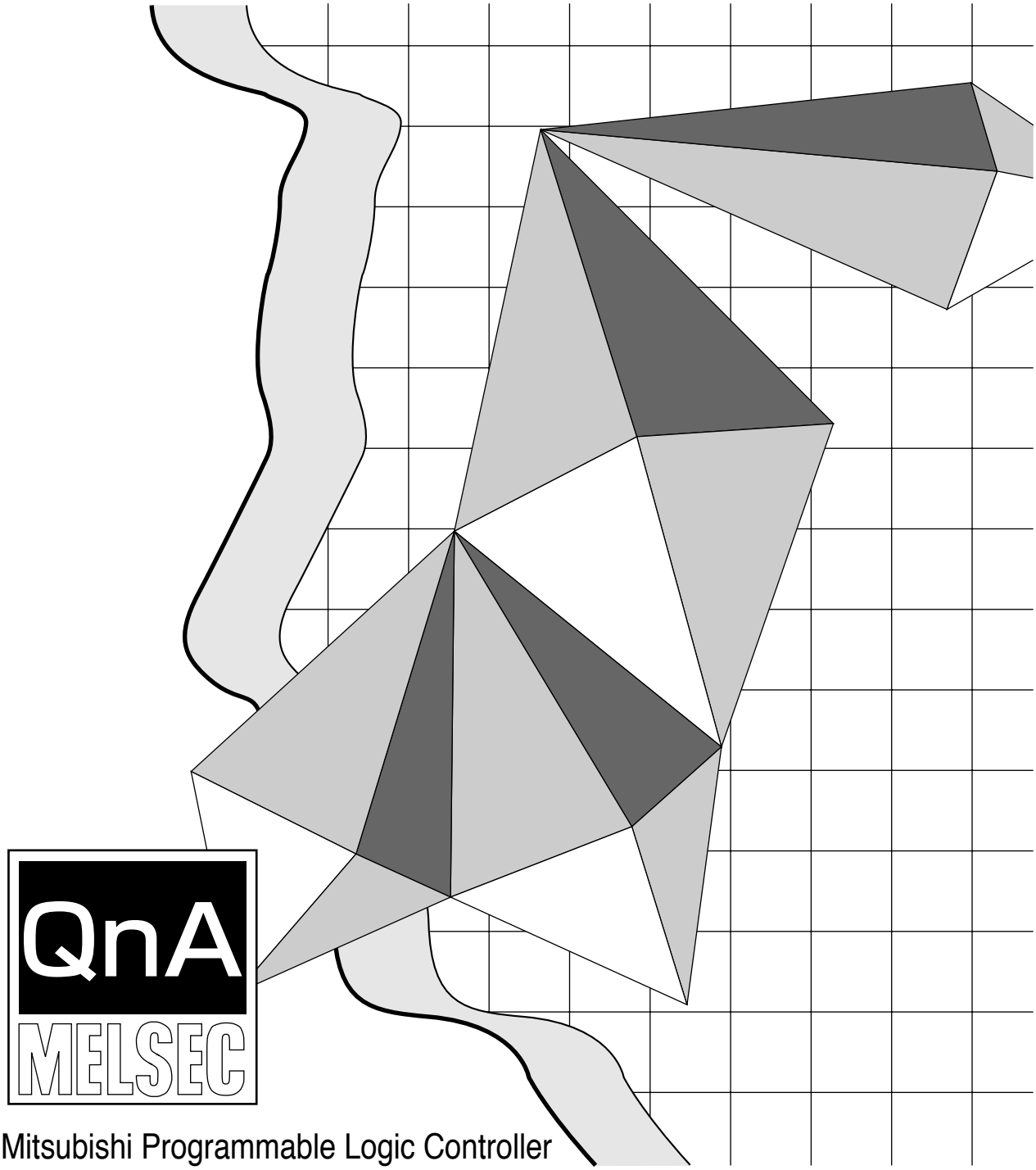
MITSUBISHI

QnA SERIES

QnACPU

Programming Manual (Fundamentals)



Mitsubishi Programmable Logic Controller

SAFETY CAUTIONS

(You must read these cautions before using the product)

In connection with the use of this product, in addition to carefully reading both this manual and the related manuals indicated in this manual, it is also essential to pay due attention to safety and handle the product correctly.
The safety cautions given here apply to this product in isolation. For information on the safety of the PLC system as a whole, refer to the CPU module User's Manual.
Store this manual carefully in a place where it is accessible for reference whenever necessary, and forward a copy of the manual to the end user.

[System Design Precautions]

DANGER

- Safety circuits should be installed external to the programmable controller to ensure that the system as a whole will continue to operate safely in the event of an external power supply malfunction or a programmable controller failure. Erroneous outputs and operation could result in an accident.
  1) The following circuitry should be installed outside the programmable controller:
     Interlock circuitry for the emergency stop circuit protective circuit, and for reciprocal operations such as forward/reverse, etc., and interlock circuitry for upper/lower positioning limits, etc., to prevent machine damage.
  2) When the programmable controller detects an abnormal condition, processing is stopped and all outputs are switched OFF. This happens in the following cases:
     - When the power supply module's over-current or over-voltage protection device is activated.
     - When an error (watchdog timer error, etc.) is detected at the PC CPU by the self-diagnosis function.

     Some errors, such as input/output control errors, cannot be detected by the PC CPU, and there may be cases when all outputs are turned ON when such errors occur. In order to ensure that the machine operates safely in such cases, a failsafe circuit or mechanism should be provided outside the programmable controller. Refer to the CPU module user's manual for an example of such a failsafe circuit.
  3) Outputs may become stuck at ON or OFF due to an output module relay or transistor failure. An external circuit should therefore be provided to monitor output signals whose incorrect operation could cause serious accidents.

- A circuit should be installed which permits the external power supply to be switched ON only after the programmable controller power has been switched ON. Accidents caused by erroneous outputs and motion could result if the external power supply is switched ON first.

- When a data link communication error occurs, the status shown below will be established at the faulty station. In order to ensure that the system operates safely at such times, an interlock circuit should be provided in the sequence program (using the communication status information).
  Erroneous outputs and operation could result in an accident.
  1) The data link data which existed prior to the error will be held.
  2) All outputs will be switched OFF at MELSECNET (II, /B, /10) remote I/O stations.
  3) At the MELSECNET/MINI-S3 remote I/O stations, all outputs will be switched OFF or output statuses will be held, depending on the E.C. mode setting.

  For details on procedures for checking faulty stations, and for operation statuses when such errors occur, refer to the appropriate data link manual.

[System Design Precautions]

⚠ CAUTION

- Do not bundle control lines or communication wires together with main circuit or power lines, or lay them close to these lines.
  As a guide, separate the lines by a distance of at least 100 mm, otherwise malfunctions may occur due to noise.

[Cautions on Mounting]

⚠ CAUTION

- Use the PC in an environment that conforms to the general specifications in the manual.
  Using the PC in environments outside the ranges stated in the general specifications will cause electric shock, fire, malfunction, or damage to/deterioration of the product.
- Make sure that the module fixing projection on the base of the module is properly engaged in the module fixing hole in the base unit before mounting the module.
  Failure to mount the module properly will result in malfunction or failure, or in the module falling.
- Extension cables should be securely connected to base unit and module connectors. Check for loose connection after installation.
  A poor connection could result in contact problems and erroneous inputs/outputs.
- Plug the memory cassette firmly into the memory cassette mounting connector. Check for loose connection after installation.
  A poor connection could result in erroneous operation.
- Plug the memory firmly into the memory socket. Check for loose connection after installation.
  A poor connection could result in erroneous operation.

[Cautions on Wiring]

DANGER

- Switch off the external power supply before staring installation and wiring work.
  Failure to do so could result in electrical shocks and equipment damage.
- After installation and wiring is completed, be sure to attach the terminal cover before switching the power ON and starting operation.
  Failure to do so could result in electrical shocks.

CAUTION

- Be sure to ground the FG and LG terminals, carrying out at least class 3 grounding work with a ground exclusive to the PC.
  Otherwise there will be a danger of electric shock and malfunctions.
- Carry out wiring to the PC correctly, checking the rated voltage and terminal arrangement of the product.
  Using a power supply that does not conform to the rated voltage, or carrying out wiring incorrectly, will cause fire or failure.
- Outputs from multiple power supply modules should not be connected in parallel. Failure to do so could cause the power supply module to overheat, resulting in a fire or module failure.
- Tighten the terminal screws to the stipulated torque.
  Loose screws will cause short circuits, fire, or malfunctions.
- Make sure that no foreign matter such as chips or wiring offcuts gets inside the module.
  It will cause fire, failure or malfunction.
- Connectors for external connections should be crimped, pressure welded, or soldered in the correct manner using the correct tools.
  For details regarding crimping and pressure welding tools, refer to the input/output module user's manual.
  A poor connection could cause shorts, fire, and erroneous operation.

[Cautions on Startup and Maintenance]

⚠ DANGER

- Do not touch terminals while the power is ON.
  This will cause malfunctions.
- Make sure that the battery is connected properly. Do not attempt to charge or disassemble the battery, do not heat the battery or place it in a flame, and do not short or solder the battery.
  Incorrect handling of the battery can cause battery heat generation and ruptures which could result in fire or injury.
- Switch the power off before cleaning or re-tightening terminal screws.
  Carrying out this work while power is ON will cause failure or malfunction of the module.

⚠ CAUTION

- In order to ensure safe operation, read the manual carefully to acquaint yourself with procedures for program changes, forced outputs, RUN, STOP, and PAUSE operations, etc., while operation is in progress.
  Incorrect operation could result in machine failure and injury.
- Do not disassemble or modify any module.
  This will cause failure, malfunction, injury, or fire.
- Switch the power OFF before mounting or removing the module.
  Mounting or removing it with the power ON can cause failure or malfunction of the module.
- When replacing fuses, be sure to use the prescribed fuse. A fuse of the wrong capacity could cause a fire.
- Do not drop or add an impact to the battery to be mounted in the module.
  Otherwise the battery will be broken, possibly causing internal leakage of electrolyte.
  Do not use but dispose of the battery if it has fallen or an impact is given to it.
- Before touching the module, always touch grounded metal, etc. to discharge static electricity from human body.
  Failure to do so can cause the module to fail or malfunction.

[Cautions on Disposal]

⚠ CAUTION

- Dispose of this product as industrial waste.

REVISIONS

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

Print Date	*Manual Number	Revision
Jun., 1996	IB (NA) 66614-A	First edition
Dec., 1997	IB (NA) 66614-B	Standardization of terms for the QnA series Internal Memory → Internal RAM IC Memory card → Memory card Q2AS(H)CPU(-S1) and Q4ARCPU added. Correction About Manuals Sections 1.1, 1.2, 1.3, 2.2.1, 2.2.2, 2.2.3, 2.3.2, 2.3.3, 2.4, 3.1.1, 3.1.3, 3.2, 3.2.3, 3.2.4, 3.3.2, 4.1, 4.1.1, 4.2, 4.2.7, 4.2.13, 4.3.3, 4.4, 4.5, 4.7, 4.10, 4.13.1, 5 Addition Sections 2.6.1, 4.2.1, 4.2.10, 4.3.1 Page 3-20, 3-21
Sep., 1998	IB (NA) 66614-C	Correction Contents, Section 2.2.1, 2.2.2, 2.3, 2.3.3, 2.5, 3.1.2, 3.1.3, 4.1.1, 4.2.6, 4.6, 4.7, 4.13.1
Oct., 2000	IB (NA) 66614-D	Correction Section 2.3.3, 3.2.1, 3.3.2, 4.2.10, 4.2.11, 4.10, 4.13.1
Dec., 2002	IB (NA) 66614-E	Correction Section 4.2.10
Dec., 2003	IB (NA) 66614-F	Correction Section 2.2.1, 2.2.2, 2.2.3, 2.3.2, 4.2.11 Addition WARRANTY
Dec., 2005	IB (NA) 66614-G	Correction Section 2.5, 3.1.3, 3.2.4, 4.2.10 Chapter 5, Section 6.1.2, 6.2.2
Mar., 2006	IB (NA) 66614-H	Correction Section 4.2.1

Japanese Manual Version SH-3540-I

INTRODUCTION

Thank you for choosing the Mitsubishi MELSEC-QnA Series of General Purpose Programmable Controllers. Please read this manual carefully so that the equipment is used to its optimum. A copy of this manual should be forwarded to the end User.

CONTENTS

About Manuals

The manuals related to the QnACPU are listed in the table below.
Please order those you require.

Related Manuals

Manual Name	Manual Number
QnACPU Guidebook Aimed at people using QnACPU for the first time. Describes procedures for everything from creating programs and writing created programs to the CPU, to debugging. Also describes how to use the QnACPU most effectively.	IB-66606 (13FJ10)
Q2A(S1)/Q3A/Q4ACPU User's Manual Describes the performance, functions, and handling of the Q2ACPU(S1), Q3ACPU, and Q4ACPU, and the specifications and handling of memory cards and base units. (Purchased separately)	IB-66608 (13J821)
Q4ARCPU User's Manual Describes the performance, functions, handling, etc., of the Q4ARCPU, and also the specifications and handling of bus switching modules, system control modules, power supply module, memory card, and base units. (Purchased separately)	IB-66685 (13J852)
Q2AS(H)CPU(S1) User's Manual Describes the performance, functions, handling, etc., of the Q2ASCPU, Q2ASCPU-S1, Q2ASHCPU, and Q2ASHCPU-S1, and the specifications and handling of power supply modules, memory cards and base units. (Purchased separately)	SH-3599 (BJ858)
QCPU (Q mode)/QnACPU Programming Manual (Common Instructions) Describes how to use sequence instructions, basic instructions, and application instructions. (Purchased separately)	SH-080039 (13JF58)
QnACPU Programming Manual (Special Function module) Describes the dedicated instructions for special function modules available when using the Q2ACPU(S1), Q3ACPU, and Q4ACPU. (Purchased separately)	SH-4013 (13JF56)
QnACPU Programming Manual (AD57 Instructions) Describes the dedicated instructions for controlling an AD57(S1) type CRT controller module available when using the Q2ACPU(S1), Q3ACPU, or Q4ACPU. (Purchased separately)	IB-66617 (13JF49)
QCPU (Q mode)/QnACPU Programming Manual (PID Control Instructions) Describes the dedicated instructions for PID control available when using the Q2ACPU(S1), Q3ACPU, or Q4ACPU. (Purchased separately)	SH-080040 (13JF59)
QCPU (Q mode)/QnACPU Programming Manual (SFC) Describes the performance specifications, functions, programming, debugging, and error codes, for SFC program. (Purchased separately)	SH-080041 (13JF60)
For QnA/Q4AR MELSECNET/10 Network System Reference Manual Describes the general concept, specifications, and part names and settings, for MELSECNET/10. (Purchased separately)	IB-66690 (13JF78)
MELSECNET, MELSECNET/B Data Link System Reference Manual Describes the general concept, specifications, and part names and settings, for MELSECNET (II), MELSECNET/B. (Purchased separately)	IB-66350 (13JF70)
GX Developer Version8 Operating Manual Describes the online functions of GX Developer including the programming procedure, printing out procedure, monitoring procedure, and debugging procedure. (Purchased separately)	IB-0800243E
Type SW2IVD-GPPQ GPP Function Operating Manual (Offline) Describes the how to create programs and print out data when using SW2IVD-GPPQ, and the offline functions of SW2IVD-GPPQ such as file maintenance. (Supplied with the product)	IB-66774 (13J921)
Type SW2IVD-GPPQ GPP Function Operating Manual (Online) Describes the online functions of SW2IVD-GPPQ, including the methods for monitoring and debugging. (Supplied with the product)	IB-66775 (13J922)
Type SW2IVD-GPPQ GPP Function Operating Manual (SFC) Describes SFC functions such as SFC program editing and monitoring. (Supplied with the product)	IB-66776 (13J923)
Type SW2IVD-GPPQ GPP Software package Operating Manual (Q6TEL) Describes the system configuration, operation methods, etc. (Supplied with the product)	IB-66777

1. GENERAL DESCRIPTION

This manual describes the required program types, the I/O processing, the devices, etc., when programming for the following CPU modules in the MELSEC QnA series.

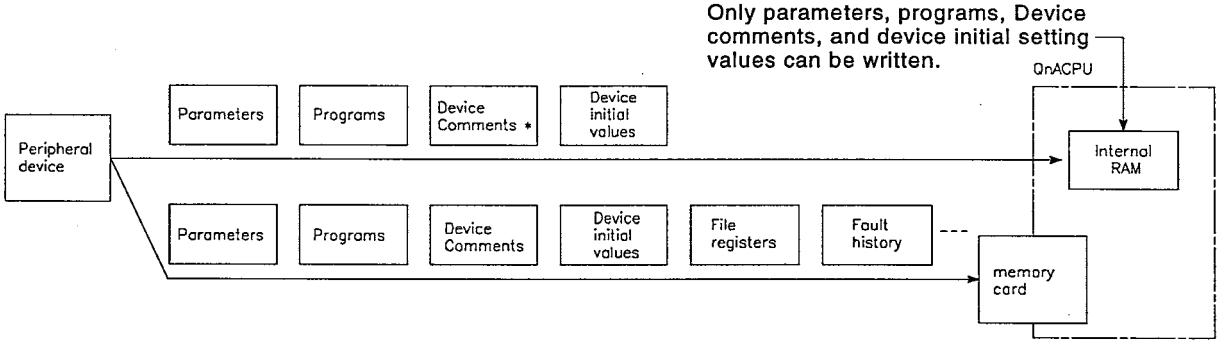
- Q2AS(H)CPU(S1)
- Q2ACPU(S1)
- Q3ACPU
- Q4ACPU
- Q4ARCPU

All of the above modules are referred to generically in this manual as "QnACPU".

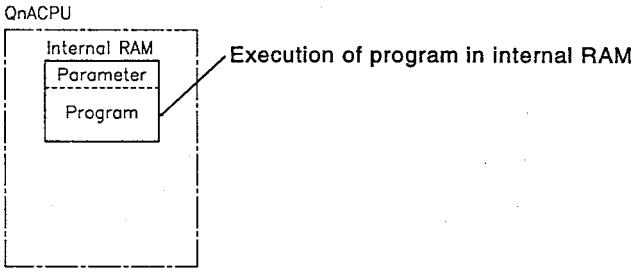
1.1 Programs

(1) Program management by memory card is possible

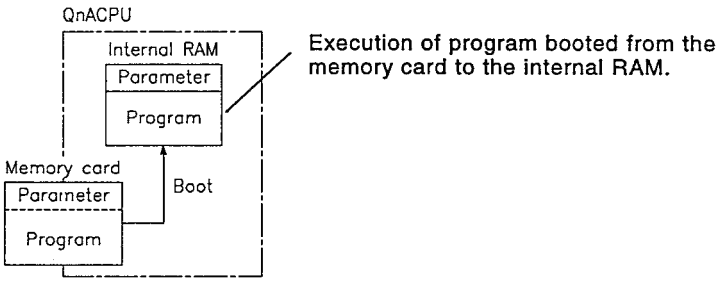
(a) Programs created at a peripheral device can be stored in the QnACPU's internal RAM or memory card. Parameters, programs, device comments, and device initial setting values can be stored in the internal RAM and memory card. Other file register and fault history data, etc., can be stored in the memory card only (storage in the internal RAM is not possible. See Section 2.4). Note that device comments ("*" in figure below) stored in the internal RAM cannot be used in the instructions of the program currently being executed.



(b) The QnACPU processes programs which are stored in the internal RAM.

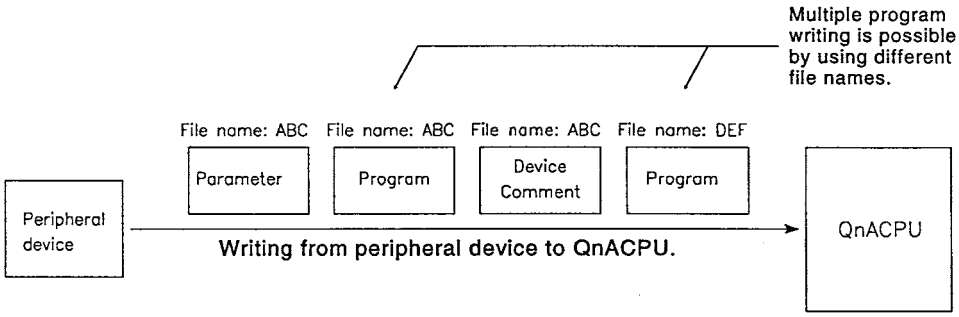


Programs stored in the memory card can be executed only after they are first read to (booted to) the QnACPU internal RAM. (Programs to be read to the QnACPU are designated by parameter settings, and the boot operation is designated by a DIP switch setting at the QnACPU.)



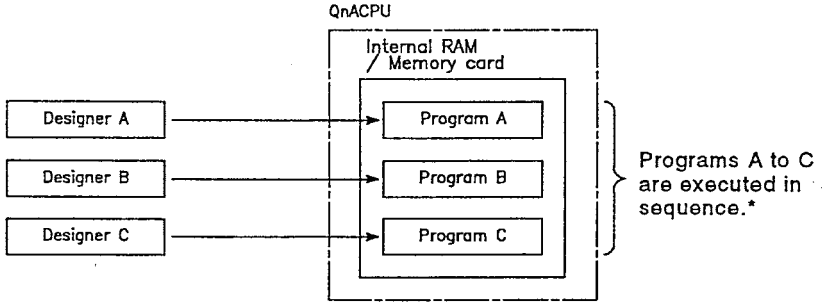
(2) Program construction

QnACPU programs are stored in a file format in the internal RAM or memory card. Multiple programs can therefore be stored in the internal RAM and memory card by using different file names.

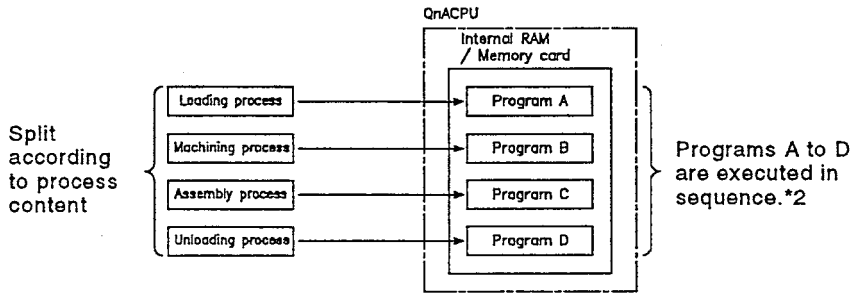


This format permits the program creation operation to be split among several designers, and allows program management and maintenance to be carried out according to the process or function in question. Moreover, revision and debugging is required only at the relevant programs when the specifications are changed.

(a) Example of program creation split among several designers:



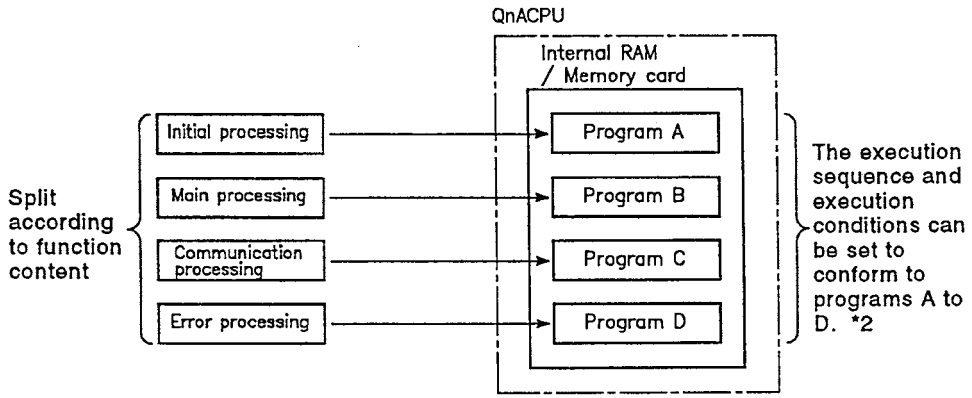
(b) Example of programs split according to process:



REMARKS

1. *: See Section 3.2 for details regarding the execution sequence.

(c) Example of programs split according to function:



REMARKS

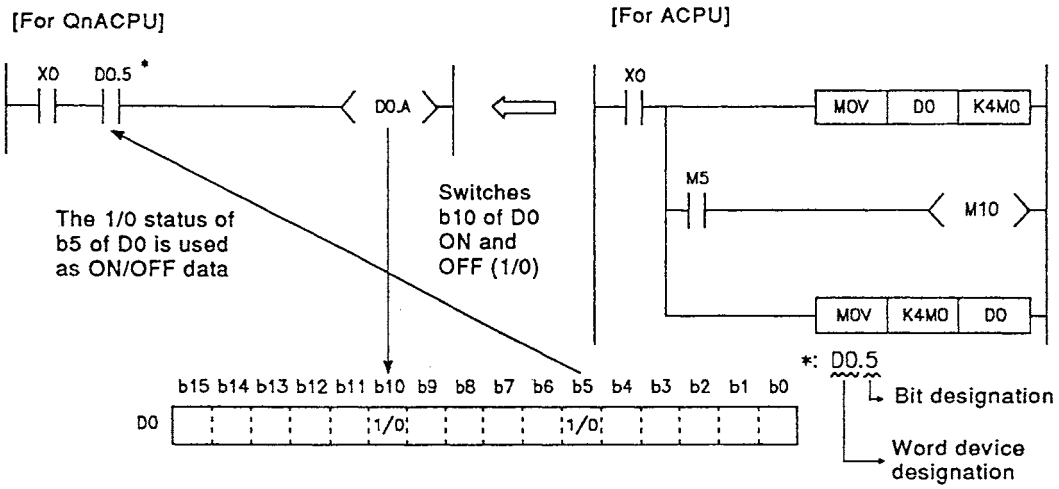
1. *1: Programs split according to process can be further split according to function.
2. *2: See Section 3.2 for details regarding the execution sequence and execution conditions.

1.2 Convenient Programming Devices and Instructions

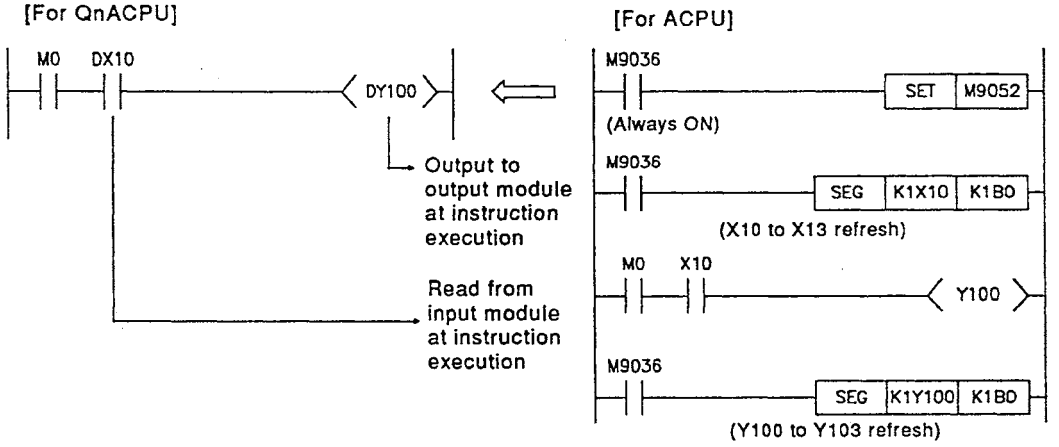
The QnACPU features devices and instructions which facilitate program creation. A few of these are described below.

(1) Flexible device designation

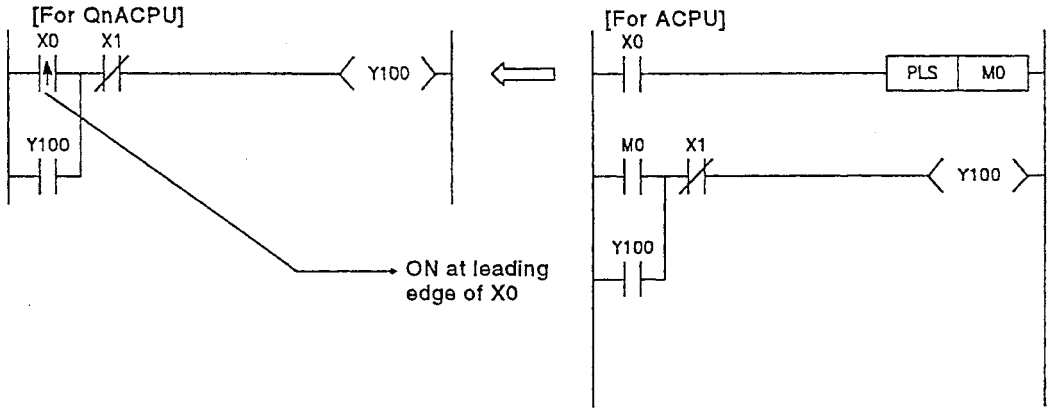
(a) Word device bits can be designated to serve as contacts or coils.



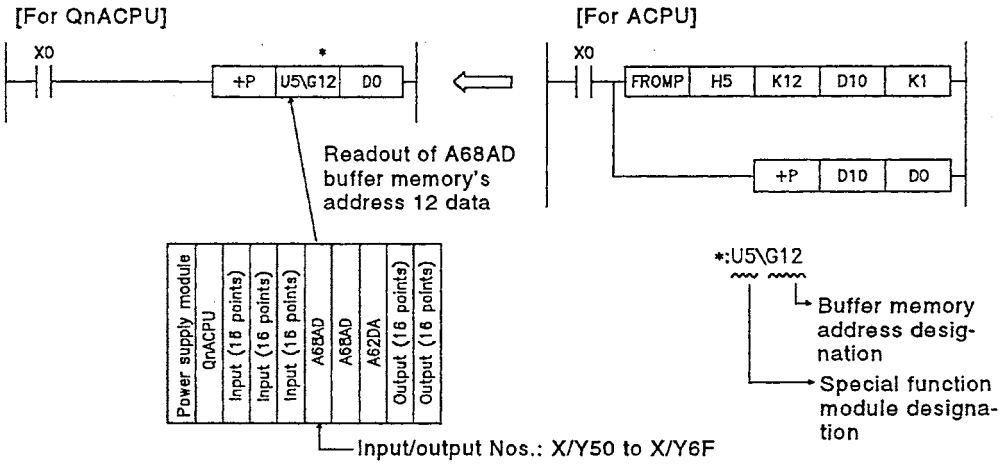
(b) Direct processing in 1-point units is possible within a program simply by using direct access inputs (DX□) and direct access outputs (DY□).



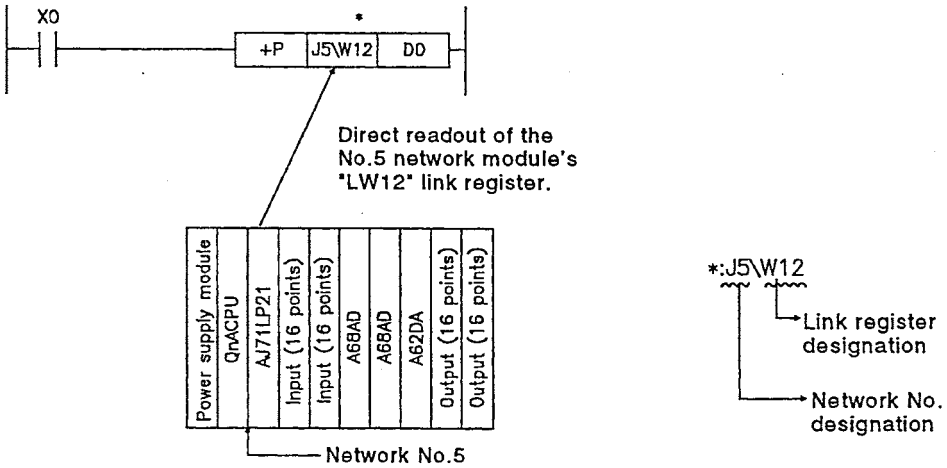
(c) Differential contacts (⊣↑⊢/⊣↓⊢) eliminate the need for converting inputs to pulses.



(d) The special function module's buffer memory can be used in the same way as devices when programming.



(e) Direct access to link devices (LX, LY, LB, LW, LSB, LSW) of MELSECNET/10 network modules is possible without refresh settings.

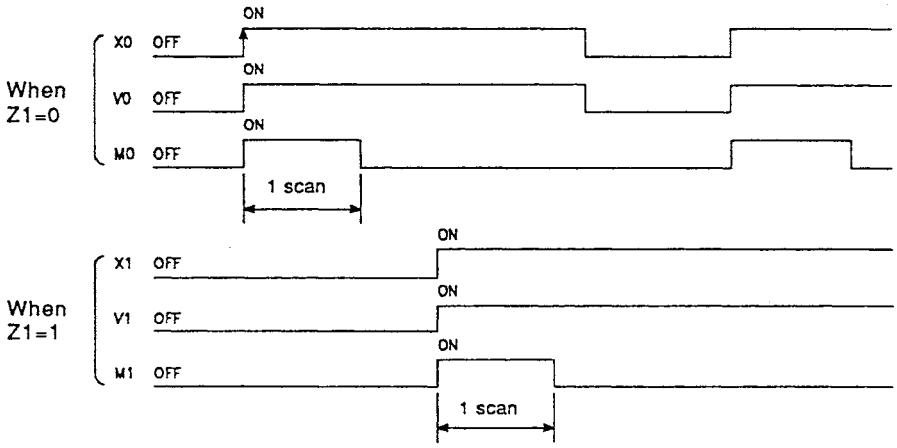


(2) Edge relays simplify pulse conversion processing

(a) The use of a relay (V) that comes ON at the leading edge of the input condition simplifies pulse processing when a contact index qualification has been made.

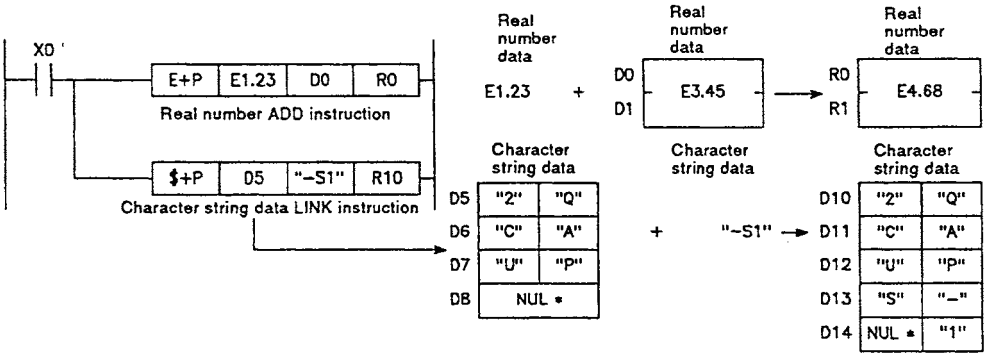
[Circuit example]

[Timing chart]



(3) Simpler data processing

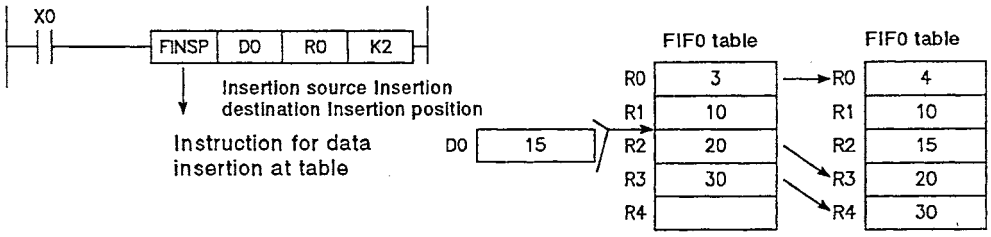
(a) Real numbers (floating decimal point data) and character string constants can be used in the programming as they are.



REMARK

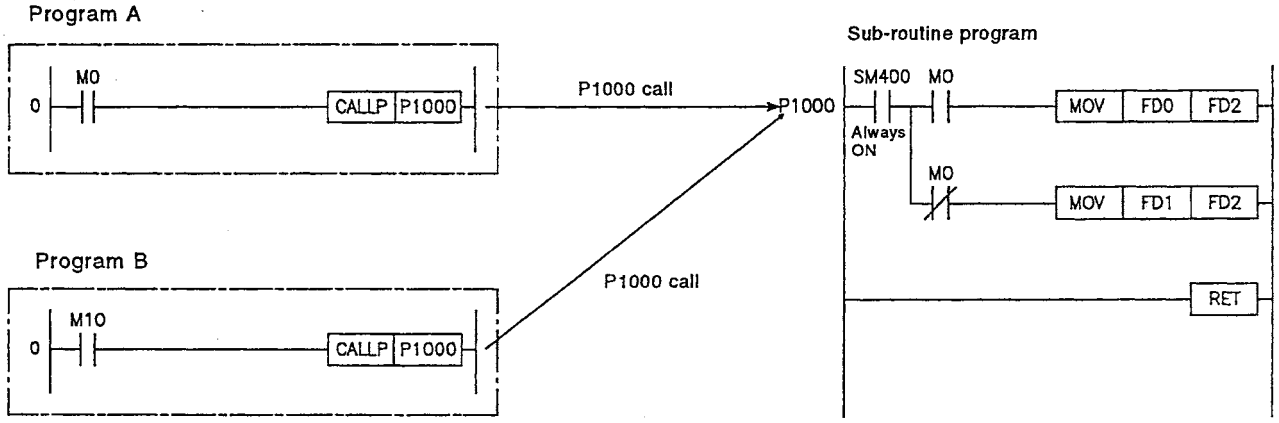
1. *: NUL indicates "00H" (character string END).

(b) Data processing instructions such as table processing instructions, etc., enable high-speed processing of large amounts of data.

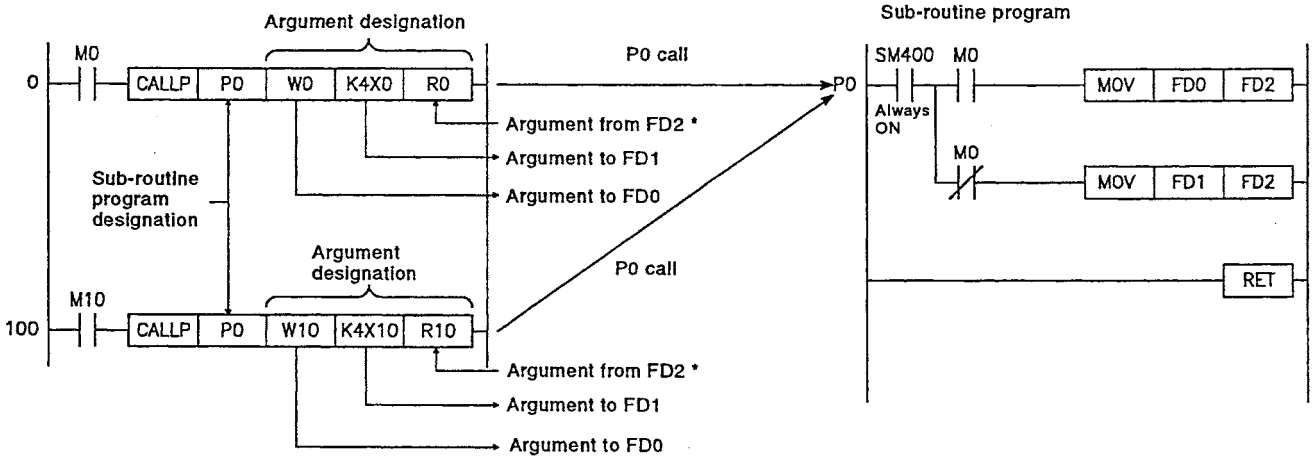


(4) Easy shared use of sub-routine programs

(a) A common pointer can be used to call the same sub-routine program from all sequence programs being executed.



(b) The use of sub-routine call instructions with arguments simplifies the creation of sub-routine programs which are called several times.



REMARK

1. The QnACPU automatically determines the argument input/output condition.

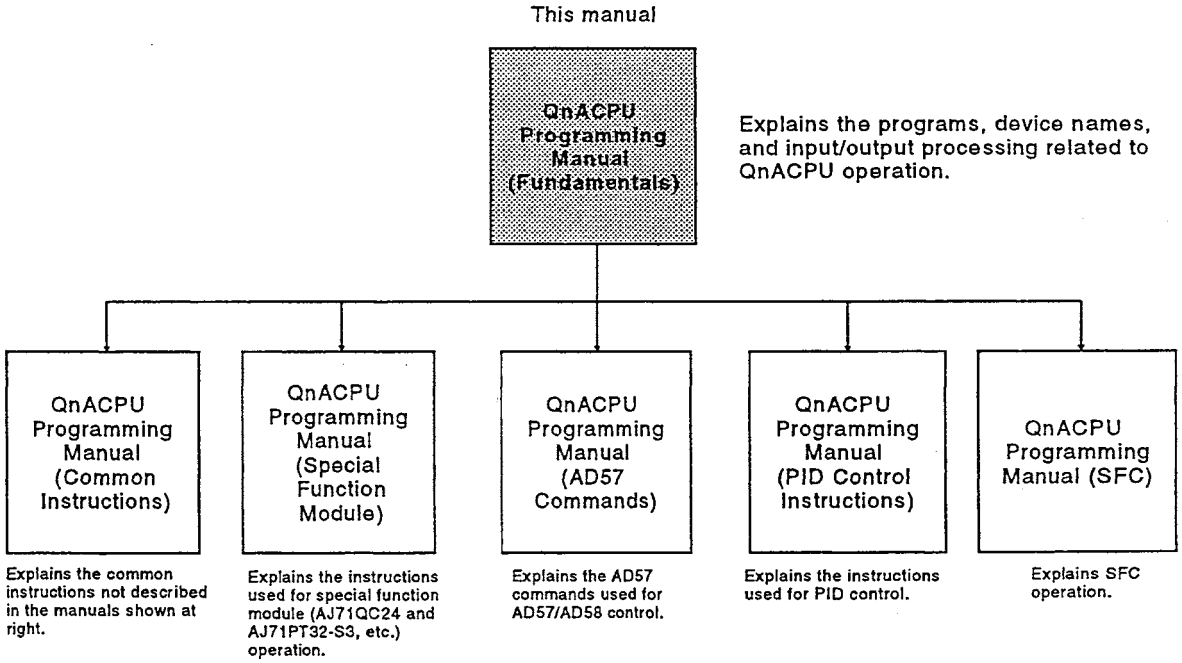
- Sub-routine program "source" data is processed as input data to the sub-routine program.
- Sub-routine program "destination" data is processed as output data from the sub-routine program.

1.3 Related Programming Manuals

In addition to this manual, the 5 manuals shown below also contain information regarding instructions used in QnACPU operation.

- QnACPU Programming Manual (Common Instructions)
- QnACPU Programming Manual (Special Function Module)
- QnACPU Programming Manual (PID Control Instructions)
- QnACPU Programming Manual (AD57 Commands)
- QnACPU Programming Manual (SFC)

Use this manual for information regarding QnACPU programs, devices, and input/output processing, etc., and refer to the other manuals for information regarding the instructions which are used.



REMARK

1) For Q4ARCPU, apart from the programming manual indicated above, there is also the Q4ARCPU Programming Manual (Application PID Edition).

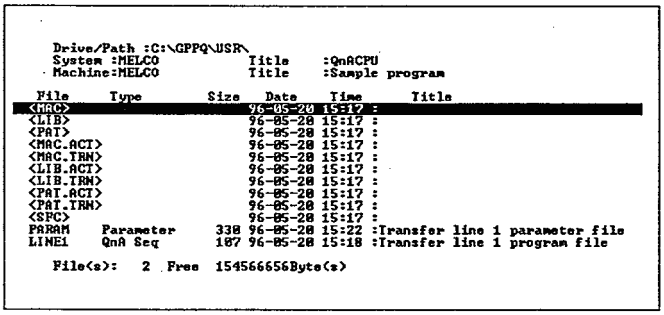
2. QnACPU FILES

Parameter, program, and comment data, etc., are assigned file names and extension names, and are then stored in the QnACPU internal RAM or memory card.
When writing this data from a peripheral device to the QnACPU, the files to be written are specified by their type (parameter, program, comment, etc.) rather than by their extension names.
(The peripheral device automatically assigns the appropriate extension name for the file type which has been specified.)
The use of different file and extension names permits multiple files to be stored in the QnACPU.

Because the QnACPU can also process a given program as one file, programs created can be managed individually according to their "designer", "process", or "function" by using different program file names. Moreover, program execution is possible for multiple programs stored at the QnACPU. (See Chapter 3 for QnACPU program execution details.)

The QnACPU stores files in the available areas in the internal RAM and memory card.
If the continuous available memory area is insufficient to accommodate a file for which a writing request has occurred (from the peripheral device), writing to the internal RAM and card will be impossible. (See Section 2.6.2)

A file name, file size, and creation date will be appended to each file.
The file list shown below is displayed at the peripheral device.



The file list display items are explained below.

(1) File name
The file name consists of the file name (max. 8 chars.) and the extension (3 chars.) The QnACPU distinguishes between upper case and lower case characters. (At the peripheral device, all characters are converted to upper case characters.)
An extension name which corresponds to the file type designated when the file was written in the peripheral device is automatically appended to the file name.

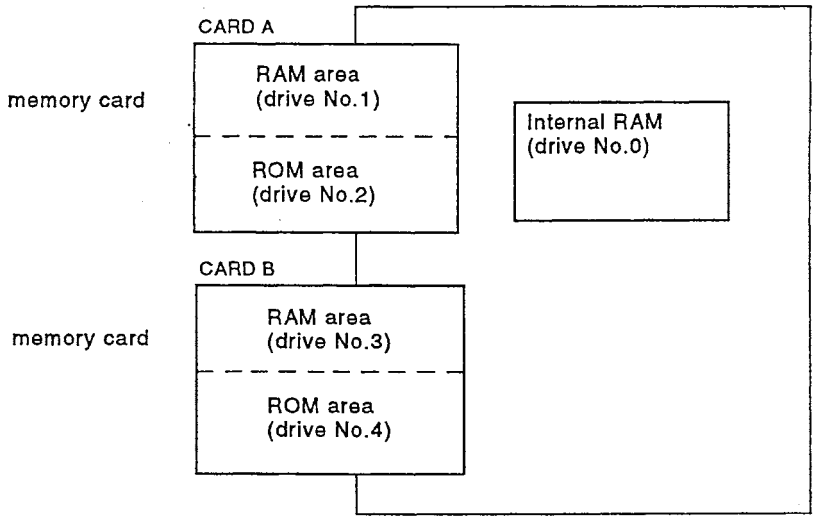
(2) Size
The file size is indicated in byte units.
Files are stored in the internal RAM in 4-byte units (1 step), and at the memory card in 1-byte units. When calculating a file's size, please note that at least 64 bytes (132 bytes for programs) will be added to all user created files other than file registers.

(3) Data & time
The date & time when the file was written from the peripheral device to the QnACPU is indicated.

(4) Title
Indicates the user file application, etc. (max. of 32 chars.)

2.1 QnACPU Internal RAM & Memory Cards

Designating the internal RAM & memory cards
The QnACPU features 2 types of file storage area: the internal RAM, and the memory cards, with each memory area being assigned a drive No.*
The internal RAM is assigned drive No.0, the "CARD A" memory card is assigned drive Nos.1 & 2, and the "CARD B" memory card is assigned drive Nos.3 & 4.



REMARK

1) *: When writing parameter data and programs, etc., from the peripheral device to the QnACPU, the memory to which the data is written (internal RAM/memory card) is designated by the drive number.

2.2 Internal RAM

The memory map and memory size of the QnACPU internal RAM are described in this section.

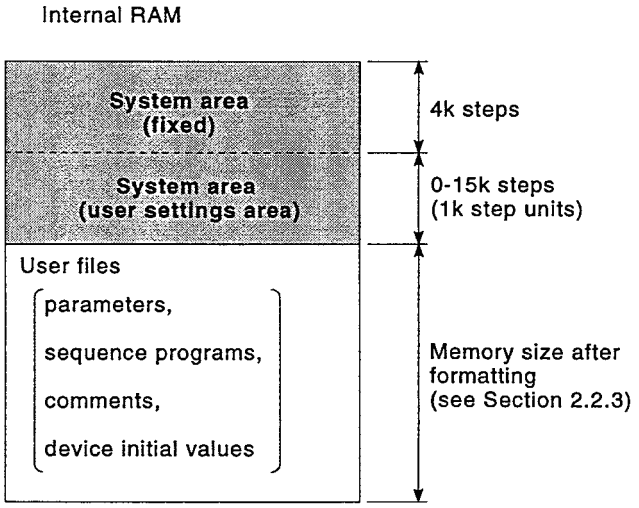
POINTS

(1) Before the QnACPU internal RAM can be used for the first time, it must be formatted. For details regarding the internal RAM formatting procedure, refer to the "SW□IVD/NX-GPPQ Type GPP Function Software Package Operating Manual (Online)."

(2) Program files are stored in the internal RAM in 1k step units.

2.2.1 Memory map

Files are stored in the internal RAM in the following format.



2.2.2 Formatting precautions

The QnACPU internal RAM can only be used after being formatted at the peripheral device.
When formatting the internal RAM, designate whether or not a system area is to be allocated for user settings. Up to 15 k steps (in 1-step units) can be allocated for the user setting system area.

The system area user setting data is used for communication with the serial communication unit, and for registering monitor data from peripheral devices connected to other stations in the network.
Although the designation of a user setting area speeds up monitoring from the serial communication unit and other network stations, However, note that the amount of space available for user files.
he capacity for the user files is reduced if the user setting area is reserved in the system area.

2.2.3 Memory capacity after formatting

The memory capacity available for user files varies according to the capacity reserved for the user setting area of the system area.
The memory capacity available for user files with formatted internal RAM is shown in Table 2.1.
The memory capacity available for user files can be checked in the peripheral device file list.

Table 2.1 Memory capacity that can be used for user files after formatting

CPU Model Name	Memory Capacity
Q2ACPU, Q2AS(H)CPU	28k steps (114688 bytes) to 13 k steps (53248 bytes)
Q2ACPU-S1, Q2AS(H)CPU-S1	60k steps (245760 bytes) to 45 k steps (184320 bytes)
Q3ACPU	92k steps (376832 bytes) to 77 k steps (315392 bytes)
Q4ACPU	124k steps (507904 bytes) to 109 k steps (446464 bytes)

2.3 Memory Card

The memory map and memory capacity of the QnACPU memory card are described in this section.

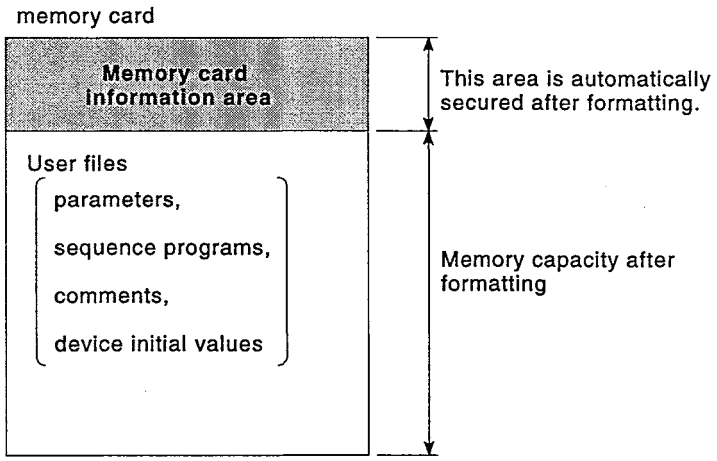
POINTS

(1) Before the QnACPU memory card can be used for the first time, it must be formatted by a peripheral device. Parameter data and program files, etc., cannot be stored at the memory card until it has been formatted. For details regarding the memory card formatting procedure, refer to the "SW□IVD/NX-GPPQ Type GPP Function Software Package Operating Manual (Online)."

(2) Program files are stored in the memory card in 512 bytes (128 steps) step units.

2.3.1 Memory map

Files are stored in the memory card in the following format.



2.3.2 Memory capacity after formatting

After the memory card has been formatted, the memory card capacity which is displayed at the peripheral device's file list display will be as shown below.

Table 2.2 Memory Size after Formatting

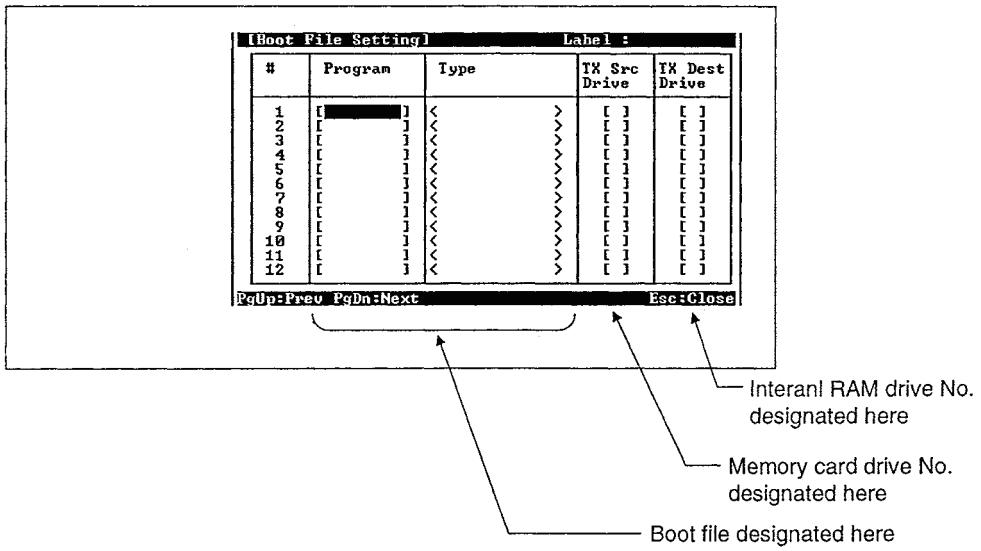
Memory Card Model Name	Memory Capacity (k-bytes)			Max. Number of Files Stored		
	SRAM	E^2PROM	Flash Memory	SRAM	E^2PROM	Flash Memory
Q1MEM-64S	59	—	—	118	—	—
Q1MEM-128S	123	—	—	128	—	—
Q1MEM-256S	250.5	—	—	128	—	—
Q1MEM-512S	506	—	—	128	—	—
Q1MEM-1MS	1016.5	—	—	128	—	—
Q1MEM-2MS	2036.0	—	—	256	—	—
Q1MEM-64SE	28.5	29.0	—	57	58	—
Q1MEM-128SE	58.5	59.0	—	117	118	—
Q1MEM-256SE	122.5	123.0	—	128	128	—
Q1MEM-512SE	250.0	250.5	—	128	128	—
Q1MEM-1MSE	505.5	506.0	—	128	128	—
Q1MEM-256SR	122.5	—	*	128	—	128
Q1MEM-512SR	250.0	—	*	128	—	128
Q1MEM-1MSR	505.5	—	*	128	—	128
Q1MEM-2MSR	1016.0	—	*	128	—	128

* Depends on the specifications of the memory card reader/writer that does the formatting.

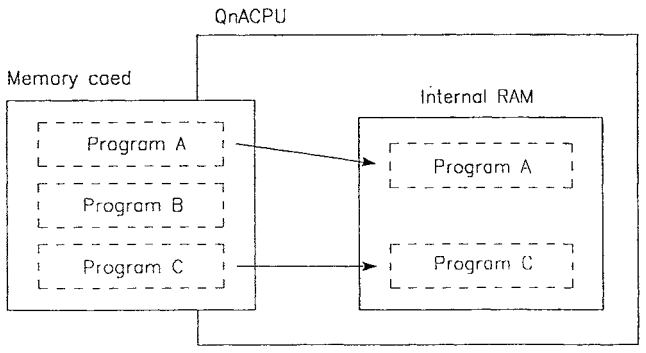
2.3.3 Executing memory card programs (boot run)

(1) The QnACPU only processes programs which are stored in the internal RAM. Therefore, programs stored in the memory card must be booted (read) to the internal RAM using the boot file name designated by a parameter setting. The files designated with parameters are booted (read) from the memory card to the internal RAM according to the order specified in the boot file.

[Boot file setting window]



(2) File additions from the memory card to the internal RAM, changes, and deletions are impossible while the QnACPU is in the RUN status. To execute these operations, STOP the QnACPU, designate the desired boot file using the parameter, then rest the CPU. After completion of reset, RUN the CPU.



Boot processing procedure

1. Designate (by parameter setting) the file to be transferred.
2. Reset the QnACPU.
3. Designated file is transferred from the memory card to the internal RAM.
4. Execution of the transferred file begins.

2.4 File Types & Storage Destinations of Files Managed by QnACPU

(1) File types & storage destinations of files managed by QnACPU

Files which can be stored in the internal RAM, and those which can be stored in the memory card are determined according to the file type.
Storable files and their storage destinations are shown in Table 2.3 below.
An memory card may or may not be necessary, depending on the type of file to be stored in the QnACPU.

Table 2.3 Files & Storage Destinations in QnACPU

Item	File Type	File Name *4	Storage Drives *1					Restrictions	Reference
			0	1	2	3	4		
For Program	Parameter	********.QPA	○	○	○	○	○	1 file per drive	Section 5
	Sequence program/ SFC program	********.APG	●*2	○*2	○*2	○*2	○*2		Section 3
For devices	Device comment	********.QCD	○*5	○	○	○	○	Max. 124 files	User's *3
	Device initial values	********.QDI	○	○	○	○	○	Max. 124 files	Section 4.13
	File registers	********.QDR	X	○	Δ	○	Δ	Max. 124 files	Section 4.7
	Simulation data	********.QDS	X	○	X	○	X		User's *3
	Local device	********.QDL	X	○	X	○	X	1 file per CPU	Section 4.13
For debugging	Sampling trace data	********.QTS	X	○	X	○	X		User's *3
	Status latch data	********.QTL							
	Program trace data	********.QTP							
	SFC trace data	********.QTR							
For fault diagnosis	Fault history data	********.QFD	X	○	X	○	X		User's *3

REMARKS

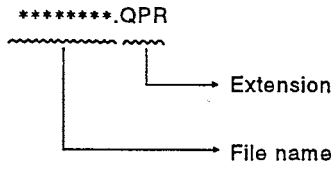
1) *1: Symbols used in the above table are explained below.

Symbol	Meaning
●	Must be stored.
○	Stored when required.
X	Cannot be stored.
Δ	When flash memory is used: File reading only (writing is impossible). When E^2PROM is used: Writing is possible by EROMWR instruction.

2) *2: QnACPU processes programs stored in drive 0 (internal RAM). Programs stored in drives 1-4 must be booted to the internal RAM in order to be executed.

3) *3: User's Manual of the CPU module used.

4) *4: The file name configuration is as follows:

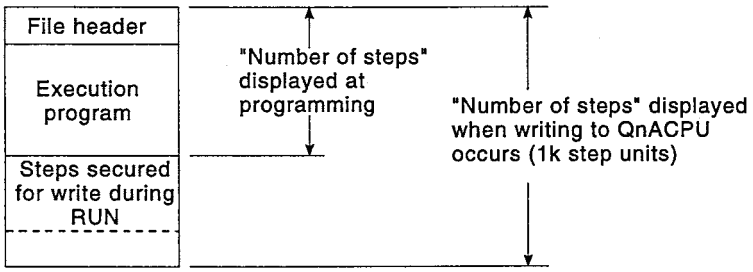


Since files are designated at the peripheral device with the file name and file type, there is no need to think about the extention.
The peripheral device converts the file type of the designated file to the extention before writing it to the QnACPU.

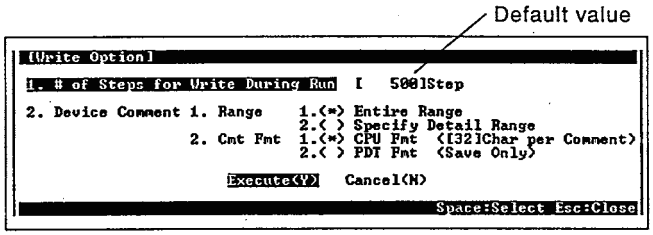
5) *5: Device comments stored at drive 0 cannot be used at the comment instructions (LEDC, etc.) of sequence programs and SFC programs.

2.5 Program File Configuration

Program files consist of a file header, an execution program, and steps secured for write during RUN.
As shown below, the size of a program stored in the QnACPU includes all the above components.

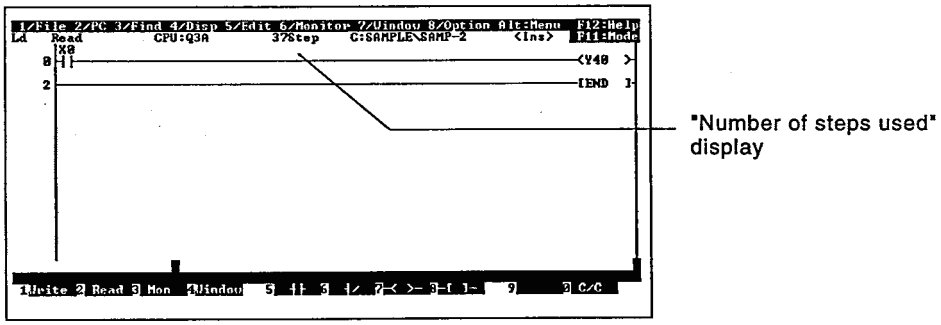


- File header : The file name, file size, and file creation data, etc., are stored in this area.
  The file header size ranges is from 34 to 35 steps (136 to 140 bytes). (Default: 34 steps. When setting the retentive timer, it has 35 steps.)
- Execution program : The created program is stored in this area.
  1 step = 4 bytes.
- Steps secured for write during RUN : This area is used when write during RUN that increases the number of steps is executed at a peripheral device.
  Default value = 500 steps (2000 bytes).
  The number of steps can be changed using the write options at the peripheral device's PC menu.



During programming at the peripheral device, the total of the file header size and the number of execution program steps is displayed as the "number of steps used."

[Ladder readout screen]



REMARK

Program storage capacity differs depending on the memory where programs are stored.

Memory	Program Capacity Unit
Internal memory	4096 bytes (1k steps)
Memory card	512 bytes

2.6 File Operation and File Handling Precautions

2.6.1 File operation

Using the "online" function of the GPP peripheral device, the file operations shown in Table 2.4 below are possible with regard to files stored in the internal RAM and memory cards.
However the available file operations will vary according to the presence or absence of an entry code (registered by peripheral device), the QnACPU "write protect" switch setting status, and the QnACPU RUN/STOP status.

Table 2.4 File Operations from Peripheral Device

File Operation	Operation Enabled/Disabled *1				Operation Description
	A	B	C	D	
File list	o	Δ*2	o	o	A list of files stored in memory is displayed.
Read	o	Δ*2	o	o	Files are read from memory.
Write *4	Δ*2	Δ*2	x	Δ*3	Files are written to memory.
Program writing during RUN status	Δ*2	Δ*2	x	o	Program writing is executed during the QnACPU RUN status.
Rename	Δ*2	Δ*2	x	Δ*3	The name of a file stored in memory is changed.
Copy	Δ*2	Δ*2	x	Δ*3	A file stored in memory is copied.
Delete	Δ*2	Δ*2	x	x	A file stored in memory is deleted.
PC memory organize *4	Δ*2	Δ*2	x	x	Memory files which ate no longer contiguous are re-organized to make them contiguous.
PC memory format	Δ*2	Δ*2	x	x	Memory formatting is executed.

REMARKS

1) *1: The codes (A, B, C, D) used at the "operation enabled/disabled" item in the above table are explained below.

- Operation enabled/disabled

Code	Description	Reference
A	When "write prohibit" entry code is registered at CPU	User's Manual of the CPU module used (Detailed information)
B	When "read/write display prohibit" entry code is registered at CPU	
C	When the CPU's "system protect" switch is ON	
D	When a CPU RUN/STEP-RUN status is in effect	

- symbols used in table

Symbol	Description
o	Execution enabled
Δ	Execution enabled with some restrictions
x	Execution disabled

2) *2: Execution of parameter and program files is only possible when the entry codes match.

3) *3: The status is "operation disabled" (X) in the following cases:
   - Parameter program + program file
   - When a new file is created during the RUN status
     (When the same file as the file being copied does not exist at the copy destination)
   - When copying to drive D during the RUN status

4) *4: In order to secure a contiguous area for the data of a designated file, a file shift mayoccur if the file size is increased.

5) Reading/writing to the memory card's E^2PROM is possible in the same manner as at its RAM.
   However the processing time is longer when writing to the E^2PROM than it is when writing to the RAM.

6) File cannot be written directly from the peripheral device to the IC memory card's flash ROM. To do this, a memory card reader/writer set must be installed at the GPP function peripheral device, and writing executed via that reader/writer.

2.6.2 File handling precautions

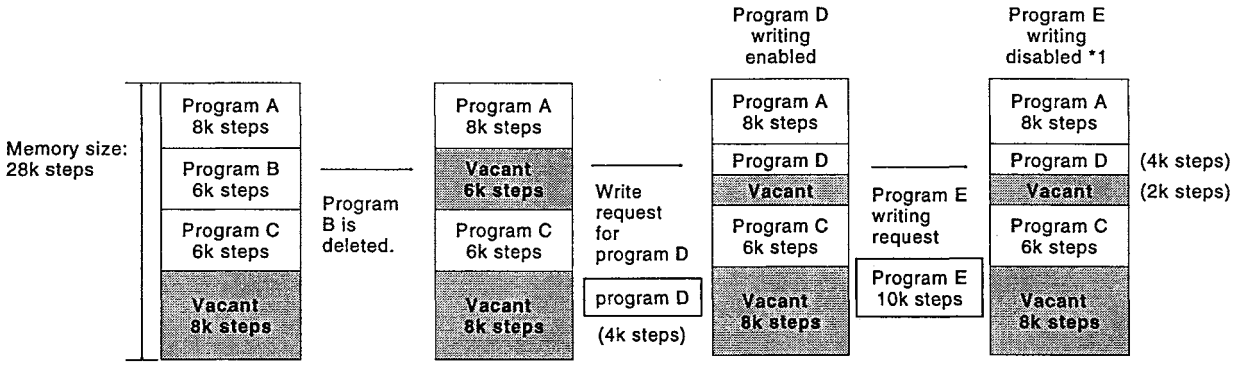
Precautions regarding the handling QnACPU files are discussed in this section.

(1) File contiguousness in the memory

(a) Internal RAM and memory card files are basically arranged in a contiguous manner.
However, following a number of file deletions and writing operations, the amount of contiguous vacant space may be reduced to the point where files can no longer be stored, even though the overall vacant space is adequate.

(b) When the overall vacant space exceeds the size of the file to be written, the "PC memory organize" function (the GPP function peripheral device online mode) can be executed to gather all the non-contiguous vacant space into a single contiguous area.

Example 1: When writing to the QnACPU internal RAM is impossible:
To simplify the explanation, system files and parameters, etc., are excluded.

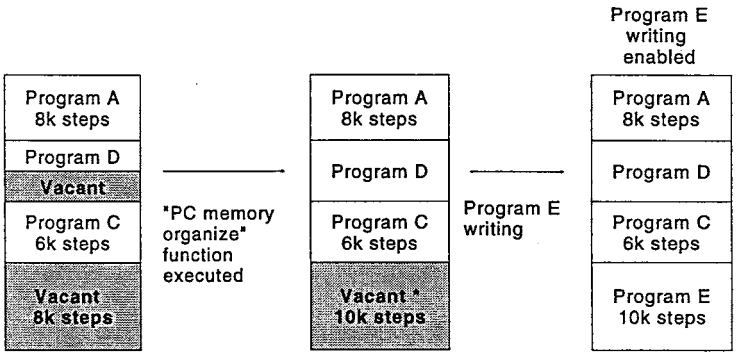


Overall vacant space	8k steps	14k steps	10k steps	10k steps
Contiguous vacant space	8k steps	8k steps 6k steps	8k steps 2k steps	8k steps 2k steps

REMARK

1) *1: Writing to the internal RAM is impossible because the amount of contiguous vacant space is only 8k steps.

Example 2: When the "PC memory organize" function is executed:
When the internal RAM condition is as shown in Example 1 on the previous page, the "PC memory organize" function can be executed to secure 10k steps of contiguous vacant space ("*" in fig. below).



Overall vacant space	10k steps	10k steps	0k steps
Contiguous vacant space	8k steps	10k steps	0k steps

(2) Power OFF (or reset) during program operation

(a) If power is switched OFF (or a reset occurs) during a file operation which will not cause a file shift, the memory data will not be lost.

(b) If the QnACPU battery backup is in effect, the memory data will not be lost if the power is switched OFF (or reset occurs) during a file operation which causes a file shift.
Files stored in the memory card will not be lost unless the memory card is removed from the QnACPU while the power is OFF.

POINTS

(1) The following file operations can cause a file shift:

- File size change
- PC memory organize function
- New file creation

(2) If a power OFF occurs during the above operations, the data up to the power OFF will be stored in the QnACPU internal RAM, and will be restored when power is switched ON again. A battery backup is required in order to save internal RAM data is this manner.

(3) Write during RUN when program file size is increased

(a) The QnACPU program file size is the created program space plus the steps secured for write during RUN. When write during RUN is executed using the peripheral device's GPP function, the program size should not exceed the file space secured when initial program file writing occurred.

(b) Write during RUN is impossible if the size of an edited program exceeds the program file space secured when initial program file writing occurred.

- If a file's size is likely to be increased by write during RUN, set the steps secured for write during RUN in advance at the peripheral device.
- If the QnACPU is stopped, writing is possible even if the file size has been increased.

(4) Simultaneous access of a single file from multiple peripheral devices

(a) The QnACPU permits a file which is being accessed by an RS-422 connected peripheral device to be accessed simultaneously by another peripheral device which is connected via a network or serial communication module.

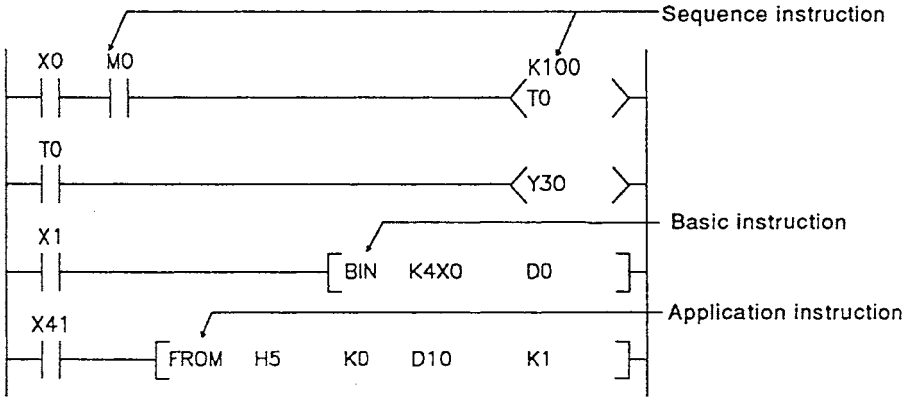
3. SEQUENCE PROGRAM CONFIGURATION & EXECUTION CONDITIONS

Sequence programs and SFC programs can be executed at the QnACPU. This chapter describes the sequence program configuration and execution conditions.
SFC programs are not described in this manual.
For details regarding SFC programs, refer to the QnACPU Programming Manual (SFC).

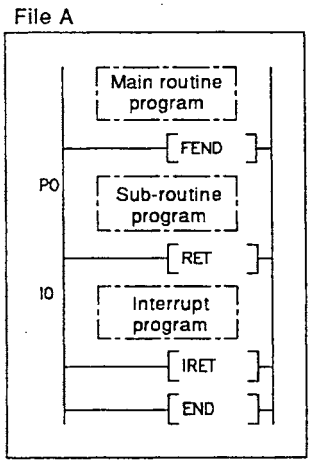
3.1 Sequence Program

(1) Definition of sequence program

(a) A sequence program is created using QnACPU sequence instructions, basic instructions, and application instructions, etc.



(b) There are 3 types of sequence program: main routine programs, sub-routine programs, and interrupt programs.
For details regarding these programs, refer to the following sections of this manual:
- Main routine programs : Section 3.1.1
- Sub-routine programs : Section 3.1.2
- Interrupt programs : Section 3.1.3



REMARK

1) For details regarding the QnACPU sequence instructions, basic instructions, and application instructions, refer to the "QnACPU Programming Manual (Common Instructions)".

(2) Sequence program writing format

Programming for sequence programs is possible using either the relay symbolic language (ladder mode), or the logic symbolic language (list mode).

(a) Relay symbolic language

- The relay symbolic language is based on the relay control ladder.
  Programming expressions are similar to the relay control sequence ladder.
- Relay symbolic language programming occurs in ladder block units.
  A ladder block is the smallest unit of sequence program processing, with the ladder beginning from the left bus and ending at the right bus.

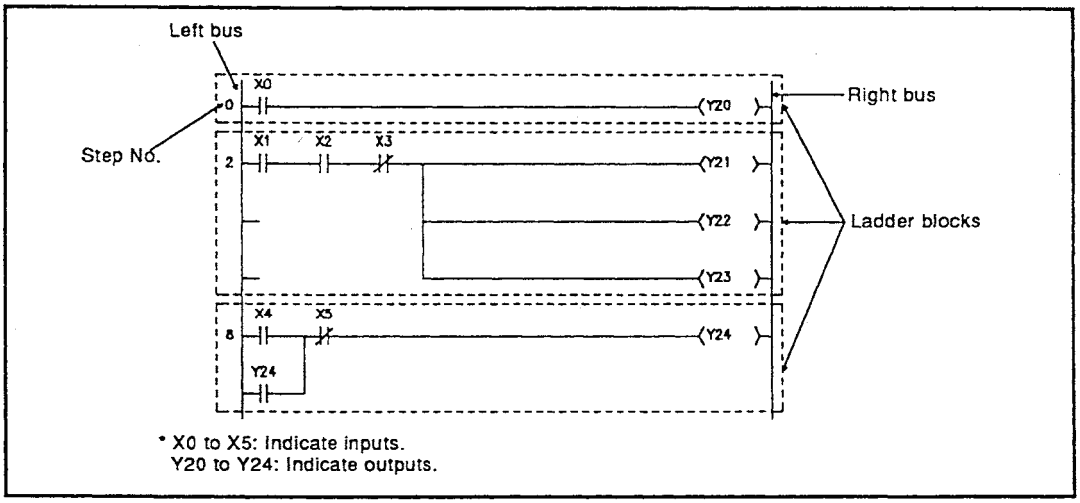


Fig.3.1 Ladder Block

(b) Logic symbolic language (list mode)

The logic symbolic language uses dedicated instructions instead of the contact symbols, coil symbols, etc., used in the relay symbolic language.

(c) Program processing

Sequence programs are processed in order, beginning from step 0 and ending at the END instruction.
Processing of relay symbolic language ladder blocks begins from the left bus, and proceeds from left to right. When one ladder block is completed, processing proceeds downward to the next ladder block.

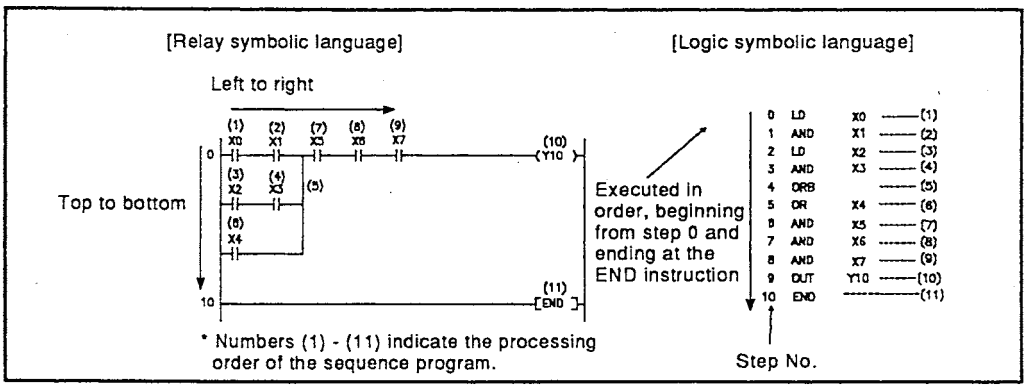


Table 3.2 Sequence Program Processing

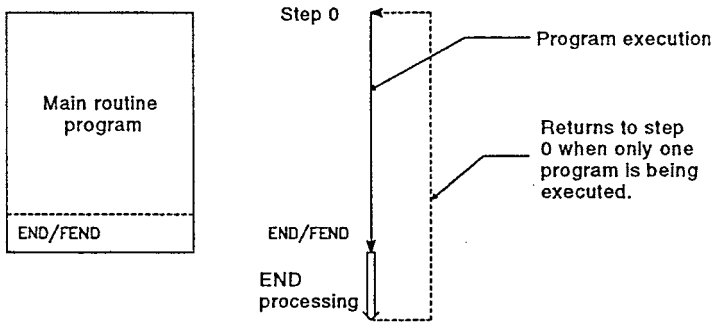
3.1.1 Main routine program

(1) Definition of main routine program

(a) A main routine program is a program which begins from step 0 and ends at the END/FEND instruction.

(b) The main routine program execution begins from step 0 and ends at the END/FEND instruction.

1) If only one program is being executed, processing will begin from step 0 again after the END/FEND instruction is processed.



2) If multiple programs are being executed, processing which occurs after the END/FEND instruction varies according to the designated execution conditions. (See item (2) below)

(2) Execution conditions for main routine programs

If multiple programs are being executed, the following four types of execution conditions can be designated (by program settings in the parameters) according to the application in question.

- Initial execution program: See Section 3.2.1.
- Scan execution program: See Section 3.2.2.
- Low-speed execution program: See Section 3.2.3.
- Standby program: See Section 3.2.4.

3.1.2 Sub-routine programs

(1) Definition of sub-routine program

(a) A sub-routine program is a program which begins from a pointer (P□) and ends at a RET instruction.

(b) A sub-routine program is executed only when called by a CALL (P) instruction from the main routine program.

(2) Sub-routine program application

(a) The overall step count can be reduced by using a sub-routine program as a program which is executed several times in one scan.

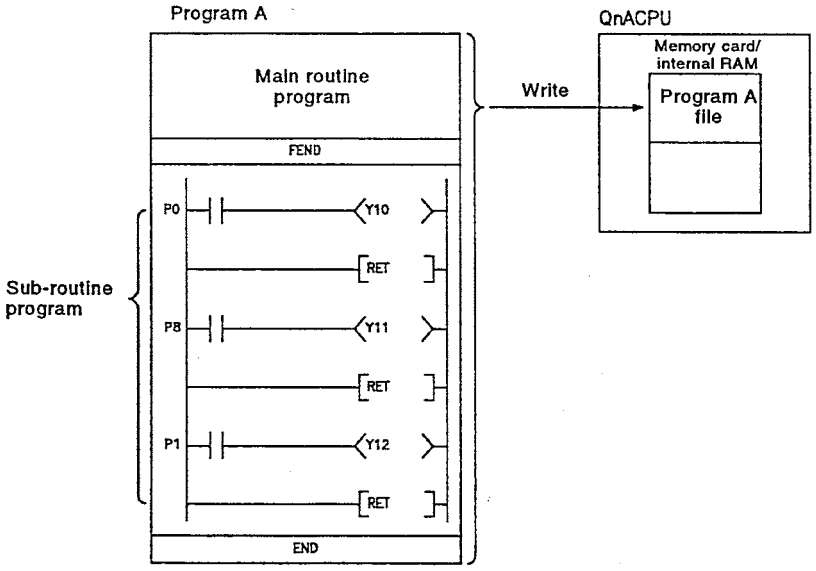
(b) The step count of a constantly executed program can be reduced by using a sub-routine program as a program which is executed only when a given condition is satisfied.

(3) Sub-routine program management

Sub-routine programs are created after the main program (after FEND instruction), and the combination of main and sub-routine programs can be managed as one program.
Sub-routine programs can also be managed as separate, discrete programs (standby programs). (See Section 3.2.4 for details regarding standby programs).

(a) When created after the main program

- A sub-routine program is created between the main program's FEND and END instructions.
- Because there are no restrictions regarding the order in which sub-routine programs are created, there is no need to set the pointers in ascending order when creating multiple sub-routine programs.
- Either a local pointer or a common pointer may be used. *



REMARKS

1) *: See Section 4.9 for details regarding local and common pointers.

2) See Section 4.8 for details regarding sub-routine program nesting.

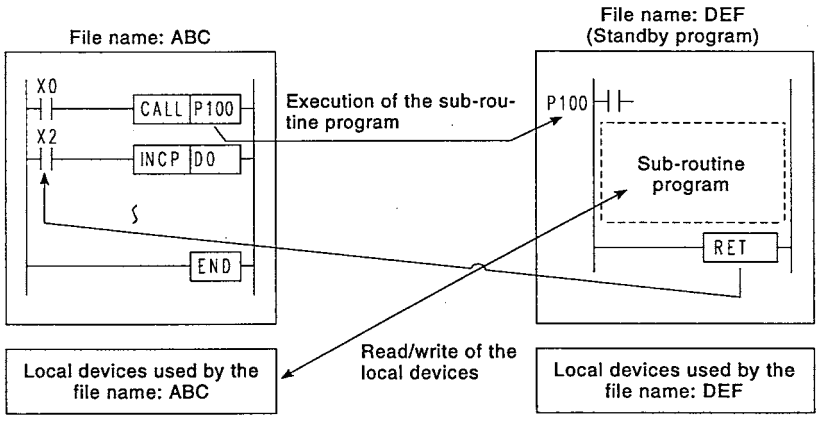
(4) Using local devices used by the file where a sub-routine program is stored

It is possible to use local devices that are used by the file where a sub-routine program is stored when executing a sub-routine program. Whether or not such local devices are used is set by special relay "SM776".

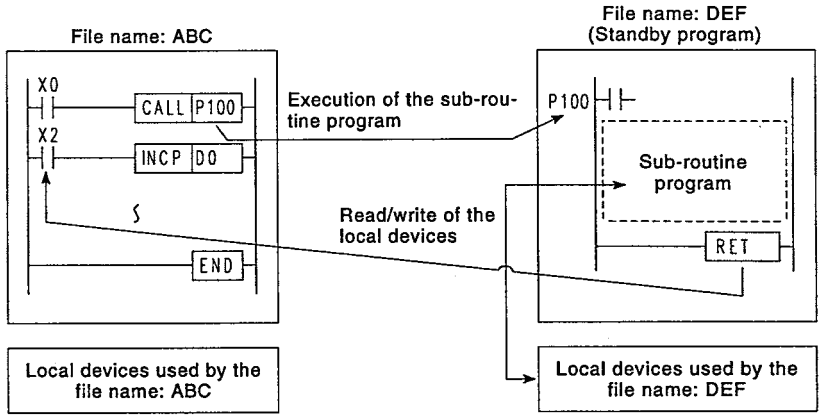
(a) Switching over local devices by setting ON/OFF for a special relay

	SM776
OFF	Executes calculation by the local devices that are used by the file where the sub-routine program was called.
ON	**Executes calculation by the local devices that are used by the file where the sub-routine program is stored.**

[Operation at "SM776 : OFF" when the function version is supported or not supported]



[Operation at "SM776 : ON" when the function version is supported]



(b) Cautions

- If SM776 is ON, the local device data is read when the sub-routine program is called and the local device data is saved after the execution of the RET instruction.
  Accordingly, scan time is elongated by the time as shown below when a sub-routine program is executed once with the setting of "SM776: ON".

- Q2ACPU(S1), Q2ASCPU(S1) : 560 + 1.3 * (Number of words of a local device) [μs]
- Q3ACPU : 425 + 1.0 * (Number of words of a local device) [μs]
- Q4ACPU, Q2ASHCPU(S1) : 220 + 0.8 * (Number of words of a local device) [μs]

- ON/OFF setting of SM776 is possible in unit of QnACPU or Q2AS(H)CPU.
  Setting in unit of file is not possible.

- If the ON/OFF setting of SM776 is changed while a sequence program is executed, the control is made according to the information after change.

3.1.3 Interrupt programs

(1) Definition of interrupt program

(a) An interrupt program is a program which begins at the interrupt pointer (I□)*, and ends at the IRET instruction.

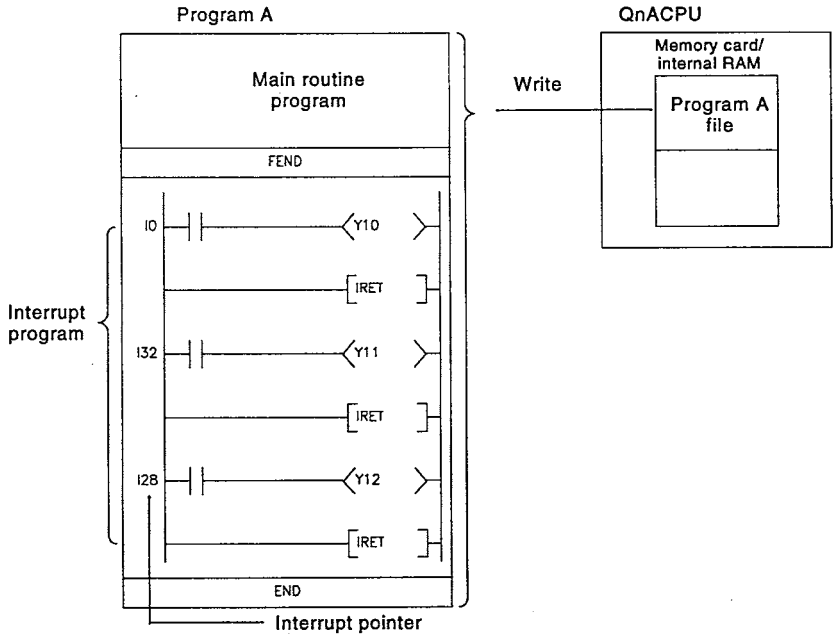
(b) Interrupt programs are executed only when an interrupt factor occurs. *

(2) Interrupt program management

Interrupt programs are created after the main program (after the FEND instruction), and the combination of main and sub-routine programs can be managed as one program.
Interrupt programs can also be managed as separate, discrete programs (standby programs). (See Section 3.2.4 for details regarding standby programs). However, the same interrupt program pointer number cannot be used more than once in the program being executed by the QnACPU.

(a) When created after the main program

- An interrupt program is created between the main program's FEND and END instructions.
- Because there are no restrictions regarding the order in which interrupt programs are created, there is no need to set the interrupt pointers in ascending order when creating multiple interrupt programs.



REMARK

1) *: See Section 4.10 for details regarding interrupt factors and interrupt pointers.

(3) Executing interrupt programs

(a) In order to execute an interrupt program, IMASK and EI instructions are required to obtain permission for the interruption. *1

- If an interrupt factor occurs prior to an "interruption permitted" status, the interrupt program for the factor in question will be executed when an "interruption permitted" status is established.
- If an interrupt factor occurs during a STOP/PAUSE, the interrupt program for the factor in question will be executed when an "interruption permitted" condition is established following a return to the RUN status.

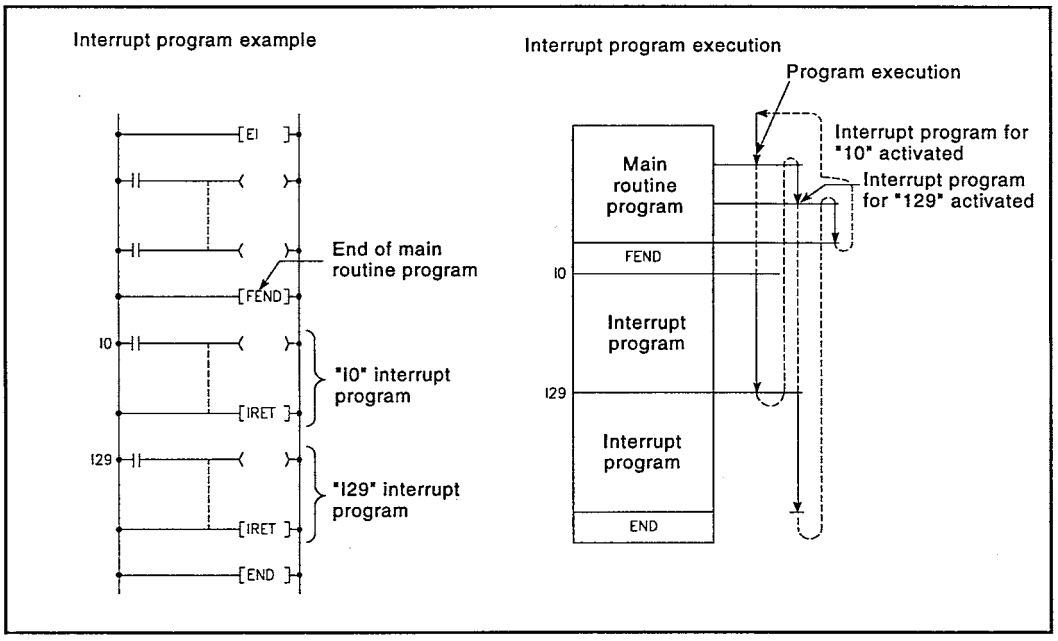


Fig.3.3 Interrupt Program Execution

(b) When an interrupt factor occurs, the interrupt program with the interrupt pointer number corresponding to that factor is executed. However, interrupt program execution varies according to the condition at that time.

1) When multiple interruptions are designated:

When multiple interrupt programs are activated simultaneously, the programs will be executed in order, beginning from the interrupt program with the highest priority interrupt pointer number. *2
The remaining interrupt programs remain on standby until processing of the higher priority interrupt program is completed.

2) When an instruction is being executed:

Interruptions are prohibited during execution of instructions. If an interrupt factor occurs during execution of an instruction, the interrupt program will be executed after processing of the instruction is completed.

REMARKS

1) *1: For details regarding the IMASK and EI instructions, refer to the "QnACPU Programming Manual (Common Instructions)

2) *2: See Section 4.10 for details regarding the priority ranking of interrupt programs.

3) Interruption during a link refresh:

If an interrupt factor occurs during a link refresh operation, the link refresh operation is suspended, and the interrupt program is executed.

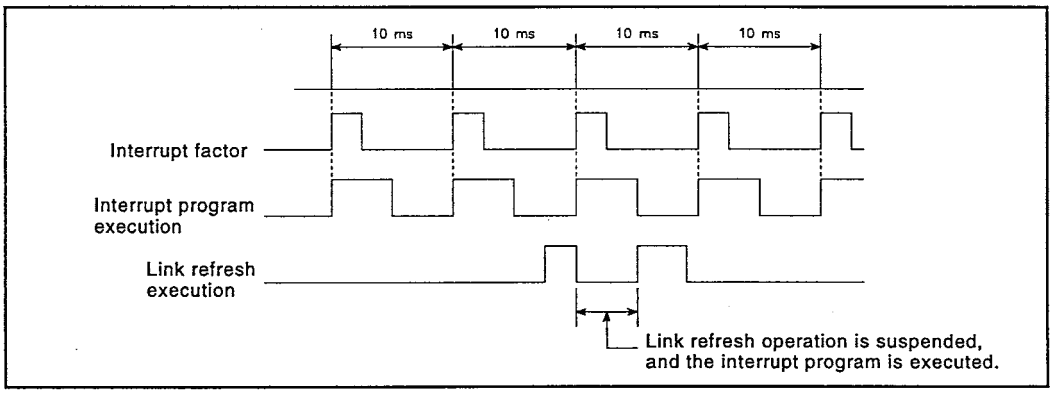


Fig.3.4 Interruption during Link Refresh Operation

4) Interruption during END processing:

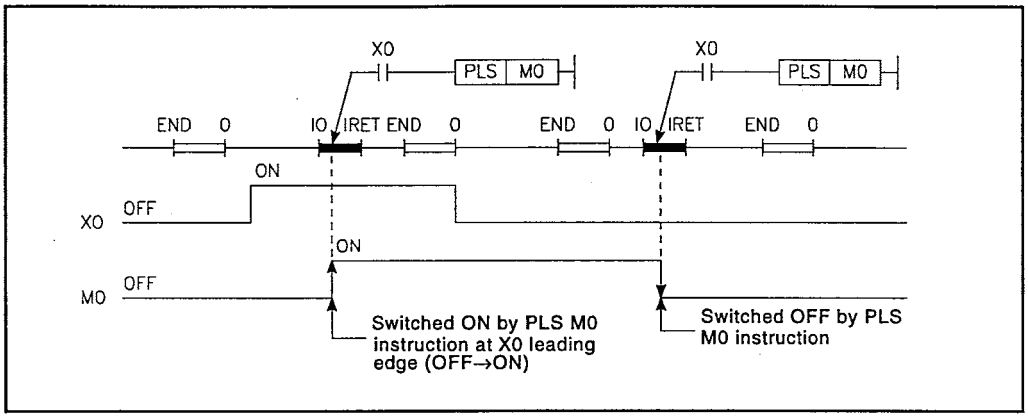
i) If an interrupt factor occurs during general data processing at an END instruction, the interrupt program will be executed after the general data processing is completed.

ii) If an interrupt factor occurs during an END instruction waiting period during constant scanning, the interrupt program corresponding to that factor will be executed.

(c) See Section 4.6 for details regarding index register processing when switching to an interrupt program from a scan execution program or low-speed execution program.

(4) Program creation restrictions

(a) A device which is switched ON by a PLS instruction in an interrupt program will remain ON until that interrupt program is executed again.



(b) A "DI" status (interruption prohibited) is established during execution of an interrupt program.
Do not execute EI/DI instructions in the interrupt program.

(c) Timers cannot be used in interrupt programs.
As timers are used at OUT T□ instructions to update present values and switch contacts ON and OFF, the use of a timer in the interrupt program would make a normal time count impossible.

(d) Local devices cannot be used in interrupt programs. (See Section 4.13.1.)

(e) When an interrupt program is executed in the middle of measuring such as scan time and execution time, the time for the program is added to the measurement time.
Accordingly, once the interrupt program is executed, each value stored to the special registers below and the monitored values of each peripheral device will be longer than the case of not being executed.

1) Special registers
- SD520, SD521: Current scan time
- SD522, SD523: Initial scan time
- SD524, SD525: Minimum scan time
- SD526, SD527: Maximum scan time
- SD528, SD529: Current scan time for low speed
- SD532, SD533: Minimum scan time for low speed
- SD534, SD535: Maximum scan time for low speed
- SD540, SD541: END processing time
- SD542, SD543: Constant scan wait time
- SD544, SD545: Cumulative execution time for low speed execution type programs
- SD546, SD547: Low speed execution time
- SD548, SD549: Scan program execution time
- SD551, SD552: Service interval time

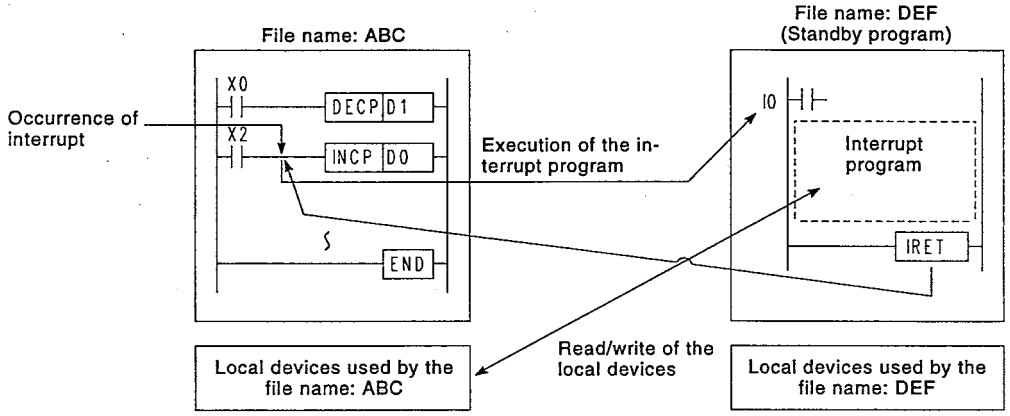
2) GX Developer monitor values
- Execution time measurement
- Scan time measurement
- Constant scan

(5) Using local devices that are used by the file where an interrupt program is stored
It is possible to use local devices that are used by the file where an interrupt program is stored when executing an interrupt program.
Whether or not such local devices are used is set by special relay "SM777".

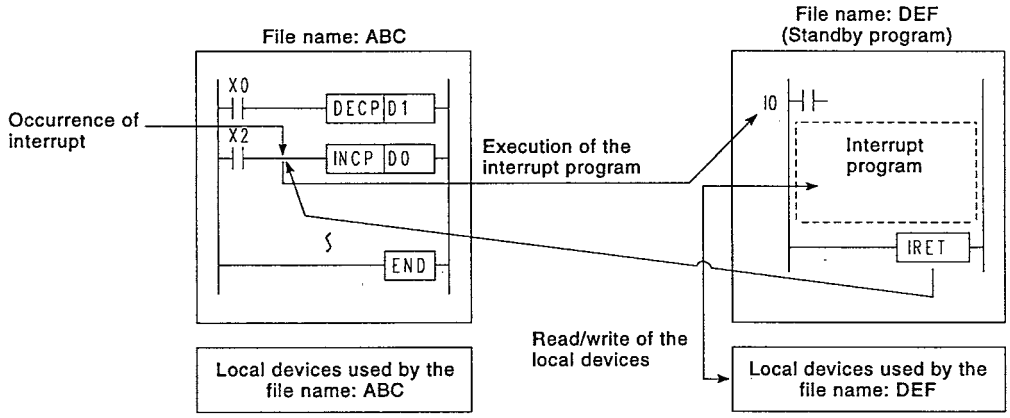
(a) Switching over local devices by setting ON/OFF for a special relay

	SM777
OFF	Executes calculation by the local devices that are used by the file which was executed before the execution of the interrupt program.
ON	Executes calculation by the local devices that are used by the file where the interrupt program is stored.

[Operation at "SM777 : OFF" when the function version is supported or not supported]



[Operation at "SM777 : ON" when the function version is supported]



(b) Cautions

- If SM777 is ON, the local device data is read before the interrupt program is executed and the local device data is saved after the execution of the IRET instruction.
  Accordingly, scan time is elongated by the time shown as below when an interrupt program is executed once with the setting of "SM777: ON".

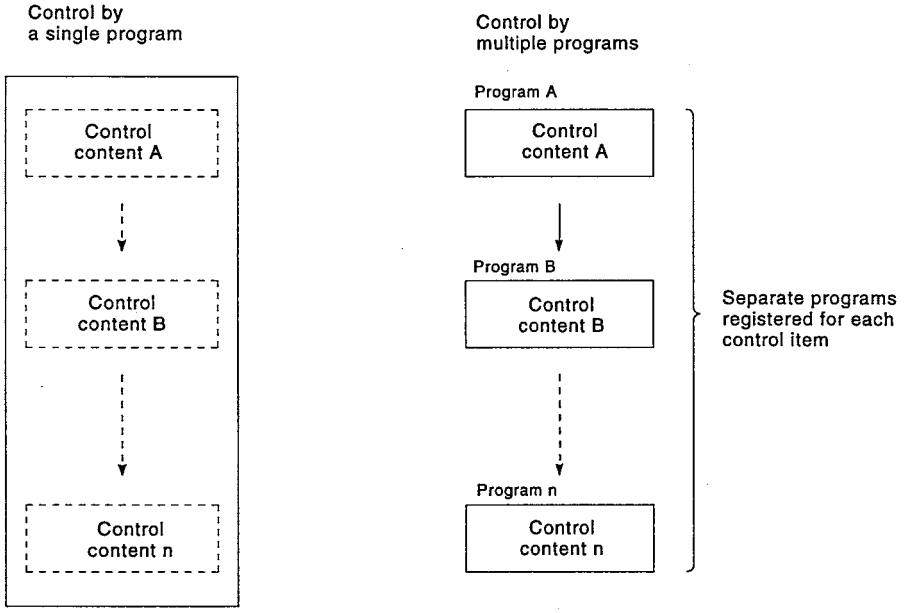
- Q2ACPU(S1), Q2ASCPU(S1) : 560 + 1.3 * (Number of words of a local device) [μs]
- Q3ACPU : 425 + 1.0 * (Number of words of a local device) [μs]
- Q4ACPU, Q2ASHCPU(S1) : 220 + 0.8 * (Number of words of a local device) [μs]

- ON/OFF setting of SM777 is possible in unit of QnACPU or Q2AS(H)CPU.
  Setting in unit of file is not possible.

- If the ON/OFF setting of SM777 is changed while a sequence program is executed, the control is made according to the information after change.

3.2 Program Execution Conditions & Operation Processing

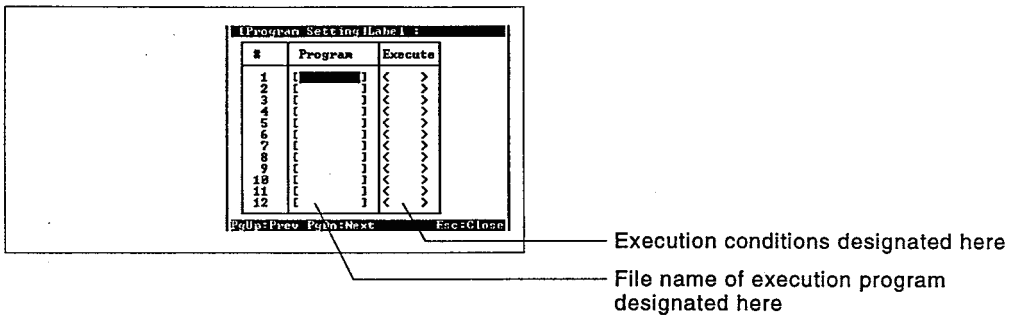
Programs executed by the QnACPU are stored in the CPU's internal RAM, or in memory cards.
Programs can be stored in the internal RAM or memory cards as a single program, or they can be split into separate programs for each control function.
This permits the programming procedure to be split up among several program designers who can design separate programs for each operation.



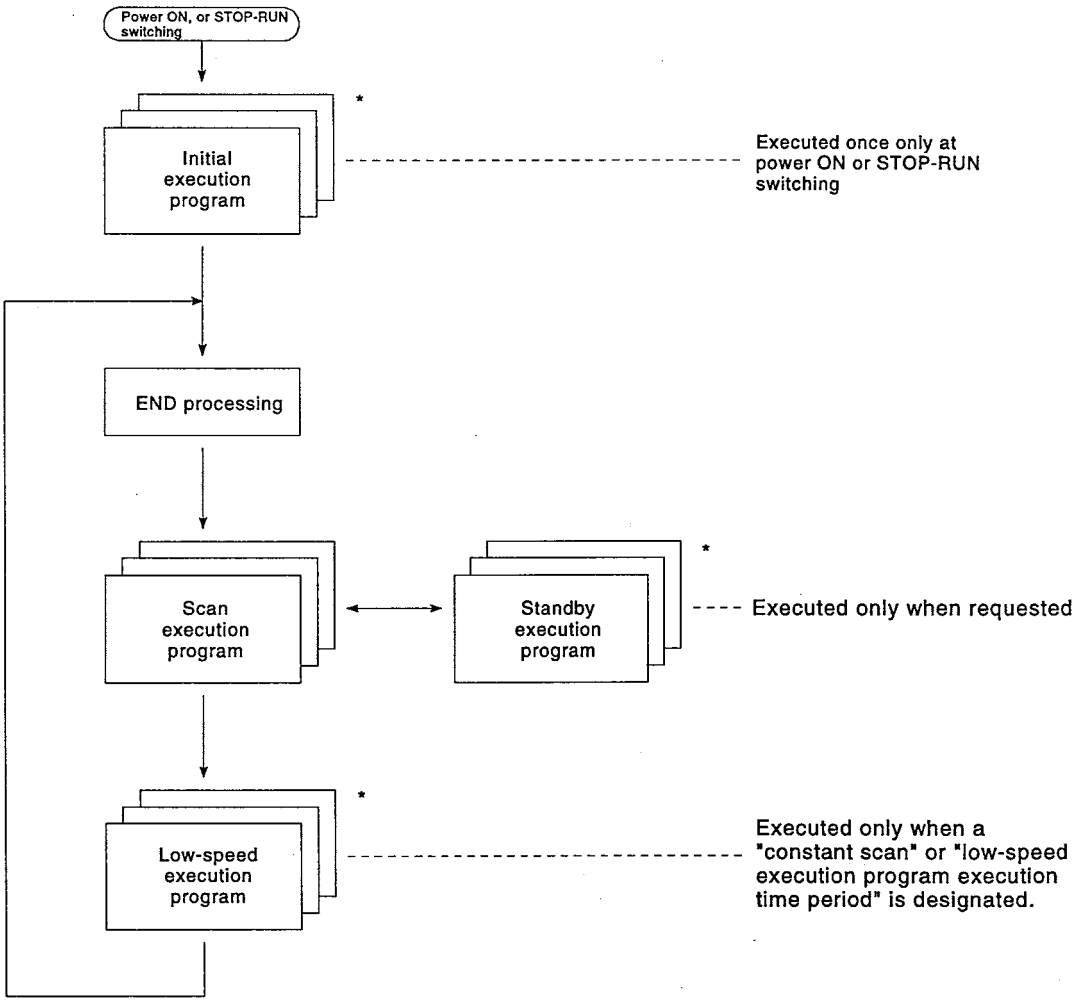
When an operation is split up into multiple programs, an "execution type" setting must be designated in the program settings in the parameters.
The QnACPU executes each of the "execution type" programs in their setting order.
There are 4 execution types: initial execution, scan execution, low-speed execution, and standby.

- Initial execution : This program type is executed once only at power ON, or when STOP-RUN switching occurs. (See Section 3.2.1)
- Scan execution : This program type is executed once per scan, beginning from the scan which follows execution of the initial execution program. (See Section 3.2.2)
- Low-speed execution : This program type is executed only when a constant scan setting is made or when a time is set for execution of low-speed execution type programs.
  - When a constant scan setting is made, the program is executed during the surplus time of a scan execution type program.
  - When a time for execution of low-speed execution type programs is set, the program is executed during this set time. (See Section 3.2.3.)
- Standby type : This program is executed only when its execution is requested. (See Section 3.2.4)

[Program setting window]



The program operation steps which occur at power ON or STOP-RUN switching are shown below.



POINT

(1) Designation of all execution types is not required at the QnACPU.* Initial execution, low-speed execution, and standby type programs indicated by an asterisk are used as required.

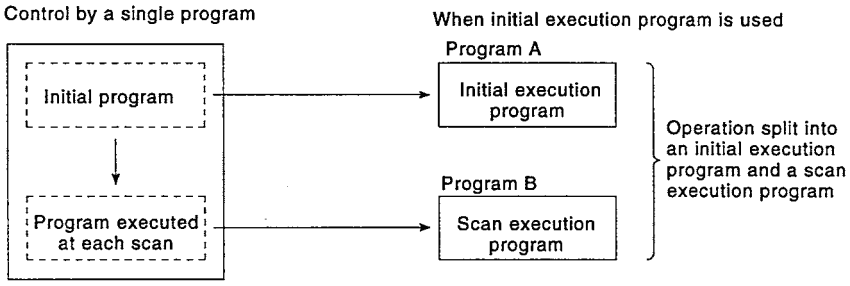
3.2.1 Initial execution programs

(1) Definition

(a) An initial execution program is executed once only at power ON, or when STOP-RUN switching occurs.

(b) This program's execution type is designated as "initial" in the program settings parameters.

(c) In the same manner as the initial processing for the special function module, the initial execution program is executed only once, and is not required in subsequent scans.

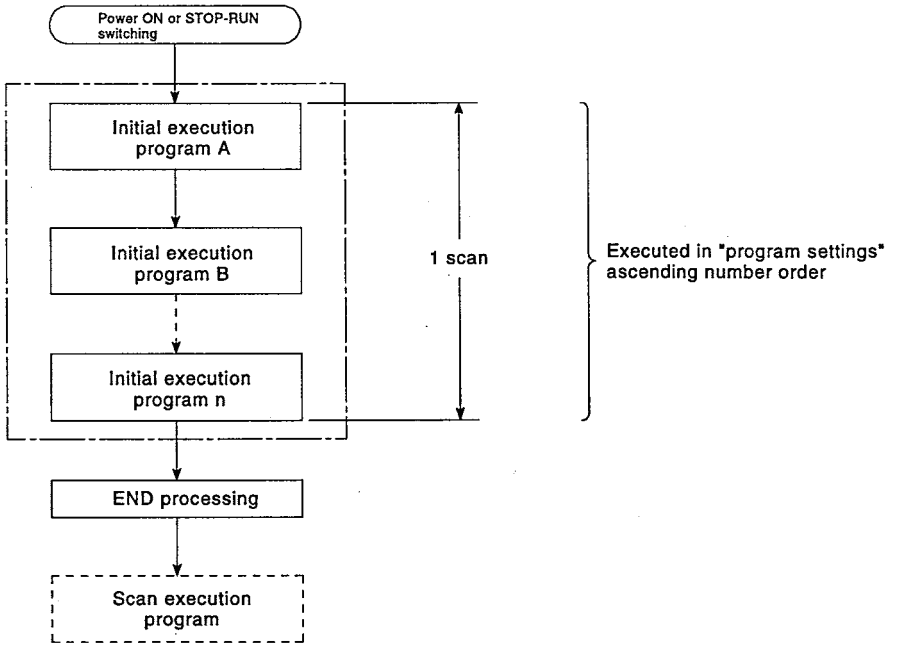


(2) Using multiple initial execution programs

When multiple initial execution programs are used, they are executed one by one in ascending number order (program settings parameter setting).

(3) END processing

END processing occurs when all initial execution programs are completed, and the scan execution program is then executed from the next scan.



POINT

1) *: Instructions with "completion devices" cannot be used in initial execution programs.

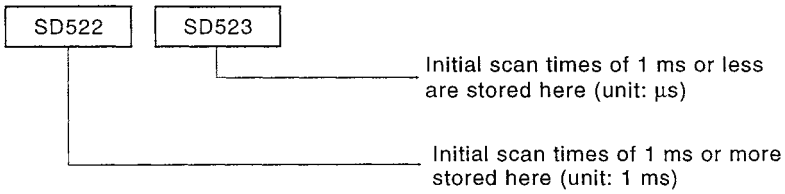
(4) Initial scan time

(a) This is the sum of initial execution type program run time and END processing time.

If multiple initial execution programs are used, this is the execution time period in which all those programs are executed.

(b) The QnACPU measures the initial scan time and stores the result in special registers (SD522, SD523). *1

The initial scan time can therefore be checked by monitoring the SD522 and SD523 special registers.



If the SD522 value is "3", and the SD523 value is "400", the initial scan time is 3.4 ms.

(5) Initial execution time monitor

(a) This timer monitors initial scan time and has no default value setting. If such monitoring is desired, designate the timer setting in a 10 ms to 2000 ms range at the PC RAS settings parameter. (Setting unit:10ms)

(b) A low-spped execution type program is executed after the execution of an initial execution type program is completed. When using a low-speed execution type program, set a time longer than the sum of the initial scan time and low-speed execution type program run time.

(c) If the execution time of the initial execution program exceeds this timer setting, a "WDT ERROR" occurs, and QnACPU operation is stopped.

POINTS

(1) *1: The accuracy of the initial scan time stored at the special registers is ± 0.1 ms.The initial scan time count will continue even if a watchdog time reset instruction (WDT) is executed at the sequence program.

(2) When a monitor timer setting is designated for the initial execution time, there will be a 10 ms error in the count value. Therefore, a monitor timer setting (t) of 10 ms will result in a "WDT ERROR" if the initial scan time is in the following range: 10 ms < t < 20 ms.

3.2.2 Scan execution programs

(1) Definition

(a) Scan execution programs are executed once per scan, beginning from the scan which follows execution of the initial execution program.

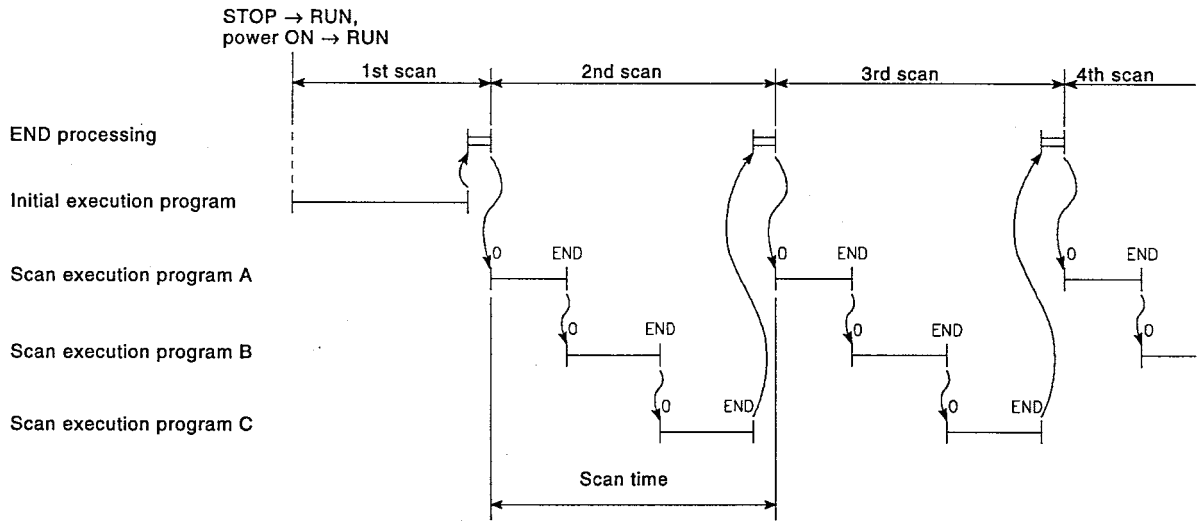
(b) This program's execution type is designated as "scan" in the program settings parameters.

(2) Executing multiple scan execution programs.

When multiple scan execution programs are used, they are executed one by one, in ascending number order (program settings parameter setting).

(3) END processing

END processing occurs when all scan execution programs are completed, and execution the begins again from the first scan n execution program.
END processing (general data processing, link refresh) is possible after each of the programs by designating a COM instruction at the end of the scan execution programs.



(4) Constant scan setting *1

When constant scanning is designated, the scan execution program is executed at each designated constant scan period.

REMARK

1) *1: The "constant scan" function executes the scan type program repeatedly at regular intervals. For details, refer to the User's Manual of the CPU module used.

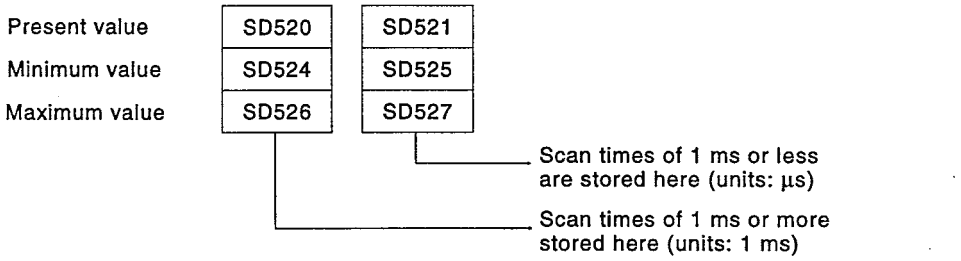
POINT

1) See Section 4.6 for details regarding index register processing when an interrupt program is executed while a scan execution program is in progress.

(5) Scan time

- The "scan time" is the total time required for scan type program execution and END processing.
  If multiple scan execution programs are used, the "scan time" is the total time required to execute all the programs.

- The scan time "present value", "minimum value", and "maximum value" are measured at the QnACPU, and the results are stored in special registers (SD520, AS521, SD524-SD527). *
  The initial scan time can therefore be checked by monitoring the SD520, SD521, SD524-SD527 special registers.



If the SD520 value is "3", and the SD521 value is "400", the initial scan time is 3.4 ms.

(6) WDT (Watchdog timer)

This is the timer which monitors the scan time, and its default setting is 200 ms.
This WDT setting can be designated in a 10 ms to 2000 ms range in the PC RAS settings parameters. (Setting units: 10 ms)
If a low-speed execution program is used, a WDT setting value should be designated which is greater than the scan time plus the execution time of the low-speed execution program.
If the scan time (execution time for scan execution program + low-speed execution program) exceeds the WDT setting value, a "WDT ERROR" occurs, and QnACPU operation is stopped.

POINTS

(1) *1: The accuracy of the scan time stored at the special registers is ± 0.1 ms.
The scan time count will continue even if a watchdog time reset instruction (WDT) is executed at the sequence program.

(2) The WDT measurement error is 10 ms.
Therefore, a WDT setting (t) of 10 ms will result in a "WDT ERROR" if the scan time is in the following range: 10 ms < t < 20 ms.

3.2.3 Low-speed execution programs

(1) Definition

(a) Low-speed execution programs are executed only during "constant scanning surplus time" or during the period designated for low-speed execution program execution.

- For a constant scan time with enhanced control accuracy, designate a constant scan time setting at the PC RAS parameters. (Setting range: 5-2000 ms, setting units: 5 ms)
- To secure execution time for low-speed execution programs at each scan, designate a low-speed execution program execution time setting in the PC RAS parameters. (Setting range: 1-2000 ms, setting units: 1 ms)
- In order to execute a low-speed execution program, one of the above settings ("constant scan time " or "low-speed execution program execution time") must be designated.

(b) The execution type of the low-speed execution program is designated as "low-speed" in the program settings parameters.

(c) The low-speed execution program type is used for programs which do not require execution in each scan, for example programs for printer output.

(2) Executing multiple low-speed execution programs

When multiple low-speed execution programs are used, they are executed one by one, in ascending number order (program settings parameter setting).

(3) Low-speed execution program execution time at 1 scan

(a) If all the low-speed execution program operation is completed within one scan and there is surplus time, the processing executed after that depends on the setting status of special register SM330 and the execution condition for low-speed execution type programs.

- Asynchronous method (SM330 = OFF)
  : Method in which low-speed execution type program operation is continued in the surplus time.
- Synchronous method (SM330 = ON)
  : Method in which even if there is surplus time, low-speed execution program operation is not continued, and operation starts again from the next scan.

Operation method for low-speed execution type programs	SM330 setting status	Execution condition for low-speed execution type programs	
		When "constant scan time" is set	When "low-speed execution program execution time" is set
Asynchronous method	OFF	The low-speed execution type program is re-executed *1.	The low-speed execution type program is re-executed *2.
Synchronous method	ON	Constant scan waiting time is generated *3.	Scan execution type program operation is started *4.

*1 If a "constant scan time" has been designated, the low-speed program will be executed during the constant scan's surplus time.
Therefore, the low-speed execution program's execution time varies from scan to scan.
As the low-speed execution program will not be executed at all if the constant scan's surplus time is 2 ms or less, a "constant scan time" setting should be designated which provides a surplus time of more than 2 ms.

*2 If a "low-speed execution program execution time" has been designated, the low-speed execution program will be executed in accordance with that time setting. Therefore, the scan time will vary from scan to scan.

*3 If a "constant scan time" has been designated, the surplus time after completion of low-speed END processing is waiting time, and execution of a scan execution type program starts when the constant scan time has elapsed.
Constant scan waiting time
= (constant scan setting time) — (scan time)
— (low-speed scan time)
This means that the scan time is constant in each scan.
However, if the surplus time after the constant scan is less than 2 ms, low-speed execution type programs cannot be executed. If using a low-speed execution type program, set the constant scan time so that the surplus time is 2 ms or longer.

*4 If a "low-speed execution program execution time" has been designated, scan execution type program operation is started ignoring the surplus time after completion of low-speed END processing.

$$\frac{\text{Surplus time in low-speed program execution time}}{\text{(ignored)}}$$

= (Set time for low-speed program execution time)
— (low-speed scan time)
This means that the scan time differs in each scan.

(b) If a low-speed execution program cannot be processed within constant scan surplus time or within the low-speed execution program execution time, program execution is temporarily stopped and the remainder of the program is executed in the next scan.

POINTS

(1) See Section 4.6 for details regarding index register processing when switching from a scan execution program to a low-speed execution program occurs.

(2) See Section 4.6 for details regarding index register processing when an interrupt program, is executed while a low-speed execution program is in progress.

(3) The "low-speed execution program execution time" setting should be such that the [scan time] + [low-speed execution program execution time] sum is less than the WDT setting value.

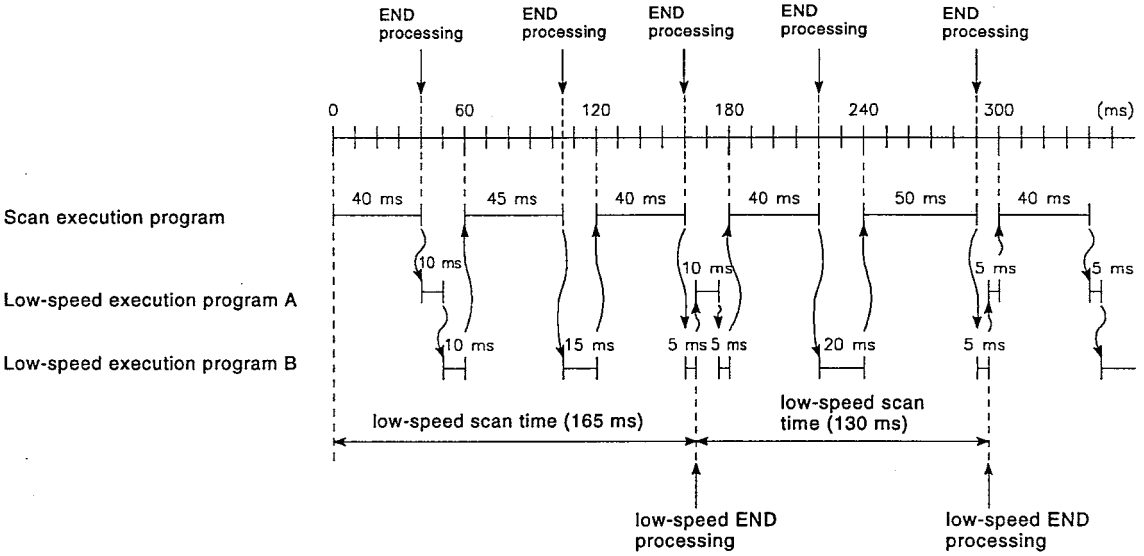
(4) The COM instruction cannot be used in low-speed programs.

Asynchronous method

(1) "Constant scan time" setting

The low-speed execution program operation which occurs under the following conditions is illustrated below.

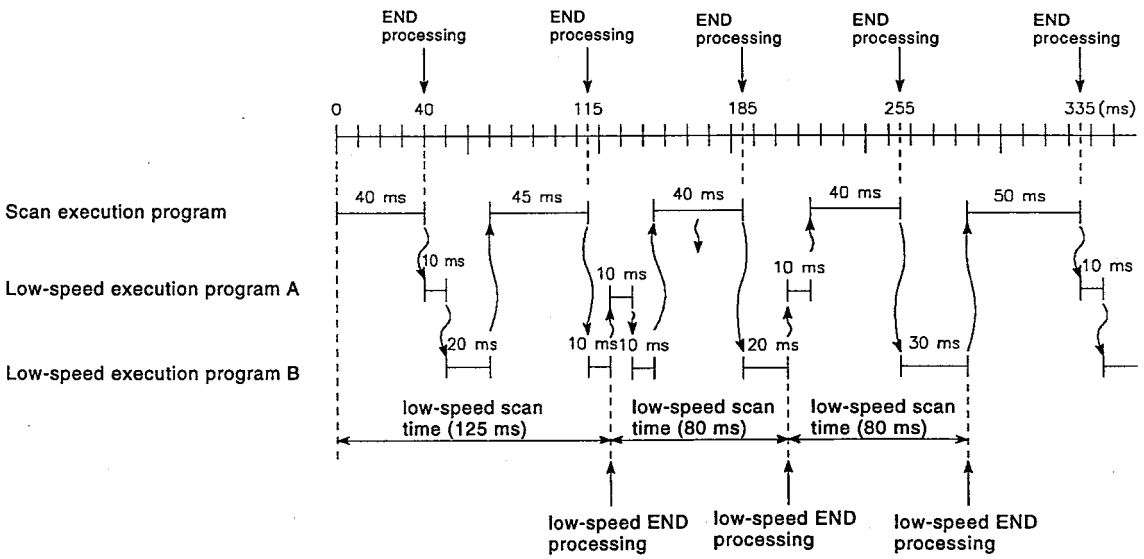
- Constant scan time : 60 ms
- Total scan execution program time : 40 ms to 50 ms
- Execution time of low-speed execution program A : 10 ms
- Execution time of low-speed execution program B : 30 ms
- END processing : 0 ms (0 ms is used to simplify the illustration)
- Low-speed END processing : 0 ms (0 ms is used to simplify the illustration)



(2) "Low-speed execution program execution time" setting

The low-speed execution program operation which occurs under the following conditions is illustrated below.

- Low-speed execution program execution time : 30 ms
- Total scan execution program time : 40 ms to 50 ms
- Execution time of low-speed execution program A : 10 ms
- Execution time of low-speed execution program B : 30 ms
- END processing : 0 ms (0 ms is used to simplify the illustration)
- Low-speed END processing : 0 ms (0 ms is used to simplify the illustration)

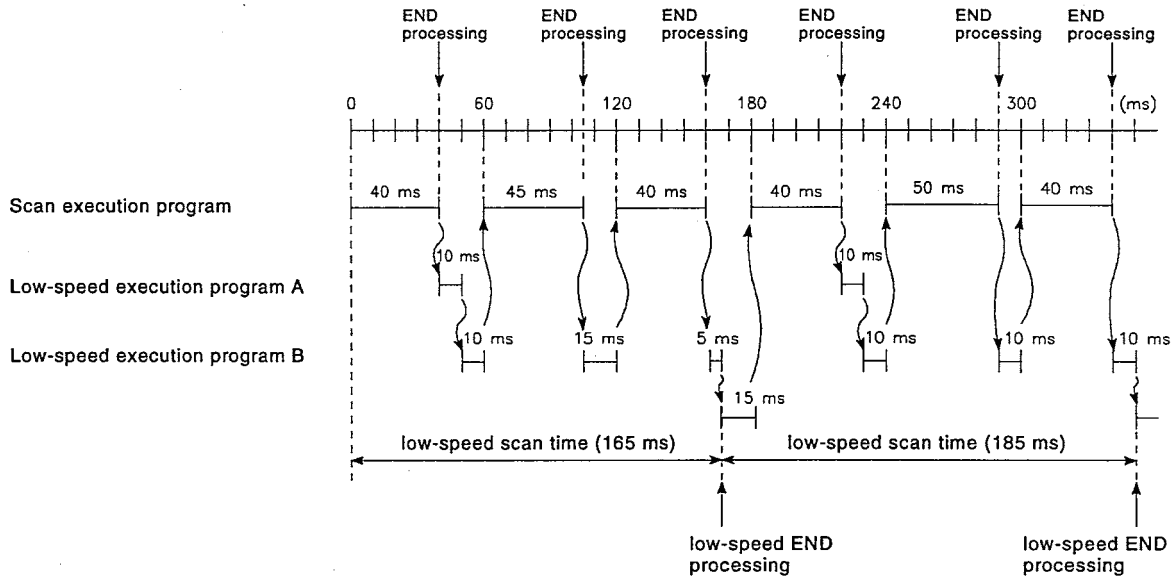


Synchronous method

(1) "Constant scan time" setting

The low-speed execution program operation which occurs under the following conditions is illustrated below.

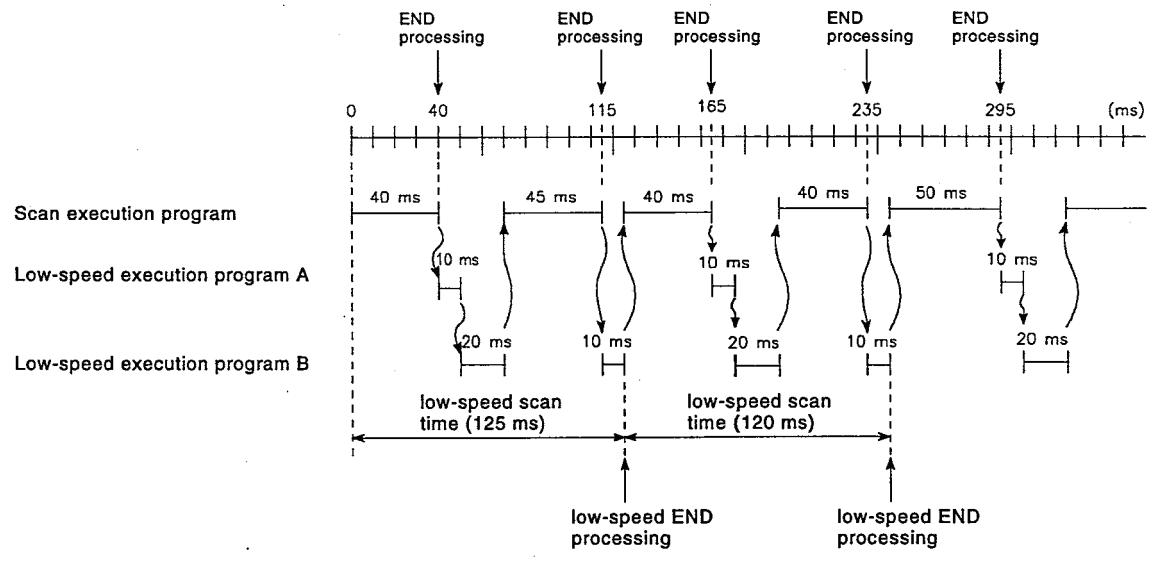
- Constant scan time : 60 ms
- Total scan execution program time : 40 ms to 50 ms
- Execution time of low-speed execution program A : 10 ms
- Execution time of low-speed execution program B : 30 ms
- END processing : 0 ms (0 ms is used to simplify the illustration)
- Low-speed END processing : 0 ms (0 ms is used to simplify the illustration)



(2) "Low-speed execution program execution time" setting

The low-speed execution program operation which occurs under the following conditions is illustrated below.

- Low-speed execution program execution time : 30 ms
- Total scan execution program time : 40 ms to 50 ms
- Execution time of low-speed execution program A : 10 ms
- Execution time of low-speed execution program B : 30 ms
- END processing : 0 ms (0 ms is used to simplify the illustration)
- Low-speed END processing : 0 ms (0 ms is used to simplify the illustration)



(4) END processing

Low-speed END processing occurs after all the low-speed execution programs have been executed. Low-speed processing includes the following items:

- Low-speed execution program special relay/special register setting.
- Low-speed execution program write during RUN.
- Low-speed scan time measurement.
- Low-speed execution program watchdog timer resetting.

When low-speed END processing is completed, execution of the low-speed execution programs begins again from the first program.

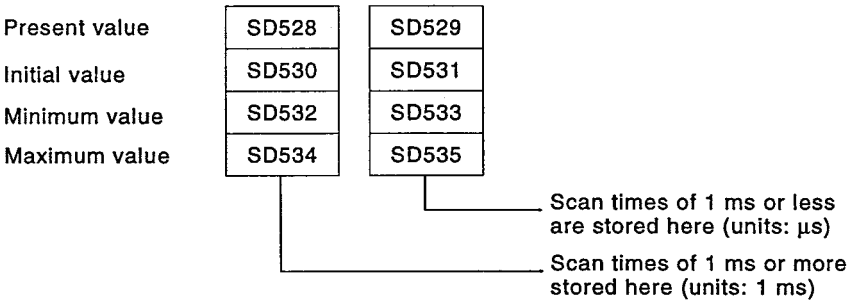
POINT

(1) During execution of low-speed execution programs, the "constant scan" time may deviate by the amount of the [maximum instruction processing time] + [low-speed END processing time].

(5) Low-speed scan time

(a) The "low-speed scan time" is the total time required for low-speed execution program execution and low-speed END processing.
If multiple low-speed execution programs are used, the "low-speed scan time" is the total time required to execute all the programs, plus the low-speed END processing time.

(b) The low-speed scan time is measured by the QnACPU, and the result is stored in special registers (SD528-SD535). *1
The low-speed scan time can therefore be checked by monitoring the SD528-SD535 special registers.



If the SD528 value is "50", and the SD529 value is "400", the low-speed scan time is 50.4 ms.

(6) Low-speed execution time monitor

The execution period of the low-speed execution program can be monitored by this timer (there is no timer default setting). If such monitoring is desired, designate the timer setting in a 10 ms to 2000 ms range at the PC RAS settings in the parameter mode (Setting units: 10 ms)
If the execution time of the low-speed execution program exceeds this timer setting, a "PRG TIME OVER" error occurs (QnACPU operation is not stopped).

POINTS

(1) *1: The accuracy of the scan time stored at the special registers is ± 0.1 ms.
The scan time count will continue even if a watchdog time reset instruction (WDT) is executed in the sequence program.

(2) The low-speed execution time measurement occurs at low-speed END processing. Therefore a PRG TIME OVER error will occur if the low-speed execution monitor time (t) is designated as "100 ms", and the measured low-speed scan time (at low-speed END processing) exceeds 100 ms.

3.2.4 Standby programs

(1) Definition

(a) Standby programs are programs which are executed only when requested.

(b) Standby programs are used for the following applications.

1) Placing programs in the library

Sub-routine and interrupt programs are converted to standby programs which are managed separately from the main program.

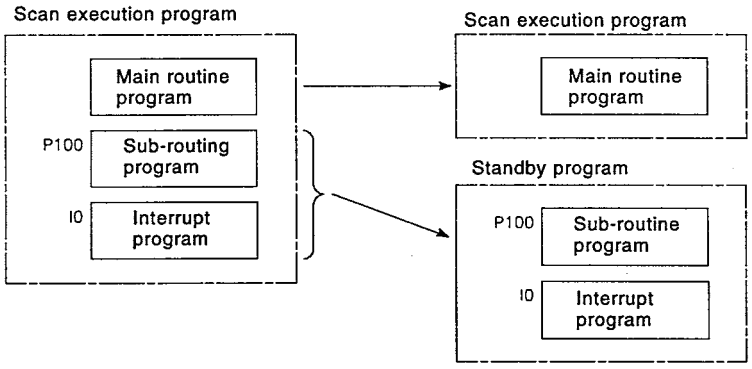
2) Changing the program setup

Main routine programs are registered as standby programs, with required programs then being converted to scan execution programs for execution. Programs which are not required are converted to standby programs.

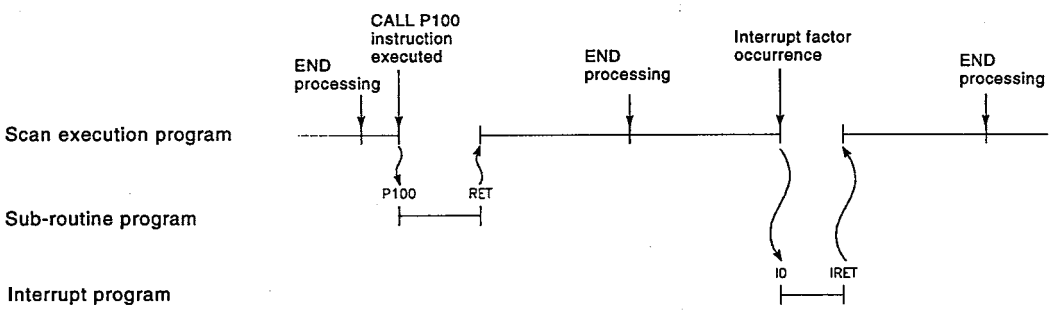
(2) Placing programs in the library

(a) Placing programs in the library

1) This application is used to manage sub-routine and interrupt programs separately from the main routine program.
Multiple sub-routine and interrupt programs can be created for a single standby program.



2) When standby program execution is completed, processing returns to the program which was active before the standby program was executed.
The operation which occurs when a standby program's sub-routine and interrupt programs are executed is shown below.



(3) Changing the program setup

(a) This function can be used to create and execute programs for all systems.
Programs designated by parameter setting as "standby" programs can be converted to scan execution programs and executed in a sequence program.
The following instructions are used by the QnACPU to convert a program's type:

1) PSCAN : Converts a standby program to a scan execution program.

2) PLOW : Converts a standby program to a low-speed execution program.

3) PSTOP : Converts a scan execution program or low-speed execution program to a standby program.

4) POFF : Converts a scan execution program or low-speed execution program to a standby program.
(Switching to the standby program takes place after output is turned OFF.)

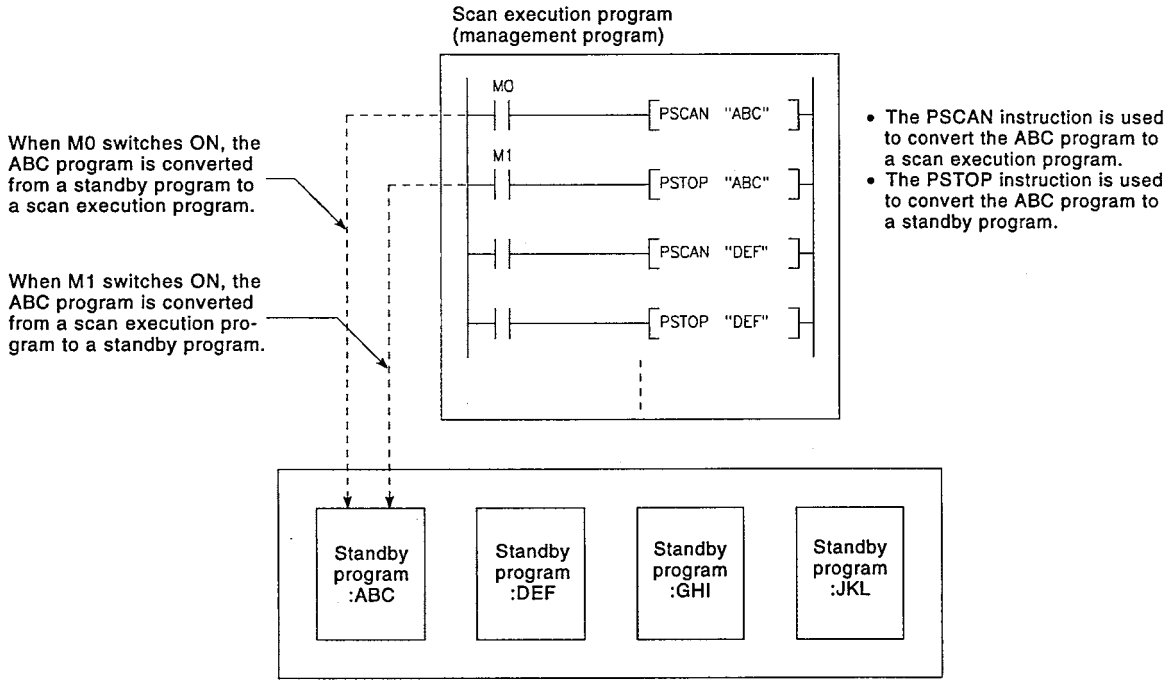
<table>
<tr><th>Executed instruction / Execution type before change</th><th>PSCAN</th><th>PSTOP</th><th>POFF</th><th>PLOW</th></tr>
<tr><td>Scan execution type</td><td>No change - remains scan type execution.</td><td rowspan="2">Becomes standby type.</td><td rowspan="2">Output turned OFF in next scan.
Becomes standby type from the next scan after that.</td><td rowspan="3">Becomes low-speed type.</td></tr>
<tr><td>Initial execution type</td><td rowspan="2">Becomes scan execution type.</td></tr>
<tr><td>Standby type</td><td>No change - remains standby type.</td><td>No processing.</td></tr>
<tr><td>Low-speed execution type</td><td>Low-speed execution type execution is stopped: becomes scan execution type from the next scan.
(Execution from step 0)</td><td>Low-speed execution type execution is stopped: becomes standby type from next scan.</td><td>Low-speed type execution is stopped, and output is turned OFF in the next scan.
Becomes standby type from the next scan after that.</td><td>No change - remains low-speed type.</td></tr>
</table>

(b) The following methods can be used to convert a program which is to be executed.

1) Selecting the program to be executed from a single management program:

- Using a constantly executed scan execution program as the management program, a standby program which conforms to the designated conditions is converted to a scan execution program and is executed.
  Scan execution programs which are not required can be converted to standby programs.

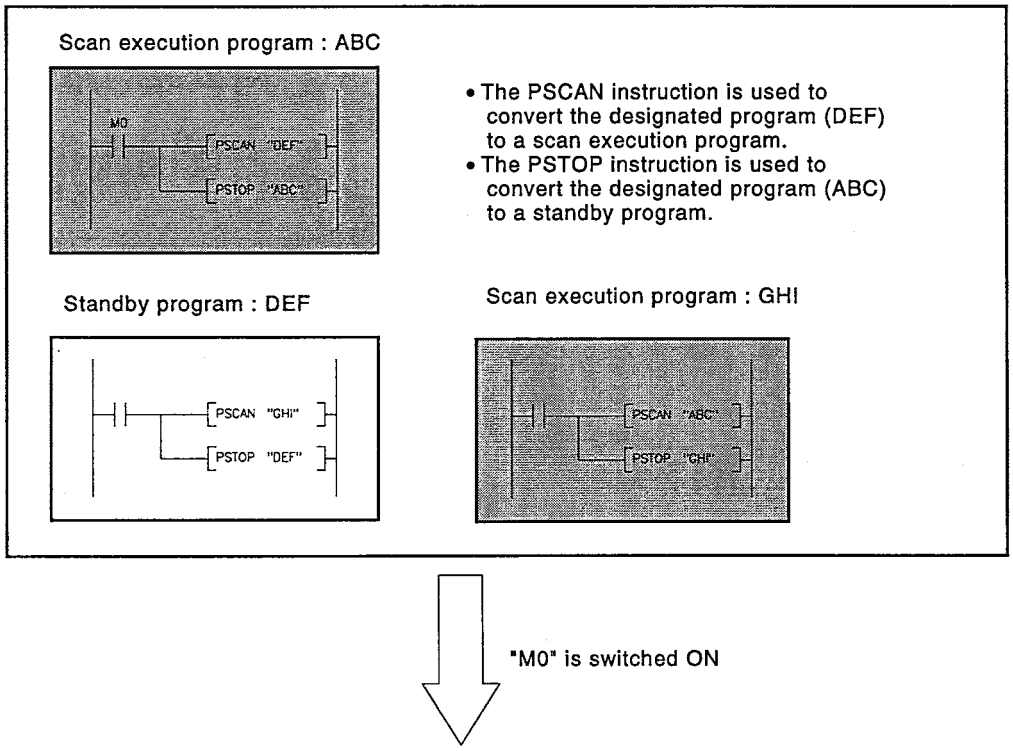
- The operation which occurs when "ABC", "DEF", "GHI", and "JKL" standby programs (at a single management program) are converted is illustrated below.



2) Converting the scan execution program being executed to another type of program:

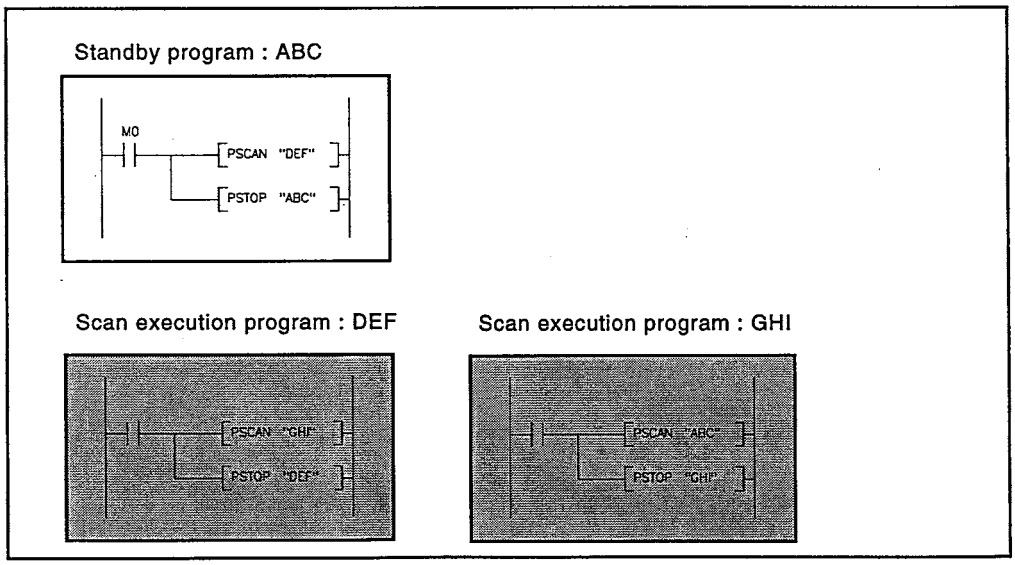
- For the scan execution program being executed, the next program to be executed is converted from a standby program to a scan execution program.
- In the illustration below, the ABC and GHI programs are designated as scan execution programs, and DEF is designated as a standby program. The illustration shows the operation which occurs when the ABC and DEF program types are converted when the conditions are satisfied.

[Before execution of PSCAN and PSTOP instructions]

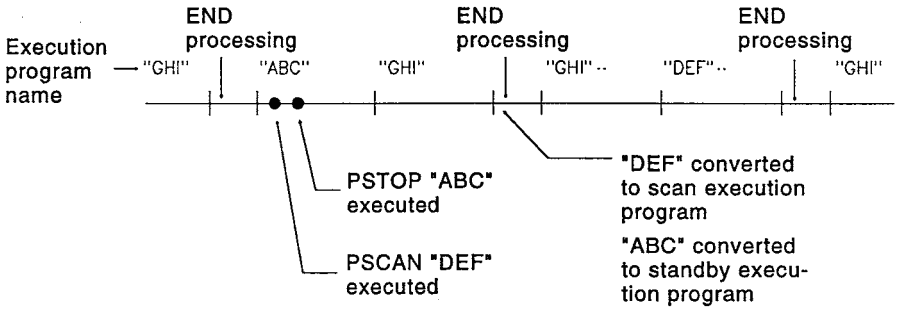


"M0" is switched ON

[After execution of PSCAN and PSTOP instructions]



(c) As program execution type conversions by PSCAN and PSTOP instructions occur at the END processing, such conversions are impossible during program execution.
When different execution types have been set for the same program in the same scan, the execution type will be that specified by the execution switching command that was executed last.



REMARK

1) *: The order of GHI and DEF program execution is determined by the program settings parameters.

(4) Precautions for creating stand-by type programs

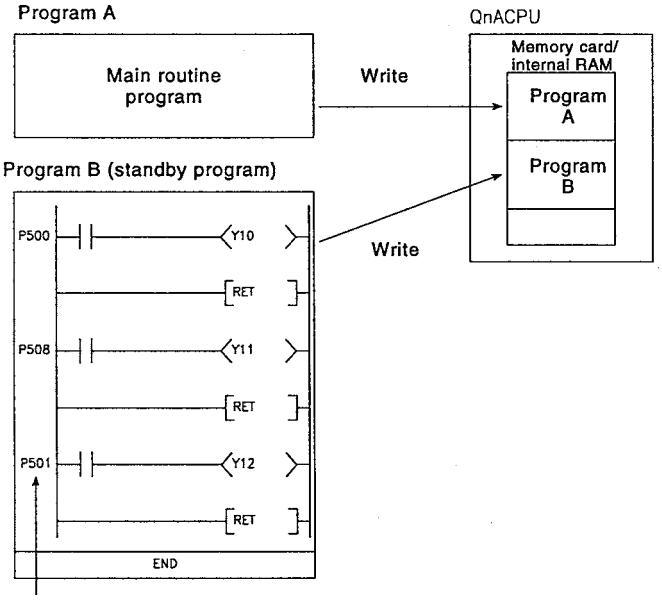
(a) Because current value is updated and contact ON/OFF is switched when the OUT T□ instruction is executed, timers cannot be used in stand-by type programs.
However when using the program set the stand-by type changing into the scan execution type, the timer is available.

(b) Gathering sub-routine programs into a single program

1) Create the sub-routine programs in order, beginning from step 0 of the standby program. An END instruction is required at the end of the sub-routine program.

2) Because there are no restrictions regarding the creation sequence of sub-routine programs, the pointer numbers need not be assigned in ascending order when creating multiple sub-routine programs.

3) Use only common pointers. *
Sub-routine programs with common pointers can be called from all programs executed by the QnACPU.



4) When local devices are used in sub-routine programs, operation is carried out in accordance with the local device values at the origin of the sub-routine call (program in which the CALL/ECALL instruction is executed).
Local device values are not stored or reset before or after executing the sub-routine program of a standby program.

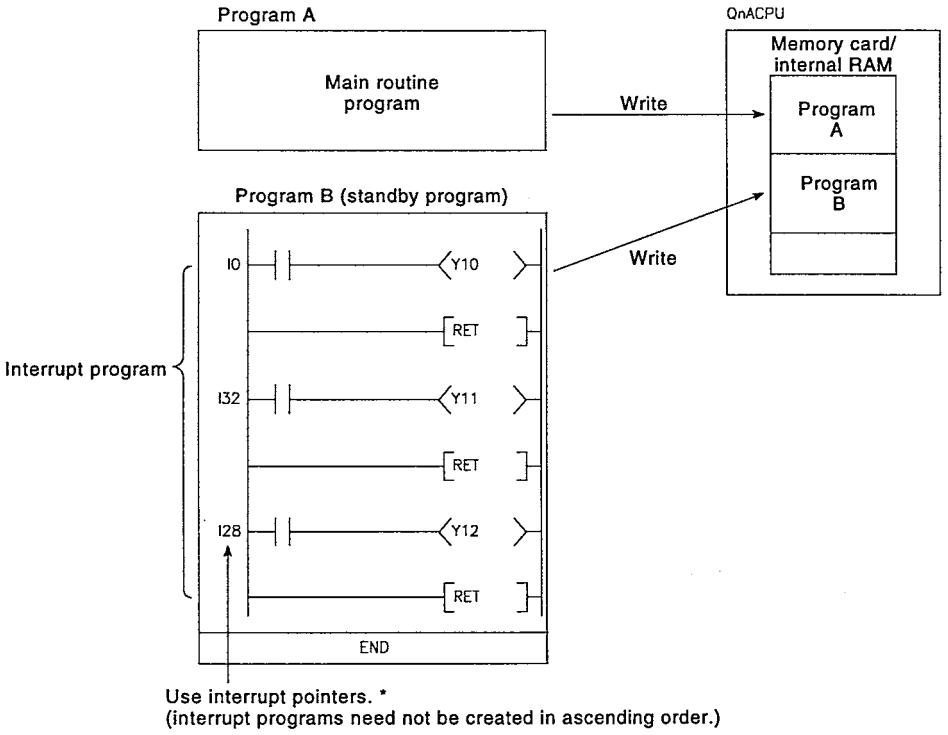
REMARK

1) *: See Section 4.9.2 for details regarding common pointers.

(c) Gathering interrupt programs into a single program

1) Create the interrupt programs in order, beginning from step 0 of the standby program. An END instruction is required at the end of the interrupt program.

2) Because there are no restrictions regarding the creation sequence of interrupt programs, the pointer numbers need not be assigned in ascending order when creating multiple interrupt programs.



REMARK

1) *: See Section 4.10 for details regarding interrupt pointers.

3.3 Input/Output Processing & Response Lag

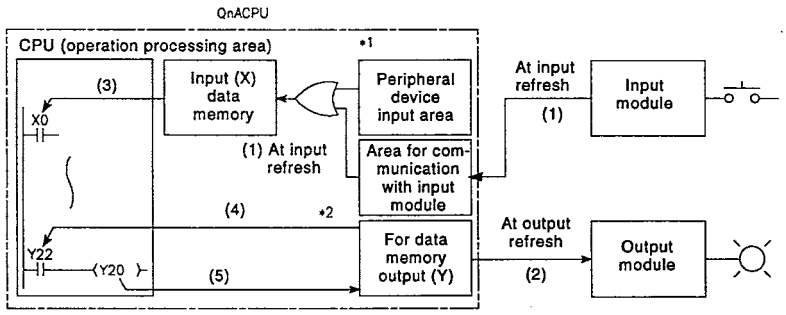
The QnACPU features a refresh type input/output processing format in which a batch communication with the input/output module occurs at END processing.
A direct communication format is also possible by using direct access inputs /outputs at the sequence program to enable direct communication with the input/output module when the sequence program instructions are executed.
For details regarding direct inputs and direct outputs, refer to Sections 4.2.1 and 4.2.2, respectively.

3.3.1 Refresh mode

(1) Definition
With the refresh mode, batch communication with the input/output modules occurs at END processing.

(a) Batch reading of the input module ON/OFF information is executed in the QnACPU's internal input data memory when END processing occurs. This ON/OFF data (in the input data memory) is then used for processing which occurs when a sequence program is executed.

(b) The processing result of the output (Y) sequence program is output to the QnACPU's internal output data memory, and batch output of the ON/OFF data (in output data memory) to the output module is executed when END processing occurs.



- Input refresh:
  Input information is read in a batch ((1)) from the input module at END processing, and is stored in the input (X) data memory by an OR operation in the peripheral device input area.
- Output refres1h:
  Data in the output (Y) data memory is output in a batch ((2)) to the output module at END processing.
- When an input contact instruction has been executed:
  Input information is read ((3)) from the input (X) data memory, and a sequence program is executed.
- When an output contact instruction has been executed:
  Output information is read ((4)) from the output (Y) data memory, and a sequence program is executed.
- When an output OUT instruction has been executed:
  The sequence program operation result ((5)) is stored in the output (Y) data memory.

Fig.3.5 Input/Output Information Flow at Refresh Mode

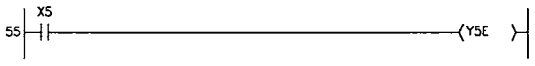
REMARKS

1) *1: See Section 3.3.2, item 1).
2) *2: See Section 3.3.2, item 2).

(2) Response lag

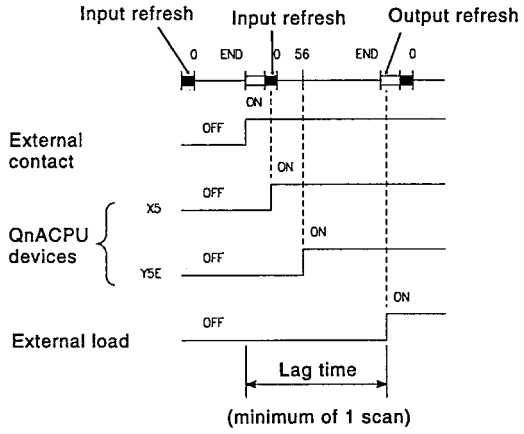
Output response lags of up to 2 scans can result from input module changes. (See Fig. 3.6)

Ladder examples



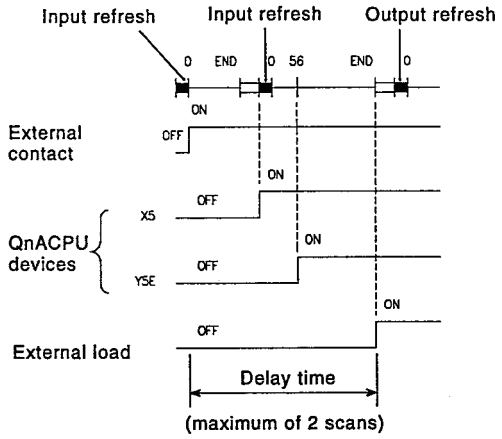
Ladder for switching the Y5E output ON in response to an X5 input ON.

Fastest possible Y5E ON



The fastest possible Y5E ON occurs if the external contact is switched ON immediately prior to the refresh operation. X5 then switches ON at the input refresh, Y5E at step 56 switches ON, and the external load switches ON at the output refresh following execution of the END instruction. In this case, the time lag between the external contact ON and the external load ON is 1 scan.

Slowest possible Y5E ON



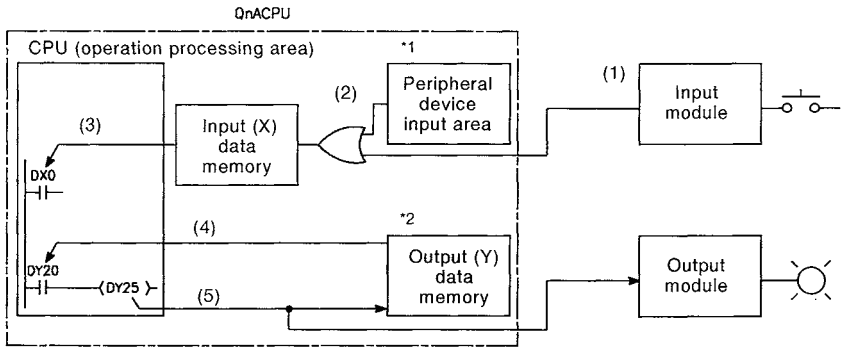
The slowest possible Y5E ON occurs if the external contact is switched ON immediately after the refresh operation. X5 then switches ON at the next input refresh, Y5E at step 56 switches ON, and the external load switches ON at the output refresh following execution of the END instruction.
In this case, the time lag between the external contact ON and the external load ON is 2 scans.

Fig.3.6 Output "Y" Change In Response to Input "X" Change

3.3.2 Direct mode

(1) Definition

In the direct mode the communication with the input/output modules is performed when executing sequence program instructions.
With QnACPU, direct mode I/O processing can be executed by using direct access inputs (DX) and direct access outputs (DY).



- When an input contact instruction has been executed:
  An OR operation is executed for the input module's input information ((1)) and the peripheral device input area's input information ((2)), and the result is stored in the input (X) data memory. This data is then used as input information ((3)) at sequence program execution.

- When an output contact instruction has been executed:
  Output information ((4)) is read from the output (Y) data memory, and a sequence program is executed.

- When an output OUT instruction has been executed:
  The sequence program's operation result ((5)) is output to the output module, and is stored in the output (Y) data memory.

Fig.3.7 Input/Output Information Flow at Direct Mode

REMARKS

1) *1: The peripheral device input area can be switched ON and OFF by the following:
   - Test operation by peripheral device.
   - A link refresh by the MELSECNET (/B) data link system.
   - A network refresh by the MELSECNET /10 network system.
   - Writing from a serial communication module.
   - Automatic refresh of CC-Link.

2) *2: The output (Y) data memory can be switched ON and OFF by the following:
   - Test operation by peripheral device.
   - A link refresh by the MELSECNET (/B) data link system.
   - A network refresh by the MELSECNET /10 network system.
   - Writing from the serial communication module.
   - Automatic refresh of MELSECNET/MINI or CC-Link

POINT

(1) When specifying the input (X) as the receive data storage device in automatic refresh setting for MELSECNET/MINI or CC-Link, use the I/O number later than those used with the module loaded on the main or extension base.
If the I/O number used for the I/O of the receive data storage device is within the I/O number range used with the module loaded on the main or extension base, the CPU module will import both the input ON/OFF data from the input module and the automatic refresh ON/OFF data of MELSECNET/MINI or CC-Link.
Hence, the CPU module will result in an input (X) fault.

(2) When turning ON/OFF the input (X) using the sequence program instruction, use the same input number as indicated below.
•Input number used with the module loaded on the main or extension base.
•Input number used with MELSECNET(/B)
•Input number used with MELSECNET/MINI or CC-Link
When you use the same input number as indicated above, the data imported in the refresh of the input module or MELSECNET(/B) or in the automatic refresh of MELSECNET/MINI or CC-Link is written over the ON/OFF status of the sequence program instruction.

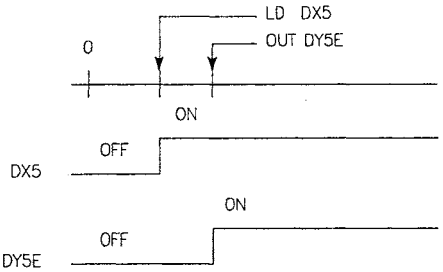
(2) Response lag

Output response lags of up to 1 scan can result from input module changes. (See Fig. 3.8)

Ladder examples

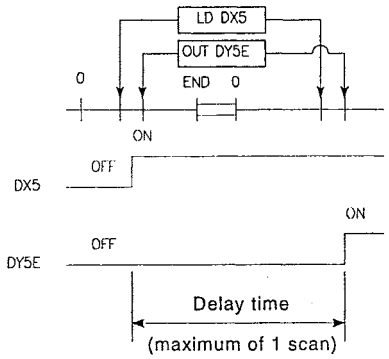
Ladder for switching the DY5E output ON in response to an DX5 input ON.

Fastest possible DY5E ON



The fastest possible DY5E output ON occurs if the DX5 input is switched ON immediately prior to the step 55 operation. If DX5 is ON when step 55's LDDX5 is executed, DY5E will switch ON within that scan.
This condition represents the minimum time lag between the DX5 input ON and the DY5E output ON.

Slowest possible DY5E ON



The slowest possible DY5E output ON occurs if the DX5 input is switched ON immediately after the step 55 operation. In this case, the DY5E output will switch ON draing the next scan.
This condition represents the maximum time lag (1 scan) between the DX5 input ON and the DY5E output ON.

Fig. 3.8 Output "Y" Change in Response to Input "X" Change

3.4 Numeric Values which Can Be Used in Sequence Programs

Numeric and alphabetic data are expressed by "0" (OFF) and "1" (ON) numerals in the QnACPU.
This method of expression is called "binary code" (BIN).
The hexadecimal (HEX) expression method in which BIN data are expressed in 4-bit units, and the BCD (binary coded decimal) expression method are also possible for the QnACPU.
The numeric expressions for the BIN, HEX, BCD, and Decimal (DEC) notations are shown in Table 3.1 below.

Table 3.1 BIN, HEX, BCD, and Decimal Numeric Expressions

DEC (Decimal)	HEX (Hexadecimal)	BIN (Binary)	BCD (Binary Coded Decimal)
0	0	0	0
1	1	1	1
2	2	10	10
3	3	11	11
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
9	9	1001	1001
10	A	1010	1 0000
11	B	1011	1 0001
12	C	1100	1 0010
13	D	1101	1 0011
14	E	1110	1 0100
15	F	1111	1 0101
16	10	1 0000	1 0110
17	11	1 0001	1 0111
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
47	2F	10 1111	100 0111

Single precision floating decimal point real numbers may also be used.
(See Section 3.4.4)

(1) External numeric inputs to QnACPU

When designating numeric settings for the QnACPU from an external source (digital switch, etc.), a BCD (binary coded decimal) setting can be designated which is the same as a decimal setting.
However, because the BCD method involves BIN expressions being processed in the same manner as decimal expressions, the QnACPU operation based on such values will be different from the operation specified by the designated value.
A BIN instruction is therefore provided for the QnACPU to convert BCD input data to the BIN data which is used by the QnACPU.
A program which converts numeric data to BIN data can be created at the sequence program in order to allow numeric settings to be designated from an external source without regard to the corresponding BIN values.

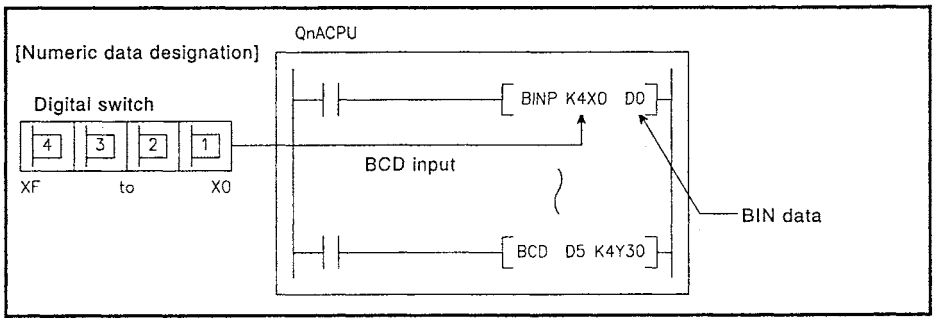


Fig. 3.9 Digital Switch Data Input to QnACPU

(2) External numeric outputs from QnACPU

A digital display can be used to display numeric data which is output from the QnACPU. However, because the QnACPU uses BIN data, it cannot be displayed at the digital display as is. A BCD instruction is therefore provided for the QnACPU to convert the BIN data to BCD data. A program which converts BIN data to BCD data can be created at the sequence program in order to display the output data in a manner identical to decimal data.

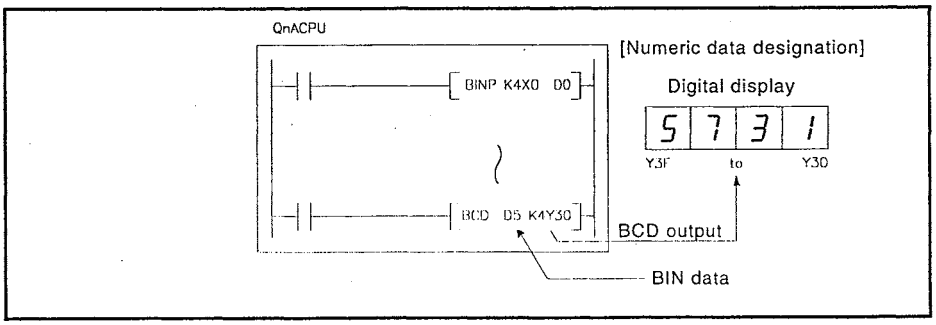


Fig.3.10 Digital Display of Data from QnACPU

3.4.1 BIN (Binary Code)

(1) Binary code

In binary code, numeric values are expressed by numerals "0" (OFF) and "1" (ON) numerals. When counting in the decimal system, a carry to the "tens" column occurs following 9 (8-9-10). In the binary system, this carry occurs following 1 (0-1-10). The binary "10" therefore represents the decimal "2". Binary values and their respective decimal values are shown in Fig.3.2 below.

Table 3.2 Binary and Decimal Numeric Value Comparison

DEC (Decimal)	BIN (Binary)	
0	0000	
1	0001	Carry
2	0010	
3	0011	Carry
4	0100	
5	0101	
6	0110	
7	0111	Carry
8	1000	
9	1001	
10	1010	
11	1011	

(2) Binary numeric expression

QnACPU registers (data registers, link registers, etc.) consist of 16 bits, with a "2^n" value is allocated to each of the register bits.
The most significant bit (initial bit) is used to discriminate between "positive" and "negative".

1. When most significant bit is "0"...Positive

2. When most significant bit is "1"...Negative

The numeric expressions for the QnACPU registers are shown in Fig.3.11 below.

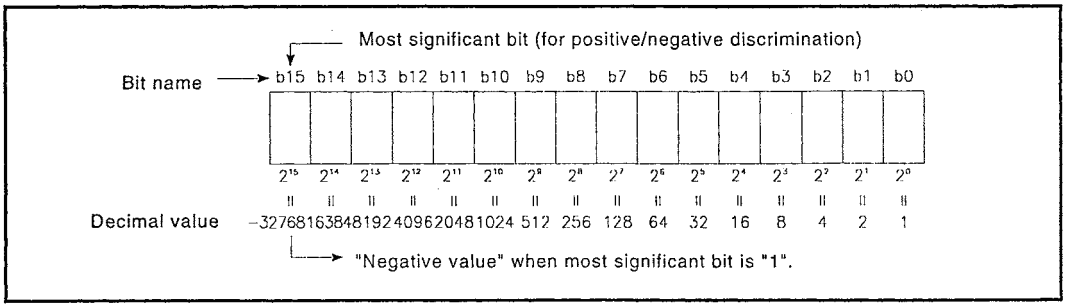


Fig.3.11 Numeric Expressions for QnACPU Registers

(a) Usable numeric data for QnACPU

As shown in Fig.3.11, the numeric expression range is -32768 to 32767. Therefore, numeric data within this range can be stored in the QnACPU registers.

3.4.2 HEX (Hexadecimal)

(1) Hexadecimal notation

In the hexadecimal system, 4 bits of binary data are expressed by 1 digit. 4 bits of binary data can express 16 values (0-15).
In the hexadecimal system, values from 0 to 15 are expressed by 1 digit. This is accomplished by using alphabetic characters following "9", with a carry occurring after "F", as follows:
A comparison of binary, hexadecimal, and decimal numeric expressions is shown in Table 3.3 below.

Table 3.3 Comparison of BIN, HEX, & DEC Numeric Expressions

DEC (Decimal)	HEX (Hexadecimal)	BIN (Binary)
0	0	0
1	1	1
2	2	10
3	3	11
.	.	.
.	.	.
.	.	.
.	.	.
.	.	.
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111
16	10	1 0000
17	11	1 0001
.	.	.
.	.	.
.	.	.
.	.	.
.	.	.
47	2F	10 1111

Carry

(2) Hexadecimal numeric expression

QnACPU registers (data registers, link registers, etc.) consist of 16 bits. Therefore, as expressed in hexadecimal code, the numeric value range which can be stored is 0 to $FFFF_H$.

3.4.3 BCD (Binary Coded Decimal)

(1) BCD notation

BCD numeric expressions are binary expressions with a carry format identical to that of the decimal system.
As with the hexadecimal system, BCD expressions are the equivalent of 4 binary bits, although the BCD system does not use the A-F alphabetic characters.
A comparison of binary, BCD, and decimal numeric expressions is shown in Table 3.4 below.

Table 3.4 Comparison of Binary, BCD, and Decimal Numeric Expressions

DEC (Decimal)	BIN (Binary)	BCD (Binary Coded Decimal)
0	0	0
1	1	1
2	10	10
3	11	11
4	100	100
5	101	101
6	110	110
7	111	111
8	1000	1000
9	1001	1001
10	1010	1 0000
11	1011	1 0001
12	1100	1 0010

Carry

(2) QnACPU registers (data registers, link registers, etc.) consist of 16 bits. Therefore, as expressed in BCD code, the range of numeric values to be stored is 0-9999.

3.4.4 Real numbers

(1) Real numbers

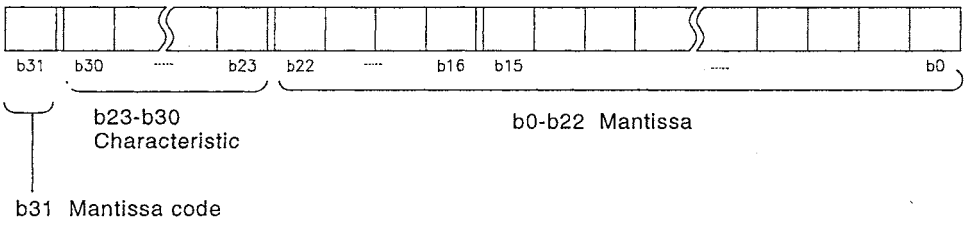
Real numbers are single precision floating decimal point data.

(2) Internal expression of floating decimal point data

The QnACPU's internal expression of received floating decimal point real number data is explained below.
Floating decimal point data is expressed as shown below, using 2 word devices.

1. [Mantissa] x 2 (characteristic)

The bit configuration used for internal expression of floating decimal point data is shown and explained below.



- Mantissa code: The mantissa code is expressed at b31 as follows.
  0: Positive
  1: Negative

- Characteristic: The "n" of "2n" is expressed in various ways at b23-b30, depending on the b23-b30 BIN value.

b23 - b30	FF$_H$	FE$_H$	FD$_H$		81$_H$	80$_H$	7F$_H$	7E$_H$		02$_H$	01$_H$	00$_H$
n	Not used	127	126		2	1	0	-1		-125	-126	Not used

- Mantissa: For a binary value of 1.XXXXXX..., the "XXXXXX" portion of the value is expressed at b0-b22 (23 bits).

POINTS

- The monitor function for peripheral devices permits monitoring the data on floating decimal point of the QnACPU.
- For a "0" value, "0" will be indicated at all the b0-b31 bits.

- Calculation examples are shown below (the nnnnn "X" indicates an X-system data expression)

(1) Storing "10"

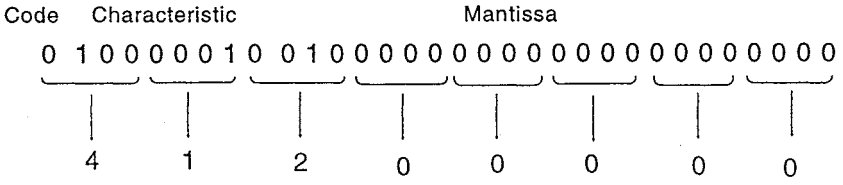
$(10)_{10} \rightarrow (1010)_2 \rightarrow (1.01000..... \times 2^3)_2$

Mantissa code: Positive → 0

Characteristic : 3 → 82_H → $(10000010)_2$

Mantissa : $(010\ 00000\ 00000\ 00000\ 00000)_2$

The data expression will therefore be 41200000_H, as shown below.



(2) Storing "0.75"

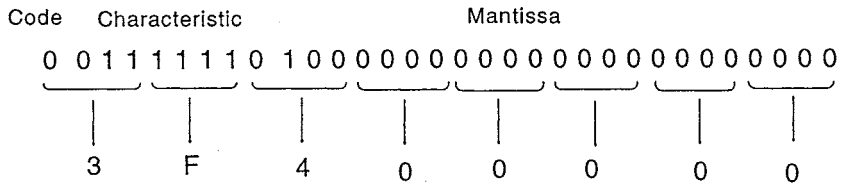
$(0.75)_{10} \rightarrow (0.11)_2 \rightarrow (1.100.... \times 2^{-1})_2$

Mantissa code: Positive → 0

Characteristic : -1 → $7E_H$ → $(01111110)_2$

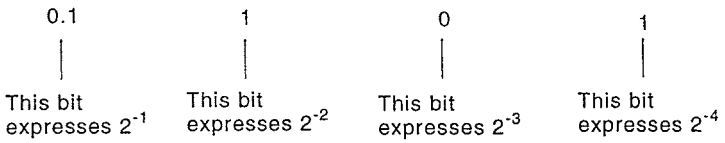
Mantissa : $(100\ 00000\ 00000\ 00000\ 00000)_2$

The data expression will therefore be $3F400000_H$, as shown below.



REMARK

At the binary system, the portion of the value following the decimal point is calculated as follows:



$(0.1101)_2 = 2^{-1}+2^{-2}+2^{-4} = 0.5+0.25+0.125=(0.875)_{10}$

3.5 Character String Data

(1) Character String Data

The QnACPU uses ASCII code data.

(2) ASCII code character strings

ASCII code character strings are shown in the Table below. "00H" (NUL code) is used at the end of a character string.

b8	b7	b6	b5	b4	b3	b2	b1	Column / Row	0	0	0	0	0	0	0	0
									0	0	0	0	1	1	1	1
									0	0	1	1	0	0	1	1
									0	1	0	1	0	1	0	1
									0	1	2	3	4	5	6	7
				0	0	0	0	0	NUL		(SP)	0	@	P	`	p
				0	0	0	1	1			!	1	A	Q	a	q
				0	0	1	0	2			"	2	B	R	b	r
				0	0	1	1	3			#	3	C	S	c	s
				0	1	0	0	4			$	4	D	T	d	t
				0	1	0	1	5			%	5	E	U	e	u
				0	1	1	0	6			&	6	F	V	f	v
				0	1	1	1	7			'	7	G	W	g	w
				1	0	0	0	8			(	8	H	X	h	x
				1	0	0	1	9			)	9	I	Y	i	y
				1	0	1	0	A			*	:	J	Z	j	z
				1	0	1	1	B			+	;	K	[	k	{
				1	1	0	0	C			,	<	L	\	l	|
				1	1	0	1	D			-	=	M	]	m	}
				1	1	1	0	E			.	>	N	^	n	~
				1	1	1	1	F			/	?	O	_	o	

4. DEVICES

4.1 Device List

4.1.1 Device list

The names and data ranges of devices which can be used in the QnACPU are shown in Table 4.1 below.

Table 4.1 Device List

Class	Type	Device Name	Default Values		Parameter Designated Setting Range	Reference Section
			Number of Points	Range Used		
Internal user devices	Bit devices	Input	8192 points	X0 to X1FFF	Fixed	4.2.1
		Output	8192 points	Y0 to Y1FFF		4.2.2
		Internal relay	8192 points	M0 to M8191	Change possible for 28 k words or less. *	4.2.3
		Latch relay	8192 points	L0 to L8191		4.2.4
		Annunciator	2048 points	F0 to F2047		4.2.5
		Edge relay	2048 points	V0 to V2047		4.2.6
		Step relay*3	8192 points	S0 to S511 per block		4.2.9
		Link special relay*3	2048 points	SB0 to SB7FF		4.2.8
		Link relay	8192 points	B0 to B1FFF		4.2.7
	Word devices	Timer*1	2048 points	T0 to T2047		4.2.10
		Retentive timer*1	0 points	(ST0 to ST2047)		
		Counter*1	1024 points	C0 to C1023		4.2.11
		Data register	12288 points	D0 to D12287		4.2.12
		Link register	8192 points	W0 to W1FFF		4.2.13
		Link special register*3	2048 points	SW0 to SW7FF		4.3.14
Internal system devices	Bit devices	Function input	5 points	FX0 to FX4	Impossible	4.3.1
		Function output	5 points	FY0 to FY4		4.3.1
		Special relay	2048 points	SM0 to SM2047		4.3.2
	Word devices	Function register	5 points	FD0 to FD4		4.3.1
		Special register	2048 points	SD0 to SD2047		4.3.3
Link direct devices	Bit device	Link input	8192 points	Jn\X0 to Jn\X1FFF	Impossible	4.4
		Link output	8192 points	Jn\Y0 to Jn\Y1FFF		
		Link relay	8192 points	Jn\B0 to Jn\B1FFF		
		Link special relay	512 points	Jn\SB0 to Jn\SB1FF		
	Word device	Link register	8192 points	Jn\W0 to Jn\W1FFF		
		Link special register	512 points	Jn\SW0 to Jn\SW1FF		
Special function module device	Word device	Buffer register	16384 points	Un\G0 to Un\G16383	Impossible	4.5

Class	Type	Device Name	Default Values		Parameter Designated Setting Range	Reference Section
			Number of Points	Range Used		
Index register	Word device	Index register	16 points	Z0 to Z15	Impossible	4.6
File register	Word device	File register	0 points	——	0 to 1024 k points (1 k units)	4.7
Nesting	——	Nesting	15 points	N0 to N14	Impossible	4.8
Pointers	——	Pointer	4096 points	P0 to P4095	Impossible	4.9
		Interrupt pointer	48 points	I0 to 147		4.10
Other	Bit devices	SFC block	320 points	BL0 to BL319	Impossible	4.11.1
		SFC transition device	512 points	TR0 to TR511		4.11.2
	——	Network No.	256 points	J1 to J255		4.11.3
		I/O No.		U0 to UFF		4.11.4
Constants	——	Decimal constants	K-2147483648-K2147483647			4.12.1
		Hexadecimal constants	H0 to HFFFFFFFF			4.12.1
		Real number constants	E ± 1.17549-38 to E ± 3.40282+38			4.12.3
		Character string constants	"ABC", "123"			4.12.4

REMARKS

1) *1: For the timer, retentive timer, and counter, bit devices are used for the "number of points" and the "coil", and the word device is used for the "present value".

2) *2: The actual number of usable points varies according the the special module. For details regarding the buffer memory's "number of points", refer to the Special Function Module Manual.

3) *3: Inputs, outputs, step relays, link special relays, link special registers remain at their default values, which cannot be changed.

4.1.2 Setting units in the internal user device

For all QnACPU internal user devices other than the input (X), output (Y), and step relay (S) devices, the number of points used can be changed within a 28.8 k word range by the "device setting" parameters. The items to consider when making such changes are discussed below.

(1) Setting range

(a) The number of device points is designated in 16-point unit.

(b) A maximum of 32 k points can be designated for one type of device. The maximum total number of points for the internal relay, latch relay, annunciator, edge relay, link relay, timer, retentive timer, and counter, is 64 k points. 1 points is calculated as 2 points (1 for coil, 1 for contact) for the timer, retentive timer, and counter.

(2) Memory size (Bit device size)

(a) For bit devices:
For bit devices, 16 points are calculated as 1 word.

$$\text{(Word device size)} = \frac{\text{(M+L+F+B+SB total number of points)}}{16} \text{ (Word)}$$

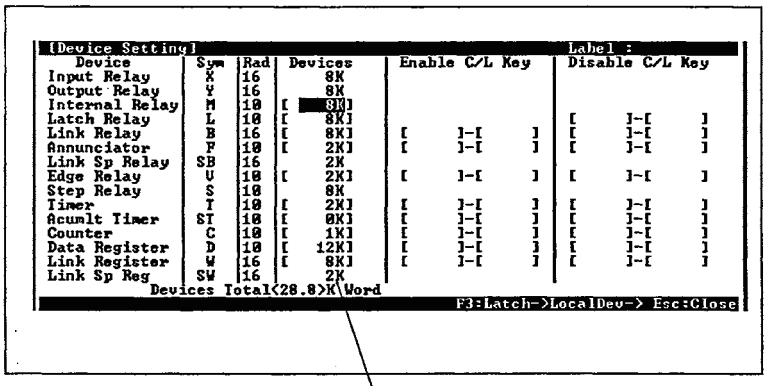
(b) For timer (T) retentive timer (ST), and Counter (C):
For the timer, retentive timer, and counter, 16 points are calculated as 18 words.

$$\text{(Timer, retentive, counter size)} = \frac{\text{(T,ST,C total number of points)}}{16} \times 18 \text{ (Words)}$$

(c) For word devices:
For data registers (D), link registers (W), and special link registers (SW), 16 points are calculated as 16 words.

$$\text{(Word device size)} = \frac{\text{(D,W,SW total number of points)}}{16} \times 16 \text{ (Words)}$$

[Device setting screen]



Default values
"Number of points" can be changed at devices where a "number of points" value is shown in brackets.

4.2 Internal User Devices

Internal user devices can be used for various user applications.
The "number of usable points" setting is designated in advance (default value) for internal user devices. However, this setting can be changed within a 28.8k word range by a peripheral device parameter setting.
(See Section 4.1 for details regarding the internal user device default value and the setting range which can be designated by parameter setting)

POINT

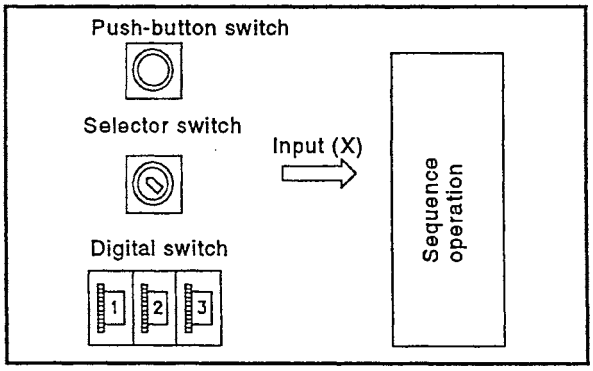
(1) When an internal user device's "number of usable points" setting is changed, files which were created under the previous setting cannot be used as they are. In order to use these files, the following operation is required after changing the "number of usable points" setting:
- The sequence program
- The SFC program

The sequence program and SFC program must be read from the QnACPU to the peripheral device, and then they must be written back to the QnACPU again.

4.2.1 Inputs (X)

(1) Definition

(a) Inputs are commands or data transmitted to the QnACPU from a peripheral device by push-button switches, selector switches, limit switches, digital switches, etc.



(b) The input point is the Xn virtual relay inside the QnACPU, with the program using the Xn's N/O contact or N/C contact.

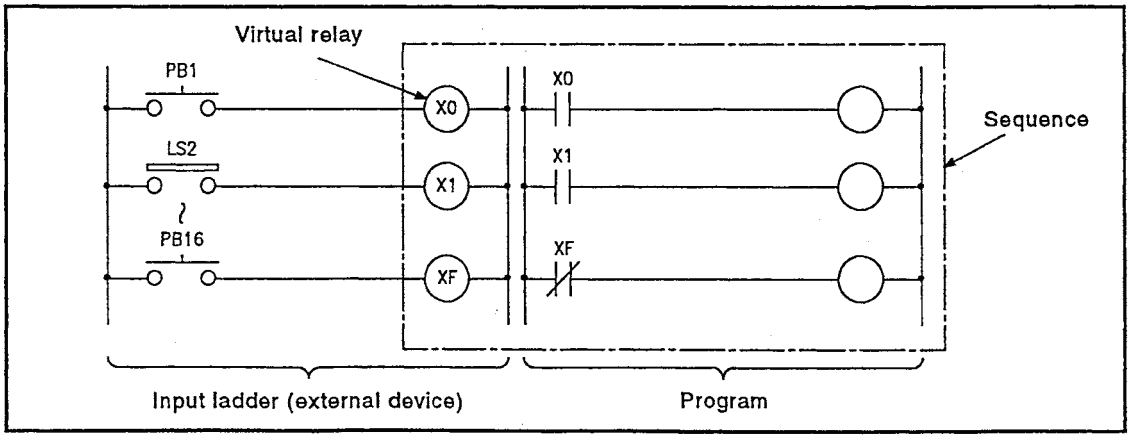


Fig.4.1 Inputs (X)

(c) There are no restrictions regarding the number of Xn N/O contacts and N/C contacts used in a program.

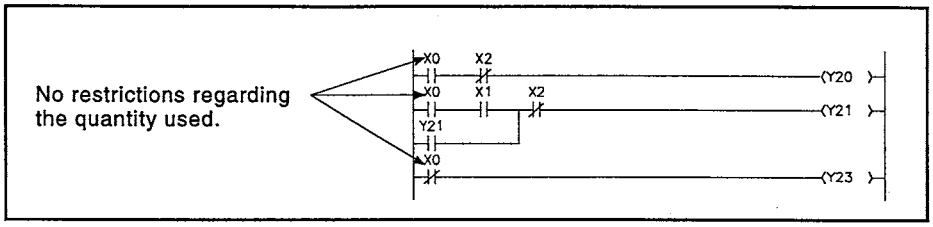


Fig.4.2 Input (x) Used in Program

(d) "X" cannot be used in the following area;

- where the output module is mounted, or;
- of the empty slots preceding the last mounted slot.

If an accidental use of the wrong "X" occurs, it may result in malfunctions of the systems.

In addition, "X" that cannot be used is sometimes displayed on the screen during monitoring. This does not give any effect on the system operation.

(2) Reading the inputs

(a) There are 2 types of input: "refresh inputs" and "direct access inputs".

1) Refresh inputs are ON/OFF data read from the input module using the refresh mode. *1
These inputs are indicated as "X□" in the sequence program.
For example, a "100" input becomes "X100".

2) Direct access inputs are ON/OFF data read from the input module using the direct mode. *2
These inputs are indicated as "DX□" in the sequence program.
For example, a "100" input becomes "DX100".

REMARKS

1) *1: See Section 3.3.1 for details regarding the refresh mode.

2) *2: See Section 3.3.2 for details regarding the direct mode.

(b) The same input number can be designated for a refresh input and a direct access input.
If used as a refresh input after being used as a direct access input, operation will be based on the ON/OFF data read at the direct access input.

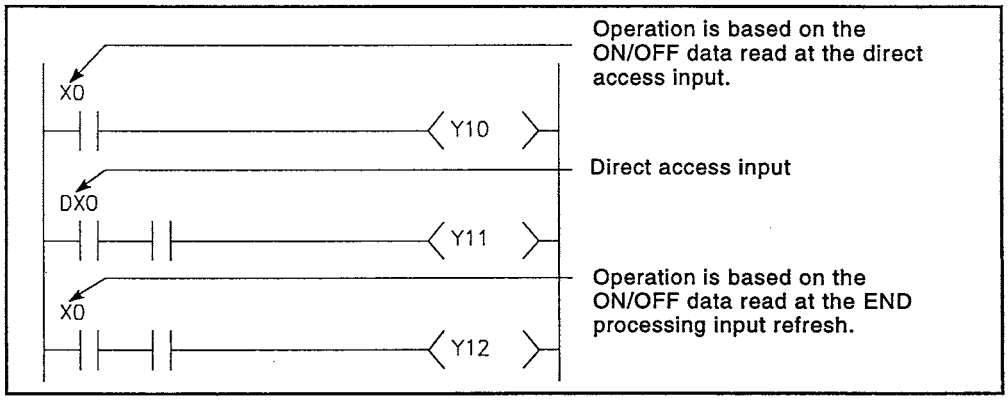


Fig.4.3 Refresh Input & Direct Access Input

POINT

(1) Direct access inputs can only be used in one point units.
They cannot be specified with digit designation.
- LD DX0 ——————Can be used
- MOV K4DX0 D0 ——Cannot be used

Digit designation for direct access inputs is not possible.

(c) Differences between refresh inputs & direct access inputs
With direct access inputs, the input module is directly accessed by the executed instruction, and the processing speed is therefore slower than that for refresh inputs.
Moreover, direct access inputs can only be used for inputs used with the input module and special function module (modules installed at base unit and extension base unit). The refresh and direct input differences are shown in Table 4.2 below.

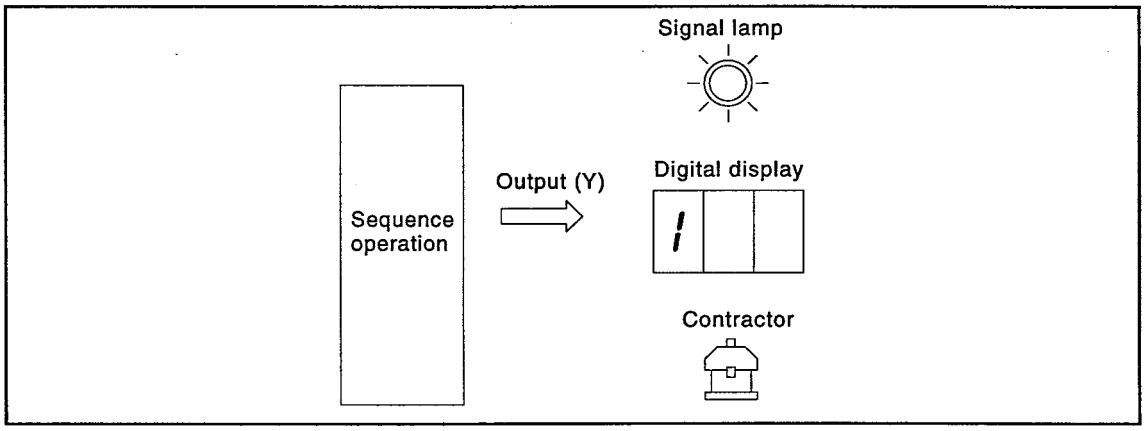
Table 4.2 Differences between Refresh Inputs & Direct Access Inputs

Item	Refresh Input	Direct Access Input
Processing speed	0.075 to 0.2 μs	Approx. 10 μs
Input module installed at base/ extension base unit	Usable	Usable
Inputs of special function module installed at base/extension base unit		
Inputs of I/O link module installed at base/extension base unit		
Inputs used at MELSECNET/10 network system	Usable	Unusable
Inputs used at MELSECNET (II/B) data link		
Inputs used at MELSECNET/MINI-S3 link		

4.2.2 Outputs (Y)

(1) Definition

(a) Outputs are program control results which are output to external destinations (solenoid, electromagnetic switch, signal lamp, digital display, etc.).



(b) Outputs occur at one N/O contact or its equivalent.

(c) There are no restrictions regarding the number of output Yn N/O contacts and N/C contacts used in a program.

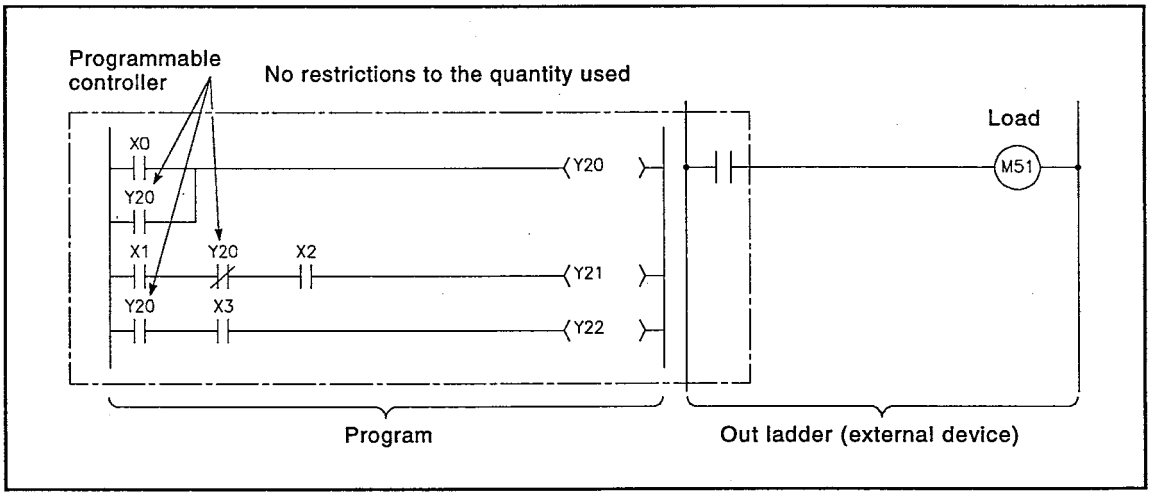
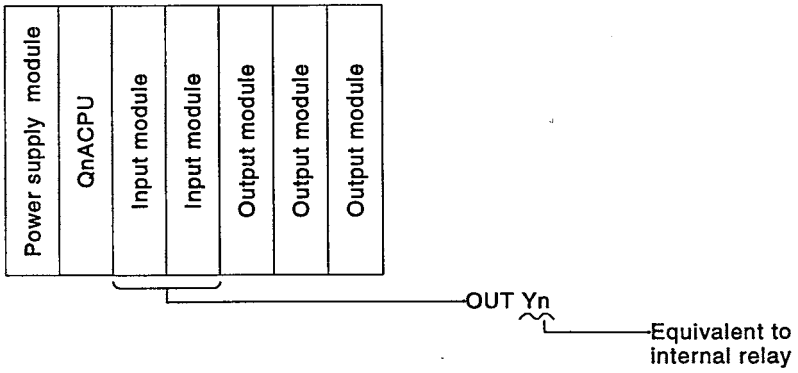


Fig.4.4 Output (Y) Operation

(2) Using outputs as internal relays (M)
"Y" inputs corresponding to vacant slots and slots where input modules are installed can serve as internal relays (M).



(3) Output method

(a) There are 2 types of output:"refresh outputs" and "direct access outputs".

1) Refresh outputs are ON/OFF data which is output to the output module using the refresh mode. *1
These outputs are indicated as "Y□" in the sequence program.
For example, a "100" input becomes "Y100".

2) Direct access outputs are ON/OFF data which is output to the output module using the direct mode. *2
These outputs are indicated as "DY□" in the sequence program.
For example, a "100" input becomes "DY100".

(b) Differences between refresh outputs & direct access outputs

With direct access outputs, the output module is directly accessed by executing an instruction, and the processing speed is therefore slower than that for refresh outputs.
Moreover, direct access outputs can only be used for outputs used with the output module and special function module (modules installed at base unit and extension base unit). The refresh and direct output differences are shown in Table 4.3 below.

Table 4.3 Differences between Refresh Outputs & Direct Access Outputs

Item	Refresh Output	Direct Access Output
Processing speed	0.075 to 0.02 μs	Approx. 10 μs
Output module installed at base/extension base unit	Usable	Usable
Outputs of special function module installed at base/extension base unit		
Outputs of I/O link module installed at base/extension base unit		
Outputs used at MELSECNET/10 network system	Usable	Unsaddle
Outputs used at MELSECNET (II/B) data link		
Outputs used at MELSECNET/MINI-S3 link		

REMARKS

1) *1: See Section 3.3.1 for details regarding the refresh mode.

2) *2: See Section 3.3.2 for details regarding the direct mode.

POINT

(1) Direct access outputs can only be used in one point units.
They cannot be specified with digit designation.

- OUT DY10 ——→ Can be used
- MOV D0 K4Y10 ——→ Cannot be used

Digit designation for direct access outputs is not possible.

4.2.3 Internal relays (M)

(1) Definition

(a) Internal relays are auxiliary relays which cannot be latched by the programmable controller's internal latch (memory backup).

All internal relays are switched OFF at the following times:
- When power is switched from OFF to ON.
- When a QnACPU reset occurs.
- When a QnACPU latch clear operation is executed.

(b) There are no restrictions regarding the number of contacts (N/O contacts, N/C contacts) used in the program.

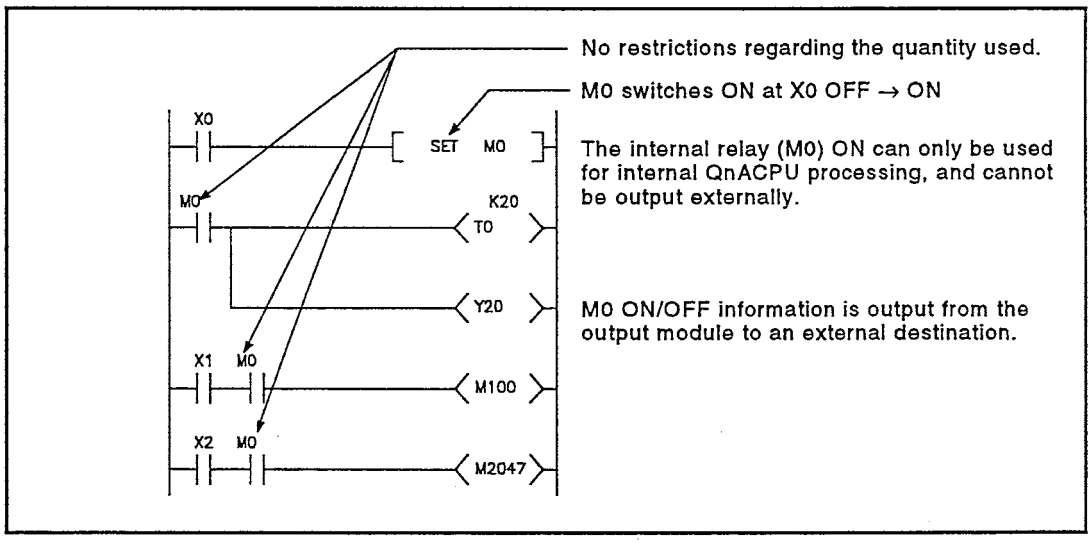


Fig.4.5 Internal Relay

(2) Procedure for external outputs

(a) Outputs (Y) are used to output sequence program operation results to an external destination.

(b) Link relays (B) are used to output ON/OFF information from MELSECNET/10 to another station.

REMARK

1) Latch relays (L) should be used when a latch (memory backup) is required. See Section 4.2.4 for details regarding latch relays.

4.2.4 Latch relays (L)

(1) Definition

(a) Latch relays are auxiliary relays which can be latched by the programmable controller's internal latch (memory backup).
Latch relay operation results (ON/OFF information) are saved even in the following cases:
· When power is switched from OFF to ON.
· When a QnACPU reset occurs.
The latch is backed up by the QnACPU battery.

(b) Latch relays can be switched OFF by the RUN/STOP key at the QnACPU. However, a latch relay cannot be switched OFF by RUN/STOP key operation if latch clear has been made ineffective for that latch relay in the device settings parameters.
For details regarding the setting for making latch clear ineffective, refer to the User's Manual of the CPU module used.

(c) There are no restrictions regarding the number of contacts (N/O contacts, N/C contacts) used in the program.

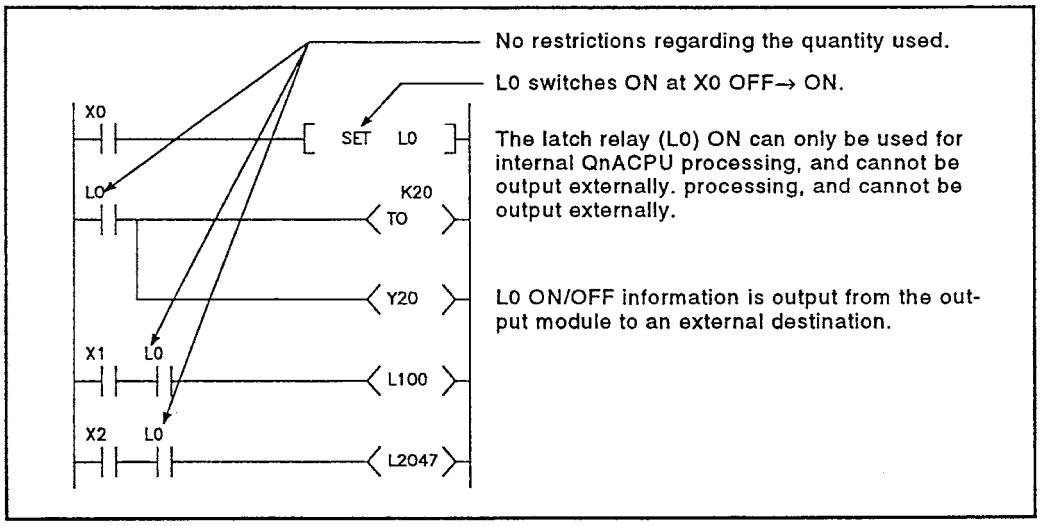


Fig.4.6 Latch Relay

(2) Procedure for external outputs

(a) Outputs (Y) are used to output sequence program operation results to an external destination.

(b) Link relays (B) are used to output ON/OFF information from MELSECNET/10 to another station.

REMARK

1) Internal relays (M) should be used when a latch (memory backup) is not required.
See Section 4.2.3 for details regarding internal relays.

4.2.5 Annunciators (F)

(1) Definition

(a) Annunciators are devices used by the user in fault detection programs.

(b) When annunciators switch ON, a special relay switches ON, and the Nos. and quantity of annunciators which switched ON are stored at the special registers.

- Special relay : SM62 ..... Switches ON if even one annunciator switches ON.
- Special register : SD62 ..... No. of first annunciator which switched ON is stored here.
  : SD63 ..... The number (quantity) of annunciators which are ON is stored here.
  : SD64 to SD79 ... Annunciator Nos. are stored in the order in which they switched ON. (The same annunciator No. is stored at SD62 and SD64.)

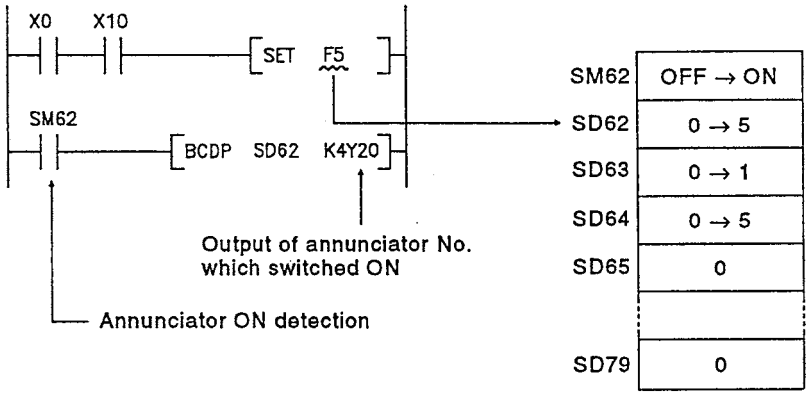
The annunciator No. stored at SD62 is also registered in the "fault history area".

(c) The use of annunciators in the fault detection program permits the user to check for the presence/absence of fault and to check the fault content (annunciator No.), by monitoring the special relay and special registers.

Example

The program which outputs the No. of the ON annunciator (F5) is shown below.

(Fault detection program)



(2) Annunciator ON procedure

(a) Annunciator ON procedure Annunciator operation can be controlled by the SET F□ and OUT F□ instructions.

1) The SET F□ instruction switches the annunciator ON at the leading edge (OFF→ON) of the input condition, and keeps the annunciator ON when the input condition switches OFF. In cases where many annunciators are used, the OUT F□ instruction can be used to speed up the scan time.

2) Although the OUT F□ instruction can switch the annunciator ON and OFF according to the input condition ON/OFF operation, it is executed in each scan.

POINT

(1) If switched ON by any method other than the SET F□ and OUT F□ instructions, the annunciator functions in the same way as the internal relay.
(Does not switch ON at SM62, and annunciator Nos. are not stored at SD62, SD64 to SD79.)

Moreover, even if an annunciator is switched OFF by the OUT F □ instruction, the special relay and special register content is not changed. Therefore, an RST F□ instruction or LEDR instruction is required. (See item (3) below,"annunciator OFF procedure & processing content")

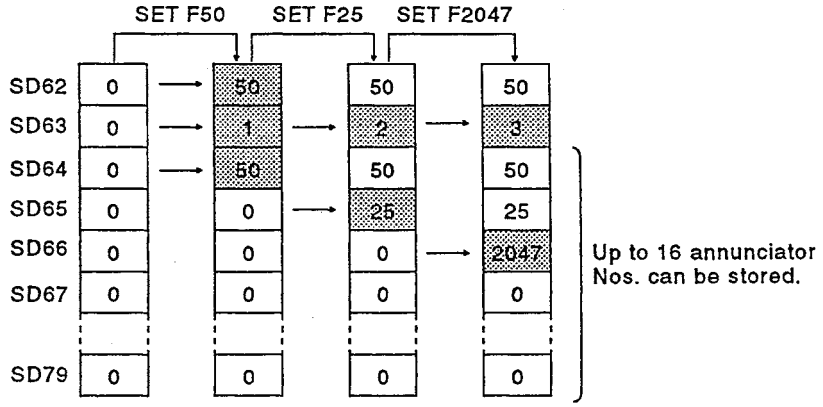
(b) Processing at annunciator ON

1) Data stored at special registers (SD62 to SD79)

a) Nos. of annunciators which switched ON are stored in order at SD64 to SD79.

b) The annunciator No. which was stored at SD64 is stored at SD62.

c) "1" is added to the SD63 value.



2) Processing at QnACPU

a) Q2ACPU(-S1), ..."USER" LED at CPU front is ON.
Q2AS(H)CPU(-S1)

b) Q3ACPU, Q4ACPU, . . . The annunciator No. stored at SD62
Q4ARCPU is displayed on the LED display
(CPU front).

(3) Annunciator OFF procedure & processing content

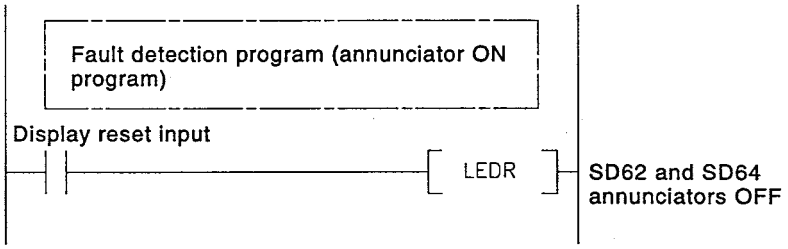
(a) Annunciator OFF procedure
An annunciator can be switched OFF by the RST F□ and LEDR instructions.

1) An annunciator No. which has been switched ON by the SET F□ instruction can be switched OFF by the RST F□ instruction.

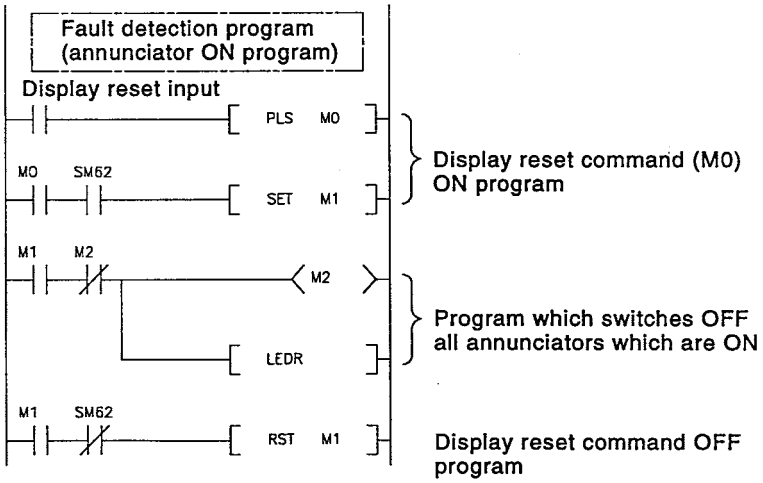
2) The LEDR instruction is used to switch OFF the annunciator Nos. stored at SD62 and SD64.

3) An annunciator No. which has been switched ON by the OUT F□ instruction is switched OFF when the OUT F□ instruction is switched OFF.
However, if an annunciator is switched OFF by the OUT F□ instruction, the "processing at annunciator OFF" (item (b) below) does not occur.
Execute the RST F□ and LEDR instructions after the annunciator has been switched OFF by the OUT F□ instruction.

1) To switch OFF only the annunciators stored at SD62 and SD64:



2) To switch OFF all annunciators which are ON:



REMARK

1) The BKRST instruction can be used to switch OFF a specified annunciator No. range. For details regarding the BKRST instruction, refer to the QnACPU Programming Manual (Common Instructions).

(b) Processing at annunciator OFF

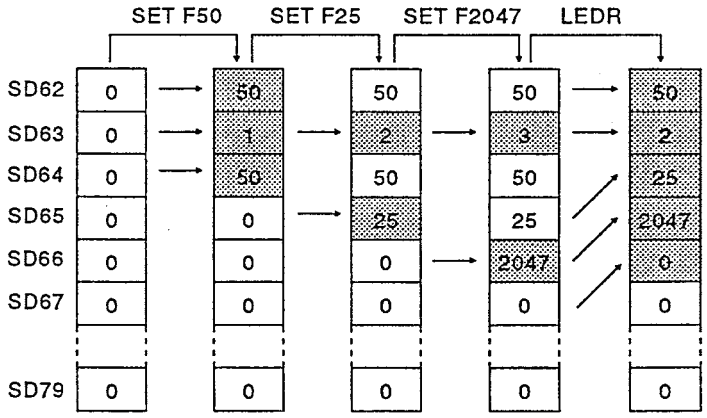
1) Special register (SD62 to SD79) data operation at LEDR instruction

a) Annunciator No. stored at SD64 is deleted, and annunciator Nos. stored at subsequent registers (SD65 to SD79) are moved up to fill the vacant space.

b) The annunciator No. stored at SD64 is stored at SD62.

c) "1" is subtracted from the SD63 value.

d) If the SD63 value is "0", SM62 is switched OFF.



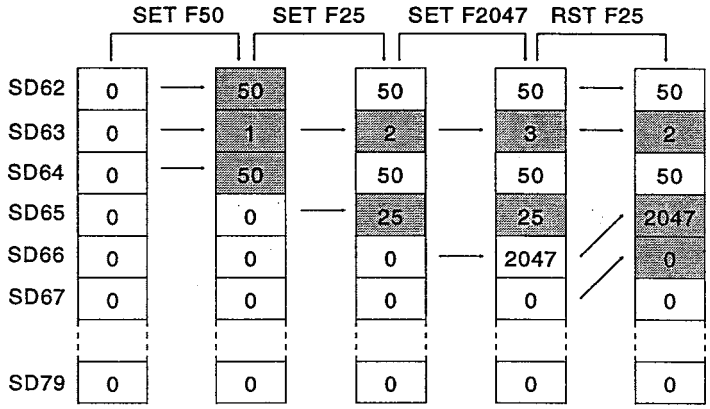
2) Special register (SD62 to SD79) data operation at RST F□ instruction, and when an annunciator is switched OFF by the OUT F□ instruction

a) The annunciator No. which was switched OFF is deleted, and all subsequent annunciator Nos. are moved up to fill the vacant space.

b) If the annunciator No. stored at SD64 was switched OFF, the new annunciator No. which is stored at SD64 is stored at SD62.

c) "1" is subtracted from the SD63 value.

d) If the SD63 value is "0", SM62 is switched OFF.



3) Processing at QnACPU

a) Q2ACPU(-S1), Q2AS(H)CPU(-S1)

- If all SD64 to SD79 annunciator Nos. are switched OFF, the "USER LED" (at CPU front) switches OFF.

b) Q3ACPU, Q4ACPU, Q4ARCPU

- Ifthe displayed annunciator number. (CPU front LED display) is switched OFF, the new annunciator No. stored at SD62 is displayed.
- If an annunciator No. other than that displayed is switched OFF, the displayed No. will not change.
- If all the SD64 to SD79 annunciator Nos. are switched OFF, the LED display will switch OFF.

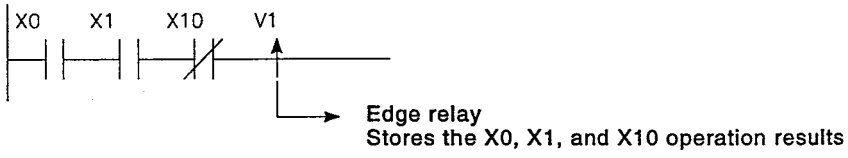
POINT

(1) If the RST F□ or LEDR instruction is not executed after switching an annunciator OFF by the OUT F□ instruction, the "processing at annunciator OFF" (see(b) above) will not occur.

4.2.6 Edge relay (V)

(1) Definition

(a) An edge relay is a device which stores the operation results (ON/OFF information) from the beginning of the ladder block. Edge relays can only be used at contacts, and cannot be used as coils.



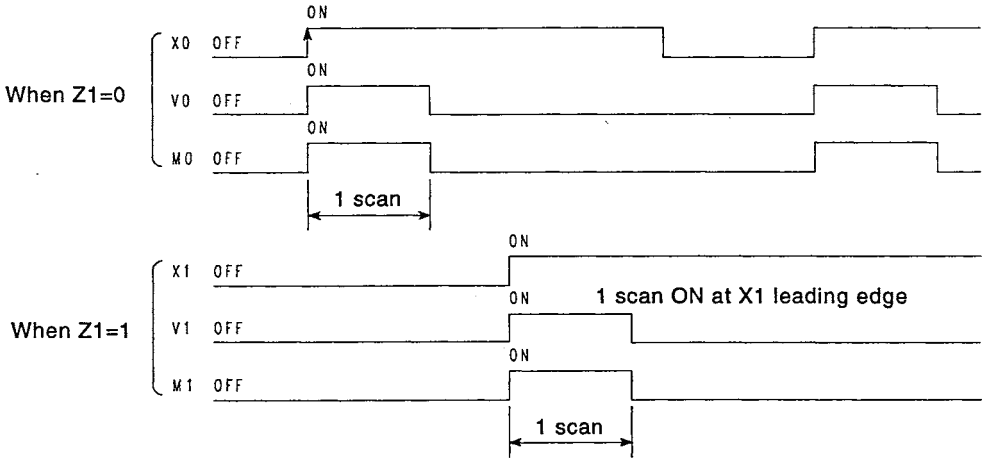
(b) The same edge relay number cannot be used twice in programs executed by the QnACPU.

(2) Edge relay applications

Edge relays are used for detecting the leading edge (OFF→ON) in programs configured using index qualification.

[Ladder example]

[Timing chart]



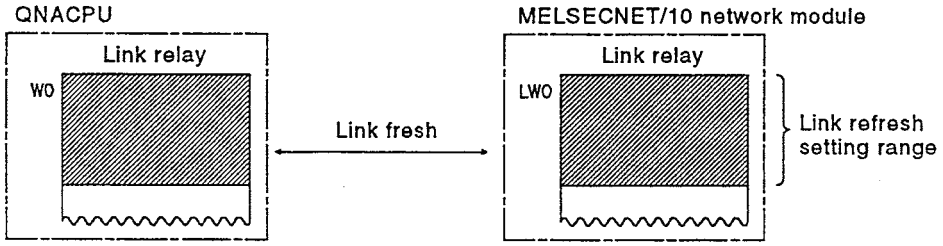
REMARK

1) *1: The ON/OFF information for X0Z1 is stored at the V0Z1 edge relay. For example, the X0 ON/OFF information is stored at V0, and the X1 ON/OFF information is stored at V1.

4.2.7 Link relays (B)

(1) Definition

(a) A link relay is the QnACPU relay used to refresh the QnACPU from the MELSECNET data link module and MELSECNET/10 network module's link relay (LB).



Internal relays or latch relays can be used for data ranges not used by the MELSECNET data link system and MELSECNET/10 network system.

- Range where no link relay latch occurs...Internal relay
- Range where link relay latch occurs.....Latch relay

(b) There are no restrictions regarding the number of contacts (N/O contacts, N/C contacts) used in the program.

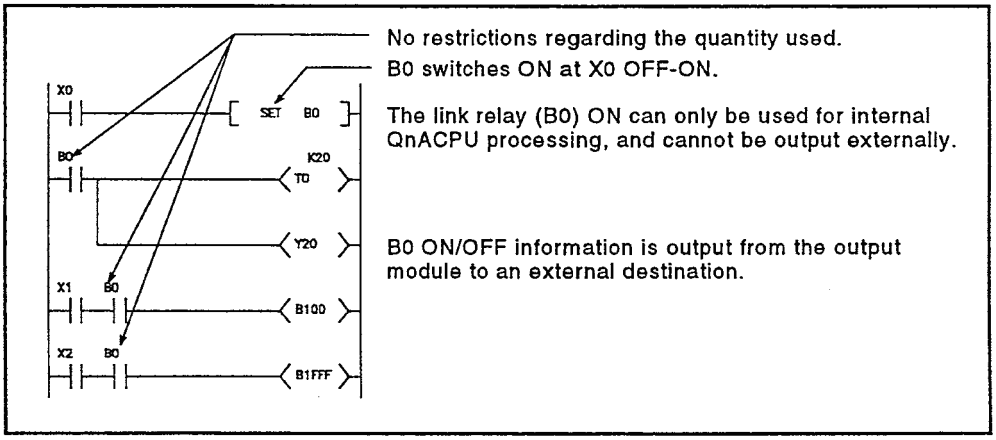


Fig.4.5 Link Relay

(2) Using link relays in the MELSECNET/10 network system

(a) If link relays are used at the MELSECNET/10 network system, the host station's ON/OFF information can be read to another station for use there. Use of link relays in the MELSECNET/10 network system permits the transfer of ON/OFF information between the control station and a normal station, and between normal stations.

(b) In order to use link relays in the MELSECNET/10 network system, a network parameter setting is required at the control station. Link relays for which no network parameter setting has been designated can be used as internal relays or latch relays.

(3) Use in a MELSECNET data link system.

(a) When link relays are used with a MELSECNET data link system, the ON/OFF statuses at the host station can be read and used at other stations.
Link relays enable the exchange of ON/OFF information between a master station and local station of a MELSECNET data link system, or between local stations.

(b) To allow use in a MELSECNET data link system, link parameters must be set at the master station.
Link relays that are not set in the link parameters can be used as substitutes for Internal relays.

REMARKS

1) For details regarding the network parameters, refer to the For QnA/Q4AR MELSECNET/10 Network System Reference Manual.

2) For details regarding link parameters, refer to the MELSECNET & MELSECNET/B Data Link System Reference Manual.

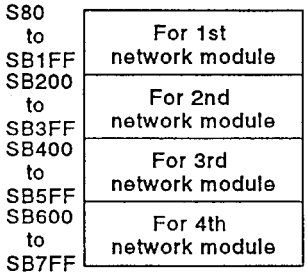
4.2.8 Special link relays (SB)

(1) Definition

(a) Special link relays (SB) are used to transmit ON/OFF data between the MELSECNET/10 network module and the user program.

(b) Because special link relays are switched ON and OFF in accordance with various problems which may occur during a data link, they serve as a tool for identifying data link problems.

(2) Number of special link relay points
There are 2048 special link relay points (SB0 to SB7FF) for each MELSECNET/10 network module. As shown below, the default "number of points" setting for QnACPU special link relays is 512 points per module.



REMARK

1) For details regarding special link relays used at the QnACPU, refer to the QnACPU Programming Manual (Common Instructions).

4.2.9 Step relays (S)

A step relay is an SFC program device. For details regarding procedures for using step relays, refer to the QnACPU Programming Manual (SFC).

POINT

Because the step relay is an SFC program dedicated device, it cannot be used as an internal relay in the sequence program.
If used in this manner, and SFC error will occur, and system operation will be stopped (system down).

4.2.10 Timers (T)

QnACPU timers are of a forward timer type, with the time measurement beginning when the coil switches ON, and ending (time out) when the present value matches the setting value. The contact is switched ON when a "time out" occurs. There are 3 timer types: low-speed timers, high-speed timers, and retentive timers.

POINT

When an OUT T instruction is executed, the coil of the timer is turned ON/OFF, its present value is updated, and its contact is turned ON/OFF. At END processing, the present value of the timer is not updated and its contact is not turned ON/OFF.

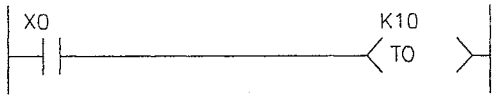
Low-speed timers

(1) Definition

(a) Low-speed timers are those that are only operative while the coil is ON.

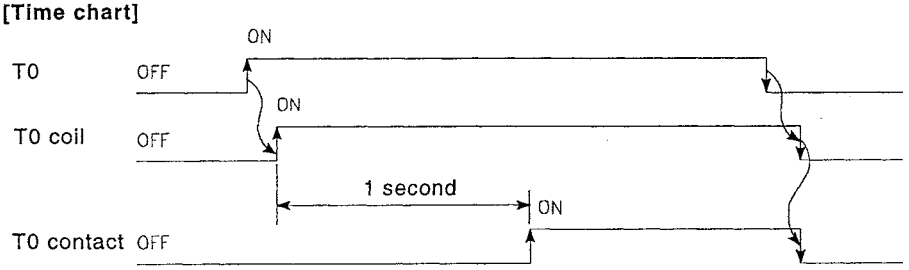
(b) The time measurement begins when the timer's coil switches ON, and the contact switches ON when a "time-out" occurs. When the timer's coil switches OFF, the present value becomes "0", and the contact switches OFF.

[Ladder example]



When X0 switches ON, the T0 coil switches ON, and the contact switches ON 1 second later. (The low-speed timer measures time in 100 ms units)

[Time chart]



(2) Measurement units

(a) The default time measurement units setting for low-speed timers is 100 ms.

(b) The time measurement units setting can be designated in 10 ms units within a 10ms to 1000 ms range.
This setting is designated in the "PC system settings" parameters.

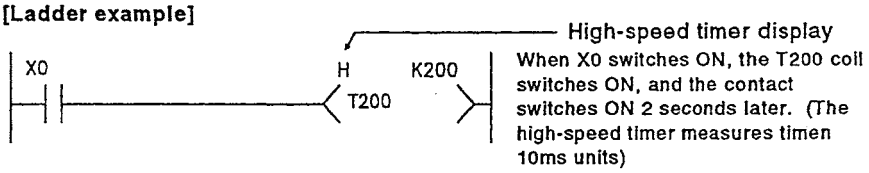
High-speed timers

(1) Definition

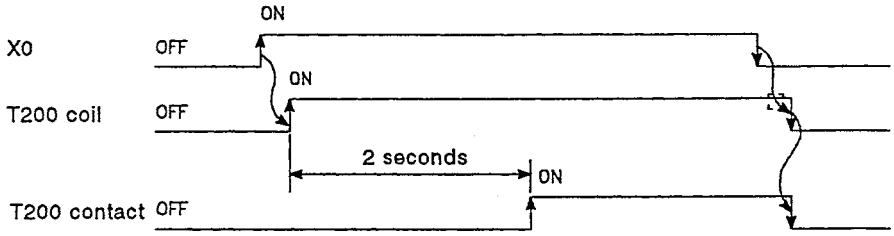
(a) High-speed timers are timers which are only operative while the coil is ON.

(b) The time measurement begins when the timer's coil switches ON, and the contact switches ON when a "time-out" occurs. When the timer's coil switches OFF, the present value becomes "0", and the contact switches OFF.

[Ladder example]



[Time chart]



(2) Measurement units

(a) The default time measurement units setting for high-speed timers is 10 ms.

(b) The time measurement units setting can be designated in 1ms units within a 1 ms to 100 ms range.
This setting is designated in the PC system settings parameters. *

Retentive timers

(1) Definition

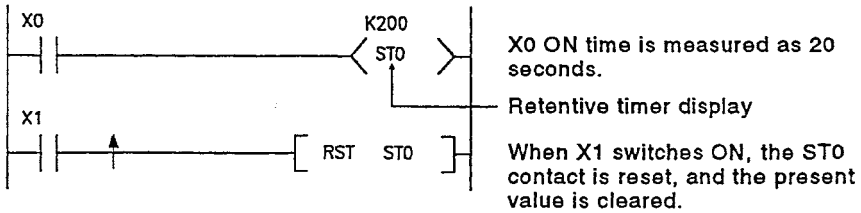
(a) Retentive timers measure the "coil ON" time.

(b) The measurement begins when the timer coil switches ON, and the contact switches ON when a time-out (coil OFF) occurs. Even when the timer coil is OFF, the present value and the contact ON/OFF status are saved. When the coil is switched ON again, the time measurement resumes from the present value which was saved.

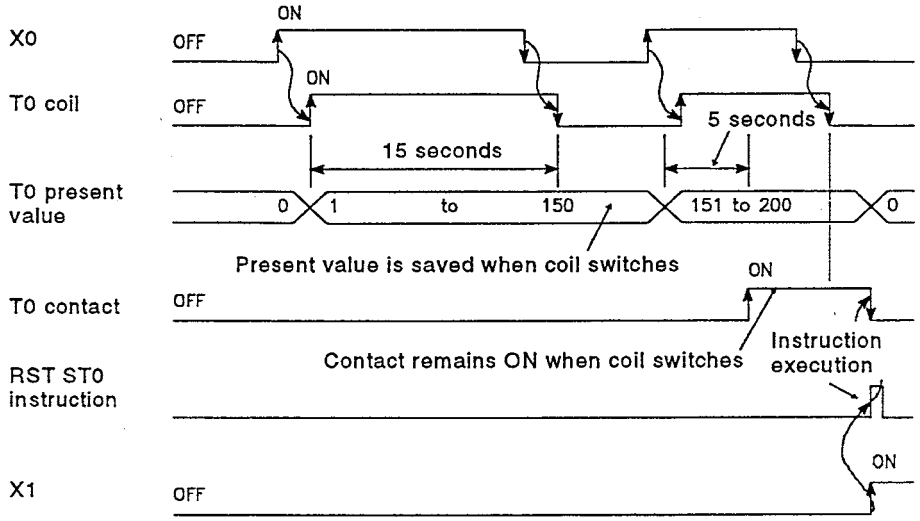
(c) There are 2 retentive timer types: low-speed retentive timer, and high-speed retentive timer.

(d) The RST T□ instruction is used to clear (reset) the present value and switch the contact OFF.

[Ladder example]



[Time chart]



(2) Measurement units

(a) The measurement units settings for retentive timers are the same those for low-speed timers and high-speed timers.
- Low-speed retentive timer : Some as low-speed timer
- High-speed retentive timer: Same as high-speed timer

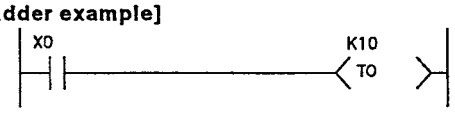
REMARKS

1) *: In order to use retentive timers, a retentive timer "number of points used" setting must be designated in the PC device settings parameters.

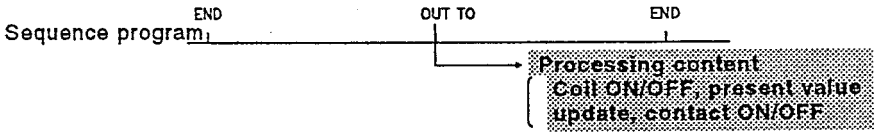
Timer Processing & accuracy

(a) When an OUT T□ instruction is executed, the following processing occurs: timer coil ON/OFF, present value update & contact ON/OFF processing. Timer present value update and contact ON/OFF processing do not occur at END processing.

[Ladder example]

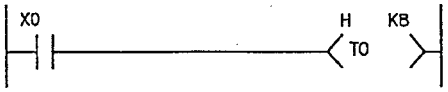


[Processing at OUT T0 instruction]

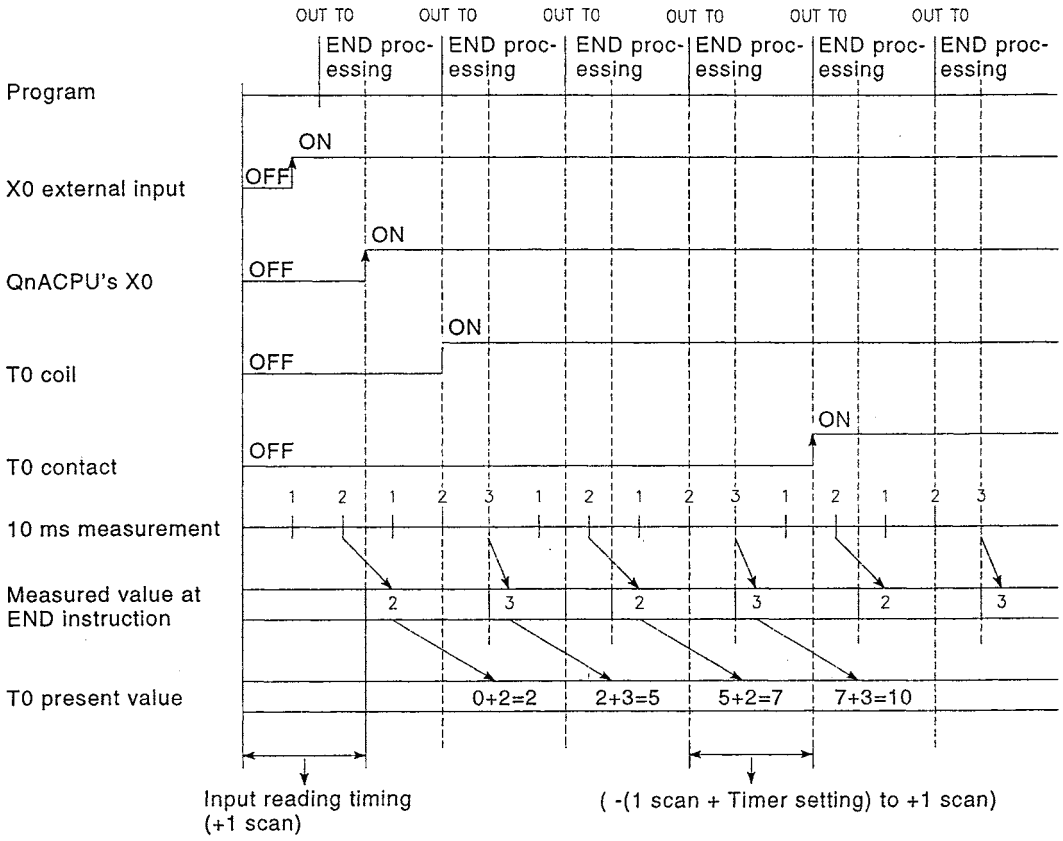


(b) When the OUT T□ instruction is executed, the present value is added to the scan time measured at the END instruction.
If the timer coil is OFF when the OUT T□ instruction is executed, the present value is not updated.

[Ladder example]



[Present value update timing]

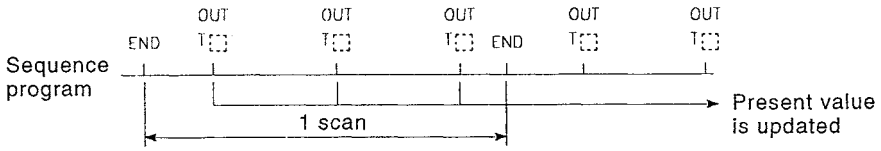


(c) The timer response accuracy from the point when input (X) reading occurs, until the point when the output occurs is +2 scans.

Precautions when using timers

The following are a few precautions regarding timer use.

(a) A given timer cannot be designated (by OUT T□) more than once in a single scan. If it is, the timer's present value will be updated at each OUT T□ instruction, resulting in a meaningless measurement.



(b) When the timer is not executed at each scan
An OUT T1 instruction cannot be given by a CJ instruction during a timer coil (Example: T1) ON.
While the OUT T□ instruction is being given, the current value is not updated.
In addition, when a timer exists in the subroutine program, pay attention to the following. During the timer coil (Example: T1) ON, be sure to execute a subroutine call which includes an OUT T1 instruction only one time at each scan.
In the case that the subroutine call is not executed at each scan, the current value is not updated.

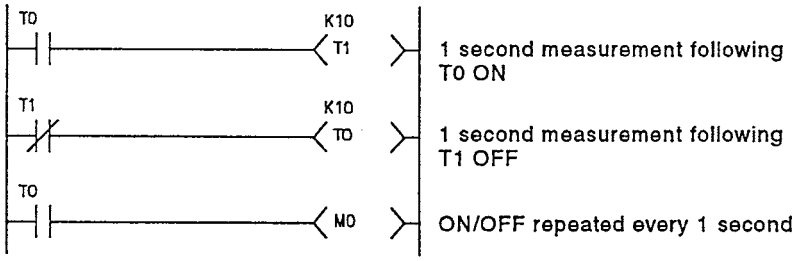
(c) Timers cannot be used in interrupt programs.

(d) If the timer set value is "0", the contact goes ON when the OUT T□ instruction is executed.

(e) If the setting value changes to a value which is higher than the present value following a timer "time-out", the "time-out" status will remain in effect, and timer operation will not occur.

(f) If a timer is used at a low-speed execution program, the present value will be added to the low-speed scan time when the OUT T□ instruction is executed.

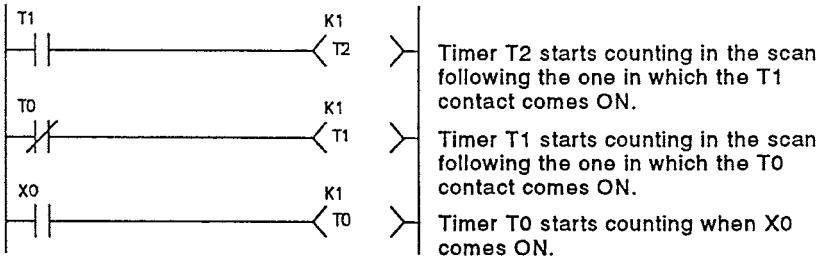
(g) If two timers are used, the ON/OFF ladders should be created as shown below.



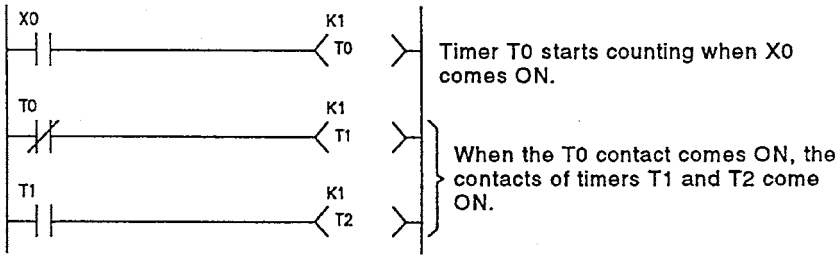
(h) When creating a program in which a timer contact is used as the trigger for counting by a different timer, write the program starting from the counter that counts later.
In the cases below, if programming is done in the same order in which the timers count, all the timers will be ON in the same scan.
- When a high-speed timer is used with a set value shorter than the scan time
- When a low-speed timer is used with a set value of "1"

Example

- When timers T0 to T2 are programmed in the reverse of the order in which they start counting:



- When timers T0 to T2 are programmed in the order in which they start counting.



4.2.11 Counters (C)

QnACPU counters are "up counter" types, with the contact being switched ON when the count value equals the setting value (count-out condition). There are two counter types: counters which count the number of input condition start-ups (leading edges) in sequence programs, and counters which count the number of interrupt factor occurrences.

POINT

When an OUT C□ instruction is executed, the following counter processing occurs: coil ON/OFF, present value update (count value + 1), and contact ON.
Counter present value update and contact ON processing do not occur at END processing.

Counter(input condition leading edge counter)

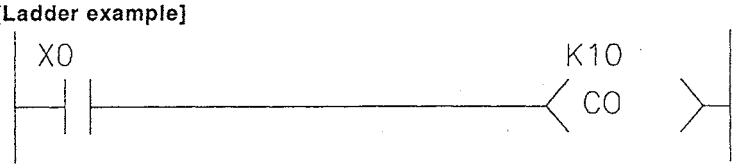
(1) Definition

A counter is a device which counts the number of input condition leading edges in sequence programs.

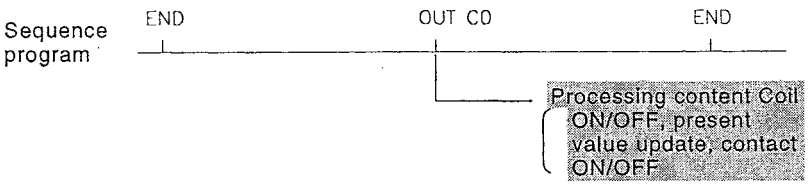
(2) Count processing

(a) When and OUT C□ instruction is executed, the follwing counter processing occurs: coil ON/OFF, present value update (count value + 1), and contact ON/OFF.
Counter present value update and contact ON/OFF processing do not occur at END processing.

[Ladder example]

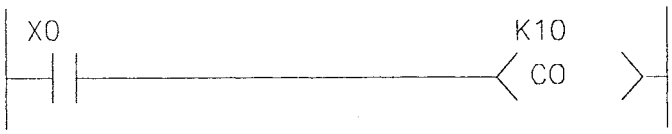


[Processing at OUT C0 instruction (X0: OFF-ON)]

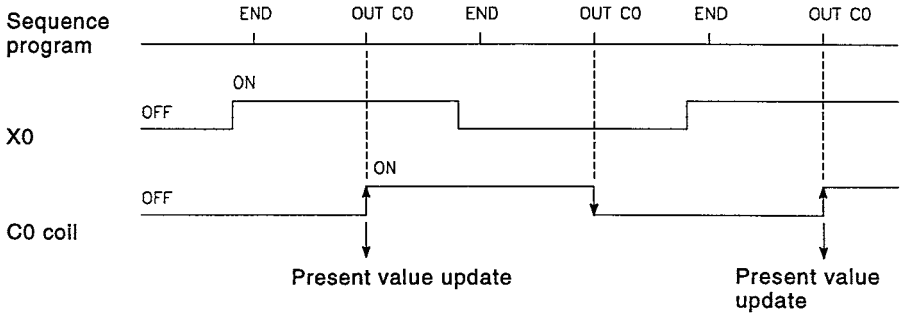


(b) The present value update (count value + 1) occurs at the leading edge (OFF → ON) of the OUT C□ instruction.
The present value is not updated in the following OUT C□ instruction statuses: OFF, ON → ON, ON → OFF

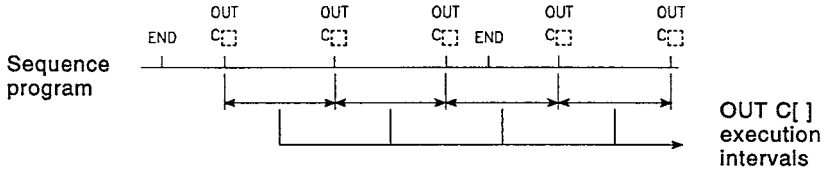
[Ladder example]



[Present value update timing]



(c) Multiple counters can be used within a single scan to achieve the maximum counting speed.
In such cases, the direct access input (DX□) method should be used for the counter input signals. *1

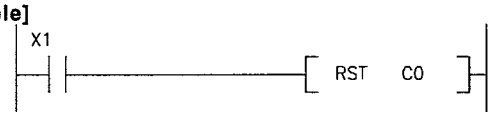


(3) Resetting the counter

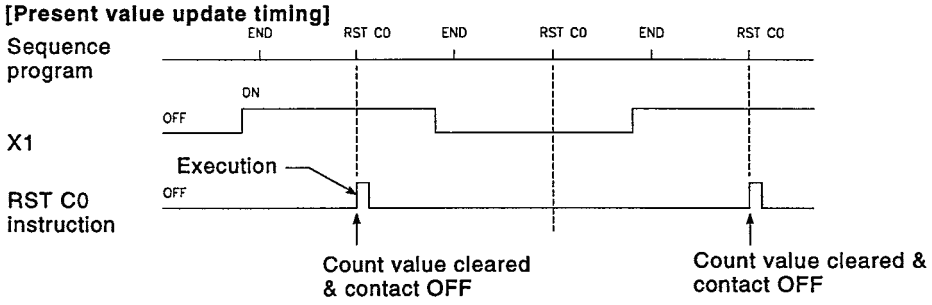
(a) Counter present values are not cleared even if the OUT C instruction switches OFF. Use the RST C□ instruction to clear the counter's present value **and coil and the contact are switched** and switch the contact OFF.

(b) The count value is cleared **and coil and the contact are switched** and the contact is switched OFF at the point when the RST C□ instruction is executed.

[Ladder example]



[Present value update timing]

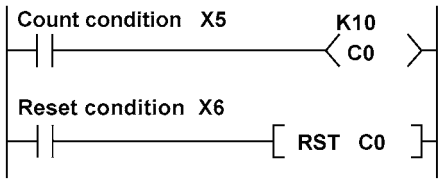


(4) Precautions for resetting the counter

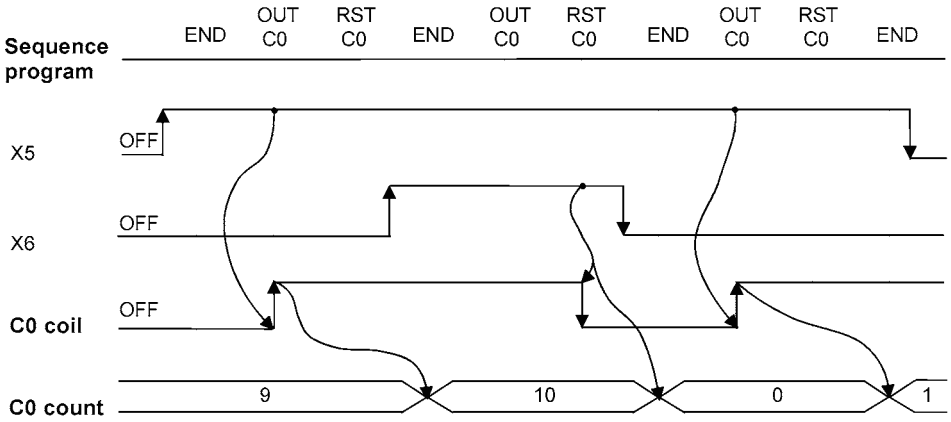
Do not reset the counter if the counting condition is satisfied.
If the counter is reset, an erroneous count may be added.

If resetting condition X6 is turned on in the circuit example shown below to execute the RST C0 instruction while the count condition X5 is active, the C0 coil is turned off then on again, so that the rising edge of the C0 coil is detected and the current value of C0 is updated (to increase the count by one).

[Circuit example]

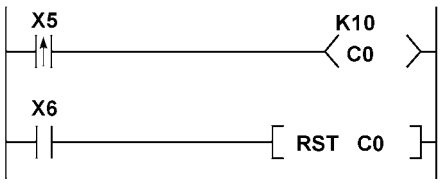


[Timing for updating current value]



To turn reset condition X6 on while the count condition X5 is active, use a differential contact to count at the rising edge of count condition X5.

[Circuit example]



(5) Maximum counting speed

The counter can count only when the input condition ON/OFF time is longer than the execution interval of the corresponding OUT C□ instruction.
The maximum counting speed is calculated by the following formula:

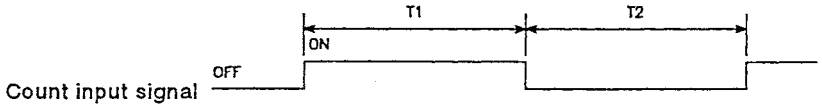
$$\text{Maximum counting speed (Cmax)} = \frac{n}{100} \times \frac{1}{t} \quad \text{[times/sec]}$$

n : Duty (%) *2
t : Execution interval or OUT C□ instruction

REMARKS

1) *1 : See Section 4.2.1 for details regarding direct access inputs.

2) *2 : The "duty" is the count input signal's ON-OFF time ratio expressed as a percentage value.



When T1 ≤ T2 : $n = \frac{T1}{T1+T2} \times 100\ (\%)$

When T1 ≤ T2 : $n = \frac{T1}{T1+T2} \times 100\ (\%)$

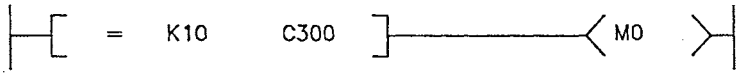
Interrupt counter

(1) Definition
Interrupt counters are devices which count the number of interrupt factor occurrences.

(2) Count processing

(a) The interrupt counter's present value is updated when an interruption occurs. If is not necessary to create a program which includes an interrupt counter function.

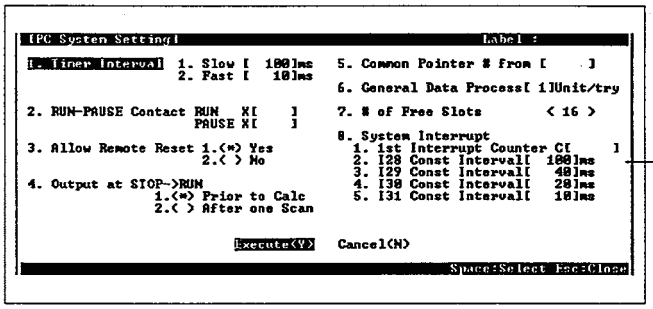
(b) Interrupt counter operation requires more than the simple designation of a setting value.
To use the interrupt counter for control purposes, comparison instructions (=, <=, etc.) must also be used to enable comparisons with the setting value, with an internal relay (M), etc., being switched ON or OFF according to the comparison result. The figure below shows a sample program in which M0 is switched ON after 10 interrupt inputs occur. (In this example, "C300" is the interrupt counter No. corresponding to 10.)



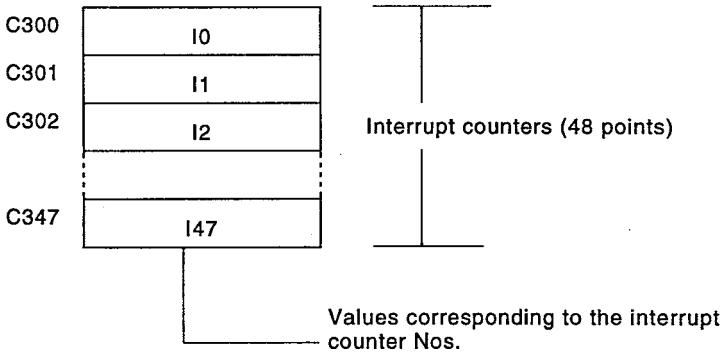
(3) Setting the interrupt counter

(a) In order to use interrupt counters, at first interrupt counter No. setting must be designated in the PC system settings parameters. 48 points are then allocated for interrupt counters, beginning from the "first counter No." which is designated.

["PC system settings" screen]



If C300 is designated as the first interrupt counter No., numbers C300-C347 will be allocated for interrupt counters.



(b) In order to use an interrupt counter, an "interruption permitted" status must be established at the main routine program.

(4) Precautions

(a) One interrupt pointer is insufficient to execute interrupt counter and interrupt program operation. Moreover, an interrupt program cannot be executed by an interrupt pointer designated for an interrupt counter.

(b) If the processing items shown below are in progress when an interruption occurs, the counting operation will be delayed until processing of these items is completed.
Even if the same interruption occurs again while processing of these items is in process, only one interruption will be counted.
- During execution of sequence program instructions
- During general data processing in END processing
- During interrupt program execution

(c) The maximum counting speed of the interrupt timer is determined by the longest processing time of the items shown below.
- Instruction with the longest processing time among the instructions used in the program
- General data processing time at END processing ...Max. 2 ms
- Interrupt program processing time

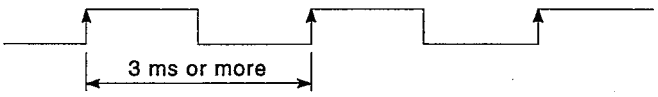
$$\text{Maximum counting speed} = \frac{1}{[\text{Longest processing time of the above 3 times}] + [500\ \mu\text{sec} \times \text{number of interrupt counter points}]}\ [\text{PPS}]$$

[Example]
- Longest instruction processing time ... 0.3 ms
- Interrupt program ..... None
- Number of interrupt counter points 2

Since here the END processing time of 2 ms (0.002 sec) is the highest value:

$$\text{Max. counting speed} = \frac{1}{0.002 + 0.0005 \times 2} \approx 333\ [\text{PPS}]$$

Based on this maximum counting speed, the input pulse signal must be as follows:



(d) The use of too many interrupt counters will increase the sequence program processing time, and may cause a "WDT ERROR". If this occurs, either reduce the number of interrupt counters, or reduce the counting speed for the input pulse signal.

(e) The interrupt counter's count value can be reset by using the RST C□ instruction in the sequence program prior to the FEND instruction.

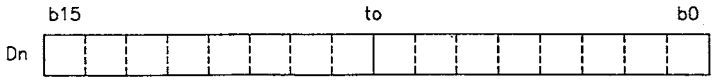
(f) The interrupt counter's count value can be read out by using the sequence program MOV instruction.

4.2.12 Data registers (D)

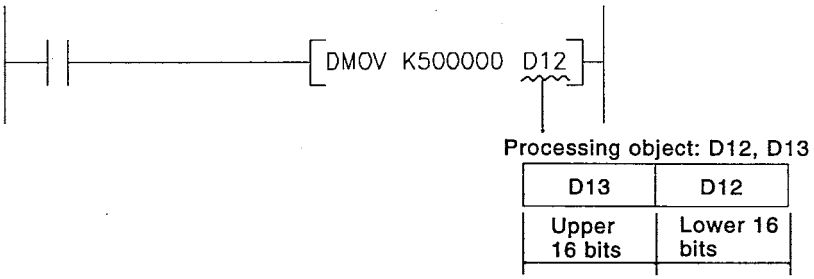
(1) Definition

(a) Data registers are memory devices which store numeric data (-32768 to 32767, or 0000_H to $FFFF_H$) in the QnACPU.

(b) Data registers consist of 16 bits per point, with reading and writing executed in 16-bit units.



(c) If the data registers are used for 32-bit instructions, the data will be stored in registers Dn and Dn + 1. The lower 16 bits of data are stored at the data register No. (Dn) designated in the sequence program, and the higher 16 bits of data are stored in the designated register No. + 1 (Dn + 1).
For example, if register D12 is designated in the DMOV instruction, the lower 16 bits are stored in D12, and the upper 16 bits are stored in D13.

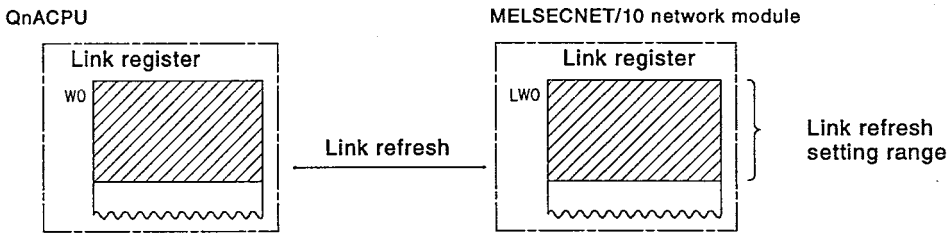


(d) Data stored by the sequence program is maintained until another data save operation occurs.

4.2.13 Link registers (W)

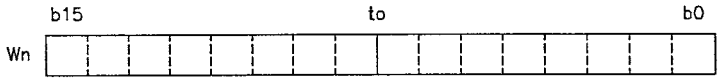
(1) Definition

(a) A link register is the QnACPU memory used to refresh the QnACPU with data from the MELSECNET/10 network module and MELSECNET/10 network module link registers (LW).
Link registers are used to store numeric data (-32768 to 32767, or 0000_H to $FFFF_H$) at the QnACPU.

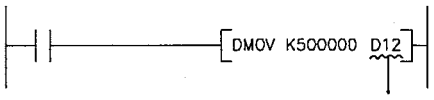


When used outside the MELSECNET data link system, MELSECNET/10 network system's range, link registers can serve as data registers.

(b) Link registers consist of 16 bits per point, with reading and writing executed in 16-bit units.



(c) If the link registers are used for 32-bit instructions, the data is stored in registers Wn and Wn + 1. The lower 16 bits of data are stored in the link register No. (Wn) designated in the sequence program, and the higher 16 bits of data are stored in the designated register No. + 1 (Wn + 1).
For example, if register W12 is designated at the DMOV instruction, the lower 16 bits are stored in W12, and the upper 16 bits are stored in W13.



Processing object: W12, W13

W13	W12
Higher 16 bits	Lower 16 bits

(d) Data stored by the sequence program is maintained until another data save operation occurs.

(2) Using link registers in a MELSECNET/10 network system

(a) If link registers are used in a MELSECNET/10 network system, the host station's numeric data can be read to another station for use there.
Use of link registers in the MELSECNET/10 network system permits the transfer of numeric data between the control station and a normal station, and between normal stations.

(b) In order to use link registers in the MELSECNET/10 network system, network parameter settings must be made at the control station.
Link registers not set in the network parameter settings can be used as data registers.

(3) Using link registers in MELSECNET data link systems

(a) If link registers are used at the MELSECNET data link systems, the host station's numeric data can be read to another station for use there.
Use of link registers in a MELSECNET data link systems permits the transfer of numeric data between the master station and a local station, and between local stations.

(b) In order to use link registers in the MELSECNET data link system, network parameter settings must be made at the control station.
Link registers not set in the network parameter settings can be used as data registers.

REMARKS

(1) For details regarding network parameters, refer to the For QnA/Q4AR MELSECNET/10 network System Reference Manual.

(2) For details regarding link parameters, refer to the MELSECNET & MELSECNET/B Data Link System Reference Manual.

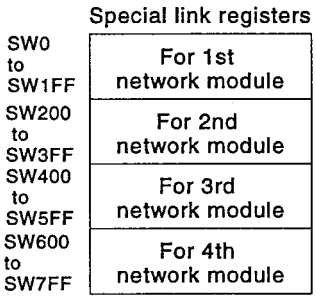
4.2.14 Special link registers (SW)

(1) Definition

(a) Special link registers are used to transfer data between the MELSECNET/10 network module and the user program.

(b) Because the data link information is stored as numeric data, the special link registers serve as a tool for identifying the locations and causes of faults.

(2) Number of special link register points
There are 2048 special link register points (SW0 to SW7FF) used by MELSECNET/10 network modules. As shown below, the default "number of points" setting for QnACPU special link registers is 512 points per module.



REMARK

1) For details regarding special link registers used in the QnACPU, refer to the QnACPU Programming Manual (Common Instructions).

4.3 Internal System Devices

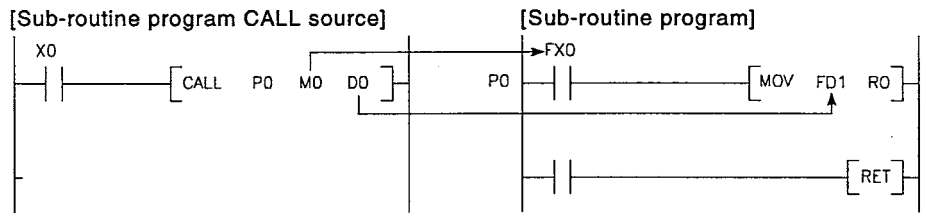
Internal system devices are devices used for system operations. The allocations and sizes of internal system devices are fixed, and cannot be changed by the user.

4.3.1 Function devices (FX, FY, FD)

(1) Definition

(a) Function devices are devices used in sub-routine programs with arguments to permit data transfers between the sub-routine program with argument, and the CALL source for that sub-routine.

Example:
If FX0 and FD1 are used at the sub-routine program, and if M0 and D0 are designated by the sub-routine CALL instruction, the M0 ON/OFF data is transferred toFX0, and the D0 data is transferred to FD1.



(b) Because the function devices used for each sub-routine program CALL source can be set, the same sub-routine program can be used without regard to other sub-routine CALL sources.

(2) Types of function devices
There are 3 function device types: function input devices (FX), function output devices (FY), and function register devices (FD).

(a) Function input devices (FX)

- These devices are used to designate inputs of ON/OFF data to a sub-routine program.
- In the sub-routine program, these devices are used for reading and processing bit data designated by sub-routine with argument CALL instruction.
- All the QnACPU bit data designation devices can be used.

(b) Function output devices (FY)

- These devices are used to designate outputs of sub-routine program operation results (ON/OFF data) to the sub-routine program CALL source.
- At sub-routine programs with arguments, the operation results are stored at the designated device.
- All bit data designation devices except QnACPU inputs (X, DX) can be used.

REMARK

1) Function devices can only be monitored during execution of a subroutine program with an argument.
When monitoring function devices, designate a step number in the subroutine with argument for which the function device is used.

(c) Function registers

- These devices are used to designate data transfers between the sub-routine CALL source and the sub-routine program.
- The function register input/output condition is automatically determined by the QnACPU. If the sub-routine program data is the source data, the data is designated as sub-routine input data. If the sub-routine program data is the destination data, the data is designated as sub-routine output data.
- 1 point occupies 4 words.
- The QnACPU word data designation device can be used.

REMARK

1) For details on the use of function devices, see QnACPU Programming Manual (Common Instructions).

4.3.2 Special relays (SM)

(1) Definition

(a) Special relays are QnACPU internal relays with fixed applications. They are used for ON/OFF data communications between the QnACPU system and the user program.

(2) Special relay classifications
Special relays are classified according to their applications, as shown below.

(a) For fault diagnosis : SM0-SM199

(b) System information: SM200-SM399

(c) System clock/system counter : SM400-SM499

(d) Scan information : SM500-SM599

(e) Memory card information : SM600-SM699

(f) Instruction related : SM700-SM799

(g) For debugging : SM800-SM899

(h) Latch area : SM900-SM999

(i) For ACPU : SM1000-SM1299

REMARK

1) For details regarding special relays which can be used by the QnACPU, refer to the QnACPU Programming Manual (Common Instructions).

4.3.3 Special registers (SD)

(1) Definition

(a) Special registers are QnACPU internal registers with fixed applications. They are used for ON/OFF data communications between the QnACPU system and the user program.

(2) Special register classifications
Special registers are classified according to their applications, as shown below.

(a) For fault diagnosis : SD0 to SD199

(b) System information: SD200 to SD399

(c) System clock/system counter : SD400 to SD499

(d) Scan information : SD500 to SD599

(e) Memory card information : SD600 to SD699

(f) Instruction related : SD700 to SD799

(g) For debugging : SD800 to SD899

(h) Latch area : SD900 to SD999

(i) For ACPU : SD1000 to SD1299

REMARK

1) For details regarding special relays which can be used by the QnACPU, refer to the QnACPU Programming Manual (Common Instructions).

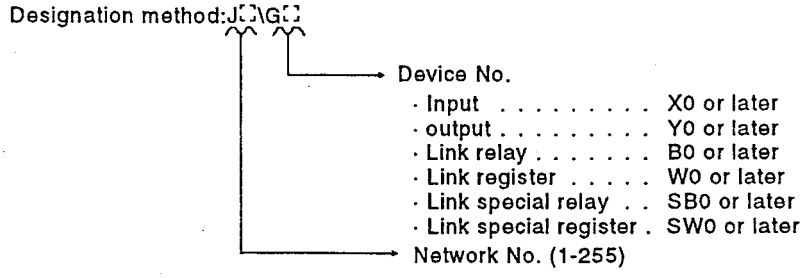
4.4 Link Direct Devices (J□\□)

(1) Definition

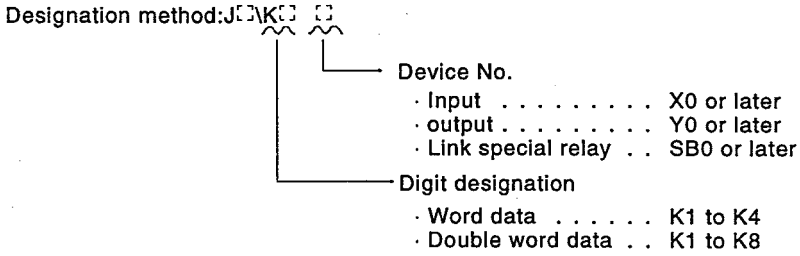
(a) At END processing, a data refresh (data transfer) operation occurs between the QnACPU and the MELSECNET/10 network system modules. Link direct devices are used at that time to directly access the link devices in the MELSECNET/10 network modules.

(b) Designation method

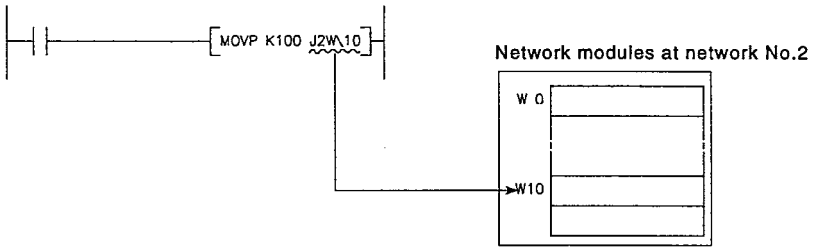
1) Link direct devices are designated by network No. and device No.



2) When inputs, outputs, and link special relays are used as word data or double word data, digit designation is necessary.



Example : For link register 10 (W10) of network No.2, the designation would be "J2\W10"



3) For a bit device (X, Y, B, SB), digit designation is necessary.

Designation example :J1/1X0, J10/K4B0

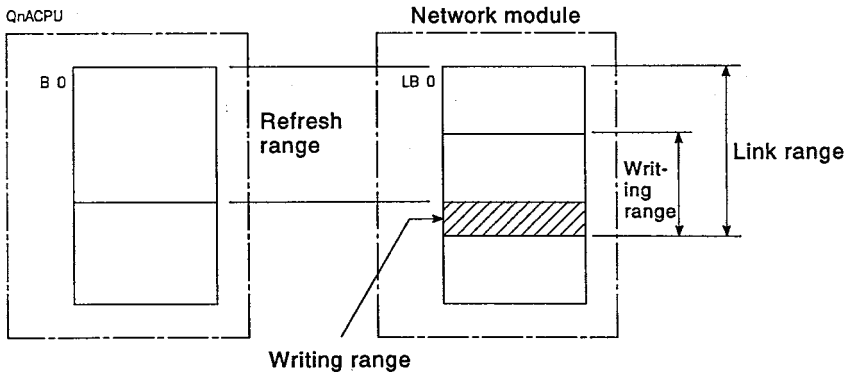
(2) Designation range

Link direct device designations are possible for all athe link devices in network modules.

(Device outside the range specified by the network refresh parameters can also be designated.)

(a) Writing

1) Writing is executed within that part of the link device range set as the send range in the common parameters of the network parameters that is outside the range specified as the "refresh range" in the network refresh parameters.
However, when an output outside the refresh range is turned ON, even if the QnACPU is set to the STOP status it will not be refreshed and therefore will not go OFF.

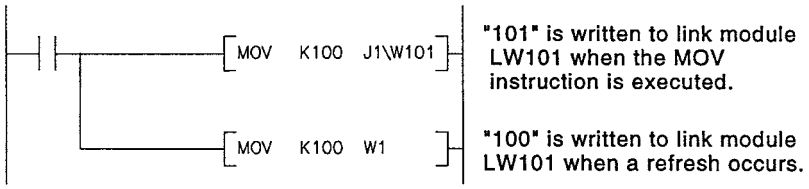


2) Although writing is also possible in the "refresh range" portion of the link device range (specified by the refresh parameters), the link module's link device data will be rewritten when a refresh operation occurs. Therefore, when writing by link direct device, the same data should also be written to the QnACPU related devices (designated by the refresh parameters).

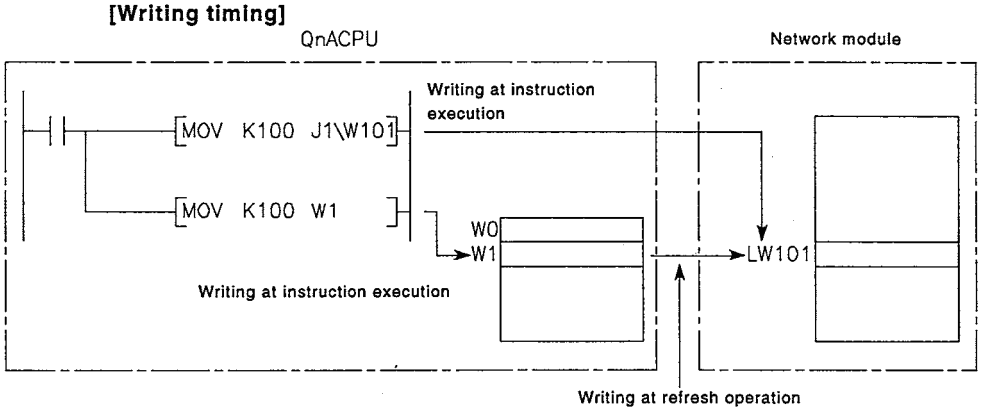
[Refresh parameter settings]

- Network No.: 1
- QnACPU (W0 to W3F) ↔ network modules (LW100 to LW13F)

[Sequence program]



[Writing timing]



3) When data is written to another station's writing range using a link direct device, the data which is received from that station will replace the written data.

(b) Reading
Reading by link direct device is possible in the entire link device range of network modules.

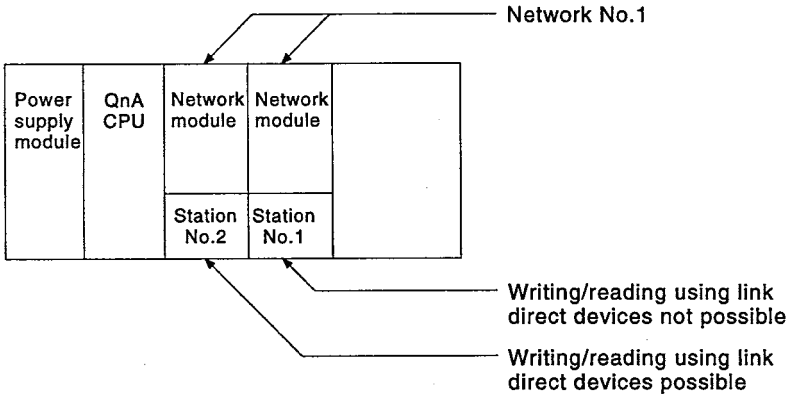
(3) Differences between "link direct devices" and "link refresh
"The differences between "link direct devices" and "link refresh" are shown in Table 4.4 below.

Table 4.4 Differences between "Link direct Devices" and "Link Refresh"

Item		Link Direct Device	Link Refresh
Program notation method	Link relay	J□/K4B0 or later	B0 or later
	Link register	J□W0 or later	W0 or later
	Link special relay	J□/K4SB0 or later	SB0 or later
	Link special register	J□SW0 or later	SW0 or later
Number of steps		2 steps	1 step
Network module access range		All network module link devices	Parameter designated range
Access data guarantee range		Word units (16 bits)	

POINT

(1) Only one network module capable of writing/reading link direct devices can be used per network number.
If two or more network modules are installed at the same network number, the network module with the lowest first I/O number will be the one that handles writing/reading using link direct devices.
For example, if station No.1 and station No.2 network modules are installed in network No.1 as shown in the figure below, the station No.2 network module will handle link direct device operations.



REMARKS

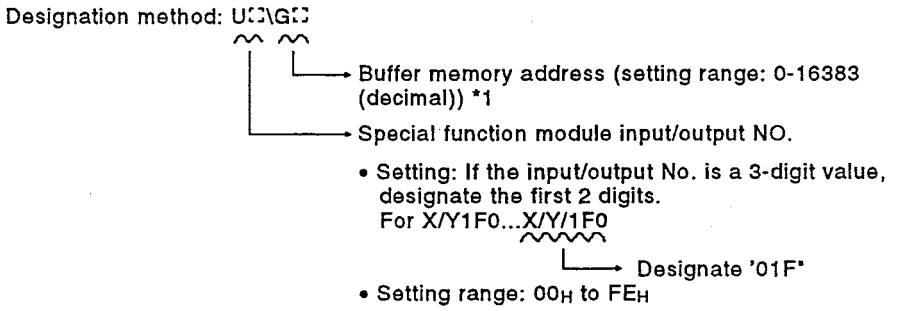
1) For details regarding the MELSECNET/10 network system, refer to the
   • For QnA/Q4AR MELSECNET/10 Network System Reference Manual.
2) For details regarding network parameters, common parameters, and network refresh parameters, refer to the following manuals:
   • Detailed information
   For QnA/Q4AR MELSECNET/10 Network System Reference Manual
   • Setting procedures
   SW□IVD-GPPQ Type GPP Function Software Package Operating Manual (Offline)

4.5 Special Function Module Devices (U□\G□)

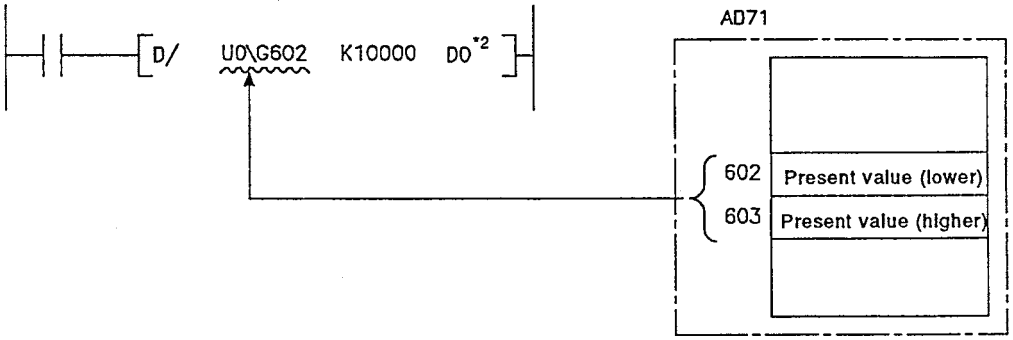
(1) Definition

(a) The special function module devices allow the QnACPU to directly access the buffer memories of special function modules which are installed at the main base unit and extension base unit. These devices cannot be used in this manner for special function modules installed at remote stations of a MELSECNET/10 network system or a MELSECNET (II, /B) data link system.

(b) Special function module devices are designated by the special function module input/output No., and the buffer memory address.



To convert the X-axis present value (buffer memories: 602, 603) (X-axis of AD71 positioning module installed at slot 0 of the base unit) to "mm" units (1/1000), and store it in D0 and D1, designate the following:



(2) Processing speed

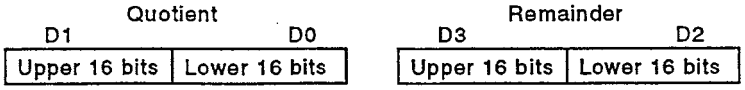
The processing speed for special function module devices is the total of the "FROM/TO instruction processing speed" and the "instruction processing speed".

If the same buffer memory of the same special function module is used two or more times in a sequence program, the processing speed can be increased by using the FROM instruction to read that buffer memory data to a QnACPU device.

REMARKS

1) *1: For details regarding buffer memory addresses and applications, refer to the manual for the special function module in question.

2) *2: The quotient and remainder are stored in D0 to D3.



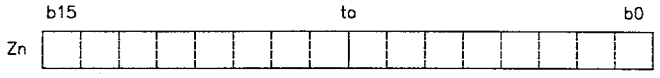
4.6 Index Registers (Z)

(1) Definition

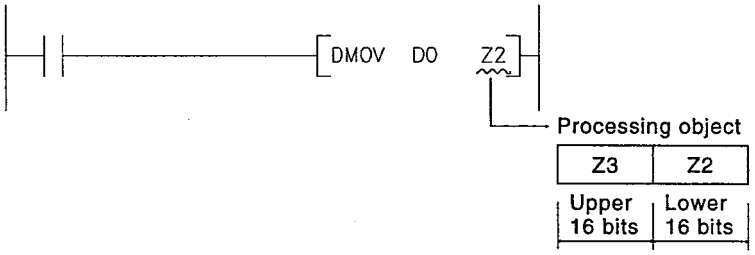
(a) Index devices are used in the sequence program for indirect setting (index qualification) designations.

(b) There are 16 index registers (Z0-Z15).

(c) Index registers consist of 16 bits per point, with reading and writing occurring in 16-bit units.



(d) If the index registers are used for 32-bit instructions, the data is stored in registers Zn and Zn +1. The lower 16 bits of data are stored in the index register No. (Zn) designated in the sequence program, and the upper 16 bits of data are stored in the designated index register No. + 1 (Zn + 1).
For example, if register Z2 is designated in the DMOV instruction, the lower 16 bits are stored in Z2, and the upper 16 bits are stored at Z3.



(2) Index register processing at program switching
When switching from a scan execution or low-speed execution program to another program type, the index register (Z0-Z15) data is saved (protected).
This data is reset when switching back to the scan execution or low-speed execution program occurs.

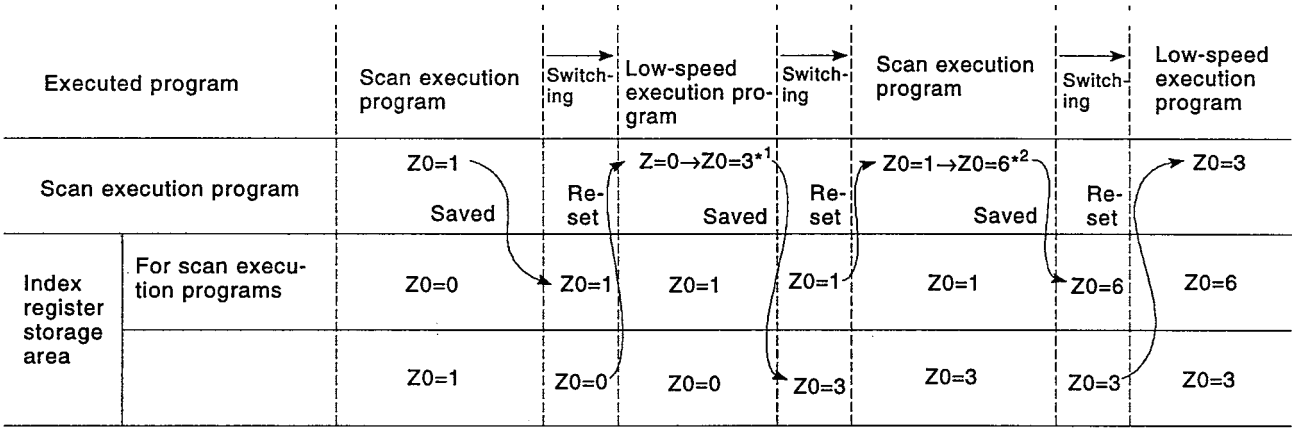
REMARK

1) For details regarding index qualifications using the index registers, refer to the QnACPU Programming Manual (Common Instructions).

(a) Switching between scan execution and low-speed execution programs

1) When switching from a scan execution program to a low-speed execution program occurs, the scan execution program's index register data is saved, and the low-speed execution program's index register data is reset.

2) When switching from a low-speed execution program to a scan execution program occurs, the low-speed execution program's index register data is saved, and the scan execution program's index register data is reset.



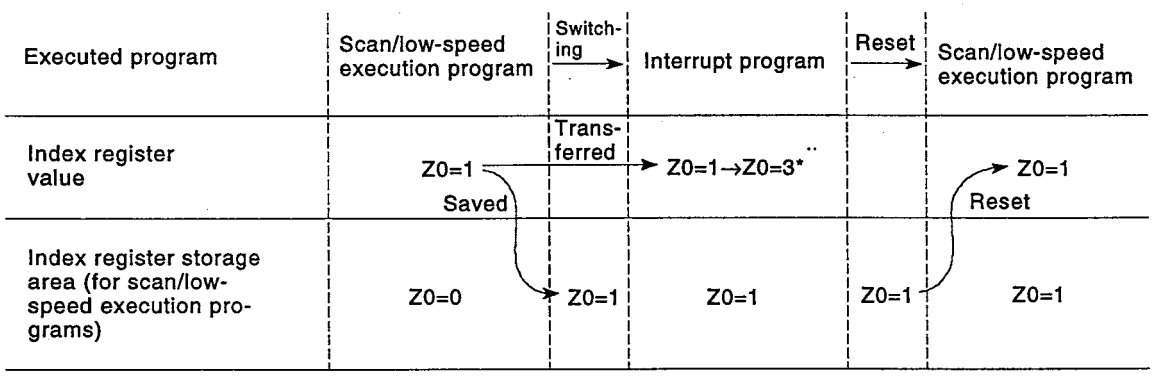
*1: For a low-speed execution program, Z0 is changed to 3.
*2: For scan execution program, Z0 is changed to 6.

Word devices should be used for exchanges of index register data between scan execution programs and low-speed execution programs.

(b) Switching between scan/low-speed execution programs and interrupt programs

1) When the scan/low-speed execution program is switched to the interrupt program, the scan/low-speed execution program's index register value is first saved, and is then transferred to the interrupt program.

2) When the interrupt program is swithed to the scan/low-speed execution program, the saved index register value is reset.



*: At interrupt programs, Z0 is changed to 3.

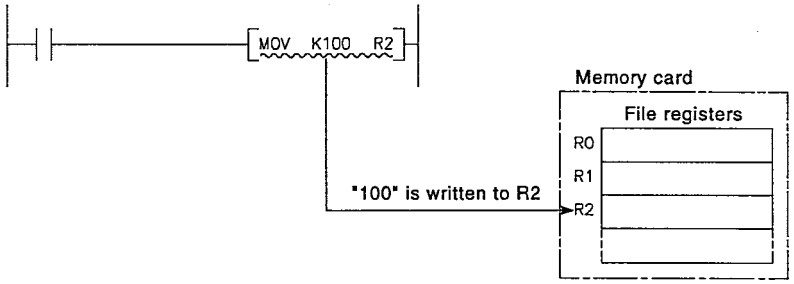
Word devices should be used to transfer index register data from an interrupt program to a scan execution program or low-speed execution program.

4.7 File Registers (R)

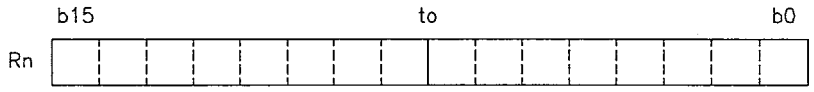
(1) Definition

(a) File registers are expansion devices for data registers.

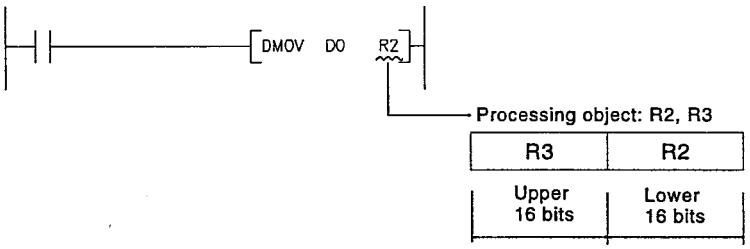
(b) File register data is stored in files in the QnACPU memory card. Therefore, the memory card is required when using file registers.



(c) File registers consist of 16 bits per point, with reading and writing occurring in 16-bit units.



(d) If the file registers are used for 32-bit instructions, the data will be stored in registers Rn and Rn + 1.
The lower 16 bits of data are stored in the file register No. (Rn) designated in the sequence program, and the upper 16 bits of data are stored in the designated file register No.+ 1 (Rn + 1).
For example, if register R2 is designated in the DMOV instruction, the lower 16 bits are stored in R2, and the upper 16 bits are stored in R3.



(2) File register capacity

Each file can be expanded to a maximum of 32 blocks (1018k words) in 1 block (32k words) units.
However, the permissible number of expansion blocks varies according to the capacity of the memory card being used, and the size of the sequence programs stored in the memory card.

REMARK

1) For details regarding the QnACPU memory cards , See Section 2.3.

(3) Differences in memory card access method by memory card type

The following three types of memory card are used to store a file register. Memory card access method differs depending on the memory type.

RAM

(a) Read/write using a program is allowed.

(b) PC read/write through the device setting is allowed.

(c) The file register data can be changed by any of the following methods.

1) Online test operation by GPP

2) Batch write command by the dedicated protocol of QC24 and QE71

3) Device write or random write command from a GOT900 series

E^2PROM

(a) Read using a program is allowed but write using a program is not allowed.

(b) PC read/write through the device setting is allowed.

(c) The file register data can be changed by any of the following methods.

1) Batch write command by the dedicated protocol of QC24 and QE71
(CPU must be in the STOP/PAUSE state.)
This is possible from the following software version of the CPU.

CPU Type	Software Version
QnA	L or later
Q2AS(H)	T or later
Q4AR	S or later

Flash ROM

(a) Read using a program is allowed but write using a program is not allowed.

(b) PC read/write through the device setting is not allowed.

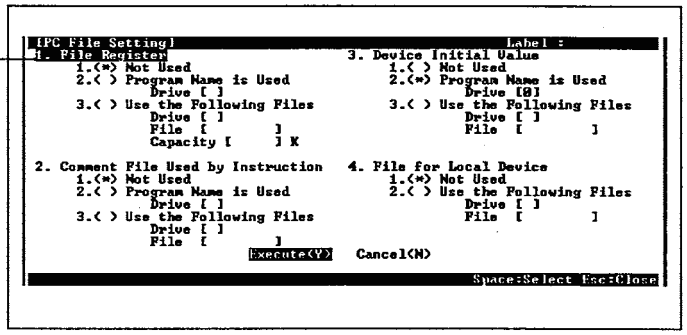
(c) File read/write through a reader/writer is allowed.
(For details, refer to the Operation Manual of the GPP.)

(4) Designating file registers for use

The memory card can hold a total of 124 file registers.
The memory card file registers which are to be used in the sequence program are determined by the PC file settings parameters.

[PC file settings screen]

Designate file registers to be used



(a) When "Not Use" is selected

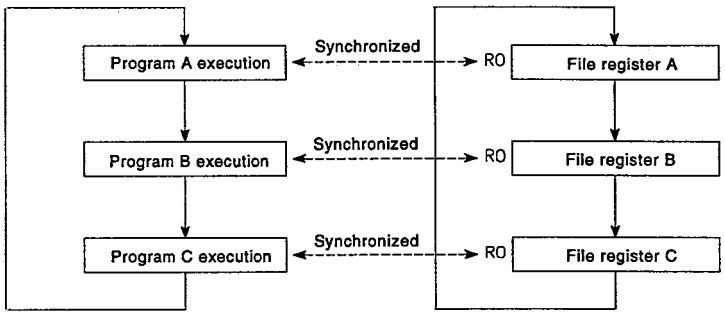
This item is selected in order to designate which file registers are to be used in the sequence program.
The QDRSET instruction is used to designate which file registers are to be used.

(b) When "Program Name is Used" is selected

This item is selected when the file registers having the same file name as the sequence program are to be used.
If the program is changed, the file registers are automatically changed to conform to the new program name.
There are also cases where it is convenient to use the file registers as local devices which can only be used with the program currently being executed.

Example:

When file registers (A-C) having the same name as the programs (A-C) are to be used, operation is as shown below.
- At program A execution...File register A is accessed
- At program B execution...File register B is accessed
- At program C execution...File register C is accessed



In the cases listed below, the file register of the file name that was executed at the end of one scan is accessed.
However, if there is no file register for the program executed at the end of the scan, file register access is not possible.

- Access from a peripheral device
- Access from another station in the network
- Access from a serial communication module

(c) Use the Following Files

This item is selected when a given file register is to be shared by all executed programs.
By designating the file register "Drive", "File", and "Capacity" setting, files for the parameter designated file registers will be created when a QnACPU RUN status is established.

(5) Registering file register files in the QnACPU

(a) If an item other than "Used the Following Files" is selected in the PC file setting screen, the file register files must be registered in the QnACPU. (If file registers having the same file name as the programs are to be used, they should be registered in the drive designated by the PC file settings parameters.)

1) When file register files are not registered in the QnACPU:

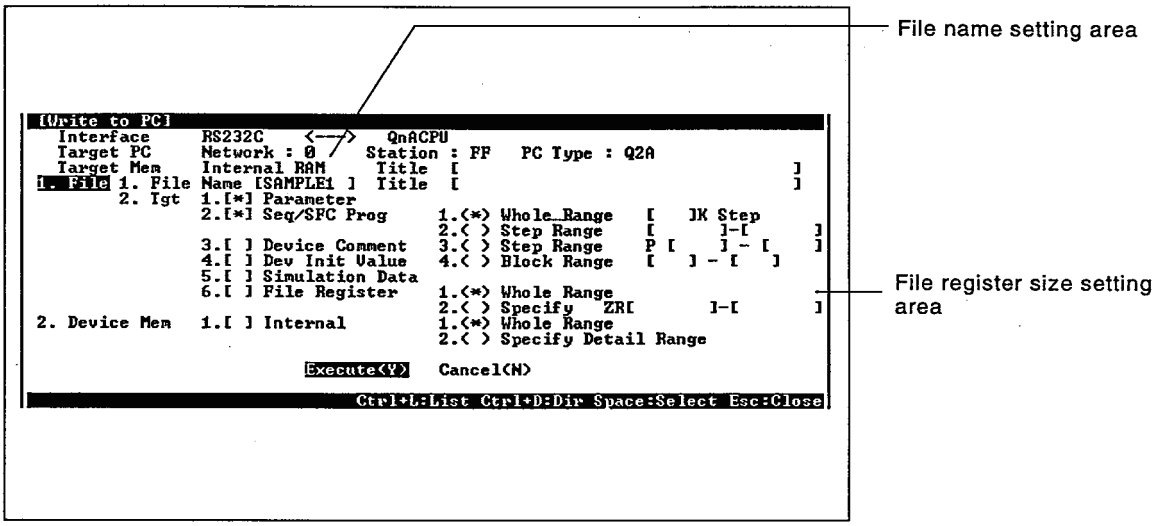
No error occurs even if reading/writing to file registers is executed. However, all read outs from file registers are stored as "$FFFF_H$".

2) Reading/writing to file registers outside the registered range:

No error occurs even if reading/writing occurs to these file registers. However, all read outs from these file registers are stored as "$FFFF_H$".

(b) To register file register files in the QnACPU, designate the file name are file register size settings at the peripheral device's "PC write screen", then write this data to the QnACPU.
The file register size is designated from ZR0, in 1k-point (1024 points) units. *1

[PC write screen]



(6) File register size check

(a) If file registers are used in the QnACPU, program so that writing/reading to the file registers occurs when the file register size is equal to or greater than the range actually required.

- A file register size check should be executed at step 0 of programs in which file registers are used.
- After switching to another file register file using the QDRSET instruction, execute a file size check.
- When using the RSET instruction to switch blocks, check that the switching destination block has a size of 1k points or more before executing the RSET instruction.

(File register size) > [32k points x (switching block No.) + 1k points]

(b) The available file register size can be checked in the file register capacity storage register (SD 647). *2
The file register size is stored in SD647 in 1k-point units.

The "less than 1k-points" surplus portion of a file register size is not stored.
In order to ensure an accurate "range of use" check, be sure to designate the file register setting in 1k-point (1024 points) units.

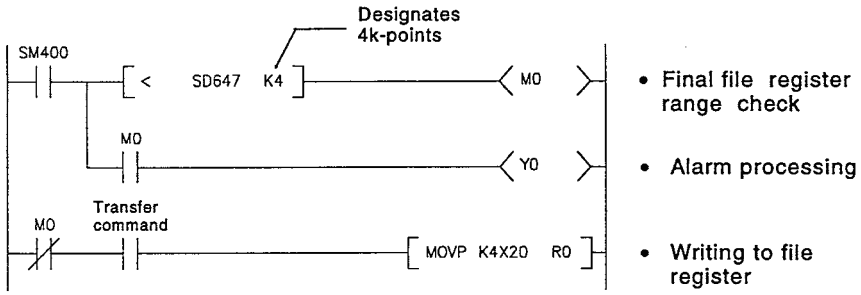
(c) Checking the file register size

1) The file register size used for each sequence program can be checked.

2) On the basis of the total file register size set in SD647 (in the sequence program), it is possible to determine if the file register size exceeds the number of points used.

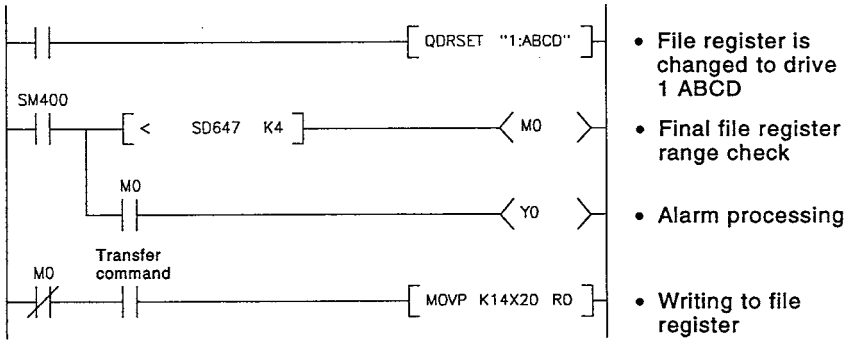
[Program example 1]

The file register "range of use" is checked at the beginning of each program.



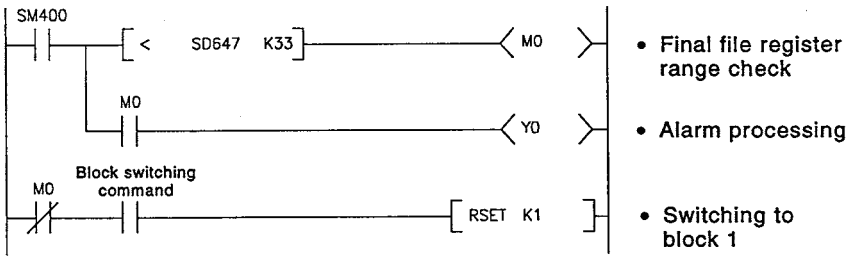
[Program example 2]

The file register "range of use" is checked after executing the QDRSET instruction.



[Program example 3]

For block switching.



REMARKS

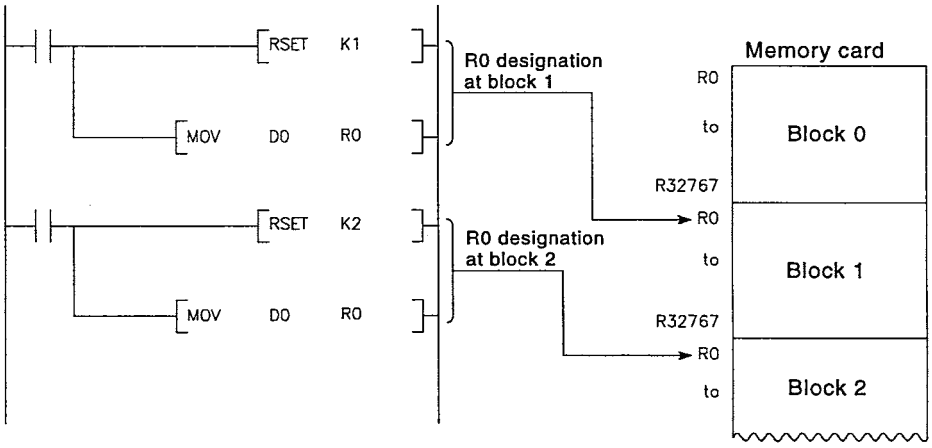
1) *1 : Even if the file register setting range is not designated from ZR0, the file creation range will still begin from ZR0, and end at the final file No.
For example, if the file register writing range is designated as ZR1000 to ZR2047, the file creation range will be ZR0 to ZR2047.
As the data from ZR0 to ZR999 data will be useless in this case, be sure to designate the range from ZR0.

2) *2 : When switching to another file register occurs, the size of the new file register file is stored at SD647.

(7) File register designation method

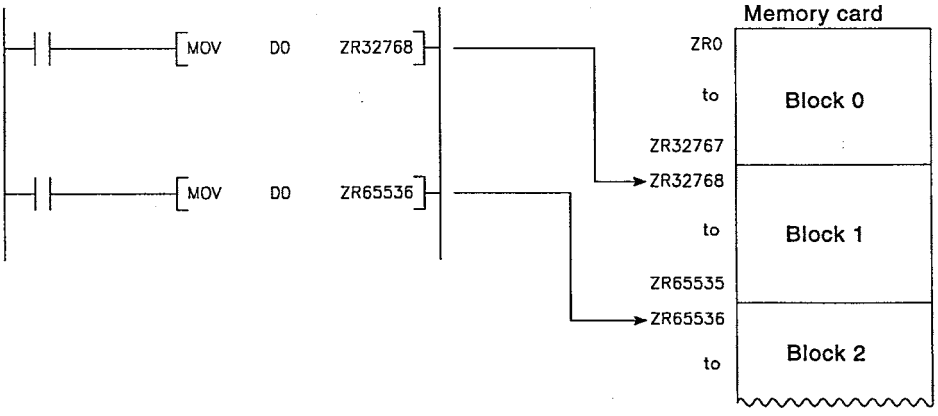
(a) Block switching format

The block switching format designates the number of file register points in 32k point (R0-R32767) units.
If multiple blocks are used, the RSET instruction is used to switch to another block No. for further file register settings.
Settings are designated in the R0 to R32767 range in each block.



(b) Serial number access format

This format is used for designating file register settings beyond 32k points without switching blocks Nos. Multiple blocks of file registers can be used as a continuous file register.



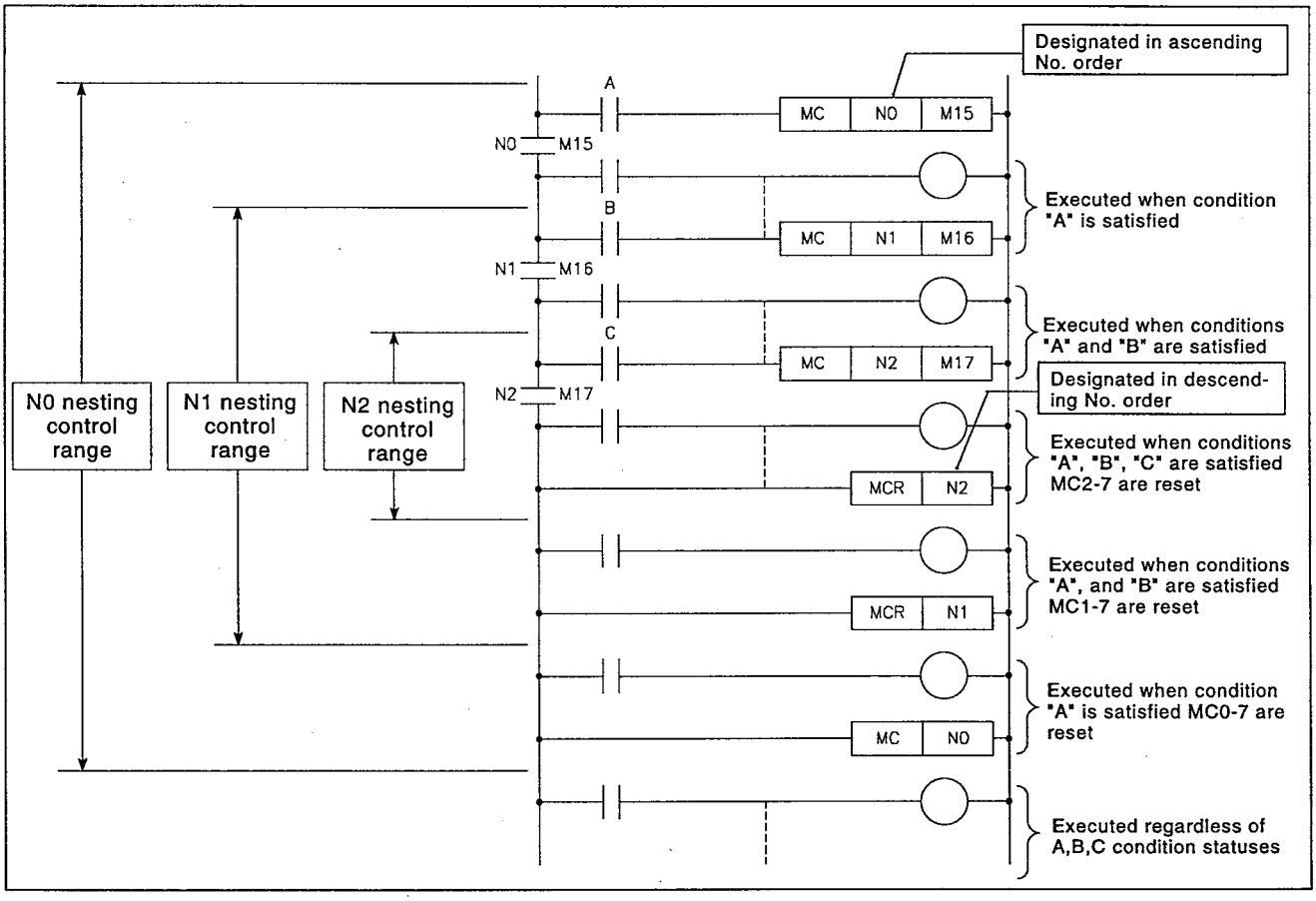
4.8 Nesting (N)

(1) Definition

Nesting devices are used with master control instructions.

(2) Designation method with master control

The master control instructions are used to open and close the ladders' common bus so that switching of ladders may be executed efficiently by the sequence program.
It is designated with the MC and MCR master control instructions.
For details on how to use master control, refer to the QnACPU Programming Manual (Common Instructions).



4.9 Pointers

(1) Definition

Pointers are devices used in branch instructions. A total of 4096 pointers can be used (total for all programs).

(2) Pointer applications

(a) Pointers are used in jump instructions (CJ, SCJ, JMP) to designate jump destinations and labels (jump destination beginning).

(b) Pointers are used in sub-routine CALL instructions (CALL, CALLP) to designate the CALL destination and label (sub-routine beginning).

(3) Pointer types

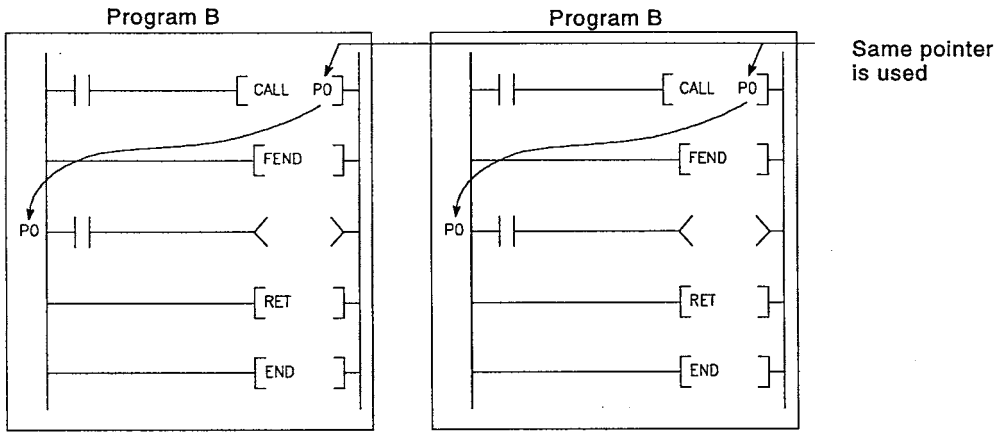
There are 2 pointer types: "local pointers" which are used independently in QnACPU programs, and "common pointers" which are used to call sub-routine programs from all programs executed in the QnACPU.
(See Section 4.9.1 for details regarding "local pointers", and section 4.9.2 for details regarding "common pointers".)

4.9.1 Local pointers

(1) Definition

(a) Local pointers are pointers which can be used independently in QnACPU program jump instructions and sub-routine call instructions. Local pointers cannot be used from other program jump instructions and sub-routine CALL instructions.

(b) the same pointer No. can be used in each of the programs.

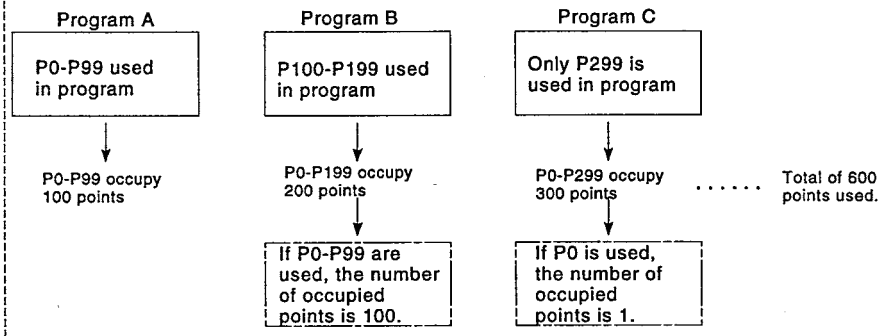


(2) Number of local pointer points

Local pointers up to the specified number of points can be divided among all the programs.
The highest local pointer No. represents the upper limit of the "number of points used" in each program.
Therefore, when local pointers are used at several programs, the pointer settings should begin from P0.
If the total number of pointers (total for all programs) exceeds the designated range, a pointer configuration error (error code:4020) occurs.

Example:

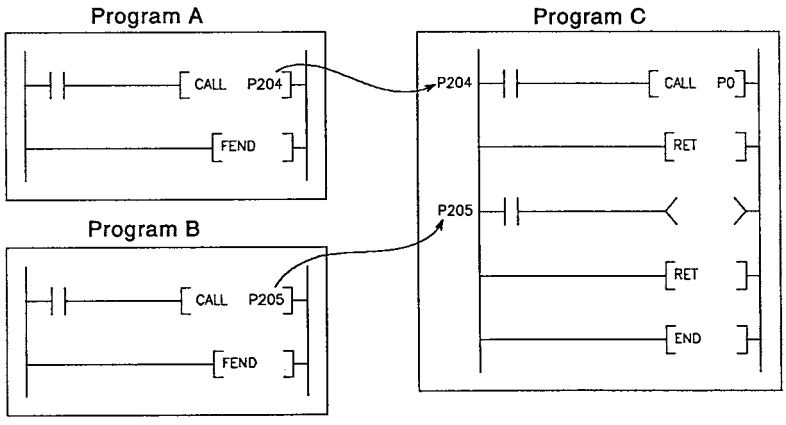
If the number of local pointer points is designated as "400", they cannot be used as follows.



4.9.2 Common pointers

(1) Definition

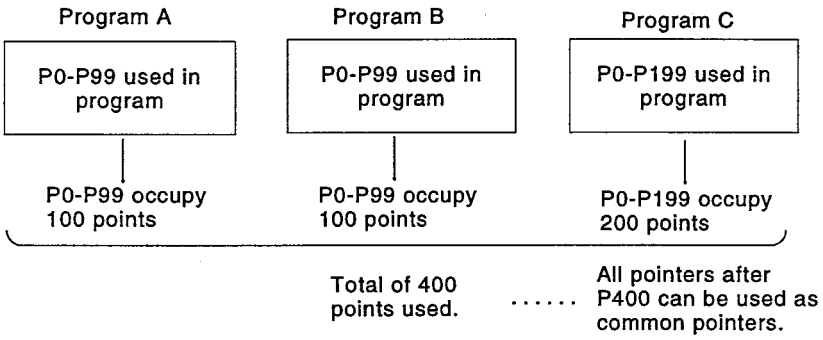
(a) Common pointers are used to call sub-routine programs from all programs being executed in the QnACPU.



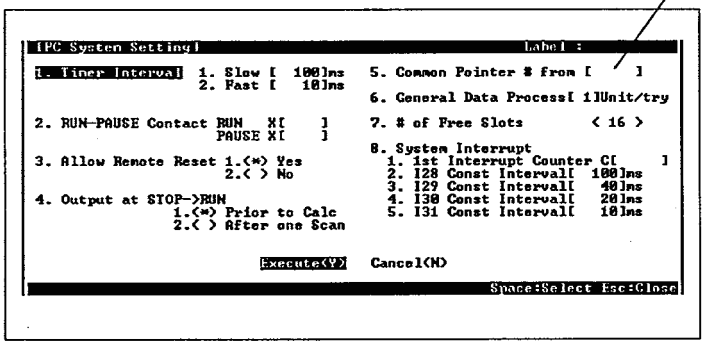
(b) The same pointer No. cannot be used again as a label. Such use will result in a pointer configuration error (error code:4021).

(2) Common pointer range of use

In order to use common pointers, the first common pointer No. must be designated in the PC system settings parameters.
All pointers which follow that "first No." will become common pointers. However, only pointer numbers subsequent to the local pointer range can be designated (by parameter setting) as common pointers.



[PC system settings screen]



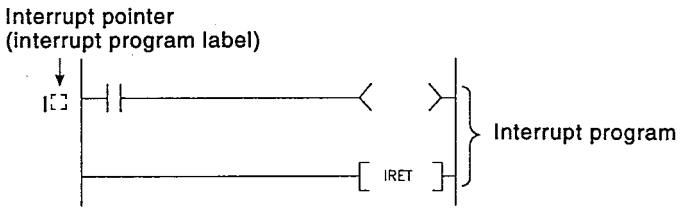
POINT

(1) In the jump instruction, jumping to common pointers in other programs is not allowed. Common pointers should be used only with sub-routine call instructions.

4.10 Interrupt pointers (I)

(1) Definition

(a) Interrupt pointers are used as labels at the beginning of interrupt programs.



(b) A total of 48 interrupt points (10-147) can be used (total for all programs)

(2) Interrupt pointer No. & interrupt factor

(a) As shown below, there are four types of interrupt factor.

- AI61 factor.... Interrupt input from the AI61 interruption module.
- Sequence start generator module factor
  .... Interruption input from special function modules such as a computer link module, etc., which can dictate an interrupt start to the QnACPU (AI61 excluded).
- Internal time factor
  .... Fixed cycle interruption by QnACPU's internal timer.
- Error interruption
  .... Interruption by an error that does not stop sequence program operation.

(b) A list of interrupt pointer Nos. and interrupt factors is given in Table 4.5 below.

Table 4.5 List of Interrupt Pointer Nos. & Interrupt Factors

I No.	Interrupt Factors		Priority Ranking	I No.	Interrupt factors		Priority Ranking	I No.	Interrupt factors		Priority Ranking
I0	AI61 interrupt module factor	1st point	29	I16	Sequence start generator module factor*1	1st point	17	I32	Error factor*3*4	Errors that stop operation	1
I1		2nd point	30	I17		2nd point	18	I33		Vacant	——
I2		3rd point	31	I18		3rd point	19	I34		UNIT VERIFY ERR. FUZE BREAK OFF SP. UNIT ERRO	2
I3		4th point	32	I19		4th point	20				
I4		5th point	33	I20		5th point	21				
I5		6th point	34	I21		6th point	22	I35		OPERATION ERROR SFCP OPE. ERROR	3
I6		7th point	35	I22		7th point	23				
I7		8th point	36	I23		8th point	24				
I8		9th point	37	I24		9th point	25	I36		ICM.OPE.ERROR FILE OPE. ERROR	4
I9		10th point	38	I25		10th point	26				
I10		11th point	39	I26		11th point	27	I37		EXTEND INS. ERR.	5
I11		12th point	40	I27		12th point	28	I38		PRG. TIME OVER	6
I12		13th point	41	I28	Internal timer factor*2	100 ms	48	I39		CHK instruction execution annunciator detection	7
I13		14th point	42	I29		40 ms	47				
I14		15th point	43	I30		20 ms	46	I40 to I46		Vacant	——
I15		16th point	44	I31		10 ms	45	I47		Used as label of EDRSET instruction.	——

REMARKS

1) *1 : 1st to 12th points are allocated in order, beginning from the sequence start generator module installed closest to the QnACPU.

2) *2 : The internal times shown are the default setting times.
These times can be designated in 5 ms units through a 5 ms-1000 ms range (PC system settings parameters).

3) *3 : When an error interruption with "132 (error that stops operation)" occurs, the QnACPU is not stopped until 132 processing is completed.

4) *4 : Execution of error interruptions is prohibited when the power is turned on and during a QnACPU reset. When using interrupt pointer Nos. 132 to 139, set the interruption permitted status by using the IMASK instruction.

4.11 Other Devices

4.11.1 SFC block device (BL)

This device is used for checking if the block designated by the SFC program is active.
For details regarding the use of SFC block devices, refer to the QnACPU Programming Manual (SFC).

4.11.2 SFC transition device (TR)

This device is used for checking if a forced transition is designated for a specified transition condition in a specified SFC program block.
For details regarding the use of SFC transition devices, refer to the QnACPU Programming Manual (SFC).

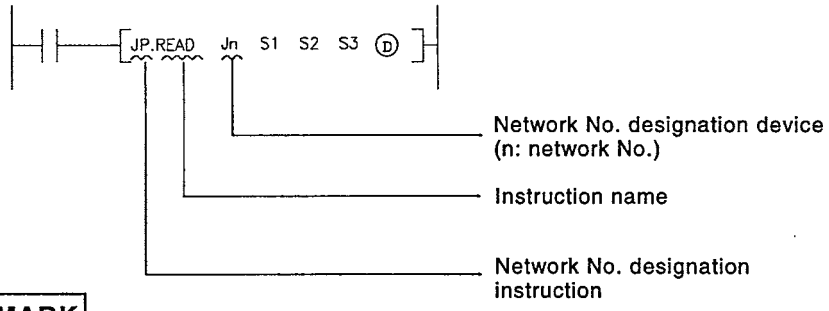
4.11.3 Network No. designation device (J)

(1) Definition

The network No. designation device is used to designate the network No. in data link instructions.

(2) Designating network No. designation device

The network No. designation device is designated in the data link instruction as shown below.



REMARK

1) For details regarding data link instructions, refer to the QnACPU Programming Manual (Common instructions).

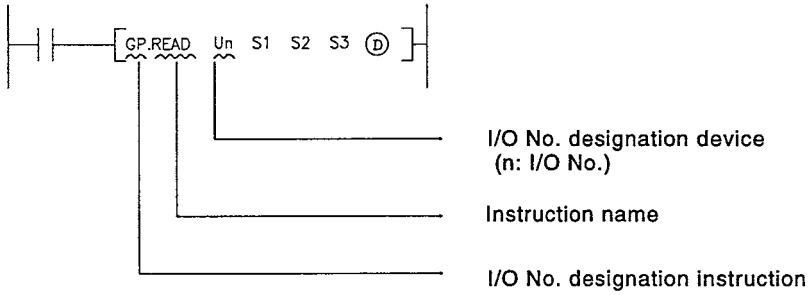
4.11.4 I/O No. designation device (U)

(1) Definition

I/O No. designation devices are used with data link instructions or special function module instruction module instructions to designate I/O numbers.

(2) Designating the I/O No. designation device

I/O No. designation devices are designated with the data link instructions or special function module instructions as shown below.



REMARK

1) For details regarding special function module instructions, refer to the QnACPU Programming Manual (Special Function Module).

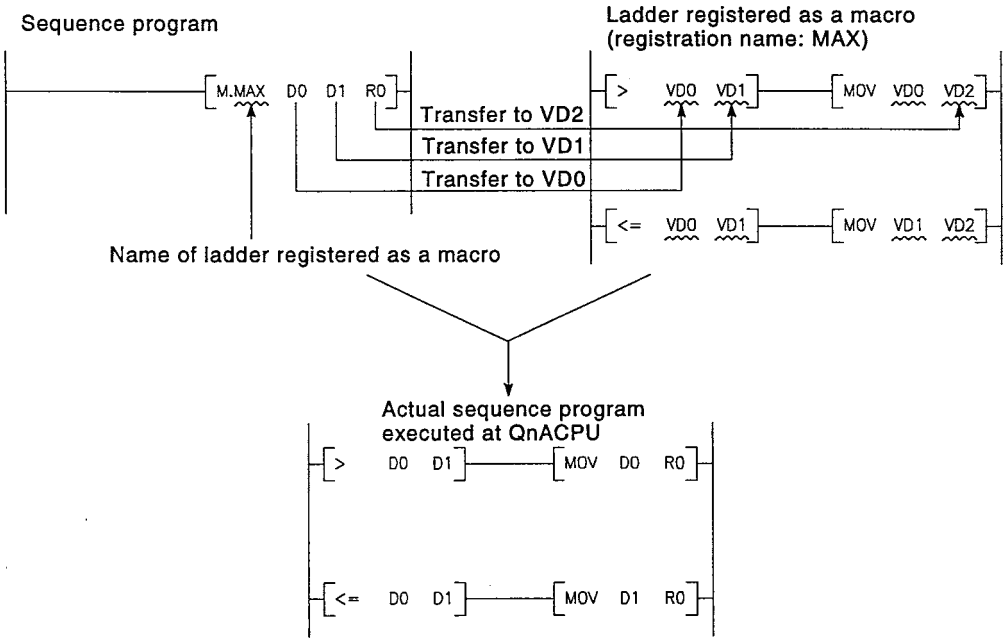
4.11.5 Macro instruction argument device (VD)

(1) Definition

Macro instruction argument devices are used with ladders registered as macros.
When a VD□ setting is designated for a ladder registered as a macro, conversion to the designated device occurs when the macro instruction is executed.

(2) Designating macro instruction argument devices

Macro instruction argument devices are designated for those devices set as "VD□" in ladders registered as macro instructions in macro registration at a peripheral device.*
When using macro instructions in a sequence program, designate devices to correspond to the instruction argument devices used with the ladders registered as macros, in ascending order.



REMARK

1) * : With the macro instruction argument device, VD0 to VD4 can be used in one ladder registered as a macro instruction.

4.12 Constants

4.12.1 Decimal constants (K)

(1) Definition

Decimal constants are devices which designate decimal data in sequence programs.
They are designated as "K□□□□"settings (e.g. K1234), and are stored in the QnACPU in binary (BIN) code.
See Section 3.4.1 for details regarding binary code.

(2) Designation range

The setting ranges for decimal constants are as follows:

- For word data (16 bits) . . . K-32768 to K32767
- For 2-word data (32 bits) . K-2147483648 to K2147483647

4.12.2 Hexadecimal constants (H)

(1) Definition

Hexadecimal constants are devices which designate hexadecimal or BCD data in sequence programs. (For BCD data designations, 0-9 digit designations are used.)
Hexadecimal constants are designated ad "H□□□□" settings (e.g. H1234).
See Section 3.4.3 for details regarding hexadecimal code.

(2) Designation range

The setting ranges for hexadecimal constants are as follows:

- For word data (16 bits) . . . H0 to HFFFF (H0 to H9999 for BCD)
- For 2-word data (32 bits) . H0 to HFFFFFFFF (H0 to H99999999 for BCD)

4.12.3 Real numbers (E)

(1) Definition

Real numbers are devices which designate real numbers in the sequence program.
Real numbers are designated as "E□□□"settings (e.g. E1.234).
See section 3.4.4 for details regarding real numbers.

(2) Designation range

The setting range for real numbers is -1.0×2^{127} to -1.0×2^{-126}, 0, 1.0×2^{-126} to 1.0×2^{127}.

(3) Designation method

Real numbers can be designated in sequence programs by a "normal expression" or an "exponential expression".

- Normal expression..... The specified value is designated as it is F or example, 10.2345 becomes E10.2345.
- Exponential expression... The specified value is multiplied by a "$\times 10^{n}$"exponent.
  For example, 1234 becomes E1.234 + 3.*

REMARK

1) *:The "+3" in the above example represents a 10^{n}value (10^{3}).

4.12.4 Character string (")

(1) Definition

Character string constants are devices used to designate character strings in sequence programs.
They are designated by quotation marks (e.g. "ABCD1234").

(2) Usable characters

All ASCII code characters can be used in character strings.

(3) Number of designated characters

Character strings extend from the designated character to the NUL code (00_H).

4.13 Convenient Uses for Devices

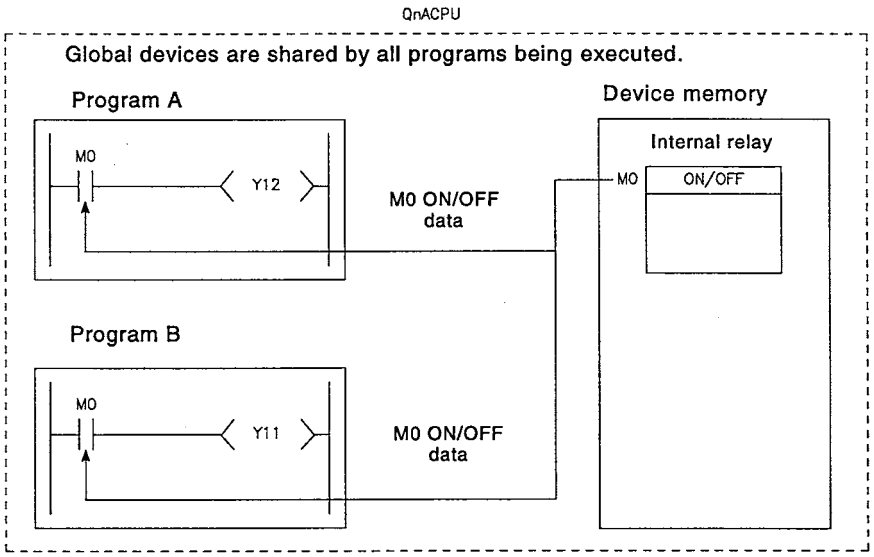
When executing multiple programs in the QnACPU, local devices among the internal user devices can be designated to execute each of the programs in an independent manner.
Moreover, the device initial settings can be used to designate device and special function module data settings without using a program.

4.13.1 Global devices & local devices

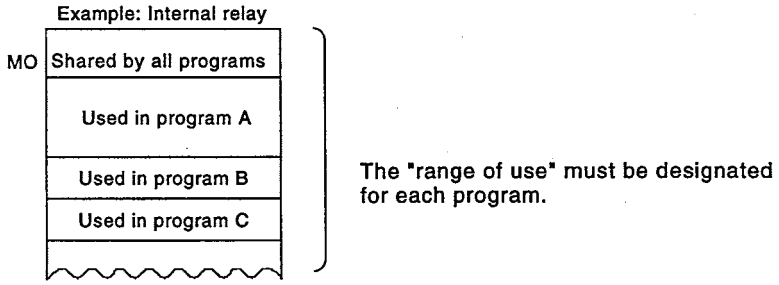
A number of programs can be stored and executed in the QnQCPU.
QnACPU devices which can be shared by all the programs are "global devices", and those used independently by each of the programs are "local devices".

(1) Global devices

(a) Global devices are devices which can be shared by all the programs being executed in the QnACPU. Global devices are stored in the QnACPU's device memory, with all programs using the same devices.

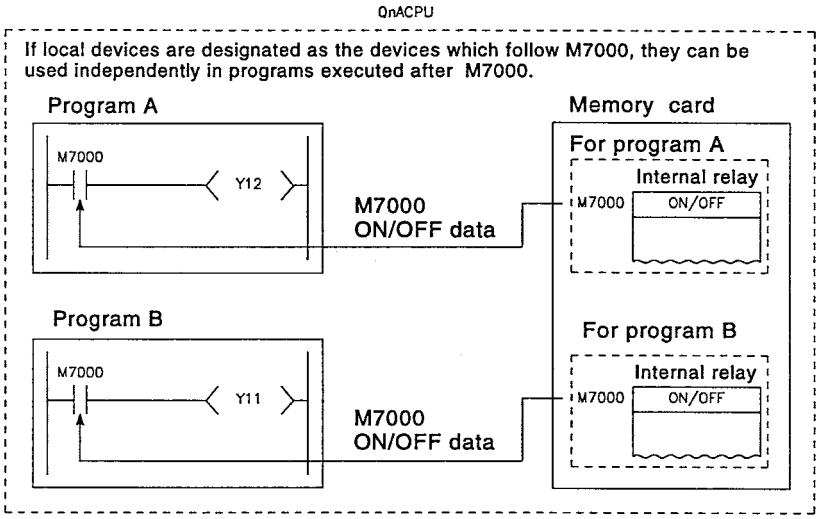


(b) When executing multiple programs, the "shared range" for all programs, and the "independent range" for each program must be designated in advance.



(2) Local devices

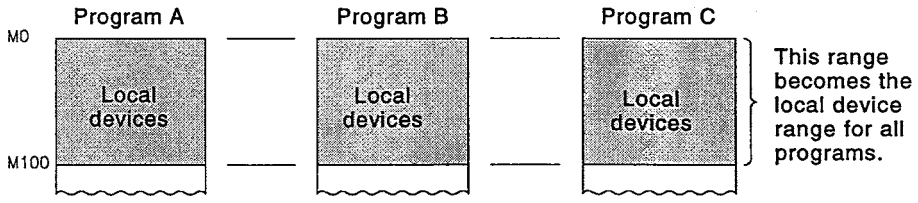
(a) Local devices are devices which are used independently by the programs.
The use of local devices permits programming of multiple "independent execution" programs without regard to other programs.
However, because local devices are stored in the memory card, an memory card is required in order to use them.
Local devices cannot be used without an memory card.



(b) Five device types can be used as local devices: internal relays (M), edge relays (V), timers (T,ST), counters (C), and data registers (D).

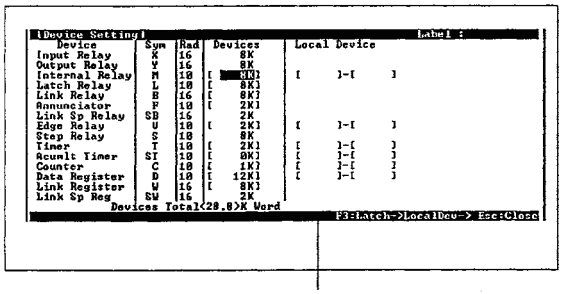
(c) Local device designation

1) In order to use the above devices as local devices, a local device range of use setting must be designated in the device settings parameters.
Note that the range designated for local devices applies in all programs, and cannot be changed for individual programs.
For example, if the local device range is designated as M0-M100, this range will be used for local devices in all programs.

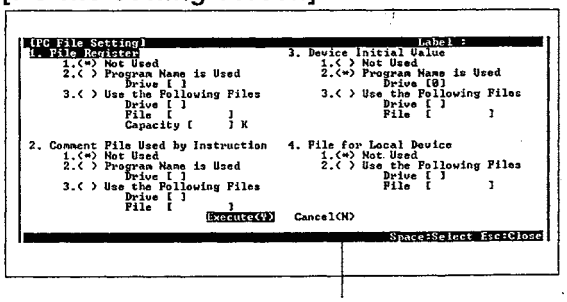


2) When local device settings are designated, the drive and file name where the local device data is to be stored must be designated in the PC file settings in the parameter mode

[Device range setting]



[PC file setting screen]



(d) When local devices are used, an exchange occurs between the local device file data stored in the memory card, and the data in the QnACPU's device area. The scan time is therefore extended by this data exchange time.

- Q2ACPU (S1)
- Q2ASCPU (S1)

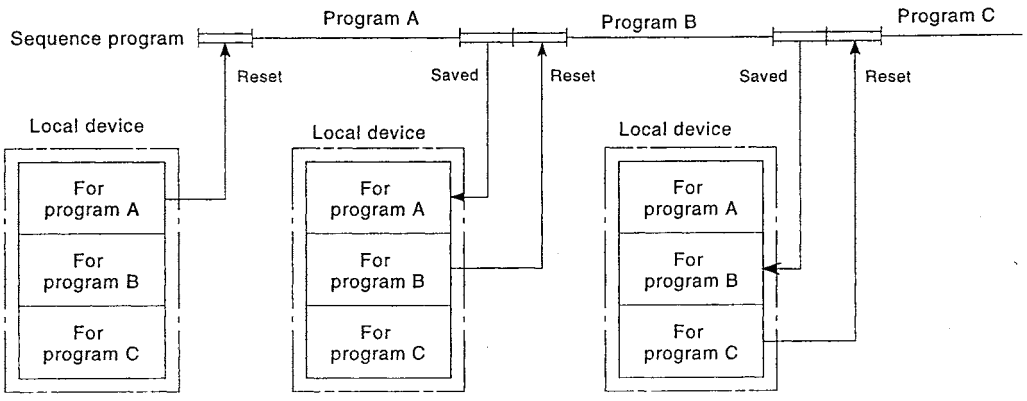
560 + 1.3 x (number of local device words)* x number of program (μs)

- Q3ACPU

320 + 1.0 x (number of local device words) x number of program (μs)

- Q4ACPU
- Q4ARCPU
- Q2ASHCPU(S1)

220 + 0.8 x (number of local device words) x number of program (μs)



(e) Data in the local device are all cleared when the CPU is switched from STOP to RUN.

REMARK

1) * : See section 4.1.2 (item 2) for details regarding the "number of words" for local devices.

POINTS

(1) Concerning the use of local devices in interrupt programs, refer to item 3.1.3.

(2) Unless specifically designated as "local devices", all devices are global devices.

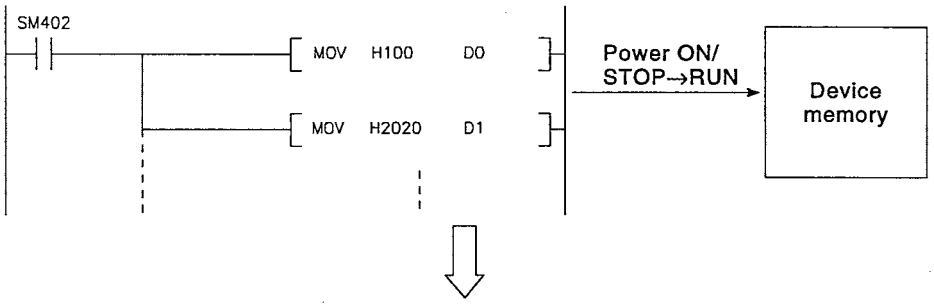
(3) Concerning the use of local devices in sub-routine programs, refer to item 3.1.2.

4.13.2 Device initial values

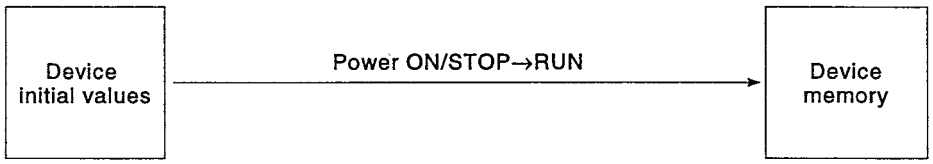
(1) Definition

(a) Using device initial setting values, the data used for a program can be stored in device or special function module buffer memories without using a data setting program (initial processing program).

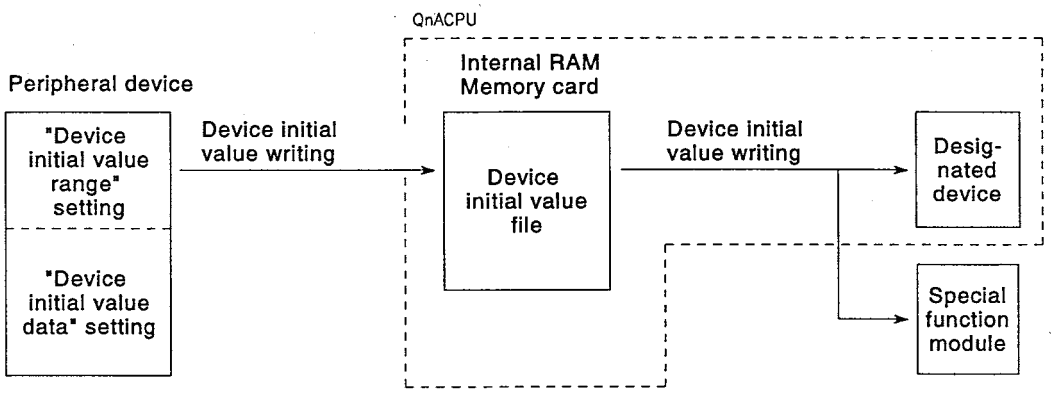
[Data setting by initial program]



[Data setting using device initial values]



(b) In order to use the device initial values, the device initial data must be created in advance at the peripheral device, and this data must be stored as a device initial value file in the QnACPU's memory card. At power ON, or on switching from STOP to RUN, the QnACPU writes the data from the device initial value file to the specified device or special function module buffer memory.



(c) Device initial values can be used at the following devices:

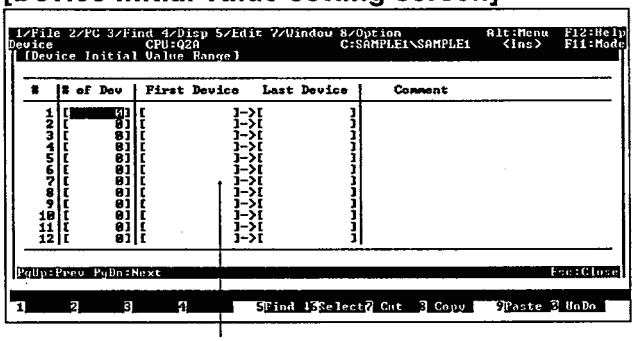
1) Timer present value (T)
2) Retentive timer present value (ST)
3) Counter present value (C)
4) Data register (D)
5) Special register (SD)
6) Link register (W)
7) Special link register (SW)
8) File register (R0-R32767)
9) Special function module device (U□\G□)
10) Link direct device (J□\W□, J□\SW□)

(2) Procedure for using device initial values

(a) Designate the device initial value range settings in the device mode, in the device initial value setting screen.

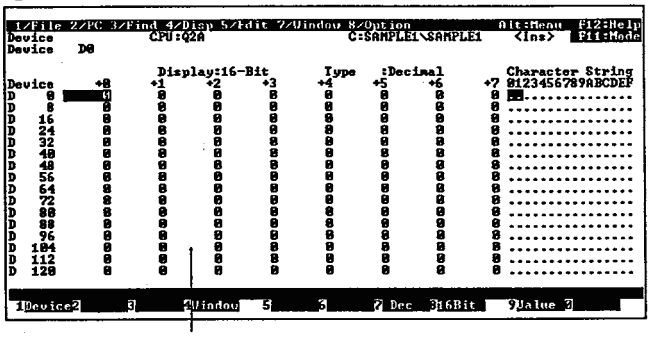
(b) Designate the device initial value data settings in the device mode screen.

[Device initial value setting screen]



Device initial value range setting area

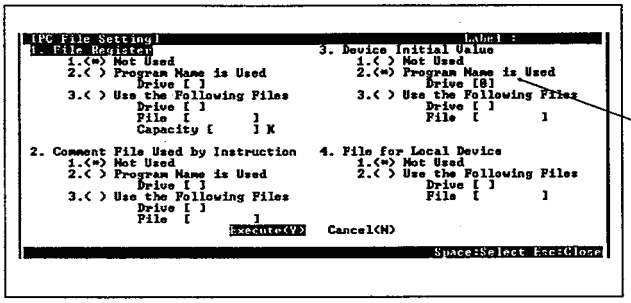
[Device mode]



Device initial value data setting area

(c) In the PC file settings in the parameter mode, designate the name of the file where the device initial value data is to be stored.

[PC file settings screen]



Setting area for file which contains the device initial value data

(d) Write the device initial value data and parameter settings to the QnACPU.

(3) Precautions for the use of device initial values

(a) In cases where both device initial value data and latch range data are present, the device initial value data takes precedence. Therefore, the latch range data is overwritten by device initial value data at power ON.

(b) Device initial values cannot be used in the following areas.

1) In an area of a special function module's buffer memory area where channel switching is required.
Example: Channel 2 area of the AJ71E71 Ethernet interface module's buffer memory.

2) In special function module areas where the writing sequence is fixed.
Example: Initial settings of A68AD analog/digital converter module.

3) In areas where no settings are desired at STOP → RUN switching (data which is set at power ON, and which is changed by the program).

REMARKS

1) For details regarding the setting procedures for the "device initial value range" and "device initial value data" items, refer to the SW□IVD-GPPQ GPP Function Software Package Operating Manual (Offline).

2) For details regarding the procedure for writing the device initial values to the QnACPU, refer to the SW□IVD-GPP GPPQ Function Software Package Operating Manual (Online).

5. PARAMETER LIST

Following is a list of parameters used in QnACPU.
For details of each of these parameters, refer to appropriate sections or manuals.

Item			Parameter No.	Description	
PC name			–	Sets labels and comments of peripheral devices to the CPU. Does not influence CPU module operation.	
	Label		0000H	Sets the CPU module labels.	
	Comment		0001H	Sets the CPU module comments.	
PC system			–	Sets parameters for the CPU module system.	
	Timer limit	Low-speed timer	1001H	Sets the time limits for low/high-speed timers.	
		High-speed timer			
	RUN-PAUSE contact		1002H	Sets the contact points to control RUN/PAUSE of the CPU module.	
	Remote reset		1003H	Sets enable/disable of remote reset operation.	
	STOP→ RUN output mode		1004H	Sets the output mode for STOP → RUN.	
	Common pointer number		1005H	Sets the head number of a common pointer.	
	General data processing		1006H	Sets the number of modules to be processed in one general data processing.	
	Number of vacant slot points		1007H	Sets the number of points for a vacant slot.	
	System-interrupt	Interrupt counter	1008H	Sets the head number of the interrupt counter and fixed-cycle-interval interrupt pointer.	
		Fixed cycle interval			
PC file			–	Sets the various files used by the CPU module.	
	File register		1100H	Sets a file-register file,, to be used. Sets a file-register file to be used.	
	Instruction comment file		1101H	Sets a file for instruction comments.	
	Device initial vale		1102H	Sets a file for device initial values.	
	File for local device		1103H	Sets the file for a local device.	
Device			–	Sets the points and latch range for each device.	
	Device points		2000H	Sets the device points to be used.	
	Latch range (Valid latch clear key)		2001H	Sets the latch range when the latch clear key is valid.	
	Latch range (Invalid latch clear key)		2002H	Sets the latch range when the latch clear key is invalid.	
	Local device		2003H	Sets the device range to be used for a local device.	

	Description		Reference section/manual
	Default value	Allowable range	
	–	–	USER'S MANUAL *
	None	Half width character, maximum of 10 characters	
	None	Half width character, maximum of 64 characters	
	–	–	–
	100 ms	10 ms to 1000 ms (10 ms/unit)	Section 4.2.10
	10 ms	1 ms to 100 ms (1 ms/unit)	
	None	X0 to X1FFF	USER'S MANUAL *
	Disabled	Enable/Disable	
	Prior to operation	Prior to operation/after 1 scan	
	None	P0 to P4095	Section 4.9.2
	1 unit	1 to 6 units	USER'S MANUAL *
	16 points	0 to 64 points (16 points/unit)	
	None	C0 to C65535	Section 4.10
	I28 → 100 ms I29 → 40 ms I30 → 20 ms I31 → 10 ms	5 to 1000 ms (5 ms/unit)	
	–	–	–
	Do not specify in parameter.	- Do not specify in parameter. - Use the same file name as the program file. - Do not create.	Section 4.7
	Do not specify in parameter.	- Do not specify in parameter. - Use the same file name as the program file. - Use specification file.	USER'S MANUAL *
	Use the same file name as the program file.	- Do not specify in parameter. - Use the same file name as the program file. - Use specification file.	Section 4.13.2
	Do not specify in parameter.	- Do not specify in parameter. - Use specification file.	Section 4.13.1
	–	–	–
	X → 8 k points Y → 8 k points M → 8 k points L → 8 k points B → 8 k points F → 2 k points SB → 2 k points V → 2 k points S → 8 k points T → 2 k points ST → 0 k points C → 1 k points D → 12 k points W → 8 k points SW → 2 k points	Fixed number of points for X(8 k points), Y(8 k points), S(8 k points), SB(2 k points) and SW(2 k points). Total points including above must not exceed 28.8 k words. Maximum of 32 k points per device can be set. Maximum of 64 k points per bit device can be set.	Section 4.1
	None	One range per device	USER'S MANUAL *
	None	One range per device	
	None	One range per device	Section 4.13.1

*: Indicates the CPU Module User's Manual being used.

Item			Parameter No.	Description	
PLC RAS setting			–	Makes various settings for RAS functions.	
	WDT setting	WDT setting	3000H	Sets the watchdog timer of the CPU module.	
		Initial execution monitor time			
		Low-speed execution monitor time			
	Error check		3001H	Sets whether the specified error will be detected or not.	
	Error-time operation mode		3002H	Sets the operation mode of the CPU module when error is detected.	
	Constant scan		3003H	Sets the constant scan time.	
	Annunciator display mode	F-number display	3004H	Sets the display mode when the annunciator turns on.	
		Comment display			
		Occurrence time			
	Error history		3005H	Sets the storage destination of the CPU module error history.	
	Low-speed program execution time		3006H	Sets the time for execution of the low-speed program.	
I/O allocation			–	Sets the loading status of each system unit.	
	Slot setting	Type	4000H	Sets the type, points, head I/O number, etc. of a module.	
		Points			
		Head XY			
		Model			
	Base setting	Power supply module	4001H	Sets the models of the power supply module and extension cable. Does not influence CPU module operation.	
		Extension cable			
MELSECNET/Ethernet setting			–	Sets the link parameters of the MELSECNET II data link system, network parameters of the MELSECNET/10 network system and the parameters of Ethernet.	
		Number of modules	5000H		
		Valid module for other station access	5001H		
	MELSECNET (II), MELSECNET/10 network setting	Parameters transferred between data links	5002H		
		Routing parameters	5003H		
		Network setting	5NM0H *1		
		Network refresh parameters	5NM1H *1		
		Common parameters	5NM2H *1		
		Station-specific parameters	5NM3H *1		
		I/O allocation	5NM4H *1		
	Ethernet network setting	Group No.	9N00H		
		IP address			

	Description		Reference section/manual
	Default value	Allowable range	
	–	–	–
	200ms	10 ms to 2000 ms (10 ms/unit)	USER'S MANUAL *2
	None	10 ms to 2000 ms (10 ms/unit)	
	None	10 ms to 2000 ms (10 ms/unit)	
	Check	Check error	
	Stop	Stop/Continue	
	None	5 ms to 2000 ms (5 ms/unit)	
	Display	Display/Do not display	
	Do not display	Display/Do not display	
	Do not display	Display/Do not display	
	Built-in RAM	Built-in RAM/Specified history file	
	None	5 ms to 2000 ms (5 ms/unit)	Section 3.2.3
	–	–	–
	None	Vacant/Input/Output/Special	USER'S MANUAL *2
	None	0 to 64 points (16 points/unit)	
	None	0 to 1FFF (10H/unit: hexadecimal)	
	None	Half-width characters, maximum of 16 characters	
	None	Half-width characters, maximum of 16 characters	
	–	Refer to the QnA/Q4AR Compatible MELSECNET/10 Network System Reference Manual.	QnA/Q4AR Compatible MELSECNET/10 Network System Reference Manual MELSECNET, MELSECNET/B Data Link System Reference Manual

*1: N and M indicate the following.
N: Indicates the module number.
M: Indicates the network type.

M	Network type	M	Network type
0H	MELSECNET/10 (default)	9H	MELSECNET II (local station)
1H	MELSECNET/10 (control station)	AH	MELSECNET/10 (standby station)
2H	MELSECNET/10 (normal station)	BH	MELSECNET/10 (multiple remote master)
3H	MELSECNET/10 (remote master station)	CH	MELSECNET/10 (parallel remote master)
4H	MELSECNET (master station)	DH	MELSECNET/10 (multiple remote sub master, there is no remote master in host CPU)
5H	MELSECNET II mixed (master station)		
6H	MELSECNET II (master station)	EH	MELSECNET/10 (multiple remote sub master, there is remote master in host CPU)
7H	MELSECNET (local station)		
8H	MELSECNET II mixed (local station)	FH	MELSECNET/10 (parallel remote sub master)

*2: Indicates the CPU Module User's Manual being used.

Item			Parameter No.	Description
MELSECNET/MINI setting			–	Sets the parameters for automatic refresh of MELSECNET/MINI system.
Number of master modules			6000H	Sets the number of MELSECNET/MINI system master modules to be used.
	MELSECNET/MINI details	Master module head I/O number	600NH *1	Sets details of MELSECNET/MINI system automatic refresh operation.
		Model & station name		
		Receive data batch refresh		
		Send data batch refresh		
		Number of communication error retries		
		FROM/TO instruction access priority		
		Receive data clear during communication error		
		Error station detection bit data		
		Error number		
		MINI link operation during CPU stop		
		Circuit error check		
Auxiliary setting			7000H	Sets the parameters to be used for multiple programs.
	Program setting			Sets the programs to be executed from among multiple programs.
	Boot setting			Sets the boot operation file, etc.
SFC setting			–	Sets the parameters necessary for the SFC program.
	SFC program start mode		8002H	
	Start condition		8003H	
	Block stop-time output mode		8005H	
X/Y allocation verification			–	Verifies the I/O allocation settings. Does not influence the CPU module operation.

	Description		Reference section/manual
	Default value	Allowable range	
	–	–	USER'S MANUAL *2
	0	0 to 8 modules	
	None	CPU module I/O points	
	MINIS3	MINIS3/MINI ( ) stations	
	200H starting with X1000	X, M, L, B, T, ST, C, D, W, R, ZR, vacant (multiple of 16 for bit devices)	
	200H starting with Y1000	Y, M, L, B, T, ST, C, D, W, R, ZR, vacant (multiple of 16 for bit devices)	
	5 times	0 to 32 times	
	CPU priority	CPU priority/Link priority	
	Clear	Clear/Retain	
	None	M, L, B, T, ST, C, D, W, R, ZR, vacant	
	None	D, W, T, ST, C, R, ZR	
	Stop	Continue/Stop	
	Retention data	Test message/OFF data/Retention data	
	–	–	Section 3.2
	None	Program name, Scan/Low speed/Default/Standby	
	None	File name, type, transfer source drive, transfer target drive	
	–	QCPU (Q Mode)/QnACPU Programming Manual (SFC)	QCPU (Q Mode)/QnACPU Programming Manual (SFC)
	–	–	SW□IVD/NX-GPPQ Operating Manual (Offline)

*1: N indicates the master module number. (N: 1 to 8)
*2: Indicates the CPU Module User's Manual being used.

6. PROCEDURE FOR WRITING PROGRAMS TO QnACPU

The procedure for writing programs (created at a peripheral device) to the QnACPU is described in this section.

6.1 Writing Procedure For 1 Program

The procedure for writing one program created at a peripheral device to the QnACPU and executing it is described here.

6.1.1 Items to consider when creating one program

In order to create a program, the program size, number of device points used, and the program file name, etc., must be set in advance.

(1) Program size considerations

Check that CPU's program capacity is adequate for storing the program and parameter data. The program capacities of the CPUs are shown below.

- Q2AS(H)CPU : 28 k steps
- Q2ACPU : 28 k steps
- Q2AS(H)CPU-S1 : 60 k steps
- Q2ACPU-S1 : 60 k steps
- Q3ACPU : 92 k steps
- Q4ACPU : 124 k steps
- Q4ARCPU : 124 k steps

If the CPU capacity is only adequate for the program, the parameter data should be stored in the memory card.

(2) Designating a program file name

The file name of the program to be stored in the QnACPU must be designated.
This file name is used when writing the program from the peripheral device to the QnACPU, and when executing the program in the QnACPU.
See Chapter 2 for details regarding file names.

(3) Designating devices

The number of devices required for the program must be determined.
See Chapter 4 for details regarding devices which can be used in the QnACPU.

(4) Device initial value setting

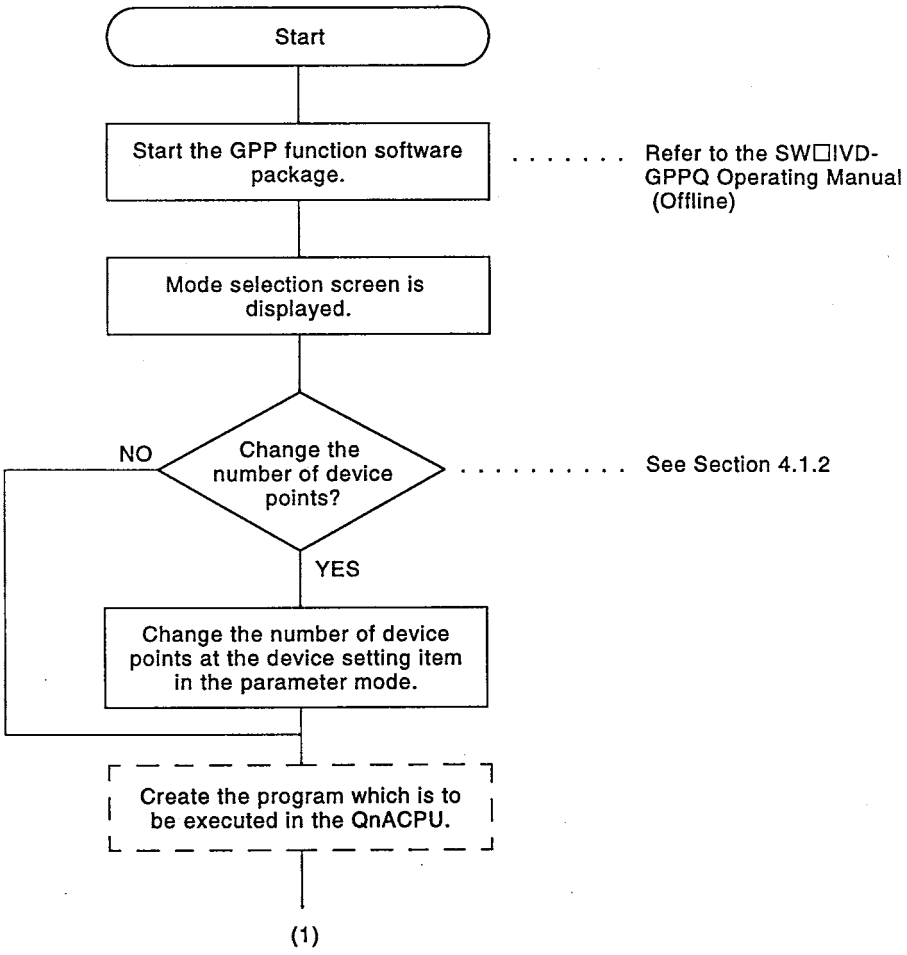
Designate whether or not the device initial value settings are to be used in the QnACPU devices and special function unit data.
See Section 4.13.2 for details regarding device initial values.

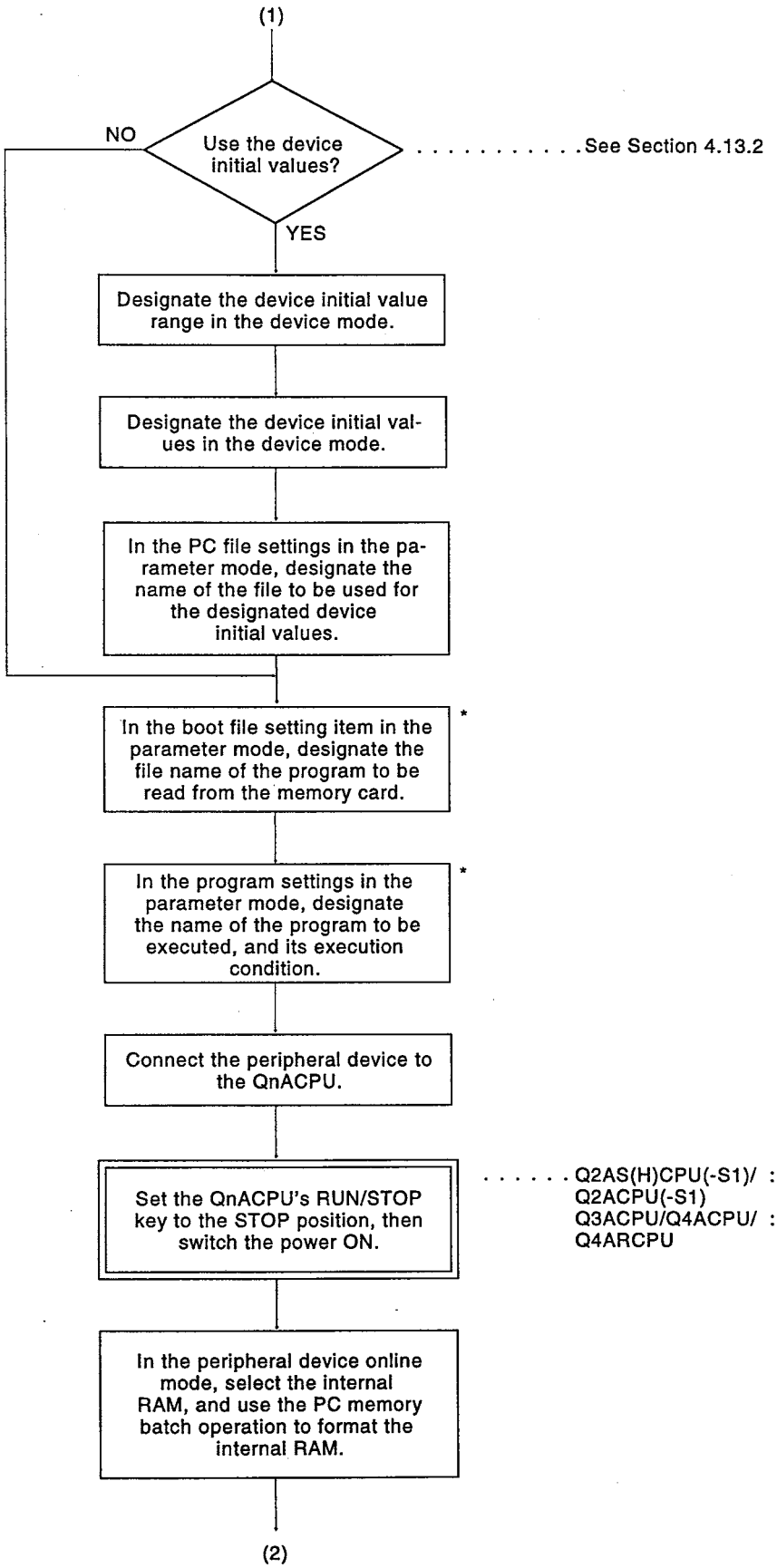
6.1.2 Procedure for writing programs to the QnACPU

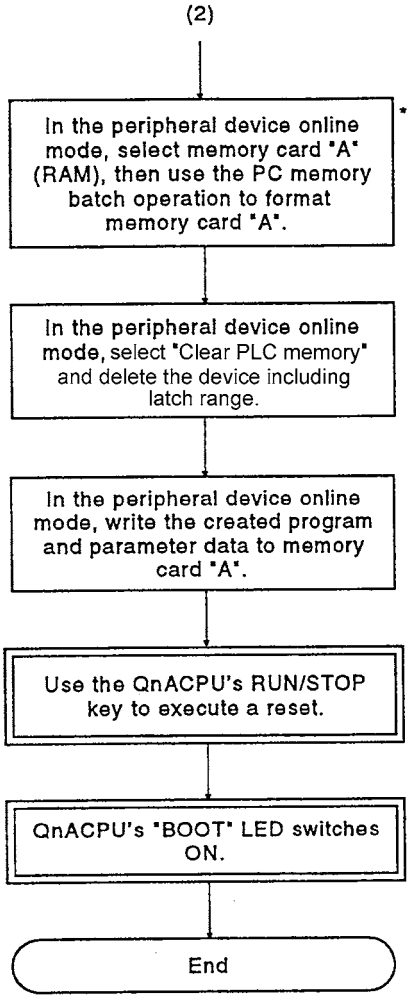
The procedure for writing programs and parameters (created at the peripheral device) to the memory card installed in the QnACPU memory card interface "A" is shown below.
In order to write programs and parameters to the QnACPU memory card, the memory card must be installed, and the boot and drive (where parameters are stored) settings must be designated by the QnACPU DIP switches (SYS 1).
For details regarding QnACPU DIP switches, refer to the User's Manual of the CPU module used.

When writing programs and parameters to the QnACPU internal RAM, the steps indicated by asterisks (*) below are not required.

Procedural steps shown in ☐ boxes are performed at the peripheral device, and those shown in ▣ boxes are performed in the QnACPU.




(1)
Use the device initial values?
NO
YES
. . . . . .See Section 4.13.2
Designate the device initial value range in the device mode.
Designate the device initial values in the device mode.
In the PC file settings in the parameter mode, designate the name of the file to be used for the designated device initial values.
In the boot file setting item in the parameter mode, designate the file name of the program to be read from the memory card.
*
In the program settings in the parameter mode, designate the name of the program to be executed, and its execution condition.
*
Connect the peripheral device to the QnACPU.
Set the QnACPU's RUN/STOP key to the STOP position, then switch the power ON.
. . . . . Q2AS(H)CPU(-S1)/ Q2ACPU(-S1) Q3ACPU/Q4ACPU/ Q4ARCPU : ERROR LED switches ON : PARAMETER ERROR is displayed
In the peripheral device online mode, select the internal RAM, and use the PC memory batch operation to format the internal RAM.
(2)


(2)
In the peripheral device online mode, select memory card "A" (RAM), then use the PC memory batch operation to format memory card "A".
In the peripheral device online mode, select "Clear PLC memory" and delete the device including latch range.
In the peripheral device online mode, write the created program and parameter data to memory card "A".
Use the QnACPU's RUN/STOP key to execute a reset.
QnACPU's "BOOT" LED switches ON.
End

6.2 Procedure For Multiple Programs

The procedure for writing multiple programs (programs split up according to function, process, designer) to the QnACPU is described below.

6.2.1 Items to consider when creating multiple programs

To create multiple programs, it is necessary to decide in advance the size of each program, the device used, and the program file name, etc.

(1) Program size considerations

Check that the CPU's program capacity is adequate for storing the programs. The program capacities of the CPUs are shown below.

- Q2AS(H)CPU : 28 k steps
- Q2ACPU : 28 k steps
- Q2AS(H)CPU-S1 : 60 k steps
- Q2ACPU-S1 : 60 k steps
- Q3ACPU : 92 k steps
- Q4ACPU : 124 k steps
- Q4ARCPU : 124 k steps

Decide whether the parameters are to be stored in the internal RAM or in the memory card.
If they are to be stored in the internal RAM, the area available for the program will be the capacity shown above, minus the parameter data size.

(2) Designating a program file name

Designat the file name of the program to be stored in the QnACPU. This file name is used when writing the program from the peripheral device to the QnACPU, and when executing the program at the QnACPU. See Chapter 2 for details regarding file names.

(3) Designating the program execution conditions

In order to execute multiple programs in QnACPU, execution conditions must be designated for each program.
Execution is impossible for programs without file name and execution condition settings.
See Section 3.2 for details regarding execution conditions.

(4) Designating devices

(a) Designate the number of device points used in each program, and the number of device points which are shared by all programs.
See Chapter 4 for details regarding devices which can be used in the QnACPU.

(b) Designate whether or not the internal relays, edge relays, timers, counters, and data registers of each program are to be designated as local pointers.
See Section 4.13.1 for details regarding local pointers.

(c) When creating sub-routine programs, designate whether or not common pointers are to be used.
See Section 4.9.2 for details regarding common pointers.

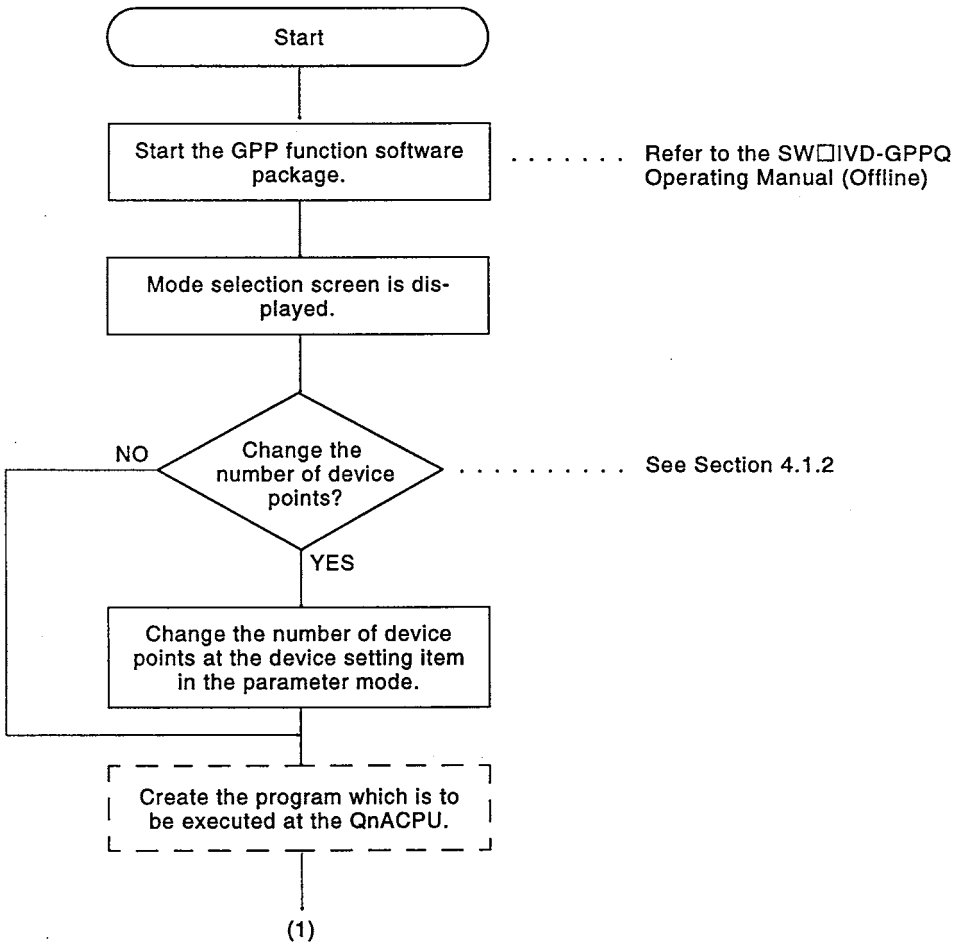
(5) Device initial value setting
Designate whether or not the device initial value settings are to be used for the QnACPU devices and special function unit data.
See Section 4.13.2 for details regarding device initial values.

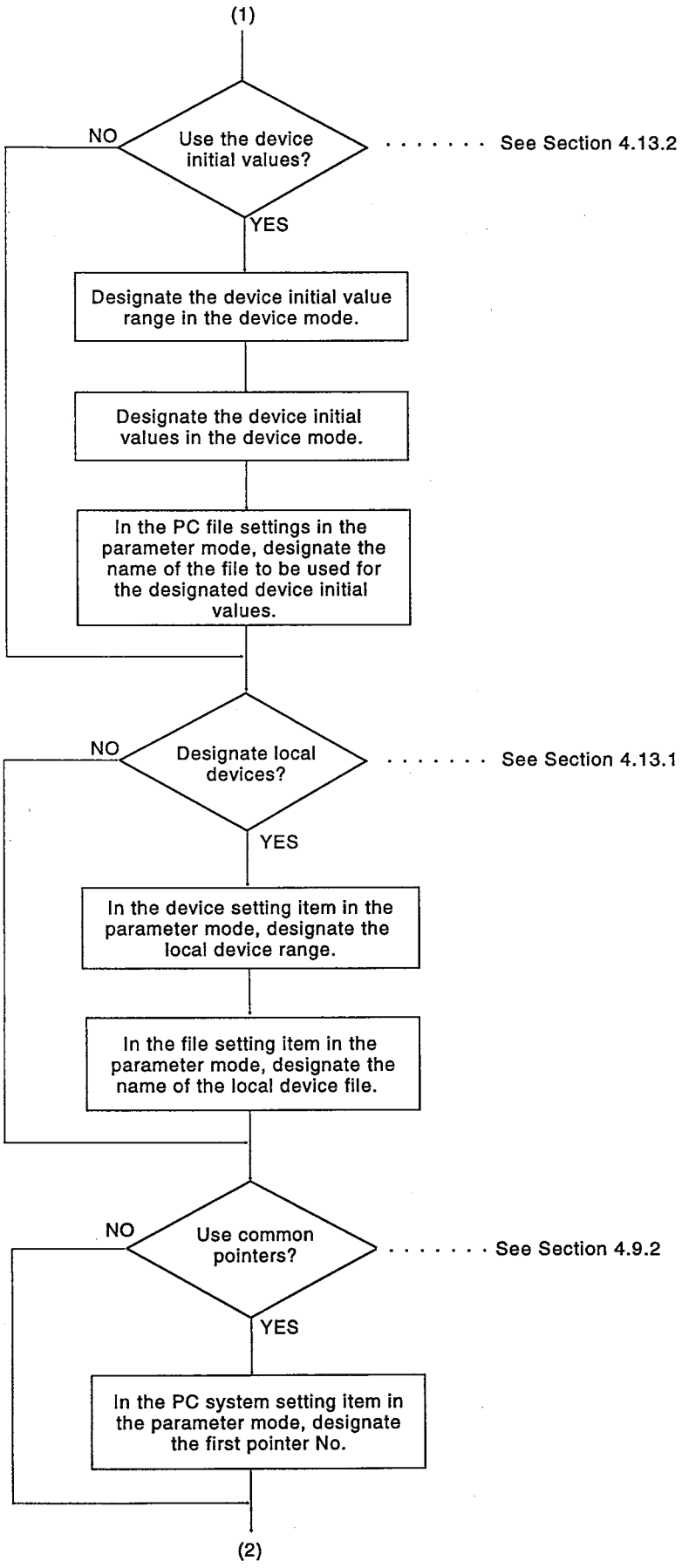
6.2.2 Procedure for writing programs to the QnACPU

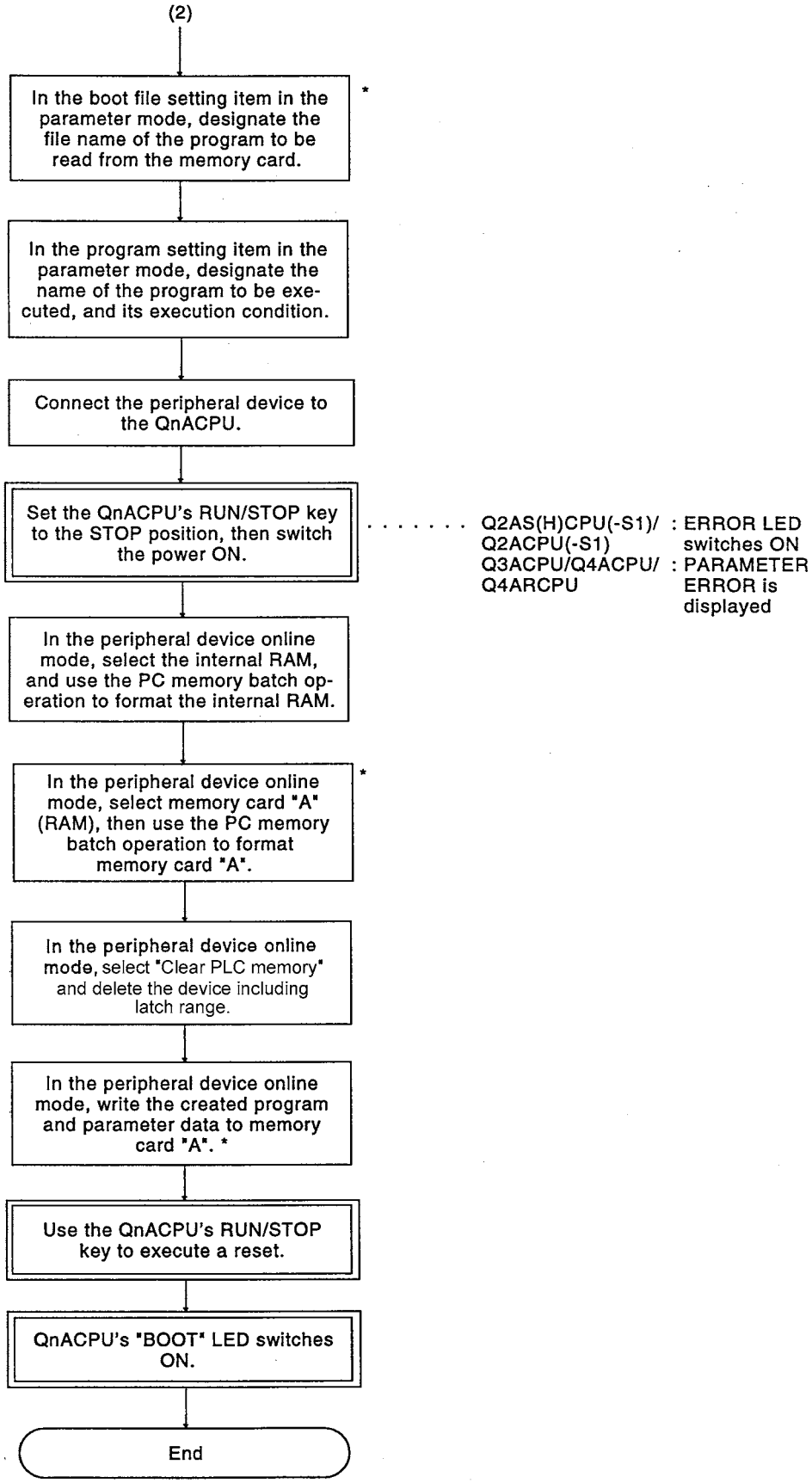
The procedure for writing programs and parameters (created at the peripheral device) to the memory card installed in the QnACPU memory card interface "A" is shown below.
In order to write programs and parameters to the QnACPU memory card, the memory card must be installed, and the boot and drive (where parameters are stored) settings must be designated by the QnACPU DIP switches (SYS 1).
For details regarding QnACPU DIP switches, refer to the User's Manual.

When writing programs and parameters to the QnACPU internal RAM, the steps indicated by asterisks (*) below are not required.

Procedural steps shown in □ boxes are performed at the peripheral device, and those shown in ▣ boxes are performed at the QnACPU.




(1)
Use the device initial values?
NO
YES
See Section 4.13.2
Designate the device initial value range in the device mode.
Designate the device initial values in the device mode.
In the PC file settings in the parameter mode, designate the name of the file to be used for the designated device initial values.
Designate local devices?
NO
YES
See Section 4.13.1
In the device setting item in the parameter mode, designate the local device range.
In the file setting item in the parameter mode, designate the name of the local device file.
Use common pointers?
NO
YES
See Section 4.9.2
In the PC system setting item in the parameter mode, designate the first pointer No.
(2)


(2)
In the boot file setting item in the parameter mode, designate the file name of the program to be read from the memory card.
*
In the program setting item in the parameter mode, designate the name of the program to be executed, and its execution condition.
Connect the peripheral device to the QnACPU.
Set the QnACPU's RUN/STOP key to the STOP position, then switch the power ON.
Q2AS(H)CPU(-S1)/ Q2ACPU(-S1) : ERROR LED switches ON
Q3ACPU/Q4ACPU/ Q4ARCPU : PARAMETER ERROR is displayed
In the peripheral device online mode, select the internal RAM, and use the PC memory batch operation to format the internal RAM.
In the peripheral device online mode, select memory card "A" (RAM), then use the PC memory batch operation to format memory card "A".
*
In the peripheral device online mode, select "Clear PLC memory" and delete the device including latch range.
In the peripheral device online mode, write the created program and parameter data to memory card "A". *
Use the QnACPU's RUN/STOP key to execute a reset.
QnACPU's "BOOT" LED switches ON.
End

INDEX

WARRANTY

Please confirm the following product warranty details before using this product.

1. Gratis Warranty Term and Gratis Warranty Range

If any faults or defects (hereinafter "Failure") found to be the responsibility of Mitsubishi occurs during use of the product within the gratis warranty term, the product shall be repaired at no cost via the sales representative or Mitsubishi Service Company.
However, if repairs are required onsite at domestic or overseas location, expenses to send an engineer will be solely at the customer's discretion. Mitsubishi shall not be held responsible for any re-commissioning, maintenance, or testing on-site that involves replacement of the failed module.

[Gratis Warranty Term]

The gratis warranty term of the product shall be for one year after the date of purchase or delivery to a designated place.
Note that after manufacture and shipment from Mitsubishi, the maximum distribution period shall be six (6) months, and the longest gratis warranty term after manufacturing shall be eighteen (18) months. The gratis warranty term of repair parts shall not exceed the gratis warranty term before repairs.

[Gratis Warranty Range]

(1) The range shall be limited to normal use within the usage state, usage methods and usage environment, etc., which follow the conditions and precautions, etc., given in the instruction manual, user's manual and caution labels on the product.
(2) Even within the gratis warranty term, repairs shall be charged for in the following cases.
1. Failure occurring from inappropriate storage or handling, carelessness or negligence by the user. Failure caused by the user's hardware or software design.
2. Failure caused by unapproved modifications, etc., to the product by the user.
3. When the Mitsubishi product is assembled into a user's device, Failure that could have been avoided if functions or structures, judged as necessary in the legal safety measures the user's device is subject to or as necessary by industry standards, had been provided.
4. Failure that could have been avoided if consumable parts (battery, backlight, fuse, etc.) designated in the instruction manual had been correctly serviced or replaced.
5. Failure caused by external irresistible forces such as fires or abnormal voltages, and Failure caused by force majeure such as earthquakes, lightning, wind and water damage.
6. Failure caused by reasons unpredictable by scientific technology standards at time of shipment from Mitsubishi.
7. Any other failure found not to be the responsibility of Mitsubishi or that admitted not to be so by the user.

2. Onerous repair term after discontinuation of production

(1) Mitsubishi shall accept onerous product repairs for seven (7) years after production of the product is discontinued. Discontinuation of production shall be notified with Mitsubishi Technical Bulletins, etc.
(2) Product supply (including repair parts) is not available after production is discontinued.

3. Overseas service

Overseas, repairs shall be accepted by Mitsubishi's local overseas FA Center. Note that the repair conditions at each FA Center may differ.

4. Exclusion of loss in opportunity and secondary loss from warranty liability

Regardless of the gratis warranty term, Mitsubishi shall not be liable for compensation of damages caused by any cause found not to be the responsibility of Mitsubishi, loss in opportunity, lost profits incurred to the user by Failures of Mitsubishi products, special damages and secondary damages whether foreseeable or not , compensation for accidents, and compensation for damages to products other than Mitsubishi products, replacement by the user, maintenance of on-site equipment, start-up test run and other tasks.

5. Changes in product specifications

The specifications given in the catalogs, manuals or technical documents are subject to change without prior notice.

6. Product application

(1) In using the Mitsubishi MELSEC programmable logic controller, the usage conditions shall be that the application will not lead to a major accident even if any problem or fault should occur in the programmable logic controller device, and that backup and fail-safe functions are systematically provided outside of the device for any problem or fault.
(2) The Mitsubishi programmable logic controller has been designed and manufactured for applications in general industries, etc. Thus, applications in which the public could be affected such as in nuclear power plants and other power plants operated by respective power companies, and applications in which a special quality assurance system is required, such as for Railway companies or Public service purposes shall be excluded from the programmable logic controller applications.
In addition, applications in which human life or property that could be greatly affected, such as in aircraft, medical applications, incineration and fuel devices, manned transportation, equipment for recreation and amusement, and safety devices, shall also be excluded from the programmable logic controller range of applications.
However, in certain cases, some applications may be possible, providing the user consults their local Mitsubishi representative outlining the special requirements of the project, and providing that all parties concerned agree to the special circumstances, solely at the users discretion.

QnACPU
Programming Manual (Fundamentals)

MODEL	QNA-P(KISO)-E
MODEL CODE	13JF46
IB(NA)-66614-H(0603)MEE	

MITSUBISHI ELECTRIC CORPORATION

HEAD OFFICE : TOKYO BUILDING, 2-7-3 MARUNOUCHI, CHIYODA-KU, TOKYO 100-8310, JAPAN
NAGOYA WORKS : 1-14 , YADA-MINAMI 5-CHOME , HIGASHI-KU, NAGOYA , JAPAN