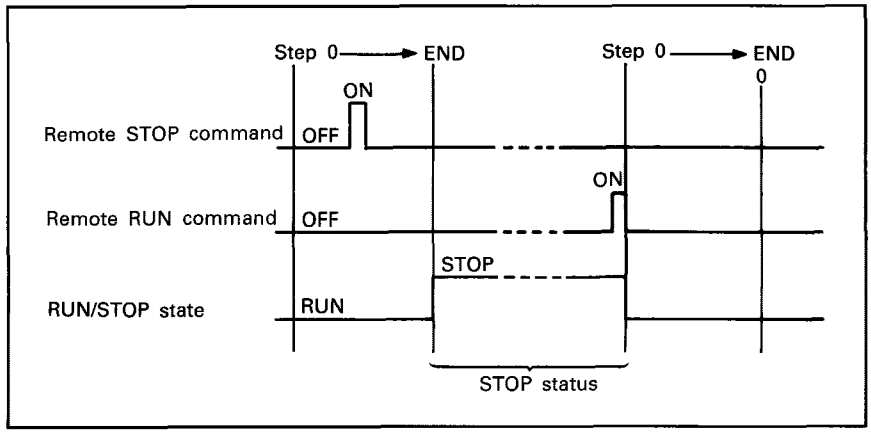


Fig. 4.11 RUN/STOP Timing Using Peripheral or IFMEM

CAUTION

(a) Note the following as the SCPU gives priority to STOP.

1) The SCPU module is set to stopP when the remote stop command is given from any of the remote RUN contacts, peripheral, or IFMEM.

2) To set the SCPU module from STOP state back to RUN, the remote run command must be provided by the external factor (remote RUN contacts, peripheral, IFMEM) which has set the CPU to STOP.

4.31 PAUSE

APPLICATION

The PAUSE function allows process control, etc., to be continued after the SCPU module is set to STOP.

FUNCTION

(1) Definition

The PAUSE function stops the operation processing of the SCPU while holding the state of all outputs (Y).

(2) Operation

(a) M9041 is switched ON at the END of a scan during which the PAUSE state has been set.
The operation processing stops when the next scan has been executed to the END (FEND) instruction after M9041 is switched on.

(b) The SCPU retains all output states after operation of one scan after M9041 is switched ON.
Any output that should be switched off in PAUSE state must be interlocked using M9041.

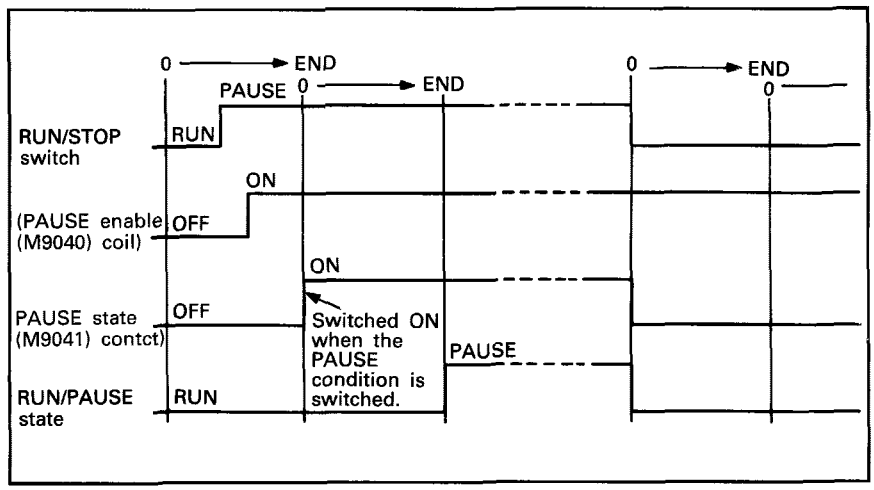
OPERATION

(a) The SCPU may be set to PAUSE using one of the following:

1) The RUN/STOP switch;
2) The peripherals;
3) The IFMEM.

RUN key switch

Operation is stopped when the RUN key switch has been set to "PAUSE" and the next scan has been executed to the END (FEND) instruction.
Operation is resumed by setting the RUN key switch to RUN or by switching M9040 to OFF using a peripheral.

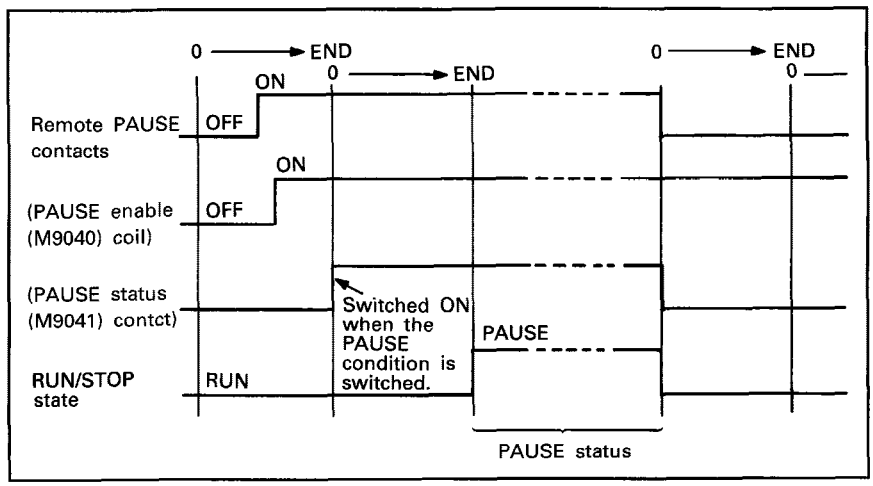


PAUSE Timing Using the RUN/STOP Switch

Remote PAUSE contacts

(1) Operation is stopped when the remote PAUSE contacts and M9040 are set simultaneously to ON and the next scan has been executed to the END (FEND) instruction.

(2) Operation is resumed by setting either the remote PAUSE contacts to OFF or by switching M9040 to OFF by a peripheral, IFMEM, etc.

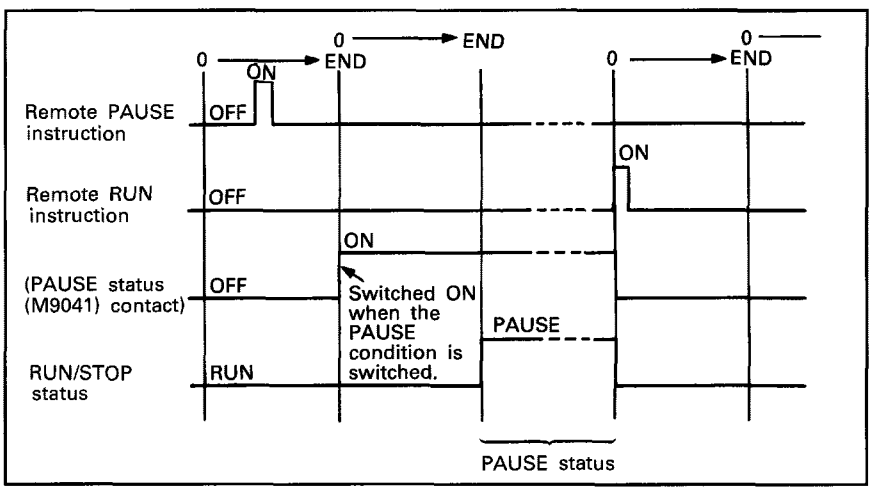


PAUSE Timing by Remote PAUSE Contacts

Peripheral and MCPU

1) Operation is stopped when the remote PAUSE instruction is received from the peripheral and the next scan has been executed to the END (FEND) instruction.

2) Operation is resumed when the remote RUN instruction is received from the peripheral.



PAUSE Timing by Peripheral and IFMEM

4.32 STATUS LATCH

APPLICATION

The status latch can be used to check device data when a given condition is satisfied during debugging.

FUNCTION

(1) Definition

The status latch function allows the contents of all devices to be stored in the status latch area when the SLT instruction is executed.
The date stored in the status latch area can be read and monitored by a peripheral (with an exception of the PU).

(2) Stored data

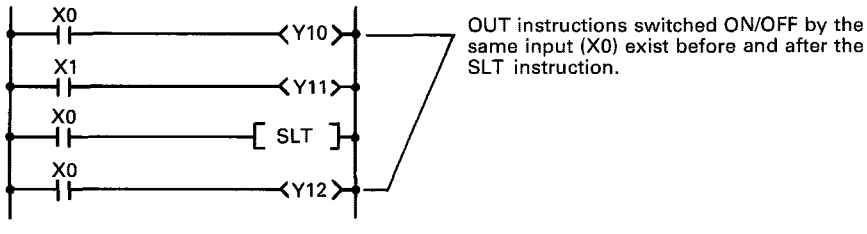
(a) The content of the devices stored in the status latch area are the following:
1) X, Y, M, L, S, F, B ····· ON/OFF data
2) T, C ····· Contact, coil ON/OFF data and present value of contacts and coils
3) D, W, A, Z, V, R ····· Stored data

(3) Data storing timing

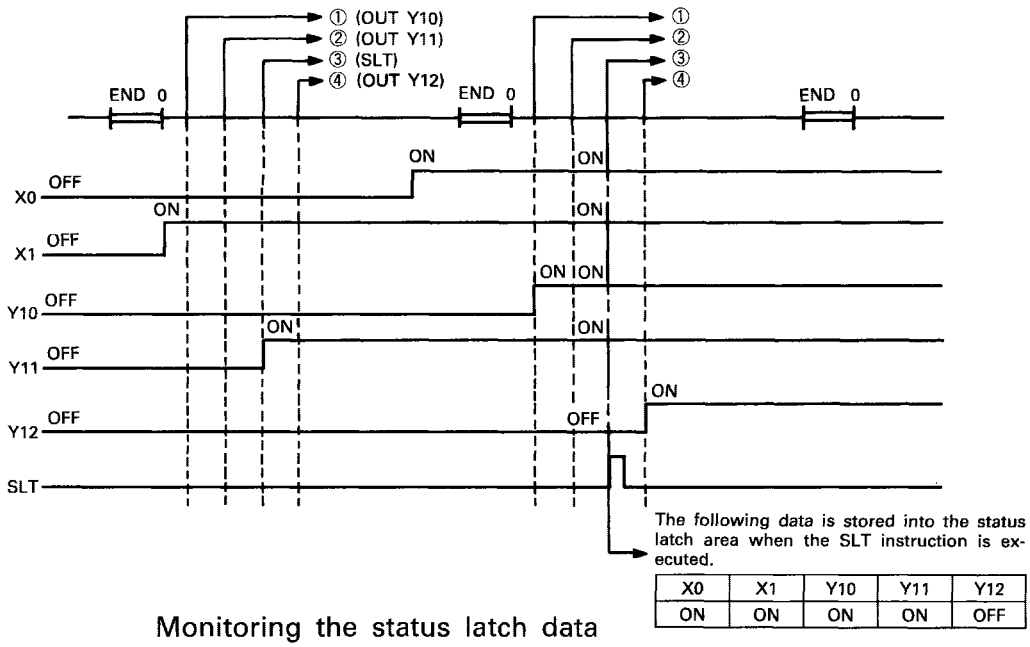
(a) Data is stored into the status latch area when the SLT instruction is executed.
Any device data that has changed after the execution of the SLT instruction is not stored into the status latch area.

(b) The following circuit provides an example of data storage when the SLT instruction has been executed.

[Circuit Example]

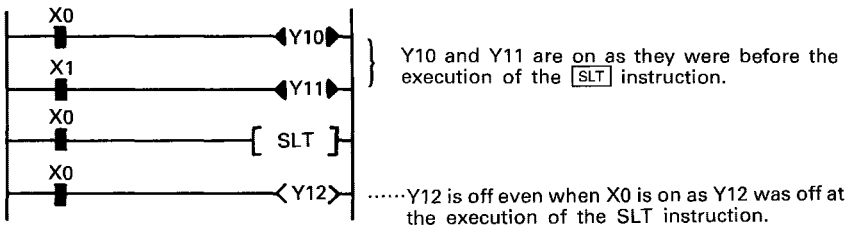


Timing chart



X0	X1	Y10	Y11	Y12
ON	ON	ON	ON	OFF

Monitoring the status latch data



OPERATION

(a) Setting the status latch area
The parameter setting of the peripherals, with the exception of the PU, set the status latch area and are written to the SCPU.

(b) Executing the status latch
Data is written to the status latch area when the SLT instruction is executed using the sequence program.

(c) Resuming the status latch
To Reset the SLT instruction by euecuting the SLTR instruction. This will cawse the SLT instruction to be executed again after it has been executed in the sequence program.

CAUTION

(1) Execution of the SLT instruction increases scan time as indicated below.
The watch dog timer of the SCPU should be set in consideration of the increase in scan time.

	Device Memory Only	Device Memory and File Register
Processing time (ms)	8.5 ms	24.6 ms

4.33 SAMPLING TRACE

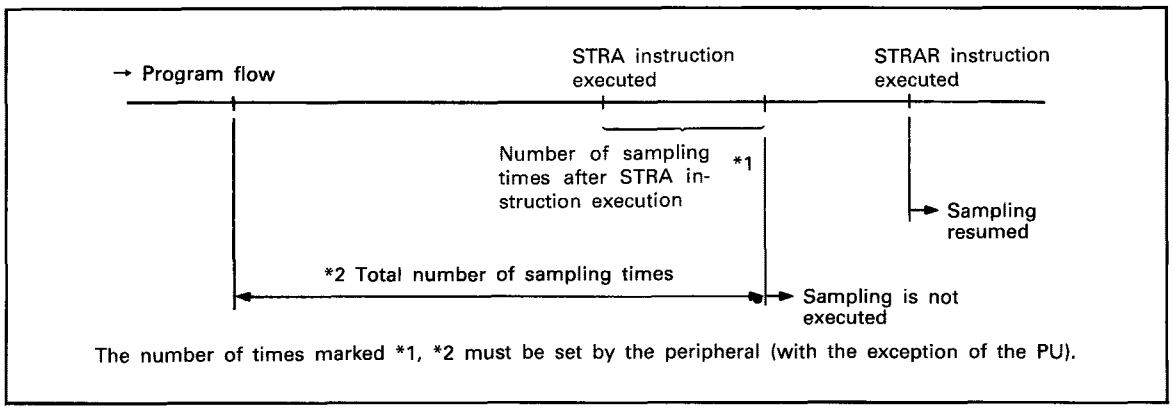
APPLICATION

The sampling trace shortens the time required for debugging by allowing the periodic monitoring of the contents of devices being used in programs.

FUNCTIONS

(1) Definition

The sampling trace stores data sampled at specified intervals (sampling periods) of the specified device in the sampling trace area.
Execution of the STRA instruction results in sampling occurring a specified number of times and the data results being stored in the sampling trace area.
The data stored in the status latch area can be read and monitored by a peripheral (with exception of the PU).



(2) Devices used

(a) The devices which may be used for the sampling trace are the following:

1) Bit devices (X, Y, M, L, S, F, B, T/C coil, T/C contact)
····· Maximum 8 points

2) Word devices (T/C present value, D, W, R, A, Z, V)
····· Maximum 3 points

OPERATION

(a) Setting the sampling trace area

Specify the sampling trace area using a peripheral (with the exception of the PU) and write to the SCPU.

(b) Setting the sampling trace data

Set the following data using a peripheral (with the exception of the PU) and write to the SCPU.

1) Number of sampling trace times
2) Devices to be traced
3) Sampling period

(c) Starting the sampling trace

Sampling trace may be initiated using one of the two following methods:

1) Peripheral (with the exception of the PU)

2) Switching on M9047.

(d) Terminating and stopping the sampling trace

To terminate:

By executing the STRA instruction in a sequence program, sampling is executed the specified number of times, data is latched, and the sampling trace is terminated.

To stop:

Sampling trace may be stopped by either of the following methods.

*Using a peripheral (with the exception of the PU)
*Switching OFF M9047

(e) Checking the sampling trace area data using the peripheral (with the exception of the PU).

(f) Resuming the sampling trace

Execute the STRAR instruction using the sequence program to resume the sampling trace.

4.34 STEP-RUN

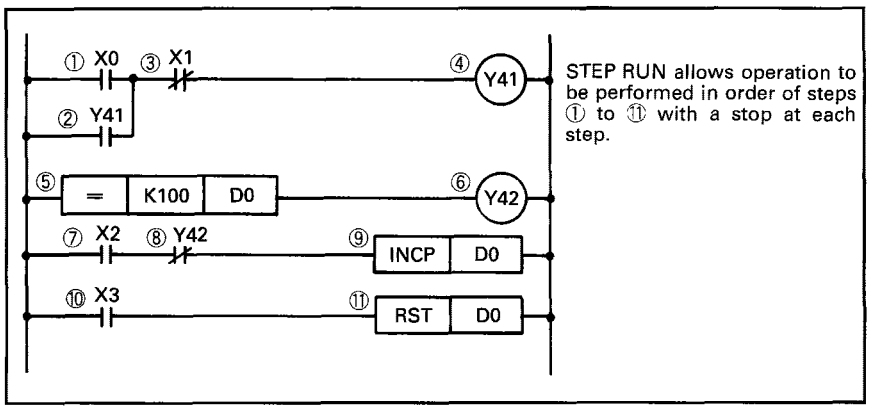
APPLICATION

The high speed of normal SCPU operation sometimes makes timing difficult to turn input signals ON/OFF during debugging. STEP-RUN operation executes the sequence program in a manner that allows monitoring of the actual status of the sequence program and content of each device when the input signals are turned ON/OFF.

FUNCTION

(1) Definition

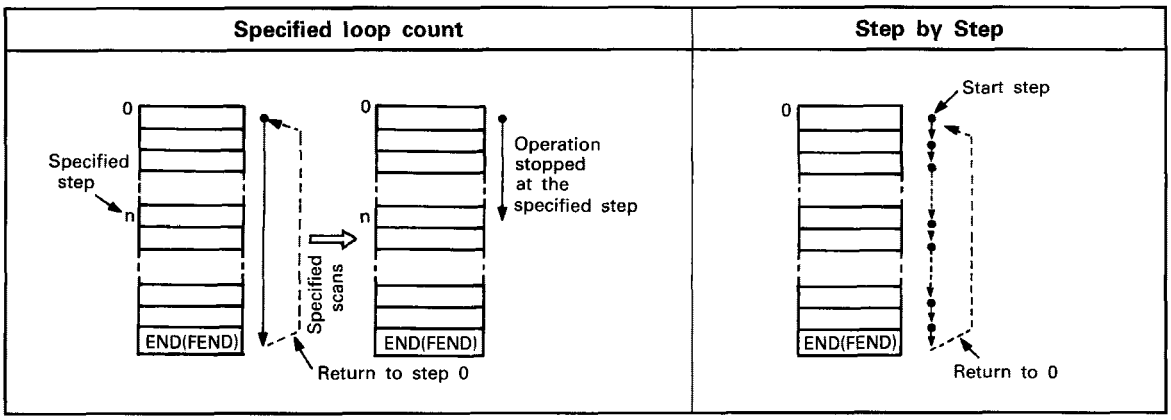
STEP-RUN operation executes the sequence program operation one instruction at a time.



(2) Types

1) Specified loop count ···· Operation is stopped at the specified step after the SCPU sequence program is executed the specified number of scans.

2) Step by step ····· Operation is executed instruction by instruction of the SCPU sequence program oseration, starting at step 0 or the current step.

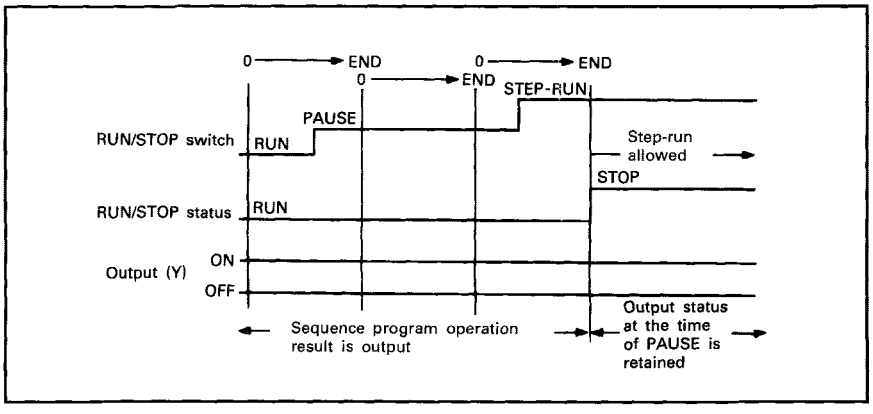


(3) Output (Y) state with RUN/STOP switch in STEP-RUN

(a) The RUN/STOP switch may be set to STEP-RUN in either of the two ways:

1) RUN PAUSE STEP-RUN

When the switch is set to STEP-RUN, operation is stopped with all outputs maintaining at the state set immediately prior to the switch being set to STEP-RUN.

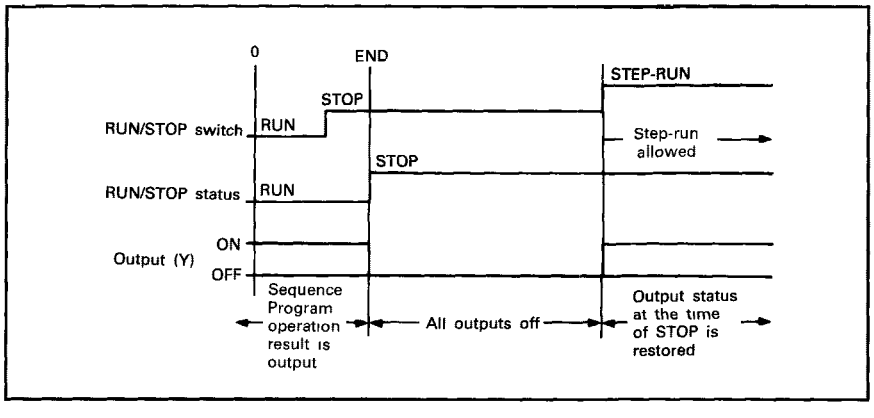


Timing for RUN PAUSE STEP-RUN

2) RUN STOP STEP-RUN

Depending on the setting of the parameter STOP RUN display mode, the following conditions are set.

* "Re-output operation conditions of that prior to STOP":
  When set to OFF and operation is stopped.
  When set to STEP-RUN, the output status at the time STOP was set is output while operation is stopped.
* "Output after operation executed"
  When set to STOP, all output are set to OFF and operation is stopped.
  When set to STEP-RUN, the output status at the time STOP was set is not re-output while operation is stopped.



Timing for RUN STOP STEP-RUN

(4) Timer, special timing clock processings during step-run

(a) The processing used for the timers during execution of the sequence program and the special timing clocks (M9030 to M9034) is as follows:

1) Timers

a) 10 ms timer ····· 10 ms incremented every scan
b) 100 ms timer ····· 100 ms incremented every 10 scans

2) Special timing clocks

a) M9030 (0.1s clock) ···· Switched on/off every 5 scans
b) M9031 (0.2s clock) ···· Switched ON/OFF every 10 scans
c) M9032 (1s clock)····· Switched on/off every 50 scans
d) M9033 (2s clock) ····· Switched on/off every 100 scans
e) M9034 (1m clock) ····· Switched on/off every 3000 scans

OPERATION

(a) Set the RUN/STOP switch to STEP-RUN.

(b) Use the peripherals (with the exception of the PU) to execute step operation.
Refer to the operating manuals of the peripheral equipment (with the exception of the PU) for information concerning step operation.

CAUTION

(a) When the step-run is performed with the loop count specified, the number of loops is counted when the step specified to stop the operation is executed.
Therefore, if the step specified to stop the operation is not executed by an instruction such as CJ, the number of loops is not counted.

(b) When the RUN key switch is switched from STEP-RUN STOP or RUN STOP, the status of the output existing immediately prior to the STOP is stored in the internal memory of the SCPU at the time STOP was set.
When the RUN key switch is switched from STOP STEP-RUN or STOP RUN, the status of the output existing immediately prior to the STOP is stored in the internal memory of the SCPU at the time STOP was set.
When the RUN key switch is switched from STOP STEP-RUN or STOP RUN, the outputs stored in the internal memory of the SCPU is output again prior to operation being restarted. If the outputs stored in the internal memory of the SCPU at the time STOP status was set are not to be output again, switch STOP STEP-RUN or STOP RUN after resetting.

4.35 OFFLINE SWITCH

APPLICATION

The OFFLINE switch allows the following checks to be conducted in the test mode of the peripherals by disconnecting the output of the OUT instructions from the sequence program.

1) Output module operation check

2) Output module and external device wiring check

FUNCTION

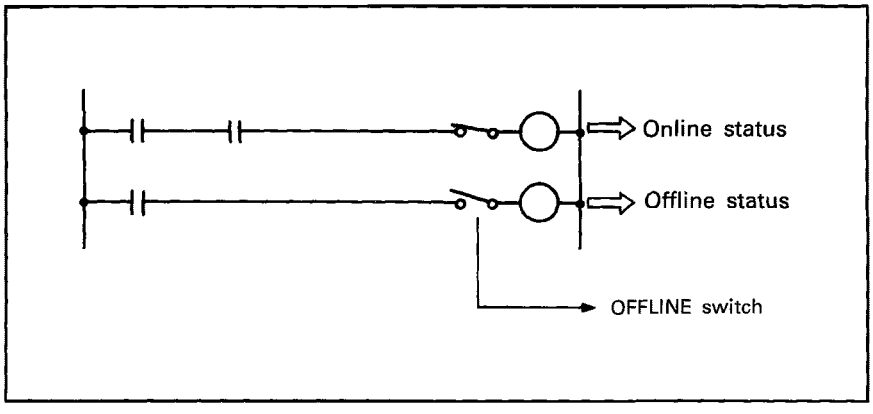
(1) Definition

(a) The OFFLINE function disconnects devices (Y, M, L, S, B, F) used with the OUT instruction from the sequence program.

(b) Online/Offline status is set when the imaginary OFFLINE switches, as those shown below, are closed/opened.

1) Opening the OFFLINE switch
Offline status is set. The OUT instruction device is disconnected from the sequence program.

2) Closing the OFFLINE switch
Online status is set. The OUT instruction device is controlled by the sequence program.



Online/Offline Status

(2) Device status in offline mode

(a) OUT instruction devices remain in the state that they were immediately prior to entering offline mode.

(b) If set/reset is forced by the peripheral in offline mode, devices remain in the state that they were forced.

OPERATION

(a) Setting the OFFLINE switch
Set the OFFLINE switch using the peripheral.

(b) Canceling the OFFLINE switch

1) Use the peripheral.
2) Reset the SCPU.

CAUTION

After the test operation is over, the OFFLINE switch must be canceled to enter online mode.

4.36 Real Time CLOCK FUNCTION

APPLICATION

(a) Allows real-time clock management by using the clock of one SCPU.

(b) Allows time management using a single SCPU when data link operations are being executed.

FUNCTION

(1) Definition

Allows the clock to be operated in accordance with the data set in the SCPU.
When power to the programmable controller is turned off, the clock is operated by the memory cassette battery.

(2) Clock data

(a) The clock data includes the year, month, day, hour, minute, second and day of the week, and is set to the clock devices.

1) Year...Expressed by the 2 least significant digits
2) Leap year...Automatically updated
3) Time...24 hours basis (0 to 23 o'clock)

(b) Clock data may be set and read by using special relays and registers.

(c) Clock data accuracy depends on the ambient temperature.

Ambient Temperature (C)	Accuracy (Weekly difference, Section)
+40	+15.5
+25	+2.75
0	+6.5

(d) When M9027 is set to ON, the following clock data is displayed on the option board: month, day, hour, minute, and second. Since error messages are given higher priority, clock data will not be displayed when an error occurs.

(3) Special relays, registers

(a) Special relays

Device	Description	Erplanation
M9025	Clock data set request	Writes clock data from D9025 through D9028 to the clock devices after the END instruction is executed during the scan when M9025 is switched on.
M9026	Clock data error	Switched on when any clock data set is not BCD.
M9027	Clock data display	When M9027 stays ON, clock data is displayed to the LED on the front panel of the CPU module.
M9028	Clock data read request	When M9028 stays ON, clock data is read to D9025 to D9028 after the END instruction is executed.

(b) Special Registers

Device	Description	Erplanation
D9025	Clock data (Year, month)	b15 b0 Month (01 to 12 in BCD) Year (00 to 99 in BCD)
D9026	Clock data (Day, hour)	b15 b0 Hour (00 to 23 in BCD) Day (01 to 31 in BCD)
D9027	Clock data (Minute, second)	b15 b0 Second (00 to 59 in BCD) Minute (00 to 59 in BCD)
D9028	Clock data (Day of the week)	b15 b0 Day of the week (00 to 06 in BCD) 0 Correspondence between the day of the week and number

Day of the Week	Sun	Mon	Tue	Wed	Thu	Fri	Sat
Stored data	0	1	2	3	4	5	6

OPERATION

(a) Writing the clock data to clock devices

1) Store the clock data to D9025 to D9028 in BCD code.

2) Switch on M9025

CAUTION

(a) The clock data must be written to the clock when using the clock function.

(b) All clock data must be stored in D9025 to D9028 even when part of the data is modified.

(c) Normal clock operation cannot be performed if invalid data is written.

Example

Month: 13
Day: 32

(d) Clock operation is backed up by the battery located on the A7BD-A3N-B circuit board. Clock operation will be discontinued if the battery connector is disconnected.

5. PRE-OPERATION SETTINGS AND PROCEDURES

5.1 Handling

This section gives handling instructions for the A7BDE-A3N-PT32S3 A3-CPU Programmable Controller option card.

(1) The A7BDE-A3N-PT32S3 is packaged in a wrapping that protects against damage by static electricity. Be sure to enclose the A7BDE-A3N-PT32S3 in this special wrapping whenever it is being moved or stored.

(2) Do not touch the components or conductive areas on the printed board, because damage may be caused by static electricity.

(3) When mounting the A7BDE-A3N-PT32S3, hold the printed circuit board by the edges or the mounting fixtures. Insert the connector into the circuit firmly.

(4) Do not drop the A7BDE-A3N-PT32S3 or subject it to shocks.

(5) Do not remove the printed circuit board from the mounting fixtures, as damage may result.

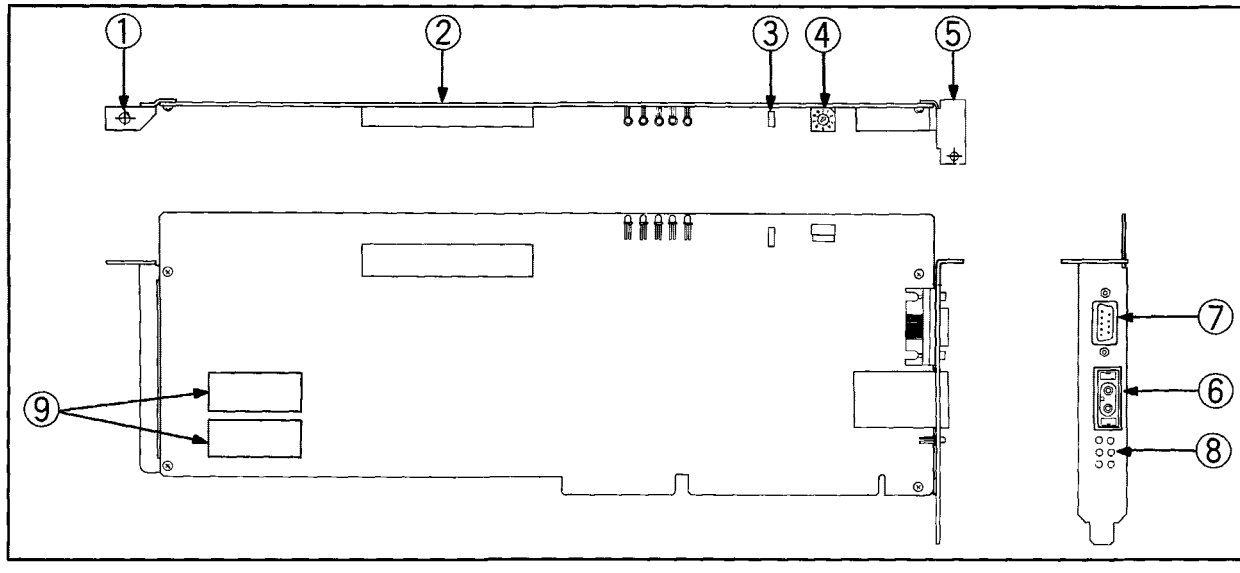
(6) When mounting the A7BDE-A3N-PT32S3, ensure that no wire cutoffs enter from the upper sections.

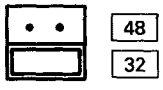
(7) Tighten the A7BDE-A3N-PT32S3 fixing screws (M4) to a torque of 12 to 19 Kg.cm.

5.2 A7BDE-A3N-PT32S3 Nomenclature

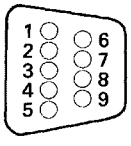
The following section describes the components, their names, and locations on the A7BDE-A3N-PT32S3 interface board.

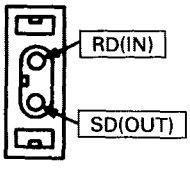
5.3 A7BDE-A3N-PT32S3A Nomenclature



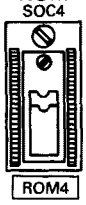
<table>
<tr><th>Number</th><th>Name</th><th>Description</th></tr>
<tr><td>①
⑤</td><td>Mounting fixture</td><td>Fixture for fixing the A7BDE-A3N-PT32S3A printed board onto the PC-AT®.</td></tr>
<tr><td>②</td><td>Connector for the A7BDE-A3N-B printed board</td><td>Connector for connecting the A7BDE-A3N-PT32S3A printed board and the A7BDE-A3N-B printed board via the ACP2PC cable.</td></tr>
<tr><td>③</td><td>Jumper for the use mode switch
 
</td><td>This jumper determines whether the master module operates in the extension mode or the I/O dedicated mode.
Extension mode ·····Jumper is placed in the "48" position.
I/O dedicated mode ·····Jumper is placed in the "32" position.

REMARK

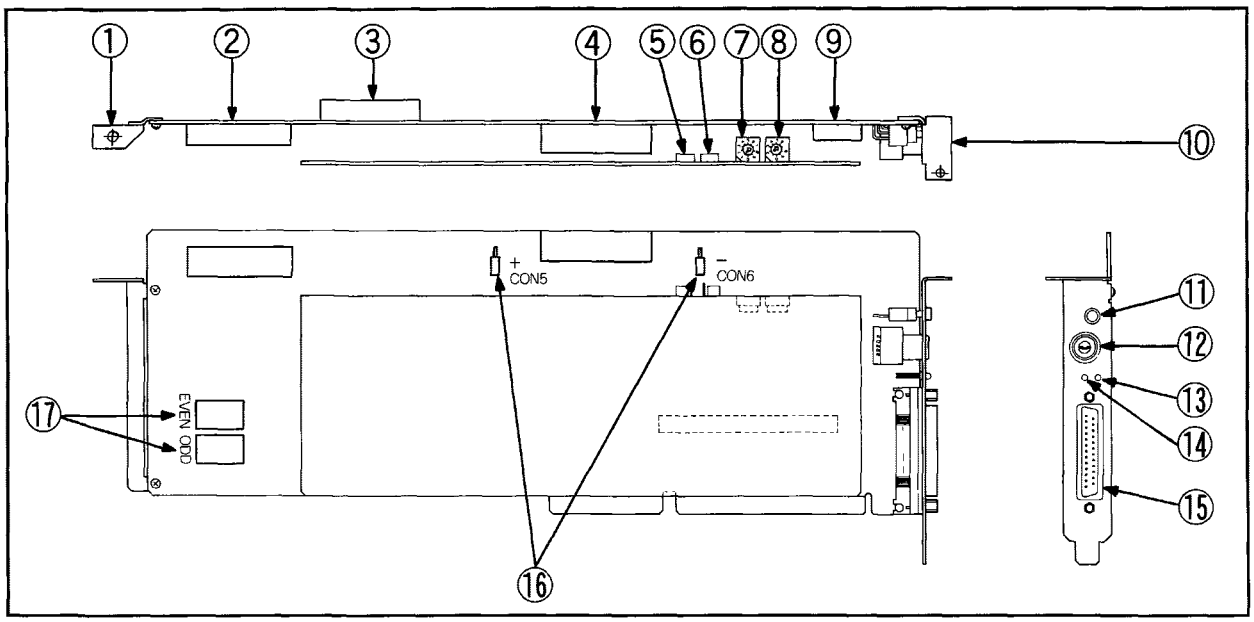
1. The jumper is set in the "32" position when shipped from the factory.
2. "32" and "48" are the number of I/O points in the master module when set in the corresponding mode.</td></tr>
<tr><td>④</td><td>Mode setting switch</td><td>Sets the operation mode to MELSECNET/MINI (For more details, see Section 5.9)</td></tr>
<tr><td>⑥</td><td> 

Connector for twisted-pair link</td><td>Connector for twisted pair link of MELSECNET/MINI-S3
<table><tr><th>Pin No.</th><th>Signal</th><th>Remarks</th></tr><tr><td>1</td><td>SDA</td><td rowspan="9">(1) The SG of pin No. 6 to 7 set internally.
(2) Connector type 17 JE-23090-02-D8A (DDK)
(3) The A7BDE-A3N-PT32S3A does not have an FG terminal. Connect the shield of the shielded cable to the connector cover.</td></tr><tr><td>2</td><td>SDB</td></tr><tr><td>3</td><td>RDA</td></tr><tr><td>4</td><td>RDB</td></tr><tr><td>5</td><td>Not Used</td></tr><tr><td>6</td><td>SG</td></tr><tr><td>7</td><td>SG</td></tr><tr><td>8</td><td>SG</td></tr><tr><td>9</td><td>SG</td></tr></table></td></tr>
<tr><td>⑦</td><td>Connector for the optical fiber cable
 
</td><td>This connector is used for an optical fiber cable when communication with remote units is conducted via an optical data link.

RD(IN) : Connected to SD(OUT) of the previous station.
SD(OUT) : Connected to RD(IN) of the succeeding station.</td></tr>
<tr><td>⑧</td><td>LEDs for operation status display</td><td>Indicates the operation status of the MELSECNET/MINI.
<table><tr><th>LED Name</th><th>Content</th></tr><tr><td>RUN</td><td>Lit when master module is operating normally. Out when a hardware error occurs.</td></tr><tr><td>SD</td><td>Flickers during data sending.</td></tr><tr><td>RD</td><td>Flickers during data receiving.</td></tr><tr><td>RD.E</td><td>Lit when receive data error occurs.</td></tr><tr><td>L.E</td><td>Lit when loop error occurs.</td></tr><tr><td>RM.E.</td><td>Lit when communication error occurs in a station within the loop.</td></tr></table></td></tr>
</table>

Number	Name	Description
⑨	Installation socket for the initial data ROM SOC3  ROM3	This socket is used to install the ROM containing the initial data when the master module is used in the extension mode. (The ROM need not be when the master module is used in the dedicated mode.) Initial data is written to the ROM using the SW□-MINIP type system floppy disk.
⑩	Installation socket for the message ROM SOC4  ROM4	This socket is used to install the ROM containing message data used for display on the LCD of the operating box when the operating box is used in the MINI-S3 link. (The ROM need not be installed when the operating box is not used.) Message data is written to the ROM using the SW□-MINIP type system floppy disk.

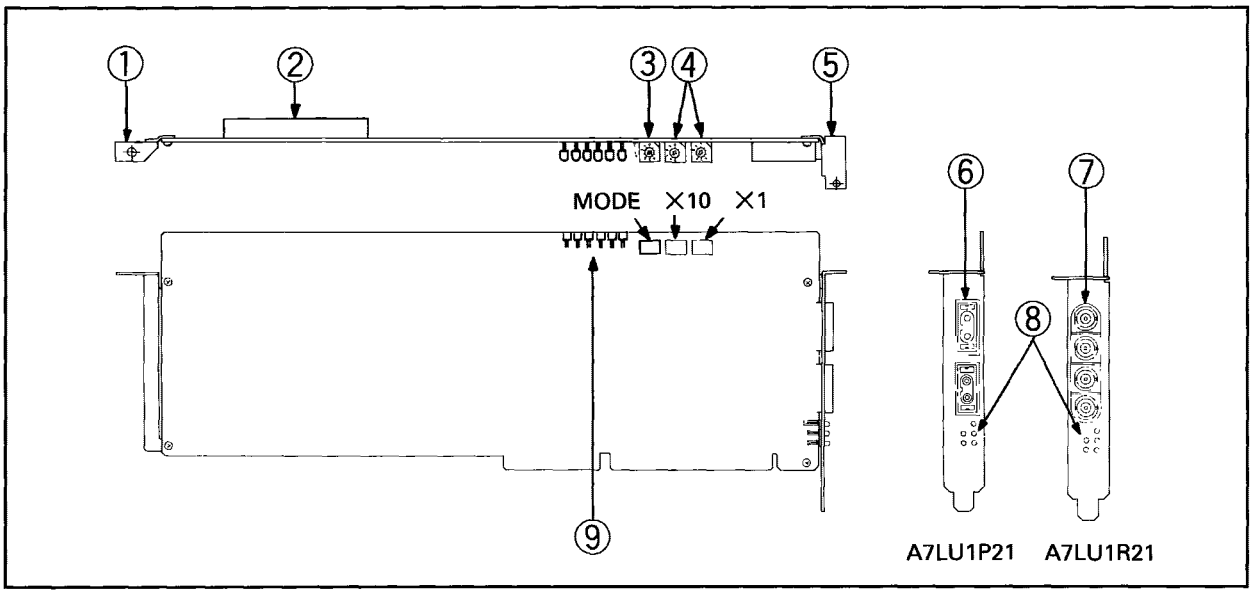
5.4 A7BDE-A3N-B.C Nomenclature

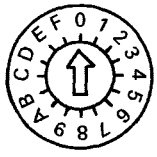


Number	Name	Description
① ⑩	Mounting fixture	Fixture for fixing the A7BDE-A3N-B and C printed boards onto the PC-AT® module.
②	Connector for the A7LU1P21/R21	Connector for connecting the A7BDE-A3N-B and C printed boards and the A7LU1P21/R1 (MELSECNET data link module) printed board via the ACP2LU1 cable.
③	Connector for the A7BDE-A3N-PT32S3A	Connector for connecting the A7BDE-A3N-B and C printed boards and the A7BDE-A3N-PT32S3A printed board via the ACP2PC cable.
④	Battery (K6BAT)	Battery for backup power for the IC-RAM memory and latching function during power failures or when power is not ON.
⑤	Jumper 1	Set to AT when shipped. Do not change.
⑥	Jumper 2	Set to 100H when shipped. May also be set to 300H. See Section 5.15.
⑦	Board interrupt setting switch	This dial sets the A7BDE-A3N interrupt (IRQ) number. For details see Section 5.16.
⑧	Board No. setting switch	This dial sets the A7BDE-A3N Board Number. For details see Section 5.15.
⑨	ROM/RAM memory switching protection switch	The area protected in the RAM memory varies depending on the setting of the ROM/RAM switch setting. This switch should be set to ROM if the sequence program is stored in the ROM and to RAM if stored in the RAM. See Section 5.17.
⑪	LATCH CLEAR switch	The LATCH CLEAR switch sets to either OFF or 0 the device memories of devices with latch ranges set by parameters. Note that the special relays (M9000 to 9255), special registers (D9000 to 9255), and file registers are not affected. (Effective only when the RUN/STOP switch is in STOP.)
⑫	RUN/STOP key switch	RUN: Executes operation of sequence program STOP: Stops operation of sequence program PAUSE: Stops operation of the sequence program while maintaining output status of conditions existing just prior to the pause. STEP-RUN: Executes step operation of the sequence program
⑬	ERROR LED	Lit: A watch dog timer error or self-diagnosis error occurred due to faulty hardware. Flicker: Annunciator (F) was set.

Number	Name	Description
⑭	RUN LED	Indicates SCPU operation status. Lit: Operation being conducted with RUN/STOP key switch in either RUN or STEP-RUN. Extinguished: Operation stopped with the RUN/STOP key switch in either STOP or PAUSE, or a WDT (error code 25) error has occurred. Flicker: An error stopping operation occurred during self-diagnosis. Flickering also occurs for about 2 sec. when a LATCH CLEAR has been executed.
⑮	RS-422 connector	Connector for peripherals (Use protective cover when not in use.)
⑯	Connector for the battery leads	Connects the K6BAT red lead to + terminal of CON5, blue lead to the - terminal of CON6.
⑰	ROM socket	Connects ROM in which the sequence program is loaded. Ensure that the installed memory is of the same type. Even and odd (address) memories should be installed in the EVEN and ODD locations respectively.

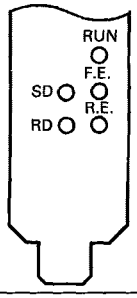
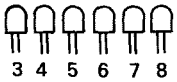
5.5 A7LU1EP21/R21 Nomenclature



Number	Name	Description
① ⑤	Mounting fixture	Fixture for fixing the A7LU1EP21/R21 printed board onto the PC-AT® module.
②	Connector for connecting the A7BDE-A3N-B and C printed boards.	Connector for connecting the A7LU1EP21/R21 printed board and the A7BDE-A3N-B and C printed boards via the ACP2LU1 cable.
③	Mode switching switch MODE 	The mode switch provides the following functions.

Setting No.	Name	Description
0	Online	Automatic return to line during normal operation
1	Online	No automatic return to line during normal operation
2	Online	Said station is disconnected from the line
3	Test mode 1	Forward loop test
4	Test mode 2	Reverse loop test
5	Test mode 3	Station-to-station test mode (master station)
6	Test mode 4	Station-to-station test mode (slave station)
7	Test mode 5	Self-loop test
8	—	Not used
9	—	Not used
A ~ C	—	Not usable
D ~ F	—	Not used

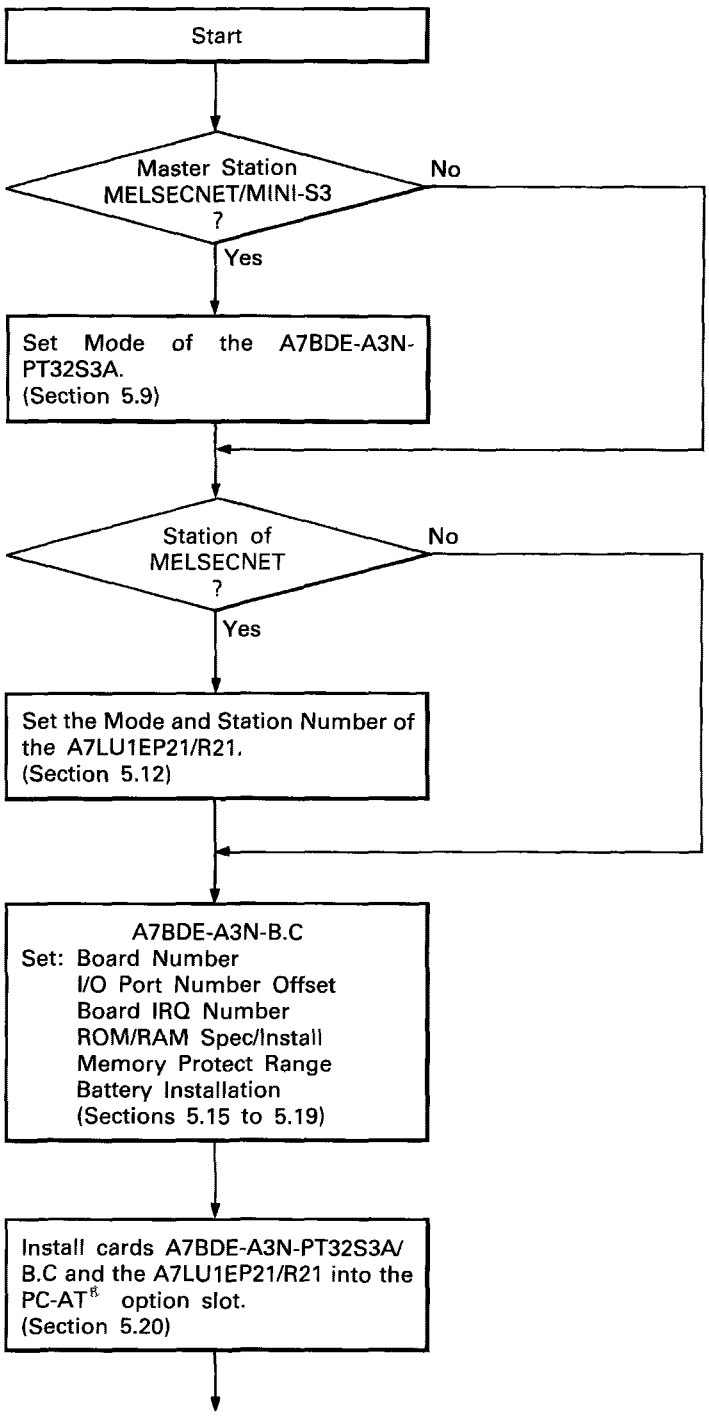
Number	Name	Description
④	Station number setting switch X10 X1	This switch is used to set station numbers for the MELSECNET data link *Sets station numbers 00 to 64. *X10 sets the tens column of numbers. *X1 sets the ones column of numbers. *Setting for the master station is "00". *Settings for the local stations are "01" to "64".
⑥	Connector for fiber optic cable	(1) The cable terminals are configured in the following manner. PC-AT® upper end; IN; OUT; LED IN: Reverse loop send, Forward loop receive OUT: Reverse loop send, Forward loop receive (2) The cables are connected in the following manner. OUT IN; OUT IN; OUT IN; Upper end Master station; Station #1; Station #2 IN : Connected to OUT of the previous station. OUT : Connected to IN of the next station.
⑦	Connector for coaxial cable	(1) The cable terminals are configured in the following manner. PC-AT® upper end; LED R-SD: Reverse loop send F-RD: Forward loop receive F-SD: Reverse loop send R-RD: Forward loop receive (2) The cables are connected in the following manner. (OUT) (IN) R-RD F-SD F-RD R-SD; (Upper end) Master station; Station #1; Station #2 (IN)R-SD : Connected to the (OUT)R-RD of the previous station (IN)F-RD : Connected to the (OUT)F-SD of the previous station (OUT)F-SD : Connected to the (IN)F-RD of the next station (OUT)R-RD : Connected to the (IN)R-SD of the next station

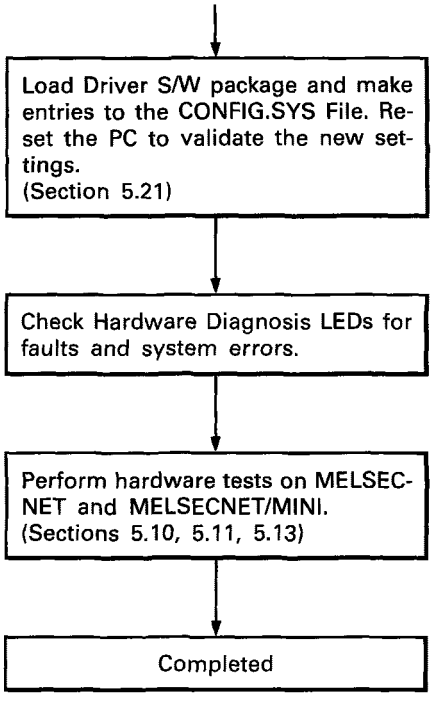
<table>
<tr><th>Number</th><th>Name</th><th>Description</th></tr>
<tr><td>⑧</td><td>LED1 for display of operation status.
 
</td><td>The LED displays operation status and information concerning abnormal conditions.
<table>
<tr><th>LED Name</th><th>Description</th></tr>
<tr><td>RUN</td><td>Lights during normal I/F board operation. Extinguishes if abnormal condition occurs.</td></tr>
<tr><td>F.E.</td><td>Lights if forward loop receive data error occurs or if forward loop cable should open.</td></tr>
<tr><td>R.E.</td><td>Lights if reverse loop receive data error occurs or if reverse loop cable should open.</td></tr>
<tr><td>SD</td><td>Lights during data send.</td></tr>
<tr><td>RD</td><td>Lights during data receive.</td></tr>
</table></td></tr>
<tr><td>⑨</td><td>LED2 for display of operation status
 
</td><td>The LED displays operation status and information concerning abnormal conditions.
<table>
<tr><th>LED No.</th><th>LED Name</th><th>Description</th></tr>
<tr><td>3</td><td>CRC</td><td>Lights if code check error occurs.</td></tr>
<tr><td>4</td><td>OVER</td><td>Lights if data latch delay error occurs.</td></tr>
<tr><td>5</td><td>AB.IF</td><td>Lights when all data is 1.</td></tr>
<tr><td>6</td><td>TIME</td><td>Lights when specified time is exceeded.</td></tr>
<tr><td>7</td><td>DATA</td><td>Lights if receive data error exists.</td></tr>
<tr><td>8</td><td>UNDER</td><td>Lights if receive data error exists.</td></tr>
</table></td></tr>
</table>

5.6 Pre-Operation Settings and Procedures

The following sections provide the various procedures, names, and settings required prior to operation of the A7BDE-A3N-PT32S3.

5.7 Pre-Operation Settings Procedure Flow Chart




Load Driver S/W package and make entries to the CONFIG.SYS File. Reset the PC to validate the new settings.
(Section 5.21)
Check Hardware Diagnosis LEDs for faults and system errors.
Perform hardware tests on MELSECNET and MELSECNET/MINI.
(Sections 5.10, 5.11, 5.13)
Completed

5.8 A7BDE-A3N-PT32S3A/B.C and A7LU1EP21/R21 Hardware Settings

The following sections describe how to select and set the various hardware switches, required before operation of the cards may begin. Ensure that the PC is off when new settings are being made.

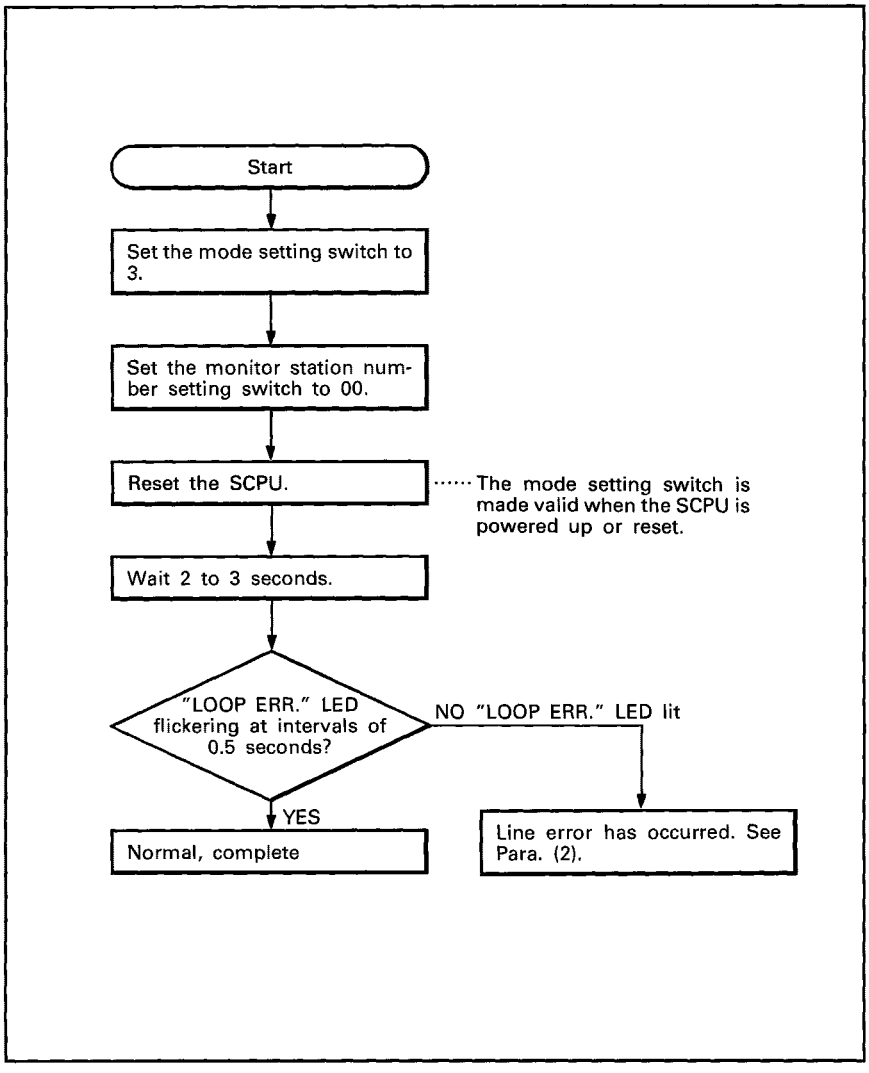
5.9 A7BDE-A3N-PT32S3A MELSECNET/MINI-S3 Mode Setting

The A7BDE-A3N-PT32S3A MELSECNET/MINI-S3 option card has five operating modes: three online modes and two test modes. They are selected by means of a dial switch located near the top of the card. The function of each mode is described in the table below. For further details, please consult the MELSECNET/MINI User's Manual.

Switch No.	Switch Name	Contents	Remarks
0	ONLINE (A.R.)	System automatically returns to online. When a communication error occurs in a remote I/O station, only that station is disconnected. I/O refreshing continues with other properly operating stations. The disconnected station automatically returns to the system when the station status returns to normal.	Online mode
1	ONLINE (U.R.)	System does not automatically return to online. When a communication error occurs in a remote I/O station, only that station is disconnected. I/O refreshing continues with other properly operating stations. Even if the station with which the communication error occurred returns to normal, it does not return to the system unless a startup is performed.	Online mode When online status is not automatically returned to the system, the outputs of the remote I/O station in which the communication error occurred are all set to OFF regardless of the E.C. MODE switch settings (ON/OFF) of the remote I/O station.
2	ONLINE (E.S.)	System stops when an online error is detected. When a communication error occurs in a remote I/O station, even only one, all remote I/O stations disconnect from the system (I/O refresh is stopped). Even if the station with which the communication error occurred returns to normal, it does not return to the system unless a startup is performed.	Online mode
3	TEST 1	Line check mode This mode checks for hardware errors in the MINI link and breaks in the cables.	Test mode
4	TEST 2	Luminous energy check mode measures the level of luminous energy on the receiving side of the remote I/O stations participating in the optical data link.	Test mode
5 ∫ 9	—	Not used.	When the switch number is set to 5, the TEST LED will light although there is no cause for an error. When the switch numbers are set to 6 through 9, the RUN LED and TEST LED all extinguish.

5.10 A7BDE-A3N-PT32S3A MELSECNET/MINI-S3 Line Check Mode

Line check mode is used to check the transmitting/receiving hardware, and check for fiber optic/twisted pair cable breakage. The general procedure is given in the flow chart below.

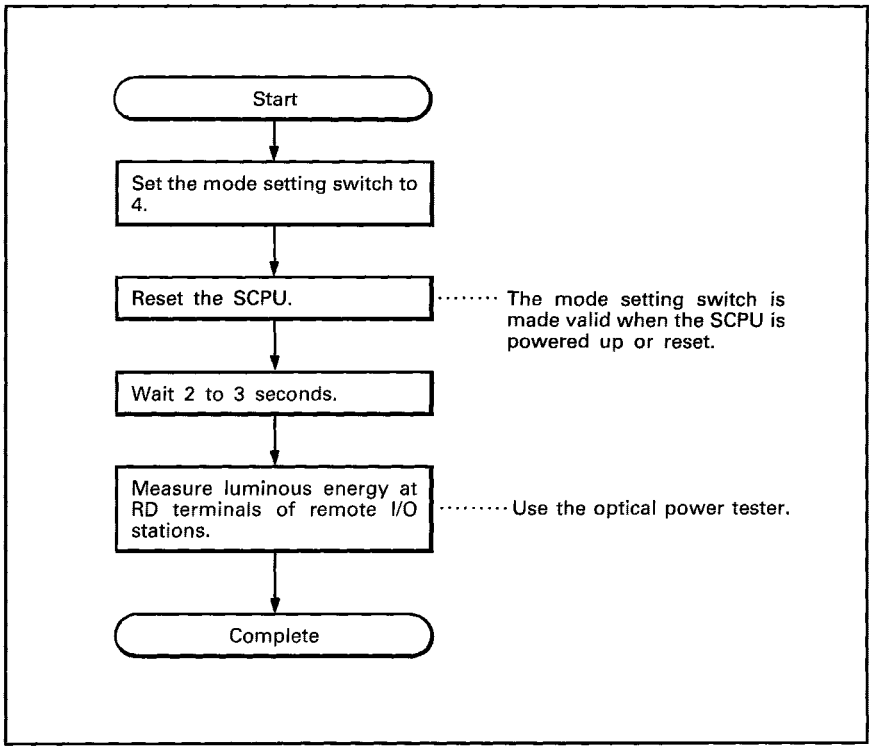


POINT

In an optical system, line check should only be performed after measuring the luminous energy of the loop.

5.11 A7BDE-A3N-PT32S3A MELSECNET/MINI-S3 Luminous Energy Check Mode

This mode is used to test the received luminous energy at the RD terminals, and to determine if the fiber optic cable connectors have been correctly fabricated. The general procedure is given in the flow chart below.



POINT

The luminous energy check is performed using an optical power tester available from Mitsubishi Electric.

5.12 A7LU1EP21/R21 Mode and Station Number Setting

The A7LU1EP21/R21 MELSECNET interface option card has eight operating modes: three on-line modes and five test modes. They are selected with a dial switch located near the top of the card. The function of each mode is described in the table below. For further details, please consult the Type Datalink User's Manual.

Dial Number	Mode Name	Description
0	On-Line Auto Return	Enables network communication, and will automatically return a normally operating station back Online after any faults have occurred.
1	On-Line No Auto Return	Enables network communication, but will only return a normally operating station back Online if the CPU is reset after any faults have occurred.
2	Off-Line	Disables communication with the network. If the station is the network master, the entire network will also be disabled.
3	Forward loop test mode	Used to check all fiber optic cables and coaxial cables of the data link system; this mode checks the forward loop which is used for normal operation.
4	Reverse loop test mode	Used to check all fiber optic cables and coaxial cables of the data link system; this mode checks the reverse loop which is used for loop-back if an error occurs.
5	Station-to-station test mode (master station)	Used to check the lines between two stations; sets the lower numbered station to main station, and the higher numbered station to subordinate.
6	Station-to-station test mode (slave station)	
7	Self-loop test mode	Enables self-checking of the sending-receiving hardware.
8 to F	Not Used	

POINT

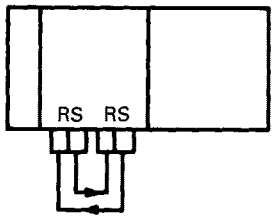
If the A7LU1EP21/R21 is installed, but communication via MELSECNET is not required:-

1. **Set the A7LU1EP21/R21 mode to Off-Line. If not, a link parameter error will be indicated. This does not affect the sequence program operation.**

2. **If MELSECNET is not connected, the status indicated by the LEDs must be regarded as indeterminate. Correct operation of the link module may be checked using the loop-back test.**

5.13 MELSECNET Self Loop-Back Test

The self loop-back test is used to check the transmitting and receiving circuits of the A7LU1EP21/R21. Data is sent from the transmitting terminal of the forward loop, to the receiving terminal of the forward loop, and must be received within a pre-set period of time. This test may also be performed for the reverse loop, e.g.



1) Test status

- Connect a cable from the host station forward loop sending side to its forward loop receiving side and connect a cable from the reverse loop sending side to the reverse loop receiving side.
- Set the station to STOP. (For a remote I/O station, set master station to STOP.)
- Set the mode select switch to "7" and reset.

2) Test result

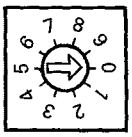
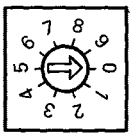
Determine the test result by the LEDs on the front of the link unit.

- For normal status, the six LEDs, "CRC", "OVER", "AB.IF", "TIME", "DATA", and "UNDER" flicker in order.
- If an error occurs, one of the LEDs is lit and the test is stopped. (For error indication, refer to A7LU1P21/R21 Nomenclature.

  Example: When the forward loop is broken, the "F.LOOP" or "TIME" LEDs are lit.

5.14 A7LU1EP21/R21 Station Number Setting

(1) The following table provides information concerning the setting of station numbers.

Dials	Description
X10  X1 	(1) X10 switch: To set "10's" of the station number. (2) X1 switch: To set "1's" of the station numbers. (3) Setting for the master station is (00). (4) Settings for local stations are between (01) and (64).

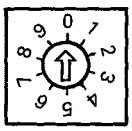
(2) The station number dial is set to (00) when shipped.

(3) Please refer to the Type Data Link Users Manual for instructions related to station number setting, when the PC is configured within MELSECNET.

5.15 A7BDE-A3N-B.C Board Number and I/O Port Number Setting

The board number setting specifies the I/O Port Number address, and a 16K Byte memory area of the PC-AT® to be accessed by the Device Driver. Each board number setting has a corresponding I/O Port Number Address that is allocated to the A7BDE-A3N-B.C. In addition, an offset to this address may be specified by means of a "jumper connector". (set to either 100H or 300H).

(1) The following table provides information regarding the board number settings, the corresponding memory area head address, and I/O Port Numbers.

Dial	Dial Number	Memory Area Head Address	I/O Port Number Head Address		
			100H	300H	*1
	0	D0000H	100H	300H	
	1		1100H	1300H	
	2	D4000H	2100H	2300H	
	3		3100H	3300H	
	4	D8000H	4100H	4300H	
	5		5100H	5300H	
	6	DC000H	6100H	6300H	
	7		7100H	7300H	
	8	DO NOT SET			
	9				

*1 Jumper Setting (100H or 300H)

(2) When setting the dial numbers, ensure that the new settings do not conflict with those on previously installed option cards. The board number must be set within the range 0 to 7.

(3) The dial number is set to zero when shipped.

(4) The jumper is set to 100H when shipped.

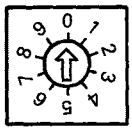
POINT

The above table shows the actual I/O port memory locations corresponding to the dial and jumper settings. Please note that the CONFIG.SYS file requires the dial number (0-7) and jumper (100H or 300H) settings, not the actual I/O port head address.

5.16 A7BDE-A3N-B.C Board IRQ Number Setting

The board IRQ number indicates which option board is accessing the operating system.

(1) The following table gives the allowable A7BDE-A3N-B.C IRQ identification numbers.

Dial	Dial Number	IRQ Number
	0	3
	1	4
	2	5
	3	7
	4	10
	5	11
	6	12
	7	15
	8	DO NOT SET
	9	

(2) When setting the dial numbers, ensure that the A7BDE-A3N-B.C IRQ numbers do not conflict with the settings of other option boards. Check that only the numbers (0) to (7) have been used.

(3) The dial number is set to zero when shipped.

POINT

Ensure that the IRQ number set for the A7BDE-A3N-B.C does not conflict with those previously used or reserved for other applications. Please consult the documentation that accompanied the computer for information on reserved IRQ numbers.

5.17 A7BDE-A3N-B.C ROM/RAM Specification

The A7BDE-A3N-B.C has a bank of DIP switches located near the top of the card. These are used to specify the type of memory being used, either ROM or RAM, and also RAM memory location ranges to be write protected. By write-protecting RAM memory locations, data such as sequence programs and parameters cannot be accidently changed or corrupted by malfunctioning peripheral equipment. Details are provided in the table below.

ROM/RAM Switch Memory Protect Switch	Switch No.	Description	Switch Setting Status	
			ON	OFF
→ ON 1 2 3 4 5 6 7 8	1	ROM/RAM switching	RAM operation	ROM operation
	2	Protect 0 to 16 KB of memory	Protects memory	Does not protect memory
	3	Protect 16 to 32 KB of memory		
	4	Protect 32 to 48 KB of memory		
	5	Protect 48 to 64 KB of memory		
	6 ~ 8	Not Used		

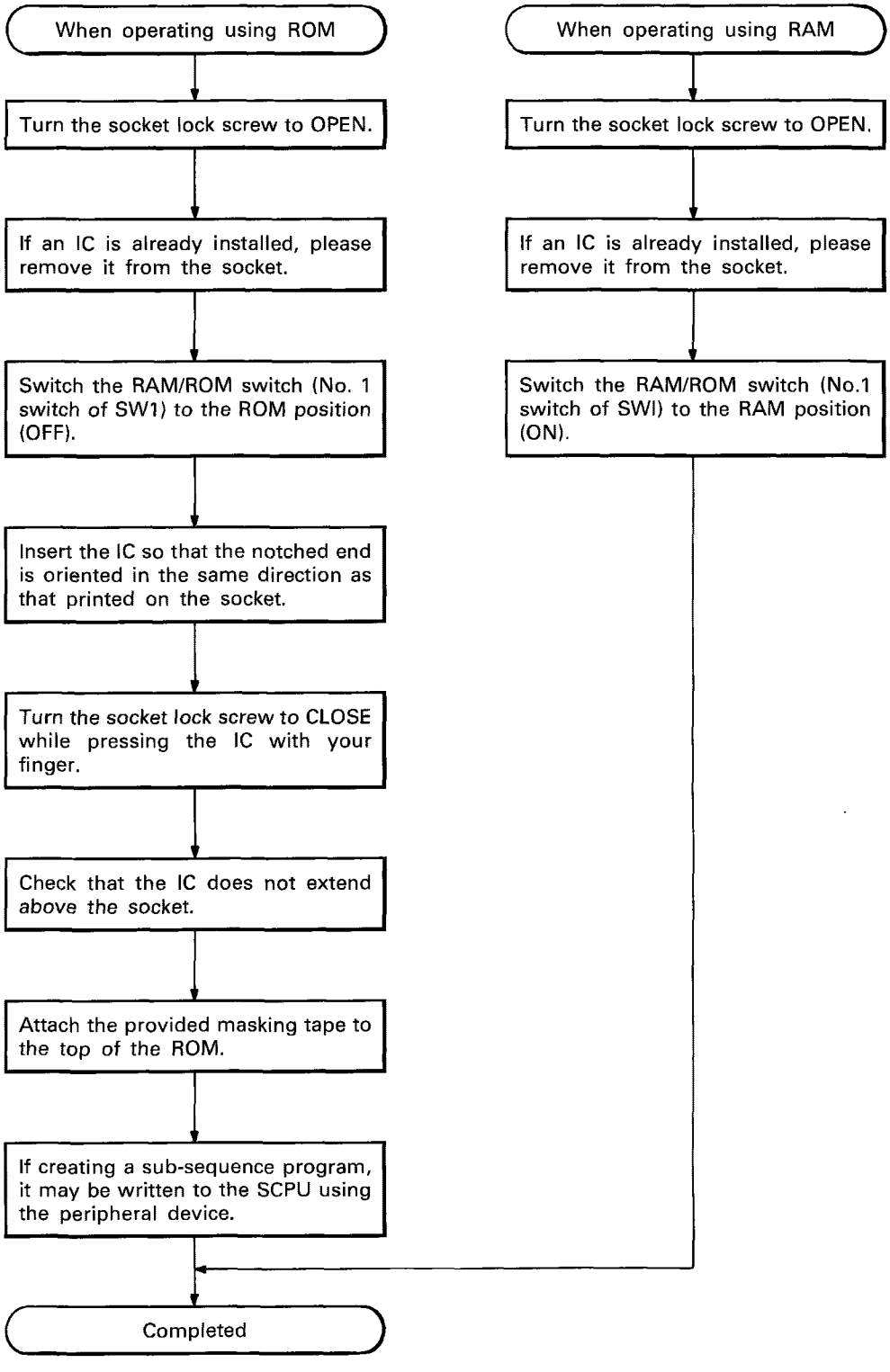
POINT

(1) Set memory protect settings taking into consideration the addresses (step numbers) of each memory area (sequence program, microcomputer program, sub-sequence program, comment, sampling trace, status switch, and file register).

(2) Do not use the memory protect function when executing sampling tracing and status latching. Use of the memory protect function will prevent the data from being stored in the memory.

5.18 A7BDE-A3N-B.C ROM Installation

The flow chart below gives the correct procedure when installing ROM.



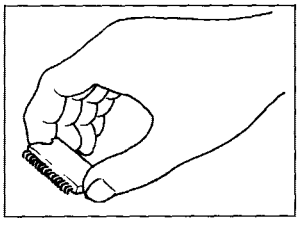
*1 This is necessary since writing the main sequence program in the ROM results in the addresses for storing a sub-sequence program to be changed.

POINT

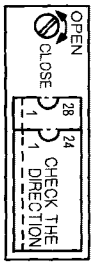
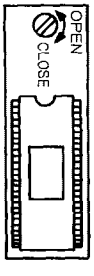
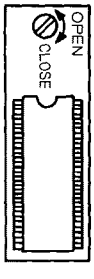
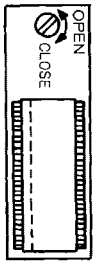
Installation of ROM

The following explains how the ROM should be mounted in the ROM sockets.

(a) How to hold the IC
Touching the leads of the memory chip can result in destruction of the memory due to static electricity. The pins could also be bent, preventing their proper insertion. It is recommended that an IC be held in the manner shown below.

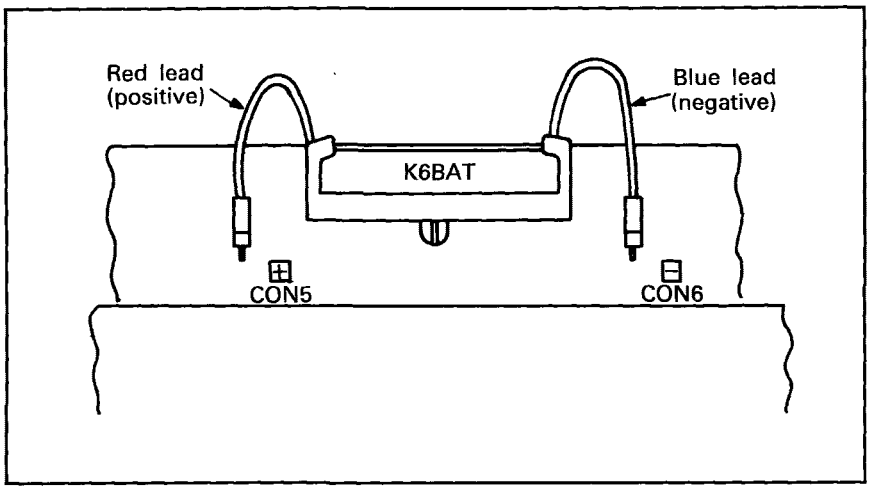


(b) Correct mounting direction of the IC
The memory chips will be destroyed if the memory chips are installed in the wrong direction and power is turned ON. The memory sockets, EP-ROM and IC-RAM are provided with notch marks which should be aligned correctly when installing the memory chips.

Socket	EP-ROM	IC-RAM	
		Notch type	Broken line type
			

5.19 A7BDE-A3N-B.C Battery Installation

The correct battery installation method is shown in the diagram below.



POINT

The leads of the K6BAT should be removed to prevent the battery losing its charge during shipment or storage. The battery leads need only be connected when the RAM memory back-up, or real time clock functions are required.

Replacement of Battery

The special auxiliary relays M9006 and M9007, are switched on to indicate that the battery life has reduced to a minimum value, as indicated below and it must be replaced if continued power failure RAM and/or data back-up is required.

Even if these special relays turn on, the contents of the program and power failure compensation are not lost immediately. However, if the ON state is overlooked, the PC RAM memory contents may be lost.

Battery Life (Total power failure time) [Hr]		
Guaranteed value (Min)	**Actually applied value (Typ)**	**Remaining time after M9006, M9007 are switched ON.**
12000 Hours	43200 Hours	240 Hours

5.20 Option Card Installation

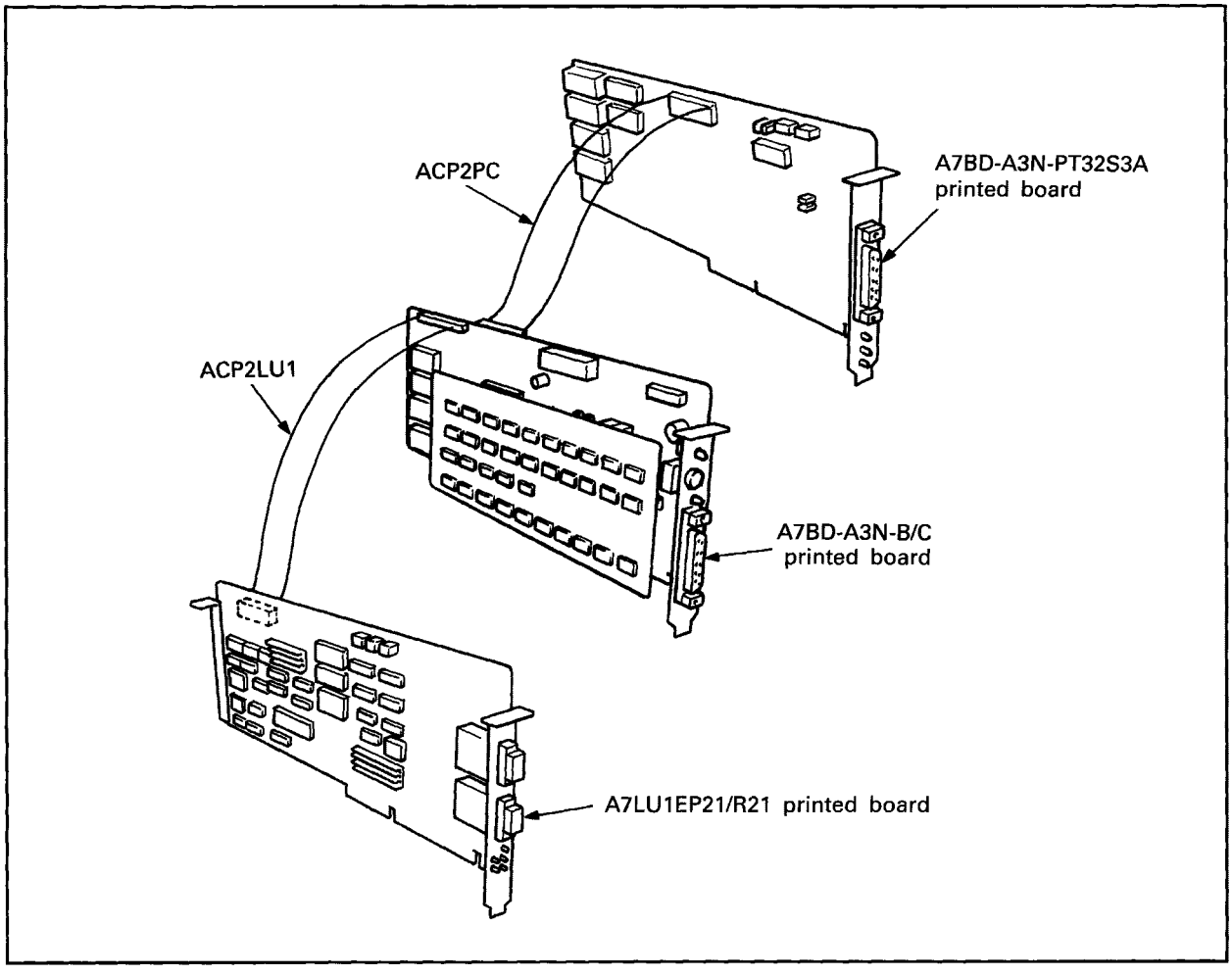
The three option cards are connected together using the cables ACP2LU1 and ACP2PC. Due to the positioning of the cable sockets, installation of the cards into the PC, must be performed in a particular order. For example:

The A7BDE-A3N-PT32S3A is installed into option slot eight.

The A7BDE-A3N-B.C is installed into option slot seven.

The A7LU1EP21/R12 is installed into option slot six.

The diagram below gives the general configuration of the three option cards, when installed together.



5.21 System Software Driver Entry Method

This section describes the procedure for installing the Driver software into the PC.

After loading the Driver system file onto the hard-disk, add the following, using a text editor, to the CONFIG.SYS file on the operating system data disk.

DEVICE=[Drive:] [Path] driver name INT-A__BD_ INT-B_ _00H

(1) INT-A__ ····· Software Interupt number for use when the application requests the driver to perform processing.
Set between 60H and FFH.

(2) BD_ ····· Option Board Number switch setting.
Set between 0 and 7.

(3) INT-B_ ····· Option Board Interrupt (IRQ) setting.
Set between 0 and 7.

(4) _00H····· I/O Port Number Offset.
Set to 100H or 300H.

Example.

DEVICE=C:\MA3N.SYS INT-A90 BD1 INT-B4 100H

i.e. (a) Driver-MA3N.SYS is loaded in the root directory of drive C:\
(b) The option board has been assigned interrupt vector 90H.
(c) The option board number is set to 1.
(d) The option board interrupt number is set to 4. (IRQ 10.)
(e) The I/O port number offset is set to 100H.

POINT

The following message is displayed at normal installation.
MELSEC DRIVER M-A3N.SYS Ver. 00A.
For further driver messages at start-up, please see the appendix.

MEMO

6. PROGRAMMING

This chapter describes the programming procedure of the A7BDE-A3N-PT32S3. There are two main sections. The first provides details on the software driver interface formats, should an assembler code custom access library be written, e.g. to be used with a PASCAL compiler. The second section gives specifications and program examples on the supplied access function library. This library is compatible with the Microsoft-C® compiler and linker.

6.1 Main Library Processes

No.	Processing Timing	Library Processing	System Call
1	First call	Checks that the driver is being started up. Reads the INT number entered into CONFIG. SYS file. Performs the same processing as second and subsequent calls.	Opens the driver. Reads from the driver using I/O control.
2	Second and subsequent calls	Pushes arguments onto stack. Generates interruption in accordance with INT number. Restores stack.	

6.2 The Software Driver Functions

The A7BDE-A3N-PT32S3 driver software has five functions to link the access function library with the option board, and thereby allow access to the SCPU, MCPU, high speed device memory, and stations of MELSECNET.

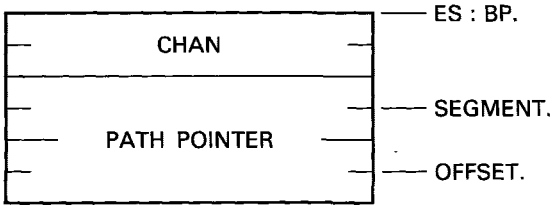
NUMBER	NAME	CODE	FUNCTION
(1)	OPEN	1H	Opens the communication line to start operation of the A7BDE-A3N-PT32S3.
(2)	CLOSE	2H	Closes the communication line when terminating operation of the A7BDE-A3N-PT32S3.
(3)	RECEIVE	3H	Enables reading of data from the host A7BDE-A3N-PT32S3 and stations of MELSECNET.
(4)	SEND	4H	Enables writing of data to the host A7BDE-A3N-PT32S3 and stations of MELSECNET.
(5)	SYNC	5H	Enables synchronization of data read and write for RECEIVE or SEND.

6.3 Assembler Interface Specification - OPEN Function

Code 1H.

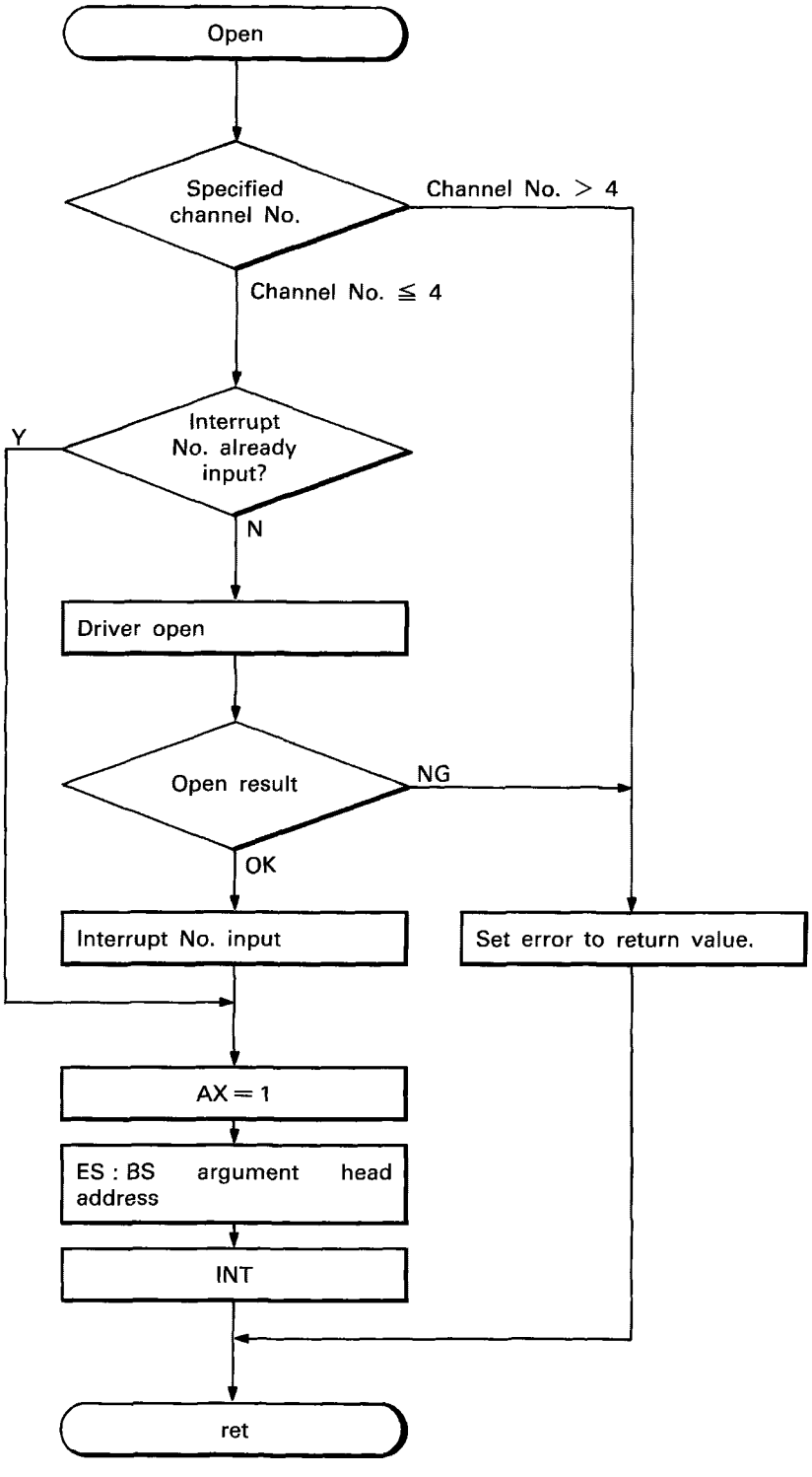
Call Procedure AX = 1. (OPEN function number)
ES : BP = Head address of argument.
INT = As set in CONFIG.SYS file. (60-FF).

Memory Status



Returned Value AX = Return Value. (For details see the error code list in the appendix.)

6.4 Open Processing

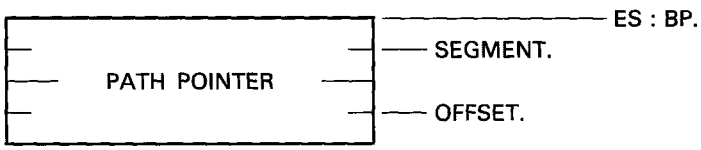


6.5 Assembler Interface Specification - CLOSE Function

Code 2H.

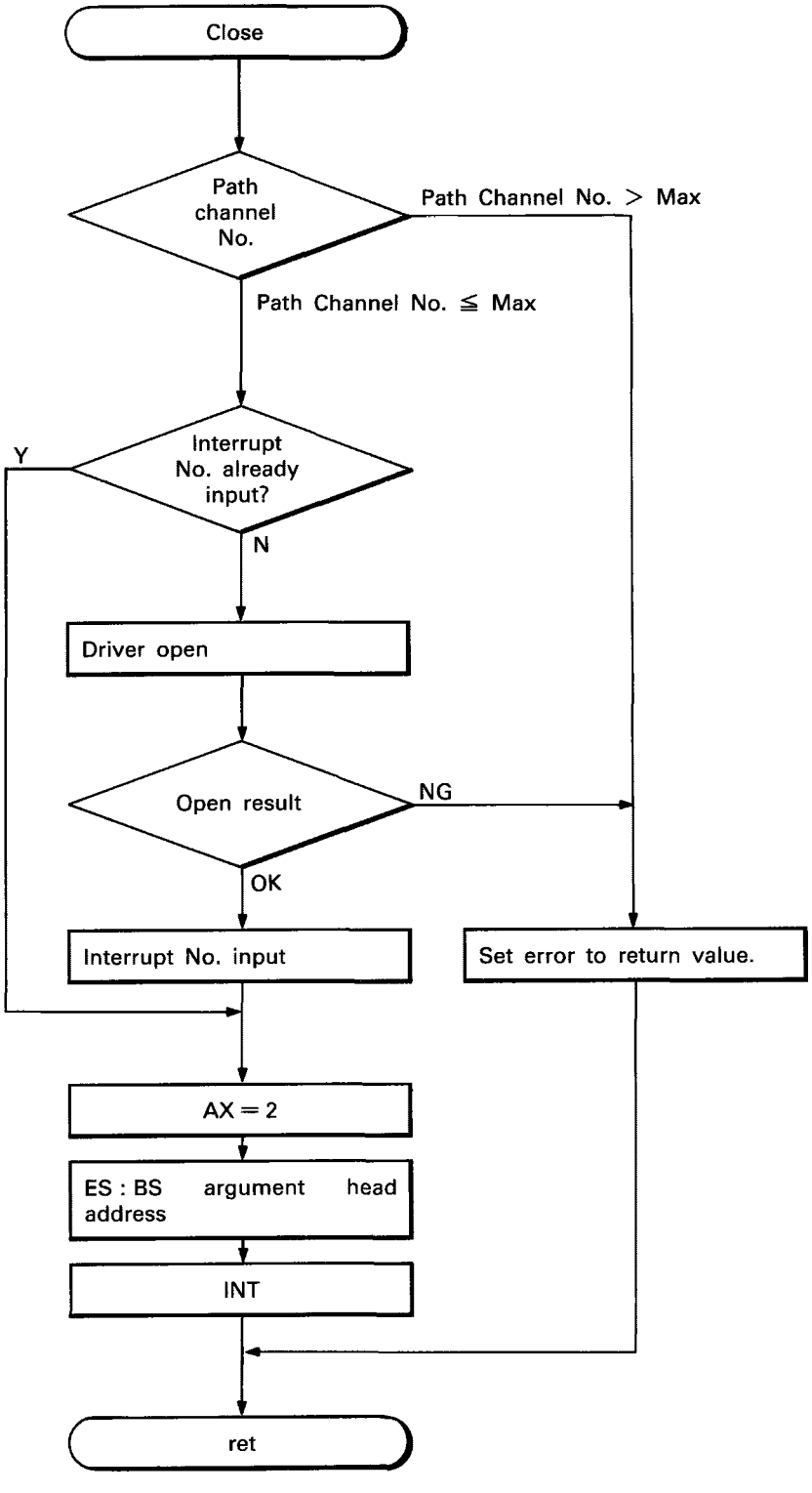
Call Procedure AX = 2. (CLOSE function number)
ES : BP = Head address of argument.
INT = As set in CONFIG.SYS file. (60-FF).

Memory Status



Returned Value AX = Return Value. (For details see the error code list in the appendix.)

6.6 Close Processing

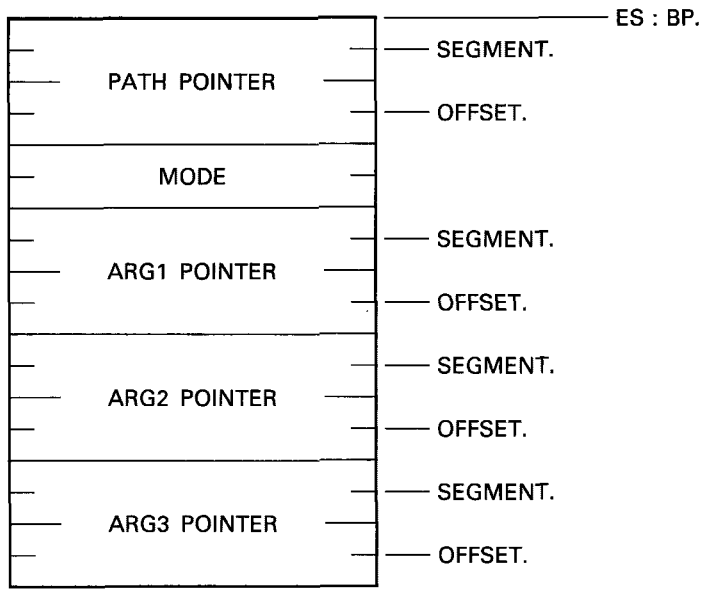


6.7 Assembler Interface Specification - RECEIVE Function

Code 3H.

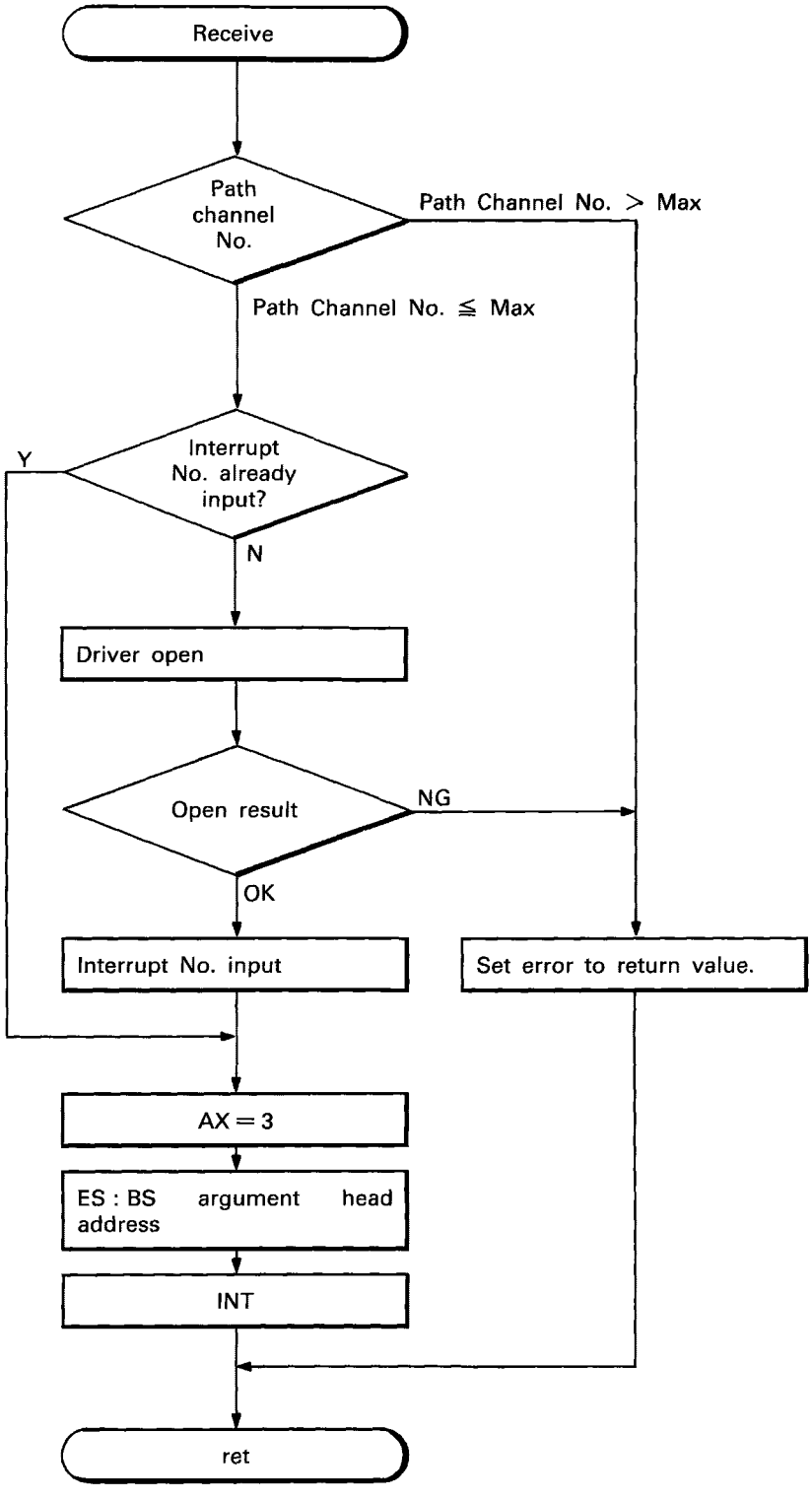
Call Procedure AX = 3. (RECEIVE function number)
ES : BP = Head address of argument.
INT = As set in CONFIG.SYS file. (60-FF).

Memory Status



Returned Value AX = Return Value. (For details see the error code list in the appendix.)

6.8 Receive Processing

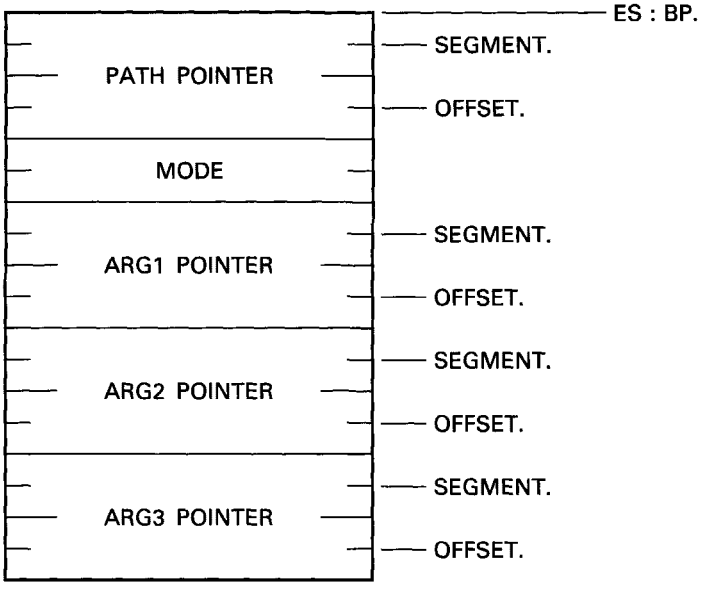


6.9 Assembler Interface Specification - SEND Function

Code 4H.

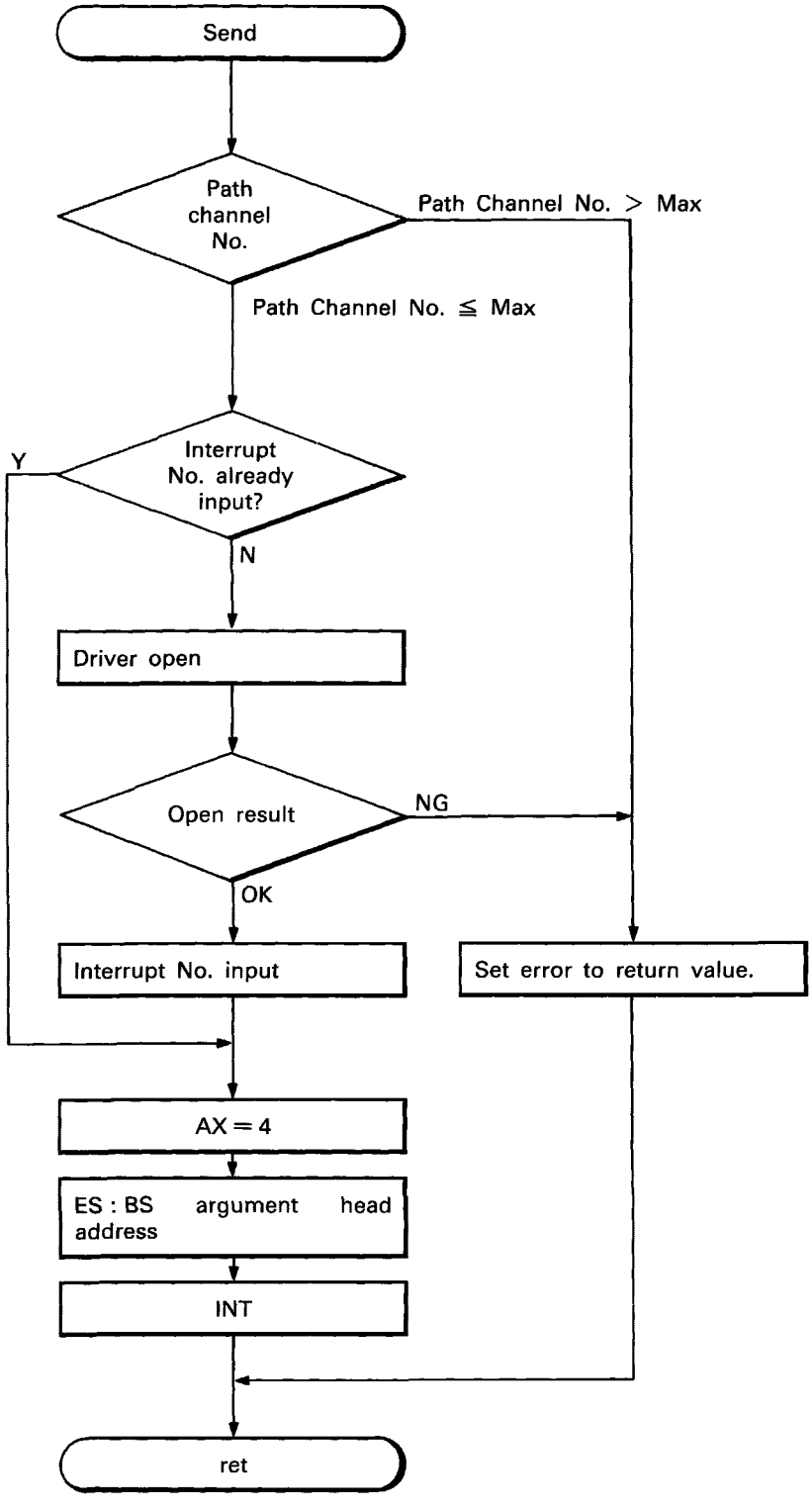
Call Procedure AX = 4. (SEND function number)
ES : BP = Head address of argument.
INT = As set in CONFIG.SYS file. (60-FF).

Memory Status



Returned Value AX = Return Value. (For details see the error code list in the appendix.)

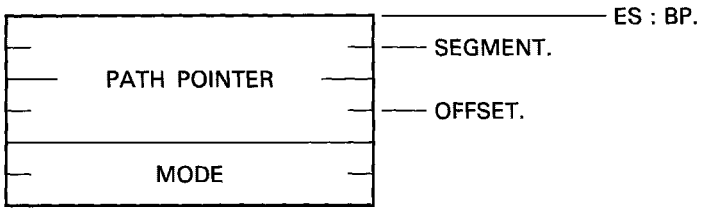
6.10 Send Processing



6.11 Assembler Interface Specification - SYNC Function

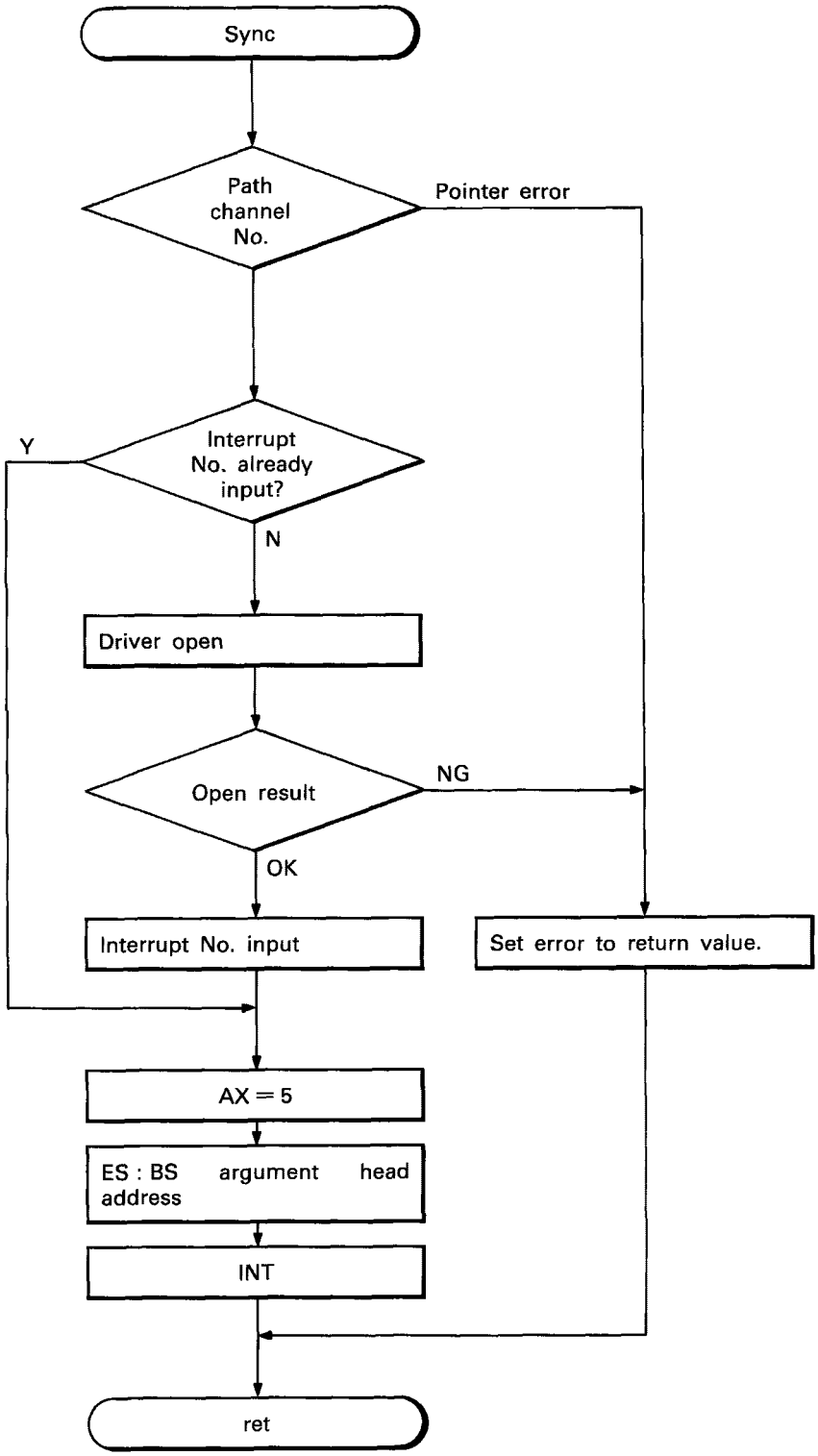
Code 5H.

Call Procedure AX = 5. (Complete synchronisation-function number)
ES : BP = Head address of argument.
INT = As set in CONFIG.SYS file. (60-FF).



Returned Value AX = Return Value. (For details see the error code list in the appendix.)

6.12 Sync Processing



6.13 The Access Function Library

The access function library consists of an include file and five functions:-

```
#include <nyuserc.h>

  nl1open.
  nl1close.
  nl1receive.
  nl1send.
  nl1sync.
```

These functions enable access to the host A7BDE-A3N-PT32S3, and stations of MELSECNET, or MELSECNET/MINI-S3.

The functions nl1open and nl1close start and finish communications. nl1open specifies the communications channel, i.e. access to the A7BDE-A3N-PT32S3, and receives a path line (path), to be used by the other functions. This path line remains open until terminated by nl1close.

Nl1send and nl1receive, transfer data to and from the PC application program and the host A7BDE-A3N-PT32S3. Both functions have five arguments, path, mod, arg1, arg2, and, arg3. arg1 is a structure that specifies the processing code of the called function, and if on a network, the PLC to be accessed. Each processing code has a set of arguments (arg2 and arg3), whose formats define a specific operation. e.g. batch read/write, remote run/stop/pause. The arguments take the form of a memory table, to which the relevant data needed to specify an operation, is written. The various argument formats are given in the proceeding section.

When sending or receiving data to and from MELSECNET stations, data transmission over the network, may cause long processing times, and a delay before the function return value is received. However, once the operation data has been sent to the A7BDE-A3N-PT32S3, transmission is performed independently of the PC. The 'mod' argument allows a return value to be immediately received, so that other program processing may continue. The application program may later enquire if the transmission of data has been completed, using the nl1sync function.

Include File <nyuserc.h>

The include file <nyuserc.h> defines the structure NLARG1 and the constant PATH. i.e.

```
typedef struct
        {
                short       demand;
                short       loop;
                short       station;
        }       NLARG1;
#define PATH    long;
```

The line:- #include<nyuserc.h> must be added to the other include file declarations in any user C application programs.

SPECIFICATION	nl1open
Function:	Opens communication line when starting operation of the A7BDE-A3N-PT32S3.

Syntax:	#include<nyuserc.h>
	ret=nl1open (chan, & path);

Remarks:	short ret — Returned value of function. short chan — Channel number setting. (0) 0.=A7BDE-A3N-PT32S3. PATH *path — Pointer of the opened path.
Returned Value:	A returned value of (0) indicates a normal termination. Other values indicate an abnormal termination. For details, see the error code list in the appendix.
Explanation:	After the line has been opened correctly, path (*path) is set. All communication driver functions use this path. This path remains effective until the line is closed with the nl1close function.

EXAMPLE — **nl1open**

```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;

main( )
{

  int chan;
  short ret;
  char ch;

  printf ("Open Path (Y/N)?\t");
  ch = getche ( );

  if (ch == 'Y' || ch == 'y')
  {

        chan = 0;
        ret = nl1open (chan, &path);
        printf ("\nReturn value (open) = %x\n", ret);
  }
  else
  {

        printf ("\nPath not closed.\n");
  }
}
```

SPECIFICATION	nl1close
Function:	Closes the communication line when terminating operation of an A7BDE-A3N-PT32S3.

Syntax:	#include<nyuserc.h>
	ret=nl1close (path);

Remarks:	short ret Returned value of function. path Pointer of the opened path.
Returned Value:	A returned value of (0) indicates a normal termination. Other values indicate an abnormal termination. For details, see the error code list in the appendix.
Explanation:	Closes the opened channel.

EXAMPLE **nl1close**

```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;

main( )
{

  short ret;
  char ch;

  printf ("Close Path (Y/N)?\t");
  ch = getche ( );

  if (ch == 'Y' || ch == 'y')
  {

        ret = nl1close (path);
        printf ("\nReturn value (close) = %x\n", ret);
  }
  else
  {

        printf ("\nPath not closed.\n");
  }
}
```

POINT

The function nl1close, should only be used after the channel has been opened by nl1open. If nl1close is processed before nl1open, an error value will be returned.

SPECIFICATION	nl1receive
Function:	Reads data from the A7BDE-A3N-PT32S3, and stations of MELSECNET.

Syntax:	#include<nyuserc.h>
	ret=nl1receive(path, mod, &arg1, arg2, arg3);

Remarks:	short ret — Returned value of function. PATH *path — Pointer of the opened path. short mod — Calling Mode. NLARG1 *arg1 — Argument 1 pointer. char *arg2 — Argument 2 pointer. char *arg3 — Argument 3 pointer.
Returned Value:	A returned value of (0) indicates a normal termination. Other values indicate an abnormal termination. For details, see the error code list in the appendix.
Explanation.	This function is used to read data from the host A7BDE-A3N-PT32S3, and stations of MELSECNET. The operation of the function is defined by four arguments, which have the following specification.
mod:	Mod specifies the calling mode of nl1receive. (0/1) (0) Wait for completion of communications processing. (1) To immediately receive a return value, and continue with additional programs. If ret is less than zero, communications processing is incomplete.

SPECIFICATION	nl1receive

arg1:	Argument one is a structure, as defined in the include file <nyuserc.h>, which specifies the request details. i.e. processing code, loop number, and station number.
	
	e.g. struct NL1LARGE {
	short demand;
	short loop;
	short station;
	};
	
	Where:
	
	demand = Processing code.
	loop; = Loop number. (set at 0)
	station; = Station number. (00 to 64)

arg2: arg3:	Arguments two and three specify the request and receive data location. The format of the memory tables depend upon the operation processing code. Examples are given in the following section.

SPECIFICATION	nl1send
Function:	Writes data to the host A7BDE-A3N-PT32S3, and stations of MELSECNET.

Syntax:	#include<nyuserc.h>
	ret=nl1send(path, mod, &arg1, arg2, arg3);

Remarks:	short ret — Returned value of function. path — Pointer of the opened path. short mod — Calling Mode NLARG1 arg1 — Argument 1 pointer. char arg2 — Argument 2 pointer. char arg3 — Argument 3 pointer.
Returned Value:	A returned value of (0) indicates a normal termination. Other values indicate an abnormal termination. For details, see the error code list in the appendix.
Explanation.	This function is used to write data to the host A7BDE-A3N-PT32S3 and stations of MELSECNET. The operation of the function is defined by four arguments, which have the following specifications.
mod:	Mod specifies the calling mode of nl1send. (0/1) (0) Wait for completion of communications processing. (1) To immediately receive a return value, and continue with additional programs. If ret is less than zero, communications processing is incomplete.

SPECIFICATION	nl1send

arg1:	Argument one is a structure, as defined in the include file <nyuserc.h>, which specifies the request details. i.e. processing code, loop number, and station number.
	e.g. struct NL1LARGE {
	short demand;
	short loop;
	short station;
	};
	Where:
	demand = Processing code.
	loop; = Loop number. (set at 0)
	station; = Station number. (00 to 64)

arg2: arg3:	Arguments two and three specify the request and send data. The format of the memory tables depend upon the operation processing code. Examples are given in the following section.

SPECIFICATION	nl1sync
Function:	Used in conjunction with nl1send and nl1receive, to determine if communications processing is complete.

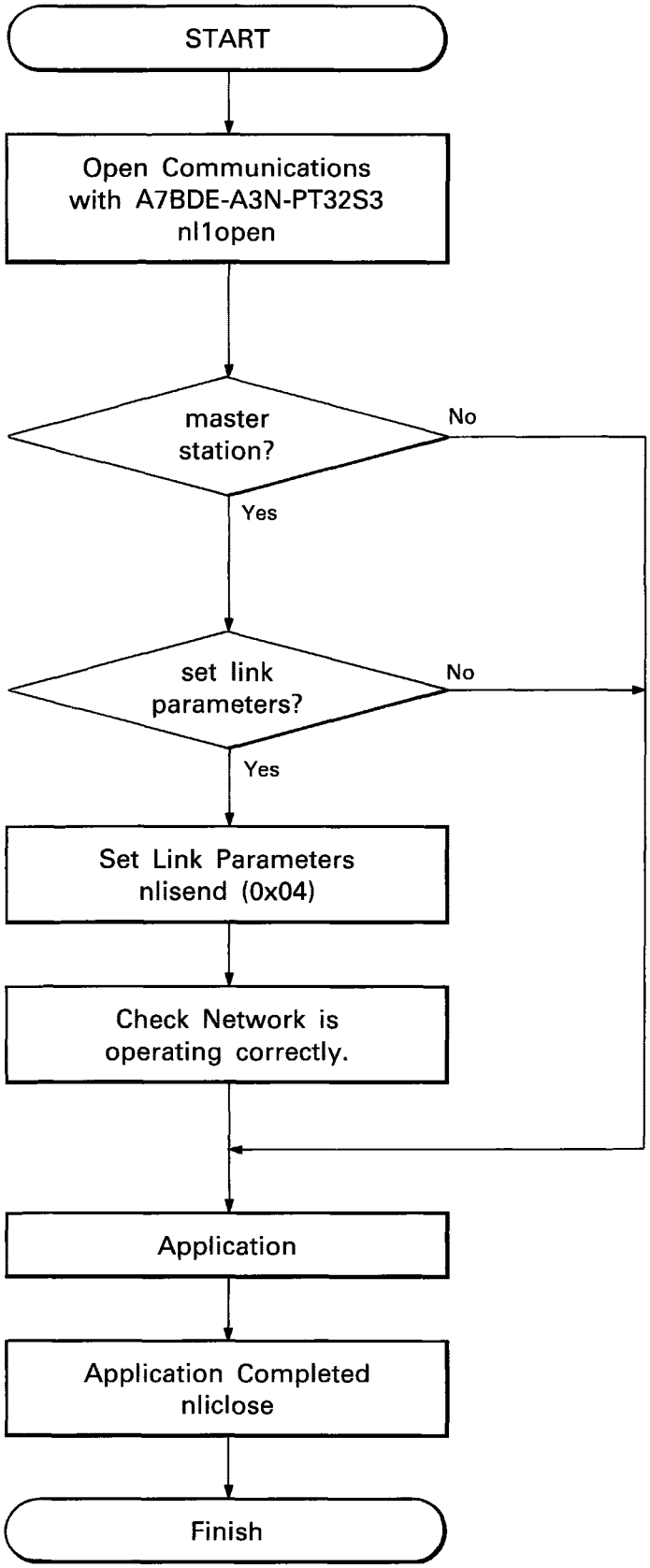
Syntax:	#include<nyuserc.h>
	ret=nl1sync (path, mod);

Remarks:	short ret — Returned value of function. short mod — Calling Mode. path — Pointer of opened path.
Returned Value:	A returned value of (0) indicates a normal termination. A returned value of (−1) indicates that communications processing is incomplete. All other return values indicate abnormal termination.
Explanation.	This function is used to sense if communication via MELSECNET, is complete, and that all data has been transferred.
mod:	Mod specifies the calling mode of nl1sync. (0/1) (0) Wait for completion of communications processing. (1) To immediately receive a return value, and continue with additional programs. If ret is less than zero, communications processing is incomplete.

EXAMPLE **nl1sync**

```
mod = 1;
mode = 1;
ret = nl1receive (path, mode, &arg1, arg2, arg3);
if (ret>0)
{
        Error processing
}
else if (ret = = 0)
        Normal termination
else
while (ret<0)
{
        Other processing
        ret = nl1sync (path, mode);
}
if (ret>0)
{
        Error processing
}
```

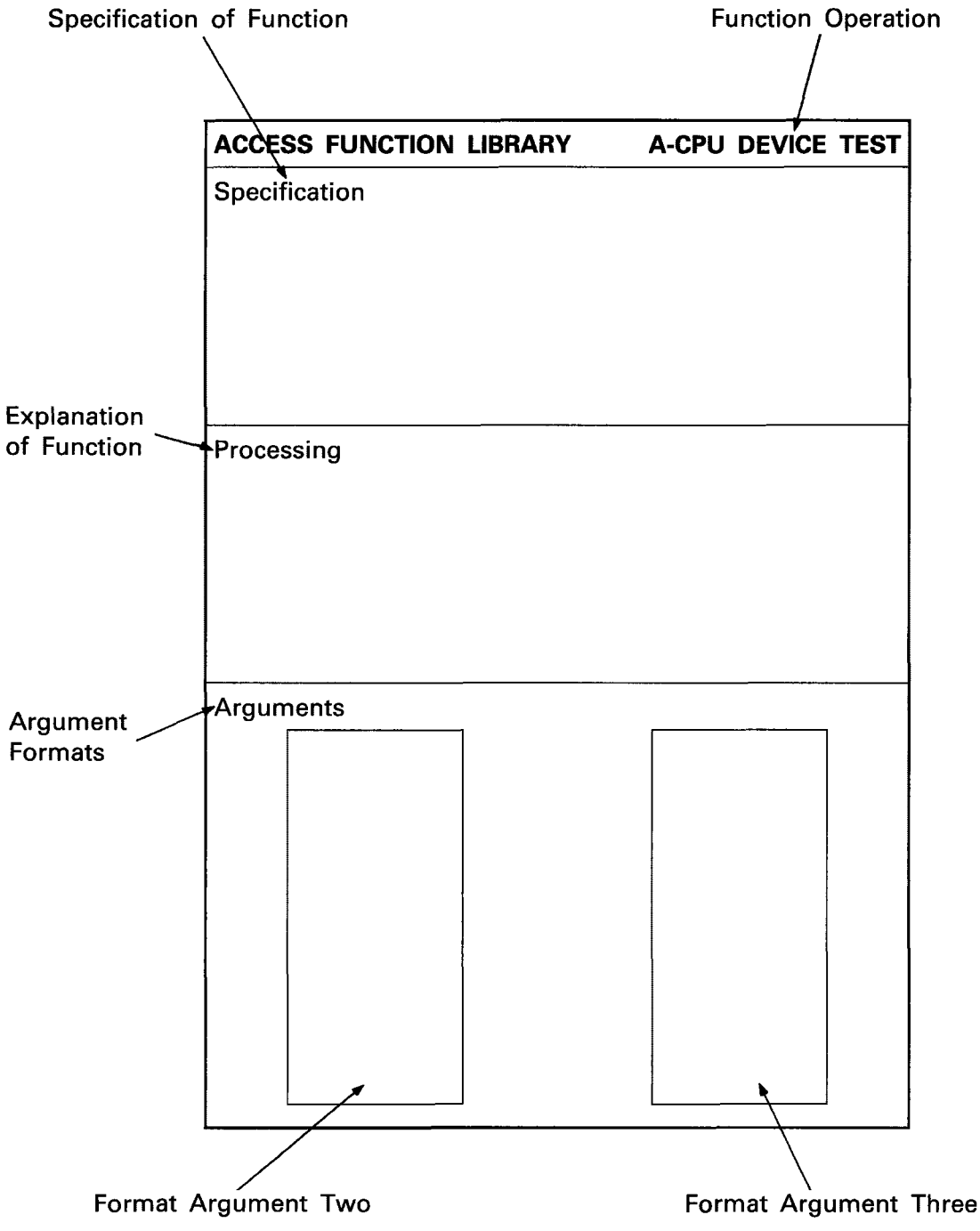
6.15 Programming Procedure



6.16 Access Function Specification And Example Sheets

No.	Item	Function	Processing	A3N MASTER STATION			A3N SLAVE STATION			Processing Code (HEX)	Remark
				HOST	SLAVE		HOST	MASTER			
					ACPU	A7BDE		ACPU	A7BDE		
1	ACPU access	ACPU memory access	Batch read	○	○	—	○	○	—	2	Page 6-26
2			Batch write	○	○	—	○	○	—	4	Page 6-28
3			Random read	○	○	—	○	○	—	5	Page 6-30
4			Random write	○	○	—	○	○	—	6	Page 6-32
5		ACPU sequence program access	Batch read	○	○	—	○	○	—	1	Page 6-34
6			Batch write	○	○	—	○	○	—	3	Page 6-36
7			SCPU Interrupt program starting	○	—	—	○	—	—	100	Page 6-38
8		ACPU control	Remote RUN/STOP/PAUSE	○	○	—	○	○	—	18	Page 6-40
9			Requested ACPU check	○	○	○	○	○	○	8	Page 6-42
10			Parameter analysis request	○	○	—	○	○	—	27	Page 6-44
11	Special module access	Special module access	Shared memory batch read	○	○	—	○	○	—	10	Page 6-46
12			Shared memory batch write	○	○	—	○	○	—	12	Page 6-48
13	A7BDE-A3N -PT32S3 General Access	IFMEM Access	Batch read	○	—	—	○	—	—	200	Page 6-50
14			Batch write	○	—	—	○	—	—	201	Page 6-52
15			Random read	○	—	—	○	—	—	202	Page 6-54
16			Random write	○	—	—	○	—	—	203	Page 6-56
17			IFMEM input X write	○	—	—	○	—	—	204	Page 6-58
18			IFMEM output Y read	○	—	—	○	—	—	205	Page 6-60
19		High-Speed Device Memory Access	Transfer setting for A3N device memory	○	—	—	○	—	—	803	Page 6-62
20			Batch read	○	—	—	○	—	—	206	Page 6-64
21			Batch write	○	—	—	○	—	—	208	Page 6-66
22			Random read	○	—	—	○	—	—	207	Page 6-68
23			Random write	○	—	—	○	—	—	209	Page 6-70
24	A7BDE-A3N -PT32S3 CARD STATUS MONITOR and CONTROL	A7BDE-A3N -PT32S3 Card Status Monitor and control	Reading LED status	○	—	—	○	—	—	700	Page 6-72
25			Reading switch status	○	—	—	○	—	—	701	Page 6-74
26			A3N board version read	○	—	—	○	—	—	702	Page 6-76
27			Resetting A3N board	○	—	—	○	—	—	800	Page 6-78
28			Resetting A3N indicator	○	—	—	○	—	—	80A	Page 6-80
29	General data	General data	Data free transmission	—	—	○	—	—	○	40	Page 6-82

6.17 Explanation of Access Function Specification Sheets



POINT

Please note that the arguments are set in multiples of bytes.

ACCESS FUNCTION LIBRARY **A-CPU MEMORY BATCH READ**

Specification

Function:	A-CPU Memory Access
Application:	Batch Read
Function Name:	nl1receive
Processing Code:	0x02
Driver Function Number:	3H

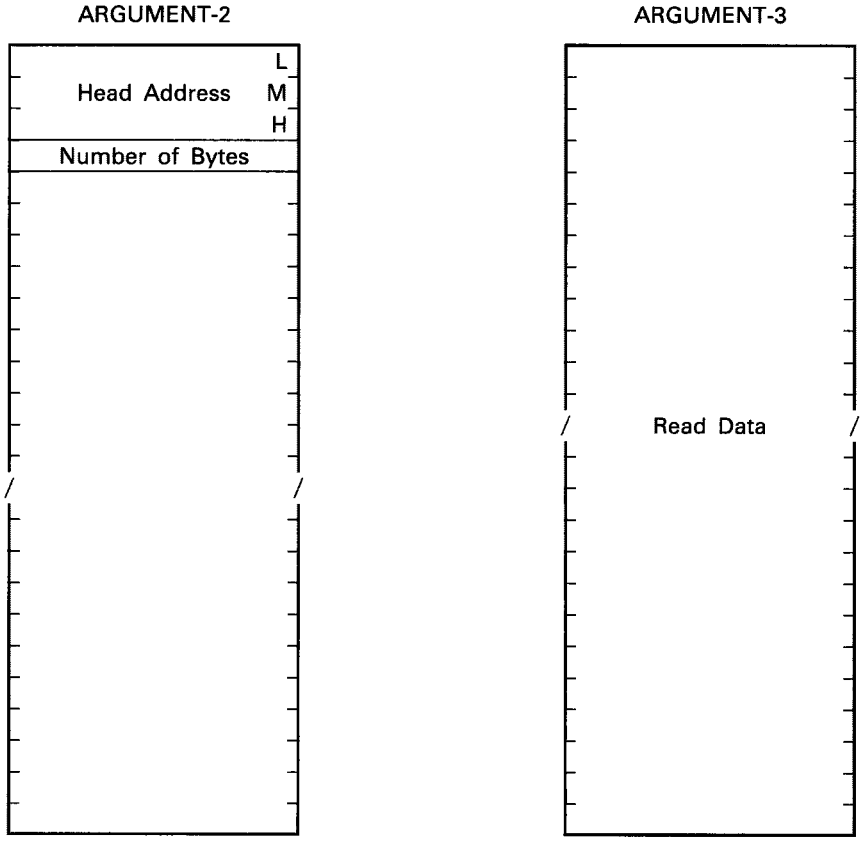
Processing

Processing code 0x02 enables batch-read of the A7BDE-A3N-PT32S3 SCPU and A-Series PLC memory locations. i.e. status of devices, system data table, parameter settings, micro-program area, file registers etc. Please see the appendix for head addresses and read data formats.

Argument two specifies the head address and number of bytes to be read. (maximum of 128 bytes)

Argument three receives the returned data. Format is dependent on the requested data.

Argument Formats



EXAMPLE **A-CPU MEMORY BATCH READ**

```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;
NLARG1 arg1;

main ( )
{
  int chan, mod, i;
  short ret;
  unsigned char *arg2;
  unsigned char *arg3;
  char buff2 [512];
  char buff3 [512];

  arg2 = buff2;
  arg3 = buff3;

/* OPEN OF I/F BOARD */

/* A-CPU MEMORY BATCH READ */

/* This program reads and displays the status of inputs */
/* X0 to X3F, of station one of MELSECNET. */

  mod = 0;

  arg1. demand = 0x02;
  arg1. loop = 0x00;
  arg1. station = 0x01;

  buff2 [0] = 0x00;
  buff2 [1] = 0x08;
  buff2 [2] = 0x00;
  buff2 [3] = 0x10;

  ret = nl1receive (path, mod, &arg1, arg2, arg3);
  printf ("Return value (ACPU batch rd) = %X\n", ret);

  i = 0;
  while (i < 16)
  {
    printf ("buff3 [%3d] = %4X\n", i, buff3 [i]);
    i = i + 2;
  }

/* CLOSE */

}
```

ACCESS FUNCTION LIBRARY	A-CPU MEMORY BATCH WRITE

Specification

Function:	A-CPU Memory Access
Application:	Batch Write
Function Name:	nl1send
Processing Code:	0x04
Driver Function Number:	4H

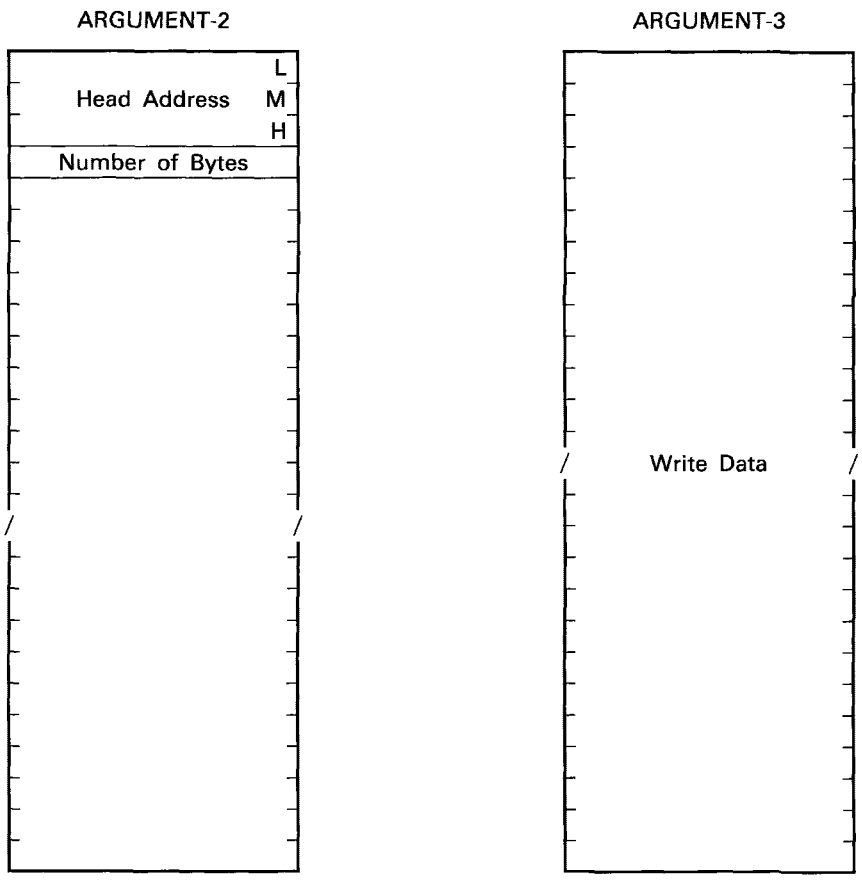
Processing

Processing code 0x04 enables batch-write of the A7BDE-A3N-PT32S3 SCPU and A-Series PLC memory locations. i.e. status of devices, parameter settings, micro-program area, file registers etc. Please see the appendix for head addresses and read data formats.

Argument two specifies the head address and number of bytes to be written. (maximum of 128 bytes)

Argument three contains the write data.

Argument Formats



EXAMPLE **A-CPU MEMORY BATCH WRITE**

```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;
NLARG1 arg1;

main ( )
{
  int chan, mod, i;
  short ret;
  unsigned char *arg2;
  unsigned char *arg3;
  char buff2 [512];
  char buff3 [512];

  arg2 = buff2;
  arg3 = buff3;

/* OPEN OF I/F BOARD */

/* A-CPU MEMORY BATCH WRITE */

/* This program writes the bit code 0xff to outputs Y40-Y7f */
/* of the host A7BDE-A3N-PT32S3. i.e. switches them all 'on'. */

  mod = 0;

  arg1. demand = 0x04;
  arg1. loop = 0x00;
  arg1. station = 0xff;

  buff2 [0] = 0x1c;
  buff2 [1] = 0x82;
  buff2 [2] = 0x00;
  buff2 [3] = 0x10;

  i = 0;
  while (i < 16)
  {
    buff3 [ i ] = 0xff:
    i = i + 2;
  }

  ret = nl1send (path, mod, &arg1, arg2, arg3);
  printf ("Return value (ACPU batch wr) = %X\n", ret);

/* CLOSE */

}
```

ACCESS FUNCTION LIBRARY	A-CPU MEMORY RANDOM READ

Specification

Function:	A-CPU Memory Access
Application:	Random Read
Function Name:	nl1receive
Processing Code:	0x05
Driver Function Number:	3H

Processing

Processing code 0x05 enables random-read of the A7BDE-A3N-PT32S3 SCPU and A-CPU device memory locations. i.e. X/Y inputs/outputs, relays, registers, timers/counters etc. Please see the appendix for head addresses and read data format.

Argument two specifies the number of points and their corresponding memory addresses. The maximum number of points that may be set in one argument is 40. Each point is one byte.

Argument three receives the returned data.

Argument Formats

ARGUMENT-2

Number of Point	
Address First Point	L M H
Address Second Point	L M H
/	/

ARGUMENT-3

First Point Data
Second Point Data
/

EXAMPLE **A-CPU MEMORY RANDOM READ**

```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;
NLARG1 arg1;

main ( )
{
  int chan, mod, i;
  short ret;
  unsigned char *arg2;
  unsigned char *arg3;
  char buff2 [512];
  char buff3 [512];

  arg2 = buff2;
  arg3 = buff3;

/* OPEN OF I/F BOARD */

/* A-CPU MEMORY ACCESS RANDOM READ */

/* This program reads random data from station one, */
/* specifically the status of X0 to X7. */

  mod = 0;

  arg1. demand = 0x05;
  arg1. loop = 0x00;
  arg1. station = 0x01;

  buff2 [0] = 0x01;
  buff2 [1] = 0x00;
  buff2 [2] = 0x80;
  buff2 [3] = 0x00;

  ret = nl1receive (path, mod, &arg1, arg2, arg3);
  printf ("Return value (ACPU rnd rd) = %X\n", ret);

  printf ("buff3 [0] = %X\n", buff3 [0]);

/* CLOSE */

}
```

ACCESS FUNCTION LIBRARY	A-CPU MEMORY RANDOM WRITE

Specification

Function:	A-CPU Memory Access
Application:	Random Write
Function Name:	nl1send
Processing Code:	0x06
Driver Function Number:	4H

Processing

Processing code 0x06 enables random-write to the A7BDE-A3N-PT32S3 SCPU and A-CPU device memory locations. i.e. X/Y inputs/outputs, relays, registers, timers/counters etc. Please see the appendix for head addresses and write data format.

Argument two specifies the number of points (maximum of 24)

Argument three specifies the sent data. Each point is specified as follows: Each Point is one byte.

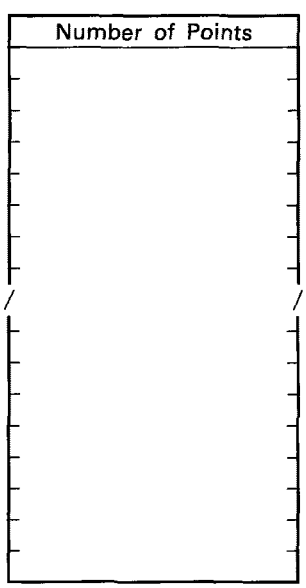
Designation: (0) Bit Set ORs contents and bit pattern data.
(1) Bit Reset ANDs contents and bit pattern data.
(2) Byte Write Writes bit pattern data to address.

Address: Memory address of specified device

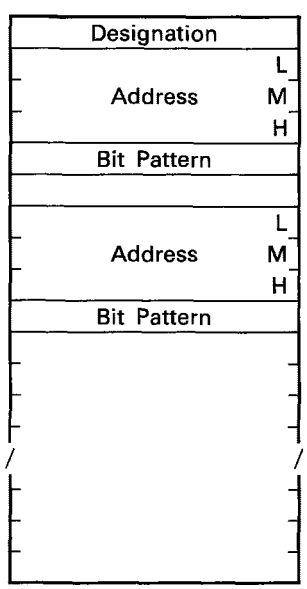
Bit Pattern: Data to be written to the device (1 = "ON")
(0 = "OFF")

Argument Formats

ARGUMENT-2



ARGUMENT-3



EXAMPLE **A-CPU MEMORY RANDOM WRITE**

```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;
NLARG1 arg1;

main ( )
{
  int chan, mod, i;
  short ret;
  unsigned char *arg2;
  unsigned char *arg3;
  char buff2 [512];
  char buff3 [512];

  arg2 = buff2;
  arg3 = buff3;

/* OPEN OF I/F BOARD */

/* A-CPU MEMORY ACCESS RANDOM WRITE */

/* This program writes 0xF0 to outputs Y40-Y48, and 0xBBAA */
/* to data register D0, of station one of MELSECNET. */

  mod = 0;

  arg1. demand = 0x06;
  arg1. loop = 0x00;
  arg1. station = 0x01;

  buff2 [0] = 0x01;

  buff3 [0] = 0x02;
  buff3 [1] = 0x08;
  buff3 [2] = 0x82;
  buff3 [3] = 0x00;
  buff3 [4] = 0xf0;
  buff3 [5] = 0x02;
  buff3 [6] = 0x00;
  buff3 [7] = 0x88;
  buff3 [8] = 0x00;
  buff3 [9] = 0xaa;
  buff3 [10] = 0x02;
  buff3 [11] = 0x01;
  buff3 [12] = 0x88;
  buff3 [13] = 0x00;
  buff3 [14] = 0xbb;

  ret = nl1send (path, mod, &arg1, arg2, arg3);
  printf ("Return value (random write) = %X\n", ret);

/* CLOSE */
}
```

ACCESS FUNCTION LIBRARY **SEQUENCE PROGRAM BATCH READ**

Specification

Function:	A-CPU Sequence Program Access
Application:	Batch Read
Function Name:	nl1receive
Processing Code:	0x01
Driver Function Number:	3H

Processing

Code 0x01 specifies batch read of the A7BDE-A3N-PT32S3 SCPU and A-CPU sequence program and timer/counter memory area.

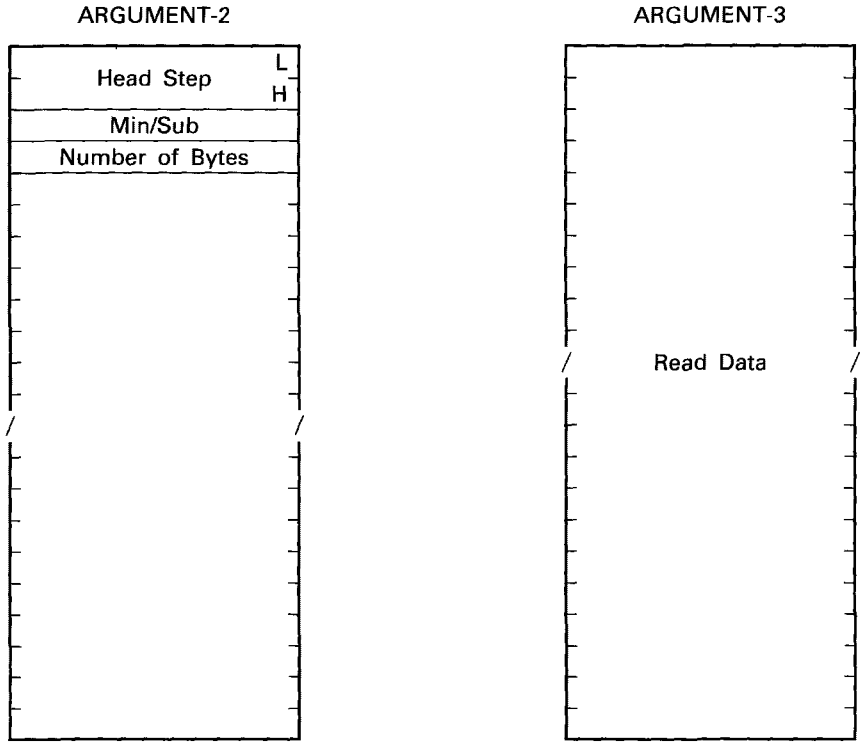
(see appendix for T/C step addresses)

Argument two specifies the head step number, main or sub program areas (A3 type CPU only), and the number of bytes to be read. (maximum of 128)

Note:	Main/Sub Setting	A0J2, A1, A2	(0)	(fixed)
		A3,A3H,A3M	(0)	(main)
			(1)	(sub)

Argument three receives the returned data. (1 step = 2 bytes)

Argument Formats



EXAMPLE **SEQUENCE PROGRAM BATCH READ**

```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;
NLARG1 arg1;

main ( )
{
  int chan, mod, i;
  short ret;
  unsigned char *arg2;
  unsigned char *arg3;
  char buff2 [512];
  char buff3 [512];

  arg2 = buff2;
  arg3 = buff3;

/* OPEN OF I/F BOARD */

/* A-CPU SEQUENCE PROGRAM READ */

/* This program reads from the A7BDE-A3N-PT32S3 SCPU, the sequence */
/* program step zero to step thirty two.    Note: One step */
/* requires two bytes of memory. */

  mod = 0;

  arg1. demand = 0x01;
  arg1. loop = 0x00;
  arg1. station = 0xff;

  buff2 [0] = 0x00;
  buff2 [1] = 0x00;
  buff2 [2] = 0x00;
  buff2 [3] = 0x40;

  ret = nl1receive (path, mod, &arg1, arg2, arg3);
  printf ("Return value (prog read) = %X\n", ret);

  for (i = 0 ; i < 0x40 ; i++)
  {
    printf ("buff3 [%2d] = %4x\n", i, buff3 [i]);
  }

/* CLOSE */

}
```

ACCESS FUNCTION LIBRARY	SEQUENCE PROGRAM BATCH WRITE

Specification

Function:	A-CPU Sequence Program Access
Application:	Batch Write
Function Name:	nl1send
Processing Code:	0x03
Driver Function Number:	4H

Processing

Code 0x03 specifies batch write to the A7BDE-A3N-PT32S3 SCPU and A-CPU sequence program and timer/counter memory area.

(see appendix for T/C stop addresses)

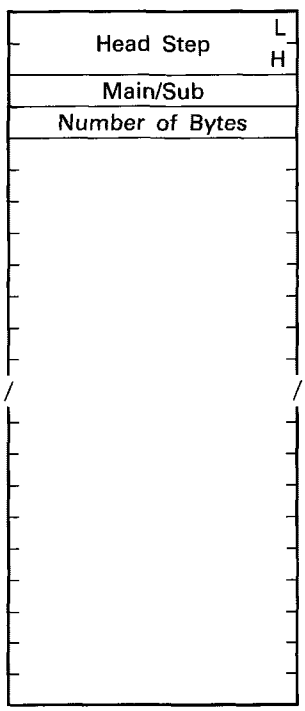
Argument two specifies the head step number, main or sub program areas (A3 type CPU only), and the number of bytes to be written. (maximum of 128)

Note:	Main/Sub Setting	A0J2, A1, A2	(0)	(fixed)
		A3, A3H, A3M	(0)	(main)
			(1)	(sub)

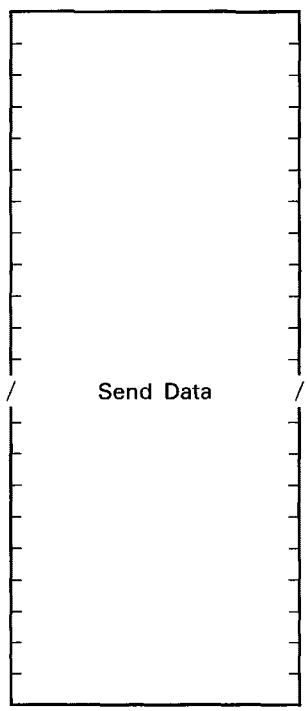
Argument three contains the sent data.

Argument Formats

ARGUMENT-2



ARGUMENT-3



EXAMPLE **SEQUENCE PROGRAM BATCH WRITE**

```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;
NLARG1 arg1;

main ( )
{
  int chan, mod, i;
  short ret;
  unsigned char *arg2;
  unsigned char *arg3;
  char buff2 [512];
  char buff3 [512];

  arg2 = buff2;
  arg3 = buff3;

/* OPEN OF I/F BOARD */

/* A-CPU SEQUENCE PROGRAM WRITE */

/* This program writes the instruction LD X020, to step */
/* zero of the host A7BDE-A3N-PT32S3 SCPU sequence program. */

  mod = 0;

  arg1. demand = 0x03;
  arg1. loop = 0;
  arg1. station = 0xff;

  buff2 [0] = 0;
  buff2 [1] = 0;
  buff2 [2] = 0;
  buff2 [3] = 2;

  buff3 [0] = 0x20;
  buff3 [1] = 0x40;

  ret = nl1send (path, mod, &arg1, arg2, arg3);
  printf ("Return value (prog write) = %X\n", ret);

/* CLOSE */

}
```

ACCESS FUNCTION LIBRARY	SCPU INTERRUPT PROGRAM START

Specification

Function:	SCPU Sequence Program Access
Application:	Interrupt Program Start
Function Name:	nl1send
Processing Code:	0x100
Driver Function Number:	4H

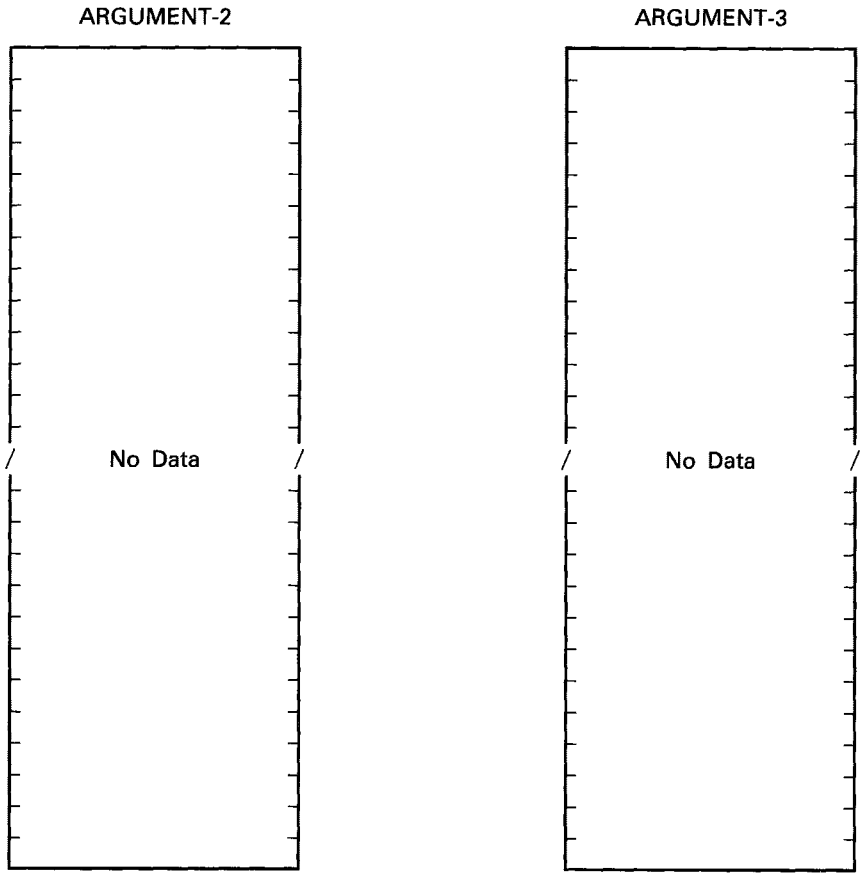
Processing

Code 0x100 enables the application program to initiate the processing of a host A7BDE-A3N-PT32S3 SCPU interrupt program. The SCPU interrupt sequence program, is indicated by pointer I16. The access station number must be set to 0xff.

Please note, if an interrupt program does not exist at I16, and this function is processed, the "CAN'T EXECUTE (1)" error occurs, and A7BDE-A3N-PT32S3 operation stops.

Arguments two and three require no set data.

Argument Formats



EXAMPLE **SCPU INTERRUPT PROGRAM START**

```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;
NLARG1 arg1;

main ( )
{
  int chan, mod;
  short ret;
  unsigned char *arg2;
  unsigned char *arg3;
  char buff2 [512];
  char buff3 [512];

  arg2 = buff2;
  arg3 = buff3;

/* OPEN OF I/F BOARD */

/* A7BDE-A3N-PT32S3 SCPU INTERRUPT PROGRAM START */

/* This program initiates the processing of an interrupt */
/* sequence program indicated by the pointer I16, in the */
/* host A7BDE-A3N-PT32S3 SCPU. */

  mod = 0;

  arg1. demand = 0x100;
  arg1. loop = 0x00;
  arg1. station = 0xff;

  ret = nl1send (path, mod, &arg1, arg2, arg3);
  printf ("Return value (interrupt) = %x\n", ret);

/* CLOSE */

}
```

ACCESS FUNCTION LIBRARY	A-CPU REMOTE CONTROL

Specification

Function:	A-CPU Control
Application:	Remote Run/Stop/Pause
Function Name:	nl1send
Processing Code:	0x18
Driver Function Number:	4H

Processing

Code 0x18 enables software setting of the host A7BDE-A3N-PT32S3 and A-CPU operating status. i.e Run/Stop/Pause.

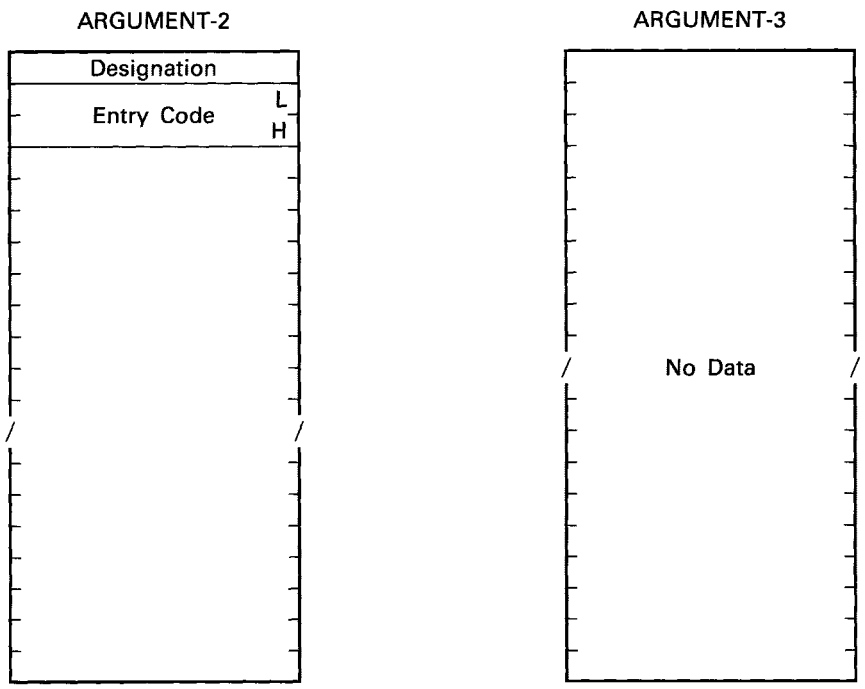
Argument two specifies the status designation and entry code.

Note: Designation = (0) Run
(1) Stop
(2) Pause
Entry Code = (4) (fixed)

Argument three requires no data.

Further details of remote Run/Stop/Pause control may be found in section 4.20 SCPU Operation. Please note that the operating status can not be software set to RUN if the Key Switch is set to STOP.

Argument Formats



EXAMPLE **A-CPU REMOTE CONTROL**

```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;
NLARG1 arg1;

main ( )
{
  int chan, mod, i;
  short ret;
  unsigned char *arg2;
  unsigned char *arg3;
  char buff2 [512];
  char buff3 [512];

  arg2 = buff2;
  arg3 = buff3;

/* OPEN OF I/F BOARD */

/* A-CPU REMOTE RUN/STOP/PAUSE */

/* This program sets the running conditions (Run/Stop/Pause) */
/* of the host A7BDE-A3N-PT32S3 SCPU. */

  mod = 0;

  arg1. demand = 0x18;
  arg1. loop = 0x00;
  arg1. station = 0xff;

  buff2 [1] = 4;
  buff2 [2] = 0;

  printf ("Select Run/Stop/Pause (0/1/2)\t");
  scanf ("%d", &buff2 [0]);

  ret = nl1send (path, mod, &arg1, arg2, arg3);
  printf ("Return value (run/stop) = %X\n", ret);

/* CLOSE */

}
```

ACCESS FUNCTION LIBRARY **A-CPU CHECK REQUEST**

Specification

Function:	A-CPU Control
Application:	A-CPU Check Request
Function Name:	nl1receive
Processing Code:	0x08
Driver Function Number:	3H

Processing

Code 0x08 enables reading of the CPU code of the accessed A-PLC, and the address of the system data table.

Argument two requires no set data.

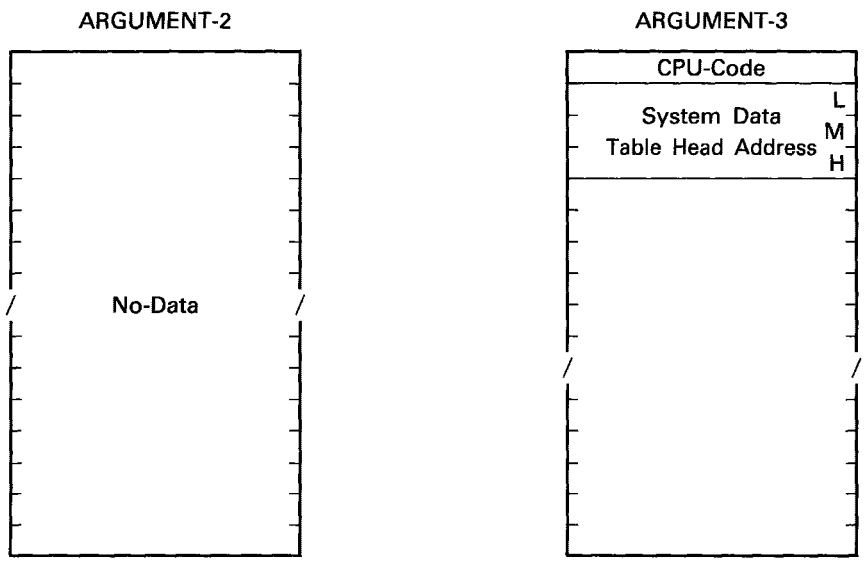
Argument three receives the CPU code and system data table address. The system data table contains the device specifications of the accessed A-PLC.

CPU Codes:

A7BDE-J71P21/R21	0x90	A3CPU	0xA4
A0J2CPU	0xA0	A3HCPU/A3MCPU	0xA4
A1CPU	0xA1	AJ72P25/R25	0xAB
A2CPU	0xA2	A0J2P25/R25	0xAB

After reading the address, the system data table may be read using the function A-CPU Memory Access - Batch Read. (nl1receive processing code 0x02). For details on the system data table configuration, see the appendix.

Argument Formats



EXAMPLE **A-CPU CHECK REQUEST**

```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;
NLARG1 arg1;

main ( )
{
  int chan, mod, i;
  short ret;
  unsigned char *arg2;
  unsigned char *arg3;
  char buff2 [512];
  char buff3 [512];

  arg2 = buff2;
  arg3 = buff3;

/* OPEN OF I/F BOARD */

/* A-CPU CHECK */

/* This program reads the type of CPU and system data table */
/* address of station one. */

  mod = 0;

  arg1. demand = 0x08;
  arg1. loop = 0x00;
  arg1. station = 0x01;

  ret = nl1receive (path, mod, &arg1, arg2, arg3);
  printf ("Return value (cpu check) = %X\n", ret);

  i = 0;
  while (i < 4)
  {
    printf ("buff3 [%2d] = %4X\n", i, buff3 [i]);
    i++;
  }

/* CLOSE */

}
```

ACCESS FUNCTION LIBRARY **A-CPU PARAMETER ANALYSIS**

Specification

Function:	A-CPU Control
Application:	Parameter Analysis Request
Function Name:	nl1send
Processing Code:	0x27
Driver Function Number:	4H

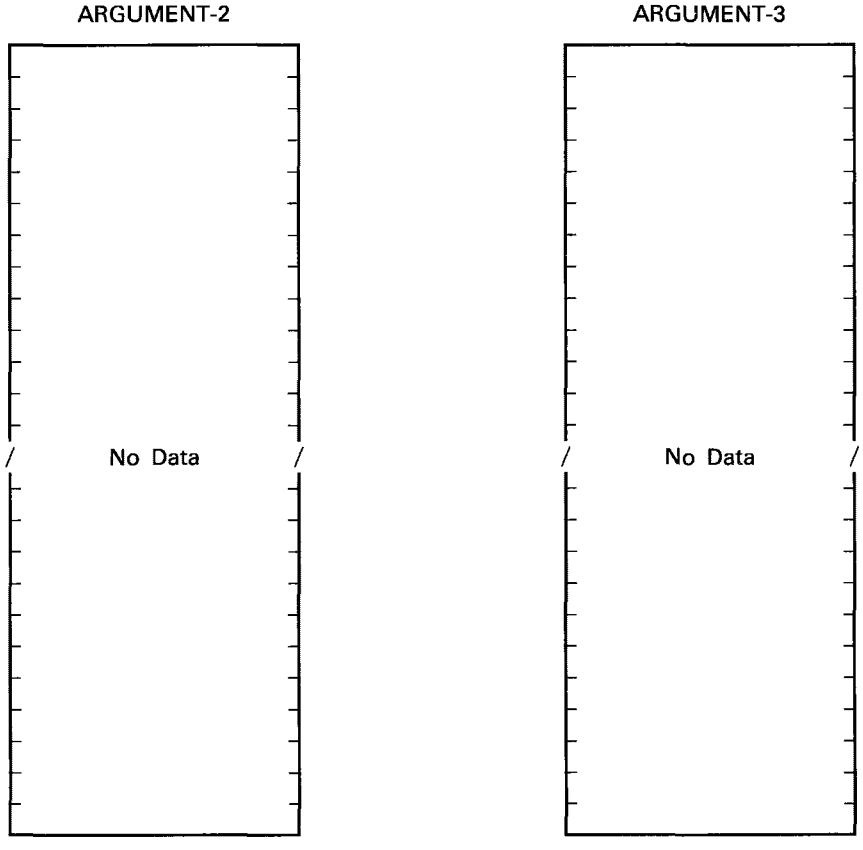
Processing

Code 0x27 specifies a parameter analysis request. This operation must be performed after any A-CPU parameter has been changed, to validate the new data.

New parameter settings may be written to the ACPU user memory area, however in normal operation, the parameter settings must be transferred to the ACPU work area. If parameter analysis is not performed, operation will continue with the previous parameter settings, still stored in the ACPU work area.

Argument two and three require no set data.

Argument Formats



EXAMPLE **A-CPU PARAMETER ANALYSIS**

```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;
NLARG1 arg1;

main ( )
{
  int chan, mod, i;
  short ret;
  unsigned char *arg2;
  unsigned char *arg3;
  char buff2 [512];
  char buff3 [512];

  arg2 = buff2;
  arg3 = buff3;

/* OPEN OF I/F BOARD */

/* PARAMETER ANALYSIS */

/* This program requests parameter analysis of the host */
/* A7BDE-A3N-PT32S3 SCPU. Parameter analysis must be performed */
/* after any changes have been made to the existing */
/* parameters. */

  mod = 0;

  arg1. demand = 0x27;
  arg1. loop = 0x00;
  arg1. station = 0xff;

  ret = nl1send (path, mod, &arg1, arg2, arg3);
  printf ("Return value (parameters analysis) = %X\n",ret);

/* CLOSE */

}
```

ACCESS FUNCTION LIBRARY	S. F. MODULE MEMORY BATCH READ

Specification

Function:	Special Function Module Access
Application:	2-Port Memory Batch Read
Function Name:	nl1receive
Processing Code:	0x10
Driver Function Number:	3H

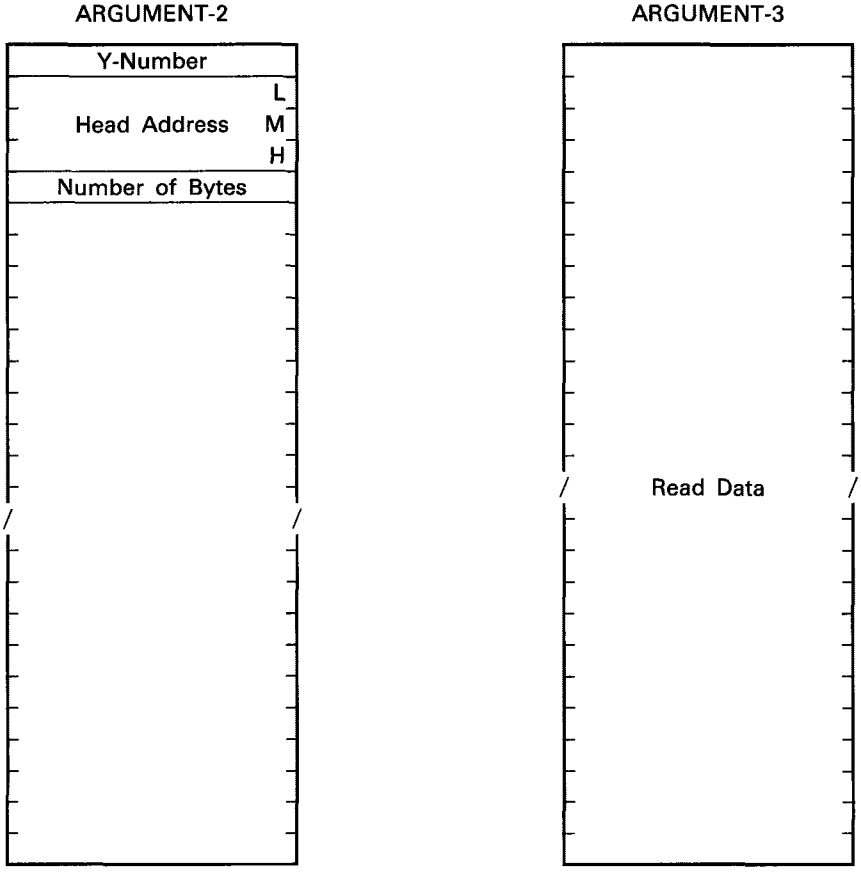
Processing

Code 0x10 specifies batch read of special function module 2-Port memory area. See the appendix for the various memory maps.

Argument two has three parameters. The two most significant digits of the special function module final Y-number. e.g set the Y-number to (07) if the module exists at location Y-number 60-7F. The two port memory head address, and the number of bytes to be read (maximum 128), must also be specified.

Argument three receives the read data.

Argument Formats



EXAMPLE **S. F. MODULE MEMORY BATCH READ**

```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;
NLARG1 arg1;

main ( )
{
  int chan, mod, i;
  short ret;
  unsigned char *arg2;
  unsigned char *arg3;
  char buff2 [512];
  char buff3 [512];

  arg2 = buff2;
  arg3 = buff3;

/* OPEN OF I/F BOARD */

/* SPECIAL MODULE ACCESS BATCH READ */

/* This program reads the buffer memory (channel one) of an */
/* A68AD located at slot head address 0x80 of station one. */

  mod = 0;

  arg1. demand = 0x10;
  arg1. loop = 0x00;
  arg1. station = 0x01;

  buff2 [0] = 0x09;
  buff2 [1] = 0x94;
  buff2 [2] = 0x00;
  buff2 [3] = 0x00;
  buff2 [4] = 2;

  ret = nl1receive (path, mod, &arg1, arg2, arg3);
  printf ("Return value (S.Mod. read) = %x\n", ret);

  printf ("buff3 [0] = %d\n",buff3 [0]);
  printf ("buff3 [1] = %d\n",buff3 [1]);

/* CLOSE */

}
```

ACCESS FUNCTION LIBRARY	S. F. MODULE MEMORY BATCH WRITE

Specification

Function:	Special Function Module Access
Application:	2-Port Memory Batch Write
Function Name:	nl1send
Processing Code:	0x12
Driver Function Number:	4H

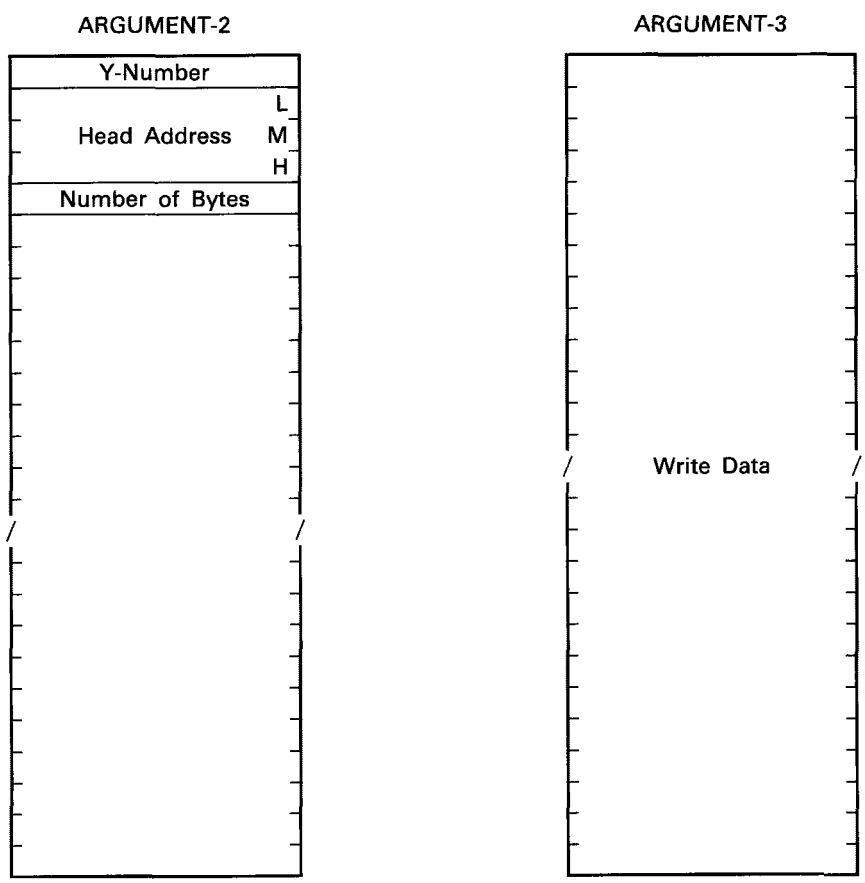
Processing

Code 0x12 specifies batch write of special function module 2-Port memory areas. See the appendix for the various memory maps.

Argument two has three parameters. The two most significant digits of the special function module final Y-number. e.g set the Y-number to (07) if the module exists at location Y-number 60-7F. The two port memory head address, and the number of bytes to be read (maximum 128), must also be specified.

Argument three contains the send data.

Argument Formats



EXAMPLE **S. F. MODULE MEMORY BATCH WRITE**

```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;
NLARG1 arg1;

main ( )
{
  int chan, mod, i;
  short ret;
  unsigned char *arg2;
  unsigned char *arg3;
  char buff2 [512];
  char buff3 [512];

  arg2 = buff2;
  arg3 = buff3;

/* OPEN OF I/F BOARD */

/* SPECIAL MODULE ACCESS BATCH WRITE */

/* This program writes to the buffer memory (channel one) */
/* of an A62DA located at station one. */

  mod = 0;

  arg1. demand = 0x12;
  arg1. loop = 0x00;
  arg1. station = 0x01;

  buff2 [0] = 0x0b;
  buff2 [1] = 0x10;
  buff2 [2] = 0x00;
  buff2 [3] = 0x00;
  buff2 [4] = 0x02;

  buff3 [0] = 0xa0;
  buff3 [1] = 0x00;

  ret = nl1send (path, mod, &arg1, arg2, arg3);
  printf ("Return value (S.Mod. write) = %x\n", ret);

/* CLOSE */

}
```

ACCESS FUNCTION LIBRARY	IFMEM BUFFER MEMORY BATCH READ

Specification

Function:	IFMEM Buffer Memory Access
Application:	Batch Read
Function Name:	nl1receive
Processing Code:	0x200
Driver Function Number:	3H

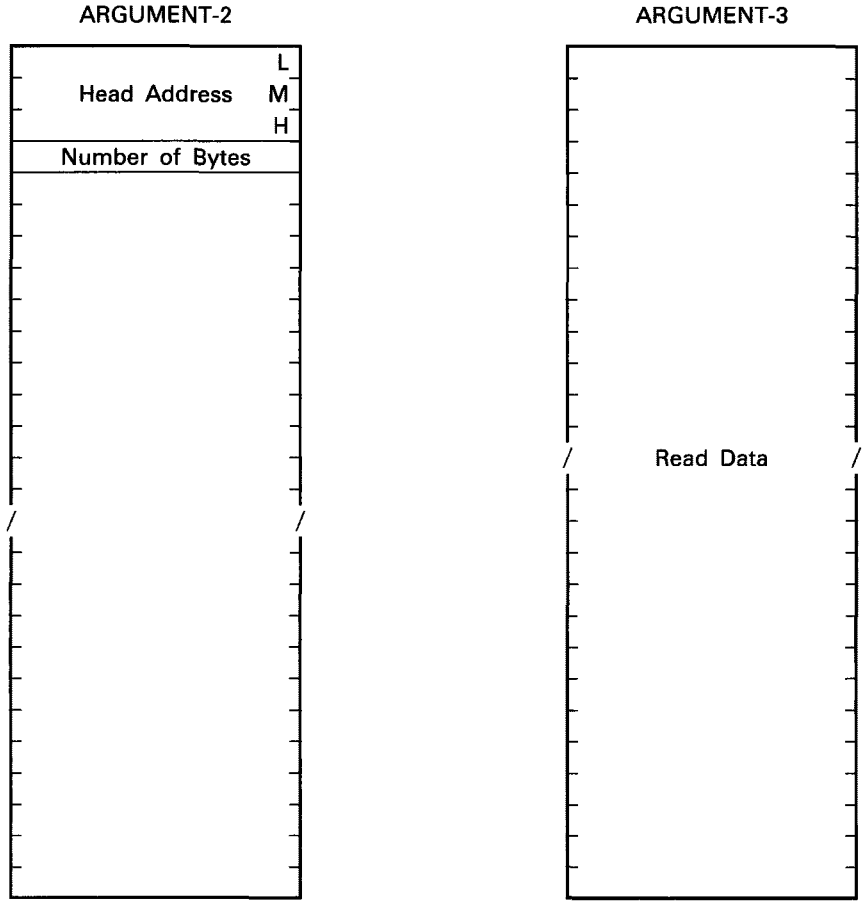
Processing

Processing code 0x200 enables batch-read of the IFMEM buffer memory. i.e. locations 0x800 to 0x1fff. Please see section 4.4 IFMEM Operation, for further information. Access station must be specified as 0xff.

Argument two specifies the buffer memory head address and number of bytes to be read. (maximum of 128 bytes)

Argument three receives the returned data. Format is dependent on the requested data.

Argument Formats



EXAMPLE **IFMEM BUFFER MEMORY BATCH READ**

```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;
NLARG1 arg1;

main ( )
{
  int chan, mod, i;
  short ret;
  unsigned char *arg2;
  unsigned char *arg3;
  char buff2 [512];
  char buff3 [512];

  arg2 = buff2;
  arg3 = buff3;

/* OPEN OF I/F BOARD */

/* A7BDE-A3N-PT32S3 IFMEM BUFFER MEMORY BATCH READ */

/* This program reads and displays the contents of the */
/* first 16 bytes of the host A7BDE-A3N-PT32S3 IFMEM buffer memory. */

  mod = 0;

  arg1. demand = 0x200;
  arg1. loop = 0x00;
  arg1. station = 0xff;

  buff2 [0] = 0x00;
  buff2 [1] = 0x08;
  buff2 [2] = 0x00;
  buff2 [3] = 0x10;

  ret = nl1receive (path, mod, &arg1, arg2, arg3);

  i = 0;
  while (i < 16)
  {
    printf ("buff3 [%3d] = %4X\n", i, buff3 [i]);
    i++;
  }

  printf ("Return value (mcpu read) = %x\n",ret);

/* CLOSE */

}
```

ACCESS FUNCTION LIBRARY	IFMEM BUFFER MEMORY BATCH WRITE

Specification

Function:	IFMEM Buffer Memory Access
Application:	Batch Write
Function Name:	nl1send
Processing Code:	0x201
Driver Function Number:	4H

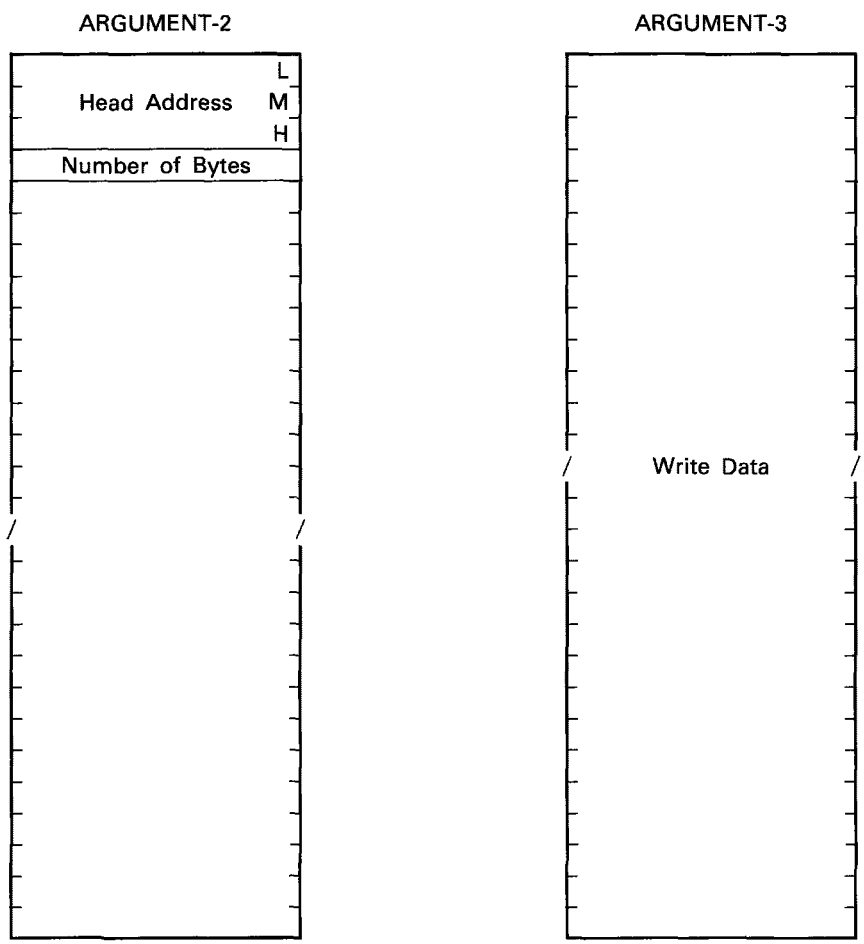
Processing

Processing code 0x201 enables batch-write of the IFMEM buffer memory. i.e. locations 0x800 to 0x1fff. Please see section 4.4 IFMEM Operation, for further information. Access station must be specified as 0xff.

Argument two specifies the buffer memory head address and number of bytes to be written. (maximum of 128 bytes)

Argument three contains the write data.

Argument Formats



EXAMPLE **IFMEM BUFFER MEMORY BATCH WRITE**

```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;
NLARG1 arg1;

main ( )
{
  int chan, mod, i;
  short ret;
  unsigned char *arg2;
  unsigned char *arg3;
  char buff2 [512];
  char buff3 [512];

  arg2 = buff2;
  arg3 = buff3;

/* OPEN OF I/F BOARD */

/* A7BDE-A3N-PT32S3 IFMEM BUFFER MEMORY BATCH WRITE */

/* This program writes the data 0xff to locations */
/* 0x810-0x820 of the host A7BDE-A3N-PT32S3 IFMEM buffer memory. */

  mod = 0;

  arg1. demand = 0x201;
  arg1. loop = 0x00;
  arg1. station = 0xff;

  buff2 [0] = 0x10;
  buff2 [1] = 0x08;
  buff2 [2] = 0x00;
  buff2 [3] = 0x10;

  i = 0;
  while (i < 16)
  {
    buff3 [i] = 0xff;
    i++;
  }

  ret = nl1send (path, mod, &arg1, arg2, arg3);
  printf ("Return value (mcpu write) = %X\n", ret);

/* CLOSE */

}
```

ACCESS FUNCTION LIBRARY	IFMEM BUFFER MEMORY RANDOM READ

Specification

Function:	IFMEM Buffer Memory Access
Application:	Random Read
Function Name:	nl1receive
Processing Code:	0x202
Driver Function Number:	3H

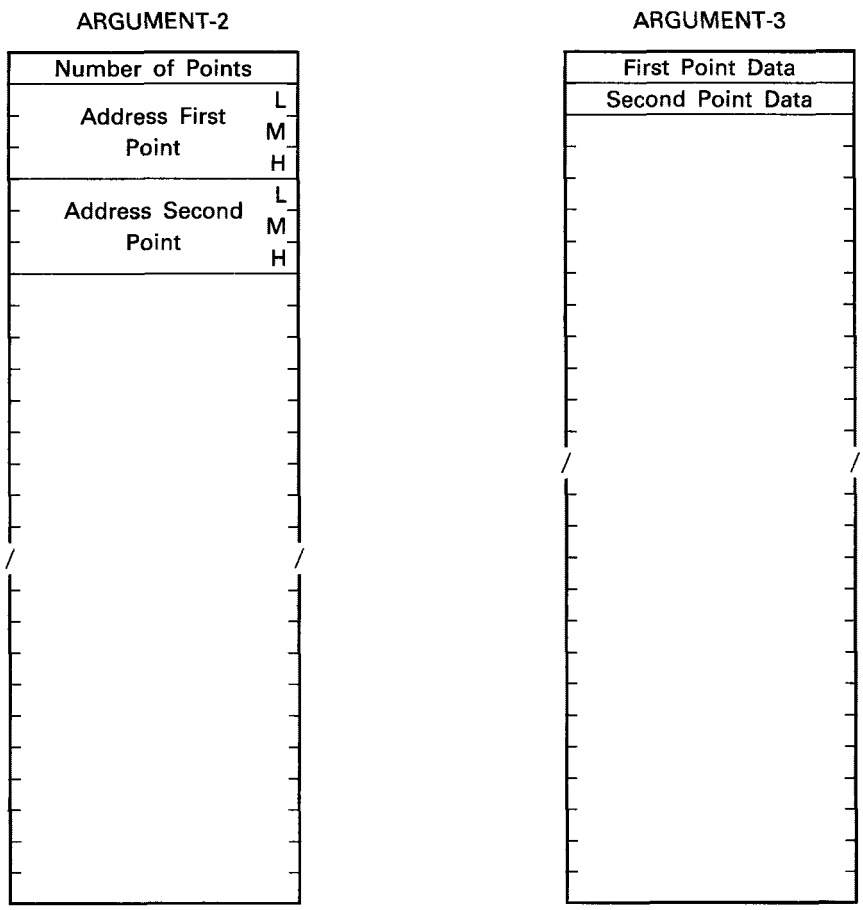
Processing

Processing code 0x202 enables random-read of the IFMEM buffer memory. i.e. locations 0x800 to 0x1fff. Please see section 4.4 IFMEM Operation, for further information. Access station must be specified as 0xff.

Argument two specifies the number of points (bytes) and their corresponding memory addresses. The maximum number of points that may be set in one argument is 40. Each Point is one byte.

Argument three receives the returned data.

Argument Formats



EXAMPLE **IFMEM BUFFER MEMORY RANDOM READ**

```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;
NLARG1 arg1;

main ( )
{
  int chan, mod, i;
  short ret;
  unsigned char *arg2;
  unsigned char *arg3;
  char buff2 [512];
  char buff3 [512];

  arg2 = buff2;
  arg3 = buff3;

/* OPEN OF I/F BOARD */

/* A7BDE-A3N-PT32S3 IFMEM BUFFER MEMORY RANDOM READ */

/* This program reads locations 0x800 and 0x810 of the */
/* A7BDE-A3N-PT32S3 IFMEM buffer memory. */

  mod = 0;

  arg1. demand = 0x202;
  arg1. loop = 0x00;
  arg1. station = 0xFF;

  buff2 [0] = 0x02;
  buff2 [1] = 0x00;
  buff2 [2] = 0x08;
  buff2 [3] = 0x00;
  buff2 [4] = 0x10;
  buff2 [5] = 0x08;
  buff2 [6] = 0x00;

  ret = nl1receive (path, mod, &arg1, arg2, arg3);
  printf ("Return value (mcpu rnd rd) = %X\n", ret);

  i = 0;
  while (i < 2)
  {
    printf ("buff3 [0] = %x\n", buff3 [0]);
    i++;
  }

/* CLOSE */

}
```

ACCESS FUNCTION LIBRARY	IFMEM BUFFER MEMORY RANDOM WRITE

Specification

Function:	IFMEM Buffer Memory Access
Application:	Random Write
Function Name:	nl1send
Processing Code:	0x203
Driver Function Number:	4H

Processing

Processing code 0x203 enables random write to the IFMEM buffer memory. i.e. locations 0x800 to 0x1fff. Please see section 4.4 IFMEM Operation, for further information. Access station must be specified as 0xff.

Argument two specifies the number of points (maximum of 24)

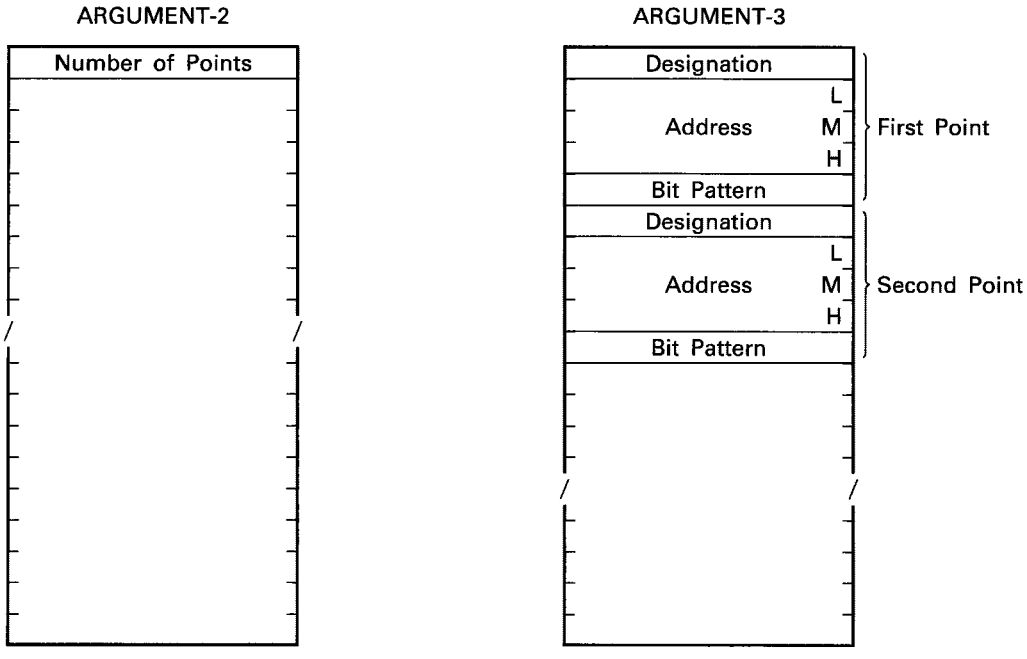
Argument three specifies the sent data. Each point is specified as follows: (1 point = 1 byte)

Designation:	(0) Bit Set	ORs contents and bit pattern data.
	(1) Bit Reset	ANDs contents and bit pattern data.
	(2) Byte Write	Writes bit pattern data to address.

Address: Memory address of specified device.

Bit Pattern: Data to be written to the device. (1 = "ON")
(0 = "OFF")

Argument Formats



EXAMPLE **IFMEM BUFFER MEMORY RANDOM WRITE**

```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;
NLARG1 arg1;

main ( )
{
  int chan, mod, i;
  short ret;
  unsigned char *arg2;
  unsigned char *arg3;
  char buff2 [512];
  char buff3 [512];

  arg2 = buff2;
  arg3 = buff3;

/* OPEN OF I/F BOARD */

/* A7BDE-A3N-PT32S3 IFMEM BUFFER MEMORY RANDOM WRITE */

/* This program writes the data 0xAA to location 0x810 */
/* of the host A7BDE-A3N-PT32S3 IFMEM buffer memory. */

  mod = 0;

  arg1. demand = 0x203;
  arg1. loop = 0x00;
  arg1. station = 0xFF;

  buff2 [0] = 0x01;

  buff3 [0] = 0x02;
  buff3 [1] = 0x10;
  buff3 [2] = 0x08;
  buff3 [3] = 0x00;
  buff3 [4] = 0xAA;

  ret = nl1send (path, mod, &arg1, arg2, arg3);
  printf ("Return value (mcpu rnd wrt) = %X\n", ret);

/* CLOSE */

}
```

ACCESS FUNCTION LIBRARY **IFMEM X-INPUT WRITE**

Specification

Function:	IFMEM Access
Application:	X-Input Write
Function Name:	nl1send
Processing Code:	0x204
Driver Function Number:	4H

Processing

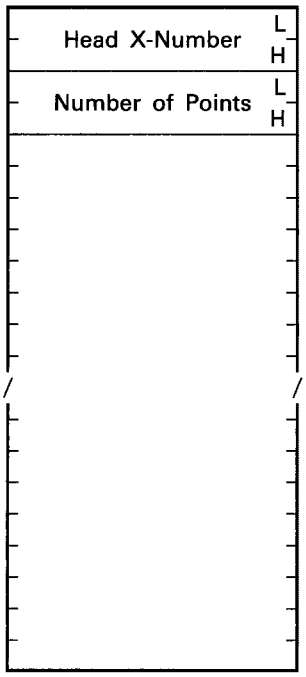
Processing code 0x204 enables the status of the IFMEM X-Inputs to be controlled by the application program. Please see section 4.4 IFMEM Operation, for further information, regarding the designation of each input.

Argument two specifies the head X-Input number, and the number of points to be written. Since devices are written in multiples of eight, the head X-input number may be set to either 0x00 or 0x08, and the number of points to either 0x08 or 0x10.

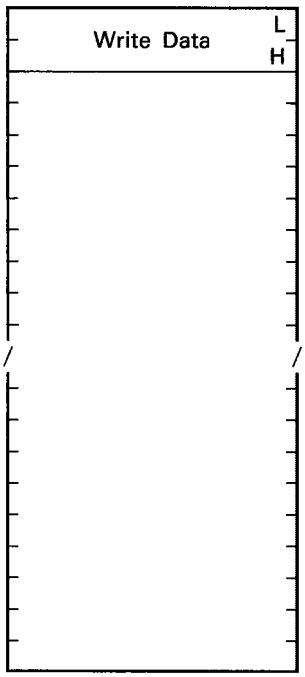
Argument three contains the sent data. The required status of the inputs is converted into a multiple of eight bit pattern. This is written to argument three, with the least significant bit of the data corresponding to the head device. A set value of one, will switch the device 'on', and a value of zero, 'off'.

Argument Formats

ARGUMENT-2



ARGUMENT-3



EXAMPLE **IFMEM X-INPUT WRITE**

```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;
NLARG1 arg1;

main ( )
{
  int chan, mod;
  short ret;
  unsigned char *arg2;
  unsigned char *arg3;
  char buff2 [512];
  char buff3 [512];

  arg2 = buff2;
  arg3 = buff3;

/* OPEN OF I/F BOARD */

/* A7BDE-A3N-PT32S3 IFMEM X-INPUT WRITE */

/* This program switches X0 and X4, of the A7BDE-A3N-PT32S3 IFMEM, */
/* 'on'. */

  mod = 0;

  arg1. demand = 0x204;
  arg1. loop = 0x00;
  arg1. station = 0xff;

  buff2 [0] = 0x00;
  buff2 [1] = 0x00;
  buff2 [2] = 0x08;
  buff2 [3] = 0x00;

  buff3 [0] = 0x11;

  ret = nl1send (path, mod, &arg1, arg2, arg3);
  printf ("Return value (mcpu X wr) = %X\n", ret);

/* CLOSE */

}
```

ACCESS FUNCTION LIBRARY **IFMEM Y-OUTPUT READ**

Specification

Function:	IFMEM Access
Application:	Y-Output Read
Function Name:	nl1receive
Processing Code:	0x205
Driver Function Number:	3H

Processing

Processing code 0x205 enables reading of the IFMEM Y-Outputs. (Y10 to Y17) Please see section 4.4 IFMEM Operation, for further information, regarding the designation of each output.

Argument two specifies the head Y-Output number, and the number of points to be read. These values are fixed at 0x10 and 0x08 respectively.

Argument three receives the returned data.

The status of the outputs is returned as group of eight bits. A value of one, indicates that the output is 'on', and a value of zero, 'off'. The least significant bit (bit 0) corresponds to the head Y-Output (Y10).

Argument Formats

ARGUMENT-2

Head Y-Number	L / H
Number of Points	L / H

ARGUMENT-3

Read Data

EXAMPLE **IFMEM Y-OUTPUT READ**

```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;
NLARG1 arg1;

main ( )
{
  int chan, mod;
  short ret;
  unsigned char *arg2;
  unsigned char *arg3;
  char buff2 [512];
  char buff3 [512];

  arg2 = buff2;
  arg3 = buff3;

/* OPEN OF I/F BOARD */

/* A7BDE-A3N-PT32S3 IFMEM Y-OUTPUT READ */

/* This program reads the status of outputs Y10 to Y17, of */
/* the A7BDE-A3N-PT32S3 IFMEM. */

  mod = 0;

  arg1. demand = 0x205;
  arg1. loop = 0x00;
  arg1. station = 0xff;

  buff2 [0] = 0x10;
  buff2 [1] = 0x00;
  buff2 [2] = 0x08;
  buff2 [3] = 0x00;

  ret = nl1receive (path, mod, &arg1, arg2, arg3);
  printf ("Return value (mcpu Y rd) = %X\n", ret);

  printf ("buff3 [0] = %x\n", buff3 [0]);

/* CLOSE */

}
```

ACCESS FUNCTION LIBRARY	H.S.MEMORY TRANSFER PARAMETERS

Specification

Function:	A7BDE-A3N-PT32S3 Access
Application:	H.S.Memory Transfer Parameters
Function Name:	nl1send
Processing Code:	0x803
Driver Function Number:	4H

Processing

Processing code 0x803 specifies the high speed device memory transfer parameters. i.e. the ranges of device statuses to be transferred from the SCPU to the high speed memory, and conversely from the high speed memory to the SCPU.

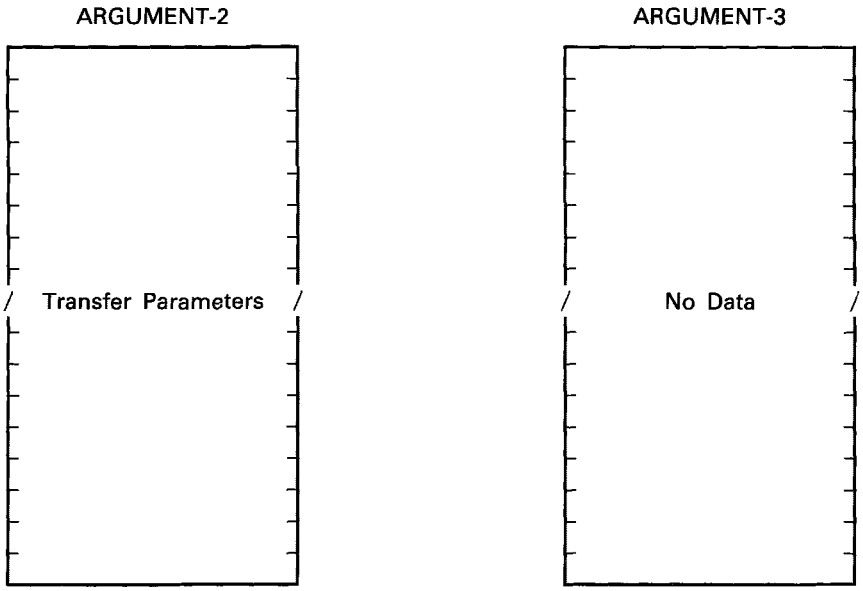
Argument two specifies the transfer parameters. The complete argument table is given in the appendix.

Please note: The ranges for timers and counters are set per point, but the coil status, contact status, and present value, for each device will be transferred.

The transfer parameters may only be set when the A7BDE-A3N-PT32S3 is in STOP mode. Operation status may be checked by the application program using processing code 0x701 Switch Status Read.

Argument three requires no set data.

Argument Formats



EXAMPLE **H.S.MEMORY TRANSFER PARAMETERS**

```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;
NLARG1 arg1;

main ( )
{
  int chan, mod, i;
  short ret;
  unsigned char *arg2;
  unsigned char *arg3;
  char buff2 [512];
  char buff3 [512];

  arg2 = buff2;
  arg3 = buff3;

/* OPEN OF I/F BOARD */

/* HIGH SPEED MEMORY DEVICE TRANSFER PARAMETERS */

/* This program specifies that data registers D0 to D19 are */
/* to be refreshed to and from the high speed device memory */
/* and the SCPU device memory. */

  mod = 0;

  arg1. demand = 0x803;
  arg1. loop = 0x00;
  arg1. station = 0xff;

  i = 0x00;
  while (i <= 0x68)
  {
    buff2 [i] = 0x00;
    i++;
  }

  buff2 [0x28] = 0x00;
  buff2 [0x29] = 0x00;
  buff2 [0x2a] = 0x28;
  buff2 [0x2b] = 0x00;
  buff2 [0x5c] = 0x00;
  buff2 [0x5d] = 0x00;
  buff2 [0x5e] = 0x28;
  buff2 [0x5f] = 0x00;

  ret = nl1send (path, mod, &arg1, arg2, arg3);
  printf ("Return value (HSM trsf prm) = %X\n", ret);

/* CLOSE */

}
```

ACCESS FUNCTION LIBRARY	HIGH SPEED MEMORY BATCH READ

Specification

Function:	A7BDE-A3N-PT32S3 Access
Application:	High Speed Memory Batch Read
Function Name:	nl1receive
Processing Code:	0x206
Driver Function Number:	3H

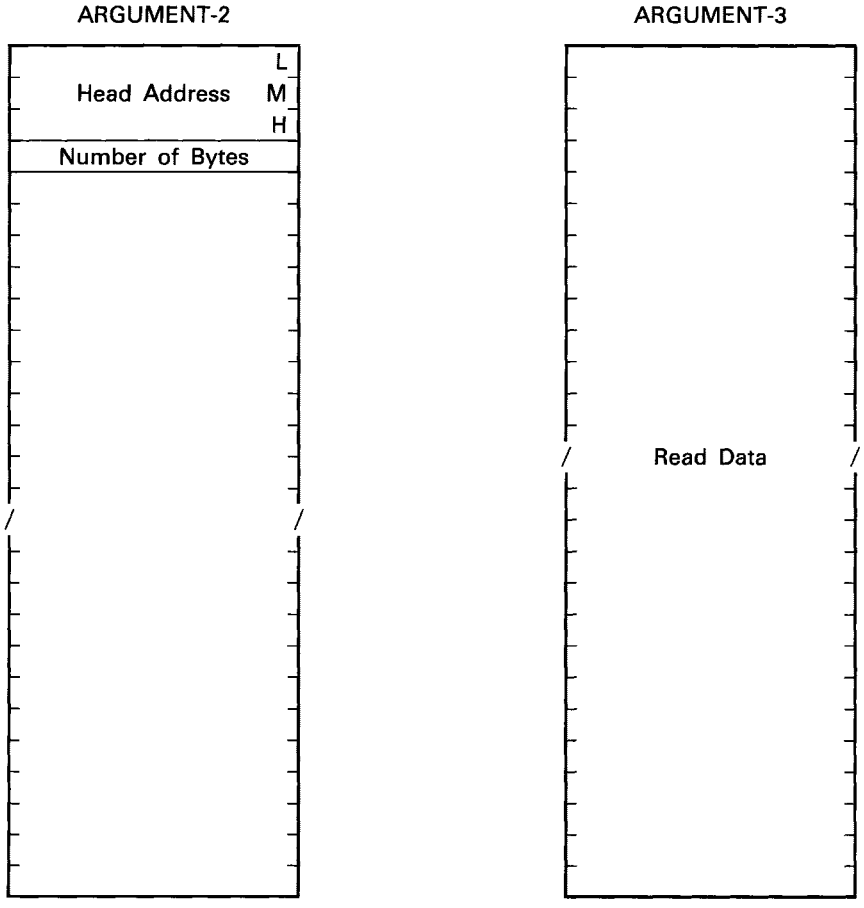
Processing

Processing code 0x206 enables batch-read of the A7BDE-A3N-PT32S3 high speed device memory. i.e. status of SCPU devices. Please see the appendix for head addresses and read data formats.

Argument two specifies the head address and number of bytes to be read. (maximum of 128 bytes)

Argument three receives the returned data. Format is dependent on the requested data.

Argument Formats



EXAMPLE **HIGH SPEED MEMORY BATCH READ**

```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;
NLARG1 arg1;

main ( )
{
  int chan, mod, i;
  short ret;
  unsigned char *arg2;
  unsigned char *arg3;
  char buff2 [512];
  char buff3 [512];

  arg2 = buff2;
  arg3 = buff3;

/* OPEN OF I/F BOARD */

/* A7BDE-A3N-PT32S3 HIGH SPEED MEMORY BATCH READ */

/* This program reads and displays the status of devices */
/* X00 to X07 from the host A7BDE-A3N-PT32S3 high speed memory. */

  mod = 0;

  arg1. demand = 0x206;
  arg1. loop = 0x00;
  arg1. station = 0xff;

  buff2 [0] = 0x00;
  buff2 [1] = 0x80;
  buff2 [2] = 0x00;
  buff2 [3] = 0x10;

  ret = nl1receive (path, mod, &arg1, arg2, arg3);
  printf ("Return value (HSM batch rd) = %X\n", ret);

  i = 0;
  while (i < 16)
  {
    printf ("buff3 [%3d] = %4X\n", i, buff3 [i]);
    i = i + 2;
  }

/* CLOSE */

}
```

ACCESS FUNCTION LIBRARY	HIGH SPEED MEMORY BATCH WRITE

Specification

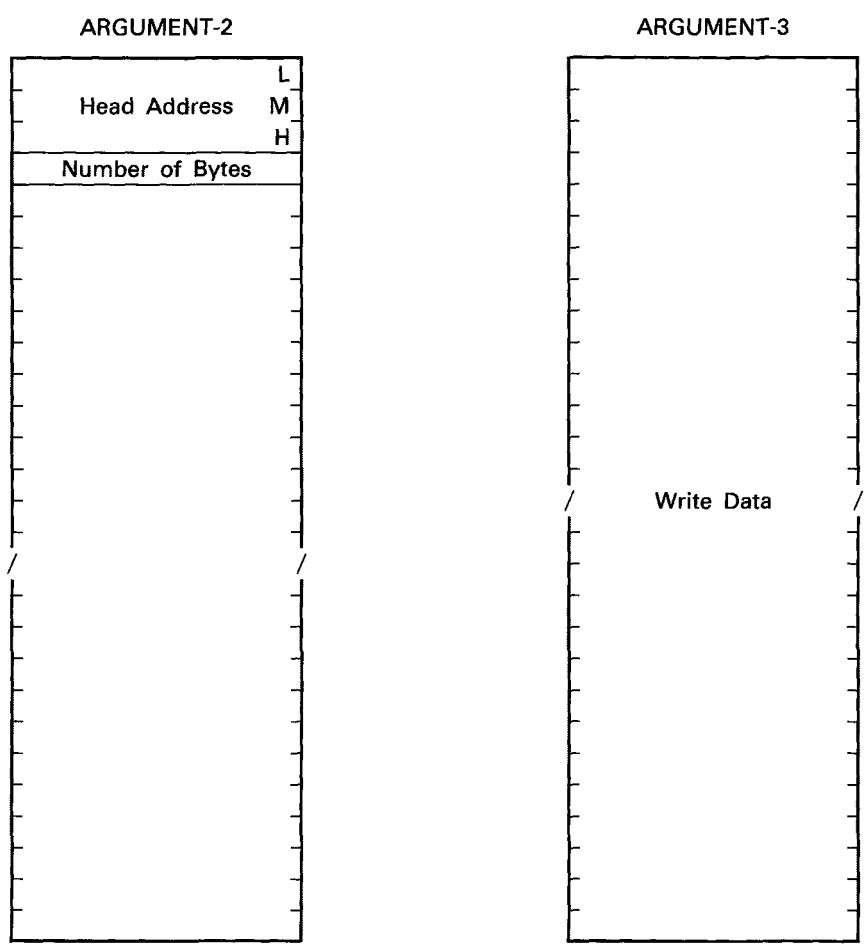
Function:	A7BDE-A3N-PT32S3 Access
Application:	High Speed Memory Batch Write
Function Name:	nl1send
Processing Code:	0x208
Driver Function Number:	4H

Processing

Processing code 0x208 enables batch-write of the A7BDE-A3N-PT32S3 high speed device memory. i.e. SCPU device status. Please see the appendix for head addresses and write data formats.
Argument two specifies the head address and number of bytes to be written. (maximum of 128 bytes)

Argument three contains the write data.

Argument Formats



EXAMPLE **HIGH SPEED MEMORY BATCH WRITE**

```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;
NLARG1 arg1;

main ( )
{
  int chan, mod, i;
  short ret;
  unsigned char *arg2;
  unsigned char *arg3;
  char buff2 [512];
  char buff3 [512];

  arg2 = buff2;
  arg3 = buff3;

/* OPEN OF I/F BOARD */

/* A7BDE-A3N-PT32S3 HIGH SPEED MEMORY BATCH WRITE */

/* This program writes the value 0xff to data registers */
/* D0-D7 of the host A7BDE-A3N-PT32S3 high speed memory. */

  mod = 0;

  arg1. demand = 0x208;
  arg1. loop = 0x00;
  arg1. station = 0xff;

  buff2 [0] = 0x00;
  buff2 [1] = 0x80;
  buff2 [2] = 0x00;
  buff2 [3] = 0x10;

  i = 0;
  while (i < 16)
  {
    buff3 [i] = 0xff;
    i = i + 2;
  }

  ret = nl1send (path, mod, &arg1, arg2, arg3);
  printf ("Return value (HSM batch wr) = %X\n", ret);

/* CLOSE */

}
```

ACCESS FUNCTION LIBRARY	HIGH SPEED MEMORY RANDOM READ

Specification

Function:	A7BDE-A3N-PT32S3 Access
Application:	High Speed memory Random Read
Function Name:	nl1receive
Processing Code:	0x207
Driver Function Number:	3H

Processing

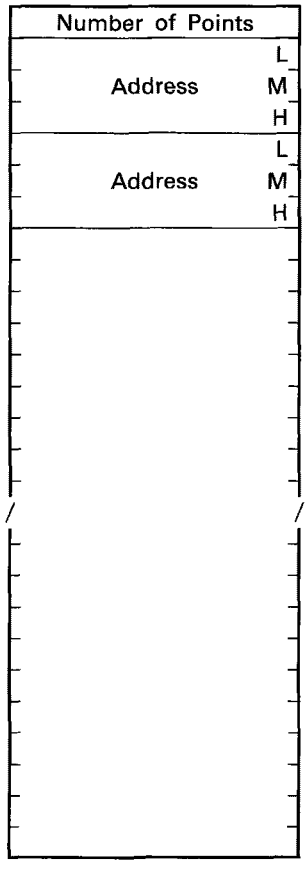
Processing code 0x207 enables random-read of the A7BDE-A3N-PT32S3 high speed device memory. i.e. SCPU device status. Please see the appendix for head addresses and read data format.

Argument two specifies the number of points and their corresponding memory addresses. The maximum number of points that may be set in one argument is 40.

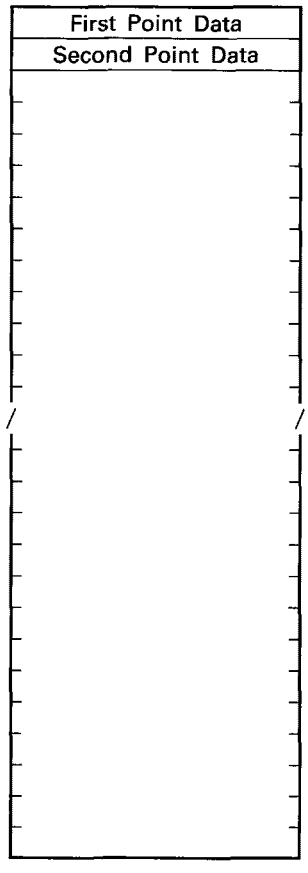
Argument three receives the returned data.

Argument Formats

ARGUMENT-2



ARGUMENT-3



EXAMPLE **HIGH SPEED MEMORY RANDOM READ**

```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;
NLARG1 arg1;

main ( )
{
  int chan, mod, i;
  short ret;
  unsigned char *arg2;
  unsigned char *arg3;
  char buff2 [512];
  char buff3 [512];

  arg2 = buff2;
  arg3 = buff3;

/* OPEN OF I/F BOARD */

/* A7BDE-A3N-PT32S3 HIGH SPEED MEMORY RANDOM READ */

/* This program reads and displays the present value of */
/* timer T0 from the host A7BDE-A3N-PT32S3 high speed memory. */

  mod = 0;

  arg1. demand = 0x207;
  arg1. loop = 0x00;
  arg1. station = 0xff;

  buff2 [0] = 0x02;
  buff2 [1] = 0x00;
  buff2 [2] = 0x98;
  buff2 [3] = 0x00;
  buff2 [4] = 0x01;
  buff2 [5] = 0x98;
  buff2 [6] = 0x00;

  ret = nl1receive (path, mod, &arg1, arg2, arg3);
  printf ("Return value (HSM rnd rd) = %X\n", ret);

  printf ("buff3 [0] = %x\n", buff3 [0]);
  printf ("buff3 [1] = %x\n", buff3 [1]);

/* CLOSE */

}
```

ACCESS FUNCTION LIBRARY	HIGH SPEED MEMORY RANDOM WRITE

Specification

Function:	A7BDE-A3N-PT32S3 Access
Application:	High Speed memory Random Write
Function Name:	nl1send
Processing Code:	0x209
Driver Function Number:	4H

Processing

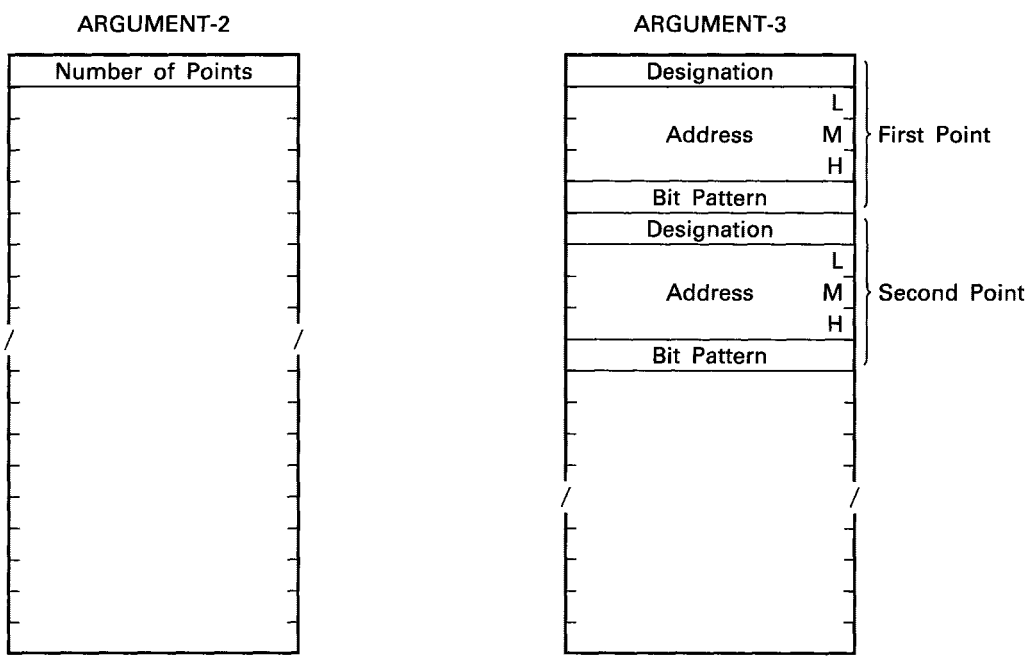
Processing code 0x209 enables random-write to the A7BDE-A3N-PT32S3 high speed device memory. i.e. SCPU device status. Please see the appendix for head addresses and write data format.

Argument two specifies the number of points (maximum of 24)

Argument three specifies the sent data. Each point is specified as follows:

Designation:	(0) Bit Set	ORs contents and bit pattern data.
	(1) Bit Reset	ANDs contents and bit pattern data.
	(2) Byte Write	Writes bit pattern data to address.
Address:	Memory address of specified device	
Bit Pattern:	Data to be written to the device	(1 = "ON") (0 = "OFF")

Argument Formats



EXAMPLE **HIGH SPEED MEMORY RANDOM WRITE**

```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;
NLARG1 arg1;

main ( )
{
  int chan, mod, i;
  short ret;
  unsigned char *arg2;
  unsigned char *arg3;
  char buff2 [512];
  char buff3 [512];

  arg2 = buff2;
  arg3 = buff3;

/* OPEN OF I/F BOARD */

/* A7BDE-A3N-PT32S3 HIGH SPEED MEMORY RANDOM WRITE */

/* This program writes to the value 0xf0f0 to data register */
/* D0 of the host A7BDE-A3N-PT32S3 high speed memory. */

  mod = 0;

  arg1. demand = 0x209;
  arg1. loop = 0x00;
  arg1. station = 0xff;

  buff2 [0] = 0x02;

  buff3 [0] = 0x02;
  buff3 [1] = 0x00;
  buff3 [2] = 0x88;
  buff3 [3] = 0x00;
  buff3 [4] = 0xf0;
  buff3 [5] = 0x02;
  buff3 [6] = 0x01;
  buff3 [7] = 0x88;
  buff3 [8] = 0x00;
  buff3 [9] = 0xf0;

  ret = nl1send (path, mod, &arg1, arg2, arg3);
  printf ("Return value (HSM rnd wr) = %X\n", ret);

/* CLOSE */

}
```

ACCESS FUNCTION LIBRARY — **A7BDE-A3N-PT32S3 LED STATUS READ**

Specification

Function:	A7BDE-A3N-PT32S3 Board Control
Application:	LED Status Read
Function Name:	nl1receive
Processing Code:	0x700
Driver Function Number:	3H

Processing

Code 0x700 enables reading of the host A7BDE-A3N-PT32S3 network LED status indicators, and self-diagnosis error messages. The access station number must be specified as 0xFF.

Argument two requires no set data.

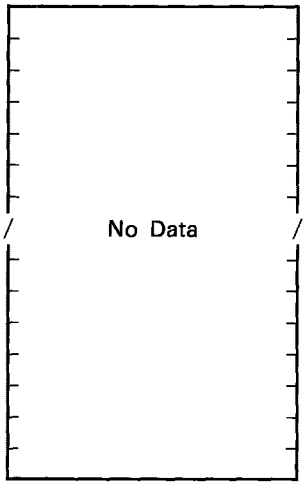
Argument three receives the returned data. The error message is contained in the first sixteen bytes, with the LED statuses, transferred as bit values in the proceeding bytes. Please see section SCPU Self Diagnosis for the various error messages. The LED statuses and their corresponding bits are as follows.

MINI LINK STATUS *1 (byte 16)				LINK STATUS *2 (byte 18)			
BIT	STATUS	BIT	STATUS	BIT	STATUS	BIT	STATUS
0	RUN	4	ALWAYS 1	0	CRC	4	DATA
1	RD	5	ALWAYS 1	1	OVER	5	UNDER
2	LOOP	6	ALWAYS 1	2	AB.IF	6	F.LOOP
3	REM	7	ALWAYS 1	3	TIME	7	R.LOOP

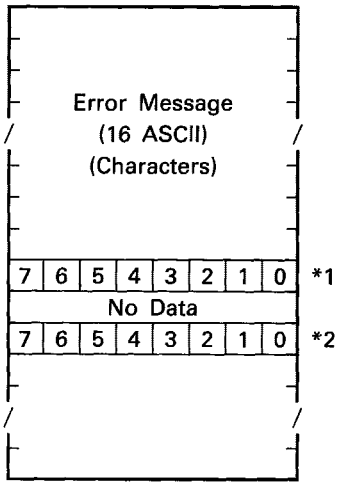
Note: (0) = 'On' (1) = 'Off'

Argument Formats

ARGUMENT-2



ARGUMENT-3



EXAMPLE **A7BDE-A3N-PT32S3 LED STATUS READ**

```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;
NLARG1 arg1;

main ( )
{
  int chan, mod;
  short ret;
  unsigned char *arg2;
  unsigned char *arg3;
  char buff2 [512];
  char buff3 [512];

  arg2 = buff2;
  arg3 = buff3;

/* OPEN OF I/F BOARD */

/* A7BDE-A3N-PT32S3 LED/ERROR STATUS READ */

/* This program reads and displays self-diagnosis error */
/* messages, and the status of the networks MELSECNET and */
/* MELSECNET/MINI error LEDs. */

  mod = 0;

  arg1. demand = 0x700;
  arg1. loop = 0x00;
  arg1. station = 0xff;

  ret = nl1receive (path, mod, &arg1, arg2, arg3);
  printf ("Return value (LED read) = %x\n", ret);

  for (i = 0x00; i  <= 0x12; i++)
  {
    if ( i < 0x10 )
    {
      printf ("buff3 [%2x] = %2c\n", i, buff3 [i]);
    }
    else
    {
      printf ("buff3 [%2x] = %2x\n", i, buff3 [i]);
    }
  }

/* CLOSE */

}
```

ACCESS FUNCTION LIBRARY **A7BDE-A3N-PT32S3 SWITCH STATUS READ**

Specification

Function:	A7BDE-A3N-PT32S3 Board Control
Application:	Switch Status Read
Function Name:	nl1receive
Processing Code:	0x701
Driver Function Number:	3H

Processing

Processing code 0x701 enables reading of the host A7BDE-A3N-PT32S3 control switch position (switch No. 0.), memory size, ROM/RAM, and protected RAM memory ranges (switch No. 1.). Please note, the access station number must be set at 0xff.

Argument two specifies the switch number.

Argument three receives the returned data. (switch 0/1 statuses)

Switch 0
Byte 1
Switch Status

Value	SWITCH STATUS
0	RUN
1	STOP
2	PAUSE
3	STEP RUN

Note: When the switch number is "0", the read status is the setting of the STOP/RUN switch, not the CPU's operating status.

Switch 1
Byte 1
Memory Protected Ranges

Bit position	Range	Value
0	20000 to 23FFF	0: Not Protected 1: Protected
1	24000 to 27FFF	
2	28000 to 2CFFF	
3	2C000 to 2FFFF	

Switch 1
Byte 2
ROM/RAM

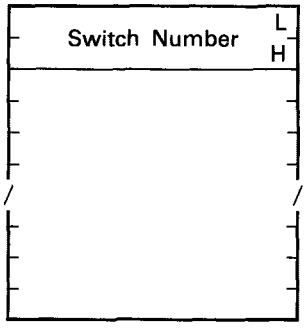
Bit Position	Value
0	0: ROM setting 1: RAM setting

Memory Size

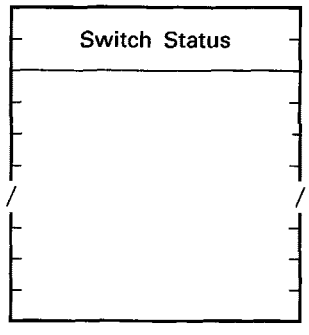
1, 2, 3	0x05 (MCA-8)

Argument Formats

ARGUMENT-2



ARGUMENT-3



EXAMPLE **A7BDE-A3N-PT32S3 SWITCH STATUS READ**

```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;
NLARG1 arg1;

main ( )
{
  int chan, mod;
  short ret;
  unsigned char *arg2;
  unsigned char *arg3;
  char buff2 [512];
  char buff3 [512];

  arg2 = buff2;
  arg3 = buff3;

/* OPEN OF I/F BOARD */

/* A7BDE-A3N-PT32S3 SWITCH STATUS READ */

/* This program reads and displays, the status of the */
/* selected switch. (switch 0 or 1) */

  mod = 0;

  arg1. demand = 0x701;
  arg1. loop = 0x00;
  arg1. station = 0xFF;

  printf ("Select Switch Number (0/1)\t");
  scanf ("%x", &buff2 [0]);

  buff2 [1] = 0x00;

  ret = nl1receive (path, mod, &arg1, arg2, arg3);
  printf ("Return value (sw. stat. rd. ) = %x\n", ret);

  printf ("buff3 [0] = %x\n", i, buff3 [0]);
  printf ("buff3 [1] = %x\n", i, buff3 [1]);

/* CLOSE */

}
```

ACCESS FUNCTION LIBRARY	A7BDE-A3N-PT32S3 VERSION READ

Specification

Function:	A7BDE-A3N-PT32S3 Board Access
Application:	Version Read
Function Name:	nl1receive
Processing Code:	0x702
Driver Function Number:	3H

Processing

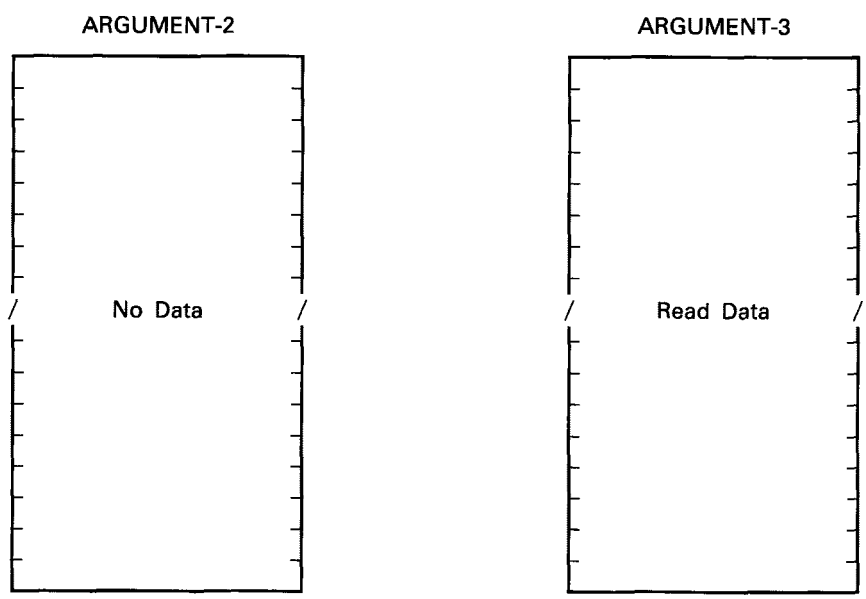
Code 0x702 specifies version read of the host A7BDE-A3N-PT32S3 option card. The access station number must be specified as 0xFF.

Argument two requires no set data.

Argument three receives the board version memory table of sixty four bytes. The table contents is as follows:

0-1H	Pass Word fixed at "SG" (ASCII Code)
2-3H	Check Sum of bytes 4 to 1FH (Hex)
4-5H	Software Version (ASCII Code)
6-BH	ROM Date Two bytes each - Year - Month - Day (ASCII)
C-FH	Reserved area (set to 0x00)
10-1FH	Software Type e.g. A3NCPU (ASCII Code)
20-2FH	Hardware Type e.g. A7BD-A3N-PT32S3 (ASCII Code)
30-31H	2-Port Memory Size e.g. 4000H i.e. 8K Bytes (Hex)
32-33H	2-Port Attribute Fixed at 0001H (Hex)
34-35H	Usable Offset (Hex)
36-3FH	Reserved Area

Argument Formats



EXAMPLE **A7BDE-A3N-PT32S3 VERSION READ**

```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;
NLARG1 arg1;

main ( )
{
  int chan, mod, i;
  short ret;
  unsigned char *arg2;
  unsigned char *arg3;
  char buff2 [512];
  char buff3 [512];

  arg2 = buff2;
  arg3 = buff3;

/* OPEN OF I/F BOARD */

/* A7BDE-A3N-PT32S3 VERSION READ */

/* This program reads the current version of the host */
/* A7BDE-A3N-PT32S3. */

  mod = 0;

  arg1. demand = 0x702;
  arg1. loop = 0x00;
  arg1. station = 0xff;

  ret = nl1receive (path, mod, &arg1, arg2, arg3);
  printf ("Return value (version read) = %x\n", ret);

  for (i = 0x00; i <= 0x35; i++)
  {
    if ((i > 0x01 && i < 0x06)        (i > 0x2f))
    {
      printf ("buff3 [%2x] = %2x\n", i, buff3 [i]);
    }
    else
    {
      printf ("buff3 [%2x] = %2c\n", i, buff3 [i]);
    }
  }

/* CLOSE */

}
```

ACCESS FUNCTION LIBRARY	A7BDE-A3N-PT32S3 BOARD RESET

Specification

Function:	A7BDE-A3N-PT32S3 Board Control
Application:	Board Reset
Function Name:	nl1send
Processing Code:	0x800
Driver Function Number:	4H

Processing

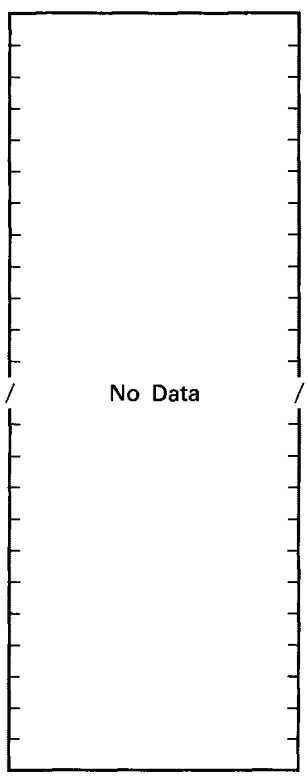
Processing code 0x800 specifies general reset of the host A7BDE-A3N-PT32S3 option card.

At reset All SCPU data is cleared, and devices reset.
SCPU operation is re-initiated.
All self-diagnosed errors are cleared.
All IFMEM data is cleared.
All high speed memory data is cleared, including the transfer parameters.
MELSECNET (if master) and MELSECNET/MINI are reset.

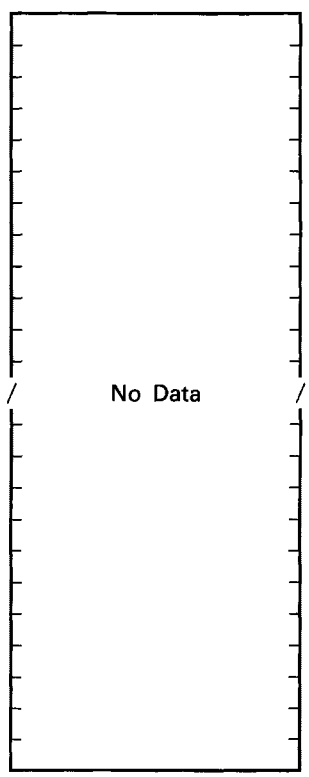
Arguments two and three require no set data.

Argument Formats

ARGUMENT-2



ARGUMENT-3



EXAMPLE **A7BDE-A3N-PT32S3 GENERAL RESET**

```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;
NLARG1 arg1;

main ( )
{
  int chan, mod, i;
  short ret;
  unsigned char *arg2;
  unsigned char *arg3;
  char buff2 [128];
  char buff3 [2048];

  arg2 = buff2;
  arg3 = buff3;

/* OPEN OF I/F BOARD */

/* A7BDE-A3N-PT32S3 GENERAL RESET */

/* This program performs general reset of the host */
/* A7BDE-A3N-PT32S3. */

  mod = 0;

  arg1. demand = 0x800;
  arg1. loop = 0x00;
  arg1. station = 0xff;

  ret = nl1send (path, mod, &arg1, arg2, arg3);
  printf ("Return value (general reset) = %x\n", ret);

/* CLOSE */

}
```

ACCESS FUNCTION LIBRARY	A7BDE-A3N-PT32S3 INDICATOR RESET

Specification

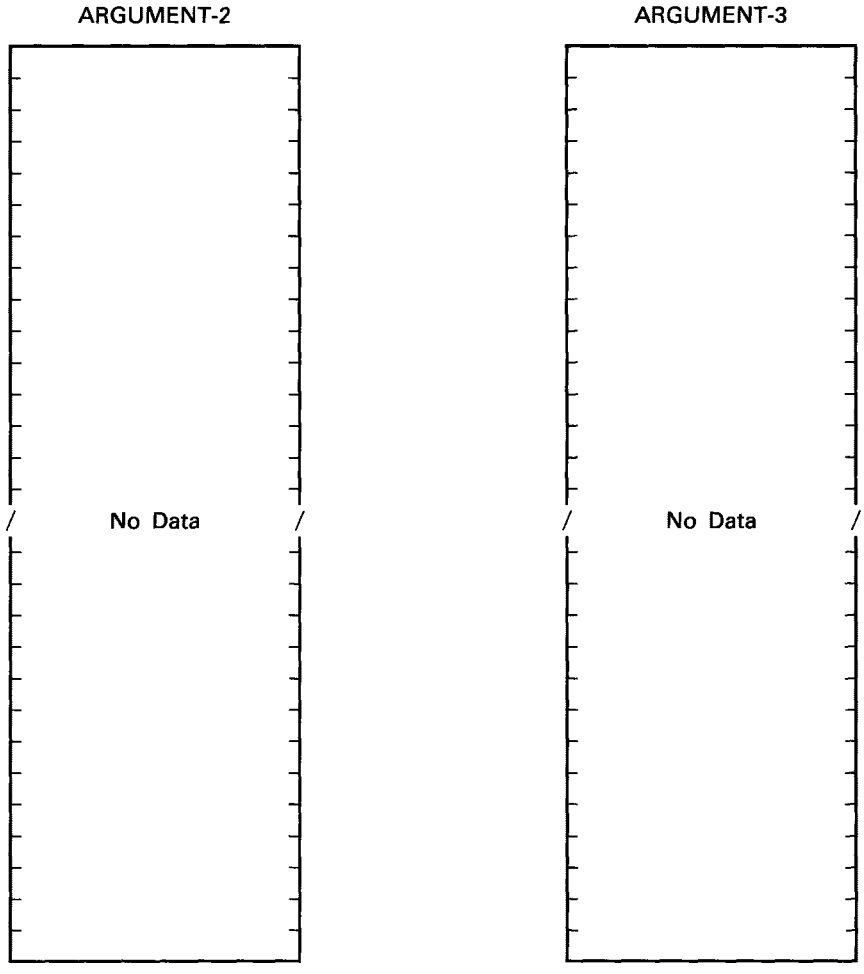
Function:	A7BDE-A3N-PT32S3 Board Control
Application:	Indicator Reset
Function Name:	nl1send
Processing Code:	0x80A
Driver Function Number:	4H

Processing

Processing code 0x80A specifies indicator reset of the A7BDE-A3N-PT32S3 option cards. i.e. all self-diagnosed errors and error messages will be cleared. If the original cause of the error has not been rectified, the same error will be indicated on the next program scan of the SCPU.

Arguments two and three require no set data.

Argument Formats



EXAMPLE **A7BDE-A3N-PT32S3 ERROR INDICATOR RESET**

```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;
NLARG1 arg1;

main ( )
{
  int chan, mod;
  short ret;
  unsigned char *arg2;
  unsigned char *arg3;
  char buff2 [512];
  char buff3 [512];

  arg2 = buff2;
  arg3 = buff3;

/* OPEN OF I/F BOARD */

/* A7BDE-A3N-PT32S3 INDICATOR RESET */

/* This program resets the self-diagnose error messages and */
/* network status LEDs. */

  mod = 0;

  arg1. demand = 0x80A;
  arg1. loop = 0x00;
  arg1. station = 0xff;

  ret = nl1send (path, mod, &arg1, arg2, arg3);
  printf ("Return value (ind. reset) = %x\n", ret);

/* CLOSE */

}
```

ACCESS FUNCTION LIBRARY	A7BDE-A3N-PT32S3 FREE DATA SEND

Specification

Function:	General Data
Application:	Data Free Send
Function Name:	nl1send
Processing Code:	0x40
Driver Function Number:	4H

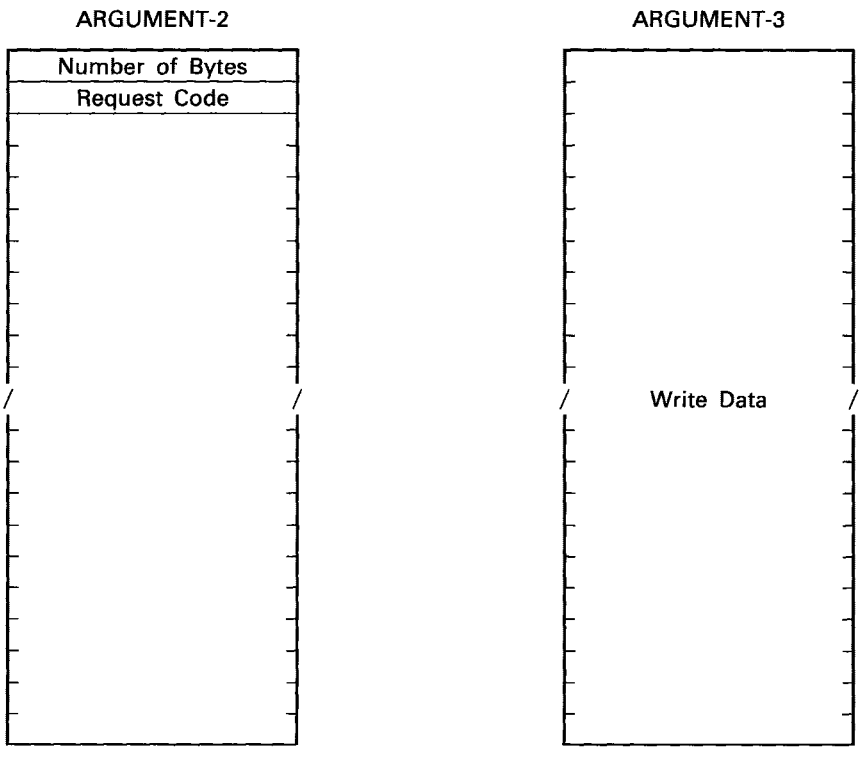
Processing

Code 0x40 enables Data Free Send between a master/local A-CPU-A7BDE-A3N-PT32S3 station, and a local/master A7BDE-J71P21/R21-PC station. (i.e. master to local, or local to master. not local to local) The free data is sent to a buffer memory location on the receiving A7BDE-J71P21/R21-PC station. The buffer memory can hold ten 130 byte messages, which may be accessed in a first in first out basis. Once the buffer memory is full, no new messages may be sent until the received data has been read.

Argument two specifies the number of bytes to be sent (128 max), and a request code. The request code labels the sent data as free data and must be specified within the range 0x80 and 0xFE.

Argument three contains the send data.

Argument Formats



EXAMPLE **A7BDE-A3N-PT32S3 FREE DATA SEND**

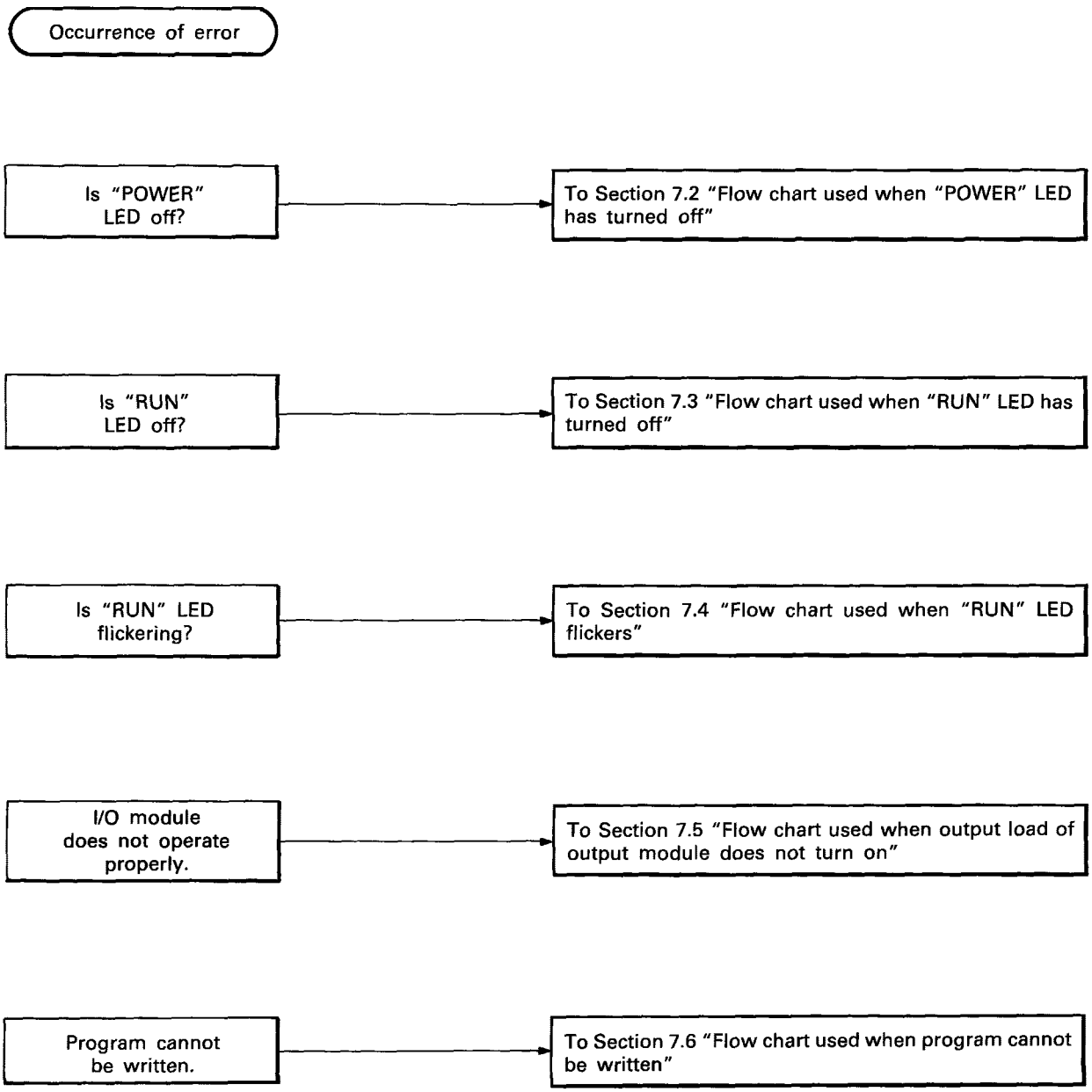
```
#include <stdio.h>
#include <nyuserc.h>

PATH *path;
NLARG1 arg1;

main ( )
{
  int chan, mod;
  short ret;
  unsigned char *arg2;
  unsigned char *arg3;
  char buff2 [512];
  char buff3 [512];

  arg2 = buff2;
  arg3 = buff3;

/* OPEN OF I/F BOARD */

/* A7BDE-A3N-PT32S3 FREE DATA SEND */

/* This program sends free data to local A7BDE-J71P21/R21 */
/* station one. */

  mod = 0;

  arg1. demand = 0x40;
  arg1. loop = 0x00;
  arg1. station = 0x01;

  buff2 [0] = 0x09;
  buff2 [1] = 0x80;

  buff3 [0] = 'A';
  buff3 [1] = '7';
  buff3 [2] = 'B';
  buff3 [3] = 'D';
  buff3 [4] = 'E';
  buff3 [5] = '-';
  buff3 [6] = 'A';
  buff3 [7] = '3';
  buff3 [8] = 'N';

  ret = nl1send (path, mod, &arg1, arg2, arg3);
  printf ("Return value (free data send) = %x\n", ret);

/* CLOSE */

}
```

7. TROUBLESHOOTING

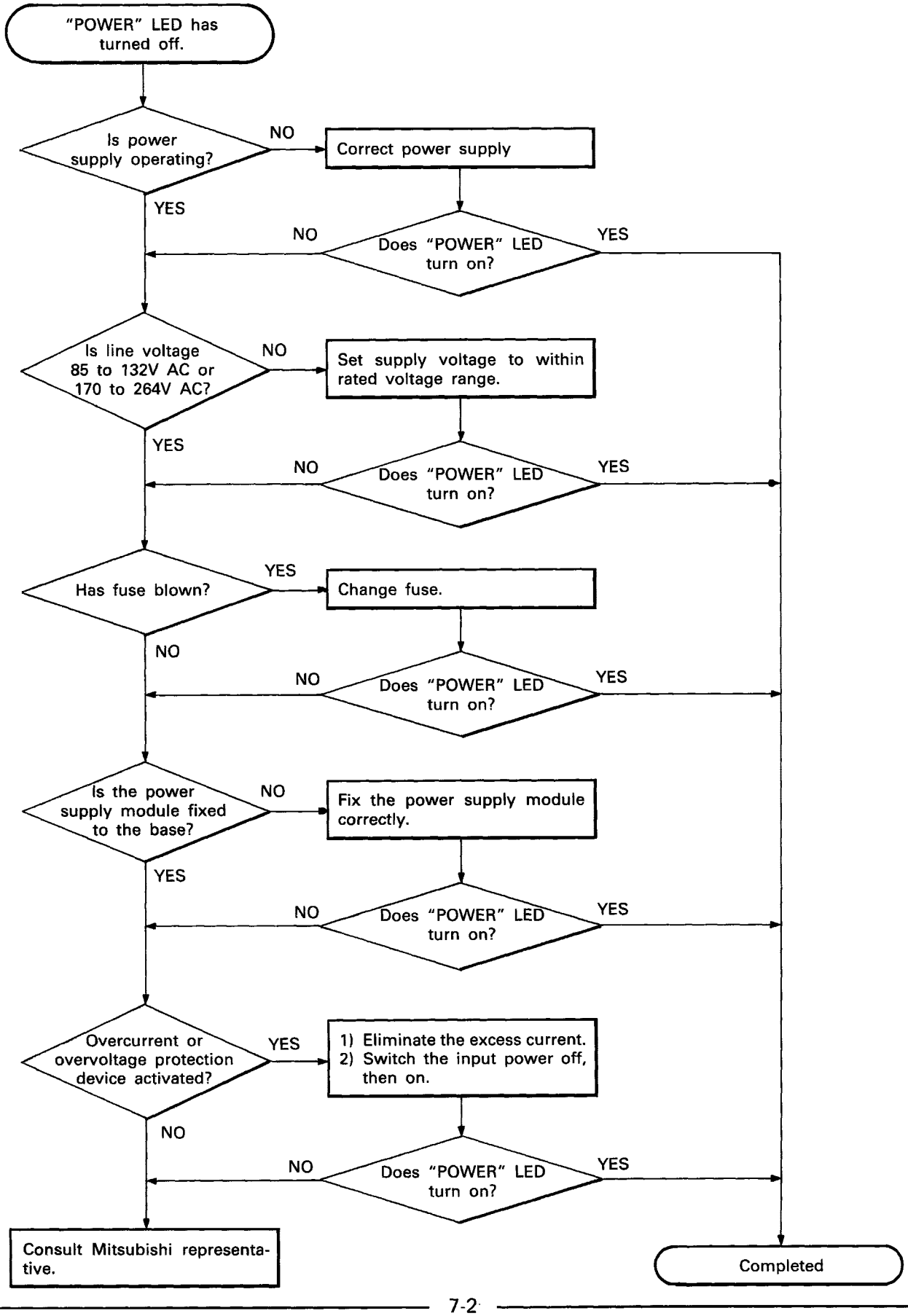
This section explains the procedure for determining the cause of problems and the errors and corrective actions for error codes.

7.1 Troubleshooting Flow Charts

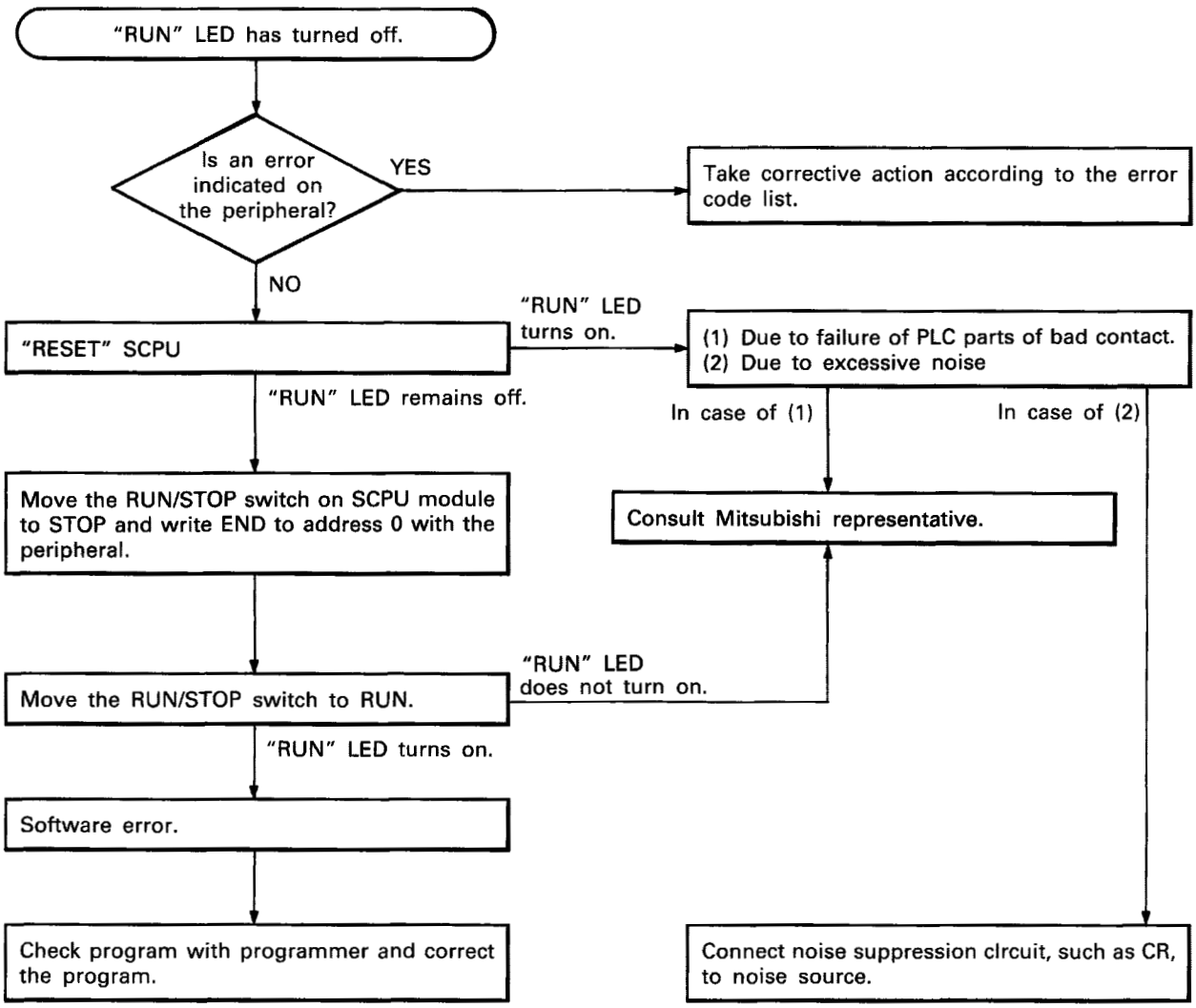
Details for fault finding may be found as follows.



7.2 Flow Chart "POWER" LED Off

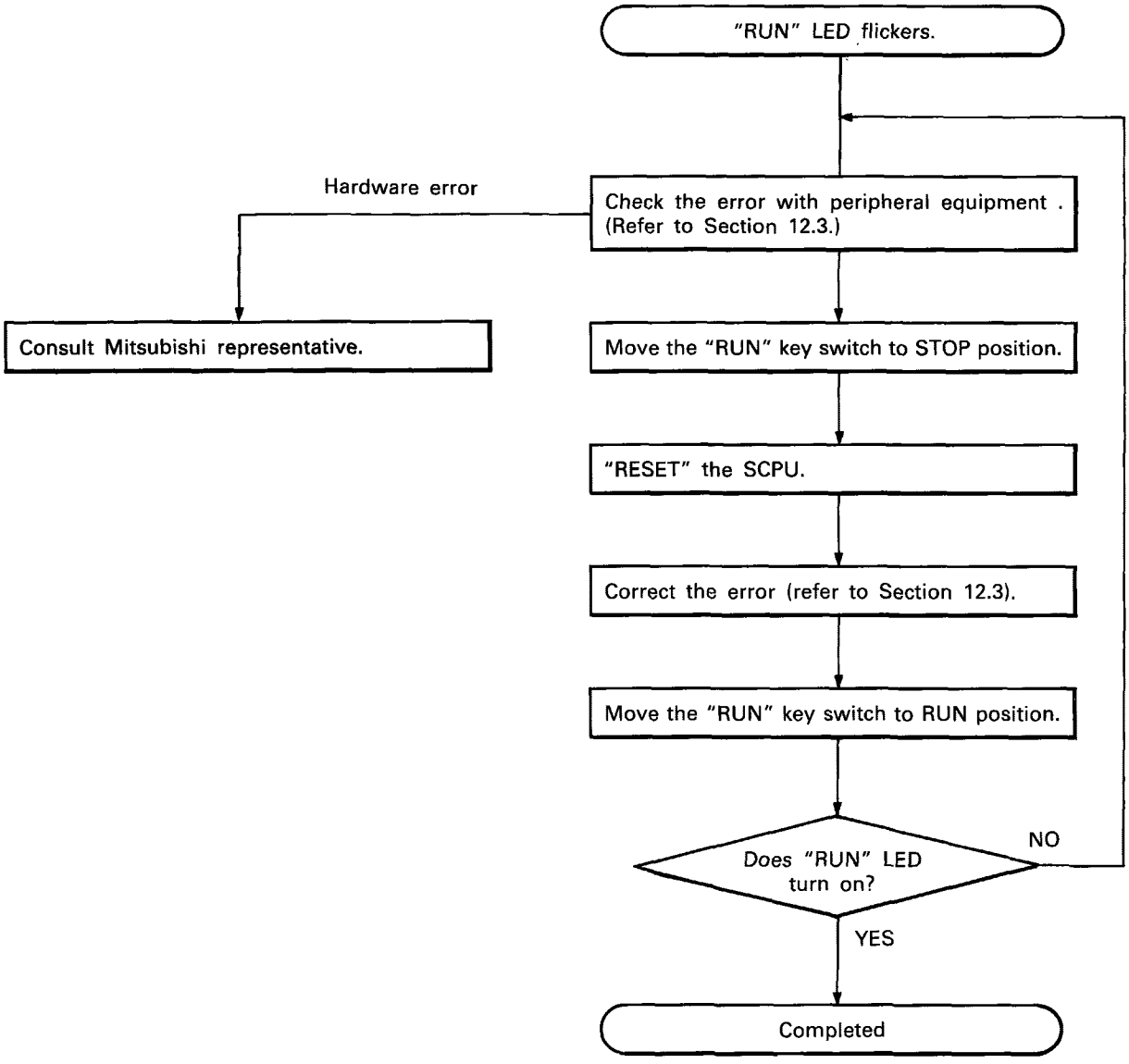


7.3 Flow Chart "RUN" LED Off

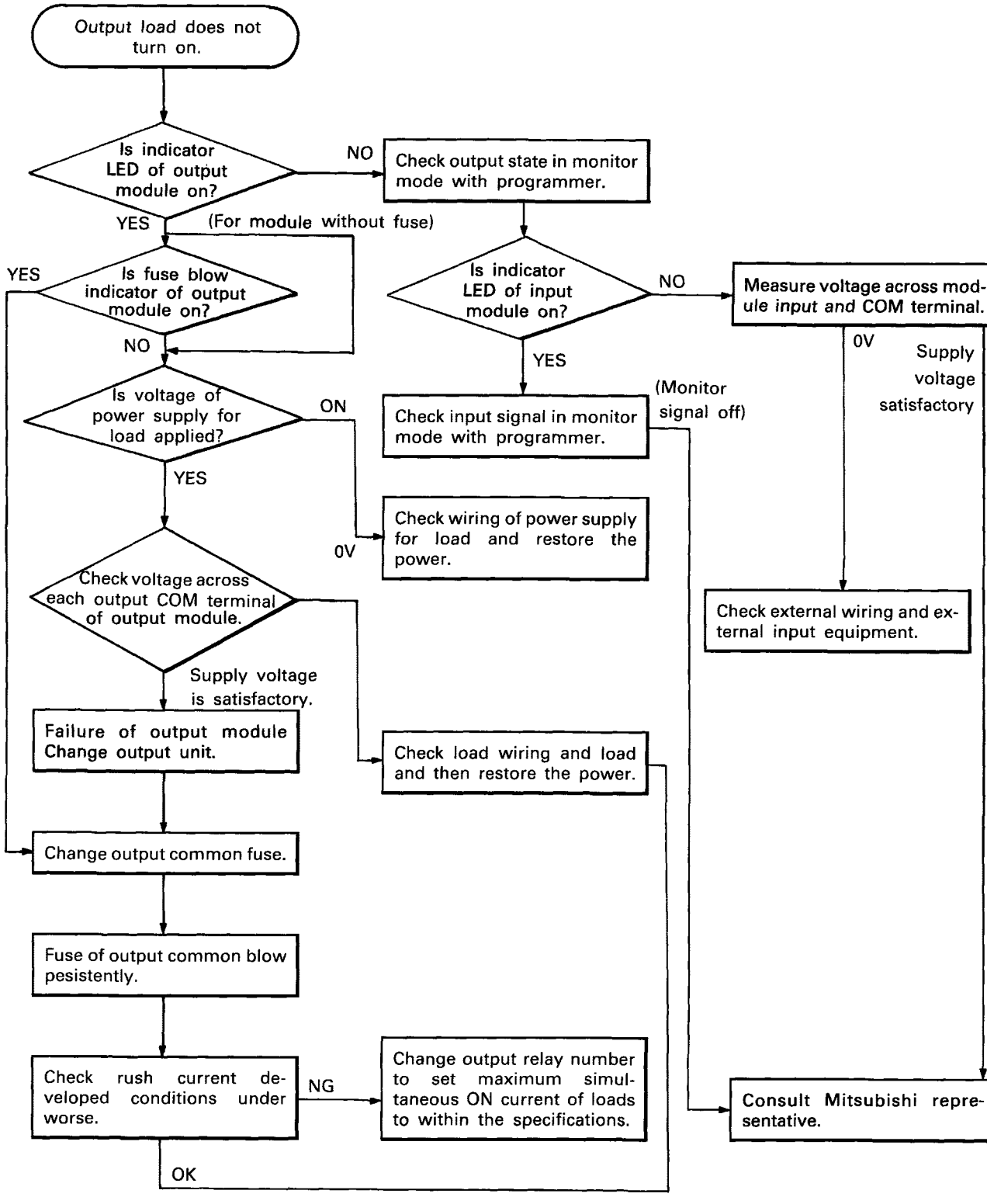


7.4 Flow Chart "RUN" LED Flickers

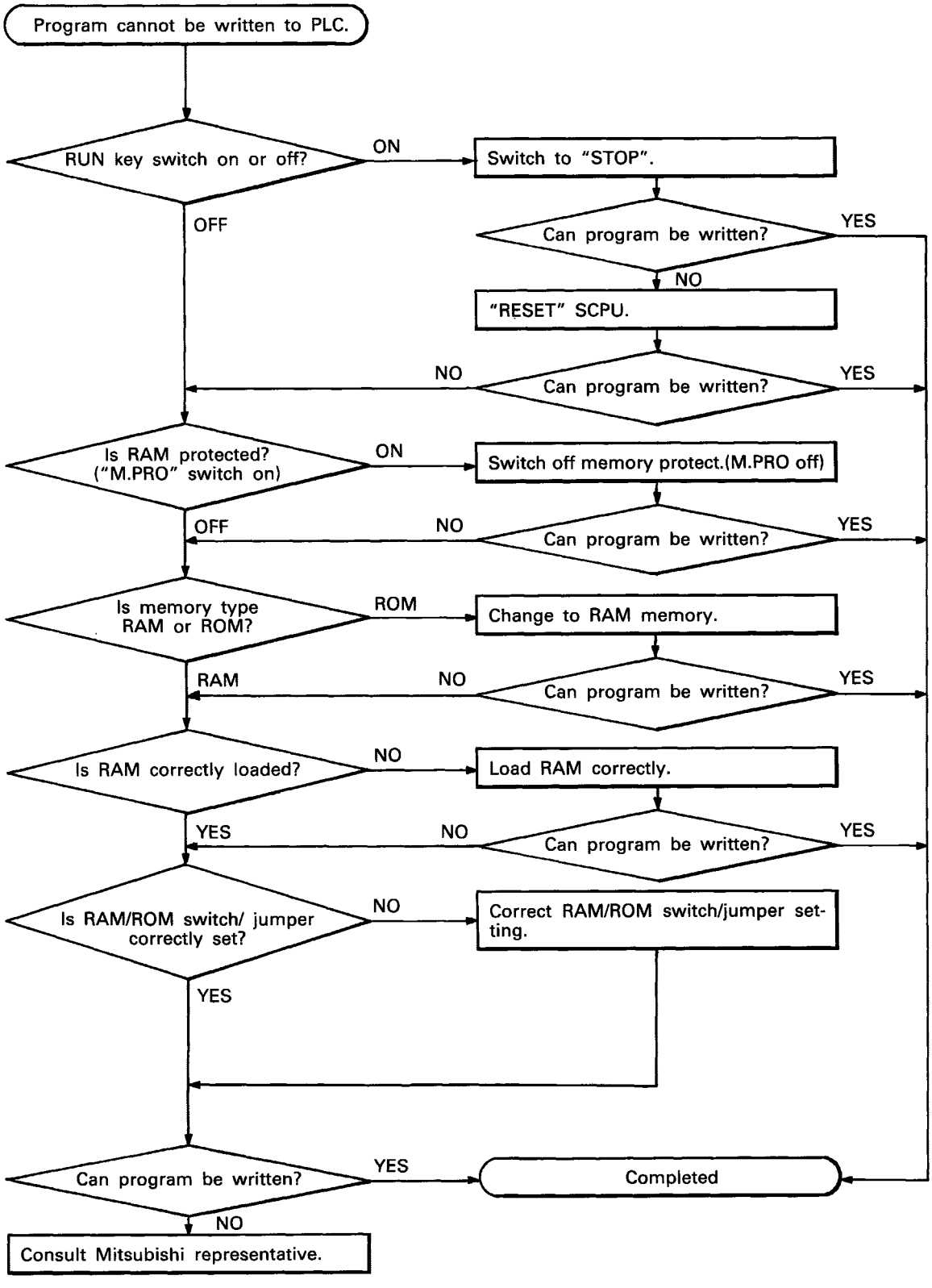
The A3NCPU is fitted with an ASCII character display which will indicate any error which has caused the RUN LED to flicker.



7.5 Flow Chart Load of Output Module does not Turn On



7.6 Malfunction in Program Down Load to PLC



7.7 Error Code List

If an error occurs is RUN mode, an error display or error code (including a step number) is stored in the special register by the self-diagnostic function. The error code reading procedure and the causes and corrective actions for errors are shown in the table below.

Error code list

Error Message	Content of Special Register D9008 (BIN value)	CPU States	Error and Cause	Corrective Action
"INSTRCT. CODE ERR" (Checked during instruction execution)	10	Stop	Instruction code, which cannot be decoded by CPU, is included in the program. (1) ROM including invalid instruction code, has been loaded. (2) Memory contents have been corrected.	(1) Read the error step by use of peripheral equipment and correct the program at that step. (2) In the case of ROM, rewrite the contents of the ROM or change the ROM.
"PARAMETER ERROR" (Checked at power on, reset, STOP to RUN, PAUSE to STEP-RUN)	11	Stop	Capacity larger than the memory capacity of CPU has been set and then write to CPU has been performed.	(1) Check the loading of CPU memory and load it correctly. (2) Read the parameter contents of CPU memory, check and correct the contents, and write them to the memory again.
"MISSING END INS." (Checked at M9056 or M9057 ON, STOP to RUN, PAUSE to STEP-RUN)	12	Stop	(1) There is no END (FEND) instruction in the program. (2) When subprogram has been set in parameters, there is no END instruction in the subprogram.	Write END at the end of the program/subprogram.
"CAN'T EXECUTE (P)" (Checked at [CJ], [SCJ], [JMP], [CALLP] execution, STOP to RUN, PAUSE to STEP-RUN)	13	Stop	(1) There is no jump destination or plural destinations specified by the [CJ], [SCJ], [CALL], [CALLP] or [JMP] instruction. (2) There is a [CHG] instruction and no setting of subprogram. (3) Although there is no [CALL] instruction, the [RET] instruction exists in the program and has been executed. (4) The [CJ], [SCJ], [CALL], [CALLP] or [JMP] instruction has been executed with its jump destination located below the END instruction. (5) The number of [FOR] instructions does not match that of [NEXT] instruction. (6) The [JMP] instruction specified between [FOR and NEXT] has caused execution to deviate from between [FOR and NEXT]. (7) The [JMP] instruction has caused execution to deviate from the subroutine before the [RET] instruction is executed. (8) The [JMP] instruction has caused execution to jump to a step or subroutine between [FOR and NEXT].	(1) Read the error step by use of peripheral equipment and correct the program at that step. (Make correction such as the insertion of jump destination or the changing of jump destinations to one.)

Error Message	Content of Special Register D9008 (BIN value)	CPU States	Error and Cause	Corrective Action
"CAN'T EXECUTE (I)" (Checked at the occurrence of interruption, STOP to RUN, PAUSE to STEP-RUN)	15	Stop	(1) Although the interrupt unit is used, there is no number of interrupt pointer I, which corresponds to that unit, in the program or there are plural numbers. (2) No [IRET] instruction has been entered in the interrupt program. (3) There is [IRET] instruction in other than the interrupt program.	(1) Check for the presence of interrupt program which corresponds to the interrupt unit and create and interrupt program or reduce the same numbers of I. (2) Check if there is [IRET] instruction in the interrupt program and enter the [IRET] instruction. (3) Check if there is [IRET] instruction in other than the interrupt program and delete the [IRET] instruction.
"CASSETTE ERROR" (Checked at power on, reset)	16	Stop	The memory cassette is not loaded.	Load the memory cassette and reset.
"RAM ERROR" (Checked at power on, reset, M9084 ON during STOP)	20	Stop	The CPU has checked if write and read operations can be performed properly to the data memory area of CPU, and as a result, either or both has not been performed.	Since this is CPU hardware error, consult Mitsubishi, representative.
"OPE. CIRCUIT ERR." (Checked at power on, reset)	21	Stop	The operation circuit, which performs the sequence processing in the CPU, does not operate properly.	
"WDT ERROR" (Checked at the execution of END instruction)	22	Stop	Scan time exceeds watch dog error monitor time. (1) Scan time of user program has become excessive. (2) Scan time has lengthened due to instantaneous power failure which occurred during scan.	(1) Calculate and check the scan time of user program and reduce the scan time by use of [CJ] instruction, etc. (2) Monitor the content of special register D9005 by use of peripheral equipment. When the content is other than 0, line voltage is insufficient. Therefore, check the power and eliminate the voltage fluctuation.
"END NOT EXECUTE" (Checked at the execution of END instruction)	24	Stop	(1) When the END instruction is executed, another instruction code has been read due to noise, etc. (2) The END instruction has changed to another instruction code for some reason.	Perform reset and run. If the same error is displayed again, it is the CPU hardware error. Therefore, consult Mitsubishi representative.
"WDT ERROR" (Checked continuously)	25	Stop	The END instruction cannot be executed with the program looped.	Check for an endless loop and correct the program.
"UNIT VERIFY ERR." (Checked at the execution of END instruction (Not checked when M9084 or M9094 is on))	31	RUN (Stop)	I/O module data is different from that at power-on. (1) The I/O module (including the special function module) is incorrectly disengaged or has been removed, or a different module has been loaded.	(1) Among special registers D9116 to D9123, the bit corresponding to the module verify error is "1". Therefore, monitor the registers by use of peripheral equipment and check for the module with "1". (2) When the fault has been corrected reset CPU.

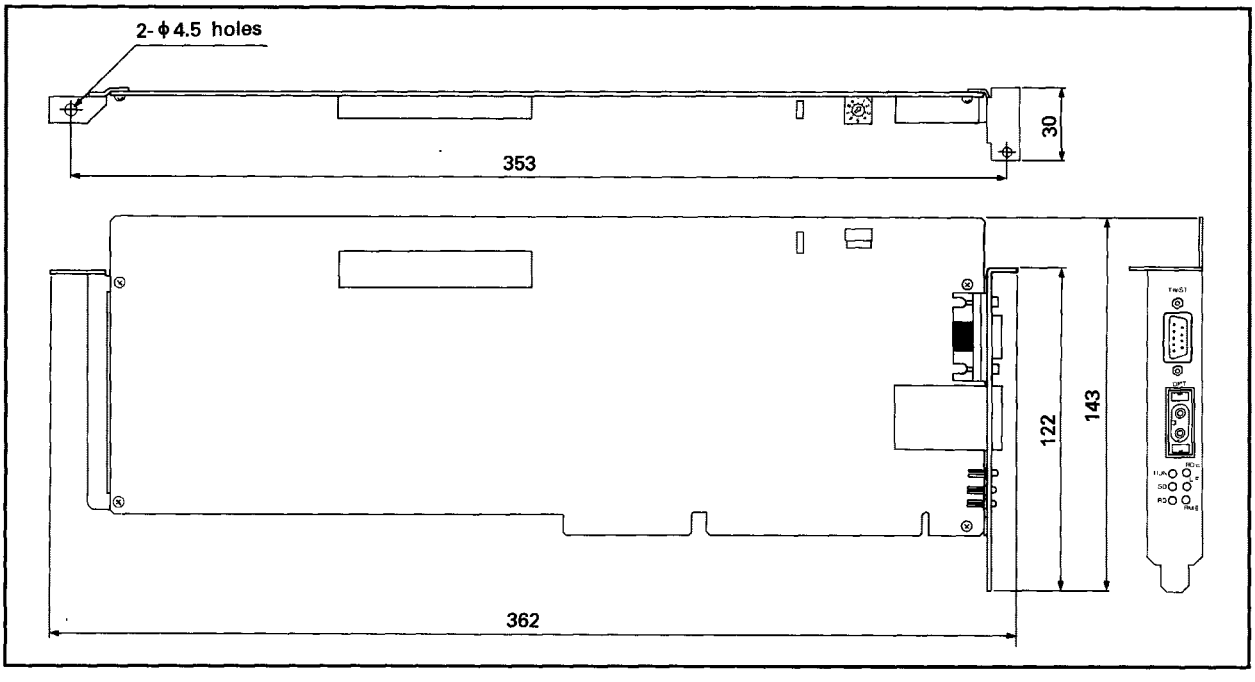
Error Message	Content of Special Register D9008 (BIN value)	CPU States	Error and Cause	Corrective Action
"FUSE BREAK OFF" (Checked at the execution of END instruction (Not checked when M9084 or M9094 is on))	32	RUN (Stop)	There is an output module of which fuse has blown.	(1) Check the fuse blow indicator LED of output module and change the fuse of module of which LED is on. (2) The check of fuse blow module can also be made by the peripheral equipment. Among special registers D9116 to D9123, the bit corresponding to the module of verify error is "1". Therefore, make checks by monitoring the registers.
"CONTROL -BUS ERR." (Checked at the execution of FROM and TO instructions)	40	Stop	The FROM and TO instructions cannot be executed. (1) Error of control bus with special function module.	Since this is the special function module, CPU module or base unit hardware error. Therefore, change the unit and check the defective module. For the defective module, consult Mitsubishi representative.
"SP. UNIT DOWN" (Checked at the execution of FROM and TO instructions)	41	Stop	When the FROM or TO instruction is executed, access has been made to the special function module but the answer is not given. (1) The accessed special function module is defective.	Since this is the accessed special function unit error, consult Mitsubishi representative.
"LINK UNIT ERROR" (Checked at power on, reset, STOP to RUN, PAUSE to STEP-RUN)	42	Stop	AJ71R22 or AJ71P22 is loaded in the master station.	Remove the AJ71R22 or PJ71P22 from the master station. After correction, perform reset and start at the initial operation.
"I/O INT. ERROR" (Checked at the occurrence of interruption)	43	Stop	Although the interrupt module is not loaded, interruption has occurred.	Since this is certain unit hardware error. Therefore, change the unit and check the defective unit. For the defective unit, consult Mitsubishi representative.
"SP.UNIT LAY.ERR." (Checked at power on, reset, STOP to RUN, PAUSE to STEP-RUN)	44	Stop	(1) Three or more computer link modules are loaded with respect to one CPU module. (2) Two or more units of AJ71P21 or AJ71R21 are loaded. (3) Two or more interrupt modules are loaded. (4) In the parameter setting of A6GPP, while I/O module is actually loaded, special function module has been set in the I/O assignment, and vice versa.	(1) Reduce the computer link modules to two or less. (2) Reduce the AJ71P21 or AJ71R21 to one or less. (3) Reduce the interrupt module to one. (4) Re-set the I/O assignment of parameter setting by use of A6GPP according to the actually loaded special function module.
"SP. UNIT ERROR" (Checked at the execution of FROM and TO instructions)	46	Stop (Run)	Access (execution of FROM to TO instruction) has been made to a location where there is no special function module.	Read the error step by use of peripheral equipment, and check and correct the content of FROM or TO instruction at that step by use of peripheral equipment.

Error Message	Content of Special Register D9008 (BIN value)	CPU States	Error and Cause	Corrective Action
"LINK PARA. ERROR" (Checked at power on, reset, STOP to RUN, PAUSE to STEP-RUN)	47	Run	(1) The contents, which have been written to the parameter area of link by setting the link range in the parameter setting of A6GPP, A6PHP or A6HGP, are different from the link parameter contents for some reason. (2) The setting of the total number of slave stations is 0.	(1) Write parameters again and make check. (2) When the error is displayed again, it is the hardware error. Therefore, consult Mitsubishi representative.
"OPERATION ERROR" (Checked at instruction execution)	50	Run	(1) The result of BCD conversion has exceeded the specified range (9999 or 99999999). (2) Setting has been performed exceeding the specified device range and operation cannot be performed. (3) File registers are used in the program without performing the capacity setting of file registers.	Read the error step by use of peripheral equipment, and check and correct the program at that step. (Check device setting range, BCD conversion value, etc.)
"BATTERY ERROR" (Checked continuously (Not checked when M9084 is on))	70	Run	(1) The battery voltage has reduced to less than the specified value. (2) The battery lead is disconnected.	(1) Change the battery. (2) When RAM or power failure compensation is used, connect the battery.

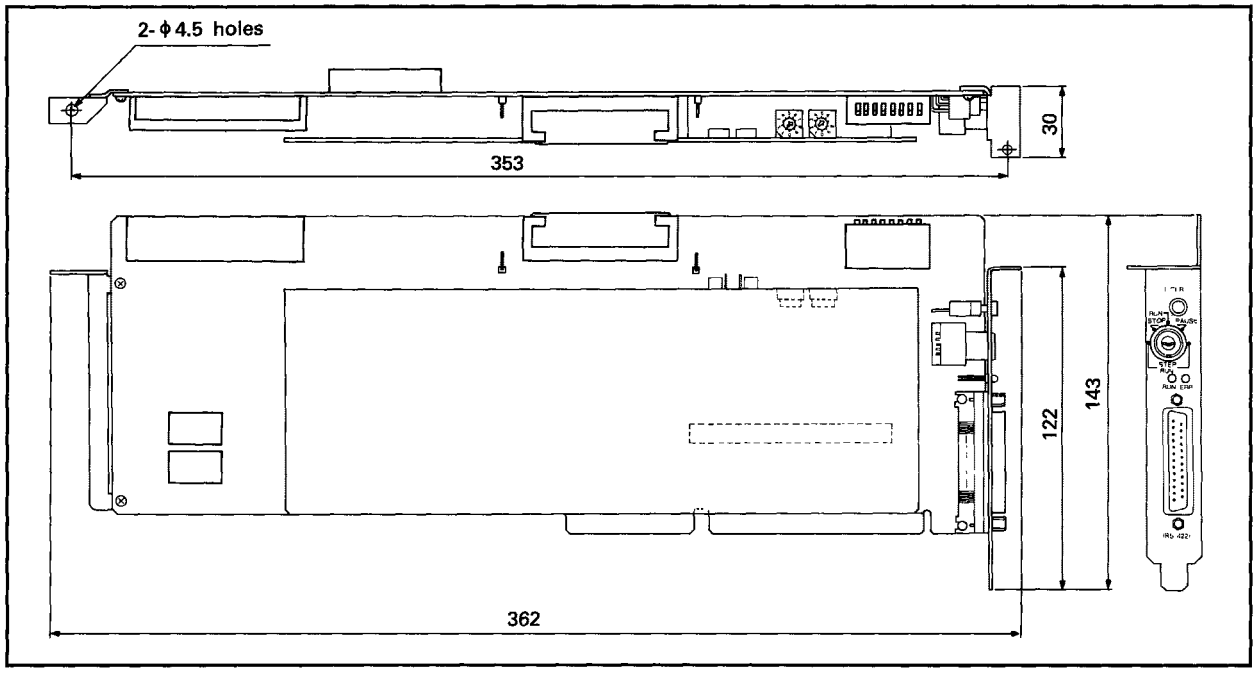
APPENDICES

APPENDIX 1 External Dimensions

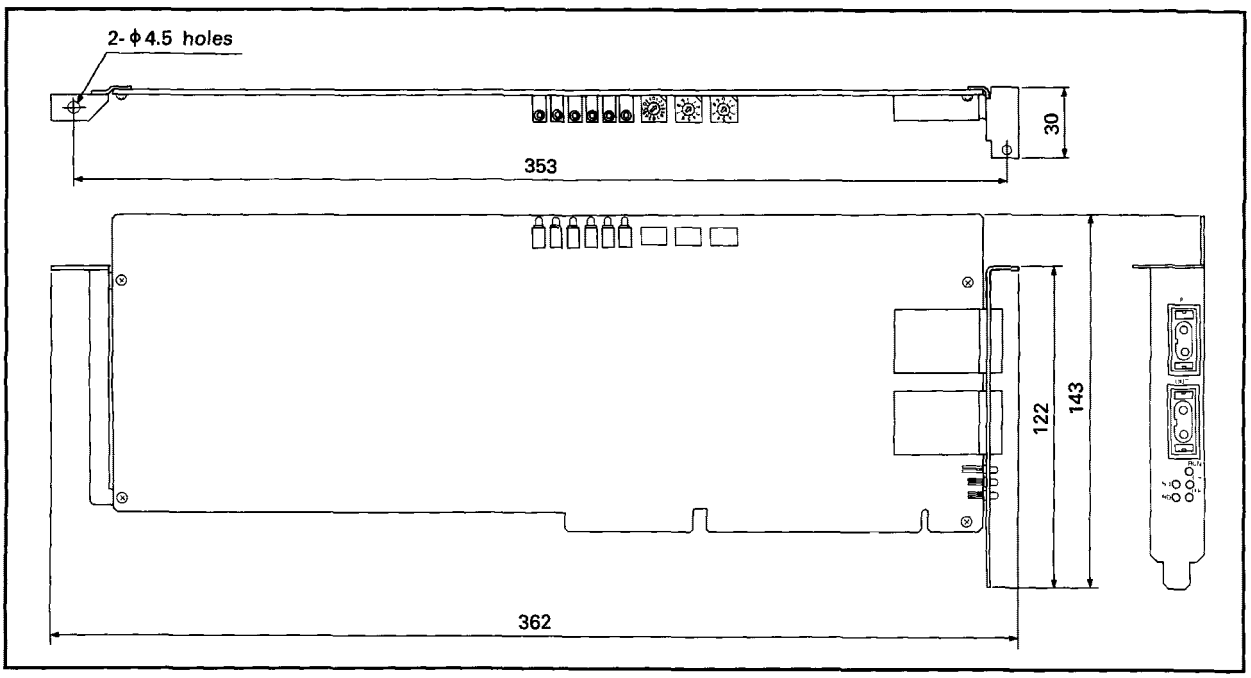
A7BDE-A3N-PT32S3A



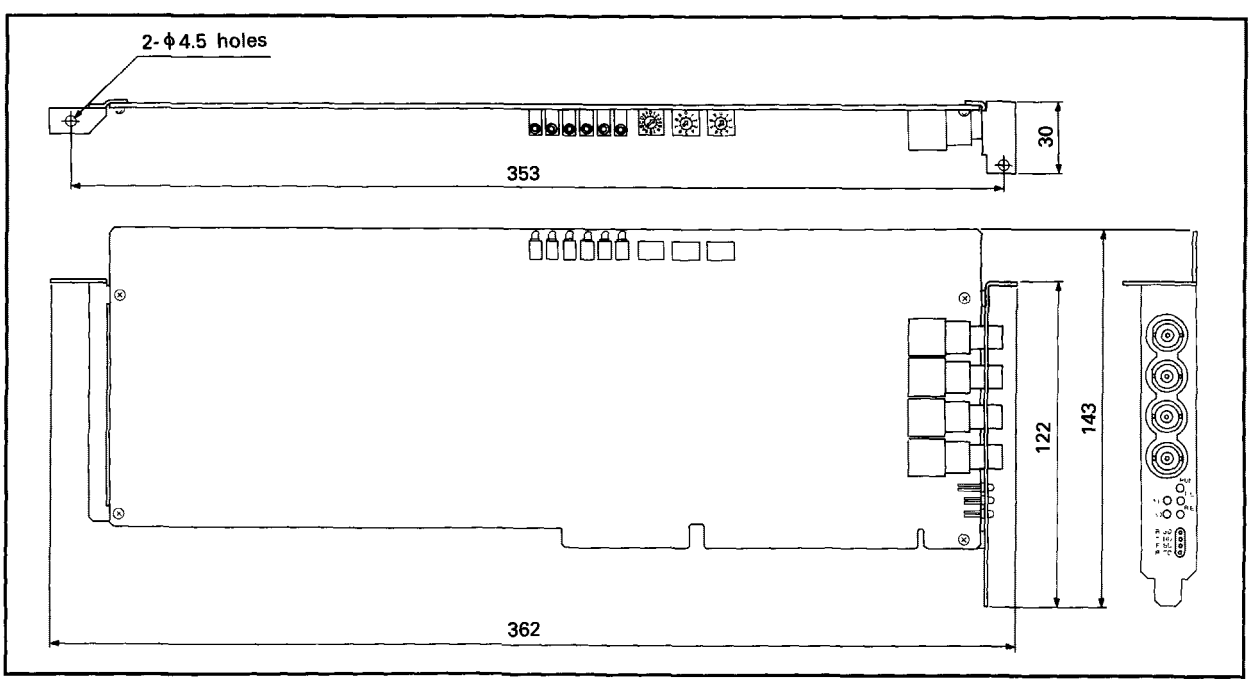
A7BDE-A3N-B.C



A7LU1EP21



A7LU1ER21



APPENDIX 2 Differences in the A7BDE-A3N-PT32S3 and A3NCPU

(a) Differences in Specifications

Item \ Type		A7BDE-A3N-PT32S3	A3NCPU
Control system		Repeated operation (using stored program)	
I/O control method		Direct method	Refresh and direct methods
Programming language		Language dedicated to sequence control Combined use of relay symbol type and logic symbolic language.	
Combined use of MELSAP language		Allowed	
Number of instructions	Sequence instruction	22 types	
	Basic instruction	132 types	
	Application instruction	107 types	109 types
Processing speed (sequence instruction) (sec/step)		1.0 to 2.3	Direct method: 1.0 to 2.3 Refresh method: 1.0
I/O points		2048 points	
Constant scan function (starting a program in fixed intervals)		Settings are possible in 10ms intervals over a range of 10 to 1990ms.	
Watch dog timer (WDT)		Settings are possible in 10ms intervals over a range of 10 to 2000ms.	
Allowable power failure period		10ms or less	20ms or less
Memory capacity		64 KB	Maximum 320 KB
Program capacity		(Main sequence program + main microcomputer program) — maximum of 30K steps main microcomputer program can be set to a maximum of 58 KB (29K steps))	
		(Sub sequence program + sub microcomputer program) — maximum of 30K steps sub microcomputer program can be set to a maximum of 58 KB (29K steps))	
Internal relay (M) (points)		1000 (M0 to 999)	Total number of M, L and S — 2048 (set by parameters)
Latch relay (L) (points)		1024 (L1000 to 2047)	
Step relay (S) (points)		0 points (None in initial status)	
Link relay (B) (point)		1024 (B0 to 3FF)	
Timer (T)	Number of points	256	
	Specifications	100ms timer: setting time 0.1 to 3276.7sec (T0 to 999) 10ms timer: setting time 0.01 to 327.67sec (T200 to 255) 100ms retentive timer: (0.1 to 3276.7sec)	(Set by parameters)
Counter (C)	Number of points	256	
	Specifications	Normal counter: setting range 1 to 32767 (C0 to 255) Interrupt counter: setting range 1 to 32767 Counters used in interrupt programs	Set by parameters

Item \ Type	A7BDE-A3N-PT32S3	A3NCPU
Data register (D) (points)	1024 (D0 to 1023)	
Link register (W) (points)	1024 (W0 to 3FF)	
Annunciator (F) (points)	256 (F0 to 255)	
File register (R) (points)	Max. 8192 (R0 to 8191)	
Accumulator (A) (points)	2 (A0,A1)	
Index register (V,Z) (points)	2 (V,Z)	
Pointer (P)	256 (P0 to 255)	
Pointer for interrupt (I)	32 (I0 to 31)	
Special relay (M)	256 (M9000 to 9255)	
Special register (D)	256 (D9000 to 9255)	
Comment points	Max. 4032	
Status latch function	Available	
Sampling trace function	Available	
Offline switch function	Available (Y, M, L, B, F)	
Annunciator display function	F number display	
Remote RUN/PAUSE contact setting	Available	
Operation mode switching when error occurs	Available	
STOP RUN output mode switching	Available	
Keyword entry	Available	
Print title entry	Available	
Assignment change of number I/O occupied points	Possible with peripherals (with the exception of the PU)	
Setting of latch range for power failure data retention	The following latch ranges are permitted: B0 to 3FF, T0 to 255, C0 to 255, D0 to 1023, W0 to 3FF	
Step operation	Break point stop and 1 instruction operation are possible.	
Clock	Year, month, date, hour, minute, second, and day of the week can be written to and read from the special register.	
LED display	None (The content displayed on the A3NCPU LEDs can be confirmed using the board information of the option board setting.)	Information can be displayed in a 16-character display on the front panel for the CPU module. The kinds of data displayed include error comments resulting from errors occurring during self-diagnosis, and comments resulting from OUTF and SETF.
Method for LED display reset	LEDs are reset using board data derived from the option board settings.	LED display is reset using the LED display reset witch.
Method for hardware reset	Hardware is reset using board data derived from the option board settings.	Hardware is reset using the reset switch.

(b) Differences in Instruction Specifications

All the instructions of the A7BDE-A3N-PT32S3 (SCPU) and A3NCPU are the same. However, the instructions listed below have varying conditions.

(1) PR/PRC instruction
The PR and PRC instructions cannot be used to display the data on the A6FD (external display unit) which is connected to an output module.
This is because the extension base unit cannot be connected to the A7BDE-A3N-PT32S3 (SCPU). Even if the PR/PRC instruction is executed to output module of a remote I/O station, a correct display cannot be obtained if to the period of the link scan time is shorter than the strobe signal duration of 10ms.

(2) SEG instruction
The SEG instruction should be used as a 7 segment decode instruction with M9052 turned OFF. If the SEG instruction is executed with M9052 ON, partial refresh processing is conducted. However, because the A7BDE-A3N-PT32S3 has direct processing only, the above partial refresh processing will not be realised.

(c) Differences in Special Relay and Special Register Specifications

All the special relays (M9000 to M9255) and the special registers (D9000 to D9255) of the A7BDE-A3N-PT32S3 (SCPU) and the A3NCPU are the same.
However, the following special relays and special registers are not used.
*M9049 (changing the number of output characters)
*M9052 (SEG instruction switch)
*M9094 (I/O exchange flag)

*D9094 (Exchange I/O first I/O number)

APPENDIX 3 Driver Start-Up Error Messages

No.	Contents		Start State
0	Message	MELSEC DRIVER M-A3N.SYS Ver.00A	Success
	Contents	Started correctly.	
1	Message	ERROR 0001 IN MELSEC DRIVER M-A3N.SYS INT-A PARAMETER ERROR	Failure
	Contents	Characters in argument (1) are not INT-A.	
2	Message	ERROR 0002 IN MELSEC DRIVER M-A3N.SYS INT-A NUMBER ERROR	Failure
	Contents	The number for argument (1) is not between 0x60 and 0xff.	
3	Message	ERROR 0003 IN MELSEC DRIVER M-A3N.SYS BD PARAMETER ERROR	Failure
	Contents	Characters in argument (2) are not BD.	
4	Message	ERROR 0004 IN MELSEC DRIVER M-A3N.SYS BD NUMBER ERROR	Failure
	Contents	The number for argument (2) is not between 0 and 7.	
5	Message	ERROR 0005 IN MELSEC DRIVER M-A3N.SYS INT-B PARAMETER ERROR	Failure
	Contents	Characters in argument (3) are not INT-B	
6	Message	ERROR 0006 IN MELSEC DRIVER M-A3N.SYS INT NUMBER ERROR	Failure
	Contents	The number for argument (3) is not between 0 and 7.	
7	Message	ERROR 0007 IN MELSEC DRIVER M-A3N.SYS BOARD NOT FOUND	Failure
	Contents	No board is found at the location indicated by argument (2). Causes: (1) The board is not loaded. (2) The number set for argument (2) overlaps the number of the other board. (3) The other board and the 2-port memory overlap each other.	
8	Message	ERROR 0008 IN MELSEC DRIVER M-A3N.SYS BOARD NOT RESPONSE.	Failure
	Contents	Communication with the board is not possible when starting the driver. Causes: (1) The board is not loaded correctly. (2) The number set for argument (2) overlaps the number of the other board.	
10	Message	ERROR 0010 IN MELSEC DRIVER N-A3N.SYS 100H/300H PARAMETER ERROR	Failure
	Contents	The number set with the I/O port setting pin on the board and the number set for argument (4) do not agree. The number set for argument (4) is not between 100H and 300H.	
11	Message	ERROR 0011 IN MELSEC DRIVER M-A3N.SYS SET UP PIN NOT "AT" ERROR	Failure
	Contents	The AT setting pin on the board is not at the AT position.	

APPENDIX 4 Function Return Values and Error Codes

The following table shows the return value for the driver functions.

Table	Error No.	Contents of Return Value
1	0x00	Normal termination
	0x01 to 0x3f, 0xffff	Board error
2	0x40 to 0x7f	Processing request error
3	0x80 to 0x0cf	Data error
4	0x0d0 to 0x0ff	Board detection error

(1) Normal termination or board error

Return Value (HEX)	Error Contents	Countermeasures
0	Normal termination	
1	The driver has not started.	Correct the error that occurred when starting the driver.
2	Board response error Time-out while waiting for a response to the processing.	Check that the board is mounted correctly.
4	A function other than the "nl1sync" is requested during SEND/RECEIVE processing. The "nl1sync" function is requested during processing other than SEND/RECEIVE processing.	Synchronize with SYNC. Correct so that SYNC is not executed.
FFFF	Status (decimal -1) During SEND/RECEIVE processing	Synchronize with SYNC.

(2) Processing request error

Return Value (HEX)	Error Contents	Countermeasures
40	Command error A command other than NL10PEN, NL1CLOSE, NL1RECEIVE, or NL1SEND is set.	Correct the command code. (Correct the library.)
41	Channel error A unregistered channel number is set.	Correct the channel number.
42	Open error The designated channel is already opened.	Specify the OPEN command only once.
43	Close error The designated channel is already closed.	Specify the CLOSE command only once.
44	Path error The designated path number has not been opened through the communication line.	Change the path number to the one opened through the communication line.
45	Processing code error An unsupported processing code has been set. The processing code requested to the A3N board host station cannot be processed by itself.	Correct the ARG1 processing code.

(3) Data error

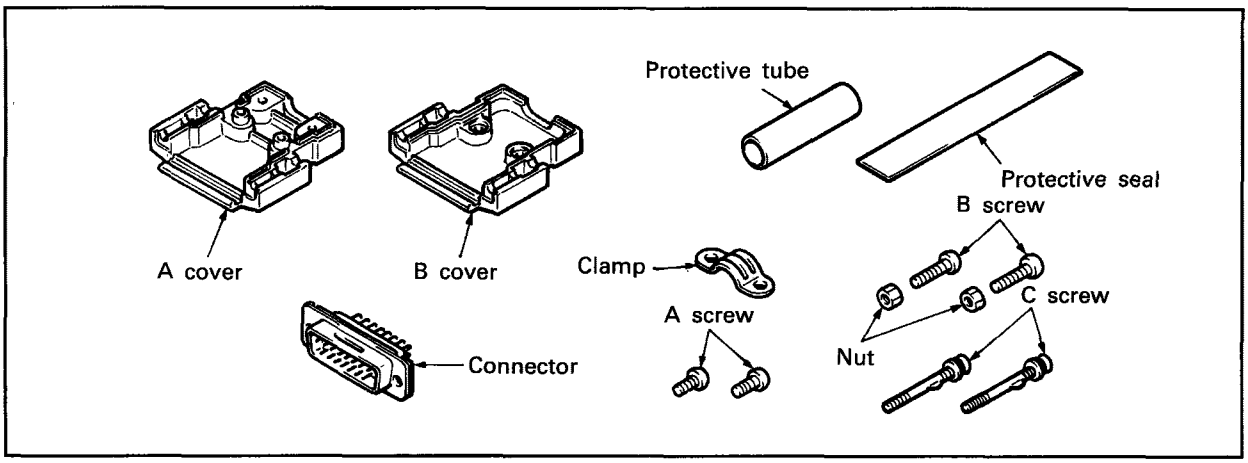
Return Value (HEX)	Error Contents	Countermeasures
80	Byte/point number read error The number of bytes (batch read) or the number of points (random read) is outside the allowable range.	Set the number within the allowable range.
82	X number or Y number error The head X number designation for writing input X is not "0" or "8". The head Y number designation for reading output Y is not "16".	Correct the X number or Y number.
83	X point or Y point number error In the "input X writing" operation, the set number is not "8" or "16" when the head number X designation is "0", or the set number is not "8" when the head number X designation is "8". In the "output Y reading" operation, the set number is not "8".	Correct the X number or Y point.
84	Byte/point number write error The number of bytes (batch write) or the number of points (random write) is outside the allowable range.	Set the number within the allowable range.
87	Remote designation error A setting other than RUN/STOP/PAUSE is set.	Set RUN/STOP/PAUSE for remote setting.
88	Random write designation error A code other than set (0), reset (1), and write (2) is set.	Set set/reset/write for random write.
89	Canceling processing The next processing request was given before the current processing was completed.	Give the next processing request only after the current processing has been completed.
8A	Switch number designation error The set switch number is not "0" or "1" for the switch reading operation.	Correct the designated switch number.

(4) Board detection error

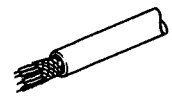
Return Value (HEX)	Error Contents	Countermeasures
E0	PC No. error The request destination station does not exist.	Correct the station number.
E1	Processing mode error The request destination ACPU cannot process the processing code. This was checked by the request destination ACPU.	Check the request destination ACPU and the processing code.
E2	Special module designation error The designated special module cannot do the require processing.	Correct the Y No.
E3	Other data error An error is contained in the part of the data, such as the request data address, head step, or the number of shift bytes.	Correct the request data.
E4	Link designation error The set request destination station cannot process the processing code. This was checked by the request destination station.	Check the request destination station and the processing code.
E8	Remote error The keyword in the remote RUN/STOP/PAUSE request does not match.	Find the source station where the corresponding remote STOP/PAUSE request is given to the request destination ACPU.
E9	Link time-over The request source stopped the link during processing.	Reestablish the link.
EA	Special module busy The designated special module is carrying out other processing.	Check the special module hardware.
EC	Request destination busy When sending general data, either the request destination receive buffer is full or the request destination station is not ready for receiving.	Give the receive request when the request destination is in a condition to receive data.
F0	Link error A request is given to an off-the-link station.	Establish the link.
F1	Special module busy error The designated special module is not ready to begin processing.	Check the special module hardware.
F2	Special module time-over No response is returned from the designated special module.	Check the special module hardware.

APPENDIX 5 Assembly of MELSECNET/MINI Twisted Pair Connector

The twisted-pair link connector is constructed of the following components.

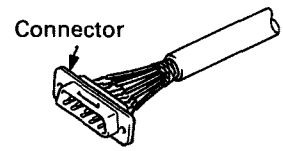


The following section provides the procedure for assembling a connector for twisted-pair link application.



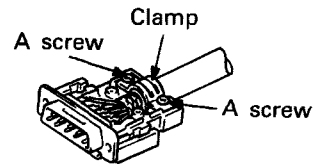
(a) Remove the outer cover of the shielded wire. The exposed shielding should be long enough for it to be clamped.

⇩



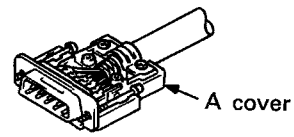
(b) Solder the wires to the connector.

⇩



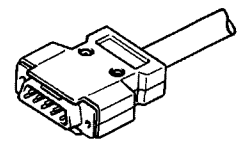
(c) Fit connector onto the A cover and clamp the shielded wire firmly with the clamp and connect to the IBM® PC/AT® FG.

⇩



(d) Mount the C screws to the A cover.

⇩



(e) Place the B cover on the A cover, place the nuts on the B screws and tighten firmly.

APPENDIX 6 Special Relays and Registers

(a) Special relay list

Special relay list
The special relays are internal relays used for specific purposes. Therefore, do not turn on or off the special relays in the program.

Number	Name	Description	Details
*1 M9000	Fuse blown	OFF: Normal ON: Presence of fuse blow module	Turned on when there is one or more output modules of which fuse has been blown. Remains on if normal status is restored.
*1 M9002	I/O module verify error	OFF: Normal ON: Presence of error	Turned on if the status of I/O module is different from entered status when power is turned on. Remains on if normal status is restored.
*1 M9005	AC DOWN detection	OFF: AC is good ON: AC is down	Turned on if power failure of within 20ms occurs. Reset when POWER switch is moved from OFF to ON position.
M9006	Battery low	OFF: Normal ON: Battery low	Turned on when battery voltage drops below the specified value. Turned off when battery voltage becomes normal.
*1 M9007	Battery low latch	OFF: Normal ON: Battery low	Turned on when battery voltage drops below the specified value. Remains on if battery voltage becomes normal
*1 M9008	Self-diagnostic error	OFF: Absence of error ON: Presence of error	Turned on when an error is found as a result of self-diagnosis.
M9009	Annunciator detection	OFF: Absence of detection ON: Presence of detection	Turned on when OUT F or SET F instruction is executed. Switched off when D9124 value is set to 0.
M9010	Operation error flag	OFF: Absence of error ON: presence of error	Turned on when operation error occurs during execution of an application instruction. Turned off when the error is eliminated.
*1 M9011	Operation error flag	OF: Absence of error ON: Presence of error	Turned on when operation error occurs during execution of an application instruction. Remains on if normal status is restored.
M9012	Carry flag	OFF: Carry off ON: Carry on	Carry flag used in an application instruction
M9016	Data memory clear flag	OFF: No processing ON: Output clear	Clears all data memory (except special relays and special registers) in the remote run mode from a computer, etc. when M9016 is 1.
M9017	Data memory clear flag	OFF: No processing ON: Output clear	Clears all unlatched data memory (except special relays and special registers) in the remote run mode from a computer, etc. when M9017 is 1.
M9020	User timing clock No. 0		Relay which repeats on/off at predetermined scan intervals. When power is turned on or reset is performed, the clock starts with off. Set the on/off intervals by executing the DUTY instruction.
M9021	User timing clock No. 1		
M9022	User timing clock No. 2		
M9023	User timing clock No. 3		
M9024	User timing clock No. 4		
*2 M9025	Clock data set request	OFF: No processing ON: Data set request	Writes clock data from D9025 to D9028 to the clock devices after the END instruction is executed at the can when M9025 is switched on.
M9026	Clock data error	OFF: No error ON: Error	Switched on when a clock data (D9025 to D9028) error occurs.
M9027	Clock data display	OFF: No processing ON: Display	Displays clock data (D9025 to D9028) on the LED on the CPU front panel.
*2 M9028	Clock data read request	OFF: No processing ON: Read request	Reads clock data in BCD to D9025 to D9028 when M9028 is switched on.
M9030	0.1 sec. clock		0.1 second, 0.2 second, 1 second, 2 second, and 1 minute clocks are generated. Not turned on and off per scan but turned on and off even during scan if the corresponding time has elapsed. Starts when power is turned on or reset is performed.
M9031	0.2 sec. clock		
M9032	1 sec. clock		
M9033	2 sec. clock		
M9034	1 min. clock		
M9036	Normally ON	ON OFF	Used as dummy contacts of initialization and application instruction in sequence program. M9036 and M9037 are switched on/off independently of the CPU RUN/STOP switch position. M9038 and M9039 are switched on/off in accordance with the RUN/STOP switch position, i.e. switched off when the switch is set to STOP. When the switch is set to other than STOP, M9038 is switched on only during 1 can and M9039 is switched off only during 1 scan.
M9037	Normally OFF	ON OFF	
M9038	On only for 1 scan after run	ON OFF	
M9039	RUN flag (off only for 1 scan after run)	ON OFF	

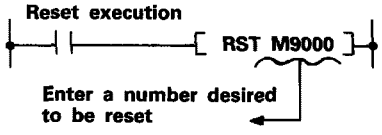
Number	Name	Description	Details
M9040	PAUSE enable coil	OFF: PAUSE disabled ON: PAUSE enabled	When RUN key switch is at PAUSE position or remote pause contact has turned on and if M9040 is on, PAUSE mode is set and M9041 is turned on.
M9041	PAUSE status contact	OFF: Not during pause ON: During pause	
M9042	Stop status contact	OFF: Not during stop ON: During stop	Switched on when the RUN/STOP switch is set to STOP.
M9043	Sampling trace completion	OFF: During sampling trace ON: Sampling trace completion	Turned on upon completion of sampling trace performed the number of times preset by parameter after STRA instruction is executed. Reset when STRAR instruction is executed.
M9044	Sampling trace	0 1: Same as STRA execution 0: Same as STRAR execution	Has the same functions as the STRA and STRAR instructions. (M9044 is forced to switch on/off by the peripheral device.) When switched on, M9044 provides the same function as the STRA instruction. When switched off, M9044 provides the same function as the STRAR instruction. At this time, the sampling trace condition is based on the value in D9044. (0 for scan, time for time (10ms increments))
M9046	Sampling trace	OFF: Except during trace ON: During trace	On during sampling trace.
M9047	Sampling trace preparation	OFF: Sampling trace stop ON: Sampling trace start	Sampling trace is not executed until M9047 is turned on. By turning off M9047, sampling trace is stopped.
M9051	CHG instruction execution disable	OFF: Disable ON: Enable	Switch on to disable CHG instruction. Switch on to request program transfer. Automatically switched off on completion of the transfer.
*2 M9053	EI/DI instruction switching	OFF: Sequence interrupt control ON: Link interrupt control	Switch on to execute the link refresh enable, disable (EI, DI) instructions.
M9054	STEP RUN flag	OFF: Not during step run ON: During step run	Switched on when the RUN/STOP switch is in STEP RUN.
M9055	Status latch completion flag	OFF: Uncompleted ON: Completed	Turned on when status latch is completed. Turned off by reset instruction.
M9056	Main program P, I set request	ON: During P, I set request OFF: Except during P, I set request	Switch on upon completion of the transfer of another program during RUN (e.g. subprogram during RUN of the main program). Automatically switched off when P, I setting is complete.
M9057	Subprogram P, I set request	ON: During P, I set request OFF: Except during P, I set request	
*2 M9084	Error check setting	OFF: Error checked ON: Error unchecked	Used to set whether or not the following error checks are made at the execution of the END instruction. (To shorten END instruction processing time) Fuse blown, I/O unit verify error, battery error

POINT

(1) All special relays are switched off by any of the power-off, latch clear and reset operations. The special relays remain unchanged when the RUN/STOP switch is set to STOP.

(2) The above relays marked *1 remain "on" if normal status is restored. Therefore, to turn them "off", use the following method:

1) Method by user program
Insert the circuit shown at right into the program and turn on the reset execution command contact to clear the special relay M.



2) Method by peripheral equipment
Forcibly reset the special relay by the test function of peripheral equipment.
For the operation procedure, refer to the manual of each peripheral equipment.

3) By moving the RESET key switch at the CPU front to the RESET position, the special relay is turned "off".

(3) Special relays marked *2 are switched on/off in the sequence program.

(4) Special relays marked *3 are switched on/off in test mode of the peripheral.

(b) Special register D

The special registers are data registers used for specific purposes. Therefore, do not write data to the special registers in the program (except the ones with numbers marked * in the table).

Number	Name	Stored Data	Description
D9000	Fuse blown	Fuse blown module number	When fuse flow modules are detected, the smallest number of the detected units is stored in hexadecimal. (Example: When fuse of Y50 to 6F output modules have blown, "50" is stored in hexadecimal.) To monitor D9000 data using a peripheral equipment, perform monitoring in hexadecimal display. (Cleared when all contents of D9100 to D9107 are reset to 0.)
D9002	I/O module verify error	E/O module verify error module number	If I/O module data is different from data entered are detected when the power is turned on, the first I/O number of the smallest number module among the detected modules is stored in hexadecimal. (Storing method is the same as that of D9000.) To monitor D9002 data using a peripheral equipment, perform monitoring in hexadecimal display. (Cleared when all contents of D9116 to D9123 are reset to 0.)
*1 D9005	AC DOWN counter	AC DOWN time count	Number "1" is added each time input voltage becomes 80% or less of rating while the CPU module is performing operation, and the value is stored in BIN code.
*1 D9008	Self-diagnostic error	Self-diagnostic error number	When an error is found as a result of self-diagnosis, the error number is stored in BIN code.
D9009	Annunciator detection	F number at which external failure has occurred	When one of F0 to 255 is turned on by OUT F or SET F, the F number, which has been detected earliest among the F numbers which have turned on, is stored in BIN coed. D9009 can be cleared by RST F or LEDR instruction. If another F number has been detected, the clearing of D9009 causes the next number to be stored in D9009.
D9009	Annunciator detection	F number at which external failure has occurred	When one of F0 to 255 is turned on by OUT F or SET F, the F number, which has been detected earliest among the F numbers which have turned on, is stored in BIN code. D9009 can be cleared by RST F or LEDR instruction or moving INDICATOR RESET switch on CPU front to ON position. If another F number has been detected, the clearing of D9009 causes the next number to be stored in D9009.
D9010	Error step	Step number at which operation error has occurred	When operation error has occurred during execution of an application instruction, the step number, at which the error has occurred, is stored in BIN code. Thereafter, each time operation error occurs, the contents of D9010 are renewed.
D9011	Error step	Step number at which operation error has occurred	When operation error has occurred during execution of an application instruction, the step number, at which the error has occurred, is stored in BIN code. Since storage into D9011 is made when M9011 changes from off to on, the contents of D9011 cannot be renewed unless M9011 is cleared by user program.
D9014	I/O control mode	I/O control mode number	The set mode is represented as follows: 0 = I/O indirect mode 1 = Input in refresh mode, output in direct mode 3 = I/O in refresh mode

Number	Name	Stored Data	Explanation
D9015	CPU operating status	CPUOperating states	• The operating states of CPU as shown below are stored in D9015. B15·····B12 B11·····B8 B7·····B4 B3·····B0 CPU RUN/STOP switch: Remains unchanged in remote 0 RUN 1 STOP 2 PAUSE*1 3 STEP RUN Remote RUN/STOP by parameter setting 0 RUN 1 STOP 2 PAUSE*1 Status in program 0 Except below 1 STOP instruction execution Remote RUN/STOP by computer 0 RUN 1 STOP 2 PAUSE*1 *1 When M9040 is turned off when the CPU is in the RUN mode, the CPU remains in the RUN mode if changed to the PAUSE mode.
D9016	ROM/RAM setting	0: ROM 1: RAM 2: E2ROM	Indicates the setting for memory chop selection. Any of 0 to 2 is stored in BIN code.
	Program number	0: Main program (ROM) 1: Main program (RAM) 2: Subprogram (RAM)	Indicates which sequence program is run presently. Any of 0 to 2 is stored in BIN code. ("2" only for A3NCPU)
D9017	Scan time	Minimum scan time (10ms increments)	If scan time is smaller than the content of D9017, the value is newly stored at each END. Namely, the minimum value of scan time is stored into D9017 in BIN code.
D9018	Scan time	Scam time (10ms increments)	Scan time is stored in BIN code at each END and always rewritten. intervals of (set value) × 10ms.

Number	Name	Stored Data	Description
D9019	Scan time	Maximum scan time (10ms increments)	If scan time is larger than the content of D9019, the value is newly stored at each END. Namely, the maximum
*2 D9020	Constant scan	Constant scan time (Specified by user in 10ms increments)	Sets user program execution intervals in 10ms increments. 0: Constant scan function unused 1 to 200: Constant scan function used, program executed at value of scan time is stored into D9019 in BIN code.
*2 D9025	Clock data	Clock data (Year, month)	Stores the year (least significant two digits) and month in BCD. B15·····B12 B11·····B8 B7·····B4 B3·····B0 Year / Month Example: 1987, July
*2 D9026	Clock data	Clock data (Day, hour)	Stores the day and hour in BCD. B15·····B12 B11·····B8 B7·····B4 B3·····B0 Day / Hour Example: 31st, 10 o'clock
*2 D9027	Clock data	Clock data (Minute, second)	Stores the minute and second in BCD. B15·····B12 B11·····B8 B7·····B4 B3·····B0 Minute / Second Example: 35 minutes, 48 seconds
*2 D9028	Clock data	Clock data (, day of the week)	Stores the day of the week in BCD. B15·····B12 B11·····B8 B7·····B4 B3·····B0 0 must be set. Example: Friday H0005 Day of the week: 0 Sunday; 1 Monday; 2 Tuesday; 3 Wednesday; 4 Thursday; 5 Friday; 6 Saturday
D9044	For sampling trace	Step or time for sampling trace	The value that D9044 contains is used as a sampling trace condition when the sampling trace instruction STRA/STRAR is executed by switching on/off M9044 from the peripheral equipment. 0 for scan Time (in 10ms increments) for time The value is stored in BIN.

<table>
<tr><th>Number</th><th>Name</th><th>Stored Data</th><th>Description</th></tr>
<tr><td>*1 D9100</td><td rowspan="8">Fuse blown module</td><td rowspan="8">Bit pattern in units of 16 points of fuse blow modules</td><td rowspan="8">Output module numbers (in units of 16 points), of which fuses have blown, are entered in bit pattern. (Preset output number when parameter setting has been performed.)

<table>
<tr><th></th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>D9100</td><td>0</td><td>0</td><td>0</td><td>1 (YC0)</td><td>0</td><td>0</td><td>0</td><td>1 (YB0)</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>D9101</td><td>1 (Y1F0)</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1 (Y1A)</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>D9107</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1 (Y7B0)</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1 (Y730)</td><td>0</td><td>0</td><td>0</td></tr>
</table>
Indicates fuse blown.

(If normal status is restored, the bit pattern is not cleared. Therefore, it is necessary to clear the bit pattern by user program.)</td></tr>
<tr><td>*1 D9101</td></tr>
<tr><td>*1 D9102</td></tr>
<tr><td>*1 D9103</td></tr>
<tr><td>*1 D9104</td></tr>
<tr><td>*1 D9105</td></tr>
<tr><td>*1 D9106</td></tr>
<tr><td>*1 D9107</td></tr>
<tr><td>*1 D9116</td><td rowspan="8">I/O module verify error</td><td rowspan="8">Bit pattern in units of 16 points of verify error modules</td><td rowspan="8">When I/O module data different from those entered at power-on has been detected, the I/O module numbers (in units of 16 points) are entered in bit pattern. (Preset I/O module numbers when parameter setting has been performed.)

<table>
<tr><th></th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>D9116</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1 (X/Y0)</td></tr>
<tr><td>D9117</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1 (X/Y180)</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>D9123</td><td>0</td><td>1 (X/Y7E0)</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>
Indicated I/O units verify error.

(If normal status is restored, the bit pattern is not cleared. Therefore, it is necessary to clear the bit pattern by user program.)</td></tr>
<tr><td>*1 D9117</td></tr>
<tr><td>*1 D9118</td></tr>
<tr><td>*1 D9119</td></tr>
<tr><td>*1 D9120</td></tr>
<tr><td>*1 D9121</td></tr>
<tr><td>*1 D9122</td></tr>
<tr><td>*1 D9123</td></tr>
<tr><td>D9124</td><td>Annunciator detection quantity</td><td>Annunciator detection quantity</td><td>When one of F0 to 255 is turned on by OUT F or SET F, value 1 is added to the contents of D9124. When RST F or LED R instruction is executed, value 1 is subtracted from the contents of D9124. (This can also be performed by the indicator reset operation in the board information of the option board setting.) Quantity, which has been turned on by OUT F or SET F is stored; the value of D9124 is maximum 8.</td></tr>
</table>

Number	Name	Stored Data	Description
D9125 D9126 D9127 D9128 D9129 D9130 D9131 D9132	Annunciator detection number	Annunciator detection number	When one of F0 to 255 is turned on by OUT F or SET F, F number, which has turned on, is entered into D9125 to D9132 in due order. F number, which has been turned off by RST F, is erased from D9125 to D9132, and the contents of data registers succeeding the data register, where the erased F number was stored, are shifted to the preceding data registers. By executing LED R instruction, the contents of D9125 to D9132 are shifted upward by one. (This can also be performed by the indicator reset operation in the board information of the option board setting.) When there are 8 annunciator detections, the 9th one is not stored into D9125 to 9132 even if detected.

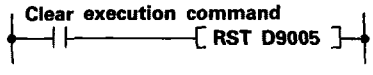
	SET F50	SET F25	SET F19	RET F25	SET F15	SET F70	SET F65	SET F38	SET F110	SET F151	SET F210	LED R	
D9009	0	50	50	50	50	50	50	50	50	50	50	50	99
D9124	0	1	2	3	2	3	4	5	6	7	8	8	8
D9125	0	50	50	50	50	50	50	50	50	50	50	50	99
D9126	0	0	25	25	99	99	99	99	99	99	99	99	15
D9127	0	0	0	99	0	15	15	15	15	15	15	15	70
D9128	0	0	0	0	0	0	70	70	70	70	70	70	65
D9129	0	0	0	0	0	0	0	65	65	65	65	65	38
D9130	0	0	0	0	0	0	0	0	38	38	38	38	110
D9131	0	0	0	0	0	0	0	0	0	110	110	110	151
D9132	0	0	0	0	0	0	0	0	0	0	151	151	210

POINT

(1) All special register data is cleared by any of the power-off, latch clear and reset operations. The data is retained when the RUN/STOP switch is set to STOP.

(2) For the above special registers marked *1, the contents or register are not cleared if normal status is restored. Therefore, to clear the contents, use the following method:

1) Method by user program
Insert the circuit shown at right into the program and turn on the clear execution command contact to clear the contents of register.



2) Method by peripheral equipmentSet the register to "0" changing the present value by the test function of peripheral equipment or set to "0" by forced reset. For the operation procedure, refer to the manual of each peripheral equipment.

3) By moving the RESET key switch at the CPU front to the RESET position, the special register is set to "0".

(3) Data is written to the special registers marked *2 by the sequence program.

APPENDIX 7 Special Link Relays and Registers

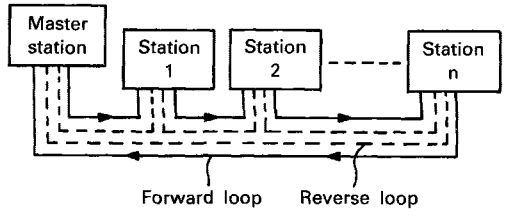
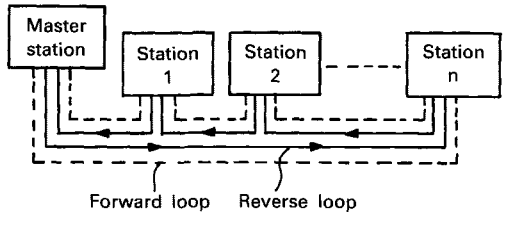
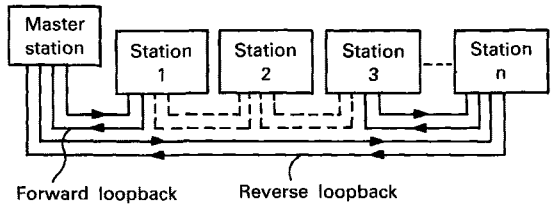
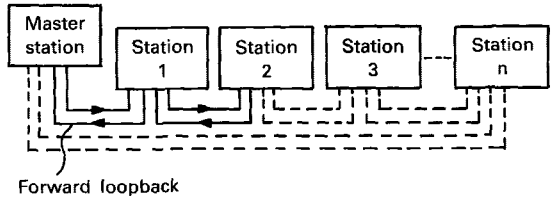
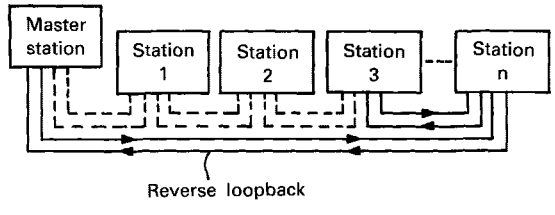
1) Link special relays only valid when the host is the master station

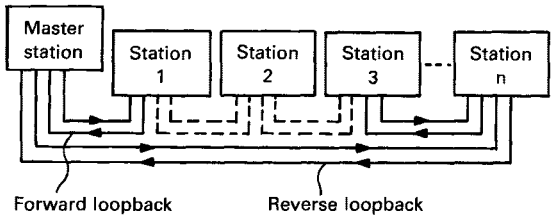
Device Number	Name	Description	
M9206	Link parameter error in the host	OFF : Normal ON : Error	Depends on whether or not the link parameter setting of the host is valid.
M9210	Link card error (master station)	OFF : Normal ON : Error	Depends on presence or absence of the link card hardware error. Judged by the CPU.
M9224	Link status	OFF : Offline ON : Online, interstation test, or loopback self-check	Depends on whether the master station is online or offline or is in interstation test or loopback self-check mode.
M9225	Forward loop error	OFF : Normal ON : Error	Depends on the error condition of the forward loop line.
M9226	Reverse loop error	OFF : Normal ON : Error	Depends on the error condition of the reverse loop line.
M9227	Loop test status	OFF : Unexecuted ON : Forward or reverse loop test being executed	Depends on whether or not the master station is executing a forward or a reverse loop test.
M9232	Local station operating status	OFF : RUN or STEP RUN mode ON : STOP or PAUSE mode	Depends on whether or not a local station is in STOP or PAUSE mode.
M9233	Local station error detect	OFF : No error ON : Error detected	Depends on whether or not a local station has detected an error in another station.
M9235	Local or remote I/O station parameter error detect	OFF : No error ON : Error detected	Depends on whether or not a local or a remote I/O station has detected any link parameter error in the master station.
M9236	Local or remote I/O station initial communicating status	OFF : Noncommunicating ON : Communicating	Depends on whether or not a local or a remote I/O station is communicating initial data (such as parameters) with the master station.
M9237	Local or remote I/O station error	OFF : Normal ON : Error	Depends on the error condition of a local or remote I/O station.
M9238	Local or remote I/O station forward/reverse loop error	OFF : Normal ON : Error	Depends on the error condition of the forward and reverse loop lines of a local or a remote I/O station.

2) Link special relays only valid when the host is a local station

Device Number	Name	Description	
M9211	Link card error (local station)	OFF : Normal ON : Error	Depends on presence or absence of the link card error. Judged by the CPU.
M9240	Link status	OFF : Online ON : Offline, interstation test, or loopback self-check	Depends on whether the local station is online or offline, or is in interstation test or loopback self-check mode.
M9241	Forward loop error	OFF : Normal ON : Error	Depends on the error condition of the forward loop line.
M9242	Reverse loop error	OFF : Normal ON : Error	Depends on the error condition of the reverse loop line.
M9243	Loopback execution	OFF : Non-executed ON : Executed	Depends on whether or not loopback is occurring at the local station.
M9246	Data unreceived	OFF : Received ON : Unreceived	Depends on whether or not data has been received from the master station.
M9247	Data unreceived	OFF : Received ON : Unreceived	Depends on on whether or not a tier three station has received data from its master station in a three-tier system.
M9250	Parameter unreceived	OFF : Received ON : Unreceived	Depends on whether or not link parameters have been received from the master station.
M9251	Link break	OFF : Normal ON : Break	Depends on the data link condition at the local station.
M9252	Loop test status	OFF : Unexecuted ON : Forward or reverse loop test is being executed.	Depends on whether or not the local station is executing a forward or a reverse loop test.
M9253	Master station operating status	OFF : RUN or STEP RUN mode ON : STOP or PAUSE mode	Depends on whether or not the master station is in STOP or PAUSE mode.
M9254	Operating status of other local stations	OFF : RUN or STEP RUN mode ON : STOP or PAUSE mode	Depends on whether or not a local station other than the host is in STOP or PAUSE mode.
M9255	Error status of other local stations	OFF : Normal ON : Error	Depends on whether or not a local station other than the host is in error.

1) Link special registers only valid when the host station is the master station

Device Number	Name	Description	
D9204	Link status	0 : Data link in forward loop 1 : Data link in reverse loop 2 : Loopback in forward/reverse direction 3 : Loopback in forward direction 4 : Loopback in reverse direction 5 : Data link impossible	Stores the present path status of the data link. Data link in forward loop  Data link in reverse loop  Loopback in forward/reverse loop  Loopback in forward loop only  Loopback in reverse loop only 

<table>
<tr><th>Device Number</th><th>Name</th><th colspan="2">Description</th></tr>
<tr><td>D9205</td><td>Loopback executing station</td><td>Station executing forward loopback</td><td rowspan="2">Stores the local or remote I/O station number at which loopback is being executed.
 

In the above example, 1 is stored into D9205 and 3 into D9206.
If data link returns to normal status (data link in forward loop), values in D9205 and D9206 remain 1 and 3.
Reset using sequence program or the RESET key.</td></tr>
<tr><td>D9206</td><td>Loopback executing station</td><td>Station executing reverse loopback</td></tr>
<tr><td>D9207</td><td>Link scan time</td><td>Maximum value</td><td rowspan="3">Stores the data link processing time with all local and remote I/O stations.
Input (X), output (Y), link relay (B), and link register (W), assigned in link parameters, communicate with the corresponding stations every link scan.
Link scan is a period of time during which data link is executed with all connected slave stations, independently of the sequence program scan time.</td></tr>
<tr><td>D9208</td><td>Link scan time</td><td>Minimum value</td></tr>
<tr><td>D9209</td><td>Link scan time</td><td>Present value</td></tr>
<tr><td>D9210</td><td>Retry count</td><td>Total number stored</td><td>Stores the number of retry times due to transmission error.
Count stops at a maximum of "FFFFH".
RESET to return the count to 0.</td></tr>
<tr><td>D9211</td><td>Loop switching count</td><td>Total number stored</td><td>Stores the number of times the loop line has been switched to reverse loop or loopback.</td></tr>
<tr><td>D9212</td><td>Local station operating status</td><td>Stores the status of stations 1 to 16</td><td rowspan="4">Stores the local station numbers which are in STOP or PAUSE mode.
<table><tr><th rowspan="2">Device Number</th><th colspan="16">Bit</th></tr><tr><th>b15</th><th>b14</th><th>b13</th><th>b12</th><th>b11</th><th>b10</th><th>b9</th><th>b8</th><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>b0</th></tr><tr><td>D9212</td><td>L16</td><td>L15</td><td>L14</td><td>L13</td><td>L12</td><td>L11</td><td>L10</td><td>L9</td><td>L8</td><td>L7</td><td>L6</td><td>L5</td><td>L4</td><td>L3</td><td>L2</td><td>L1</td></tr><tr><td>D9213</td><td>L32</td><td>L31</td><td>L30</td><td>L29</td><td>L28</td><td>L27</td><td>L26</td><td>L25</td><td>L24</td><td>L23</td><td>L22</td><td>L21</td><td>L20</td><td>L19</td><td>L18</td><td>L17</td></tr><tr><td>D9214</td><td>L48</td><td>L47</td><td>L46</td><td>L45</td><td>L44</td><td>L43</td><td>L42</td><td>L41</td><td>L40</td><td>L39</td><td>L38</td><td>L37</td><td>L36</td><td>L35</td><td>L34</td><td>L33</td></tr><tr><td>D9215</td><td>L64</td><td>L63</td><td>L62</td><td>L61</td><td>L60</td><td>L59</td><td>L58</td><td>L57</td><td>L56</td><td>L55</td><td>L54</td><td>L53</td><td>L52</td><td>L51</td><td>L50</td><td>L49</td></tr></table>
When a local station is switched to STOP or PAUSE mode, the bit corresponding to the station number in the register becomes "1".
Example: When station 7 switches to STOP mode, bit 6 in D9212 becomes "1", and when D9212 is monitored, its value is "64 (40H)".</td></tr>
<tr><td>D9213</td><td>Local station operating status</td><td>Stores the status of stations 17 to 32</td></tr>
<tr><td>D9214</td><td>Local station operating status</td><td>Stores the status of stations 33 to 48</td></tr>
<tr><td>D9215</td><td>Local station operating status</td><td>Stores the status of stations 49 to 64</td></tr>
</table>

Device Number	Name	Description	
D9216	Local station error detection	Stores the status of stations 1 to 16	Stores the local station numbers which are in error.
D9217	Local station error detection	Stores the status of stations 17 to 32	
D9218	Local station error detection	Stores the status of stations 33 to 48	
D9219	Local station error detection	Stores the status of stations 49 to 64	
D9220	Local station parameter mismatched or remote station I/O assignment error	Stores the status of stations 1 to 16	Stores the local station numbers which contain mismatched parameters or of remote station numbers for which incorrect I/O assignment has been made.
D9221	Local station parameter mismatched or remote station I/O assignment error	Stores the status of stations 17 to 32	
D9222	Local station parameter mismatched or remote station I/O assignment error	Stores the status of stations 33 to 48	
D9223	Local station parameter mismatched or remote station I/O assignment error	Stores the status of stations 49 to 64	
D9224	Initial communication between local or remote I/O stations	Stores the status of stations 1 to 16	Stores the local or remote station numbers while they are communicating the initial data with their relevant master station.
D9225	Initial communication between local or remote I/O stations	Stores the status of stations 17 to 32	
D9226	Initial communication between local or remote I/O stations	Stores the status of stations 33 to 48	
D9227	Initial communication between local or remote I/O stations	Stores the status of stations 49 to 64	

Device Number	Bit															
	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
D9216	L16	L15	L14	L13	L12	L11	L10	L9	L8	L7	L6	L5	L4	L3	L2	L1
D9217	L32	L31	L30	L29	L28	L27	L26	L25	L24	L23	L22	L21	L20	L19	L18	L17
D9218	L48	L47	L46	L45	L44	L43	L42	L41	L40	L39	L38	L37	L36	L35	L34	L33
D9219	L64	L63	L62	L61	L60	L59	L58	L57	L56	L55	L54	L53	L52	L51	L50	L49

If a local station detects an error, the bit corresponding to the station number becomes "1".
Example: When station 6 and 12 detect an error, bits 5 and 11 in D9216 become "1", and when D9216 is monitored, its value is "2080 (820H)".

Device Number	Bit															
	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
D9220	L/R 16	L/R 15	L/R 14	L/R 13	L/R 12	L/R 11	L/R 10	L/R 9	L/R 8	L/R 7	L/R 6	L/R 5	L/R 4	L/R 3	L/R 2	L/R 1
D9221	L/R 32	L/R 31	L/R 30	L/R 29	L/R 28	L/R 27	L/R 26	L/R 25	L/R 24	L/R 23	L/R 22	L/R 21	L/R 20	L/R 19	L/R 18	L/R 17
D9222	L/R 48	L/R 47	L/R 46	L/R 45	L/R 44	L/R 43	L/R 42	L/R 41	L/R 40	L/R 39	L/R 38	L/R 37	L/R 36	L/R 35	L/R 34	L/R 33
D9223	L/R 64	L/R 63	L/R 62	L/R 61	L/R 60	L/R 59	L/R 58	L/R 57	L/R 56	L/R 55	L/R 54	L/R 53	L/R 52	L/R 51	L/R 50	L/R 49

If a local station acting as the master station of tier three detects a parameter error or a remote station contains an invalid I/O assignment, the bit corresponding to the station number becomes "1".
Example: When local station 5 and remote I/O station 14 detect an error, bits 4 and 13 in D9220 become "1", and when D9220 is monitored, its value is "8208 (2010H)".

Device Number	Bit															
	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
D9224	L/R 16	L/R 15	L/R 14	L/R 13	L/R 12	L/R 11	L/R 10	L/R 9	L/R 8	L/R 7	L/R 6	L/R 5	L/R 4	L/R 3	L/R 2	L/R 1
D9225	L/R 32	L/R 31	L/R 30	L/R 29	L/R 28	L/R 27	L/R 26	L/R 25	L/R 24	L/R 23	L/R 22	L/R 21	L/R 20	L/R 19	L/R 18	L/R 17
D9226	L/R 48	L/R 47	L/R 46	L/R 45	L/R 44	L/R 43	L/R 42	L/R 41	L/R 40	L/R 39	L/R 38	L/R 37	L/R 36	L/R 35	L/R 34	L/R 33
D9227	L/R 64	L/R 63	L/R 62	L/R 61	L/R 60	L/R 59	L/R 58	L/R 57	L/R 56	L/R 55	L/R 54	L/R 53	L/R 52	L/R 51	L/R 50	L/R 49

The bit corresponding to the station number which is currently communicating the initial settings becomes "1".
Example: When stations 23 and 45 are communicating, bit 6 of D9225 and bit 12 of D9226 become "1", and when D9225 is monitored, its value is "64 (40H)", and when D9226 is monitored, its value is "4096 (1000H)".

<table>
<tr><th>Device Number</th><th>Name</th><th colspan="2">Description</th></tr>
<tr><td>D9228</td><td>Local or remote I/O station error</td><td>Stores the status of stations 1 to 16</td><td rowspan="4">Stores the local or remote station numbers which are in error.
<table>
<tr><th rowspan="2">Device Number</th><th colspan="16">Bit</th></tr>
<tr><th>b15</th><th>b14</th><th>b13</th><th>b12</th><th>b11</th><th>b10</th><th>b9</th><th>b8</th><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>b0</th></tr>
<tr><td>D9228</td><td>L/R 16</td><td>L/R 15</td><td>L/R 14</td><td>L/R 13</td><td>L/R 12</td><td>L/R 11</td><td>L/R 10</td><td>L/R 9</td><td>L/R 8</td><td>L/R 7</td><td>L/R 6</td><td>L/R 5</td><td>L/R 4</td><td>L/R 3</td><td>L/R 2</td><td>L/R 1</td></tr>
<tr><td>D9229</td><td>L/R 32</td><td>L/R 31</td><td>L/R 30</td><td>L/R 29</td><td>L/R 28</td><td>L/R 27</td><td>L/R 26</td><td>L/R 25</td><td>L/R 24</td><td>L/R 23</td><td>L/R 22</td><td>L/R 21</td><td>L/R 20</td><td>L/R 19</td><td>L/R 18</td><td>L/R 17</td></tr>
<tr><td>D9230</td><td>L/R 48</td><td>L/R 47</td><td>L/R 46</td><td>L/R 45</td><td>L/R 44</td><td>L/R 43</td><td>L/R 42</td><td>L/R 41</td><td>L/R 40</td><td>L/R 39</td><td>L/R 38</td><td>L/R 37</td><td>L/R 36</td><td>L/R 35</td><td>L/R 34</td><td>L/R 33</td></tr>
<tr><td>D9231</td><td>L/R 64</td><td>L/R 63</td><td>L/R 62</td><td>L/R 61</td><td>L/R 60</td><td>L/R 59</td><td>L/R 58</td><td>L/R 57</td><td>L/R 56</td><td>L/R 55</td><td>L/R 54</td><td>L/R 53</td><td>L/R 52</td><td>L/R 51</td><td>L/R 50</td><td>L/R 49</td></tr>
</table>
The bit corresponding to the station number with the error becomes "1".
Example: When local station 3 and remote I/O station 14 have an error, bits 2 and 13 of D9228 become "1", and when D9228 is monitored, its value is "8196 (2004H)".</td></tr>
<tr><td>D9229</td><td>Local or remote I/O station error</td><td>Stores the status of stations 17 to 32</td></tr>
<tr><td>D9230</td><td>Local or remote I/O station error</td><td>Stores the status of stations 33 to 48</td></tr>
<tr><td>D9231</td><td>Local or remote I/O station error</td><td>Stores the status of stations 49 to 64</td></tr>
<tr><td>D9232</td><td>Local or remote I/O station loop error</td><td>Stores the status of stations 1 to 8</td><td rowspan="8">Stores the local or remote station number at which a forward or reverse loop error has occurred.
<table>
<tr><th rowspan="2">Device Number</th><th colspan="16">Bit</th></tr>
<tr><th>b15</th><th>b14</th><th>b13</th><th>b12</th><th>b11</th><th>b10</th><th>b9</th><th>b8</th><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>b0</th></tr>
<tr><td rowspan="2">D9232</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td></tr>
<tr><td colspan="2">L8</td><td colspan="2">L7</td><td colspan="2">L6</td><td colspan="2">L5</td><td colspan="2">L4</td><td colspan="2">L3</td><td colspan="2">L2</td><td colspan="2">L1</td></tr>
<tr><td rowspan="2">D9233</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td></tr>
<tr><td colspan="2">L/R16</td><td colspan="2">L15</td><td colspan="2">L14</td><td colspan="2">L13</td><td colspan="2">L12</td><td colspan="2">L11</td><td colspan="2">L10</td><td colspan="2">L9</td></tr>
<tr><td rowspan="2">D9234</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td></tr>
<tr><td colspan="2">L24</td><td colspan="2">L23</td><td colspan="2">L22</td><td colspan="2">L21</td><td colspan="2">L20</td><td colspan="2">L19</td><td colspan="2">L18</td><td colspan="2">L17</td></tr>
<tr><td rowspan="2">D9235</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td></tr>
<tr><td colspan="2">L32</td><td colspan="2">L31</td><td colspan="2">L30</td><td colspan="2">L29</td><td colspan="2">L28</td><td colspan="2">L27</td><td colspan="2">L26</td><td colspan="2">L25</td></tr>
<tr><td rowspan="2">D9236</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td></tr>
<tr><td colspan="2">L40</td><td colspan="2">L39</td><td colspan="2">L38</td><td colspan="2">L37</td><td colspan="2">L36</td><td colspan="2">L35</td><td colspan="2">L34</td><td colspan="2">L33</td></tr>
<tr><td rowspan="2">D9237</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td></tr>
<tr><td colspan="2">L48</td><td colspan="2">L47</td><td colspan="2">L46</td><td colspan="2">L45</td><td colspan="2">L44</td><td colspan="2">L43</td><td colspan="2">L42</td><td colspan="2">L41</td></tr>
<tr><td rowspan="2">D9238</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td></tr>
<tr><td colspan="2">L56</td><td colspan="2">L55</td><td colspan="2">L54</td><td colspan="2">L53</td><td colspan="2">L52</td><td colspan="2">L51</td><td colspan="2">L50</td><td colspan="2">L49</td></tr>
<tr><td rowspan="2">D9239</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td><td>R</td><td>F</td></tr>
<tr><td colspan="2">L64</td><td colspan="2">L63</td><td colspan="2">L62</td><td colspan="2">L61</td><td colspan="2">L60</td><td colspan="2">L59</td><td colspan="2">L58</td><td colspan="2">L57</td></tr>
</table>
In the above table, "F" indicates a forward loop line and "R" a reverse loop line. The bit corresponding to the station number at which the forward or reverse loop error has occurred, becomes "1".
Example: When the forward loop line of station 5 has an error, bit 8 of D9232 become "1", and when D9232 is monitored, its value is "256 (100H)".</td></tr>
<tr><td>D9233</td><td>Local or remote I/O station loop error</td><td>Stores the status of stations 9 to 16</td></tr>
<tr><td>D9234</td><td>Local or remote I/O station loop error</td><td>Stores the status of stations 17 to 24</td></tr>
<tr><td>D9235</td><td>Local or remote I/O station loop error</td><td>Stores the status of stations 25 to 32</td></tr>
<tr><td>D9236</td><td>Local or remote I/O station loop error</td><td>Stores the status of stations 33 to 40</td></tr>
<tr><td>D9237</td><td>Local or remote I/O station loop error</td><td>Stores the status of stations 41 to 48</td></tr>
<tr><td>D9238</td><td>Local or remote I/O station loop error</td><td>Stores the status of stations 49 to 56</td></tr>
<tr><td>D9239</td><td>Local or remote I/O station loop error</td><td>Stores the status of stations 57 to 64</td></tr>
<tr><td>D9240*</td><td>Number of receive error detection times</td><td>Total number stored</td><td>Stores the number of times the following transmission errors have been detected:
CRC, OVER, AB.IF
Count is made to a maximum of FFFFH. RESET to return the count to 0.</td></tr>
</table>

2) Link special registers only valid when the host station is a local station

Device Number	Name	Description	
D9243	Own station number check	Stores a station number (0 to 64)	Allows a local station to confirm its own station number.
D9244	Total number of slave stations	Stores the number of slave stations	Indicates the number of slave stations in one loop.
D9245*	Number of receive error detection times	Total number stored	Stores the number of times the following transmission errors have been detected: CRC, OVER, AB.IF Count is made to a maximum of FFFFH. RESET to return the count to 0.
D9248	Local station operating status	Stores the status of stations 1 to 16	Stores the local station number which is in STOP or PAUSE mode.
D9249	Local station operating status	Stores the status of stations 17 to 32	
D9250	Local station operating status	Stores the status of stations 33 to 48	
D9251	Local station operating status	Stores the status of stations 49 to 64	

Device Number	Bit															
	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
D9248	L16	L15	L14	L13	L12	L11	L10	L9	L8	L7	L6	L5	L4	L3	L2	L1
D9249	L32	L31	L30	L29	L28	L27	L26	L25	L24	L23	L22	L21	L20	L19	L18	L17
D9250	L48	L47	L46	L45	L44	L43	L42	L41	L40	L39	L38	L37	L36	L35	L34	L33
D9251	L64	L63	L62	L61	L60	L59	L58	L57	L56	L55	L54	L53	L52	L51	L50	L49

The bit corresponding to the station number which is in STOP or PAUSE mode, becomes "1".
Example: When local stations 7 and 15 are in STOP mode, bits 6 and 14 of D9248 become "1", and when D9248 is monitored, its value is "16448 (4040H)".

Device Number	Name	Description	
D9252	Local station error	Stores the status of stations 1 to 16	Stores the local station number other than the host, which is in error.
D9253	Local station error	Stores the status of stations 17 to 32	
D9254	Local station error	Stores the status of stations 33 to 48	
D9255	Local station error	Stores the status of stations 49 to 64	

Device Number	Bit															
	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
D9252	L16	L15	L14	L13	L12	L11	L10	L9	L8	L7	L6	L5	L4	L3	L2	L1
D9253	L32	L31	L30	L29	L28	L27	L26	L25	L24	L23	L22	L21	L20	L19	L18	L17
D9254	L48	L47	L46	L45	L44	L43	L42	L41	L40	L39	L38	L37	L36	L35	L34	L33
D9255	L64	L63	L62	L61	L60	L59	L58	L57	L56	L55	L54	L53	L52	L51	L50	L49

The bit corresponding to the station number which is in error, becomes "1".
Example: When local station 12 is in error, bit 11 of D9252 becomes "1", and when D9252 is monitored, its value is "2048 (800H)".

APPENDIX 8 A-CPU Device Memory Map

The data memory area (8000H to 9FFFH) stores device data. The memory area of each device and its configuration are as indicated below.

4.4.1

Device	CPU Type	Address	
Input (X)	A1 A1(E) A1N	8000H to 803FH	X0 to FF
	A2 A2(E) A2N A2C A0J2H	8000H to 807FH	X0 to 1FF
	A3N	8000H to 81FFH	X0 to 7FF

Configuration

	Odd address								Even address							
	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
8000H	XIM7	XIM6	XIM5	XIM4	XIM3	XIM2	XIM1	XIM0	X7	X6	X5	X4	X3	X2	X1	X0
8002H	XIMF	XIME	XIMD	XIMC	XIMB	XIMA	XIM9	XIM8	XF	XE	XD	XC	XB	XA	X9	X8
8004H	XIM17	XIM16	XIM15	XIM14	XIM13	XIM12	XIM11	XIM10	X17	X16	X15	X14	X13	X12	X11	X10

- Used for storing ON/OFF data from remote station and allows read/write.
- Stored data area as follows:
  0: OFF
  1: ON

- Used for storing ON/OFF data from input unit and allows only read.
- Stored data area as follows:
  0: ON
  1: OFF

Obtain actual input by the following expression:

Input (X) = (XIM) $\vee$ $\overline{(X)}$

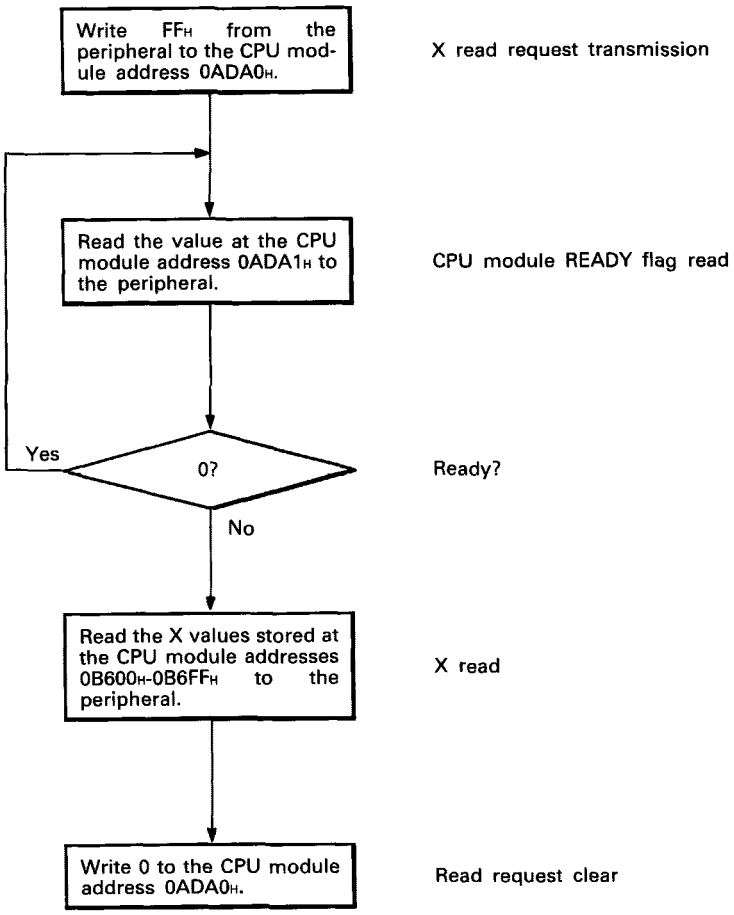
Device	CPU Type	Address	Configuration
Input (X)	A3 A3(E)	B600H to B6FFH: X0 to 7FF	

Odd address / Even address

	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
B600H	XF	XE	XD	XC	XB	XA	X9	X8	X7	X6	X5	X4	X3	X2	X1	X0
B602H	X1F	X1E	X1D	X1C	X1B	X1A	X19	X18	X17	X16	X15	X14	X13	X12	X11	X10
B604H	X2F	X2E	X2D	X2C	X2B	X2A	X29	X28	X27	X26	X25	X24	X23	X22	X21	X20

- Stores ON/OFF data from an input unit, read only.
- 0 indicates ON and 1 OFF.

Input X reading procedure (A3, A3ECPU only)



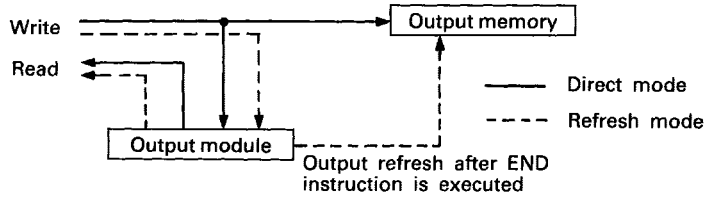
Device	CPU Type	Address	Configuration
Output (Y)	A1 A1(E) A1N	8200H to 823FH Y0 to FF	
	A2 A2(E) A2N A2C A0J2H	8200H to 827FH Y0 to 1FF	
	A3 A3(E) A3N	8200H to 83FFH Y0 to 7FF	

Odd address / Even address

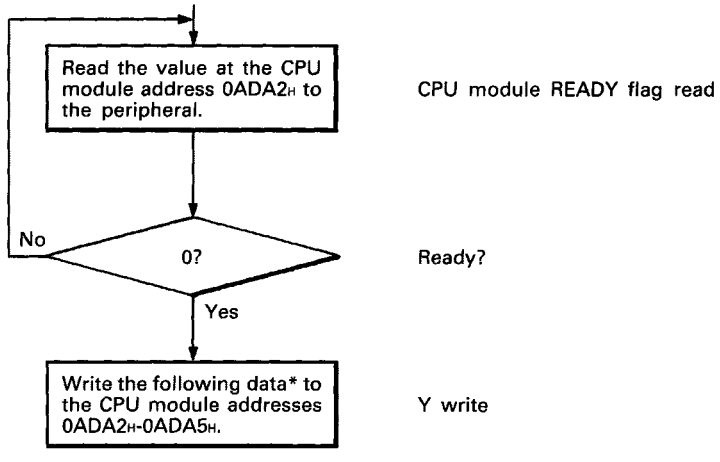
	B15 ··· B8	B7	B6	B5	B4	B3	B2	B1	B0
8200H		Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
8202H		YF	YE	YD	YC	YB	YA	Y9	Y8
8204H		Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10

- Used for storing operation result of PC and allows read/write.
- Stored data are as follows:
  0: OFF
  1: ON

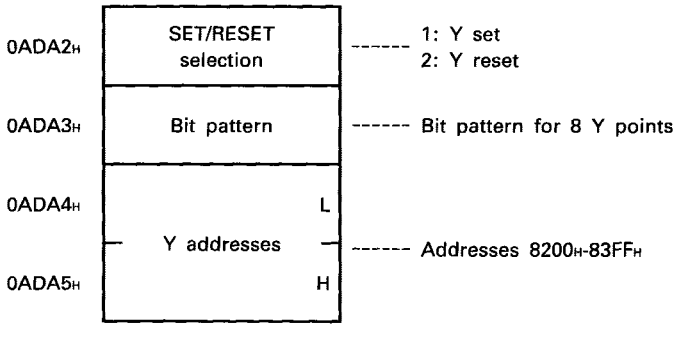
Read/write from/to output memory are performed as shown below:



Output Y writing procedure (A3, A3ECPU only)



*The data written is as follows:



Device	CPU Type	Address		Configuration
Internal relay (M) Latch relay (L) Step relay (S)	A1 A2 A3 A1N A2N A3N	8400H to 85FFH	M/L/S 0 to 2047	
Link relay (B)		8600H to 86FFH	B0 to 3FF	
Annunci-ator (F)		8700H to 873FH	F0 to 255	
Special relay (M)		8740H to 877FH	M9000 to 9255	
Timer (T) contact	A1 A2 A3 A1N A2N A3N	8780H to 87BFH	T0 to 255	
Counter (C) contact		87C0H to 87FFH	C0 to 255	
Timer (T) coil		9C00H to 9C3FH	T0 to 255	
Counter (C) coil		9C40H to 9C7FH	C0 to 255	

All devices are in one bit locations and store device ON/OFF data using eight bits at seven addresses.

0 indicates OFF and 1 ON.

Example M0 to 23 are as follos:

	Odd area								Even area							
	B15 ·····							B8	B7	B6	B5	B4	B3	B2	B1	B0
8400H									M7	M6	M5	M4	M3	M2	M1	M0
8402H									M15	M14	M13	M12	M11	M10	M9	M8
8404H									M23	M22	M21	M20	M19	M18	M17	M16

⇩

Stores PC operation results and allows read/write.

Device	CPU Type	Address		Configuration
Data register (D)	A1 A2 A3 A1N A2N A3N	8800H to 8FFFH	D0 to 1023	All devices consist are in 2 byte (16 bit) locations. Example: D0 configuration is as follows: B7 B0 8800H (L) 8801H (H) B15 B8
Link register (W)		9000H to 97FFH	W0 to 3FF	
Timer (T) present value		9800H to 99FFH	T0 to 255	
Counter (C) present value		9A00H to 9BFFH	C0 to 255	
Special register (D)		9D00H to 9EFFH	D9000 to 9255	
Accumu-lator (A0, 1)		9FF8H 9FFAH	A0 A1	
Index (Z, V)		9FFCH 9FFEH	Z V	

Device	CPU Type	Address	Configuration

Input (X) — CPU Type: A3H, A3M

Address: 8000H to 80FFH — X0 to 7FF

	Odd address								Even address							
	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
8000H	XF	XE	XD	XC	XB	XA	X9	X8	X7	X6	X5	X4	X3	X2	X1	X0
8002H	X1F	X1E	X1D	X1C	X1B	X1A	X19	X18	X17	X16	X15	X14	X13	X12	X11	X10
8004H	X2F	X2E	X2D	X2C	X2B	X2A	X29	X28	X27	X26	X25	X24	X23	X22	X21	X20

Stores ON/OFF data from input module, read only.
0 indicates OFF and 1 ON.

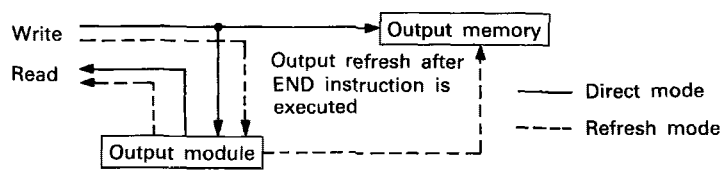
Output (Y) — CPU Type: A3H, A3M

Address: 8000H to 80FFH — Y0 to 7FF

	Odd address								Even address							
	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
8200H	YF	YE	YD	YC	YB	YA	Y9	Y8	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
8202H	Y1F	Y1E	Y1D	Y1C	Y1B	Y1A	Y19	Y18	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10
8204H	Y2F	Y2E	Y2D	Y2C	Y2B	Y2A	Y29	Y28	Y27	Y26	Y25	Y24	Y23	Y22	Y21	Y20

Stores PC operation results and allows read/write.
0 indicates OFF and 1 ON.

The output memory is accessed as shown below?



Internal relay (M) / Latch *relay (L)* / Step relay (S)

Address: 8400H to 84FFH — M/L/S *0 to 2047*

Link relay (B)

Address: 8500H to 867FH — B0 to 3FF

Annunciator (F)

Address: 8700H to 871FH — F0 to 255

Stores device ON/OFF data in one bit locations.
0 indicates OFF and 1 ON.

Example: M0 to 47 are as follows:

	Odd address								Even address							
	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
8400H	M15	M14	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0
8402H	M31	M30	M29	M28	M27	M26	M25	M24	M23	M22	M21	M20	M19	M18	M17	M16
8404H	M47	M46	M45	M44	M43	M42	M41	M40	M39	M38	M37	M36	M35	M34	M33	M32

Stores PC operation results and allows read/write.

Device	CPU Type	Address	Configuration
Special relay (M)	A3H A3M	8740H to 875FH: M9000 to 9255	
Timer (T) contact		8780H to 879FH: T0 to 255	Stores device ON/OFF data in one bit locations. 0 indicates OFF and 1 ON.
Counter (C) contact		87C0H to 87DFH: C0 to 255	Example: M0 to 47 are as follows:
Timer (T) coil		9C00H to 9C1FH: T0 to 255	
Counter (C) coil		9C40H to 9C5FH: C0 to 255	

Odd address / Even address

	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
8400H	M15	M14	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0
8402H	M31	M30	M29	M28	M27	M26	M25	M24	M23	M22	M21	M20	M19	M18	M17	M16
8404H	M47	M46	M45	M44	M43	M42	M41	M40	M39	M38	M37	M36	M35	M34	M33	M32

Stores PC operation results and allows read/write.

Device	CPU Type	Address	Configuration
Data register (D)	A3H A3M	8800H to 8FFFH: D0 to 1023	All devices are in 2 byte (16 bit) locations. Example: D0 configuration is as follows: B7 … B0 8800H (L) 8801H (H) B15 … B8
Link register (W)		9000H to 97FFH: W0 to 3FF	
Timer (T) present value		9800H to 99FFH: T0 to 255	
Counter (C) present value		9A00H to 9BFFH: C0 to 255	
Special register (D)		9D00H to 9EFFH: D9000 to 9255	
Accumulator (A0, 1)		9FF8H: A0 9FFAH: A1	
Index (Z, V)		9FFCH: Z 9FFEH: V	

Device	CPU Type	Address	Configuration
Input (X)	A0J2	Read address 6000H to 6030H X0 to 1DF Write address 6080H to 603CH X0 to 1DF	Read address Write address
Output (Y)	A0J2	6100H to 6130H Y0 to 1DF	Odd address / Even address
Internal relay (M) Latch relay (L) Step relay (S)	A0J2	6200H to 62FFH M/L/S 0 to 2047	Stores device ON/OFF data in one bit locations. 0 indicates OFF and 1 ON. Example: M0 to 47 are as follows: Odd address / Even address
Link relay (B)	A0J2	6300H to 637FH B0 to 3FF	
Annuci-ator (F)	A0J2	6380H to 639FH F0 to 255	

Read address

	B7	B6	B5	B4	B3	B2	B1	B0
6000H	X7	X6	X5	X4	X3	X2	X1	X0
6001H	XF	XE	XD	XC	XB	XA	X9	X8
6002H	X17	X16	X15	X14	X13	X12	X11	X10

Stores ON/OFF data from input unit, read only.
0 indicates ON and 1 OFF.

Write address

	B7	B6	B5	B4	B3	B2	B1	B0
6080H	X7	X6	X5	X4	X3	X2	X1	X0
6081H	XF	XE	XD	XC	XB	XA	X9	X8
6082H	X17	X16	X15	X14	X13	X12	X11	X10

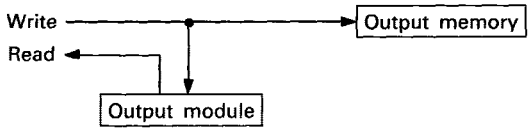
Allows ON/OFF data to be written to remote station.
0 indicates OFF and 1 ON.

Odd address / Even address

	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
6100H	YF	YE	YD	YC	YB	YA	Y9	Y8	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
6102H	Y1F	Y1E	Y1D	Y1C	Y1B	Y1A	Y19	Y18	Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10
6104H	Y2F	Y2E	Y2D	Y2C	Y2B	Y2A	Y29	Y28	Y27	Y26	Y25	Y24	Y23	Y22	Y21	Y20

Stores PC operation results and allows read/write.
0 indicates OFF and 1 ON.

The output memory is accessed as shown below:



Odd address / Even address

	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
6200H	M15	M14	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0
6202H	M31	M30	M29	M28	M27	M26	M25	M24	M23	M22	M21	M20	M19	M18	M17	M16
6204H	M47	M46	M45	M44	M43	M42	M41	M40	M39	M38	M37	M36	M35	M34	M33	M32

Stores PC operation results and allows read/write.

Device	CPU Type	Address	Configuration
Special relay (M)	A0J2	65A0H to 63BFH: M3000 to 9255	○ Stores ON/OFF data in one bit locations. ○ 0 indicates OFF and 1 ON.
Timer (T) contact		63C0H to 63CFH: T0 to 127	
Counter (C) contact		63E0H to 63EFH: C0 to 127	
Timer (T) coil		6400H to 640FH: T0 to 127	
Counter (C) coil		6420H to 642FH: C0 to 127	
Data register (D)		6500H to 68FFH: D0 to 511	All devices are in 2 byte (16 bit) locations.
Link register (W)		C000H to C7FFH: W0 to 3FF	
Timer (T) present value		6A00H to 6AFFH: T0 to 127	
Counter (C) present value		6B00H to 6BFFH: C0 to 127	
Special register (D)		6900H to 69FFH: D9000 to 9127	

Example: M9000 to M9047 are as follows:

	Odd address								Even address							
	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
63A0H	M9015	M9014	M9013	M9012	M9011	M9010	M9009	M9008	M9007	M9006	M9005	M9004	M9003	M9002	M9001	M9000
6342H	M9031	M9030	M9029	M9028	M9027	M9026	M9025	M9024	M9023	M9022	M9021	M9020	M9019	M9018	M9017	M9016
63A4H	M9047	M9046	M9045	M9044	M9043	M9042	M9041	M9040	M9039	M9038	M9037	M9036	M9035	M9034	M9033	M9032

Stores PC operation results and allows read/write.

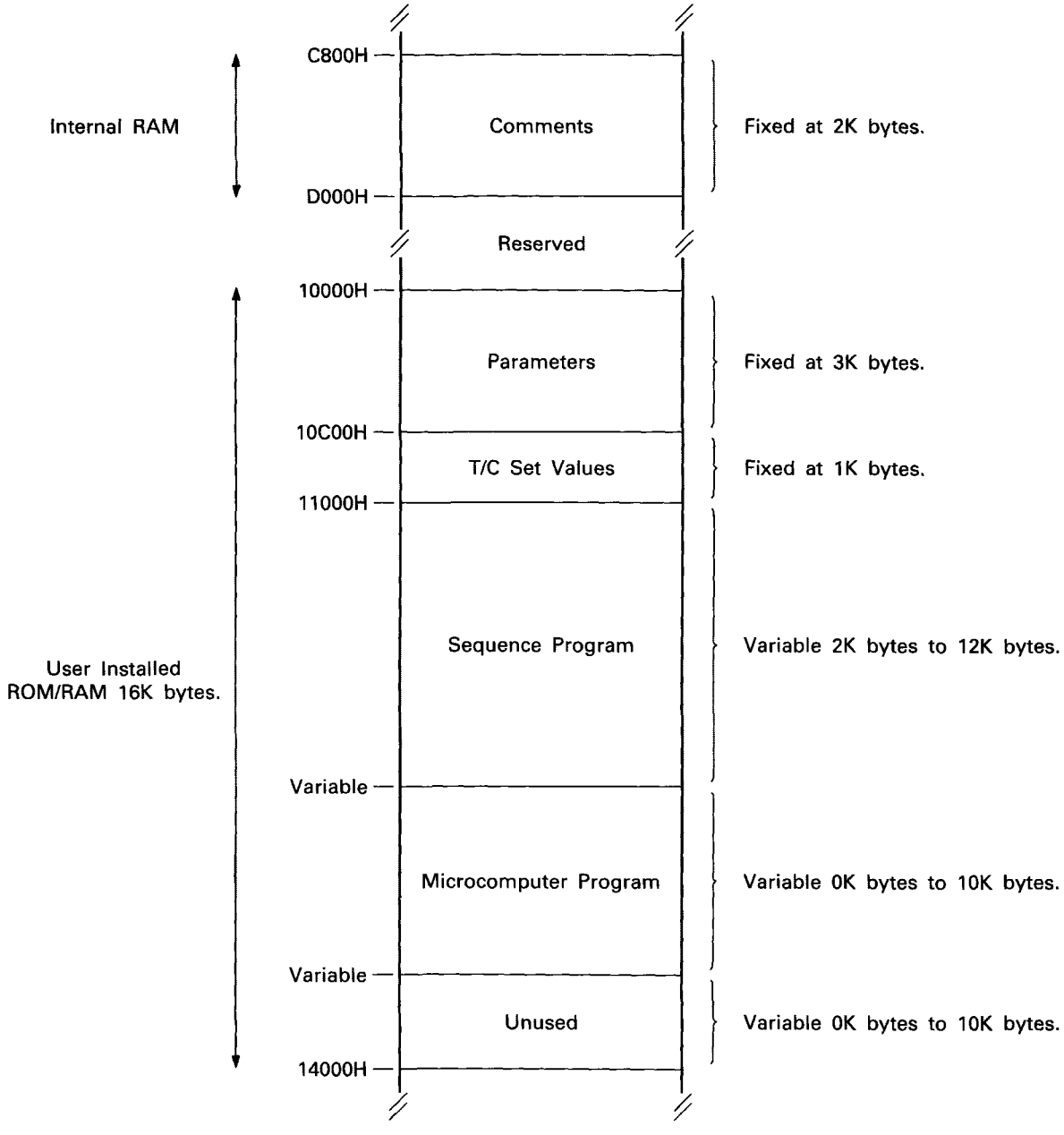
Example: D0 configuration is as follows:

	B7 … B0	
6500H	(L)	
6501H	(H)	
	B15 … B8	

Device	CPU Type	Address	Configuration
Index (Z, V)	A0J2	64FCH Z 64FEH V	

APPENDIX 9 A-CPU Memory Map — User Areas

User Installed Memory Map A1, A1E, A1N CPU RAM/ROM OPERATION

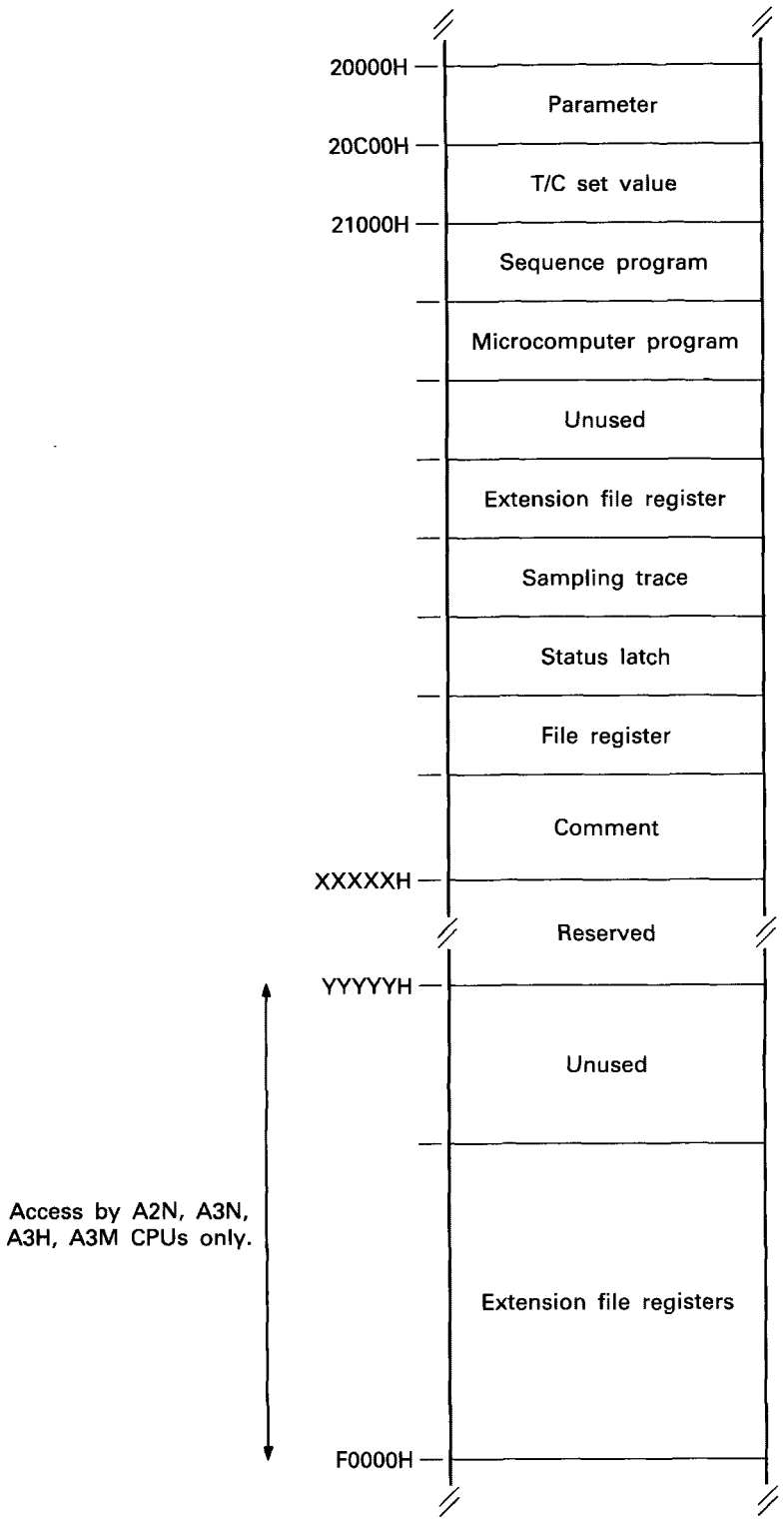


The installed memory head address remains at 10,000H for both ROM and RAM operation.

The head address of the sequence program area is fixed at 11,000H.

The head address of the Microcomputer Program and Unused areas are variable, but may be calculated from the memory parameter settings.

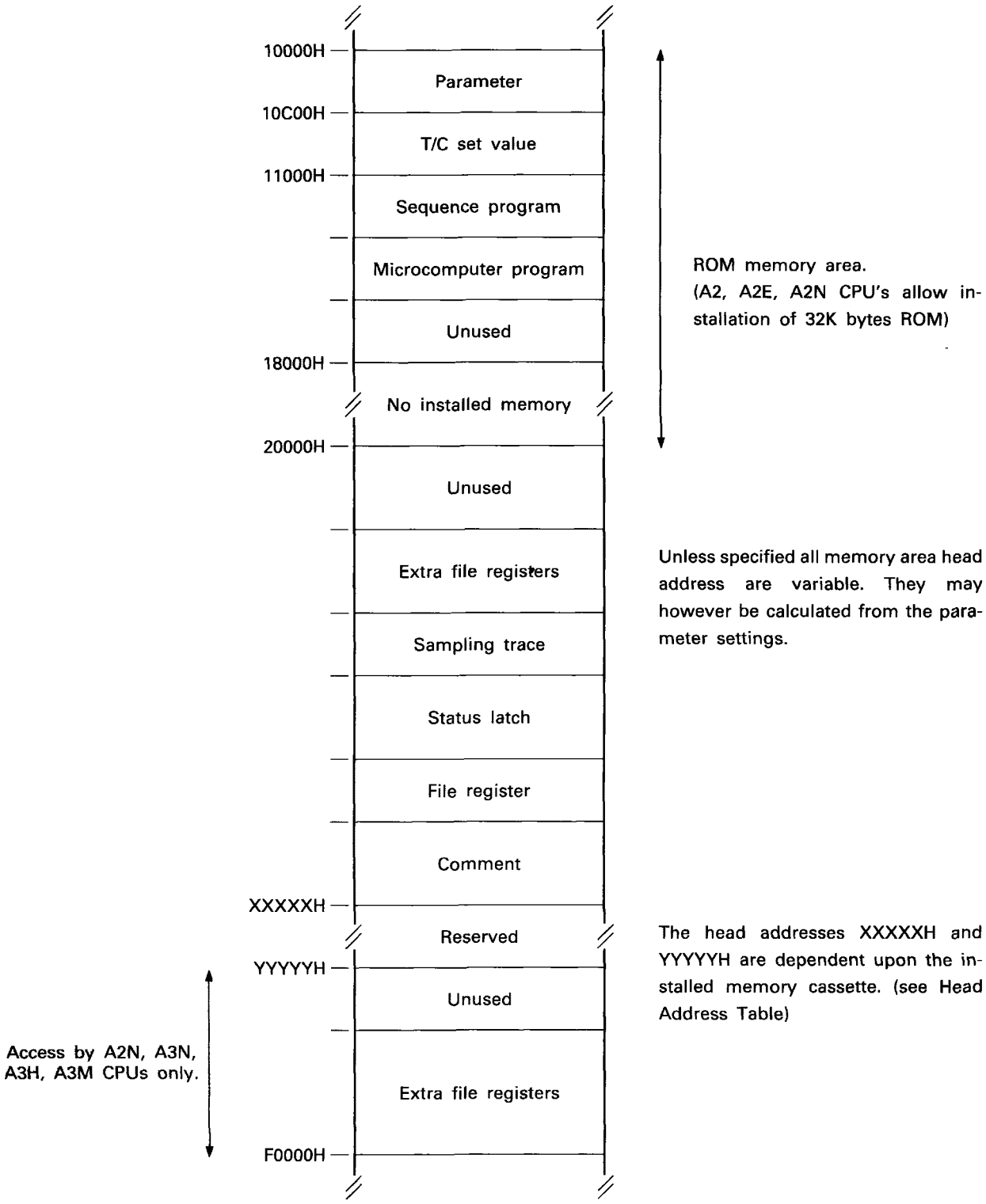
Installed Memory Map A2, A2E, A2N CPU RAM OPERATION



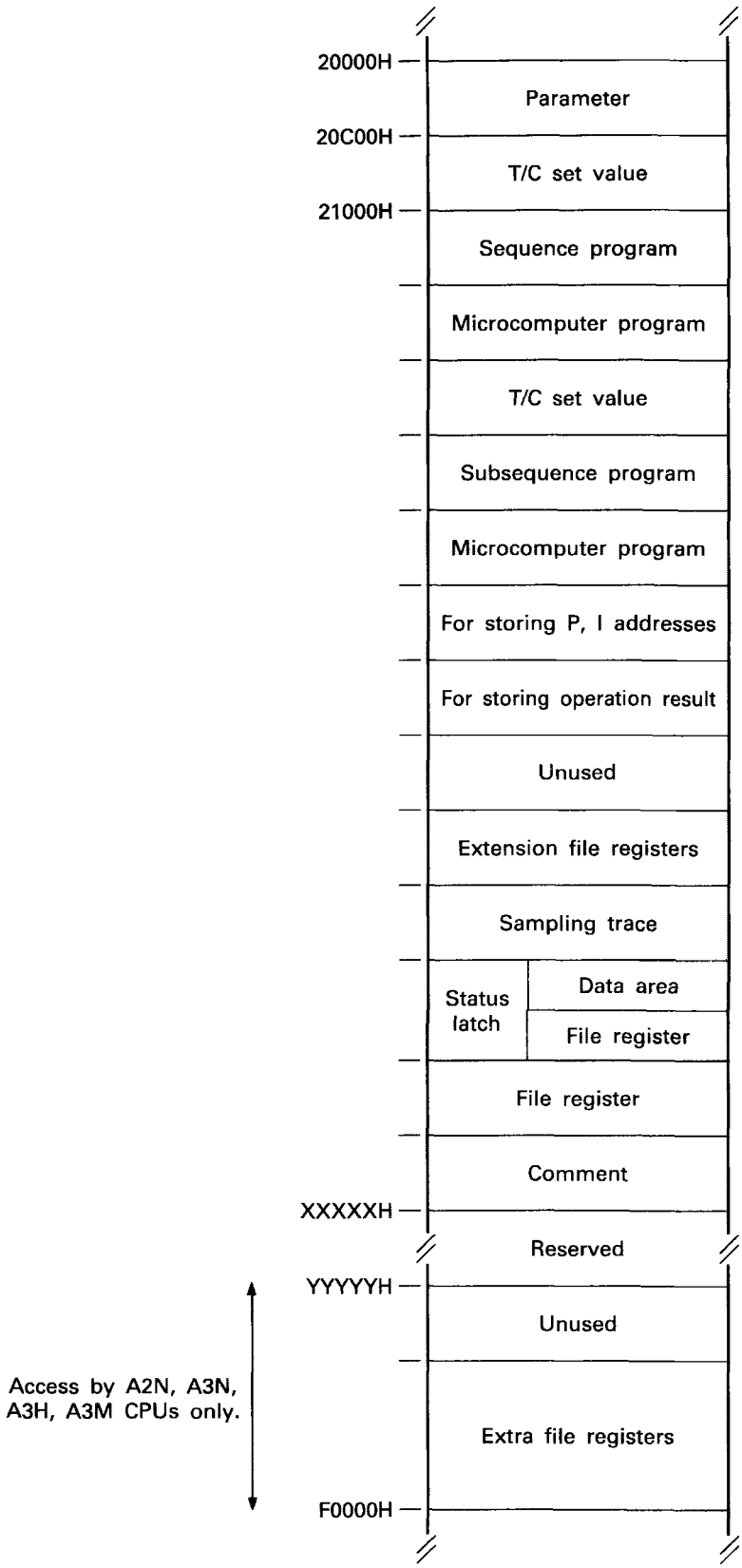
Unless specified all memory area head address are variable. They may however be calculated from the parameter settings.

The head addresses XXXXXH and YYYYYH are dependent upon the installed memory cassette. (see Head Address Table)

Installed Memory Map A2, A2E, A2N CPU ROM OPERATION



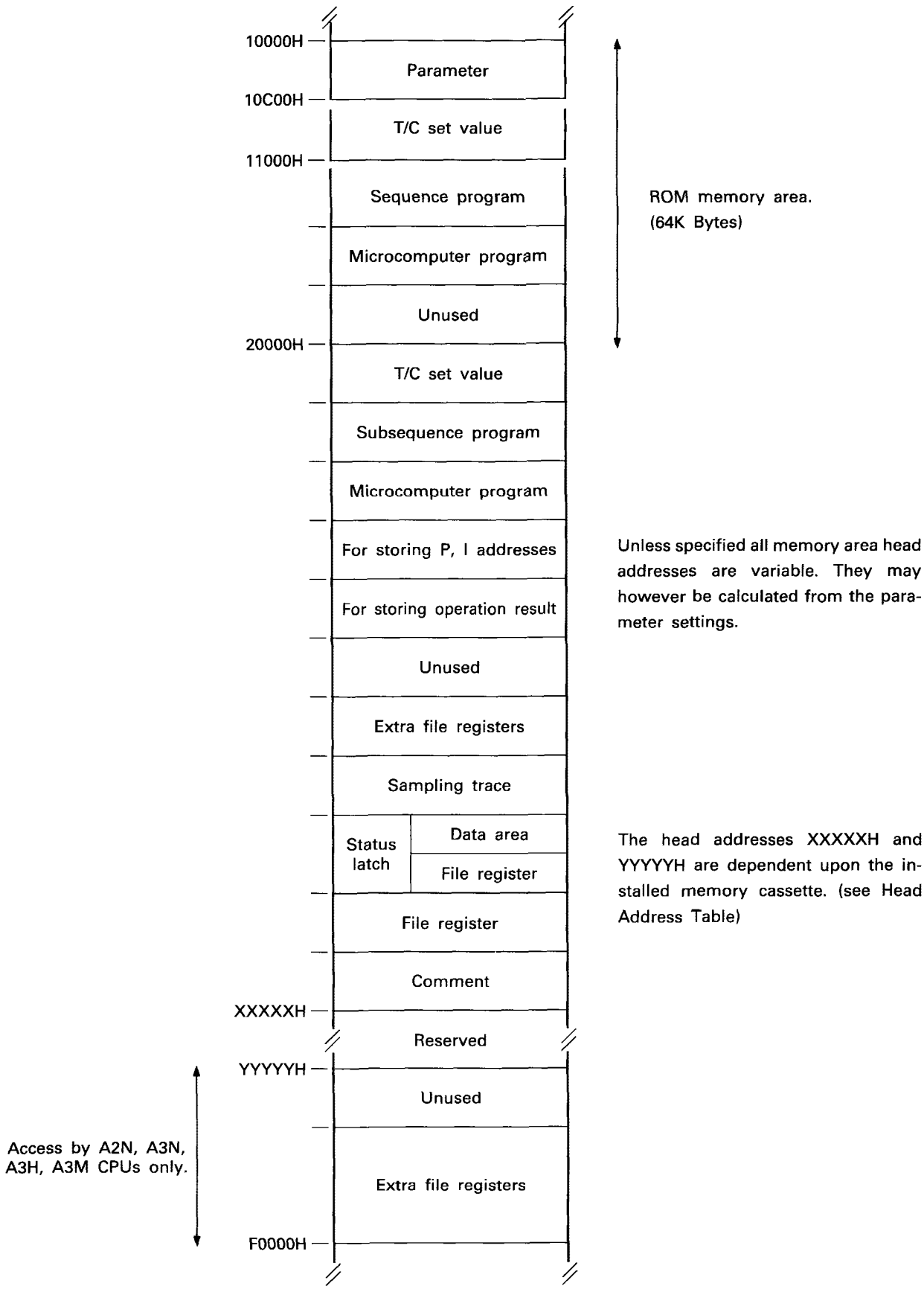
Installed Memory Map A3, A3E, A3N, A3H
RAM OPERATION



Unless specified all memory area head address are variable. They may however be calculated from the parameter settings.

The head addresses XXXXXH and YYYYYH are dependent upon the installed memory cassette. (see Head Address Table)

Installed Memory Map A3, A3E, A3N, A3H ROM OPERATION



Head Address Table

Memory Cassette		XXXXXH	YYYYYH	
A3MCA	A3NMCA			
0 (0)	0 (0)	20000H	———	
2 (16K)	2 (16K)	24000H	———	
4 (32K)	4 (32K)	28000H	———	
8 (64K)	8 (64K)	30000H	———	
12 (96K)	———	38000H	———	
———	16 (128K)	38000H	———	*1
18 (144K)	———	44000H	———	
———	24 (192K)	44000H	E4000H	A2N, A3N, A3H/M only.
———	40 (320K)	44000H	C0000H	A2N, A3N, A3H/M only.
———	56 (448K)	44000H	A0000H	A3H/M only.

*1 The remaining 32K bytes of memory, (38000H to 40000H) may be used as extra file registers, blocks 10 and 11.

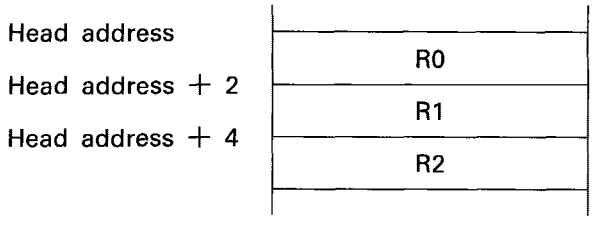
How to Calculate Extension File Register-R Addresses

The method used to calculate the actual address of extension file registers-R, differs depending on the block numbers to be accessed. i.e. block number 0, block numbers 1 to 9, or block numbers 10 to 28.

The block numbers which can or cannot be used are determined according to the CPU type, memory cassette, parameter setting contents, and/or RAM/ROM operation mode. For this information, refer to the SW1 GHPUTLP-FN1 manual.

The method used to calculate the head address of each extension file register, is indicated below:

The structure of file R of a block:



(1) Block number 0
Head address of block number 0
= 2000H
+ (memory cassette RAM capacity) *1
− (comment capacity)
− (file R capacity)

(2) Block numbers 1 to 9
Head address of block number "n"
= 2000H
+ (memory cassette RAM capacity) *1
− (comment capacity)
− (file R capacity)
− (status latch capacity)
− (16K bytes × n)

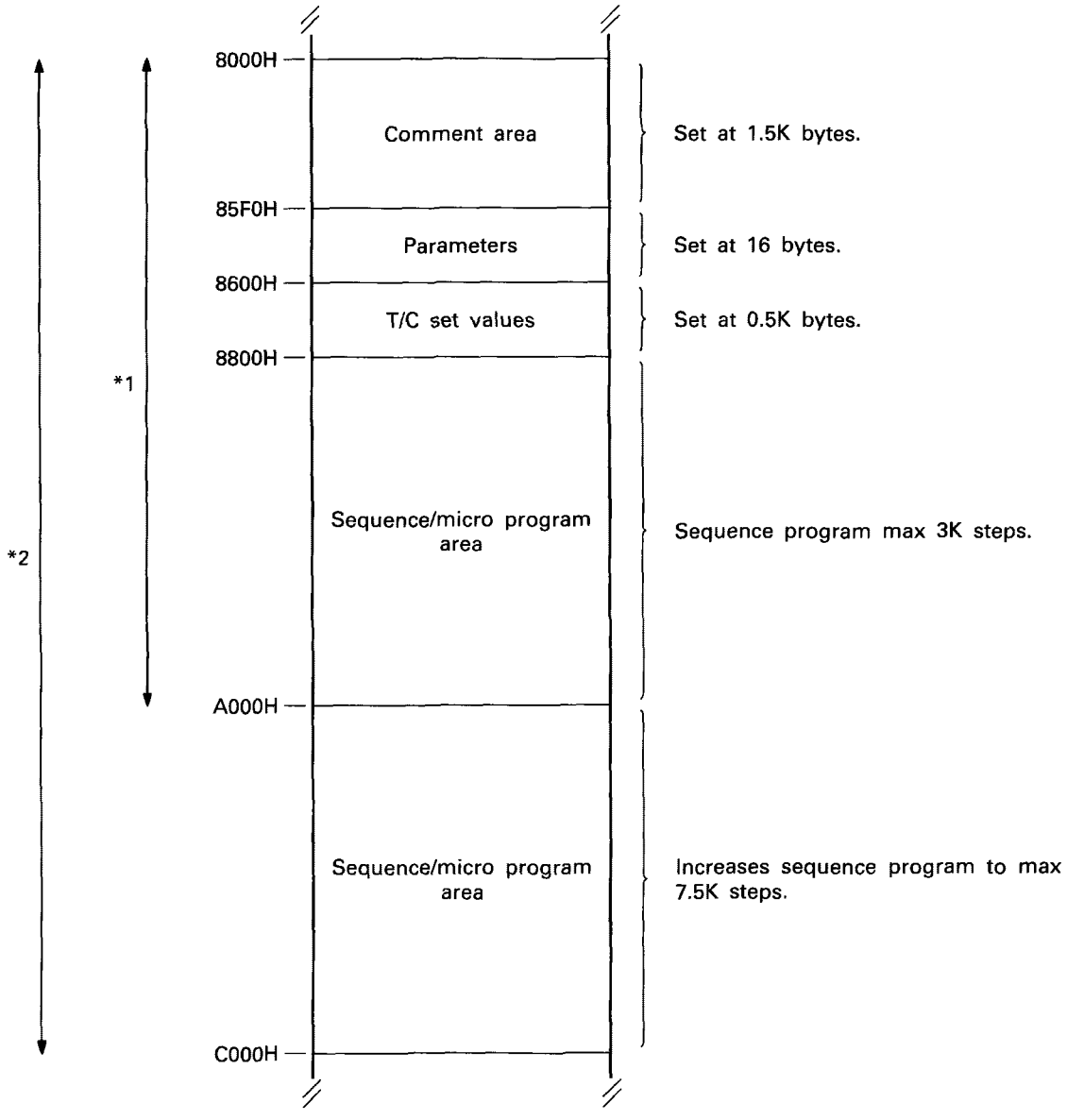
(3) Block numbers 10 to 28
The addresses are fixed according to the memory cassette capacity.
The address for each block number (10 to 28) one given overpage.

*1 for memory cassette types, A3NMCA24, 40 and 56, the RAM capacity is regarded as 144K bytes, in the above calculation.

Addresses for block numbers 10 to 28

Block No.	Memory Cassette Type	
	A3MCA-24 A3MCA-40 A3MCA-56	A3MCA-16
28	0xa0000	
27	0xa4000	
26	0xa8000	
25	0xac000	
24	0xb0000	
23	0xb4000	
22	0xb8000	
21	0xbc000	
20	0xc0000	
19	0xc4000	
18	0xc8000	
17	0xcc000	
16	0xd0000	
15	0xd4000	
14	0xd8000	
13	0xdc000	
12	0xe4000	
11	0xe8000	0x38000
10	0xec000	0x3c000

Installed Memory Map — A0J2 CPU RAM/ROM OPERATION

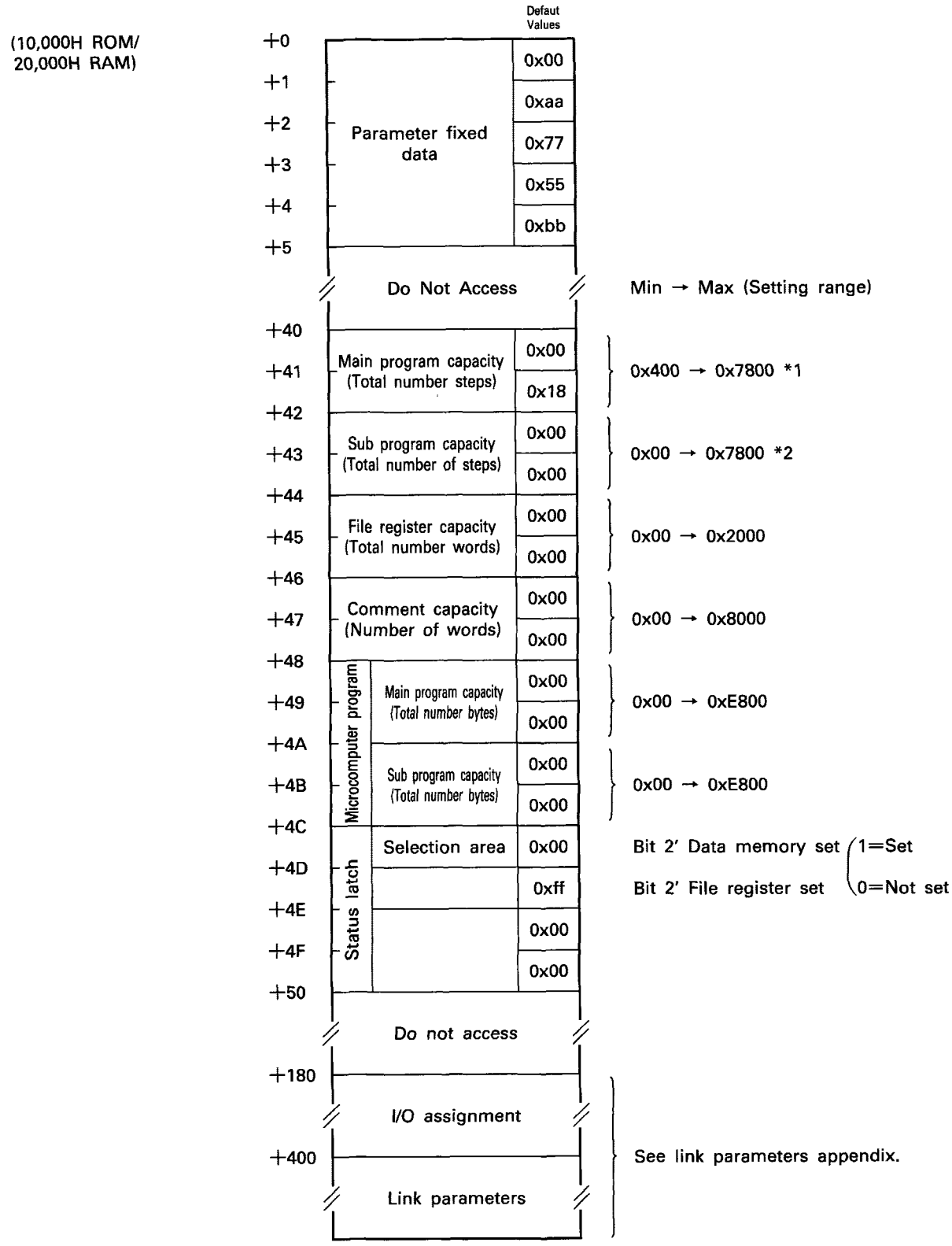


*1 8K bytes RAM, ROM, EROM
*2 16K bytes RAM, ROM

POINT

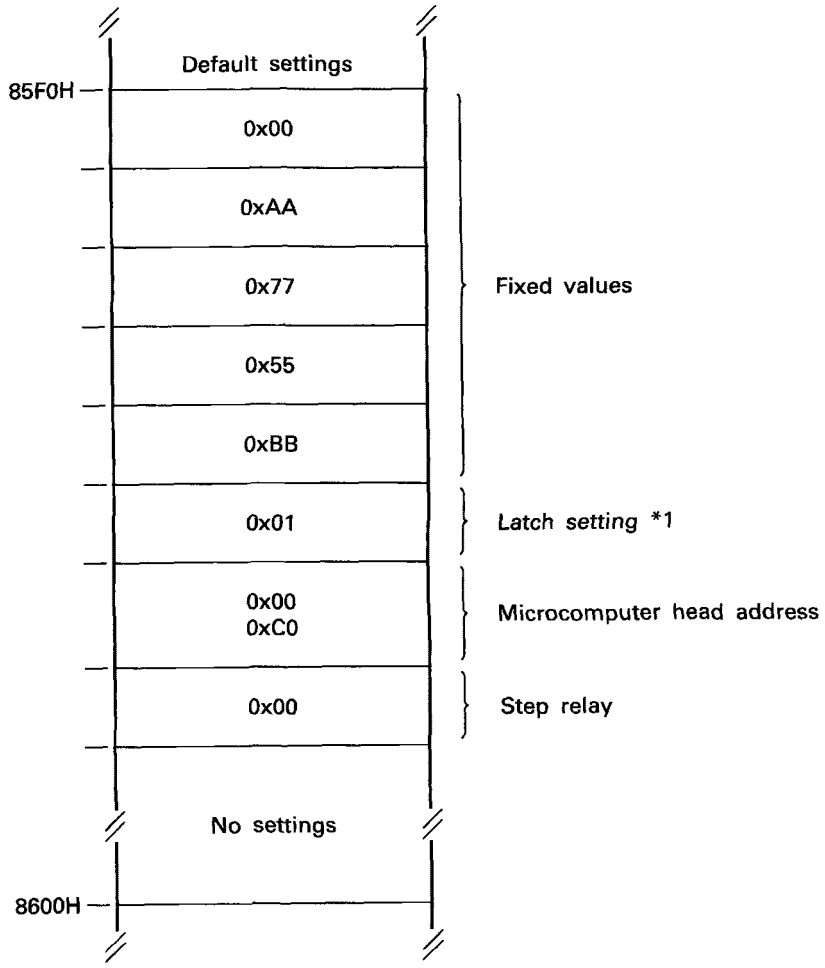
1 step = 2 bytes of memory.

Parameter Settings Memory Area — A1, A2, A3



*1 Not including T/C setting area (1K)
*2 Not including T/C setting, signal flow escape, P.I. setting areas.

Parameter Settings Memory Area — A0J2



*1 0 = No Latch
1 = Half Latch
2 = All Latch

*2 FF = Yes
00 = No

Working Area Memory Map

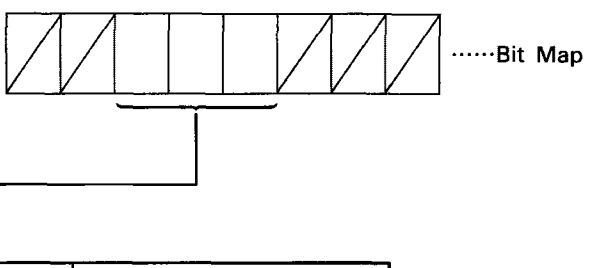
	Contents	A0J2	A1	A2	A3	A3H
1	ACPU RUN/STOP Status	0X7FFF	0Xa00	0xa000	0xa000	0x7000
2	Memory Cassette Information Area			0xad40	0xad40	0x70d40
3	Memory Protect Byte ①	0xd600	0xad40	0xad40	0xad40	0x70d40
	Information Area Byte ②		0xd800	0xd800	0xd800	0x72000
4	Sequence Program ROM/RAM Information Area	0x7ffe	0x9d20	0xad40	0xad40	0x70d40

Table Explanation

(1) A-CPU RUN/STOP Status

Status	Value
RUN	0xff
STOP	0x00

(2) Memory Cassette Information Area



Value	Installed M. Cassette	
000	MCA-18	*1
001	MCA-0	
010		
011	MCA-2	
100	MCA-4	
101	MCA-8	
110	MCA-12	*2
111	No Cassette	

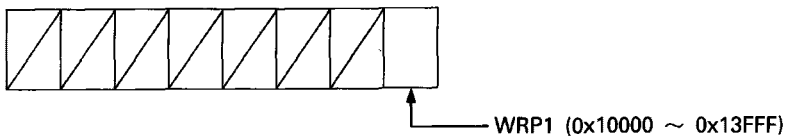
*1 Also A3NMCA 24, 40 and 56.
*2 Also A3NMCA 16

(3) Memory Protect Information Area

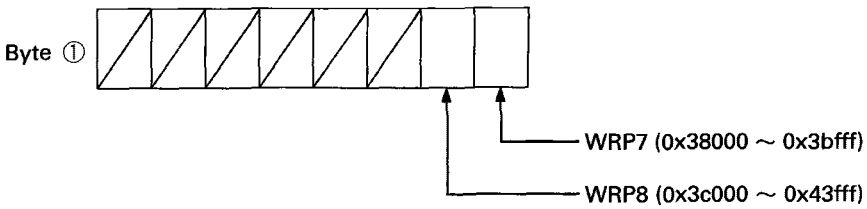
Area Contents 0: Protected
1: Unprotected

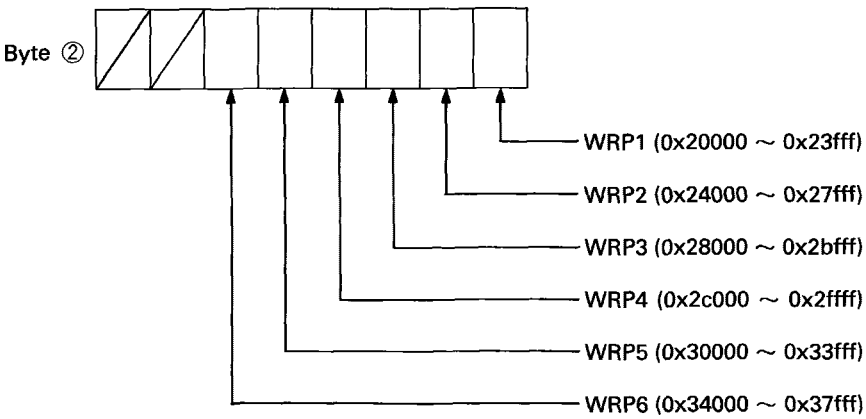
(WRP = Write Protected Range)

Bit Map A1CPU



Bit Map A2, A3, A3H CPU

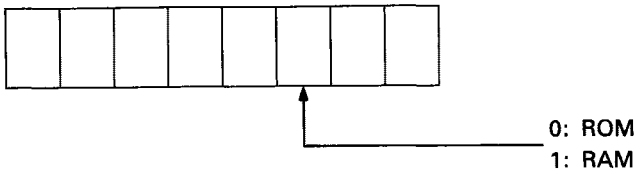




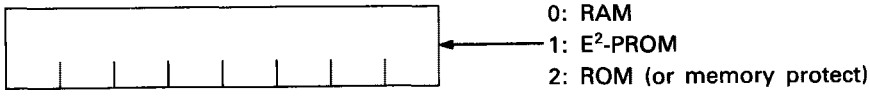
(4) Sequence Program ROM/RAM Information Area

Bit Map A1 CPU

Bit Map A2, A3, A3H CPU



Bit Map A0J2 CPU



Write Conditions

Items			Processing Contents	No. of Points Processed in a Single Communication Processing	PC CPU Status	
					STOP	RUN
Sequence program	Read	Main	Reads a main sequence program.	64 steps	○	○
		Sub	Reads a sub sequence program.			
	Write	Main	Writes a main sequence program.		○	×*
		Sub	Writes a sub sequence program.			
Parameters	Read		Reads the contents of the parameters set for the PC CPU.	128 bytes	○	○
	Write		Writes the contents of the parameters set for the PC CPU.		○	×
	Analysis request		To have the PC CPU recognize and check the changed parameter contents.		○	×
Comment	Batch read		Reads comments.	128 bytes	○	○
	Batch write		Writes comments.		○	○
Microcomputer program	Read	Main	Reads a main microcomputer program.	128 bytes	○	○
		Sub	Reads a sub microcomputer program.			
	Write	Main	Writes a main microcomputer program.		○	×*
		Sub	Writes a sub microcomputer program.			
Sampling trace	Read		Reads the sampling trace data.	128 bytes	○	×
	Write		Writes the sampling trace data.		○	×
Status latch	Read		Reads the latches status.	128 bytes	○	×
	Write		Writes the latches status.		○	×

Symbols in the PC CPU status column:

○ ····· Executable

× ····· Not executable

* ····· It is possible to write a program while the CPU is running another program (for example, writing a subprogram when a main program is being run). To do this with the A3CPU, special relay M9050 (signal flow change contact) must be OFF and special relay M9051 (CHG instruction execution inhibited) must be ON.

To do this with the A3N or the A3HCPU, special relay M9051 must be ON; special relay M9050 is not used.

APPENDIX 10 Timer/Counter Set Value Step Addresses

The processing code 0x01 allows the timer and counter set values to be read. To read the set values, define the head steps as indicated below:

Timer Set Value		Counter Set Value	
Set value	Head step	Set value	Head step
T0	0xFE00	C0	0xFF00
T1	0xFE01	C1	0xFF01
≈	≈	≈	≈
T255	0xFEFF	C255	0xFFFF

Example
To read the set values T0 to T63
Head address = $FE00_H$

Calculation of specified step
Timer : Tm= $FE00_H$ + n
Counter : Cm= $FF00_H$ + n
where, m= device number
n= hexadecimal value of device number

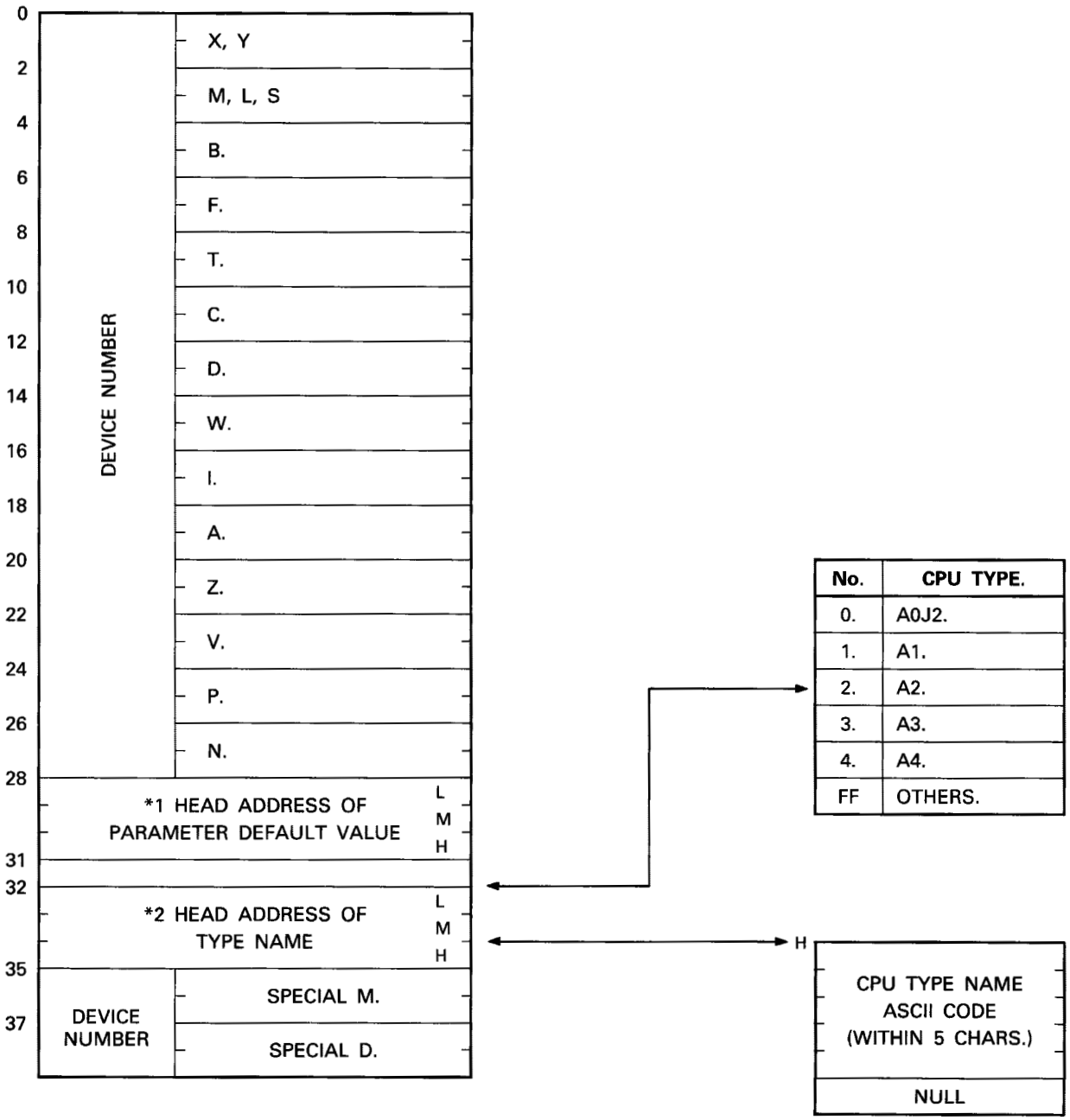
Meaning of T/C set values

T/C set values are stored as hexadecimal values as shown in the table below.

Ladder Example in Program	Setting in Program	Setting in T/C Set Value Area
K □□□□□□ T □□□	K0	0000_H
	K1	0001_H
	to	to
	K9	0009_H
K □□□□□□ C □□□	K10	$000A_H$
	to	to
	K32767	$7FFF_H$
D □□□□□ T □□□	D0	8000_H
	D1	8002_H
	D2	8004_H
D □□□□□ C □□□	to	to
	D1023	$87FE_H$

Calculation of Control Protocol value
Km= 0000_H + n
Dm= 8000_H + 2n
where, m= device number
n= hexadecimal value of device number

APPENDIX 11 System Data Table



NOTE

For CPU codes 0xA1, 0xA2, 0xA3, and 0xAB, system data addresses 31 to 38 do not exist.

*1 Contains head address of paramater default value table.

*2 Contains head address of CPU Type Name in ASCII Coding. (Six Byte Table. Five Bytes Code, One Byte Null)

APPENDIX 12 Special Function Module Buffer Memory Access

The following tables give the memory addresses and their corresponding TO/FROM Instruction Addresses of the various special function modules.
Refer to the unit manuals for details of the buffer memory contents.

(1) Type A68AD analog-digital converter module

Buffer Memory Contents	Address (Hexadecimal)		Address for FROM/TO Instruction
	Lower 8 bits	Higher 8 bits	
Number of channels	80H	81H	0
Averaging processing specification	82H	83H	1
CH1 averaging time, count	84H	85H	2
CH2 averaging time, count	86H	87H	3
CH3 averaging time, count	88H	89H	4
CH4 averaging time, count	8AH	8BH	5
CH5 averaging time, count	8CH	8DH	6
CH6 averaging time, count	8EH	8FH	7
CH7 averaging time, count	90H	91H	8
CH8 averaging time, count	92H	93H	9
CH1 digital output value	94H	95H	10
CH2 digital output value	96H	97H	11
CH3 digital output value	98H	99H	12
CH4 digital output value	9AH	9BH	13
CH5 digital output value	9CH	9DH	14
CH6 digital output value	9EH	9FH	15
CH7 digital output value	A0H	A1H	16
CH8 digital output value	A2H	A3H	17
Write data error code	C4H	C5H	34

(2) Type A62DA digital-analog converter module

Buffer Memory Contents	Address (Hexadecimal)		Address for FROM/TO Instruction
	Lower 8 bits	Higher 8 bits	
CH1 digital value	10H	11H	0
CH2 digital value	12H	13H	1
CH1 voltage set value check code	14H	15H	2
CH2 voltage set value check code	16H	17H	3
CH1 current set value check code	18H	19H	4
CH2 current set value check code	1AH	1BH	5

(3) Type A84AD analog-digital converter module

Buffer Memory Contents	Address (Hexadecimal)		Address for FROM/TO Instruction
	Lower 8 bits	Higher 8 bits	
Unused area	10H	11H	0
Averaging processing specification	12H	13H	1
CH1 averaging time, count	14H	15H	2
CH2 averaging time, count	16H	17H	3
CH3 averaging time, count	18H	19H	4
CH4 averaging time, count	1AH	1BH	5
Unused area (reserved)	——	——	——
CH1 digital I/O value	24H	25H	10
CH2 digital I/O value	26H	27H	11
CH3 digital I/O value	28H	29H	12
CH4 digital I/O value	2AH	2BH	13
CH1 internal set mode flag	2CH	2DH	14
CH2 internal set mode flag	2EH	2FH	15
CH3 internal set mode flag	30H	31H	16
CH4 internal set mode flag	32H	33H	17
CH1 temperature detector value	34H	35H	18
CH2 temperature detector value	36H	37H	19
CH3 temperature detector value	38H	39H	20
CH4 temperature detector value	3AH	3BH	21
CH1 set value check code	3CH	3DH	22
CH2 set value check code	3EH	3FH	23
CH3 set value check code	40H	41H	24
CH4 set value check code	42H	43H	25
Write data error code	44H	45H	26
Analog output permitted signal enable/disable flag	46H	47H	27
CH1 loaded module code	48H	49H	28
CH2 loaded module code	4AH	4BH	29
CH3 loaded module code	4CH	4DH	30
CH4 loaded module code	4EH	4FH	31
CH1 temperature set range (offset)	50H	51H	32
CH1 temperature set range (gain)	52H	53H	33
CH2 temperature set range (offset)	54H	55H	34
CH2 temperature set range (gain)	56H	57H	35
CH3 temperature set range (offset)	58H	59H	36
CH3 temperature set range (gain)	5AH	5BH	37
CH4 temperature set range (offset)	5CH	5DH	38
CH4 temperature set range (gain)	5EH	5FH	39

(4) Type AD61(S1) high-speed counter module

Buffer Memory Contents	Address (Hexadecimal)		Address for FROM/TO Instruction	
	Channel 1	Channel 2	CH1	CH2
Unused area (reserved)	80_H	$C0_H$	0	32
	81_H	$C1_H$		
Preset value write (lower bits)	82_H	$C2_H$	1	33
Preset value write (middle bits)	83_H	$C3_H$		
Preset value write (higher bits)	84_H	$C4_H$	2	34
	85_H	$C5_H$		
Mode register	86_H	$C6_H$	3	35
	87_H	$C7_H$		
Present value read (lower bits)	88_H	$C8_H$	4	36
Present value read (middle bits)	89_H	$C9_H$		
Present value read (higher bits)	$8A_H$	CA_H	5	37
	$8B_H$	CB_H		
Set value read, write (lower bits)	$8C_H$	CC_H	6	38
Set value read, write (middle bits)	$8D_H$	CD_H		
Set value read, write (higher bits)	$8E_H$	CE_H	7	39
	$8F_H$	CF_H		

(5) Type AD71(S1) positioning module

Buffer Memory Contents		Address (Hexadecimal)	Address for FROM/TO Instruction
X axis positioning start data		200_H to 391_H	0 to 200
Error reset		392_H 393_H	201
Y axis positioning start data		458_H to $5E9_H$	300 to 500
Positioning data	X axis positioning data	2040_H to $235F_H$	3872 to 4271
Positioning speed	X axis positioning data	2360_H to $267F_H$	4272 to 4671
Dwell time	X axis positioning data	2680_H to $299F_H$	4672 to 5071
Positioning address	X axis positioning data	$29A0_H$ to $2FDF_H$	5072 to 5871
Positioning data	Y axis positioning data	$2FE0_H$ to $32FF_H$	5872 to 6271
Positioning speed	Y axis positioning data	3300_H to $361F_H$	6272 to 6671
Dwell time	Y axis positioning data	3620_H to $393F_H$	6672 to 7071
Positioning address	Y axis positioning data	3940_H to $3F7F_H$	7072 to 7871
X axis parameter		$3F80_H$ to $3F9F_H$	7872 to 7887
Y axis parameter		$3FA8_H$ to $3FC7_H$	7892 to 7907
X axis zero return data		$3FD0_H$ to $3FDD_H$	7912 to 7917
Y axis zero return data		$3FE4_H$ to $3FF1_H$	7922 to 7928

(6) Type AD72 positioning module

Buffer Memory Contents	Address (Hexadecimal)	Address for FROM/TO Instruction
X axis positioning start data	200_H to 391_H	0 to 200
Error reset	392_H 393_H	201
Y axis positioning start data	458_H to $5E9_H$	300 to 500
Monitor area	$6B0_H$ to $6BF_H$	600 to 607
X axis positioning data	2040_H to $2FDF_H$	3872 to 5871
Y axis positioning data	$2FE0_H$ to $3F7F_H$	5872 to 7871
X axis parameters	$3F80_H$ to $3F9F_H$	7872 to 7891
Y axis parameters	$3FA8_H$ to $3FC7_H$	7892 to 7911
X axis zero return data	$3FD0_H$ to $3FDD_H$	7912 to 7917
Y axis zero return data	$3FE4_H$ to $3FE1_H$	7922 to 7928

(7) AJ71C24-S3

Address Specified by Computer	Address when Connected to Computer
1000_H to $11FF_H$	0 to FF_H
1200_H to $123F_H$	100_H to $11F_H$ Special-application area
1240_H to $1FFF_H$	120_H to $7FF_H$

APPENDIX 13 High Speed Memory Transfer Parameter Table

PC/AT → A3N

0#	X head Number
	No. of transfer bytes
	Y head Number
	No. of transfer bytes
	M head Number
	No. of transfer bytes
	L head Number
	No. of transfer bytes
10#	S head Number
	No. of transfer bytes
	B head Number
	No. of transfer bytes
	F head Number
	No. of transfer bytes
	Special M head Number
	0x00
20#	T head Number
	No. of transfer bytes
	C head Number
	No. of transfer bytes
	D head Number
	No. of transfer bytes
	W head Number
	No. of transfer bytes
30#	Special D head Number
	0x00

A3N → PC/AT

34#	X head Number
	No. of transfer bytes
	Y head Number
	No. of transfer bytes
	M head Number
	No. of transfer bytes
40#	L head Number
	No. of transfer bytes
	S head Number
	No. of transfer bytes
	B head Number
	No. of transfer bytes
	F head Number
	No. of transfer bytes
50#	Special M head Number
	No. of transfer bytes
	T head Number
	No. of transfer bytes
	C head Number
	No. of transfer bytes
	D head Number
	No. of transfer bytes
60#	W head Number
	No. of transfer bytes
	Special D head Number
68#	No. of transfer bytes

POINT

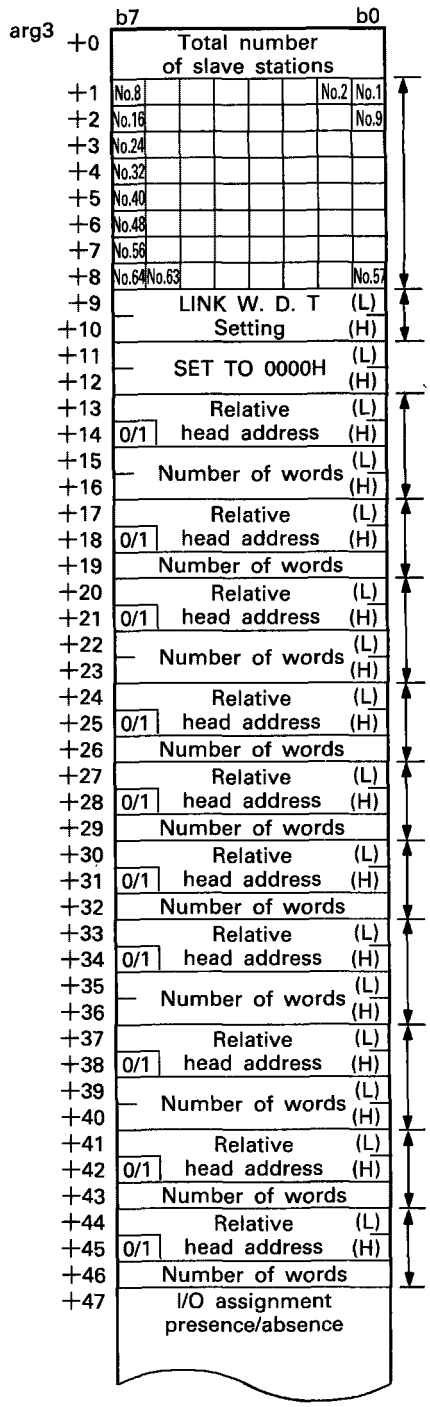
Device ranges to be transferred are specified by setting the head device number, and the number of bytes.

e.g. D40 to D59
D head number = 0x28
Number of bytes = 0x28

APPENDIX 14 Link Parameters and I/O Assignment Argument Table

Before attempting to set the link parameters, we recommend that section four of the Type Data Link Users manual is thoroughly read and understood.

Link Parameters



Slave station attributes
(each bit: 0/1 ... L/R station)
Set all bits to 0, except for the total number of slave stations.

10ms to 2sec (10ms increments)

Setting	Time (ms)
1	10
2	20
:	:

Watch Dog Timer

Total number and range of W-Registers used in the link for communication with local stations. i.e. M → L's, L's → M.

Total number and range of B-Relays used in the link.

Range of W assigned to the master station.

Range of B assigned to the master station.

Range of Y sent by M station to all L stations.

Range of X received by M station from all L stations.

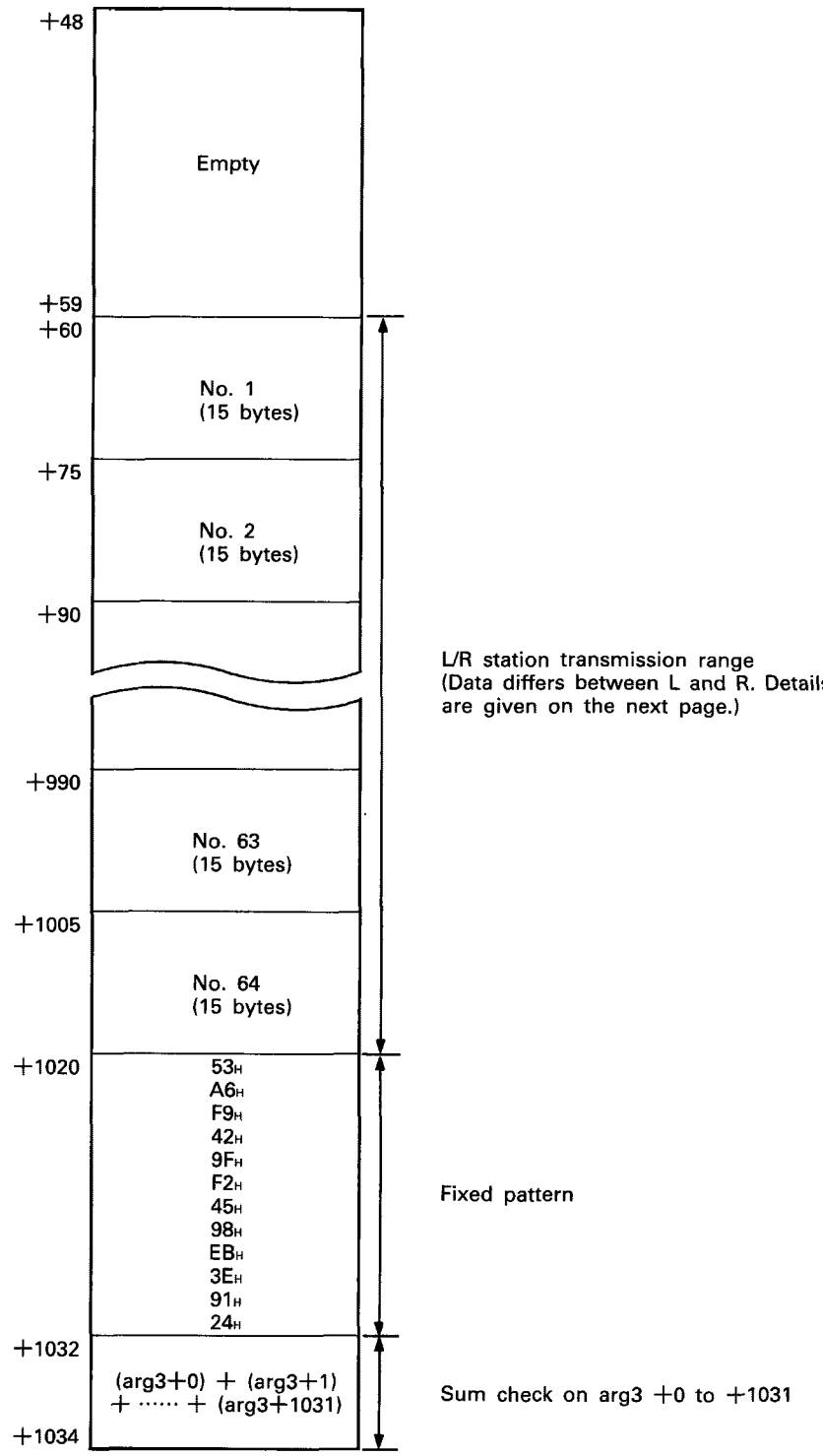
Range of W sent by M station to all R stations.

Range of W received by M station from all R stations.

Range of Y sent by M station to all R stations.

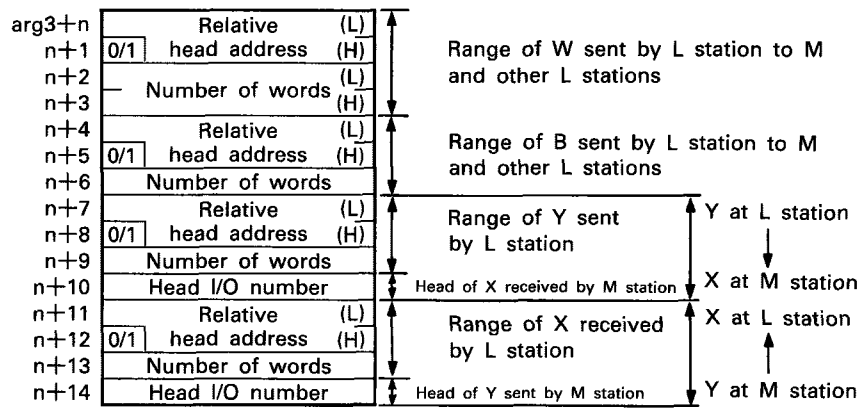
Range of X received by M station from all R stations.

00: No assignment made
01: Assignment made

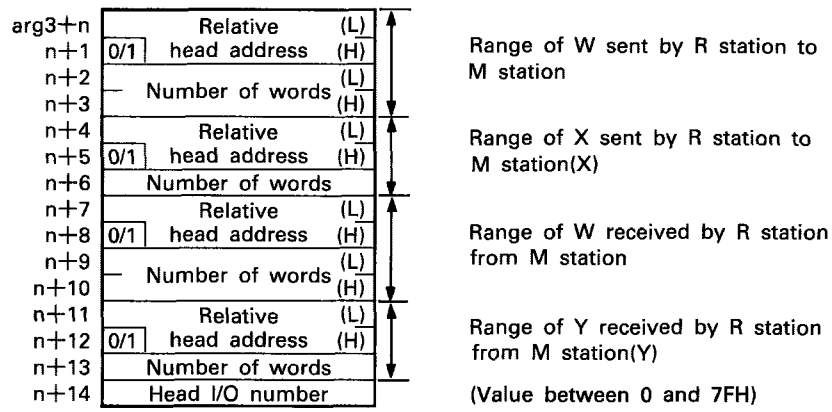

+48
Empty
+59
+60
No. 1
(15 bytes)
+75
No. 2
(15 bytes)
+90
L/R station transmission range
(Data differs between L and R. Details are given on the next page.)
+990
No. 63
(15 bytes)
+1005
No. 64
(15 bytes)
+1020
53H
A6H
F9H
42H
9FH
F2H
45H
98H
EBH
3EH
91H
24H
Fixed pattern
+1032
(arg3+0) + (arg3+1)
+ ····· + (arg3+1031)
+1034
Sum check on arg3 +0 to +1031

L/R Station Transmission Range

L station transmission range



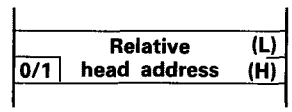
R station transmission range



POINT

Whether the corresponding data has been set or not is judged by the most significant bit at the relative head address.

Most significant bit = 0: Set
Most significant bit = 1: Not set



For all bytes not requiring set data a 1 must be written to the most significant bit of all Relative Head Address Locations.
One method is to simply write 0xFF to all unused bytes.

Argument Table Link Parameter Data Settings (All values in Hex)

(1) Relative Head Address Specification

The relative head addresses of the various devices are specified as follows.

W Registers		X, Y, B Bit Devices	
Device Number	Relative Head Address	Device Range	Relative Head Address
0	0	0 to F	0
1	2	10 to 1F	4
2	4	20 to 2F	8
⋮	⋮	⋮	⋮
3FF	7FE	7F0 to 7FF	1FC

Note: Relative Head Address W-Registers = Head W-Register Number × 4.

Relative Head Address X, Y, B = Head Device Number ÷ 4.

e.g. Head W-Register = W30
∴Relative Head Address = 30 × 4 = C0

Head Device = X80
∴Relative Head Address = 80 ÷ 4 = 20

(2) Number of Words Specification

The number of words setting, to specify device ranges, is performed as follows.

W Registers		X, Y, B Bit Devices	
Device Number	Number of Words	Device Range	Number of Words
W0	1	0 to F	1
W0 to W1	2	0 to 1F	2
W0 to W2	3	0 to 2F	3
⋮	⋮	⋮	⋮
W0 to W3FF	400	0 to 7FF	80

Note: Number of Words (W Registers) = Number of W-Registers

Number of Words (X, Y, B Devices) = Number of Devices ÷ 10

(3) Head I/O Number Specification

Head I/O Numbers are specified as follows.

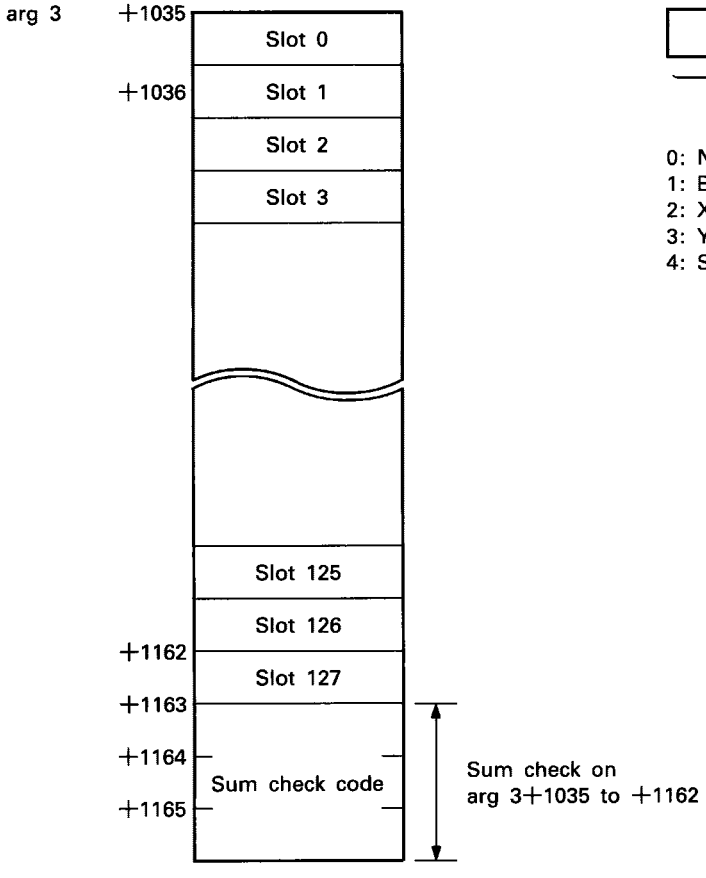
X, Y Bit Devices	
Device Range	**Head I/O Number**
0 to F	0
10 to 1F	1
20 to 2F	2
⋮	⋮
7F0 to 7FF	7F

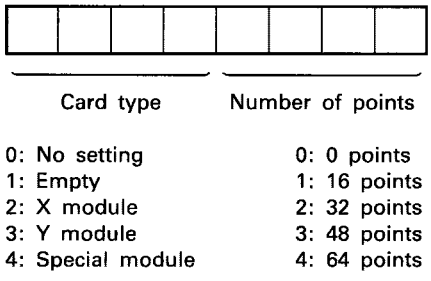
Note: Head I/O Number = Device Head Number ÷ 10

e.g. Device Head Number = X60
∴Head I/O Number = 60 ÷ 10 = 6

Slot Assignment Remote I/O

Before attempting to set slot I/O assignment, we recommend that section 4.5 (Example slot I/O assignment) of the Type Data Link User's Manual is thoroughly read and understood.





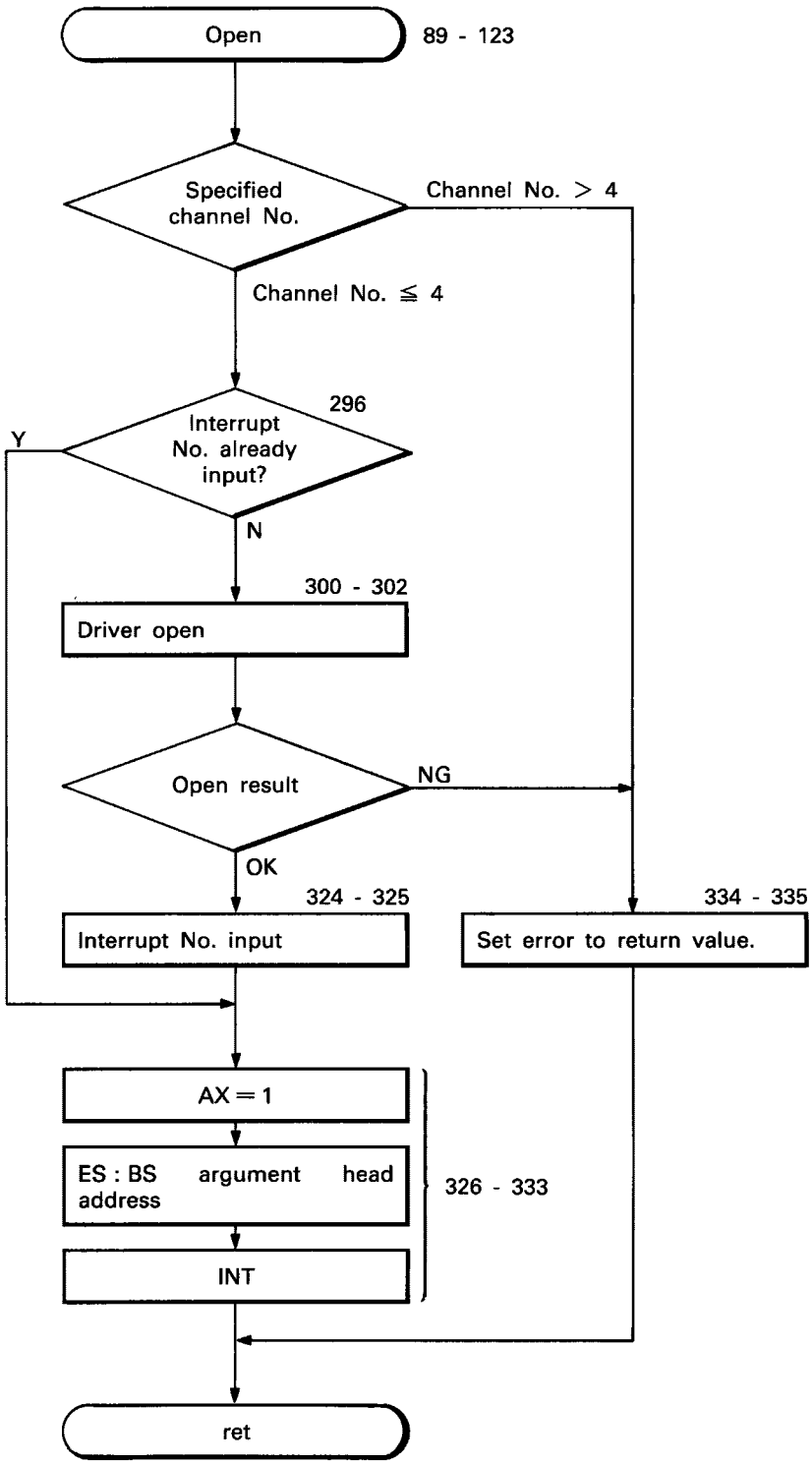
POINT

(1) When I/O Assignment is not to be specified, set all bytes (+1035 to +1166) to zero.
i.e. no setting. (including sum check code)

(2) Mapped I/O - A7BDE to Local A-CPU or A7BDE to A7BDE are specified as empty slots with the corresponding number of mapped I/O points.

APPENDIX 15 Assembler Access Functions Library — Source Code

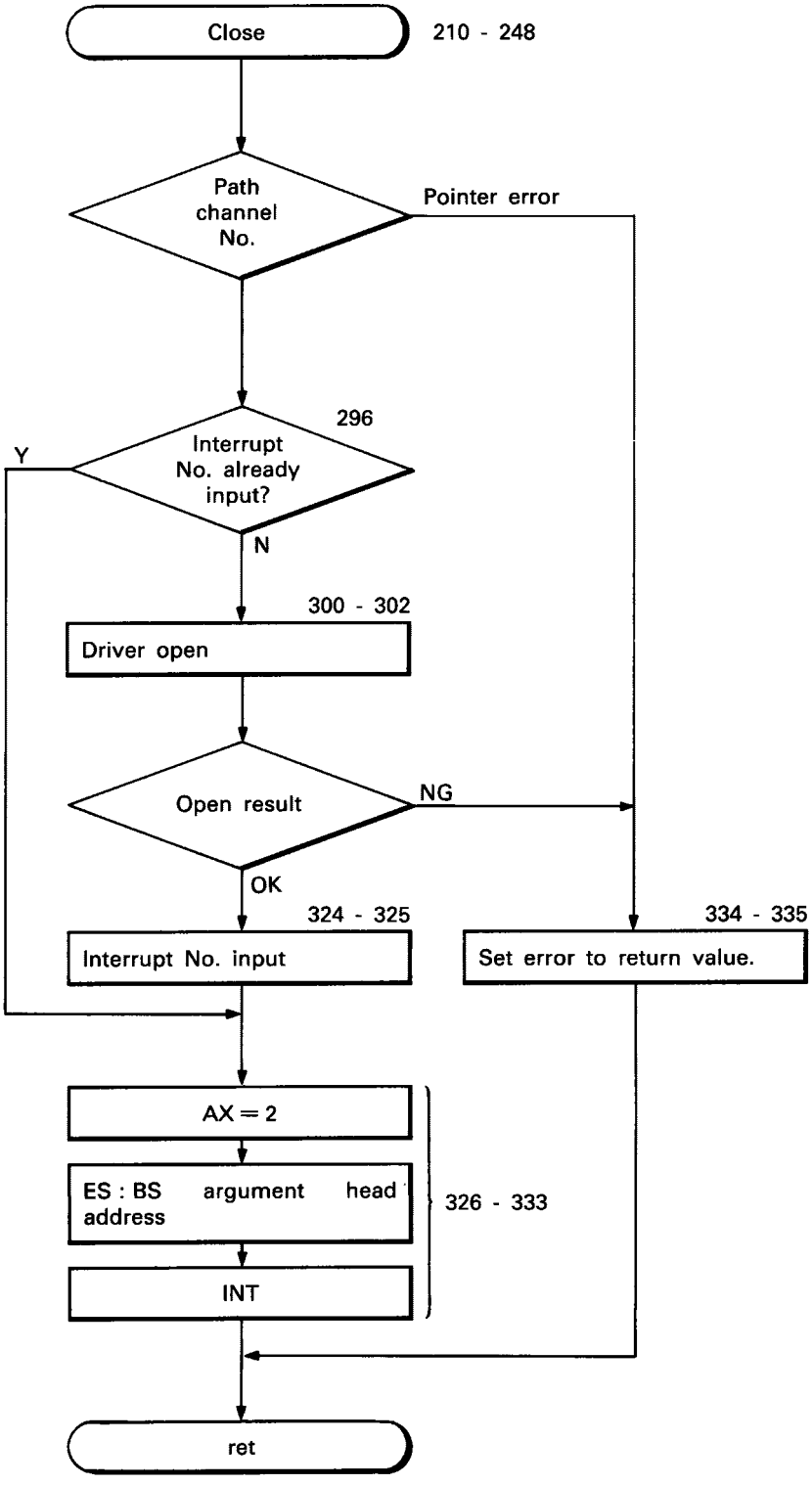
Open Processing



POINT

Numbers indicate line position in the Small Model Function Source Code.

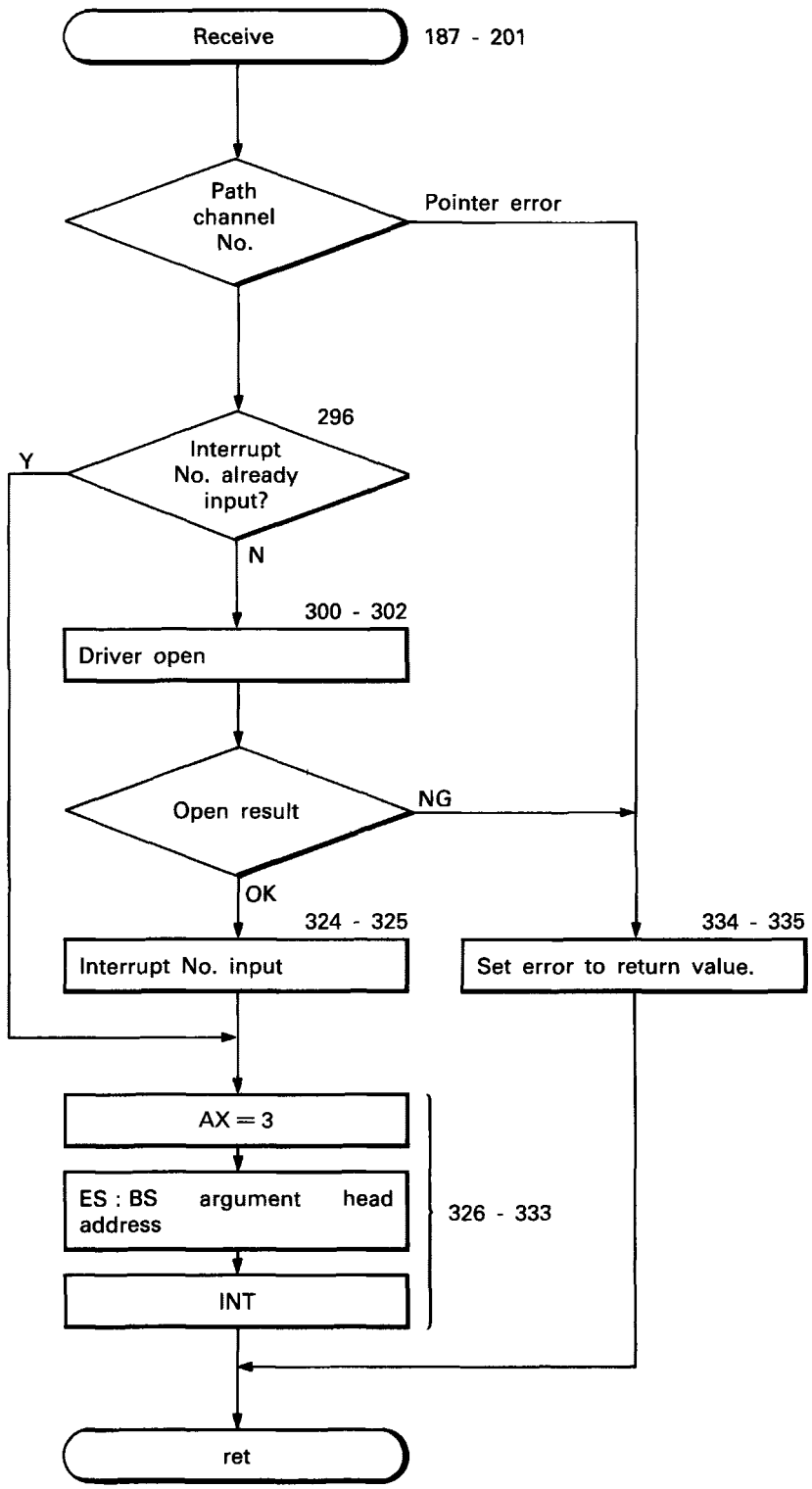
Close Processing



POINT

Numbers indicate line position in the Small Model Function Source Code.

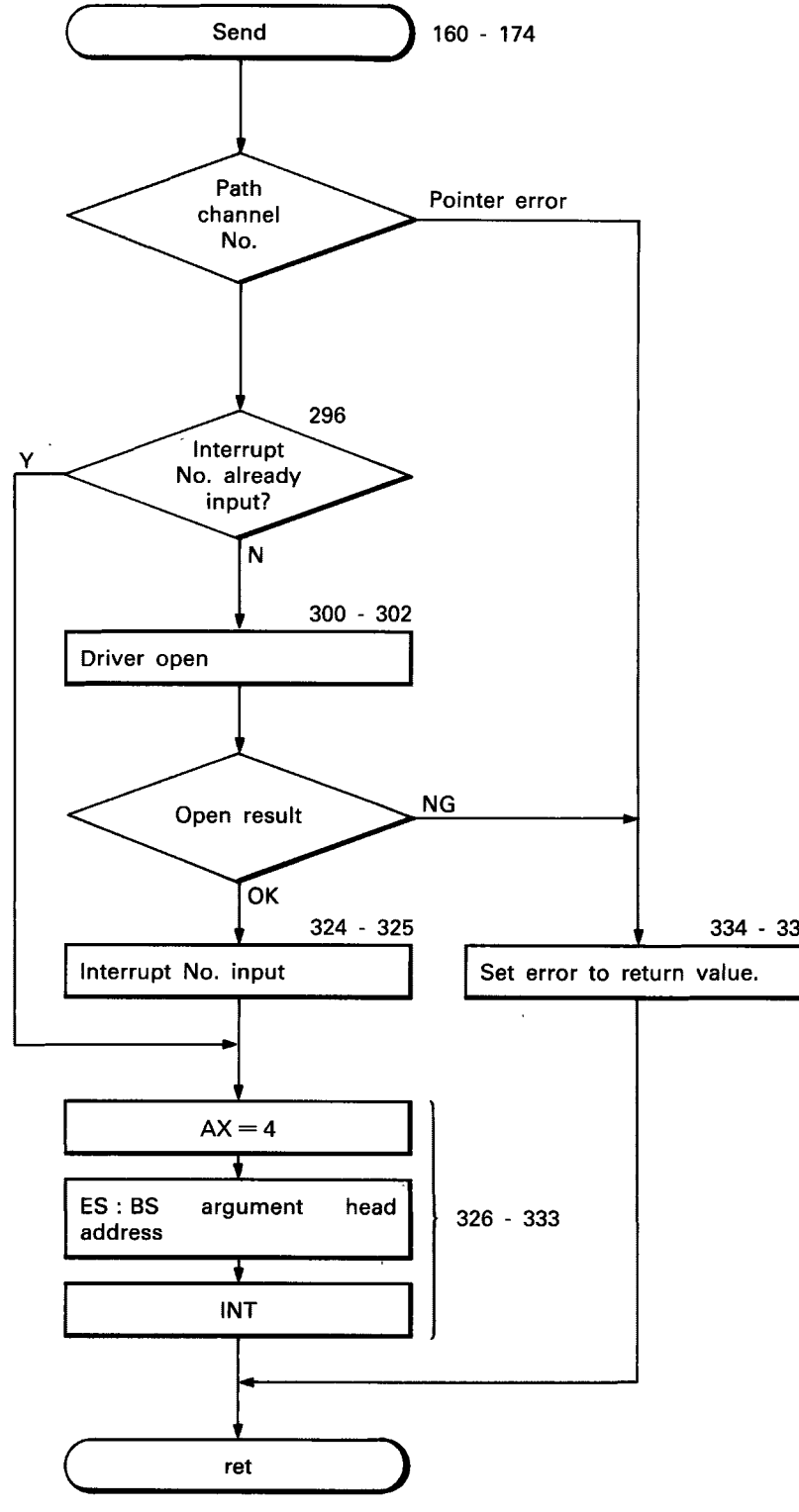
Receive Processing



POINT

Numbers indicate line position in the Small Model Function Source Code.

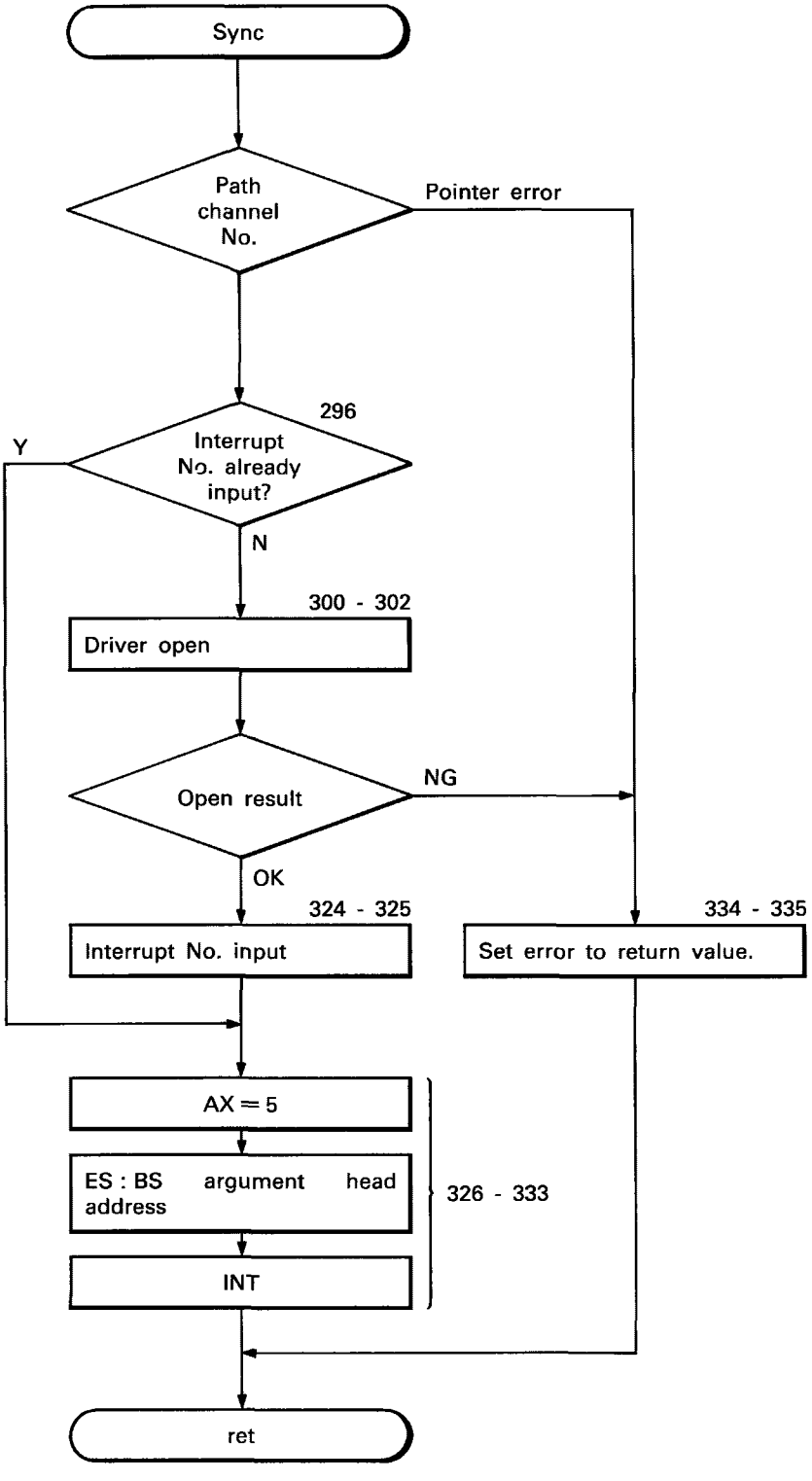
Send Processing



POINT

Numbers indicate line position in the Small Model Function Source Code.

Sync Processing



POINT

Numbers indicate line position in the Small Model Function Source Code.

```
 1:                 PAGE    80,132
 2:  ;*********************************************************************
 3:  ;*             MELSEC LIBRARY                                        *
 4:  ;*                FOR MS-DOS (FOR SMALL MODEL)                       *
 5:  ;*                     MITSUBISHI ELECTRIC CORPORATION               *
 6:  ;*********************************************************************
 7:                 name    melsec_net_lib                          ;
 8:  _text          segment word public 'code'                      ;
 9:  assume         cs:_text                                        ;
10:
11:                 org     0                                       ;
12:  ;*********************************************************************
13:  ;*             PUBLIC DECLARE                                        *
14:  ;*                     FOR LIBRARY                                   *
15:  ;*********************************************************************
16:                 public          _nl1close                       ;
17:                 public          _nl1open                        ;
18:                 public          _nl1receive                     ;
19:                 public          _nl1send                        ;
20:                 public          _nl1sync                        ;
21:  ;*********************************************************************
22:  ;*             EQU DEFINITION                                        *
23:  ;*                     FOR LIBRARY                                   *
24:  ;*********************************************************************
25:
26:  INT_OP_CODE     equ     0cdh                                    ;
27:  INT_A_STS_END   equ     01h                                     ;
28:  INT_A_STS_EMP   equ     00h                                     ;
29:  DOS_INT         equ     21h                                     ;
30:  DRV_OPEN        equ     3dh                                     ;
31:  DRV_OPN_RD_ONLY equ     00h                                     ;
32:  IOCTL           equ     44h                                     ;
33:  IOCTL_READ      equ     02h                                     ;
34:  IOCTL_READ_SIZE equ     01h                                     ;
35:  DRV_CLOSE       equ     3eh                                     ;
36:  FUNC_OPEN       equ     0001h                                   ;
37:  FUNC_CLOSE      equ     0002h                                   ;
38:  FUNC_RECEIVE    equ     0003h                                   ;
39:  FUNC_SEND       equ     0004h                                   ;
40:  FUNC_SYNC       equ     0005h                                   ;
41:  ERR_PATH        equ     0044h                                   ;
42:  ERR_CHANEL_NO   equ     0041h                                   ;
43:  ERR_NOT_FOUND   equ     0001h                                   ;
44:
45:  OPEN_ARG_WORD   equ     2
46:  OPEN_CHAN_ADR   equ     4
47:  OPEN_PATH_ADR   equ     6
48:  SYNC_ARG_WORD   equ     3
49:  SEND_ARG_WORD   equ     6
50:  REC_ARG_WORD    equ     6
51:  CLOSE_ARG_WORD  equ     2
52:
53:  ARG_PATH_ADR    equ     4
54:  OPEN_SET_CHAN   equ     0
55:  OPEN_SET_PATH_O equ     2
```

```
 56:  OPEN_SET_PATH_S equ     4
 57:  ARG_PATH_ADR_O  equ     4
 58:  ARG_PATH_ADR_S  equ     6
 59:  ARG_MODE        equ     8
 60:  ARG_ARG1_ADR_O  equ     10
 61:  ARG_ARG2_ADR_O  equ     12
 62:  ARG_ARG3_ADR_O  equ     14
 63:  SET_PATH_ADR_O  equ     0
 64:  SET_PATH_ADR_S  equ     2
 65:  SET_MODE        equ     4
 66:  SET_ARG1_ADR_O  equ     6
 67:  SET_ARG1_ADR_S  equ     8
 68:  SET_ARG2_ADR_O  equ     10
 69:  SET_ARG2_ADR_S  equ     12
 70:  SET_ARG3_ADR_O  equ     14
 71:  SET_ARG3_ADR_S  equ     16
 72:  ;*********************************************************************
 73:  ;*                      PROCESS ADDRESS TABLE FOR CHANNEL            *
 74:  ;*********************************************************************
 75:  common_adr_tbl  dw      common_a3n                          ;A3N
 76:                  dw      common_rs4                          ;RS422(MELSEC)
 77:                  dw      common_net                          ;MNET
 78:                  dw      common_rs4                          ;RS422 (OTHER)
 79:                  dw      common_rs4                          ;RS232c(OTHER)
 80:  CHANEL_MAX      equ     ($-common_adr_tbl)/2-1
 81:  ;*********************************************************************
 82:  ;*                                                                   *
 83:  ;*              bp+0  : BP                                           *
 84:  ;*              bp+2  : RETURN ADDRESS OFFSET                        *
 85:  ;*              bp+4  : CHANNEL NUMBER                               *
 86:  ;*              bp+6  : PATH ADDRESS OFFSET                          *
 87:  ;*                                                                   *
 88:  ;*********************************************************************
 89:  _nllopen        proc    near
 90:                  push    bp
 91:                  mov     bp,sp
 92:                  push    ds
 93:                  push    es
 94:                  push    di
 95:                  push    si
 96:                  push    bx
 97:                  push    cx
 98:
 99:                  mov     ax,FUNC_OPEN
100:                  mov     cs:word ptr [func],ax
101:                  mov     cx,OPEN_ARG_WORD
102:                  call    prm_set
103:          mov     ax,ss:word ptr OPEN_CHAN_ADR[bp]
104:                  cmp     ax,CHANEL_MAX
105:                  ja      nllopen_err
106:                  mov     bx,offset nllopen_ret
107:                  push    bx
108:                  add     ax,ax
109:                  mov     bx,offset common_adr_tbl
110:                  add     bx,ax
```

```
111:                    jmp     cs:word ptr [bx]
112:  nllopen_err:
113:                    mov     ax,ERR_CHANEL_NO
114:  nllopen_ret:
115:                    pop     cx
116:                    pop     bx
117:                    pop     si
118:                    pop     di
119:                    pop     es
120:                    pop     ds
121:                    pop     bp
122:                    ret
123:  _nllopen          endp
124:  ;*********************************************************************
125:  ;*                                                                   *
126:  ;*                bp+0  : BP                                         *
127:  ;*                bp+2  : RETURN ADDRESS OFFSET                      *
128:  ;*                bp+4  : PATH ADDRESS OFFSET                        *
129:  ;*                bp+6  : PATH ADDRESS SEGMENT                       *
130:  ;*                bp+8  : SINK MODE                                  *
131:  ;*                                                                   *
132:  ;*********************************************************************
133:  _nllsync          proc    near
134:                    push    bp
135:                    mov     bp,sp
136:                    push    ds
137:                    push    es
138:                    push    di
139:                    push    si
140:                    push    bx
141:                    push    cx
142:                    mov     ax,FUNC_SYNC
143:                    mov     cs:word ptr [func],ax
144:                    mov     cx,SYNC_ARG_WORD
145:                    call    prm_set
146:          jmp       nllclose05
147:  _nllsync          endp
148:  ;*********************************************************************
149:  ;*                                                                   *
150:  ;*                bp+0  : BP                                         *
151:  ;*                bp+2  : RETURN ADDRESS OFFSET                      *
152:  ;*                bp+4  : PATH ADDRESS OFFSET                        *
153:  ;*                bp+6  : PATH ADDRESS SEGMENT                       *
154:  ;*                bp+8  : SINK MODE                                  *
155:  ;*                bp+10 : ARG1 ADDRESS OFFSET                        *
156:  ;*                bp+12 : ARG2 ADDRESS OFFSET                        *
157:  ;*                bp+14 : ARG3 ADDRESS OFFSET                        *
158:  ;*                                                                   *
159:  ;*********************************************************************
160:  _nllsend          proc    near
161:                    push    bp
162:                    mov     bp,sp
163:                    push    ds
164:                    push    es
165:                    push    di
```

```
166:                    push    si
167:                    push    bx
168:                    push    cx
169:                    mov     ax,FUNC_SEND
170:                    mov     cs:word ptr [func],ax
171:                    mov     cx,SEND_ARG_WORD
172:                    call    prm_set
173:            jmp     nllclose05
174:  _nllsend          endp
175:  ;*********************************************************************
176:  ;*                                                                   *
177:  ;*                bp+0  : BP                                         *
178:  ;*                bp+2  : RETURN ADDRESS OFFSET                      *
179:  ;*                bp+4  : PATH ADDRESS OFFSET                        *
180:  ;*                bp+6  : PATH ADDRESS SEGMENT                       *
181:  ;*                bp+8  : SINK MODE                                  *
182:  ;*                bp+10 : ARG1 ADDRESS OFFSET                        *
183:  ;*                bp+12 : ARG2 ADDRESS OFFSET                        *
184:  ;*                bp+14 : ARG3 ADDRESS OFFSET                        *
185:  ;*                                                                   *
186:  ;*********************************************************************
187:  _nllreceive       proc    near
188:                    push    bp
189:                    mov     bp,sp
190:                    push    ds
191:                    push    es
192:                    push    di
193:                    push    si
194:                    push    bx
195:                    push    cx
196:                    mov     ax,FUNC_RECEIVE
197:            mov     cs:word ptr [func],ax
198:                    mov     cx,REC_ARG_WORD
199:            call    prm_set
200:            jmp     nllclose05
201:  _nllreceive       endp
202:  ;*********************************************************************
203:  ;*                                                                   *
204:  ;*                bp+0  : BP                                         *
205:  ;*                bp+2  : RETURN ADDRESS OFFSET                      *
206:  ;*                bp+4  : PATH ADDRESS OFFSET                        *
207:  ;*                bp+6  : PATH ADDRESS SEGMENT                       *
208:  ;*                                                                   *
209:  ;*********************************************************************
210:  _nllclose         proc    near
211:                    push    bp
212:                    mov     bp,sp
213:                    push    ds
214:                    push    es
215:                    push    di
216:                    push    si
217:                    push    bx
218:                    push    cx
219:                    mov     ax,FUNC_CLOSE
220:                    mov     cs:word ptr [func],ax
```

```
221:                    mov     cx,CLOSE_ARG_WORD
222:                    call    prm_set
223:  nl1close05:
224:                    push    es
225:                    les     di,ss:dword ptr ARG_PATH_ADR[bp]
226:                    mov     al,es:byte ptr [di]
227:                    pop     es
228:                    xor     ah,ah
229:                    cmp     ax,CHANEL_MAX
230:                    ja      nl1close_err
231:                    mov     bx,offset nl1close_ret
232:                    push    bx
233:                    add     ax,ax
234:                    mov     bx,offset common_adr_tbl
235:                    add     bx,ax
236:                    jmp     cs:word ptr [bx]
237:  nl1close_err:
238:                    mov     ax,ERR_PATH
239:  nl1close_ret:
240:                    pop     cx
241:                    pop     bx
242:                    pop     si
243:                    pop     di
244:                    pop     es
245:                    pop     ds
246:                    pop     bp
247:                    ret
248:  _nl1close         endp
249:
250:
251:
252:  ;*************************************************************************
253:  ;*                                                                       *
254:  ;*                                                                       *
255:  ;*************************************************************************
256:  common_prc         proc    near
257:                    ;****************
258:                    ;*     A3N      *
259:                    ;****************
260:  common_a3n:
261:                    mov     si,offset int_a_set_a3n
262:          mov       dx,offset drv_nm_a3n
263:                    mov     di,offset int_a_no_a3n
264:                    jmp     common10
265:
266:                    ;****************
267:                    ;*     RS422    *
268:                    ;****************
269:  common_rs4:
270:                    mov     si,offset int_a_set_rs4
271:          mov       dx,offset drv_nm_rs4
272:                    mov     di,offset int_a_no_rs4
273:                    jmp     common10
274:
275:                    ;****************
```

```
276:                    ;*      MNET     *
277:                    ;****************
278:  common_net:
279:                    mov     si,offset int_a_set_net
280:          mov       dx,offset drv_nm_net
281:                    mov     di,offset int_a_no_net
282:                    jmp     common10
283:
284:  ;*********************************************************************
285:  ;*                                                                   *
286:  ;*                SI : INT-A SETTED FLAG ADDRESS OFFSET              *
287:  ;*                DX : DRIVER NAME ADDRESS OFFSET                    *
288:  ;*                DI : INT-A NUMBER SAVE AREA ADDRESS OFFSET         *
289:  ;*                                                                   *
290:  ;*********************************************************************
291:  common10:
292:                    push    cs
293:                    pop     ds
294:                    mov     al,cs:byte ptr [si]                ;int-a set?
295:                    or      al,al                              ;
296:                    jnz     int_start                          ;YES
297:                                                               ;
298:                                                               ;DS = Driver name segment
299:                                                               ;DX = Driver name offset
300:                    mov     ah,DRV_OPEN                        ;AH = Function code
301:                    mov     al,DRV_OPN_RD_ONLY                 ;AL = Access code
302:                    int     DOS_INT                            ;Driver open
303:                    jc      common_err                         ;
304:
305:
306:                    mov     cs:word ptr [handle_no],ax         ;
307:
308:                                                               ;DS = Receive buffer segm

309:                    mov     dx,di                              ;DX = Receive buffer offs

310:                    mov     bx,ax                              ;BX = Handle number
311:                    mov     ah,IOCTL                           ;AH = Function code
312:                    mov     al,IOCTL_READ                      ;AL = Receive specfy
313:                    mov     cx,IOCTL_READ_SIZE                 ;CX = Receive data size
314:          int       DOS_INT
315:          jc        common_err
316:
317:                    mov     cs:byte ptr [si],INT_A_STS_END
318:
319:                    mov     bx,cs:word ptr [handle_no]         ;
320:                    mov     ah,DRV_CLOSE                       ;
321:                    int     DOS_INT                            ;
322:                    jc      common_err
323:  int_start:
324:                    mov     al,cs:byte ptr [di]
325:                    mov     cs:byte ptr [int_code],al
326:                    push    cs
327:                    pop     es
328:                    mov     bp,offset prm_area
329:                    mov     bx,bp
330:                    mov     ax,cs:word ptr [func]
```

```
331:                    db      INT_OP_CODE
332:  int_code          db      00h
333:                    jmp     common_end
334:  common_err:
335:                    mov     ax,ERR_NOT_FOUND
336:  common_end:
337:                    ret
338:  common_prc        endp
339:
340:  ;*********************************************************************
341:  ;                 Subtract number trnsfer                            *
342:  ;*********************************************************************
343:  prm_set           proc    near
344:                    push    di
345:                    push    bx
346:                    mov     bx,bp
347:                    mov     di,offset prm_area
348:                    mov     ax,cs:word ptr [func]
349:                    cmp     ax,FUNC_OPEN
350:                    jne     prm_set10
351:                    mov     ax,ss:word ptr OPEN_CHAN_ADR[bx]
352:                    mov     cs:word ptr OPEN_SET_CHAN[di],ax
353:                    mov     ax,ss:word ptr OPEN_PATH_ADR[bx]
354:                    mov     cs:word ptr OPEN_SET_PATH_O[di],ax
355:                    push    ds
356:                    pop     ax
357:                    mov     cs:word ptr OPEN_SET_PATH_S[di],ax
358:                    jmp     prm_set_end
359:  prm_set10:
360:                    mov     ax,ss:word ptr ARG_PATH_ADR_O[bx]
361:                    mov     cs:word ptr SET_PATH_ADR_O[di],ax
362:                    mov     ax,ss:word ptr ARG_PATH_ADR_S[bx]
363:                    mov     cs:word ptr SET_PATH_ADR_S[di],ax
364:                    mov     ax,cs:word ptr [func]
365:                    cmp     ax,FUNC_CLOSE
366:                    je      prm_set_end
367:                    mov     ax,ss:word ptr ARG_MODE[bx]
368:                    mov     cs:word ptr SET_MODE[di],ax
369:                    mov     ax,cs:word ptr [func]
370:                    cmp     ax,FUNC_SYNC
371:                    je      prm_set_end
372:
373:                    mov     ax,ss:word ptr ARG_ARG1_ADR_O[bx]
374:                    mov     cs:word ptr SET_ARG1_ADR_O[di],ax
375:                    push    ds
376:                    pop     ax
377:                    mov     cs:word ptr SET_ARG1_ADR_S[di],ax
378:
379:                    mov     ax,ss:word ptr ARG_ARG2_ADR_O[bx]
380:                    mov     cs:word ptr SET_ARG2_ADR_O[di],ax
381:                    push    ds
382:                    pop     ax
383:                    mov     cs:word ptr SET_ARG2_ADR_S[di],ax
384:
385:                    mov     ax,ss:word ptr ARG_ARG3_ADR_O[bx]
```

```
386:                mov     cs:word ptr SET_ARG3_ADR_O[di],ax
387:                push    ds
388:                pop     ax
389:                mov     cs:word ptr SET_ARG3_ADR_S[di],ax
390:  prm_set_end:
391:                pop     bx
392:                pop     di
393:                ret
394:  prm_set       endp
395:  ;*********************************************************************
396:  ;*            Work area for library                                  *
397:  ;*********************************************************************
398:  int_a_set_a3n db      00h
399:  int_a_set_rs4 db      00h
400:  int_a_set_net db      00h
401:  drv_nm_a3n    db      "M-A3N"
402:          db    00h
403:  drv_nm_rs4    db      "M-RS4"
404:          db    00h
405:  drv_nm_net    db      "M-MNET"
406:                db      00h
407:  int_a_no_a3n  db      00h
408:  int_a_no_rs4  db      00h
409:  int_a_no_net  db      00h
410:  handle_no     dw      0000h
411:  func          dw      ?
412:  prm_area      dw      10 dup (?)
413:  _text         ends                                        ;
414:                end                                         ;
```

```
 1:                 PAGE    80,132
 2:  ;*********************************************************************
 3:  ;*              MELSEC LIBRARY                                       *
 4:  ;*                 FOR MS-DOS (FOR LARGE MODEL)                      *
 5:  ;*                      MITSUBISHI ELECTRIC CORPORATION              *
 6:  ;*********************************************************************
 7:                 name    melsec_net_lib                          ;
 8:  melsec_lib     segment word public 'code'                      ;
 9:  assume         cs:melsec_lib                                   ;
10:
11:                 org     0                                       ;
12:  ;*********************************************************************
13:  ;*              PUBLIC DECLARE                                       *
14:  ;*                      FOR LIBRARY                                  *
15:  ;*********************************************************************
16:                 public          _nl1close                       ;
17:                 public          _nl1open                        ;
18:                 public          _nl1receive                     ;
19:                 public          _nl1send                        ;
20:                 public          _nl1sync                        ;
21:  ;*********************************************************************
22:  ;*              EQU DEFINITION                                       *
23:  ;*                      FOR LIBRARY                                  *
24:  ;*********************************************************************
25:
26:  INT_OP_CODE     equ     0cdh                                    ;
27:  INT_A_STS_END   equ     01h                                     ;
28:  INT_A_STS_EMP   equ     00h                                     ;
29:  DOS_INT         equ     21h                                     ;
30:  DRV_OPEN        equ     3dh                                     ;
31:  DRV_OPN_RD_ONLY equ     00h                                     ;
32:  IOCTL           equ     44h                                     ;
33:  IOCTL_READ      equ     02h                                     ;
34:  IOCTL_READ_SIZE equ     01h                                     ;
35:  DRV_CLOSE       equ     3eh                                     ;
36:  FUNC_OPEN       equ     0001h                                   ;
37:  FUNC_CLOSE      equ     0002h                                   ;
38:  FUNC_RECEIVE    equ     0003h                                   ;
39:  FUNC_SEND       equ     0004h                                   ;
40:  FUNC_SYNC       equ     0005h                                   ;
41:  ERR_PATH        equ     0044h                                   ;
42:  ERR_CHANEL_NO   equ     0041h                                   ;
43:  ERR_NOT_FOUND   equ     0001h                                   ;
44:
45:  OPEN_ARG_WORD   equ     3
46:  OPEN_CHAN_ADR   equ     6
47:  SYNC_ARG_WORD   equ     3
48:  SEND_ARG_WORD   equ     9
49:  REC_ARG_WORD    equ     9
50:  CLOSE_ARG_WORD  equ     2
51:  ARG_PATH_ADR    equ     6
52:  ;*********************************************************************
53:  ;*                      PROCESS ADDRESS TABLE FOR CHANNEL            *
54:  ;*********************************************************************
55:  common_adr_tbl  dw      common_a3n                              ;A3N
```

```
 56:                dw      common_rs4                          ;RS422(MELSEC)
 57:                dw      common_net                          ;MNET
 58:                dw      common_rs4                          ;RS422 (OTHER)
 59:                dw      common_rs4                          ;RS232c(OTHER)
 60:  CHANEL_MAX    equ     ($-common_adr_tbl)/2-1
 61:  ;*********************************************************************
 62:  ;*                                                                   *
 63:  ;*            bp+0  : BP                                             *
 64:  ;*            bp+2  : RETURN ADDRESS OFFSET                          *
 65:  ;*            bp+4  : RETURN ADDRESS SEGMENT                         *
 66:  ;*            bp+6  : CHANNEL NUMBER                                 *
 67:  ;*            bp+8  : PATH ADDRESS OFFSET                            *
 68:  ;*            bp+10 : PATH ADDRESS SEGMENT                           *
 69:  ;*                                                                   *
 70:  ;*********************************************************************
 71:  _nllopen      proc    far
 72:                push    bp
 73:                mov     bp,sp
 74:                push    ds
 75:                push    es
 76:                push    di
 77:                push    si
 78:                push    bx
 79:                push    cx
 80:
 81:                mov     ax,FUNC_OPEN
 82:                mov     cs:word ptr [func],ax
 83:                mov     cx,OPEN_ARG_WORD
 84:                call    prm_set
 85:        mov     ax,ss:word ptr OPEN_CHAN_ADR[bp]
 86:                cmp     ax,CHANEL_MAX
 87:                ja      nllopen_err
 88:                mov     bx,offset nllopen_ret
 89:                push    bx
 90:                add     ax,ax
 91:                mov     bx,offset common_adr_tbl
 92:                add     bx,ax
 93:                jmp     cs:word ptr [bx]
 94:  nllopen_err:
 95:                mov     ax,ERR_CHANEL_NO
 96:  nllopen_ret:
 97:                pop     cx
 98:                pop     bx
 99:                pop     si
100:                pop     di
101:                pop     es
102:                pop     ds
103:                pop     bp
104:                ret
105:  _nllopen      endp
106:  ;*********************************************************************
107:  ;*                                                                   *
108:  ;*            bp+0  : BP                                             *
109:  ;*            bp+2  : RETURN ADDRESS OFFSET                          *
110:  ;*            bp+4  : RETURN ADDRESS SEGMENT                         *
```

```
111:  ;*              bp+6  : PATH ADDRESS OFFSET                          *
112:  ;*              bp+8  : PATH ADDRESS SEGMENT                         *
113:  ;*              bp+10 : SINK MODE                                    *
114:  ;*                                                                   *
115:  ;*********************************************************************
116:  _nl1sync        proc    far
117:                  push    bp
118:                  mov     bp,sp
119:                  push    ds
120:                  push    es
121:                  push    di
122:                  push    si
123:                  push    bx
124:                  push    cx
125:                  mov     ax,FUNC_SYNC
126:                  mov     cs:word ptr [func],ax
127:                  mov     cx,SYNC_ARG_WORD
128:                  call    prm_set
129:          jmp     nl1close05
130:  _nl1sync        endp
131:  ;*********************************************************************
132:  ;*                                                                   *
133:  ;*              bp+0  : BP                                           *
134:  ;*              bp+2  : RETURN ADDRESS OFFSET                        *
135:  ;*              bp+4  : RETURN ADDRESS SEGMENT                       *
136:  ;*              bp+6  : PATH ADDRESS OFFSET                          *
137:  ;*              bp+8  : PATH ADDRESS SEGMENT                         *
138:  ;*              bp+10 : CALLING MODE                                 *
139:  ;*              bp+12 : ARG1 ADDRESS OFFSET                          *
140:  ;*              bp+14 : ARG1 ADDRESS SEGMENT                         *
141:  ;*              bp+16 : ARG2 ADDRESS OFFSET                          *
142:  ;*              bp+18 : ARG2 ADDRESS SEGMENT                         *
143:  ;*              bp+20 : ARG3 ADDRESS OFFSET                          *
144:  ;*              bp+22 : ARG3 ADDRESS SEGMENT                         *
145:  ;*                                                                   *
146:  ;*********************************************************************
147:  _nl1send         proc    far
148:                  push    bp
149:                  mov     bp,sp
150:                  push    ds
151:                  push    es
152:                  push    di
153:                  push    si
154:                  push    bx
155:                  push    cx
156:                  mov     ax,FUNC_SEND
157:                  mov     cs:word ptr [func],ax
158:                  mov     cx,SEND_ARG_WORD
159:                  call    prm_set
160:          jmp     nl1close05
161:  _nl1send         endp
162:  ;*********************************************************************
163:  ;*                                                                   *
164:  ;*              bp+0  : BP                                           *
165:  ;*              bp+2  : RETURN ADDRESS OFFSET                        *
```

```
166:  ;*              bp+4  : RETURN ADDRESS SEGMENT                        *
167:  ;*              bp+6  : PATH ADDRESS OFFSET                           *
168:  ;*              bp+8  : PATH ADDRESS SEGMENT                          *
169:  ;*              bp+10 : CALLING MODE                                  *
170:  ;*              bp+12 : ARG1 ADDRESS OFFSET                           *
171:  ;*              bp+14 : ARG1 ADDRESS SEGMENT                          *
172:  ;*              bp+16 : ARG2 ADDRESS OFFSET                           *
173:  ;*              bp+18 : ARG2 ADDRESS SEGMENT                          *
174:  ;*              bp+20 : ARG3 ADDRESS OFFSET                           *
175:  ;*              bp+22 : ARG3 ADDRESS SEGMENT                          *
176:  ;*                                                                    *
177:  ;*********************************************************************
178:  _nl1receive      proc    far
179:                  push    bp
180:                  mov     bp,sp
181:                  push    ds
182:                  push    es
183:                  push    di
184:                  push    si
185:                  push    bx
186:                  push    cx
187:                  mov     ax,FUNC_RECEIVE
188:          mov     cs:word ptr [func],ax
189:                  mov     cx,REC_ARG_WORD
190:          call    prm_set
191:          jmp     nl1close05
192:  _nl1receive      endp
193:  ;*********************************************************************
194:  ;*                                                                    *
195:  ;*              bp+0  : BP                                            *
196:  ;*              bp+2  : RETURN ADDRESS OFFSET                         *
197:  ;*              bp+4  : RETURN ADDRESS SEGMENT                        *
198:  ;*              bp+6  : PATH ADDRESS OFFSET                           *
199:  ;*              bp+8  : PATH ADDRESS SEGMENT                          *
200:  ;*                                                                    *
201:  ;*********************************************************************
202:  _nl1close        proc    far
203:                  push    bp
204:                  mov     bp,sp
205:                  push    ds
206:                  push    es
207:                  push    di
208:                  push    si
209:                  push    bx
210:                  push    cx
211:                  mov     ax,FUNC_CLOSE
212:                  mov     cs:word ptr [func],ax
213:                  mov     cx,CLOSE_ARG_WORD
214:                  call    prm_set
215:  nl1close05:
216:                  push    es
217:                  les     di,ss:dword ptr ARG_PATH_ADR[bp]
218:                  mov     al,es:byte ptr [di]
219:                  pop     es
220:                  xor     ah,ah
```

```
221:                    cmp     ax,CHANEL_MAX
222:                    ja      nllclose_err
223:                    mov     bx,offset nllclose_ret
224:                    push    bx
225:                    add     ax,ax
226:                    mov     bx,offset common_adr_tbl
227:                    add     bx,ax
228:                    jmp     cs:word ptr [bx]
229:  nllclose_err:
230:                    mov     ax,ERR_PATH
231:  nllclose_ret:
232:                    pop     cx
233:                    pop     bx
234:                    pop     si
235:                    pop     di
236:                    pop     es
237:                    pop     ds
238:                    pop     bp
239:                    ret
240:  _nllclose         endp
241:
242:
243:
244:  ;*********************************************************************
245:  ;*                                                                   *
246:  ;*                                                                   *
247:  ;*********************************************************************
248:  common_prc         proc    near
249:                    ;****************
250:                    ;*     A3N      *
251:                    ;****************
252:  common_a3n:
253:                    mov     si,offset int_a_set_a3n
254:          mov       dx,offset drv_nm_a3n
255:                    mov     di,offset int_a_no_a3n
256:                    jmp     common10
257:
258:                    ;****************
259:                    ;*     RS422    *
260:                    ;****************
261:  common_rs4:
262:                    mov     si,offset int_a_set_rs4
263:          mov       dx,offset drv_nm_rs4
264:                    mov     di,offset int_a_no_rs4
265:                    jmp     common10
266:
267:                    ;****************
268:                    ;*     MNET     *
269:                    ;****************
270:  common_net:
271:                    mov     si,offset int_a_set_net
272:          mov       dx,offset drv_nm_net
273:                    mov     di,offset int_a_no_net
274:                    jmp     common10
275:
```

```
276:  ;*********************************************************************
277:  ;*                                                                   *
278:  ;*              SI : INT-A SETTED FLAG ADDRESS OFFSET                *
279:  ;*              DX : DRIVER NAME ADDRESS OFFSET                      *
280:  ;*              DI : INT-A NUMBER SAVE AREA ADDRESS OFFSET           *
281:  ;*                                                                   *
282:  ;*********************************************************************
283:  common10:
284:                 push    cs
285:                 pop     ds
286:                 mov     al,cs:byte ptr [si]                 ;int-a set?
287:                 or      al,al                               ;
288:                 jnz     int_start                           ;YES
289:
290:                                                             ;DS = Driver name segment
291:                                                             ;DX = Driver name offset
292:                 mov     ah,DRV_OPEN                         ;AH = Function code
293:                 mov     al,DRV_OPN_RD_ONLY                  ;AL = Access code
294:                 int     DOS_INT                             ;Driver open
295:                 jc      common_err                          ;
296:
297:
298:                 mov     cs:word ptr [handle_no],ax          ;
299:
300:                                                             ;DS = Receive buffer segm

301:                 mov     dx,di                               ;DX = Receive buffer offs

302:                 mov     bx,ax                               ;BX = Handle number
303:                 mov     ah,IOCTL                            ;AH = Function code
304:                 mov     al,IOCTL_READ                       ;AL = Receive specify
305:                 mov     cx,IOCTL_READ_SIZE                  ;CX = Receive data size
306:         int     DOS_INT
307:         jc      common_err
308:
309:                 mov     cs:byte ptr [si],INT_A_STS_END
310:
311:                 mov     bx,cs:word ptr [handle_no]          ;
312:                 mov     ah,DRV_CLOSE                        ;
313:                 int     DOS_INT                             ;
314:                 jc      common_err
315:  int_start:
316:                 mov     al,cs:byte ptr [di]
317:                 mov     cs:byte ptr [int_code],al
318:                 push    cs
319:                 pop     es
320:                 mov     bp,offset prm_area
321:                 mov     bx,bp
322:                 mov     ax,cs:word ptr [func]
323:                 db      INT_OP_CODE
324:  int_code       db      00h
325:                 jmp     common_end
326:  common_err:
327:                 mov     ax,ERR_NOT_FOUND
328:  common_end:
329:                 ret
330:  common_prc      endp
```

```
331:
332:  ;*********************************************************************
333:  ;               Subtract number trnsfer                              *
334:  ;               CX = Trnsfer word number                             *
335:  ;*********************************************************************
336:  prm_set         proc    near
337:                  push    di
338:                  push    bx
339:                  mov     bx,bp
340:                  mov     di,offset prm_area
341:  prm_set_loop:
342:                  mov     ax,ss:word ptr 6[bx]
343:                  mov     cs:word ptr [di],ax
344:                  add     bx,2
345:                  add     di,2
346:                  loop    prm_set_loop
347:                  pop     bx
348:                  pop     di
349:                  ret
350:  prm_set         endp
351:  ;*********************************************************************
352:  ;*              Work area for library                                *
353:  ;*********************************************************************
354:  int_a_set_a3n   db      00h
355:  int_a_set_rs4   db      00h
356:  int_a_set_net   db      00h
357:  drv_nm_a3n      db      "M-A3N"
358:          db      00h
359:  drv_nm_rs4      db      "M-RS4"
360:          db      00h
361:  drv_nm_net      db      "M-MNET"
362:                  db      00h
363:  int_a_no_a3n    db      00h
364:  int_a_no_rs4    db      00h
365:  int_a_no_net    db      00h
366:  handle_no       dw      0000h
367:  func            dw      ?
368:  prm_area        dw      10 dup (?)
369:  melsec_lib      ends                                        ;
370:                  end                                         ;
```

IMPORTANT

The components on the printed circuit boards will be damaged by static electricity, so avoid handling them directly. If it is necessary to handle them take the following precautions.

(1) Ground human body and work bench.

(2) Do not touch the conductive areas of the printed circuit board and its electrical parts with any non-grounded tools etc.

Under no circumstances will Mitsubishi Electric be liable or responsible for any consequential damage that may arise as a result of the installation or use of this equipment.

All examples and diagrams shown in this manual are intended only as an aid to understanding the text, not to guarantee operation. Mitsubishi Electric will accept no responsibility for actual use of the product based on these illustrative examples.

Owing to the very great variety in possible applications of this equipment, you must satisfy yourself as to its suitability for your specific application.

MITSUBISHI ELECTRIC CORPORATION

HEAD OFFICE: MITSUBISHI DENKI BLDG MARUNOUCHI TOKYO 100 TELEX: J24532 CABLE MELCO TOKYO
NAGOYA WORKS: 1-14, YADA-MINAMI 5, HIGASHI-KU, NAGOYA, JAPAN

When exported from Japan, this manual does not require application to the Ministry of International Trade and Industry for service transaction permission.

IB (NA) 66253-A (9006) MEE Printed in Japan