CHAPTER 3 INTERNAL CPU FUNCTIONS

The μPD70325 and 70335 have a 16-bit CPU which is software-compatible with the μPD70116 and 70108 CPU in the native mode.

3.1 Hardware Configuration

The internal CPU in the μPD70325 and 70335 can be broadly divided into the three functional units described below. These units function together to perform processing in pipeline mode, which makes for efficient bus usage and high-speed instruction execution.

- PAU (Address calculation unit)
- EXU (Execution unit)
- BCU (Bus control unit)

3.1.1 PAU (Address calculation unit)

The PAU generates a 20-bit physical address from information provided by the EXU and then requests the BCU to be started.

3.1.2 EXU (Execution unit)

The EXU performs basic processing related to instruction execution, such as arithmetic and logical operations and data transfer. A pipeline stage for the EXU consists of 2 to 4 clock cycles, and each instruction is executed in one or several pipeline stages.

3.1.3 BCU (Bus control unit)

The BCU starts a required bus cycle based on the 20-bit physical address generated by the PAU. At the same time, it executes bus control, such as for READY, hold, and refresh.

When the PAU does not send a bus cycle start request to the BCU, the BCU generates an opcode prefetch address and prefetches an instruction. The prefetched opcode is stored in the prefetch queue.

A pipeline stage for the BCU is performed in one bus cycle.

Prefetch queue

The μPD70325 and 70335 have a six-byte instruction queue (FIFO) that can store a maximum of six bytes of the opcodes prefetched by the BCU.

The opcode stored in the queue is fetched by the EXU for execution.

When a branch, call, return, or break instruction is executed or when external interrupt servicing is performed, the queue contents are cleared and an instruction at a new location is prefetched.

Normally, an opcode is prefetched when the queue contains one or more bytes of free space.

If the average execution time of consecutively executed instructions exceeds the number of clock cycles required to prefetch the opcode of each instruction to some degree, when the EXU terminates execution of one instruction it can immediately execute the opcode stored in the queue. The fetch time from memory can thus be excluded from the instruction execution time. As a result, processing speed is faster when compared to other CPUs that fetch and execute one instruction at a time.

However, the queue's effectiveness is reduced in the following cases.

- When a number of queue clear instructions are executed, such as when a branch instruction is executed.
- When instructions having short execution times are executed consecutively.

3.1.4 CPU pipelines

The μPD70325 and 70335 CPU uses a synchronous pipeline structure. Accordingly, after processing of one pipeline stage is completed by each of the units (PAU, EXU, and BCU) that have been started simultaneously, the next stage is started. Figure 3-1 shows an example of pipeline operations.

Example Pipeline operation when the following instructions are executed.

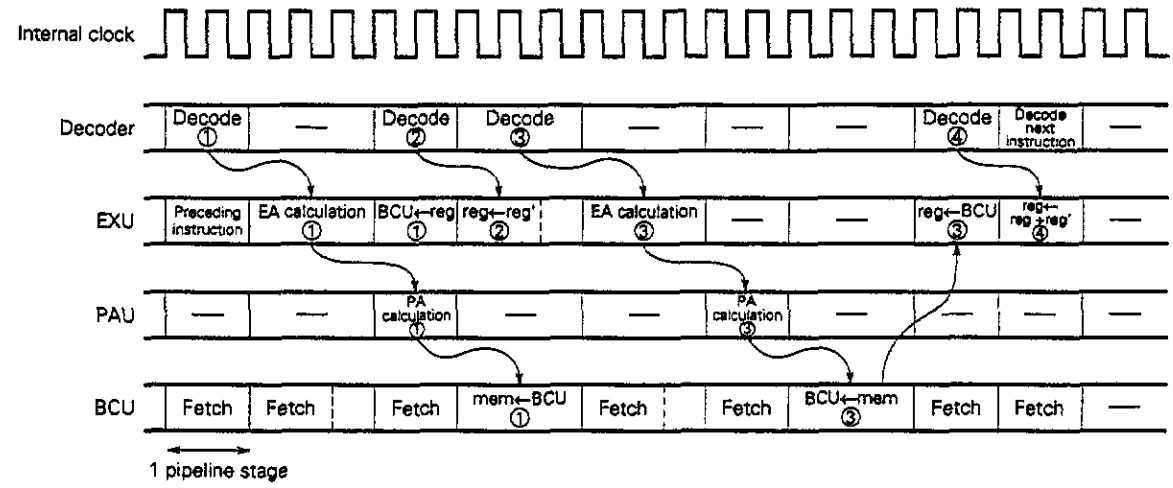
```
    :
MOV     mem, reg   ... ①
MOV     reg, reg'  ... ②
MOV     reg, mem   ... ③
ADD     reg, reg'  ... ④
    :
```

This example assumes the following conditions.

- Prefetch cycle: 2 clocks (0 wait)
- Memory read/write: 3 clocks (1 wait)

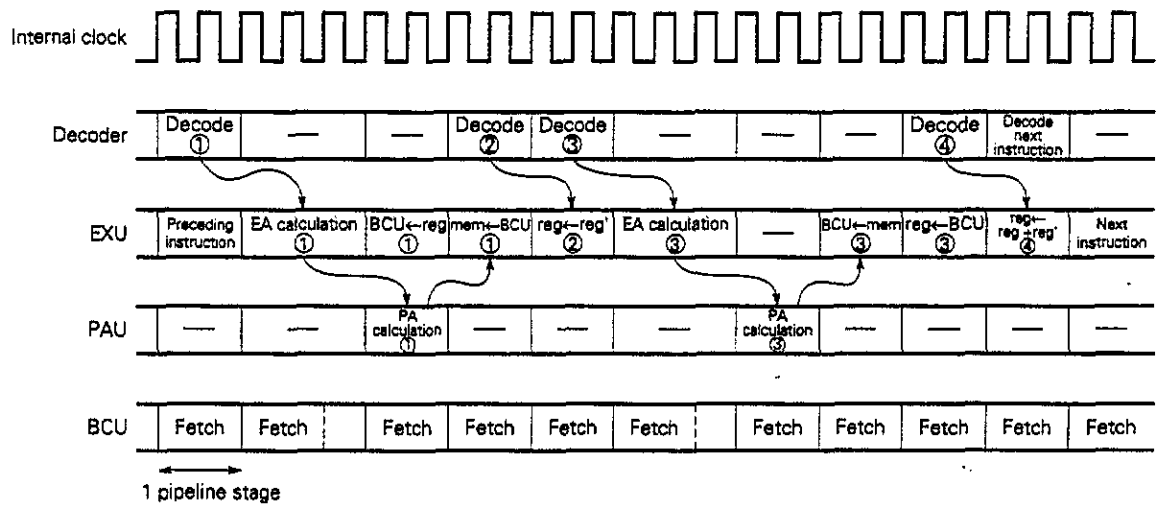
Figure 3-1. CPU Pipeline Operation Example (1/2)

(a) When internal RAM access is disabled



Remark The next instruction is executed even during the BCU's memory write operation.

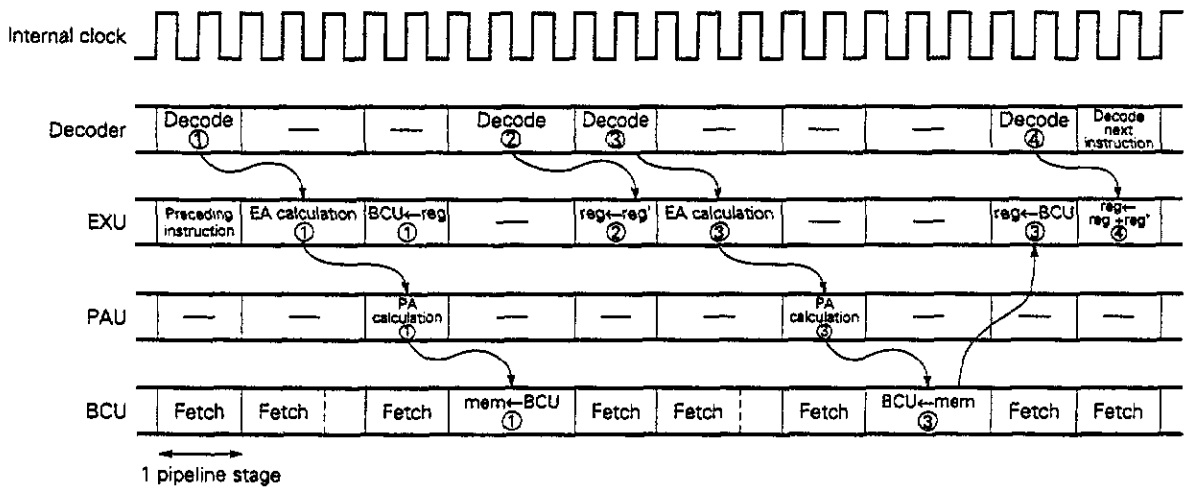
(b) When internal RAM access is enabled (access to internal RAM)



Remark The EXU executes memory write and memory read access to internal RAM. During this operation, the BCU executes program fetch.

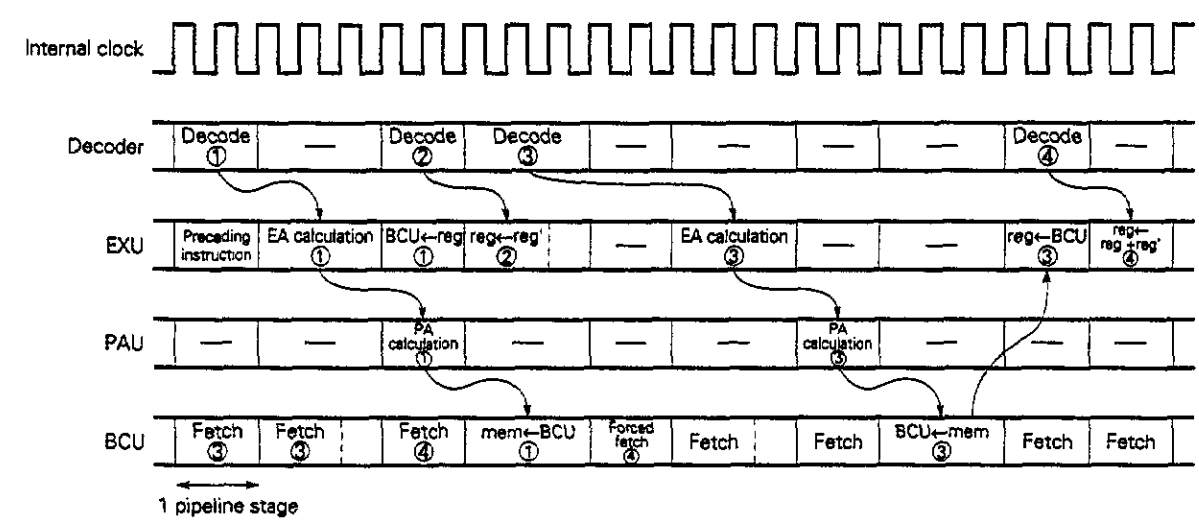
Fig. 3-1 CPU Pipeline Operation Example (2/2)

(c) When internal RAM access is enabled (access to external RAM)



Remark When internal RAM access is enabled, the EXU does not go to execution of the next instruction during BCU memory write.

(d) When a forced prefetch cycle is entered



Remark A forced prefetch cycle is started when there are less than two bytes of code stored in the prefetch queue.

Forced prefetch cycle

The μPD70325 and 70335 have a six-byte prefetch queue. Normally, prefetch is executed when there is one or more bytes of free space in the queue.

When the queue contains one byte or less of opcode, execution of the next pipeline stage is halted, and forced prefetch cycles are started until at least two bytes of opcode have been stored in the prefetch queue.

If the addresses for instructions such as branch or call instructions are not consecutive, the queue contents are cleared, and two bytes of opcode at a new location are fetched for execution of the instruction.

3.2 Registers

The CPU in the μPD70325 and 70335 has a general purpose register set that is compatible with that in the μPD70116 and 70108. It also has various special function registers to control the on-chip peripheral hardware. All of the registers are mapped in memory space. In particular, the general purpose register set can also be used for on-chip RAM, and a maximum of eight banks of the register set can be arranged on the on-chip RAM.

The addresses of these registers can be relocated in 4-Kbyte units and are specified by setting the internal data area base register (IDB), which is a special function register. (See section **3.5.2.**)

3.2.1 Register banks

The general purpose register set is mapped in the on-chip RAM area. This general purpose register set uses the bank format, and up to eight banks can be set, with 32 bytes used per bank. Of these eight banks, banks 0 and 1 are also used as macro service channels (see section **4.5.4**) and can also be accessed as data memory (see section **3.5.4**).

Normally, the CPU uses register bank 7 to execute programs. When an interrupt occurs, switching to another register bank is performed automatically. To return from the current register bank to the previous one, the return instruction RETRBI (an added instruction from the μPD70108 and 70116) from the interrupt must be executed. (See section **4.5.2.**)

Figure 3-2 shows the configuration of register banks. In each register bank, (+00H) and (+01H) are reserved areas when the register bank is used. The general purpose register set is mapped in the area of (+08H) to (+1FH), which are offset from the start address of each register bank. The area from (+02H) to (+07H) is used during register bank switching, and is therefore not available for other use.

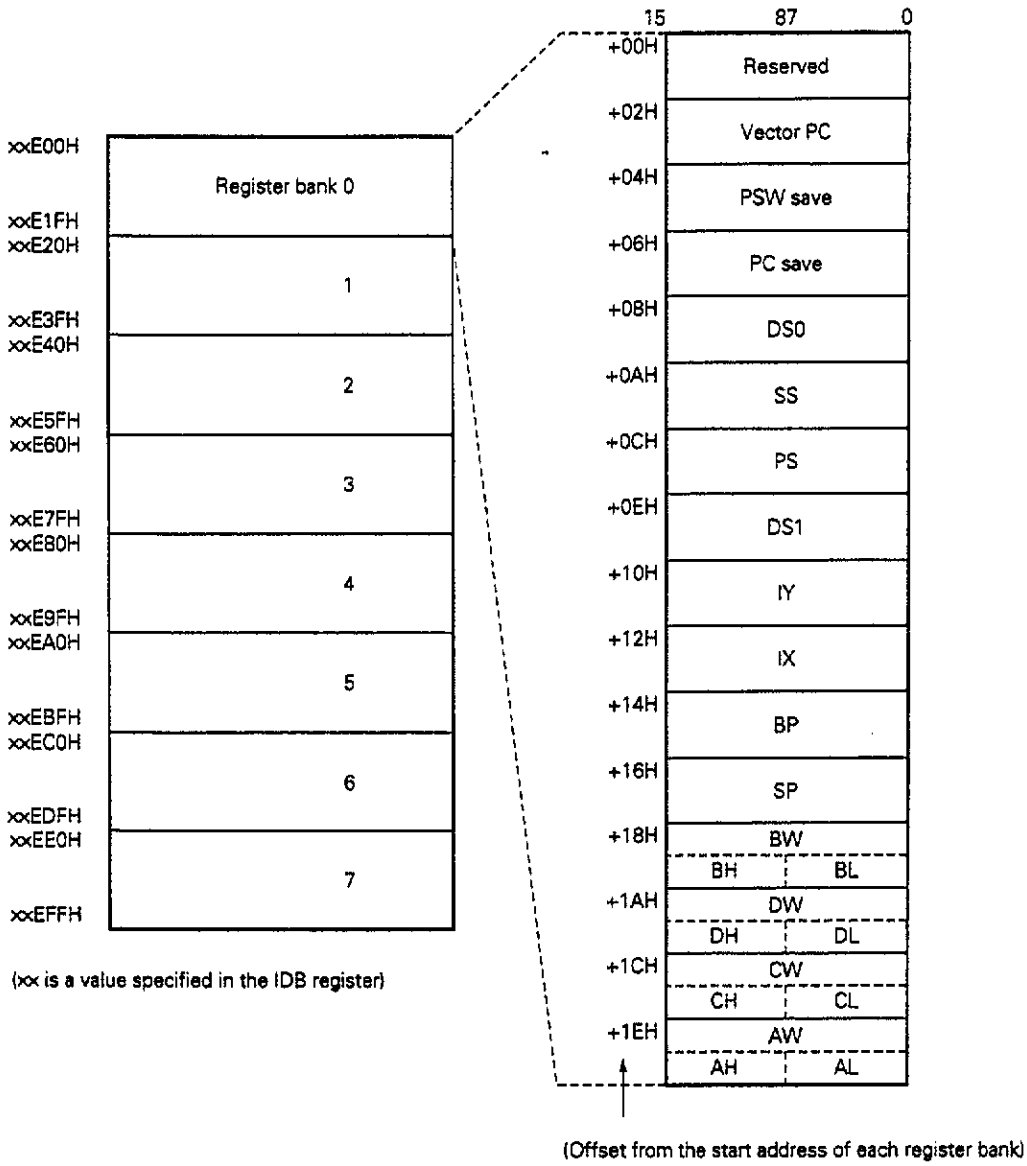
The value loaded into the program counter (PC) during register bank switching, that is, the offset value for the interrupt service routine's starting address, is set in the (+02H) area.

(+04H) is an area for saving the program status word (PSW) during register bank switching.

(+06H) is an area for saving the PC during register bank switching. (See section **4.5.2.**)

After reset, register bank 7 is automatically selected. Only the segment registers (see section **3.2.4**) in register bank 7 are initialized at reset.

Figure 3-2. Register Bank Configuration



3.2.2 General purpose registers (AW, BW, CW, and DW)

The general purpose registers contain four 16-bit registers. In addition to being accessed as 16-bit registers, these registers can also be accessed as 8-bit registers by dividing them into high-order and low-order halves (AH, AL, BH, BL, CH, CL, DH, DL).

The registers are used as 8-bit or 16-bit registers for various instructions, including transfer, arithmetic, and logical operation instructions.

These registers are also used as default registers for the types of processing specified below.

AW : *Word multiplication/division, word input/output, and data conversion*
AL : Byte multiplication/division, byte input/output, translation, BCD rotation, and data conversion
AH : Byte multiplication/division
BW : Translation
CW : Loop control branch and repeat prefix
CL : Shift instruction, rotation instruction, and BCD operation
DW : Word multiplication/division and indirect addressing input/output

These registers are mapped in the on-chip RAM. The addresses can be found using the following expression:
(IDB register **Note** value x 4096) + (0E00H) + (register bank number x 32) + (offset for each register)

Note See section **3.5.2** regarding the IDB register.

Table 3-1. General Purpose Register Offsets

Register	Offset	Register	Offset
AW	1EH	AL	1EH
		AH	1FH
BW	18H	BL	18H
		BH	19H
CW	1CH	CL	1CH
		CH	1DH
DW	1AH	DL	1AH
		DH	1BH

3.2.3 Pointers (SP and BP) and index registers (IX and IY)

SP, BP, IX, and IY are 16-bit registers that are used as base pointers or index registers when memory is being accessed in an addressing mode such as based addressing (BP), indexed addressing (IX and IY), or based indexed addressing (BP, IX, and IY). The SP is also used as a pointer when a stack is handled. Like general purpose registers, they are used for instructions such as transfer and arithmetic operations, but they cannot be used as 8-bit registers for the instructions. These registers are also used as default registers for the types of processing specified below.

SP : Stack manipulation
IX : Block transfer and BCD string operation source
IY : Block transfer and BCD string operation destination

These registers are mapped in the on-chip RAM. The addresses can be found using the following expression.
(IDB register **Note** value × 4096) + (0E00H) + (register bank number × 32) + (offset for each register)

Note See section **3.5.2** regarding the IDB register.

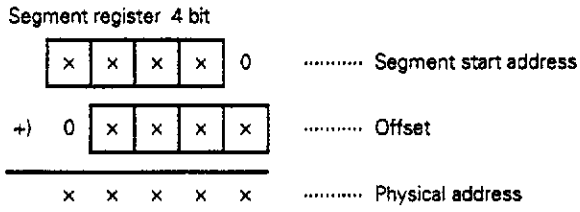
Table 3-2. Pointer and Index Register Offsets

Register	Offset
SP	16H
BP	14H
IX	12H
IY	10H

3.2.4 Segment registers (PS, SS, DS0, and DS1)

The CPU divides memory space into 64-Kbyte logical segments. The start address of each segment is specified in the segment register. Offset from the start address is specified in another register or by effective address.

Accordingly, the physical address is generated as shown below.



The segment registers are PS (Program Segment), SS (Stack Segment), DS0 (Data Segment 0), and DS1 (Data Segment 1). The segments are used for the following.

PS : Program fetch
SS : Stack manipulate instruction and addressing with BP as the base register
DS0: General variables access and source block data access (block transfer instruction, etc.)
DS1: Destination block data access (block transfer instruction, etc.)

However, a segment other than DS0 can be used if a segment override prefix is used. Similarly, another segment can be used instead of SS in the addressing mode with BP as a base register.

When reset, register bank 7's PS is initialized to FFFFH and SS, DS0, and DS1 are initialized to 0000H.

These registers are mapped in the on-chip RAM. The addresses can be found using the following expression.

(IDB register **Note** value × 4096) + (0E00H) + (register bank number × 32) + (offset for each register)

Note See section **3.5.2** regarding the IDB register.

Table 3-3. Segment Register Offsets

Register	Offset
DS0	08H
DS1	0EH
SS	0AH
PS	0CH

3.2.5 Internal data area base register (IDB)

The internal data area base register (IDB) is an 8-bit register used to determine the address of the internal data area (see section **3.5.1**), which consists of on-chip RAM (also used for general purpose registers) and a special function register area for on-chip peripheral hardware control, etc. The register can be referenced at two addresses: FFFFFH and (IDB register value × 4096 + FFFH). (See section **3.5.2**.)

3.2.6 Special function registers

The μPD70325 and 70335 have special function registers to control on-chip peripheral hardware. These registers are mapped in the special function register area in the internal data area and are read/written using the same method as normal memory access. (See section **3.5.3**).

Execution of the additional instruction BTCLR is valid only for the special function registers.

3.3 Program Counter (PC)

The program counter (PC) is a 16-bit binary counter which retains the offset of the program memory address having the program to be executed by the CPU.

The PC is incremented each time an instruction byte is fetched from the instruction queue. A new location is loaded into the PC whenever an instruction such as a branch, call, return, or break instruction is executed.

When reset, 0000H is loaded into the PC. Because the PS is initialized to FFFFH when reset, after reset the CPU starts program execution at address FFFF0H.

3.4 Program Status Word (PSW)

The program status word (PSW) consists of six status flags, five control flags, and two user flags.

- o Status flags
  - V (Overflow)
  - S (Sign)
  - Z (Zero)
  - AC (Auxiliary Carry)
  - P (Parity)
  - CY (Carry)
- o Control flags
  - RB0-RB2 (Register Bank0-2)
  - DIR (Direction)
  - IE (Interrupt Enable)
  - BRK (Break)
  - $\overline{\text{IBRK}}$ (I/O Break)
- o User flags

  F0 (user Flag0)

  F1 (user Flag1)

The status flags are automatically set to 1 or reset to 0 according to the instruction execution result (data values). The CY flag can also be directly set, reset, or inverted by executing an instruction.

The control flags are set or reset by executing an instruction for CPU operation control. The IE and BRK flags are reset whenever interrupt servicing is started.

The user flags that are available to the user can be set, reset, or tested by executing instructions.

The PSW bit configuration is shown below.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	RB2	RB1	RB0	V	DIR	IE	BRK	S	Z	F1	AC	F0	P	$\overline{\text{IBRK}}$	CY

The PSW contents can be saved and restored from the stack by executing PUSH and POP instructions. However, when the PSW contents are restored from the stack by executing the POP PSW instruction, the contents of bits 12 to 14 (RB0 to RB2) are not restored to the PSW.

The PSW's low-order eight bits can also be saved to and restored from the AH register by executing the MOV instruction.

When an interrupt occurs, the PSW contents are automatically saved in the stack before IE and BRK are reset.

When $\overline{\text{RESET}}$ is asserted, RB0 to RB2, $\overline{\text{IBRK}}$, and bit 15 are all set to 1 and other bits are reset to 0.

Caution Be sure to set PSW's bit 15 to 1.

3.4.1 CY (Carry Flag)

(1) Binary addition and subtraction

The CY (Carry Flag) is set when a byte operation results in carrying into or borrowing from bit 7. Otherwise, it is reset.

The CY is set when a word operation results in carrying into or borrowing from bit 15. Otherwise, it is reset.

Increment and decrement instructions do not affect the CY flag.

(2) Logical operations

The CY flag is reset regardless of the operation result.

(3) Binary multiplication

If AH is 0 as a result of an unsigned byte operation, the CY flag is reset. If AH is not 0, the CY flag is set.

If AH is an AL sign extension as a result of a signed byte operation, the CY flag is reset. Otherwise, the CY flag is set.

If DW is 0 as a result of an unsigned word operation, the CY flag is reset. If DW is not 0, the CY flag is set.

If DW is an AW sign extension as a result of a signed word operation, the CY flag is reset. Otherwise, the CY flag is set.

When the product of an 8-bit immediate operation is within 16 bits, the CY flag is reset. When the product exceeds 16 bits, the CY flag is set.

(4) Binary division

Undefined

(5) Shift and rotate

When a shift or rotate instruction that also manipulates the CY flag is executed, if the bit shifted to the CY flag is 1, the CY flag is set. If this bit is 0, the CY flag is reset.

3.4.2 P (Parity Flag)

(1) Binary addition and subtraction, logical operation, and shift

When there is an even number of "1" bits among the low-order eight bits of the operation result, the P flag is set. When there is an odd number of "1" bits, the P flag is reset.

When the result is all zeros, the P flag is set.

(2) Binary multiplication and division

Undefined

3.4.3 AC (Auxiliary Flag)

(1) Binary addition and subtraction

The AC flag is set when a byte operation generates either a carry from the low-order four bits to the high-order four bits or a borrow from the high-order four bits to the low-order four bits. Otherwise, the AC flag is reset.

Word operations are performed on low-order bytes in the same way as byte operations.

(2) Logical operation, binary multiplication and division, and shift and rotate

Undefined

3.4.4 Z (Zero Flag)

(1) Binary addition and subtraction, logical operation, and shift and rotate

The Z flag is set when eight bits in the result of a byte operation or 16 bits in the result of a word operation are all zeros. Otherwise, the Z flag is reset.

(2) Binary multiplication and division

Undefined

3.4.5 S (Sign Flag)

(1) Binary addition and subtraction, logical operation, and shift and rotate

The S flag is set when bit 7 of a byte operation result is 1. When bit 7 is 0, the S flag is reset.
The S flag is set when bit 15 of a word operation result is 1. When bit 15 is 0, the S flag is reset.

(2) Binary multiplication and division

Undefined

3.4.6 V (Overflow Flag)

(1) Binary addition and subtraction

When a byte operation is performed, the V flag is set if the carry values from bit 6 and bit 7 differ. If they are the same, the V flag is reset.
When a word operation is performed, the V flag is set if the carry values from bit 14 and bit 15 differ. If they are the same, the V flag is reset.

(2) Logical operation

The V flag is reset regardless of the operation result.

(3) Binary multiplication

If AH is 0 as a result of an unsigned byte operation, the V flag is reset. If AH is not 0, the V flag is set.
If AH is an AL sign extension as a result of a signed byte operation, the V flag is reset. Otherwise, the V flag is set.
If DW is 0 as a result of an unsigned word operation, the V flag is reset. If DW is not 0, the V flag is set.
If DW is an AW sign extension as a result of a signed word operation, the V flag is reset. Otherwise, the V flag is set.
When the product of an 8-bit immediate operation is within 16 bits, the V flag is reset. When the product exceeds 16 bits, the V flag is set.

(4) Binary division

The V flag is reset regardless of the operation result.

(5) Shift and rotate

As a result of one-bit left shift or rotate instruction,

When CY = most significant bit: V is reset
When CY ≠ most significant bit: V is set

As a result of one-bit right shift or rotate instruction,

When most significant bit = second most significant bit: V is reset
When most significant bit ≠ second most significant bit: V is set

When shifting or rotating by multiple bits, the V flag becomes undefined.

3.4.7 $\overline{IBRK}$ (I/O Break Flag)

$\overline{IBRK}$ controls the occurrence of software interrupts when an I/O instruction is executed.

When $\overline{IBRK}$ = 0, a software interrupt (interrupt vector 19) occurs whenever execution of an I/O instruction is attempted. Therefore, I/O instructions can be emulated by software.

When $\overline{IBRK}$ = 1, a software interrupt does not occur even when an I/O instruction is executed.

3.4.8 BRK (Break Flag)

If the BRK flag is set, a software interrupt (interrupt vector 1) occurs each time one instruction is executed. Therefore, instructions can be traced one at a time.

The BRK flag is set by executing a memory manipulate instruction only when it is saved in the stack as part of the PSW. When the BRK flag is restored to the PSW after it is set, the flag becomes valid.

3.4.9 IE (Interrupt Enable Flag)

When an IE flag is set by executing an EI instruction, interrupts are enabled. When an IE flag is reset by executing a DI instruction, interrupts are disabled.

3.4.10 DIR (Direction Flag)

The DIR flag is set by executing the SET1 DIR instruction and is reset by executing the CLR1 DIR instruction.

If a DIR flag is set, processing is performed from a high-order address to a low-order address when a block transfer or I/O instruction is executed. If the DIR flag is reset, processing is performed from a low-order address to a high-order address.

3.4.11 RB0 to RB2 (Register Banks 0 to 2 Flag)

RB0 to RB2 indicate the current register bank among the eight register banks set in the on-chip RAM.

RB0 to RB2 are not restored from the stack when the POP PSW instruction is executed.

Caution Do not change the values for RB0 to RB2 that are saved in the stack and PSW save register by an interrupt service routine.

3.4.12 F0 and F1 (User Flags 0 and 1)

F0 and F1 are flags that the user can use as desired.

F0 and F1 can be set and reset by executing a PSW-related instruction. They can also be set, reset, and tested by setting the FLAG register (a special function register).

Figure 3-3 shows the FLAG register's bit configuration.

Figure 3-3. FLAG Register

7	6	5	4	3	2	1	0
-	-	F1	-	F0	-	-	-

3.5 Memory Space

The μPD70325 and 70335 have 1 Mbyte of memory space. Figure 3-4 shows a memory map.

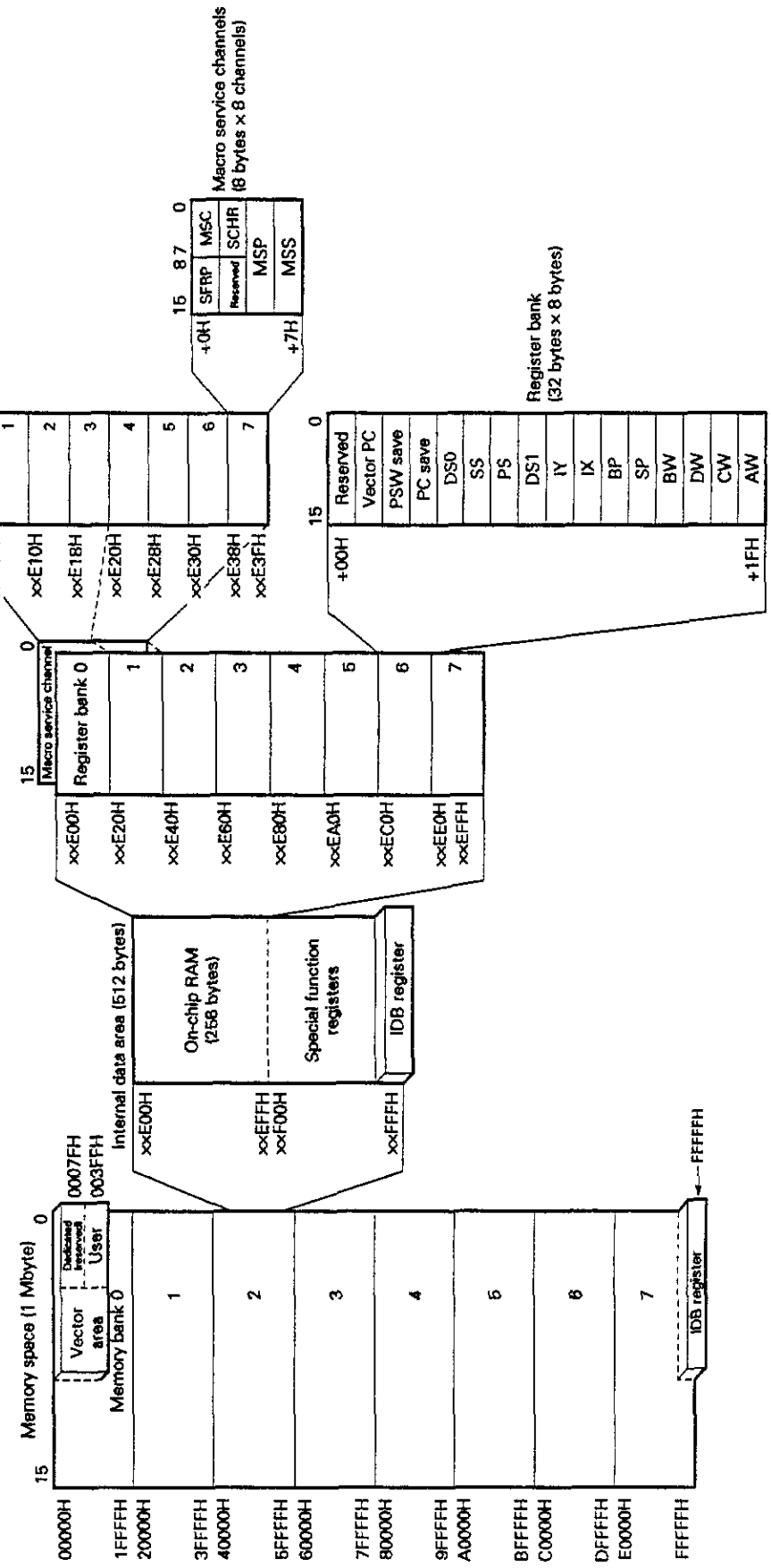
00000H to 003FFH : Vector area (if not used as a vector area, this area can be used for another purpose)
××E00H to ××FFFH : Internal data area (the location of this area can be changed in 4-Kbyte units)
FFFFCH to FFFFEH : Reserved area
FFFFFH : IDB register

Remark ××: IDB register value

When the memory space is accessed, wait cycle insertion every 128 Kbytes is programmable.
Do not access spaces that has not been memory-mapped.
See section **3.2.4** for the physical addresses.

Cautions
1. **If a word reference is made with the offset within the segment at address FFFFH, the second byte reference address is FFFFH+1 rather than 0000H in the same segment (it is 0000H in the V20 and V30).**
2. **If the combination of the segment value and offset value exceeds the physical address FFFFFH, the next address to be generated becomes 00000H.**

Figure 3-4. Memory Map



Remarks 1. xx is the IDB register value.

2. +□□H is the address's offset value. Adding the register bank's or macro service channel's start address to this value gives the actual address.

3. Macro service channels are assigned in duplicate to register banks 0 and 1.

3.5.1 Internal data area

The internal data area is a 512-byte area consisting of the on-chip RAM area and the special function register area. It can be relocated in 4-Kbyte units within the 1-Mbyte memory space. The IDB register (internal data area base register) sets the base address of the internal data area. The high-order eight bits of the 20-bit internal data area base address are specified in the IDB register and the low-order 12 bits are fixed to E00H.

When $\overline{\text{RESET}}$ is asserted, the IDB register is initialized to FFH, and therefore the internal data area is located between FFE00H and FFFFFH.

(1) Access to internal data area

As shown in Figure 3-4, the internal data area overlaps with the external memory area. The overlapping memory areas to be accessed are divided according to the memory access conditions. Figure 3-5 (1) and (2) shows the memory access conditions and memory area to be accessed.

Data in the internal data area is fetched in two clock cycles.

When an internal data area or an on-chip ROM area is accessed, control signals such as $\overline{\text{MREQ}}$, $\overline{\text{MSTB}}$, and $\overline{\text{IOSTB}}$ are not externally output.

(2) On-chip RAM area

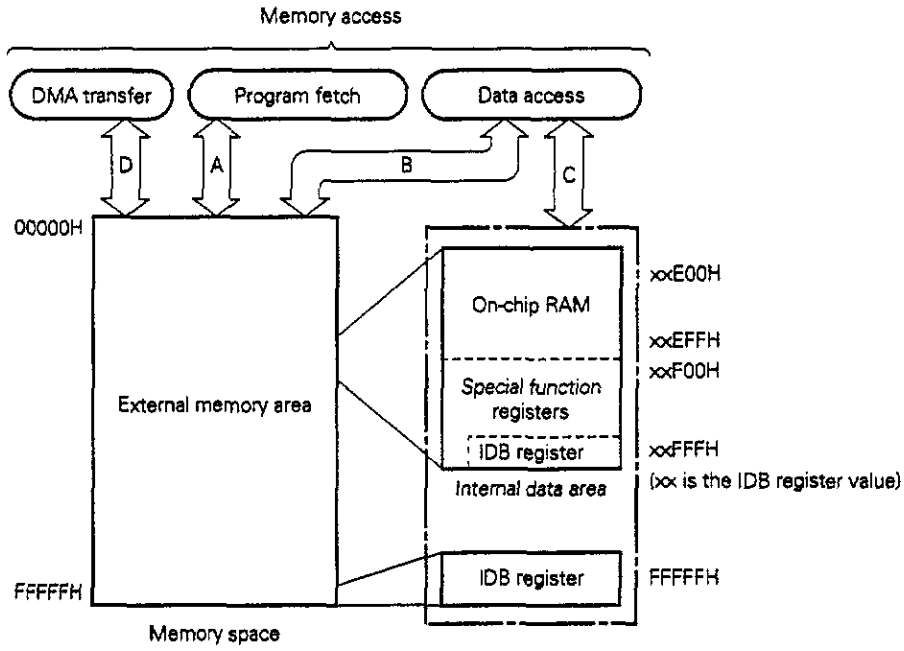
The on-chip RAM area consists of the low-order 256 bytes of the internal data area (xxE00H to xxEFFH, where xx is the IDB register value). In addition to normal use as RAM, the on-chip RAM is subject to functional assignments of register banks and macro service channels (see section **3.5.4**). Bit 6 (RAMEN) of the processor control register (PRC), a special function register, can be reset to 0 to disable access to the on-chip RAM as normal RAM.

The macro service function and DMA transfer operate normally even when the RAMEN bit is reset to 0. However, at this point, neither macro service nor DMA can be reset. To set or reset macro service or DMA, set the RAMEN bit to 1.

(3) Special function register area

The high-order 256 bytes (xxF00H to xxFFFH, where xx is the IDB register value) of the internal data area are the special function register area. The special function registers, such as the on-chip peripheral hardware mode registers and control registers, are mapped in the special function register area (see section **3.5.3**).

Figure 3-5. Memory Space Access Conditions



A : For program fetch, an area other than the internal data area is accessed.

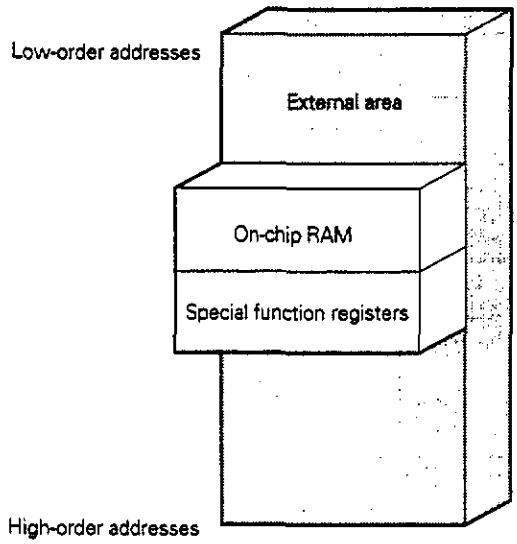
B : Data access to an area other than the internal data area, or data access to an address corresponding to the on-chip RAM area when on-chip RAM access is disabled.

C : If condition B is not satisfied, the internal data area takes precedence in being accessed.

D : For DMA transfer, the external memory area is accessed.

(1) Program fetch

External area and internal ROM area are accessed

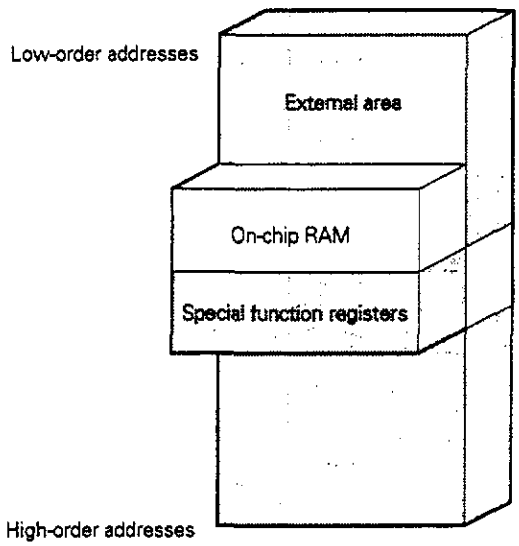


(2) Data access

(a) When on-chip RAM access is disabled

RAMEN=0 (PRC)

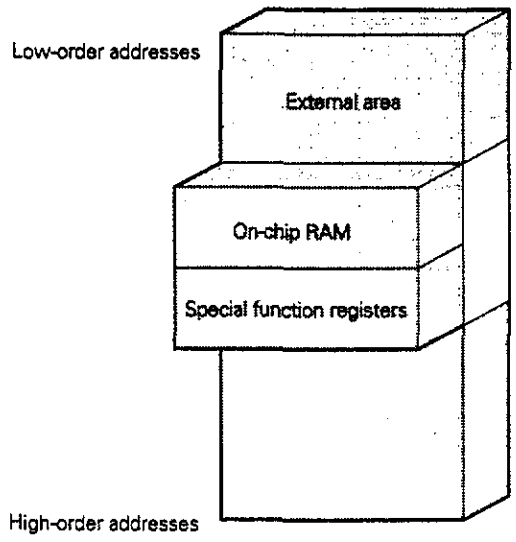
Special function registers take precedence in being accessed.



(b) When on-chip RAM access is enabled

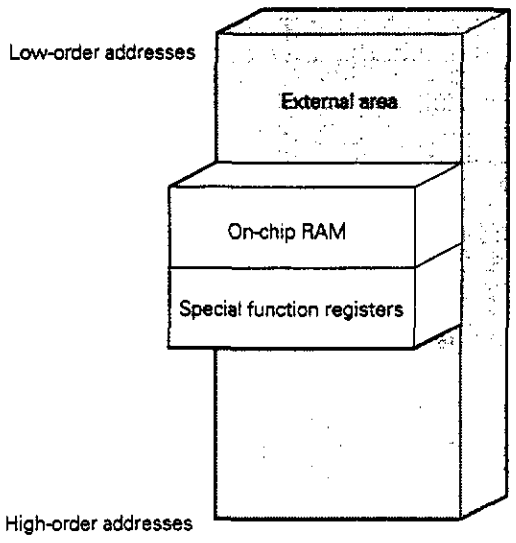
RAMEN=1 (PRC)

Internal data area takes precedence in being accessed.



(c) DMA transfer

Only external areas are accessed.



3.5.2 Internal data area base register (IDB)

The IDB register determines the physical addresses of the internal data area (on-chip RAM area and special function register area described below). The high-order eight bits of the internal data base address are specified in the IDB register. The low-order 12 bits are fixed to E00H. Access this area only as IDB.

The IDB register is allocated two addresses: xxFFFH (where xx is the IDB register value) in the special function register area and fixed address FFFFFH. The IDB register can be changed or referenced by accessing one of these addresses.

When reset, the IDB register is set to FFH. Consequently, the internal data area base address becomes FFE00H.

3.5.3 Special function register area

Special function registers, such as on-chip peripheral hardware mode registers and control registers, are mapped in the area xxF00H to xxFFFH (where xx is the IDB register value).

Programs cannot be fetched from the special function register area.

The special function registers are manipulated by accessing memory.

Table 3-4 lists the special function registers. The columns in the table have the following meanings.

- Symbol ..... The name of the special function register is represented by a symbol. It corresponds to the instruction operand.
- R/W ..... Indicates whether or not the special function register can be read and/or written.
  - R/W : Both read and write are enabled.
  - R : *Read only is enabled.*
  - W : Write only is enabled.
- Access units ..... Indicates the bit units in which the special function register can be manipulated (16 bits, 8 bits, or 1 bit).
- When reset ..... Indicates the register state when $\overline{\text{RESET}}$ is asserted.

xx in the high-order eight bits of each address is the IDB register value.

Addresses not listed in the table are reserved; a read to the reserved areas returns an undefined value, and a write to these areas is ignored.

Table 3-4. Special Function Register List (1/3)

Type	Address	Special function register name	Symbol	R/W	Access units (bits)	When reset
Port	xxF00H	Port 0	P0	R/W	8/1	Undefined
	xxF01H	Port 0 mode register	PM0	W	8	FFH
	xxF02H	Port 0 mode control register	PMC0			00H
	xxF08H	Port 1	P1	R/W	8/1	Undefined
	xxF09H	Port 1 mode register	PM1	W	8	FFH
	xxF0AH	Port 1 mode control register	PMC1			00H
	xxF10H	Port 2	P2	R/W	8/1	Undefined
	xxF11H	Port 2 mode register	PM2	W	8	FFH
	xxF12H	Port 2 mode control register	PMC2			00H
	xxF38H	Port T	PT	R	8	Undefined
	xxF3BH	Port T mode register	PMT	R/W	8/1	00H
Interrupt control	xxF40H	External interrupt mode register	INTM	R/W	8/1	00H
	xxF44H	External interrupt macro service control register 0 Note 4	EMS0			Undefined
	xxF45H	External interrupt macro service control register 1 Note 4	EMS1			
	xxF46H	External interrupt macro service control register 2 Note 4	EMS2			
	xxF4CH	External interrupt request control register 0 Note 4	EXIC0			47H
	xxF4DH	External interrupt request control register 1 Note 4	EXIC1			
	xxF4EH	External interrupt request control register 2 Note 4	EXIC2			
	xxFEFH	Interrupt source register Note 4	IRQS	R	8	Undefined
	xxFFCH	Interrupt priority register Note 4	ISPR	R	8	00H
Serial interface channel 1	xxF60H	Receive buffer register 0	RxB0	R	8	Undefined
	xxF62H	Transmit buffer register 0	TxB0	W	8	Undefined
	xxF65H	Serial reception macro service control register 0 Note 4	SRMS0	R/W	8/1	Undefined
	xxF66H	Serial transmission macro service control register 0 Note 4	STMS0			
	xxF68H	Serial mode register 0	SCM0			00H
	xxF69H	Serial control register 0	SCC0			
	xxF6AH	Baud rate generator register 0	BRG0			
	xxF6BH	Serial status register 0	SCS0	R	8	60H
	xxF6CH	Serial error interrupt request control register 0 Note 4	SEIC0	R/W	8/1	47H
	xxF6DH	Serial reception interrupt request control register 0 Note 4	SRIC0			
	xxF6EH	Serial transmission interrupt request control register 0 Note 4	STIC0			

Table 3-4. Special Function Register List (2/3)

Type	Address	Special function register name	Symbol	R/W	Access units (bits)	When reset
Serial interface channel 2	xxF70H	Receive buffer register 1	RxB1	R	8	Undefined
	xxF72H	Transmit buffer register 1	TxB1	W		
	xxF75H	Serial reception macro service control register 1 Note 4	SRMS1	R/W	8/1	Undefined
	xxF76H	Serial transmission macro service control register 1 Note 4	STMS1			Undefined
	xxF78H	Serial mode register 1	SCM1			00H
	xxF79H	Serial control register 1	SCC1			
	xxF7AH	Baud rate generator register 1	BRG1			
	xxF7BH	Serial status register 1	SCS1	R	8	60H
	xxF7CH	Serial error interrupt request control register 1 Note 4	SEIC1	R/W	8/1	47H
	xxF7DH	Serial reception interrupt request control register 1 Note 4	SRIC1			
	xxF7EH	Serial transmission interrupt request control register 1 Note 4	STIC1			
Timer unit	xxF80H	Timer register 0 Note 5	TM0	R/W	16	Undefined
	xxF82H	Modulo/timer register 0 Note 5	MD0			
	xxF88H	Timer register 1 Note 5	TM1			
	xxF8AH	Modulo/timer register 1 Note 5	MD1			
	xxF90H	Timer control register 0 Note 5	TMC0	R/W	8/1	00H
	xxF91H	Timer control register 1 Note 5	TMC1			
	xxF94H	Timer unit macro service control register 0 Note 4	TMMS0	R/W	8/1	Undefined
	xxF95H	Timer unit macro service control register 1 Note 4	TMMS1			
	xxF96H	Timer unit macro service control register 2 Note 4	TMMS2			
	xxF9CH	Timer unit interrupt request control register 0 Note 4	TMIC0			47H
	xxF9DH	Timer unit interrupt request control register 1 Note 4	TMIC1			
	xxF9EH	Timer unit interrupt request control register 2 Note 4	TMIC2			
DMA controller	xxFA0H	DMA control register 0	DMAC0	R/W	8/1	Hold Note 6
	xxFA1H	DMA mode register 0	DMAM0			00H
	xxFA2H	DMA control register 1	DMAC1			Hold Note 6
	xxFA3H	DMA mode register 1	DMAM1			00H
	xxFACH	DMA interrupt request control register 0 Note 4	DIC0	R/W	8/1	47H
	xxFADH	DMA interrupt request control register 1 Note 4	DIC1			

Table 3-4. Special Function Register List (3/3)

Type	Address	Special function register name	Symbol	R/W	Access units (bits)	When reset
DMA controller	xxFC0H	Source address pointer 0 (low-order)	SAR0L	R/W	16/8	Undefined
	xxFC1H	Source address pointer 0 (middle-order)	SAR0M			
	xxFC2H	Source address pointer 0 (high-order)	SAR0H		8	
	xxFC4H	Destination address pointer 0 (low-order)	DAR0L		16/8	
	xxFC5H	Destination address pointer 0 (middle-order)	DAR0M			
	xxFC6H	Destination address pointer 0 (high-order)	DAR0H		8	
	xxFC8H	Terminal counter 0 (low-order)	TC0L		16/8	
	xxFC9H	Terminal counter 0 (high-order)	TC0H			
	xxFD0H	Source address pointer 1 (low-order)	SAR1L		16/8	
	xxFD1H	Source address pointer 1 (middle-order)	SAR1M			
	xxFD2H	Source address pointer 1 (high-order)	SAR1H		8	
	xxFD4H	Destination address pointer 1 (low-order)	DAR1L		16/8	
	xxFD5H	Destination address pointer 1 (middle-order)	DAR1M			
	xxFD6H	Destination address pointer 1 (high-order)	DAR1H		8	
	xxFD8H	Terminal counter 1 (low-order)	TC1L		16/8	
	xxFD9H	Terminal counter 1 (high-order)	TC1H			
System control	xxFE0H	Standby control register	STBC	R/W Note 1	8/1	Hold Note 2
	xxFE1H	Refresh mode register	RFM	R/W	8/1	FCH
	xxFE8H	Wait control register	WTC	R/W	16/8	FFFFH
	xxFEAH	User flag register Note 3	FLAG	R/W	8/1	00H
	xxFEBH	Processor control register	PRC	R/W	8/1	4EH
	xxFECH	Time base interrupt request control register Note 4	TBIC			47H
	xxFFFH	Internal data area base register Note 4	IDB	R/W	8/1	FFH

Notes
1. The standby control register can be set to 1 by executing a given instruction, but cannot be cleared to 0 (W: only "1" can be written).
2. When power-on reset: 00H
3. Bit manipulation of the user flag register (FLAG) has no meaning except for bits 3 and 5. The contents of FLAG register user flags 0 and 1 (F0 and F1) can also be affected by manipulating F0 and F1 in PSW (see section **3.4.12**).
4. One wait cycle is inserted in the access cycle of these registers.
5. A maximum of six wait cycles may be inserted in the access cycle of these registers.
6. When power-on reset: Undefined.

3.5.4 On-chip RAM area

The 256-byte on-chip RAM area is contained in the area from xxE00H to xxEFFH (where xx is the IDB register value).

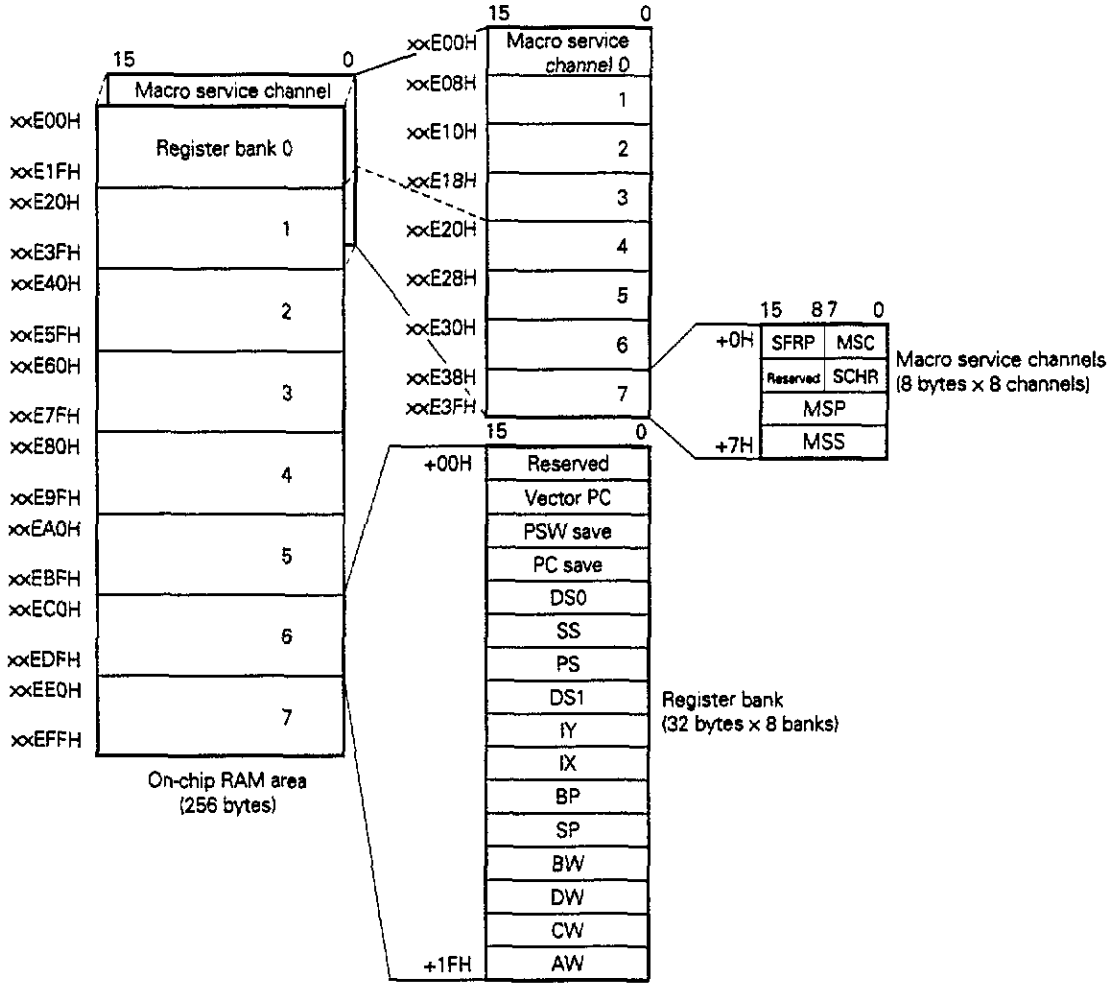
Eight register banks are mapped in the on-chip RAM. In addition, macro service channel registers are also mapped in duplicate in the on-chip RAM.

When the on-chip RAM is being accessed, the current address and data being accessed are output to the address bus and data bus as they are. The R/$\overline{W}$ signal holds its previous state. The $\overline{MREQ}$ and $\overline{MSTB}$ signals become inactive.

Memory access to the on-chip RAM can be disabled by resetting bit 6 (RAMEN) of the processor control register (PRC) to 0. Programs cannot be fetched from the on-chip RAM. When a memory access is disabled, the on-chip RAM is accessed only as a register.

Access to the on-chip RAM uses two clock cycles; the same as no-wait memory access.

Figure 3-6. On-chip RAM Area Map



3.5.5 Vector table area

This area, from 00000H to 003FFH (1 Kbyte), contains 256 vectors (4 bytes per vector) for vectored interrupts. The interrupt routine start addresses corresponding to interrupt requests, break instructions, etc., are allocated to the vector table.

Vector 0 (00000H) : Divide error
Vector 1 (00004H) : Single step
Vector 2 (00008H) : NMI input
Vector 3 (0000CH) : BRK 3 instruction
Vector 4 (00010H) : BRKV instruction
Vector 5 (00014H) : CHKIND instruction
Vector 6 (00018H) : Reserved
Vector 7 (0001CH) : FPO instruction
Vector 8 (00020H) : Reserved
| | |
Vector 11 (0002CH) : Reserved
Vector 12 (00030H) : INTSER0
Vector 13 (00034H) : INTSR0
Vector 14 (00038H) : INTST0
Vector 15 (0003CH) : Reserved
Vector 16 (00040H) : INTSER1
Vector 17 (00044H) : INTSR1
Vector 18 (00048H) : INTST1
Vector 19 (0004CH) : I/O instruction
Vector 20 (00050H) : INTD0
Vector 21 (00054H) : INTD1
Vector 22 (00058H) : Reserved
Vector 23 (0005CH) : Reserved
Vector 24 (00060H) : INTP0
Vector 25 (00064H) : INTP1
Vector 26 (00068H) : INTP2
Vector 27 (0006CH) : Reserved
Vector 28 (00070H) : INTTU0
Vector 29 (00074H) : INTTU1
Vector 30 (00078H) : INTTU2
Vector 31 (0007CH) : INTTB
Vector 32 (00080H) : } User area
| } • BRK imm8 instruction
Vector 255 (003FCH) : } • INT input

Vectors 0 to 31 are used for designated interrupt sources (part of this area is reserved) and cannot be used for other purposes.

Vectors 32 to 255 can be used for general purposes, specifically two-byte break instructions and INT input. Unused sections can be used for purposes other than vectors.

Each vector consists of four bytes. When an interrupt is acknowledged, the high-order two bytes are loaded into the program segment (PS) and the low-order two bytes are loaded into the program counter.

Example Vector 0

000H	001H	PC ← (001H, 000H)
002H	003H	PS ← (003H, 002H)

3.5.6 *External memory area*

External memory such as ROM or RAM is connected to the area from 00000H to FFFFEH in the μPD70325 and 70335. However, the areas from FFF00H to FFFEFH and from FFFFCH to FFFFEH are reserved.

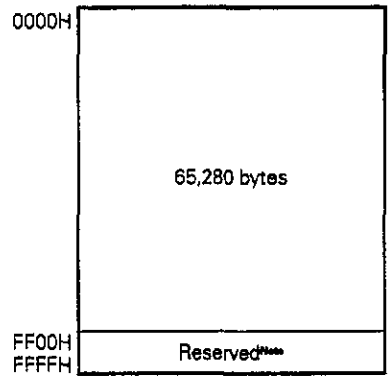
The address bus (A0 to A19), data bus (D0 to D15), and $\overline{\text{MREQ}}$, $\overline{\text{MSTB}}$, R/$\overline{\text{W}}$, and $\overline{\text{UBE}}$ signals are used to access external memory. A refresh pulse output pin ($\overline{\text{REFRQ}}$) is provided to refresh pseudo SRAM so that pseudo SRAM can be easily connected. An automatic refresh address output function is also provided to refresh DRAM so that DRAM can be easily connected (see section **5.3**). In addition, wait cycles can be inserted into memory cycles in 128-Kbyte units by software (see section **5.1**).

3.6 I/O Space

The μPD70325 and 70335 have a 64-Kbyte I/O space in addition to their 1-Mbyte memory space. Figure 3-7 shows an I/O space map.

Use the address bus (A0 to A15), data bus (D0 to D7), and $\overline{IOSTB}$, R/$\overline{W}$, $\overline{DMAAK0}$, and $\overline{DMAAK1}$ signals to access the I/O space. Zeros are output from the unused high-order four bits (A16 to A19) of the address buses. Wait cycles can be inserted into I/O cycles by software (see section **5.1**).

Figure 3-7. I/O Map (64 Kbytes)



Note Address, data, and control signals are also output when the reserved area is accessed.