CHAPTER 4 INTERRUPT FUNCTIONS

4.1 Interrupt Controller

The μPD70325 and 70335 contain an interrupt controller that can control multiple interrupts from 17 sources. The interrupt controller divides a total of 17 interrupt sources—five external and 12 internal sources—into groups for management, and can perform programmable multiprocessing control in group units. Also, one of three interrupt servicing functions; vectored interrupt, register bank switching, and macro service, can be selected according to the interrupt source property.

The number of external interrupt sources can be easily increased by connecting an external μPD71059 or an external interrupt controller equivalent to the μPD71059.

The interrupt request control registers and macro service control registers are used to set up the interrupt controller. Registers are provided for each interrupt source. The interrupt control instructions are listed below.

DI	(Disable Interrupt)
EI	(Enable Interrupt)
RETI	(Return from Interrupt)
RETRBI	(Return from Register Bank Switching Interrupt)
FINT	(Finish Interrupt), which informs the internal interrupt controller of interrupt servicing completion

The interrupt source register and the interrupt priority register are also provided to recognize the interrupt request acknowledgment state.

4.2 Interrupt Sources

There is a total of 17 interrupt sources (five external and 12 internal) for the μPD70325 and 70335. Interrupts from these 17 sources are divided into eight groups and are controlled by the interrupt controller. The configuration of these groups is fixed by hardware. Five of these eight groups (all except NMI, INT, and INTTB) can be specified priority levels from 0 to 7 (0 being the highest) by software. Functions supported by the interrupt controller vary depending upon the interrupt source. Table 4-1 lists these interrupt sources.

Table 4-1. Interrupt Source List (1/2)

Group	Interrupt source	External/ internal	Vector	Macro service	Bank switching	Priority level[Note]			Multiprocessing control
						Setting	Inter-group	Intra-group	
NMI	NMI (Non Maskable Interrupt)	External	2	Not available	Not available	Cannot be set	0	-	Not controlled
External INT	INT (INTerrupt)	External	External input	Not available	Not available	Cannot be set	7	-	Not controlled
Timer interrupt	INTTU0 (INTerrupt from Timer Unit0)	Internal	28	Available	Available	Can be set	1	1	Controlled
	INTTU1 (INTerrupt from Timer Unit1)		29					2	
	INTTU2 (INTerrupt from Timer Unit2)		30					3	
DMA	INTD0 (INTerrupt from DMA channel0)	Internal	20	Not available	Available	Can be set	2	1	Controlled
	INTD1 (INTerrupt from DMA channel1)		21					2	
External	INTP0 (INTerrupt from Peripheral#0)	External	24	Available	Available	Can be set	3	1	Controlled
	INTP1 (INTerrupt from Peripheral#1)		25					2	
	INTP2 (INTerrupt from Peripheral#2)		26					3	

Note The inter-group and intra-group priority levels indicate the acknowledgment sequence when interrupt requests having the same priority level assignments occur at the same time.

Table 4-1. Interrupt Source List (2/2)

Group	Interrupt source	External/ internal	Vector	Macro service	Bank switching	Priority level[Note]			Multiprocessing control
						Setting	Inter-group	Intra-group	
Serial interface channel 0	INTSER0 (INTerrupt from Serial ERror of channel0)	Internal	12	Not available	Available	Can be set	4	1	Controlled
	INTSR0 (INTerrupt from Serial Receiver of channel0)		13	Available				2	
	INTST0 (INTerrupt from Serial Transmitter of channel0)		14	Available				3	
Serial interface channel 1	INTSER1 (INTerrupt from Serial ERror of channel1)	Internal	16	Not available	Available	Can be set	5	1	Controlled
	INTSR1 (INTerrupt from Serial Receiver of channel1)		17	Available				2	
	INTST1 (INTerrupt from Serial Transmitter of channel1)		18	Available				3	
Time base	INTTB (INTerrupt from Time Base counter)	Internal	31	Not available	Not available	Cannot be set (fixed to 7)	6	-	Controlled

Note The inter-group and intra-group priority levels indicate the acknowledgment sequence when interrupt requests having the same priority level assignments occur at the same time.

4.3 Priority Level Control

4.3.1 Multiple interrupt priority level control

Multiple interrupt priority level control is performed in group units for all interrupts except NMI and INT.

Multiple interrupt servicing control is performed during the EI state. Therefore, set the EI using a given interrupt service routine before performing multiple servicing. However, multiprocessing control can also be performed in the DI state for interrupt responses by macro service.

(1) Multiprocessing control

Under multiprocessing control, any interrupt having a higher priority level than the interrupt being serviced is acknowledged, the interrupt being serviced is stopped, and the higher-priority interrupt is then serviced. Any interrupt that has a lower priority level than the interrupt being serviced is held pending. The pending interrupt is acknowledged when the interrupt being serviced terminates as long as the interrupt mask bit in the interrupt control register (provided for each interrupt source) has not been set and the interrupt request flag has not been reset by the current interrupt service routine.

Multiple interrupts from interrupt sources having the same priority level or from within the same group cannot be serviced.

(2) Interrupt responses to all interrupts except NMI, INT, and software interrupts

For interrupt responses to these interrupts, the FINT instruction must be executed at the end of a given interrupt service routine to inform the interrupt controller of the interrupt service routine's completion. If the FINT instruction is not executed, the only subsequent interrupts to be acknowledged are those that have a priority level higher than the interrupt for which the FINT instruction was not executed (see section **4.10 Interrupt Priority Register (ISPR)**).

Figure 4-1 shows the servicing mode for interrupts that are subject to multiprocessing control.

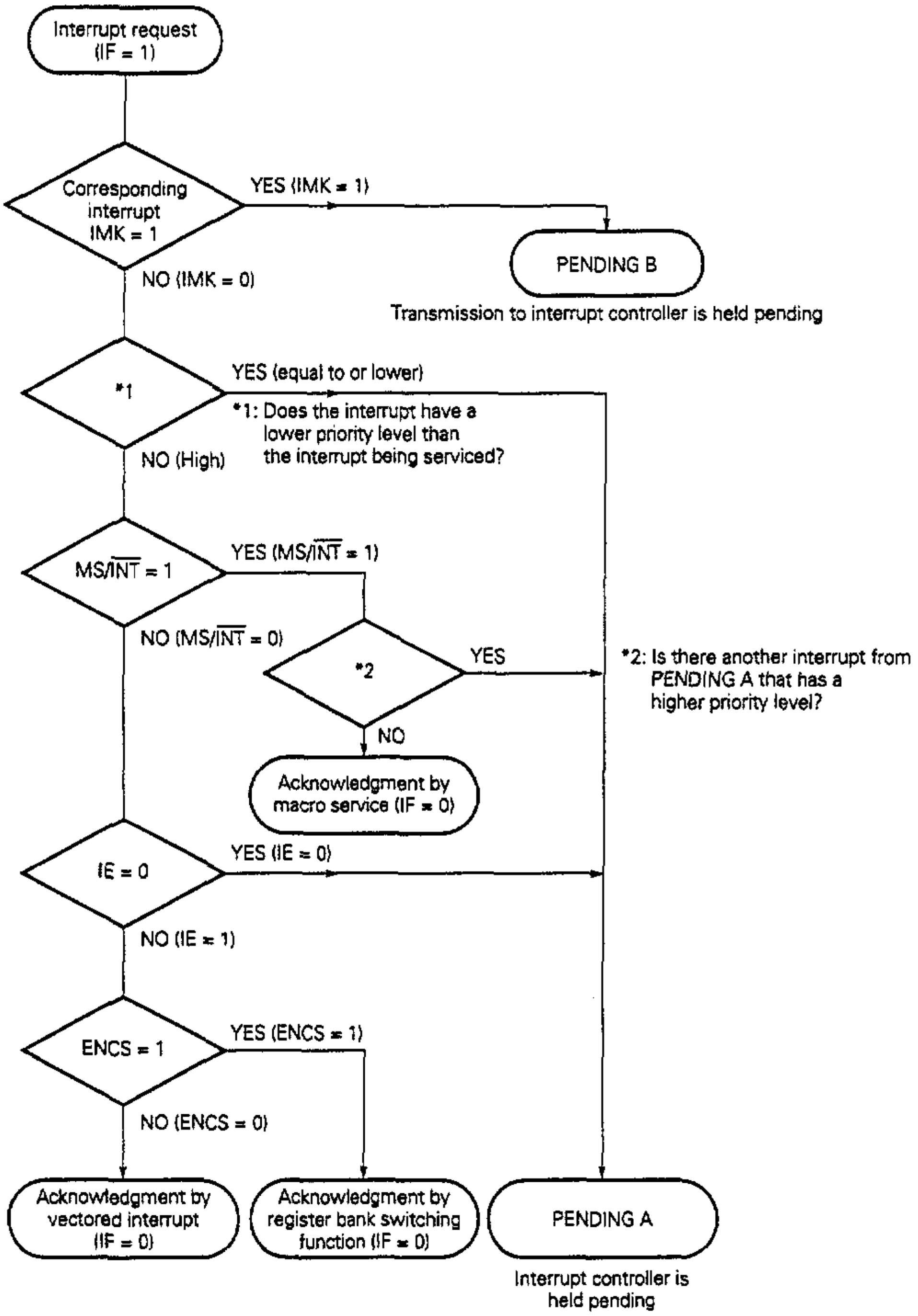
(3) Interrupt responses from NMI and INT

NMI and INT interrupt responses are not subject to multiprocessing control. INT is acknowledged when interrupts are enabled (NMI is always acknowledged).

(4) Setting of priority levels

Eight priority levels from 0 to 7 (0 being the highest) can be set as desired for each interrupt group. The priority level also indicates the number of the selected register bank when using the register bank switching function described below. The priority level is set via the three bits PR0 to PR2 in the interrupt control register that is provided for each interrupt source (the set priority level will not change as long as the settings for PR0 to PR2 remain the same). However, the priority level setting is effective only in the interrupt control register of the interrupt source that has the highest priority level in the interrupt group. This setting is ignored if it is written to any other interrupt control register. When read, the return value is fixed to 7. When a reset signal is input, all priority levels are initialized to 7. See the **Cautions on Interrupt Priority Levels and Servicing Sequence** in section **A.4**.

Figure 4-1. Servicing Mode of Interrupts Subject to Multiple Servicing Control


Interrupt request (IF = 1)
Corresponding interrupt IMK = 1
YES (IMK = 1)
PENDING B
Transmission to interrupt controller is held pending
NO (IMK = 0)
*1
YES (equal to or lower)
*1: Does the interrupt have a lower priority level than the interrupt being serviced?
NO (High)
MS/$\overline{INT}$ = 1
YES (MS/$\overline{INT}$ = 1)
NO (MS/$\overline{INT}$ = 0)
*2
YES
*2: Is there another interrupt from PENDING A that has a higher priority level?
NO
Acknowledgment by macro service (IF = 0)
IE = 0
YES (IE = 0)
NO (IE = 1)
ENCS = 1
YES (ENCS = 1)
NO (ENCS = 0)
Acknowledgment by vectored interrupt (IF = 0)
Acknowledgment by register bank switching function (IF = 0)
PENDING A
Interrupt controller is held pending

4.3.2 Priority level control for simultaneously occurring interrupts

When two or more interrupts occur simultaneously, NMI has the highest priority level for acknowledgment and INT has the lowest. The priority levels of interrupts other than NMI or INT are the same as those of multiple interrupts. Hardware-fixed priority levels are applied to groups that are specified the same priority level. Similarly, priority levels in each group are applied to the interrupts in the group.

Examples of priority level control are listed below.

Examples **1.** If INTSR0 (specified priority level 3) and INTTU2 (specified priority level 6) occur at the same time, *INTSR0 is acknowledged first.*

2. If INTP0 and INTP1 occur at the same time, INTP0 is acknowledged first.

3. If NMI and INTD1 occur at the same time, NMI is acknowledged first.

4. If INTTB and INT occur at the same time, INTTB is acknowledged first.

5. If INTSER1 and INTTU1 (both specified priority level 4) occur at the same time, INTTU1 is acknowledged first.

4.4 Interrupt Requests

(1) Occurrence of interrupt requests

When an interrupt request occurs, IF (bit 7) is set to 1 in the corresponding interrupt request control register. Setting IF to 1 indicates the occurrence of an interrupt request.

Interrupt requests also occur when IF is set by software.

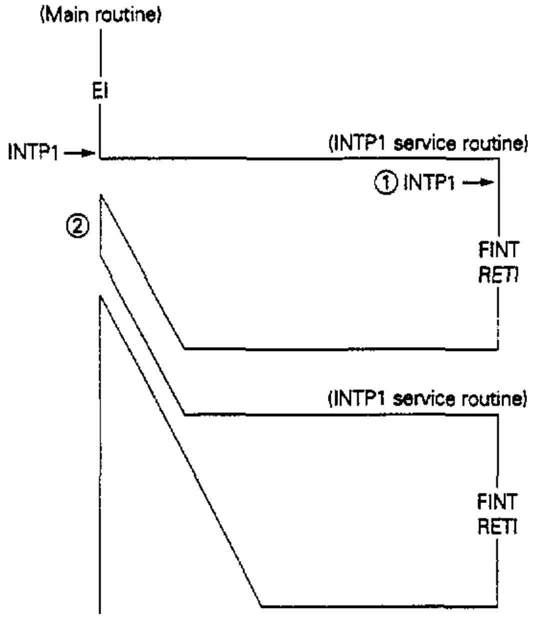
(2) Acknowledgment of interrupt request

When an interrupt request is acknowledged, IF (bit 7) is reset to 0 in the corresponding interrupt request control register. Accordingly, if the same interrupt request occurs again during interrupt servicing, IF is set to 1 and the new interrupt is held pending.

(3) Holding of interrupt requests

Only one interrupt request of the same source can be held pending (see **Figure 4-2**). This means that when IF (bit 7) has been set to 1, if an interrupt request from the same source occurs, the interrupt request is ignored. Interrupt requests that occur when IF is set to 1 are held pending. Any pending interrupt requests are canceled if the software resets IF to 0.

Figure 4-2. Multiple servicing of Same Interrupts



① If another INTP1 for the same interrupt occurs while INTP1 is being serviced or while a FINT instruction is being executed, the second INTP1 is held pending. However, only one INTP1 can be held pending.

② When servicing of the first INTP1 is completed, servicing starts for the INTP1 held pending.

(4) Polling of interrupt requests

To determine the timing of interrupt request occurrences without having an interrupt acknowledged, set the IMK bit in the interrupt control register to 1, mask the interrupt request, and poll the IF bit to detect the occurrence of interrupt requests. However, after detecting the occurrence of interrupt requests, the IF bit must be reset to 0 by software.

4.5 Interrupt Response Modes

The μPD70325 and 70335 each have three interrupt response modes; a vectored interrupt function, register bank switching function, and macro service function. Each of these functions can be selected according to the purpose of the interrupt. The interrupt controller responds to a given interrupt request according to the response mode set in the interrupt control register.

If an interrupt is acknowledged by the vectored interrupt function or register bank switching function, the contents of PC, PS, and PSW are saved using the method corresponding to the selected function. After the PSW is saved, the IE and BRK flags are reset and the DI state holds. Consequently, all interrupts except for those with NMI or macro service responses and single-step interrupts are disabled (all software interrupts except single-step interrupts can occur). (See section **4.11.**)

4.5.1 Vectored interrupts

When a vectored interrupt is acknowledged, the current contents of PSW, PC, and PS are saved on a stack, then one vector is selected from a vector table and the program is executed as an interrupt service routing starting at the address indicated by the vector. All vectors other than INT interrupts are fixed. When an INT interrupt occurs, an interrupt acknowledge cycle is generated and an interrupt vector is read from the data bus (see section **4.7 INT**). Table 4-1 lists interrupt vectors other than INT.

A return from an interrupt is made by executing an RETI instruction. However, an FINT instruction must be executed to make a return from any interrupt except NMI and INT. Whenever a return is made from an interrupt, PC, PS, and PSW are restored from the stack.

The vectored interrupt sequence is shown below (see also **Figure 4-3**).

① Write the PSW value into a temporary register.

② Clear the IE and BRK flags.

③ Read the PC value at the vector address into an internal temporary register.

④ SP ← SP - 2. Write the PSW value onto a stack when an interrupt occurs.

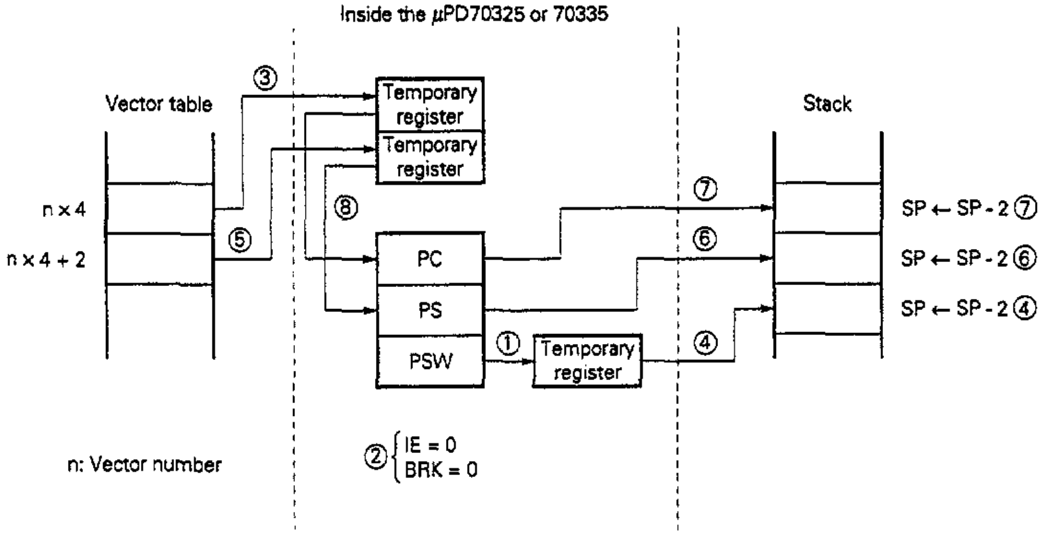
⑤ Read the PS value at the vector address into an internal temporary register.

⑥ SP ← SP - 2. Write the PS value onto a stack when an interrupt occurs.

⑦ SP ← SP - 2. Write the PC value onto a stack when an interrupt occurs.

⑧ Write the values read from the vector address into the PC and PS for a branch.

Figure 4-3. Interrupt Acknowledge Operation



4.5.2 Register bank switching function

The μPD70325 and 70335 map a general purpose register set in internal RAM and can have up to eight register banks. Automatic register bank switching during execution of the BRKCS or TSKSW instruction or during an interrupt response eliminates the need for software's conventional save processing of registers on a stack. This enables high-speed switching of program execution environments.

(1) Register bank switching when an interrupt request occurs

(a) Settings

To use the register bank switching function during an interrupt response, set a value of 1 for the ENCS bit in the interrupt control register provided for each interrupt source. One register bank can be specified for each interrupt group. The register bank number is the same as the multiple interrupt priority level and is specified in bits PR0 to PR2 in the interrupt control register. The register bank number matches the interrupt priority level.

PS and vector PC in the new register bank must be initialized before using the register bank switching function. Initialize SS and SP either before or after register bank switching. If the MOVSPA instruction is executed for initialization after switching, the value before switching is set in SS and SP, and the stack can be continuously used since before register bank switching. Initialize other registers as required. However, do not change PS within the interrupt service routine.

(b) Switching sequence

The register bank switching sequence is executed as shown below (see also **Figure 4-4**).

① Save the contents of PSW to a temporary register.
② Perform register bank switching.
③ Set IE to 0 and BRK to 0.
④ Save the PC contents and the PSW contents that were saved to a temporary register in the PC save area and PSW save area in the new register bank.
⑤ Load the interrupt service routine's start address offset into PC from the vector PC area in the register bank.

The register bank switching is now complete and the interrupt service routine is executed.

(c) Return from register bank switching

To return from a register bank switching interrupt, first execute the FINT instruction (because use of the register bank switching function is limited to interrupts subject to multiple interrupt control) and then execute the RETRBI instruction. When the RETRBI instruction is executed, PC and PSW are restored from the PC and PSW save areas in the register banks as shown in Figure 4-5. (Because they are not restored by the RETI instruction, they normally cannot be returned to the main routine.)

The register bank switching function can be used only for one interrupt in an interrupt group that is specified the same priority level (see section **4.8**).

When register bank switching is performed for several interrupt response modes within an interrupt group specified the same priority level, all must be switched to the same register bank.

Figure 4-4. Register Bank Switching Sequence

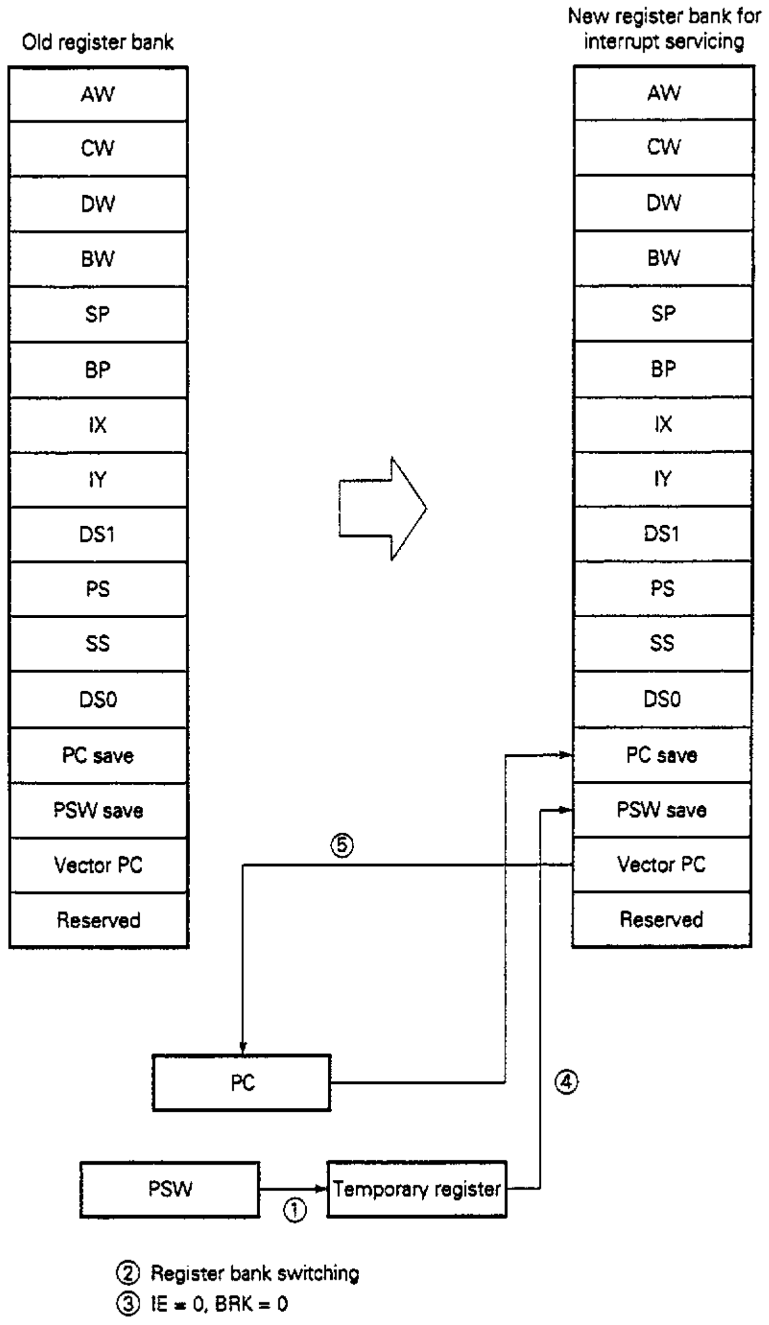
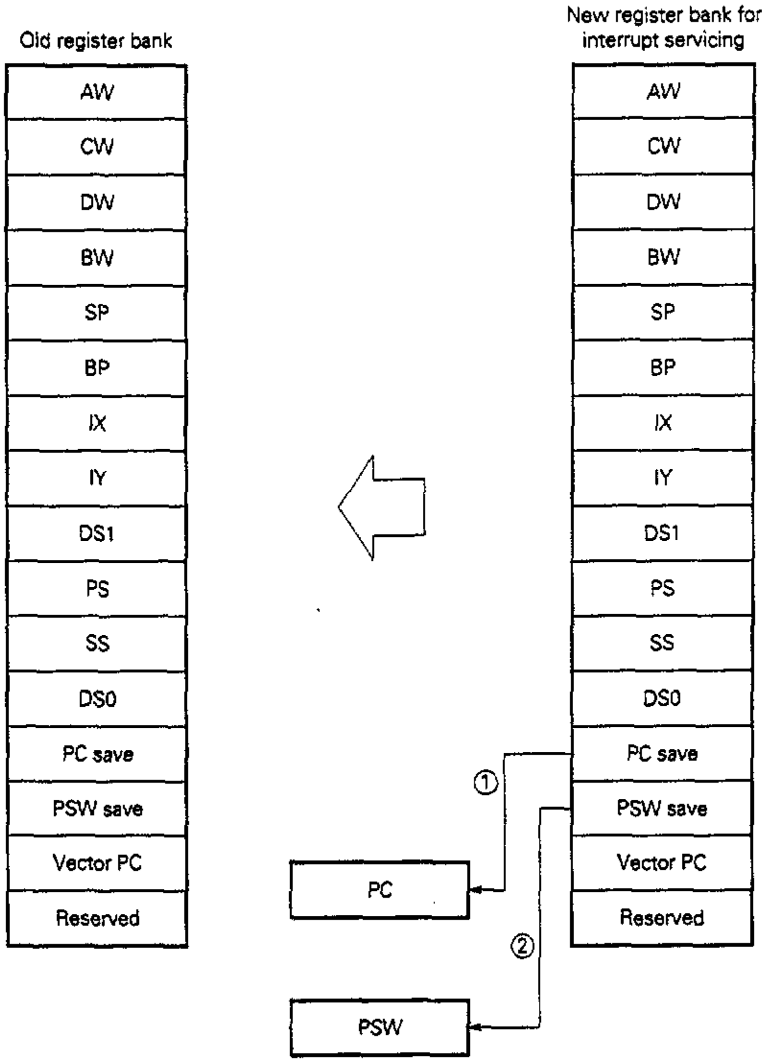


Figure 4-5. Register Bank Return Sequence



(2) Register bank switching by software (BRKCS instruction)

The BRKCS instruction can be executed to perform register bank switching. The BRKCS instruction can also be used as a high-speed subroutine call.

The number of the register bank to be switched to is specified in the low-order three bits of the 16-bit register described in the operand. Next, the PC contents and PSW contents are saved in the PC save area and PSW save area in the new register bank. Then the vector PC that was previously stored in the register bank is loaded into the PC for a branch.

(a) Settings

To execute the BRKCS instruction, the PS and vector PC in the register bank to be selected must be previously initialized. Initialize SS and SP either before or after switching. If the MOVSPA or MOVSPB instruction is executed for initialization, the value before switching is set in SS and SP in the new register bank, and the stack can be used continuously since before register bank switching.

Execute the RETRBI instruction to return from the new register bank. In this case, it is not necessary to use the FINT instruction.

(b) Switching and restore sequence

The register bank switching and restore sequence is the same as for register bank switching when an interrupt request occurs. However, the FINT instruction does not have to be executed for restoring. Restore by executing only the RETRBI instruction.

(3) Register bank switching by software (TSKSW instruction)

The TSKSW instruction can be executed to perform register bank switching. The TSKSW instruction is used for high-speed task switching.

During execution of the TSKSW instruction, the PC contents and PSW contents are first saved in the PC save area and PSW save area in the current register bank (the register bank prior to switching). Next, the number of the register bank to be switched to is specified in the low-order three bits of the 16-bit register described in the operand. Then the PC save area contents that were previously stored in the new register bank is loaded into the PC for a branch.

(a) Settings

To execute the TSKSW instruction, the PS, PC save area, SS, SP, and PSW save area in the register bank to be selected must be previously initialized. If the MOVSPB instruction is executed to initialize SS and SP, the value before switching is set in SS and SP, and the stack can be continuously used.

(b) Switching sequence

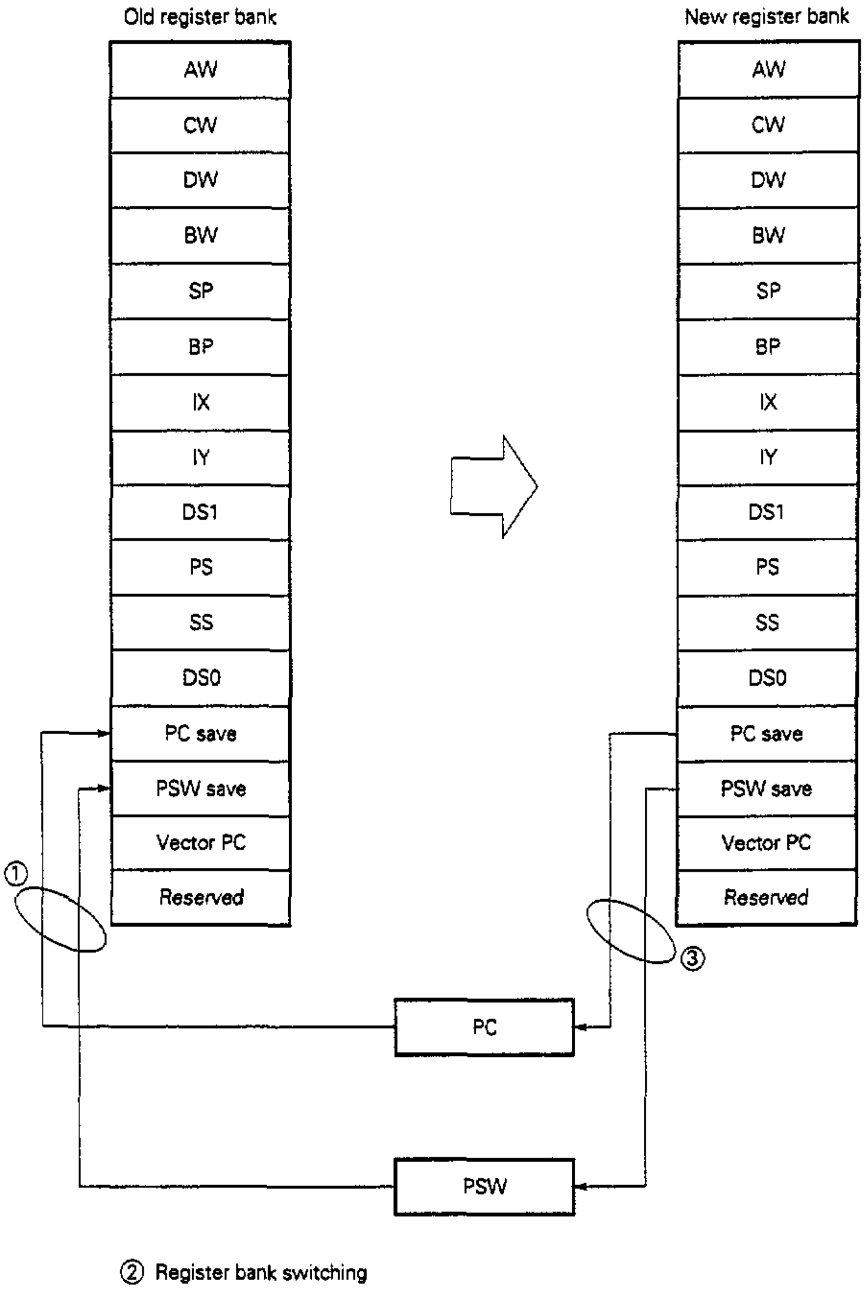
The register bank switching sequence is executed as shown below (see also **Figure 4-6**).

① Save PC and PSW in the PC save area and PSW save area in the current register bank (before switching).

② Perform register bank switching.

③ Load the value in the PSW save area in the new register bank into the PSW and the value in the PC save area into the PC for a branch.

Figure 4-6. Register Bank Switching Sequence by Executing TSKSW Instruction



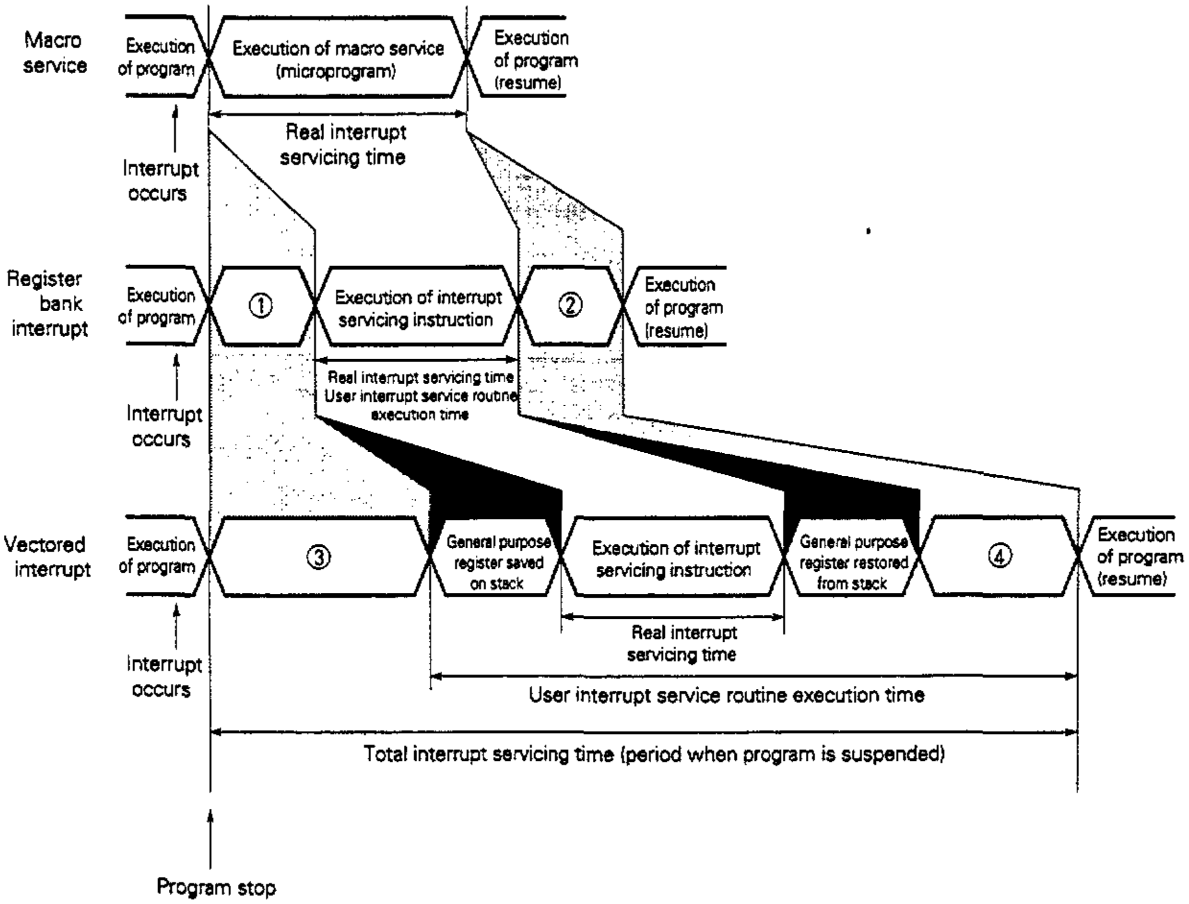
4.5.3 Macro service function

The macro service function transfers data in byte or word units between the special function register area and external memory space when an interrupt request occurs. This function enables simple processing such as simple data transfer without software interrupt servicing and can reduce interrupt servicing overhead (operations such as register saving, initialization, and restore). Processing performed by the macro service function need not be considered by software. Data that is conventionally processed by software in byte units can be processed as one unit of data, thereby enabling efficient programming.

Unlike other interrupt response modes, if the IMK bit (interrupt mask bit) of the interrupt control register provided for each interrupt source is reset and if the $\text{MS}/\overline{\text{INT}}$ bit (macro service enable bit) is set, the macro service function serves regardless of the EI or DI state (see section **4.8**). However, control based on the interrupt priority levels is effective.

When macro service has been executed as many times as specified in the macro service counter (MSC), or when the SFRP value matches the transferred data, the $\text{MS}/\overline{\text{INT}}$ bit is reset and a macro service completion interrupt is generated. This macro service completion interrupt is held pending in the DI state.

Figure 4-7. Interrupt Servicing Efficiency Using Macro Service


Macro service
Execution of program
Execution of macro service (microprogram)
Execution of program (resume)
Real interrupt servicing time
Interrupt occurs
Register bank interrupt
Execution of program
①
Execution of interrupt servicing instruction
②
Execution of program (resume)
Real interrupt servicing time
User interrupt service routine execution time
Interrupt occurs
Vectored interrupt
Execution of program
③
General purpose register saved on stack
Execution of interrupt servicing instruction
General purpose register restored from stack
④
Execution of program (resume)
Interrupt occurs
Real interrupt servicing time
User interrupt service routine execution time
Total interrupt servicing time (period when program is suspended)
Program stop

① Save PC and PSW to a register file, then read register number from vector table for a branch.
② Execute RETRBI instruction. Restore PC and PSW from register file for a branch.
③ Save PC, PS, and PSW on a stack. Read PC and PS from vector table for a branch.
④ Execute RETI instruction. Restore PC, PS, and PSW from stack for a branch.

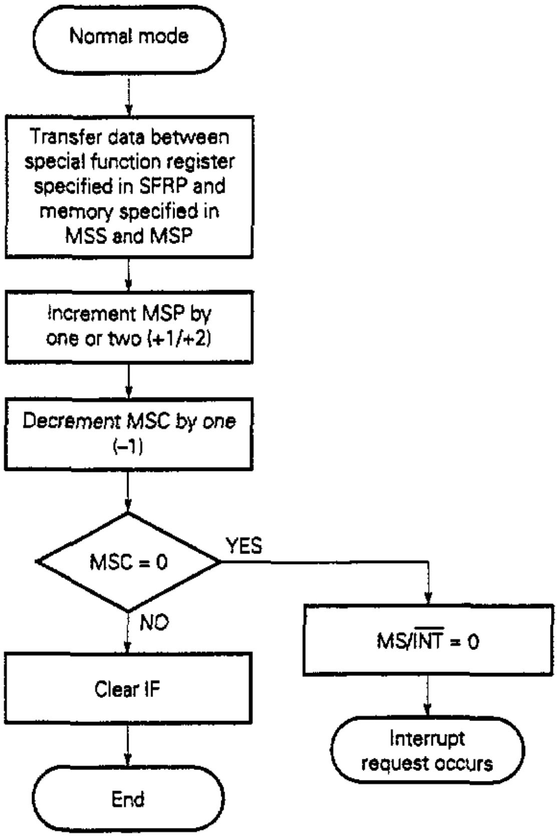
The macro service function contains the following two operation modes.

(1) Normal mode

Whenever an interrupt request occurs, one-byte or one-word data is transferred as many times as specified in the macro service counter (MSC).

Figure 4-8 shows the operation flow in normal mode.

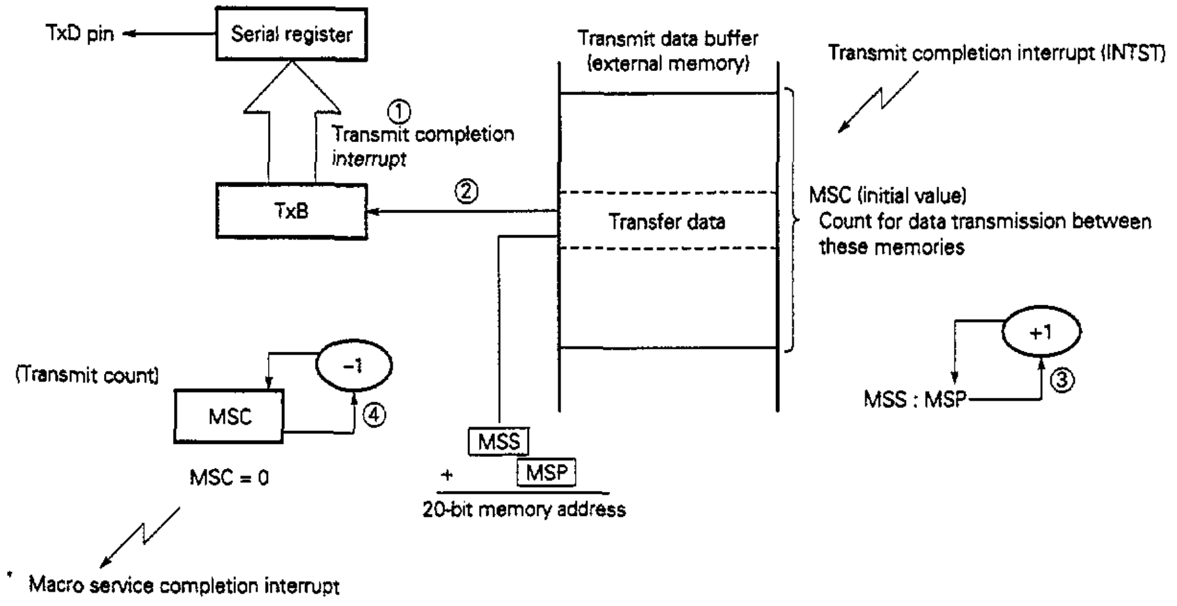
Figure 4-8. Normal Mode Operation Flow


Normal mode
Transfer data between special function register specified in SFRP and memory specified in MSS and MSP
Increment MSP by one or two (+1/+2)
Decrement MSC by one (−1)
MSC = 0
YES
NO
MS/$\overline{\text{INT}}$ = 0
Clear IF
Interrupt request occurs
End

Remark See section **4.5.4** for SFRP, MSS, MSP, and MSC.
See section **4.8** for MS/$\overline{\text{INT}}$.

An example of serial interface transmission is shown below.

Figure 4-9. Example of Serial Interface Transmission



① The TxB register contents are transferred to the serial register, and then a transmit completion interrupt occurs.

② Data is transferred to TxB from the address indicated by "MSS:MSP".

③ MSP is incremented.

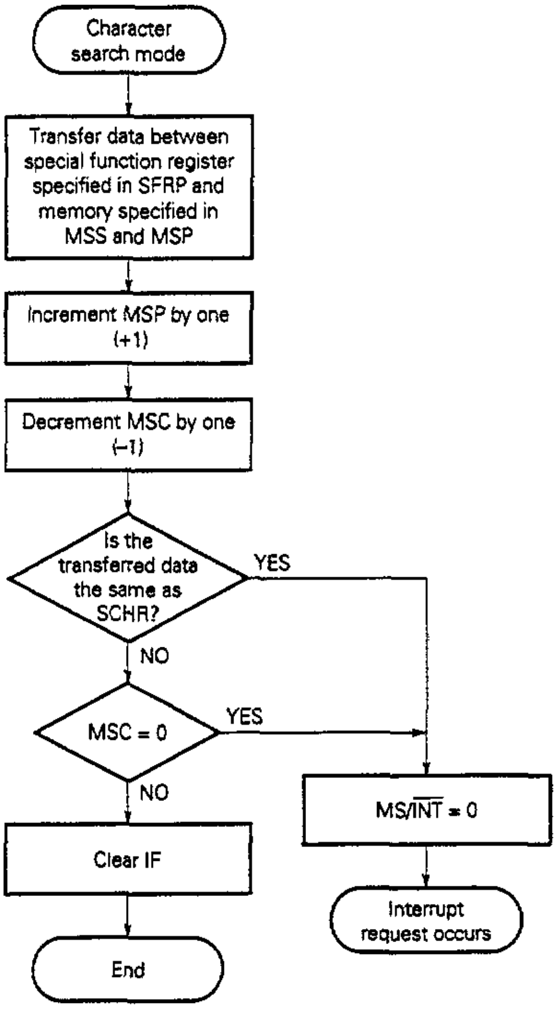
④ MSC is decremented. When MSC is 0, a macro service completion interrupt occurs.

(2) Character search mode

Whenever an interrupt request occurs, one-byte data is transferred the number of times specified in the macro service counter (MSC) or until the transferred data matches the 8-bit data previously specified as the SCHR character data.

Figure 4-10 shows the operation flow in character search mode.

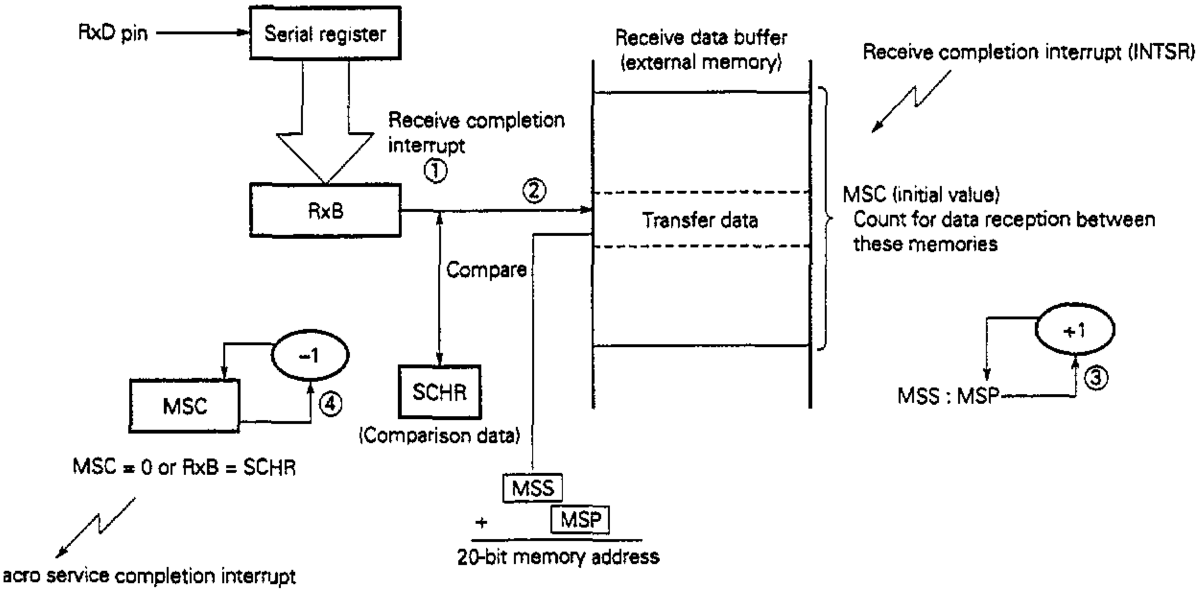
Figure 4-10. Character Search Mode Operation Flow


Character search mode
Transfer data between special function register specified in SFRP and memory specified in MSS and MSP
Increment MSP by one (+1)
Decrement MSC by one (–1)
Is the transferred data the same as SCHR?
YES
NO
MSC = 0
YES
NO
Clear IF
End
MS/$\overline{\text{INT}}$ = 0
Interrupt request occurs

Remark See section **4.5.4** for SFRP, MSS, MSP, MSC, and SCHR.
See section **4.8** for MS/$\overline{\text{INT}}$.

An example of serial interface reception (with end code) is shown below.

Figure 4-11. Example of Serial Interface Reception



① The serial register contents are transferred to the RxB register, and then a receive completion interrupt occurs.
② The contents of RxB are transferred to the address indicated by "MSS:MSP".
③ MSP is incremented.
④ MSC is decremented. When MSC is 0 or when RxB = SCHR, a macro service completion interrupt occurs.

Macro service functions are controlled by the macro service control register provided for each interrupt source subject to macro servicing and by the macro service channel specified in the macro service control register.

(3) Macro service completion interrupt

When MSC is 0 or when the transfer data matches the comparison data (during character search mode only), only the MS/$\overline{\text{INT}}$ bit is reset to 0 in the interrupt control register and the IF bit is not reset to 0. As a result, when in the EI state, an interrupt specified by the ENCS bit occurs after macro service completion. When in the DI state, this interrupt is held pending after macro service completion.

4.5.4 Macro service control register

The macro service control register is an eight-bit register that controls the macro service function. The format of this register is shown below, followed by explanations of each bit function.

7	6	5	4	3	2	1	0
MSM2	MSM1	MSM0	DIR	0	CH2	CH1	CH0

CH0 - CH2 Macro service channel specification bits

A value from 0 to 7 can be specified.

DIR Data transfer direction specification bit

When this bit is set to 0, data is transferred from memory to the special function register. When set to 1, data is transferred from the special function register to memory.

MSM0 - MSM2 Macro service mode specification bits

The operation mode (normal or character search) and the number of transfer data bits (8 or 16) for normal mode are specified by setting MSM0 to MSM2 bits in a certain combination.

MSM2	MSM1	MSM0	Operation mode
0	0	0	Normal mode (8-bit transfer)
0	0	1	Normal mode (16-bit transfer)
1	0	0	Character search mode (8-bit transfer)
Other combinations			Setting Prohibited

The macro service control register is contained in the special function register area. The register can be written/ read by making an 8-bit or 16-bit memory access.

The macro service control register is provided for each interrupt source subject to macro servicing. The interrupt sources subject to macro servicing are timer interrupts (INTTU0 to INTTU2), external interrupts (INTP0 to INTP2), and serial reception and transmission interrupts (INTSR0, INTSR1, INTST0, and INTST1). See the corresponding subject headings for the locations of the macro service control registers for each interrupt source.

Figure 4-12. Macro Service Control Register

7	6	5	4	3	2	1	0	When reset	R/W
MSM2	MSM1	MSM0	DIR	0	CH2	CH1	CH0	Undefined	R/W

Macro service channel specification

Set value for CH0 to CH2	Macro service channels
0-7	Macro service channels from 0 to 7

Data transfer direction specification

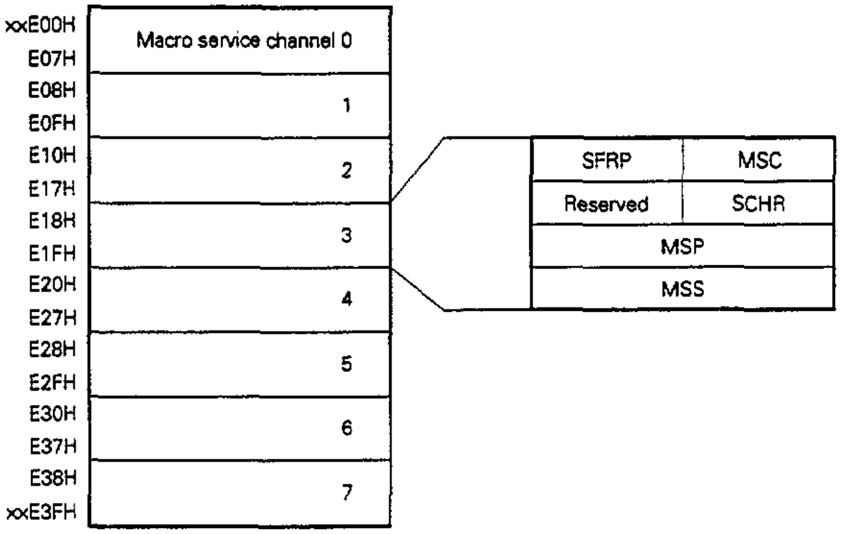
0	From memory to special function register
1	From special function register to memory

Macro service mode specification

MSM2	MSM1	MSM0	Operation mode
0	0	0	Normal mode (8-bit transfer)
0	0	1	Normal mode (16-bit transfer)
1	0	0	Character search mode (8-bit transfer)
Other combinations			Setting prohibited

The macro service channels are specified to on-chip RAM addresses xxE00H to xxE3FH (xx is the IDB register value), as shown in Figure 4-13. The data destination, data source, number of transfer bytes, and comparison character in the macro service mode are set in the macro service channel. Up to eight macro service channels can be used.

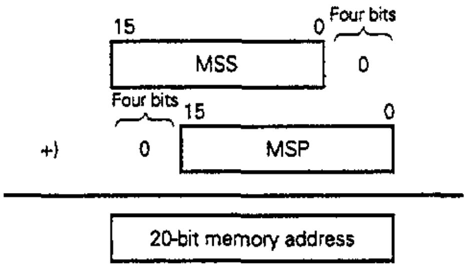
Figure 4-13. Macro Service Channel Configuration



- MSC (+0H) : Macro service transfer count
- SFRP (+1H) : Special function register address offset value. xxF00H + SFRP (xx is the IDB value) is the special function register address.
- SCHR (+2H) : 8-bit data for comparison in character search mode.
- MSP (+4H) : Offset value of memory address for data transfer when macro service is executed.
- MSS (+6H) : Segment value of memory address for data transfer when macro service is executed. The memory address for data transfer is MSS x 16 + MSP.

The value in parentheses is the offset from the start address of the macro service channel.

The macro service memory address for data transfer is indicated by the segment specified in MSS and the offset value from the segment specified in MSP.



Whenever 8-bit or 16-bit data is transferred, the MSC in the macro service channel is decremented by one and the MSP is incremented by one or two. After this, the interrupt request flag is cleared. If 0 is written to MSC, data is transferred 256 times.

Caution Because the register banks and macro service channels are assigned to the same on-chip RAM, do not use register banks when using the macro service function.

4.6 NMI (Non-Maskable Interrupt)

NMI is the highest-priority interrupt that cannot be disabled (masked). This interrupt is edge-detected. The edge direction can be selected by setting the INTM register (a special function register) bit 0 (ESNMI bit) to 1 or 0. If the ESNMI bit is 0, an NMI interrupt occurs when the falling edge is detected and if this bit is 1, an NMI interrupt occurs when the rising edge is detected. Only a vector response can be made to the interrupt, and the vector type is fixed to 2. The NMI input is also used for the P10 pin, and the level can be tested by reading P10. When NMI is acknowledged, IE is set to 0 and interrupts other than NMI are disabled. (However, a macro service interrupt response is acknowledged.)

NMI requests are acknowledged during NMI servicing.

4.7 INT (Interrupt)

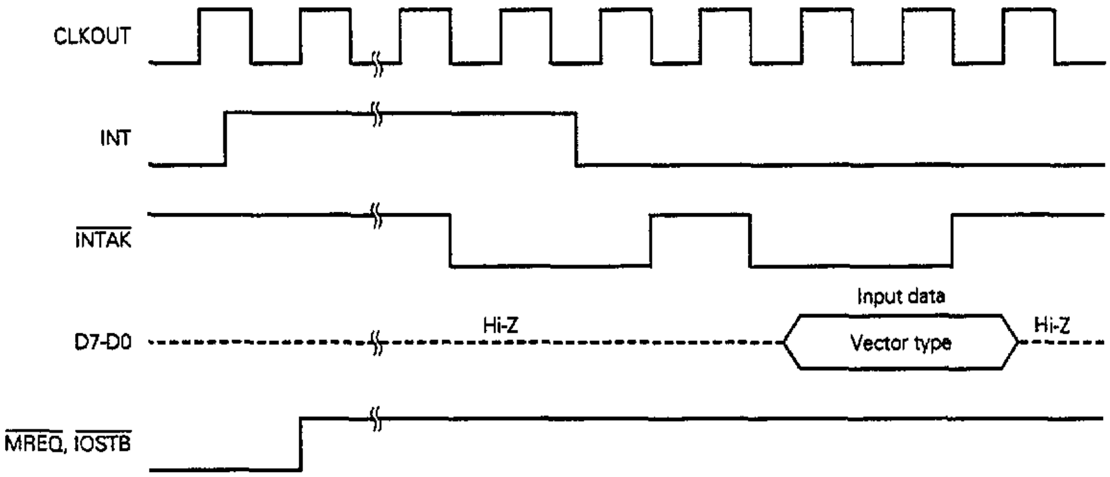
INT is a maskable interrupt. This interrupt is detected by its level (active high). INT is not subject to multiple servicing control by the interrupt controller; and is always acknowledged if interrupts are enabled (IE = 1). However, when a number of interrupts occur at once, the lowest priority level is assigned. Only a vector response can be made to INT, and the vector type is fetched from the data bus during an interrupt acknowledge cycle. The interrupt acknowledge cycle can be confirmed by $\overline{\text{INTAK}}$ output. The INT pin is also used for P14 and $\overline{\text{POLL}}$; it is selected by setting bit 4 of the port 1 mode control register (PMC1), a special function register. Thus, when the INT function is not selected, an INT interrupt does not occur even if interrupts are enabled (IE = 1). $\overline{\text{INTAK}}$ is also used for P13 and $\overline{\text{INTP2}}$, and the function is selected by setting bit 3 of PCM1 (if the $\overline{\text{INTAK}}$ function is not selected, external notification of interrupt acknowledge cycle generation is not possible). Figure 4-14 shows the timing of the interrupt acknowledge cycle.

Hold the INT signal high at least until the first $\overline{\text{INTAK}}$ signal is output.

External interrupt inputs can be expanded to a maximum of 64 by connecting an external μPD71059 or an interrupt controller equivalent to the μPD71059.

When an INT interrupt is acknowledged, interrupts are disabled (IE = 0).

Figure 4-14. INT Interrupt Acknowledge Timing



4.8 Interrupt Request Control Register

The interrupt request control register is an 8-bit register that controls interrupts other than INT and NMI.

The interrupt request control register format is shown below, followed by a description of each bit function.

7	6	5	4	3	2	1	0
IF	IMK	MS/$\overline{\text{INT}}$	ENCS	0	PR2	PR1	PR0

PR0 to PR2 : Interrupt group priority level specification bits

One level from 0 to 7 can be specified. The priority level can be specified only in the interrupt request control register for the interrupt having the highest priority level in the group; if the priority level is specified in any other interrupt request control register, it becomes insignificant (7 is always read). The priority levels in other interrupt request control registers conform to the priority level in the interrupt request control register for the interrupt having the highest priority level in the group.

The priority level also specifies the new register bank when the register bank switching function is executed.

ENCS : Bit specifying whether or not the register bank switching function is used

When this bit is set to 1, the register bank switching function is used; when set to 0, the vectored interrupt function is used.

MS/$\overline{\text{INT}}$: Interrupt response mode selection bit

When this bit is set to 1, the macro service function is used; when set to 0, the vectored interrupt or register bank switching function is used.

IMK : Interrupt mask bit

When this bit is set to 1, the corresponding interrupt is masked; when set to 0, it is not masked.

IF : Bit indicating the corresponding interrupt request

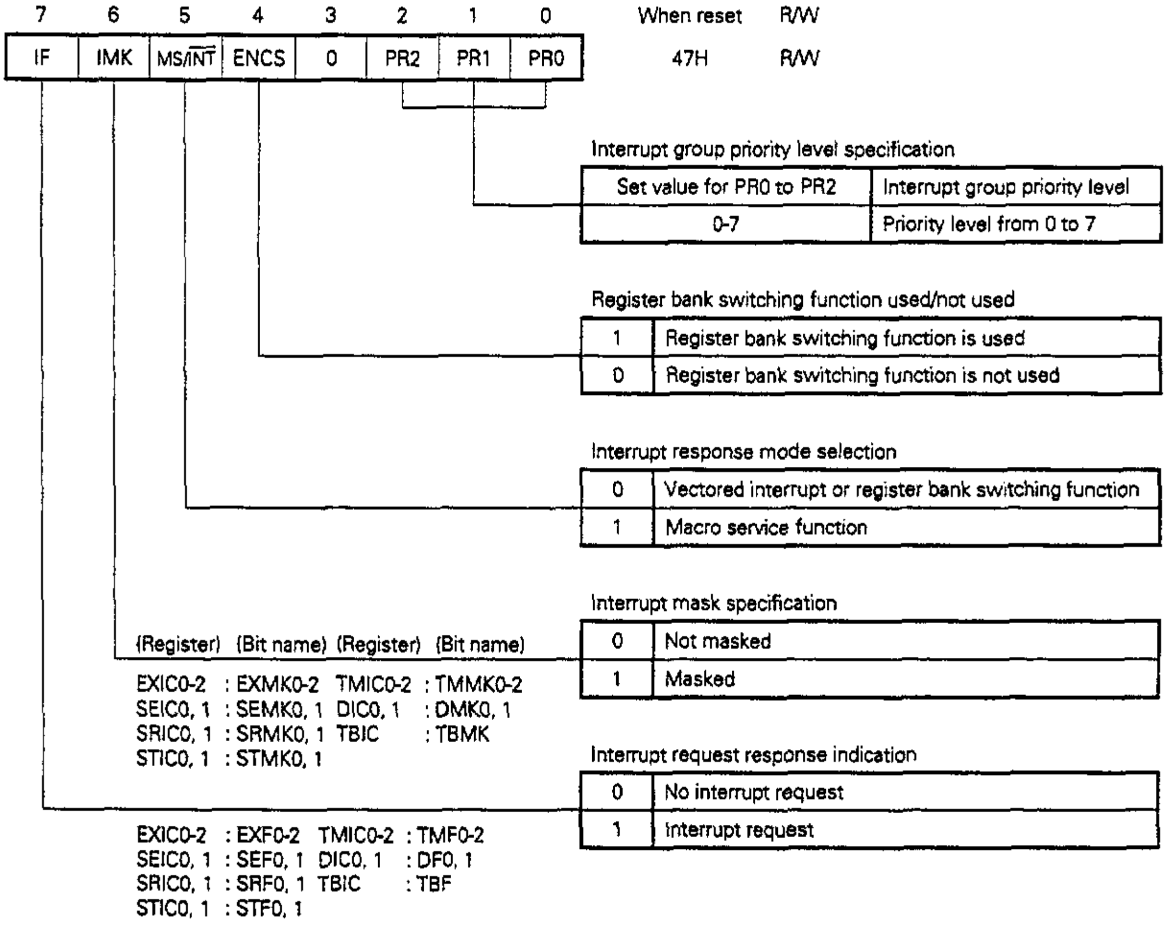
When this bit is set to 1, it indicates that the corresponding interrupt request exists; when set to 0, it indicates that the corresponding interrupt request does not exist. When the corresponding interrupt request occurs, the bit is set to 1. When the interrupt is acknowledged or an instruction such as BTCLR (an additional instruction from the μPD70108/70116) is executed, the bit is reset to 0. This bit is set to 1 whenever an interrupt request occurs, even when interrupts have been masked via the IMK bit.

Caution If the IF bit is set to 0 by a program during interrupt servicing, interrupts will no longer occur.

The interrupt request control register is contained in the special function register area. This register can be written/read by making an 8-bit or 1-bit memory access.

The interrupt request control register is provided for each interrupt source except INT or NMI. See the corresponding subject headings for the locations of the interrupt request control registers for each interrupt source.

Figure 4-15. Interrupt Request Control Register



7	6	5	4	3	2	1	0	When reset	R/W
IF	IMK	MS/$\overline{\text{INT}}$	ENCS	0	PR2	PR1	PR0	47H	R/W

Interrupt group priority level specification

Set value for PR0 to PR2	Interrupt group priority level
0-7	Priority level from 0 to 7

Register bank switching function used/not used

1	Register bank switching function is used
0	Register bank switching function is not used

Interrupt response mode selection

0	Vectored interrupt or register bank switching function
1	Macro service function

Interrupt mask specification

0	Not masked
1	Masked

(Register)	(Bit name)	(Register)	(Bit name)
EXIC0-2	: EXMK0-2	TMIC0-2	: TMMK0-2
SEIC0, 1	: SEMK0, 1	DIC0, 1	: DMK0, 1
SRIC0, 1	: SRMK0, 1	TBIC	: TBMK
STIC0, 1	: STMK0, 1		

Interrupt request response indication

0	No interrupt request
1	Interrupt request

EXIC0-2	: EXF0-2	TMIC0-2	: TMF0-2
SEIC0, 1	: SEF0, 1	DIC0, 1	: DF0, 1
SRIC0, 1	: SRF0, 1	TBIC	: TBF
STIC0, 1	: STF0, 1		

4.9 Interrupt Source Register

This register is an 8-bit register that stores the interrupt type for identifying the interrupt source of an interrupt received. The structure of the IRQS register is shown in Figure 4-16.

The higher three bits are fixed.

Figure 4-16. IRQS

7	6	5	4	3	2	1	0	When reset	R/W
0	0	0	TYPE 4	TYPE 3	TYPE 2	TYPE 1	TYPE 0	Undefined	R

Interrupt type	TYPE 4	TYPE 3	TYPE 2	TYPE 1	TYPE 0	Interrupt source
12(0CH)	0	1	1	0	0	INTSER0
13(0DH)	0	1	1	0	1	INTSR0
14(0EH)	0	1	1	1	0	INTST0
16(10H)	1	0	0	0	0	INTSER1
17(11H)	1	0	0	0	1	INTSR1
18(12H)	1	0	0	1	0	INTST1
20(14H)	1	0	1	0	0	INTD0
21(15H)	1	0	1	0	1	INTD1
24(18H)	1	1	0	0	0	INTP0
25(19H)	1	1	0	0	1	INTP1
26(1AH)	1	1	0	1	0	INTP2
28(1CH)	1	1	1	0	0	INTTU0
29(1DH)	1	1	1	0	1	INTTU1
30(1EH)	1	1	1	1	0	INTTU2
31(1FH)	1	1	1	1	1	INTTB

The interrupt type is specified according to the interrupt source (excluding NMI and INT) subject to priority level control, and has the same value as the vector number.

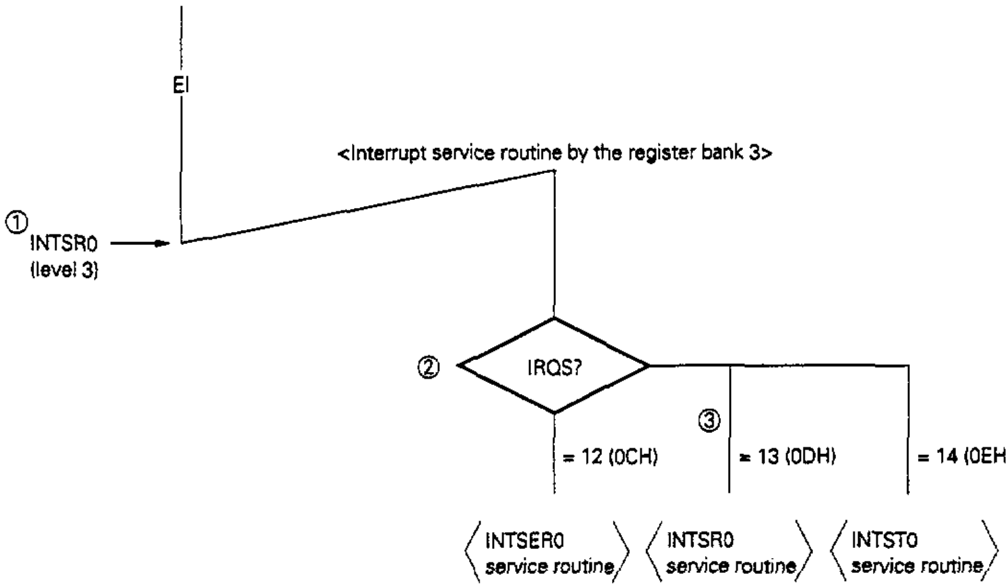
When an interrupt request (excluding a response with macro service) from an interrupt source having the interrupt type (see Figure 4-16.) is received, the interrupt type stored in the IRQS register is updated, and its value is maintained until the next interrupt request is received. It is not changed by interrupts (NMI, INT, etc.) other than the above.

Therefore, if an interrupt request source is to be identified by reading the IRQS register in the interrupt processing using register bank switching, read this register before the next interrupt request can be received by EI instruction, etc.

The IRQS register can be read by 8-bit manipulation memory access.

Next, an example of interrupt servicing by register bank switching using the IRQS is described.

<Main routine>



① INTSR0 (specified priority level 3) occurs in the interrupt enable state (EI), and interrupt servicing (register bank 3 used) starts by register bank switching.

② The IRQS is read in the DI state immediately after the servicing starts. (The IRQS should be read in the DI state in order to prevent the IRQS value from being changed by other interrupts).

③ The servicing routine of INTSR0 is selected according to the read value (13 for this case) and is executed.

4.10 Interrupt Priority Register (ISPR)

The interrupt priority register (ISPR) is an 8-bit register that indicates the multiple interrupt servicing state under the interrupt controller's priority level control. The ISPR register cannot be written to.

Figure 4-17. ISPR

7	6	5	4	3	2	1	0
Priority level 7	Priority level 6	Priority level 5	Priority level 4	Priority level 3	Priority level 2	Priority level 1	Priority level 0

When reset	R/W
00H	R

Interrupt servicing state at each priority level

0	Interrupt request is not acknowledged
1	Interrupt request is acknowledged

Bits 0 to 7 correspond to priority levels 0 to 7. When interrupt request having one priority level is acknowledged, the bit corresponding to the priority level is set to 1.

The least significant bit (corresponding to the highest priority level) among the bits set to 1 is reset to 0 by executing one FINT instruction.

When one of the bits 0 to 7 is set to 1, any interrupt request having a priority level lower than the priority level corresponding to the bit is not acknowledged and is held pending.

Any interrupt request having a priority level higher than the priority level corresponding to this bit is acknowledged, the current interrupt is stopped and held pending.

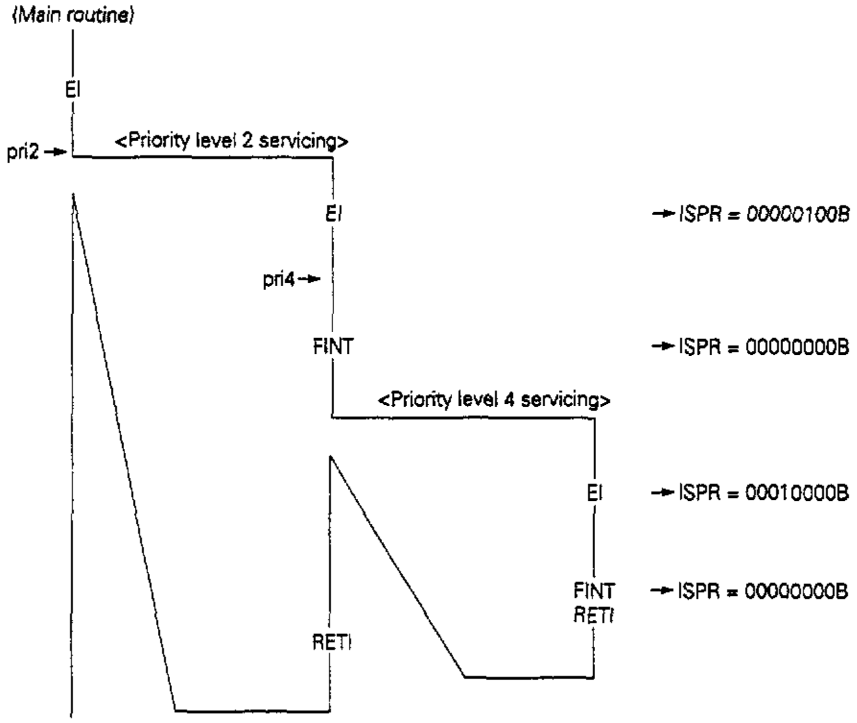
The ISPR register is contained in the special function register area and can be read by making an 8-bit memory access.

When $\overline{\text{RESET}}$ is asserted, the ISPR contents are initialized to 00H.

Cautions 1. Unless an FINT instruction is executed, the corresponding bit will not be reset to 0 when the servicing ends and, consequently, interrupts with lower priority levels cannot be acknowledged.

2. If an FINT instruction is used without being immediately followed by an RETI or RETRBI instruction, priority level control cannot be made. In such cases, control must come from the application.

Figure 4-18. ISPR States



4.11 External Interrupts

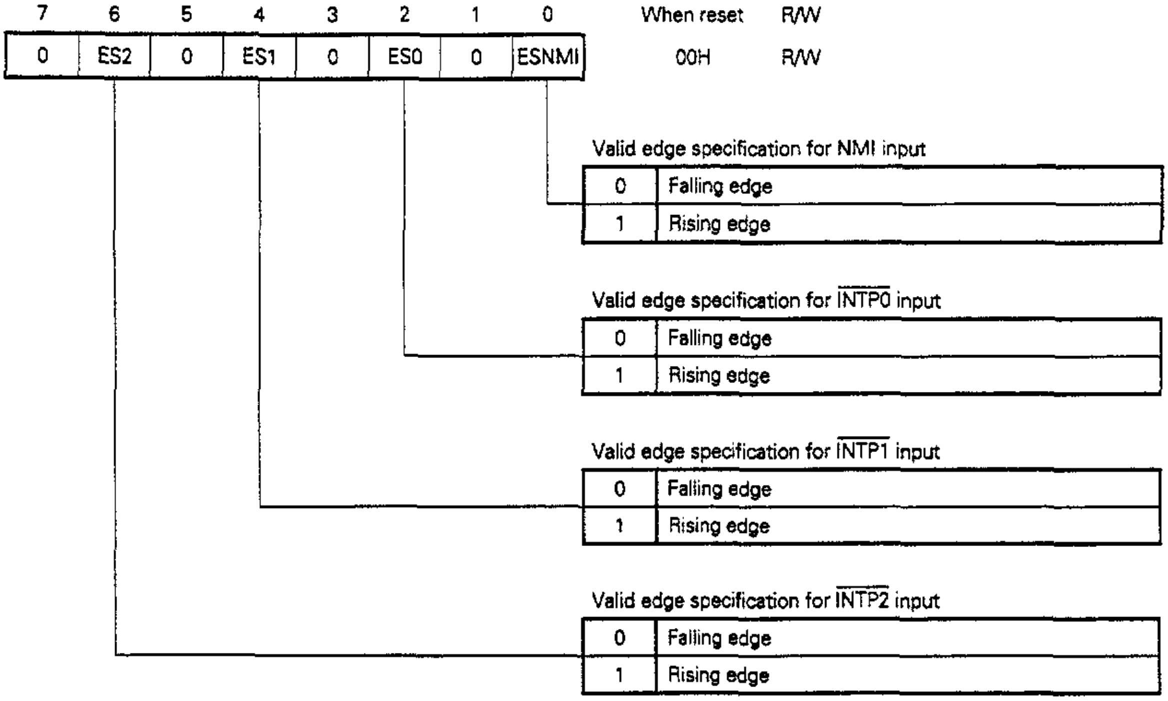
There are five sources of external interrupt requests. INT is level-detected and external interrupts other than INT (NMI and INTP0 to INTP2) are edge-detected. For each of the external interrupts other than INT that are edge-detected, the valid edge can be specified in the external interrupt mode register (INTM), a special function register.

4.11.1 External interrupt mode register (INTM)

The external interrupt mode register (INTM) is an 8-bit register that specifies the valid edge for external interrupt requests. The edge-detected interrupts are NMI and INTP0 to INTP2. The valid edges for these interrupts are specified in the INTM register.

Figure 4-19 shows the INTM register format and bit functions.

Figure 4-19. INTM



7	6	5	4	3	2	1	0	When reset	R/W
0	ES2	0	ES1	0	ES0	0	ESNMI	00H	R/W

Valid edge specification for NMI input

0	Falling edge
1	Rising edge

Valid edge specification for $\overline{\text{INTP0}}$ input

0	Falling edge
1	Rising edge

Valid edge specification for $\overline{\text{INTP1}}$ input

0	Falling edge
1	Rising edge

Valid edge specification for $\overline{\text{INTP2}}$ input

0	Falling edge
1	Rising edge

The INTM register is contained in the special function register area. It can be written/read by making an 8-bit or 1-bit memory access.

When $\overline{\text{RESET}}$ is asserted, the INTM register contents are initialized to 00H.

4.11.2 External interrupt request control registers (EXIC0 to EXIC2)

The EXICn registers (n = 0 to 2) are 8-bit registers that control interrupt requests (EXF0 to EXF2) occurring from the three external interrupt request pins ($\overline{\text{INTP0}}$ to $\overline{\text{INTP2}}$).

These three interrupt requests make up one group as external interrupt requests, and the group's priority level is programmable. Within the group, the priority levels are fixed as follows.

EXF0 >EXF1 > EXF2

Figure 4-20. EXIC0, EXIC1, and EXIC2

	7	6	5	4	3	2	1	0
EXIC0	*EXF0*	*EXMK0*	*MS/$\overline{\text{INT}}$*	*ENCS*	*0*	*PR2*	*PR1*	*PR0*
EXIC1	EXF1	EXMK1	MS/$\overline{\text{INT}}$	ENCS	0	1	1	1
EXIC2	EXF2	EXMK2	MS/$\overline{\text{INT}}$	ENCS	0	1	1	1

Caution Bits 2 to 0 of the EXIC1 and EXIC2 registers are fixed to 1.
The interrupt request priority levels in the EXIC1 and EXIC2 registers conform to the settings for bits PR2 to PR0 in the EXIC0 register.

*See section **4.8 Interrupt Request Control Register** for the description of EXICn register's bits.*

The EXICn register can be read/written by executing an 8-bit or 1-bit memory access. In this case, one wait cycle is automatically inserted.

When $\overline{\text{RESET}}$ is asserted, the register contents are reset to 47H.

4.11.3 External interrupt macro service control registers (EMS0 to EMS2)

The EMSn registers (n = 0 to 2) are 8-bit registers that control macro services started when any of three interrupt requests occur from external interrupt request pins ($\overline{\text{INTP0}}$ to $\overline{\text{INTP2}}$).

The EMS0 register controls the macro service started via the EXF0 flag.

The EMS1 register controls the macro service started via the EXF1 flag and the EMS2 register controls the macro service started via the EXF2 flag.

The EMSn register can be read/written by executing an 8-bit or 1-bit memory access. In this case, one wait cycle is inserted.

Figure 4-21. EMS0, EMS1, and EMS2

7	6	5	4	3	2	1	0
MSM2	MSM1	MSM0	DIR	0	CH2	CH1	CH0

See section **4.5.4 Macro service control register** for description of the EMSn register's bits.

4.12 Software Interrupts

The μPD70325 and 70335 each contain nine types of software interrupts (see **Table 4-2**). Six of the software interrupts are compatible with the μPD70108 and 70116 (however, emulation mode interrupts are not available). The other three interrupts are unique to the μPD70325 and 70335.

The vectors of the interrupts are predefined.

Any interrupt other than the BRK interrupt (single step interrupt) is always acknowledged if the interrupt occurrence condition becomes true (it takes precedence over hardware interrupts). The BRK flag interrupt occurs when BRK *= 1 regardless of the hardware or software. When the interrupt is acknowledged, the BRK is automatically reset,* and therefore the interrupt priority level is lower than other hardware and software interrupts, and another BRK flag interrupt does not occur during the interrupt servicing.

Table 4-2. Software Interrupts

Interrupt source	Vector	Priority level
DIVU divide error	0	1
DIV divide error		
CHKIND boundary over	5	
BRKV	4	
BRK 3	3	
BRK imm8	32-255	
BRK flag (single step)	1	2
I/O instruction ($\overline{\text{IBRK}}$ flag)	19	1
FPO instruction	7	
BRKCS instruction	-	

4.12.1 General software interrupts

The execution sequence for acknowledging software interrupts other than I/O instruction interrupts or FPO instruction interrupts is the same as that of vectored interrupts. That is, address information for the next instruction and the PSW are saved on a stack, IE and BRK are both set to 0, and the vector contents are loaded into the PS and PC.

The software interrupts are described below.

(1) DIVU and DIV divide errors

A software interrupt occurs whenever the quotient overflows while executing a divide instruction.

(2) CHKIND boundary over

When the CHKIND instruction is executed to check whether or not the index value exceeds the predefined array boundaries, a software interrupt occurs if it is determined that the index value exceeds the boundaries.

(3) BRKV

A software interrupt occurs when the overflow flag (V) is set during execution of the BRKV instruction.

(4) BRK 3

A software interrupt occurs when the BRK 3 instruction is executed.

(5) BRK imm8

A software interrupt occurs when the BRK imm8 instruction is executed.

(6) BRK flag (single step)

If BRK is set to 1, a software interrupt occurs whenever one instruction is executed. If a repeat prefix is added, it does not occur until the loop is exited.

4.12.2 I/O instruction interrupts

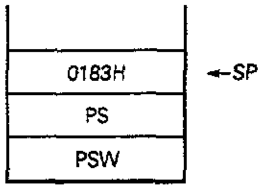
If an attempt is made to execute an I/O instruction when $\overline{\text{IBRK}}$ = 0, an interrupt occurs. When the interrupt is acknowledged, address information saved on a stack becomes the address where the I/O instruction is placed, unlike general software interrupts (see section **4.12.1**). If a prefix is added to the I/O instruction, the address information becomes the address where the prefix is placed. Other operations are the same as general software interrupts. When control is returned from the I/O instruction interrupt, the PC value in the stack must be adjusted for a normal return.

If the address information saved on the stack is made the instruction's starting address, the software can be used to determine specifically which instruction was attempted when the interrupt occurred. This function enables easy migration of programs formerly used with the *μPD70108 and 70116.*

The PSW contents immediately prior to the interrupt occurrence are saved on the stack, then IE = BRK = 0 and $\overline{\text{IBRK}}$ = 1 are set automatically. By setting $\overline{\text{IBRK}}$ = 1, any I/O instruction is executed as an I/O instruction during interrupt servicing. When control is returned from the interrupt, $\overline{\text{IBRK}}$ is automatically reset to 0.

Example 1. Without prefix

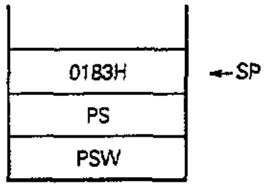
```
Address   Instruction
            :
0183H     IN   AW, DW
            :
```



Stack

Example 2. With prefix

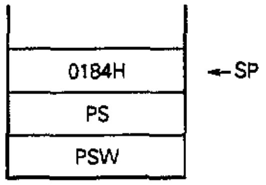
```
Address   Instruction
            :
0183H     REP
0184H     INM
            :
```



Stack

Reference General software interrupt

```
Address   Instruction
            :
0183H     BRK 3
            :
```



Stack

4.12.3 FPO instruction interrupt

Because the μPD70325 and 70335 differ from the μPD70108 and μPD70116 in their external bus structure, a floating-point math co-processor for the μPD70108 and 70116 cannot be connected. Therefore, an interrupt is generated to emulate the operation of the FPO instruction when an attempt is made to execute the FPO instruction for the coprocessor. The PC value saved on the stack becomes the starting address of the FPO instruction emulated by this interrupt (when a prefix is added, the starting address of the prefix), as is the case for I/O instruction interrupts (see section **4.12.2 I/O instruction interrupts**). Accordingly, the FPO instruction can be decoded for emulation by software. When control is returned from the FPO instruction interrupt, the PC value saved on the stack must be adjusted as with I/O instruction interrupts.

4.13 If Interrupt Requests cannot be Acknowledged

If interrupt requests cannot be acknowledged, check the following items.

- Confirm that the IF bit in the interrupt request control register is set to 1. If it has been reset to 0, check whether or not it was replaced (written over) by software.
- Confirm that the interrupt request control register's IMK bit is set to 1.
- Confirm that the ISPR register's bit having the highest priority level is set to 1.
- Confirm that the PSW's IE flag is in the DI state.

4.14 Timing at which an Interrupt cannot be Acknowledged

No interrupt can be acknowledged between each of the following instructions and the next instruction.

(i) **sreg manipulating instructions**
MOV sreg, reg16 ; MOV sreg, mem16 ; POP sreg ; POP PSW ; MOVSPB

(ii) **Prefix instructions**
PS:, SS:, DS0:, DS1:, REPC, REPNC, REP, REPE, REPZ, REPNE, REPNZ, BUSLOCK

(iii) **EI, RETI, DI**

(iv) **FINT**

Each interrupt except INT that occurs at a timing where no interrupt can be acknowledged will be acknowledged if it can be acknowledged after termination of the next instruction's execution.

4.15 Interrupt Servicing during Execution of Block Servicing Instruction

An interrupt is acknowledged during execution of a block servicing (transfer, comparison, retrieval, or I/O) instruction.

The interrupt is acknowledged at the termination of one servicing instruction's execution. At that time, the address saved on the stack or the PC save area in the register bank automatically becomes the top of the instruction containing the prefix. The incomplete block servicing can be resumed by re-executing from the prefix when returning from the interrupt.

Theoretically, up to three types of prefixes including repeat prefixes can be added to the block servicing instruction. The μPD70325 and 70335 enable the user to determine which type of block servicing instruction with prefix was being executed when the interrupt was acknowledged. The PC value is automatically decremented and saved in an area such as the stack according to the prefix addition state.

Example REP
MOVBK SS:SRC BLK, DES BLOCK

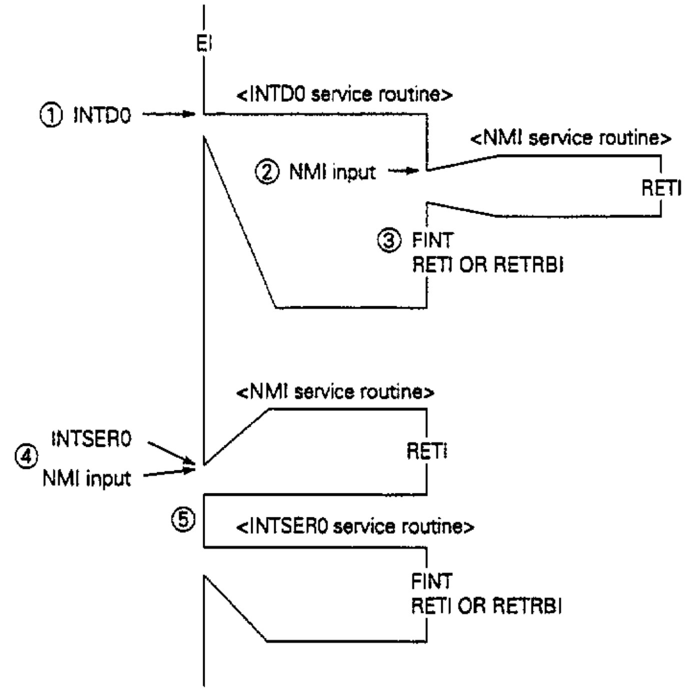
4.16 How Interrupts are Acknowledged

(1) NMI

o Is not masked by software.

o Takes precedence over all other interrupts.

<Main routine>



① INTD0 (specified priority level 2) occurs in the interrupt enable state (EI) and INTD0 servicing is started.

② Interrupts are disabled (DI) during INTD0 servicing. When NMI occurs, INTD0 servicing is disabled and NMI servicing is started.

③ When NMI servicing is completed, previously disabled INTD0 servicing is resumed.

④ When INTSER0 (specified priority level 4) and NMI occur at the same time, the NMI which takes precedence over all other interrupts is acknowledged and NMI servicing is started.

⑤ When the NMI servicing is completed, the pending INTSER0 is serviced.

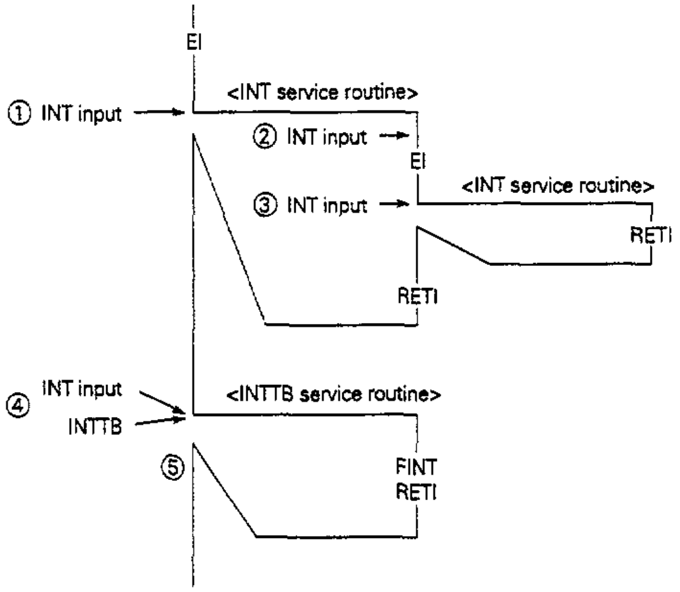
(2) INT

o Is always acknowledged if EI is set to 1 (interrupt enable state).

o Is specified the lowest priority level when multiple interrupts occur.

o Is not subject to the multiple interrupt servicing controller.

<Main routine>



① *When INT occurs in the interrupt enable state (EI), INT servicing is started.*

② When the INT is acknowledged, interrupts are automatically disabled (DI). If another INT occurs, it is not acknowledged.

③ Even when EI is set to 1 (interrupt enable state) during INT servicing, multiple servicing of INT is performed if another INT occurs.

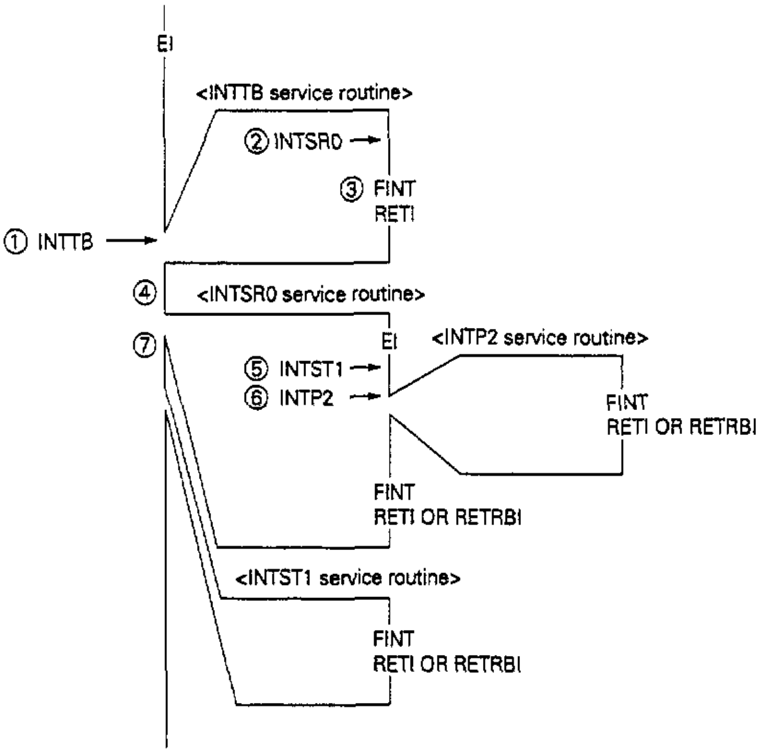
④ When INT and INTTB (specified priority level 6) occur at the same time, the INT which has the lowest priority level is not acknowledged, and INTTB servicing is started.

⑤ If INT is inactive after INTTB servicing is completed, the INT interrupt is not acknowledged.

(3) Interrupts subject to multiprocessing control

o Multiple servicing is performing according to the priority levels.

<Main routine>



① INTTB (specified priority level 6) occurs in the interrupt enable state (EI) and INTTB servicing is started.

② When the INTTB is acknowledged, interrupts are disabled (DI). Although INTSR0 (specified priority level 4) occurs and is higher than the INTTB priority level, it is not acknowledged.

③ The FINT instruction must be executed for interrupts subject to priority level control when returning control from the interrupt.

④ When INTTB servicing is completed and interrupts are enabled (EI), pending INTSR0 servicing is started.

⑤ INTST1 (specified priority level 5) occurs in the interrupt enable state (EI). Since INTSR0 interrupt servicing is being performed and INTST1 has a lower priority level than INTSR0, the INTST1 is not acknowledged.

⑥ INTP2 (specified priority level 3) occurs. Because it has a higher priority level than INTSR0, INTP2 is acknowledged and INTP2 servicing is started.

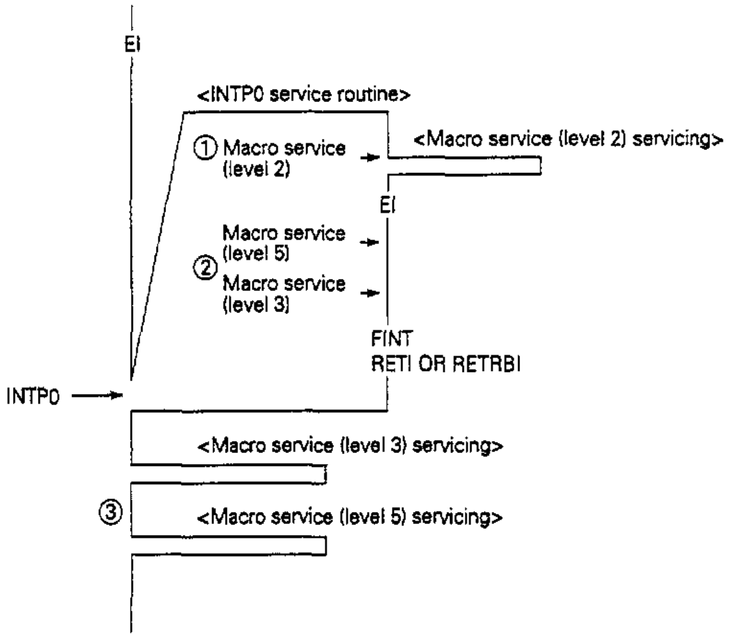
⑦ INTP2 servicing is completed. When pending INTSR0 servicing is also completed and a return is made, INTST1 is acknowledged.

(4) Macro service interrupt

o Is acknowledged regardless of the interrupt enable (EI) or disable (DI) state.

o Multiple servicing is performed according to the priority levels as with the interrupts subject to multiple servicing control.

<Main routine>


EI
<INTP0 service routine>
① Macro service (level 2)
<Macro service (level 2) servicing>
EI
Macro service (level 5)
②
Macro service (level 3)
FINT
RETI OR RETRBI
INTP0
<Macro service (level 3) servicing>
③
<Macro service (level 5) servicing>

① When a macro service interrupt (specified priority level 2) occurs during servicing of INTP0 (specified priority level 3), macro service servicing is performed even though interrupts are disabled (DI).

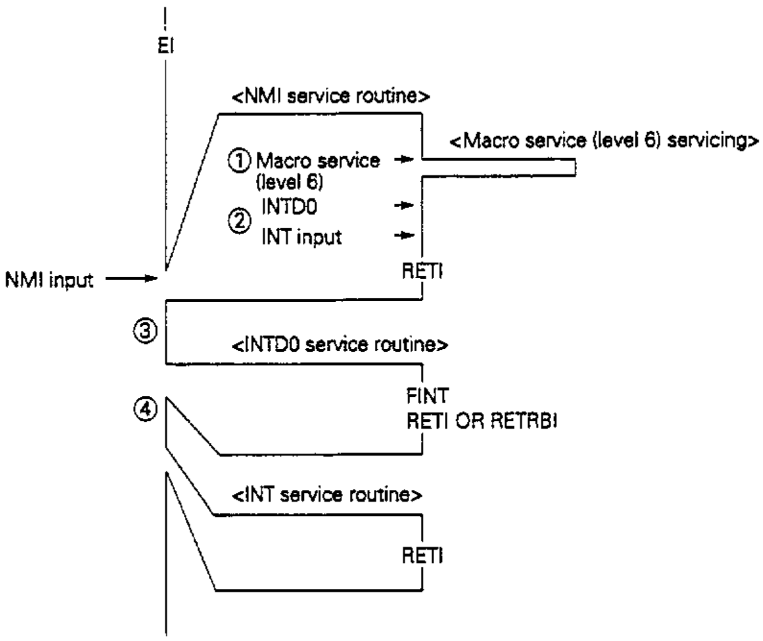
② Macro service interrupts (specified priority levels 5 and 3) occur during servicing of INTP0 placed in the interrupt enable state (EI). The macro service interrupt priority levels are lower than the INTP0 priority level, and therefore interrupts are held pending.

③ When the INTP0 servicing is completed, the pending macro service interrupts are acknowledged according to their priority levels.

(5) Macro service interrupt priority levels for other interrupts

o A macro service interrupt is also acknowledged during NMI servicing.

<Main routine>



① Even if interrupts are disabled (DI) during NMI servicing, macro service servicing is performed when a macro service interrupt occurs.

② When interrupts are disabled (DI) during NMI servicing, even if an interrupt subject to multiple servicing control (INTD0) or INT occurs, it is not acknowledged.

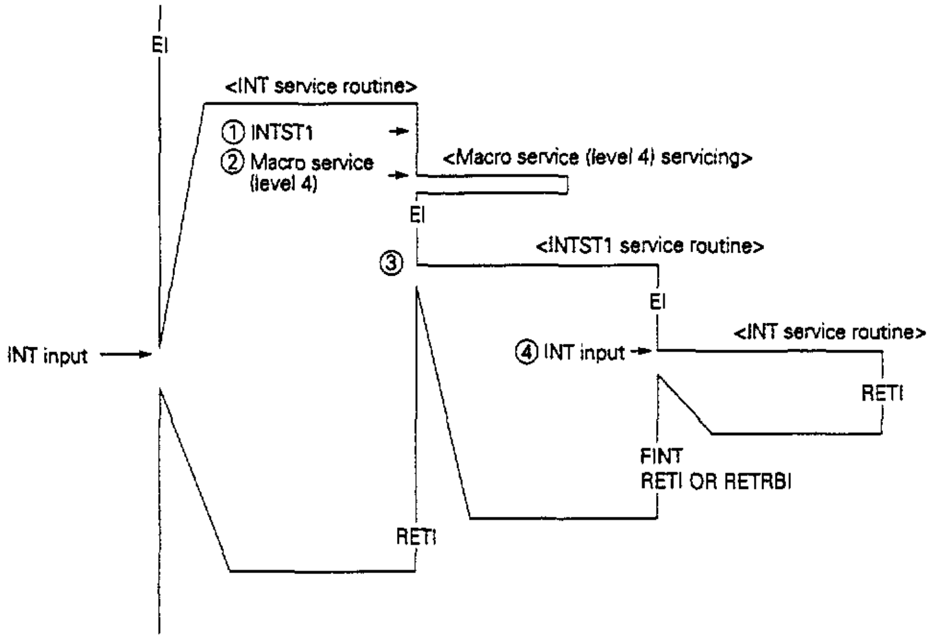
③ When the NMI servicing is completed, the pending INTD0 is acknowledged.

④ If INT is active at the completion of INTD0 servicing, INT is acknowledged.

(6) Multiple interrupt servicing

o INT and other interrupts subject to multiple servicing control, and macro service interrupts

<Main routine>



① Because INT is acknowledged and interrupts are disabled (DI), INTST1 (specified priority level 5) occurs, but is not acknowledged.

② When a macro service interrupt (specified priority level 4) occurs, macro service servicing is performed.

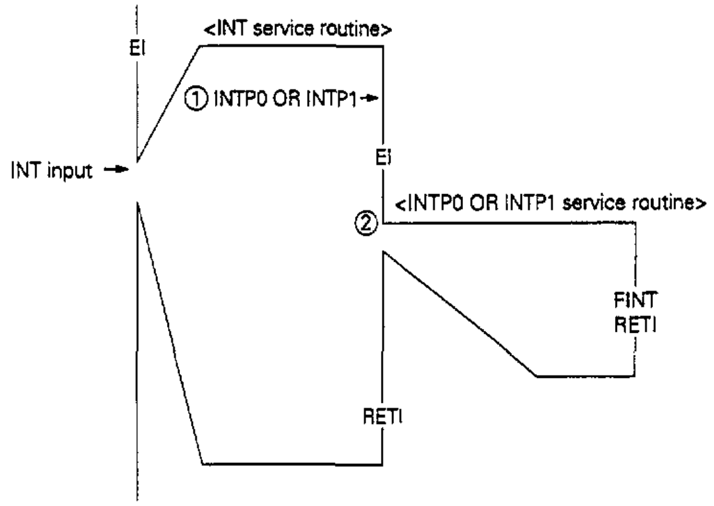
③ When EI is set to 1 (interrupt enable state), pending INTST1 servicing is started.

④ If interrupts are enabled (EI) during INTST1 servicing, INT servicing is started when INT occurs.

(7) Priority levels of external interrupts

- o If an external interrupt for INTP0 or INTP1 input occurs during servicing of an INT interrupt service routine, the interrupt is acknowledged if the interrupt flag is in the EI state.

<Main routine>

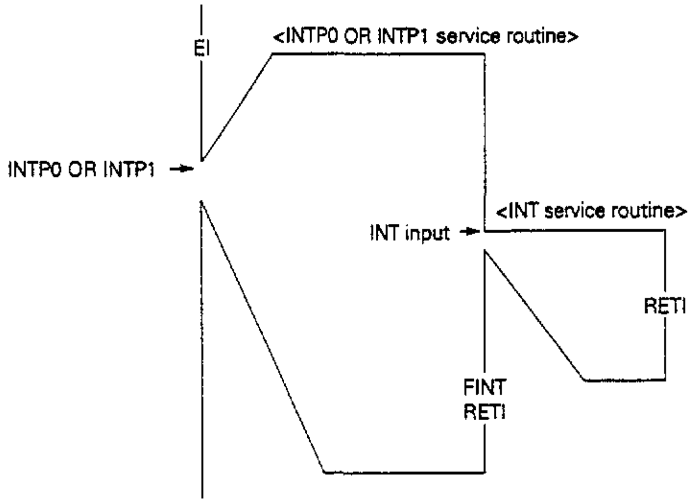


① Because the INT is acknowledged and the interrupt disable state (DI) is in effect, if interrupt requests for INTP0 or INTP1 occur they are not acknowledged.

② When the interrupt flag is in the enable state (EI), servicing is started for the pending INTP0 or INTP1 interrupt.

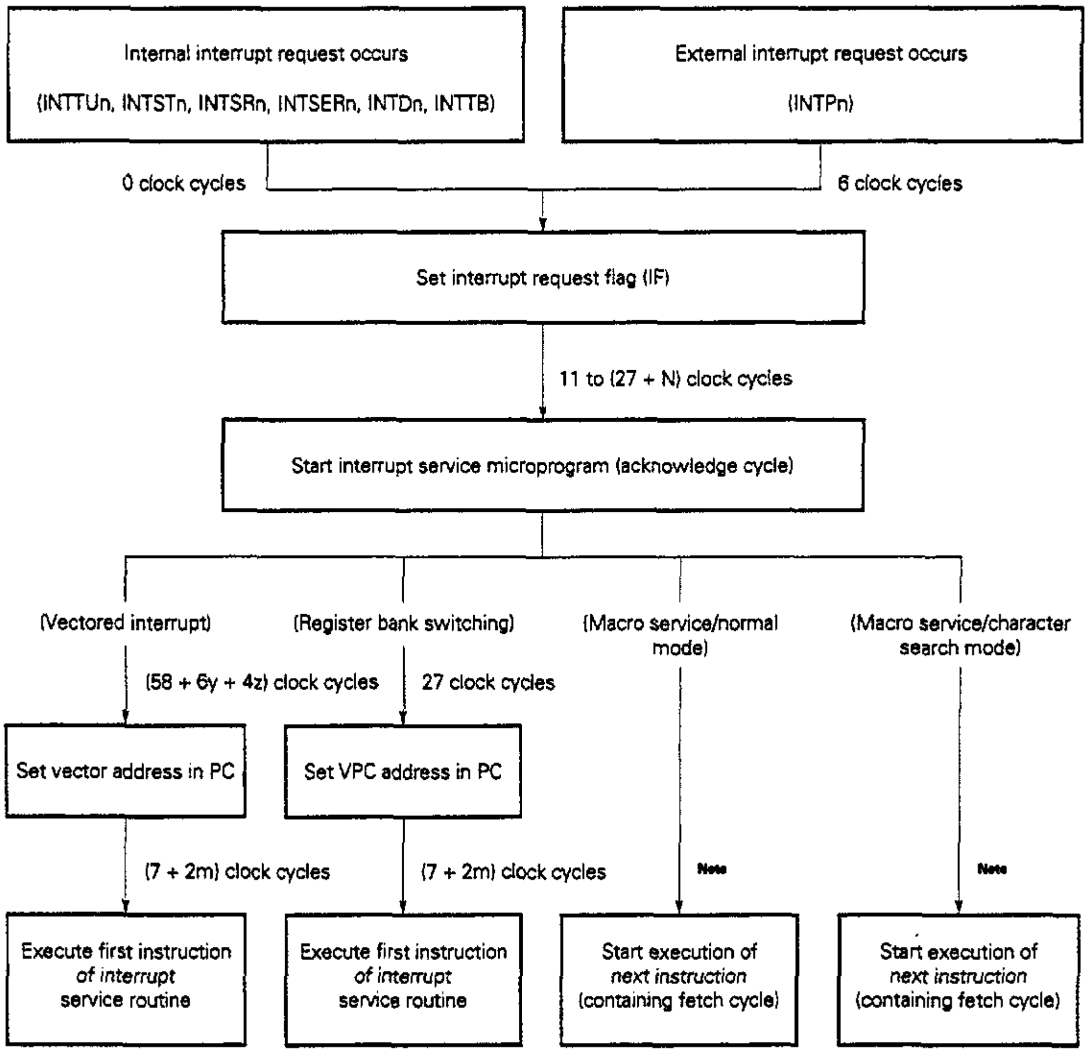
- o If an INT input occurs during execution of an INTP0 or INTP1 interrupt, the INT input is acknowledged at any time when the interrupt flag is in the EI state.

<Main routine>



4.17 Hardware Interrupt Response Time

4.17.1 V25+'s interrupt response time (number of system clock cycles)


Internal interrupt request occurs
(INTTUn, INTSTn, INTSRn, INTSERn, INTDn, INTTB)
External interrupt request occurs
(INTPn)
0 clock cycles
6 clock cycles
Set interrupt request flag (IF)
11 to (27 + N) clock cycles
Start interrupt service microprogram (acknowledge cycle)
(Vectored interrupt)
(Register bank switching)
(Macro service/normal mode)
(Macro service/character search mode)
(58 + 6y + 4z) clock cycles
27 clock cycles
Set vector address in PC
Set VPC address in PC
(7 + 2m) clock cycles
(7 + 2m) clock cycles
Note
Note
Execute first instruction of interrupt service routine
Execute first instruction of interrupt service routine
Start execution of next instruction (containing fetch cycle)
Start execution of next instruction (containing fetch cycle)

Note Macro service transfer processing time (number of system clock cycles)

N : Remaining number of execution clock cycles for the instruction being executed by the CPU at this time
y : Number of wait cycles for memory when PC, PS, and PSW are saved on a stack
z : Number of wait cycles for memory when vector PC or vector PS is read
m : Number of wait cycles for memory when the first instruction of interrupt service routine is fetched (when two bytes are fetched, instruction execution is started)

Caution Refresh cycle, hold request, DMA request, other interrupt requests, etc., are not considered.

Normal mode (unit: clock cycles)

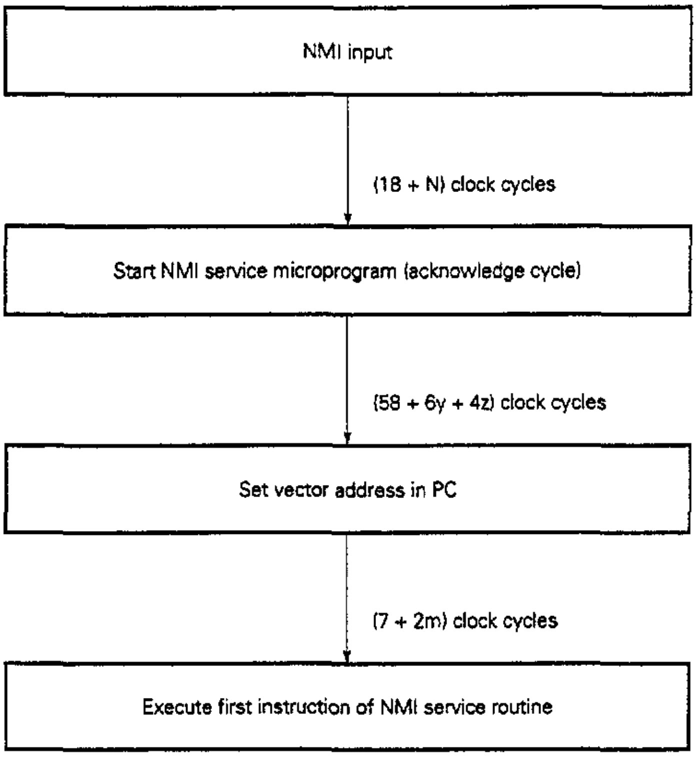
	Byte transfer		Word transfer	
	On-chip RAM access enabled	On-chip RAM access disabled	On-chip RAM access enabled	On-chip RAM access disabled
Memory to SFR	24+t	19+t	26+2t	21+2t
SFR to memory	22+t	20+t	22+2t	22+2t

Character search mode (unit: clock cycles)

	Byte transfer	
	On-chip RAM access enabled	On-chip RAM access disabled
Memory to SFR	27+t	27+t
SFR to memory	37+t	34+t

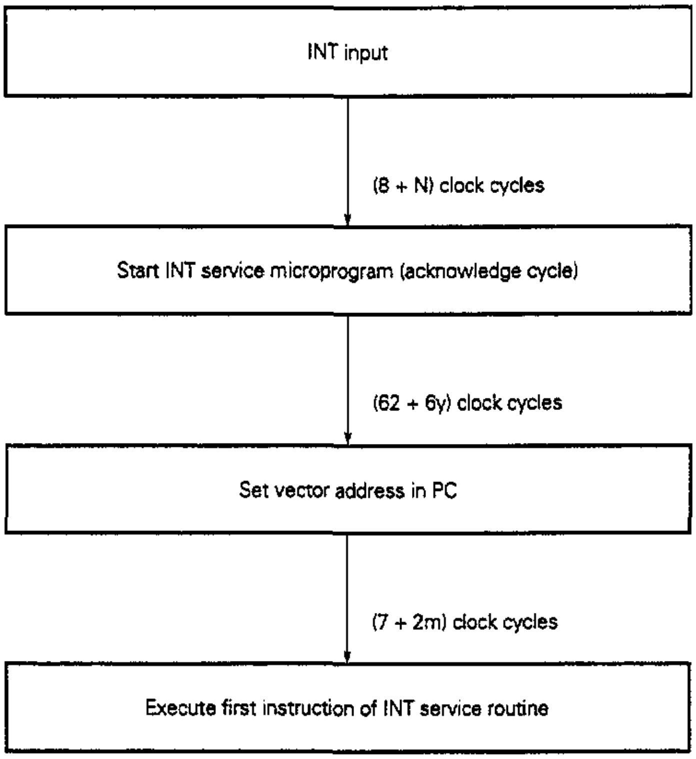
t: Number of wait cycles for memory used for data transfer

4.17.2 V25+'s NMI response time (number of system clock cycles)


NMI input
(18 + N) clock cycles
Start NMI service microprogram (acknowledge cycle)
(58 + 6y + 4z) clock cycles
Set vector address in PC
(7 + 2m) clock cycles
Execute first instruction of NMI service routine

N : Remaining number of execution clock cycles for the instruction being executed by the CPU at this time
y : Number of wait cycles for memory when PC, PS, and PSW are saved on a stack
z : Number of wait cycles for memory when vector PC or vector PS is read
m : Number of wait cycles for memory when the first instruction of interrupt service routine is fetched (when two bytes are fetched, instruction execution is started)

4.17.3 V25+'s INT response time (number of system clock cycles)


INT input
(8 + N) clock cycles
Start INT service microprogram (acknowledge cycle)
(62 + 6y) clock cycles
Set vector address in PC
(7 + 2m) clock cycles
Execute first instruction of INT service routine

N : Remaining number of execution clock cycles for the instruction being executed by the CPU at this time
y : Number of wait cycles for memory when PC, PS, and PSW are saved on a stack
z : Number of wait cycles for memory when vector PC or vector PS is read
m : Number of wait cycles for memory when the first instruction of interrupt service routine is fetched (when two bytes are fetched, instruction execution is started)

Normal mode (unit: clock cycles)

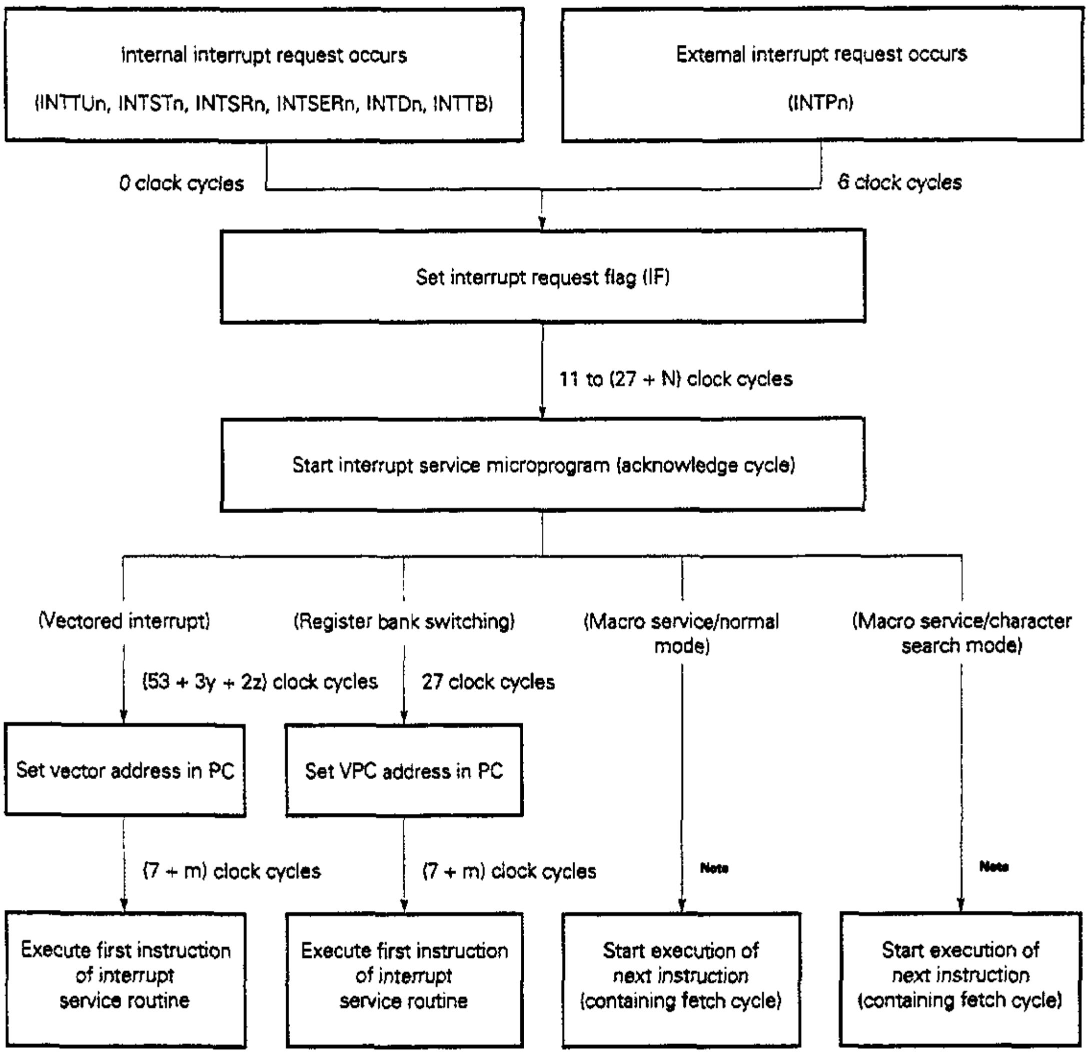
	Byte transfer		Word transfer	
	On-chip RAM access enabled	On-chip RAM access disabled	On-chip RAM access enabled	On-chip RAM access disabled
Memory to SFR	25+t	20+t	25+t	20+t
SFR to memory	22+t	21+t	22+t	21+t

Character search mode (unit: clock cycles)

	Byte transfer	
	On-chip RAM access enabled	On-chip RAM access disabled
Memory to SFR	28+t	28+t
SFR to memory	38+t	35+t

t: Number of wait cycles for memory used for data transfer

4.17.4 V35+'s interrupt response time (number of system clock cycles)

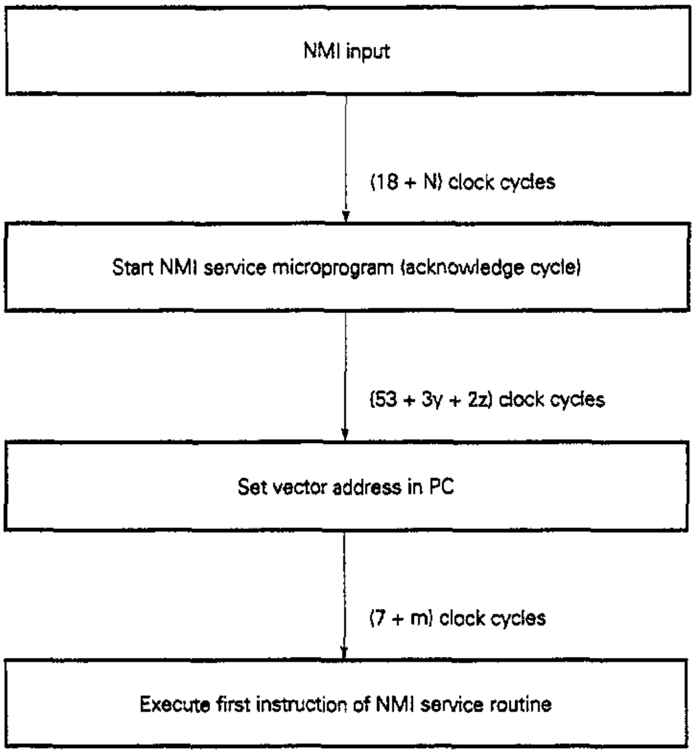

Internal interrupt request occurs
(INTTUn, INTSTn, INTSRn, INTSERn, INTDn, INTTB)
External interrupt request occurs
(INTPn)
0 clock cycles
6 clock cycles
Set interrupt request flag (IF)
11 to (27 + N) clock cycles
Start interrupt service microprogram (acknowledge cycle)
(Vectored interrupt)
(Register bank switching)
(Macro service/normal mode)
(Macro service/character search mode)
(53 + 3y + 2z) clock cycles
27 clock cycles
Set vector address in PC
Set VPC address in PC
(7 + m) clock cycles
(7 + m) clock cycles
Note
Note
Execute first instruction of interrupt service routine
Execute first instruction of interrupt service routine
Start execution of next instruction (containing fetch cycle)
Start execution of next instruction (containing fetch cycle)

Note Macro service transfer processing time (number of system clock cycles)

N : Remaining number of execution clock cycles for the instruction being executed by the CPU at this time
y : Number of wait cycles for memory when PC, PS, and PSW are saved on a stack
z : Number of wait cycles for memory when vector PC or vector PS is read
m : Number of wait cycles for memory when the first instruction of interrupt service routine is fetched (when two bytes are fetched, instruction execution is started)

Caution Refresh cycle, hold request, DMA request, other interrupt requests, etc., are not considered.

4.17.5 V35+'s NMI response time (number of system clock cycles)


NMI input
(18 + N) clock cycles
Start NMI service microprogram (acknowledge cycle)
(53 + 3y + 2z) clock cycles
Set vector address in PC
(7 + m) clock cycles
Execute first instruction of NMI service routine

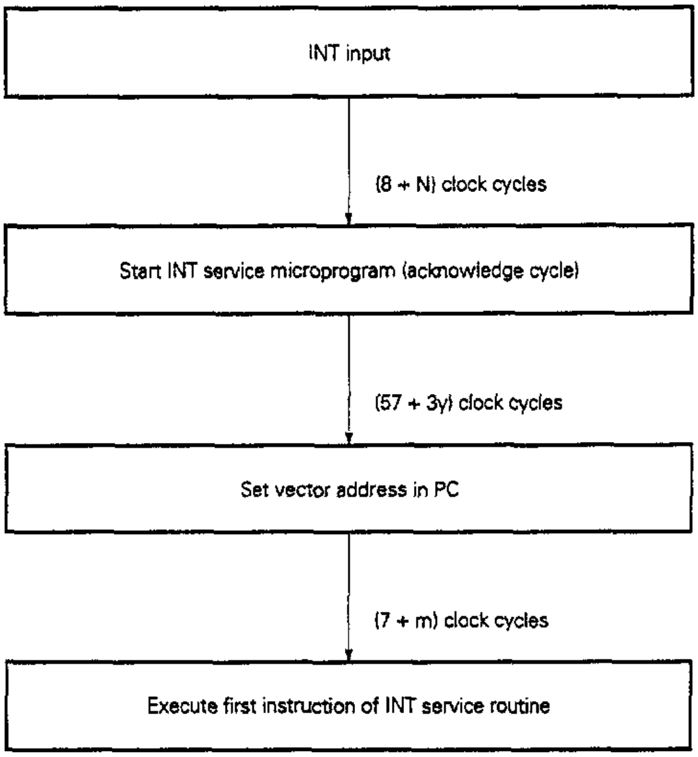
N : Remaining number of execution clock cycles for the instruction being executed by the CPU at this time

y : Number of wait cycles for memory when PC, PS, and PSW are saved on a stack

z : Number of wait cycles for memory when vector PC or vector PS is read

m : Number of wait cycles for memory when the first instruction of interrupt service routine is fetched (when two bytes are fetched, instruction execution is started)

4.17.6 V35+'s INT response time (number of system clock cycles)


INT input
(8 + N) clock cycles
Start INT service microprogram (acknowledge cycle)
(57 + 3y) clock cycles
Set vector address in PC
(7 + m) clock cycles
Execute first instruction of INT service routine

N : Remaining number of execution clock cycles for the instruction being executed by the CPU at this time

y : Number of wait cycles for memory when PC, PS, and PSW are saved on a stack

z : Number of wait cycles for memory when vector PC or vector PS is read

m : Number of wait cycles for memory when the first instruction of interrupt service routine is fetched (when two bytes are fetched, instruction execution is started)