HuC6280

CMOS 8-bit Microprocessor

HARDWARE MANUAL

TABLE OF CONTENTS

1. DESCRIPTION

The HuC6280 is a CMOS 8-bit microprocessor which contains an 8-bit parallel processing ALU, eight mapping registers for address expansion, a 7-bit interval timer, an 8-bit input port, an 8-bit output port, and a programmable sound generator (PSG) on a single chip. It has 2M bytes of address space, and operates at a speed as high as a 138 nsec bus cycle.

1.1 Features

- Monolithic CMOS 8-bit parallel processing micro-processor
- Operating frequency ..... 3.58-21.48 MHz (actual)
- Minimum instruction cycle time ..... 276 nsec
- Instruction set ..... 234 instructions in 89 groups
- I/O ports ..... Input port: 1 port × 8 bits
  Output port: 1 port × 8 bits
- Interrupts ..... External: $\overline{\text{NMI}}$, $\overline{\text{IRQ1}}$, $\overline{\text{IRQ2}}$
  Internal: Timer interrupt and BRK instruction
- On-chip 7-bit interval timer
- Stack area ..... 256 bytes (maximum)
- Single power supply 5V
- Package ..... 80-pin plastic flat package
- Address space ..... Controlled by eight mapping registers
  (2M bytes for physical addresses and 64K bytes for logical addresses)

2. FUNCTIONS

2.1 Register Set

The HuC6280 has a total of ten 8-bit registers:

① Accumulator (general-purpose) (ACC)
② Register X (X)
③ Register Y (Y)
④ Program counter high (PCH)
⑤ Program counter low (PCL)

NOTE: PCH and PCL make up a complete program counter.

⑥ Stack pointer (S)
⑦ Status register (P)
⑧ Source high (SH)
⑨ Destination high (DH)
⑩ Length high (LH)

NOTE: SH, DH and LH are used when a block transfer instruction is executed.

2.1.1 Accumulator (ACC)

The ACC is an 8-bit general-purpose register. When the memory operation flag (T) is set to "0", most ALU operations are performed by the ACC. The contents of the ACC are loaded into the ALU, and the result of an operation is stored in the ACC.

The ACC also is used for the transfer of data from memory to memory, or between memory and a peripheral circuit.

When a block transfer instruction (TII-TDD) is executed, the ACC saves the current data into the stack and functions as a length low register for counting the block length.

2.1.2 Registers X and Y

Both X and Y registers are an 8-bit general-purpose register that works mainly for index addressing.

With the memory operation flag (T) set to "1", register X is used to specify a memory address on page 0 which is the destination of an operation.

When a block transfer instruction (TII-TDD) is executed, register X saves the current data into the stack and functions as a source low register for specifying the source address.

When a block transfer instruction (TII-TDD) is executed, register Y saves the current data into the stack and functions as a destination low register for specifying the destination address.

2.1.3 Program Counter (PCH, PCL)

The program counter is a 16bit upcounter which consists of a program counter high (PCH) and a program counter low (PCL).

Each time an instruction is executed, the program counter is automatically incremented and specifies the address of the next instruction to execute or the address of its operand.

When a system reset occurs, the CPU loads low byte of address data into the PCL from physical address $001FFE_{16}$ and high byte into the PCH from physical address $001FFF_{16}$, then starts.

● Save or Restore Operations of Program Counter

(a) BSR or JSR instruction

When a BSR or JSR instruction is executed, the contents of the program counter are saved into the stack with the PCH preceding the PCL. The saved data in the stack points the address of the last byte of the BSR or JSR instruction.

When an RTS instruction is executed, the program counter loads the address data from the stack and increments by 1. Then the program returns from the subroutine.

(b) Interrupt

When an interrupt occurs, the contents of the program counter are also saved into the stack. (The PCH, PCL and P are pushed into the stack in that order.) The data thus pushed in the stack points the address of the instruction that follows the interrupt handling routine inserted. When an RTI instruction is executed, the program counter loads the address data from the stack and the program returns from the interrupt handling routine. At the same time, the value of the status register (P) is also restored from the stack.

(c) BRK instruction

When a BRK instruction is executed, the contents of the program counter are saved into the stack after completion of the instruction. The saved data in the stack points the address of the BRK instruction + 2. (The PCH, PCL and P are pushed into the stack in that order.)

When an RTI instruction is executed, the program returns to the address "BRK + 2". At the same time, the value of the status register (P) is also restored from the stack.

2.1.4 Stack Pointer (S)

The stack pointer (S) is an 8bit register which contains the loworder 8 bits of the highest address of a free stack area. The stack pointer is decremented after the data is pushed into the stack, and incremented before it is pulled from the stack. After a system reset occurs, the contents of the stack pointer become invalid. The initial routine must set appropriate data in the stack pointer. When the stack pointer is output onto the address bus, the high byte should always be 21_{16}. Therefore, the stack area of the HuC6280 consists of up to 256 bytes at logical addresses from $21FF_{16}$ to 2100_{16}.

2.1.5 Status Register (P)

The status register (P) is an 8bit register. Its bit organization is shown in Fig. 2-1-1.

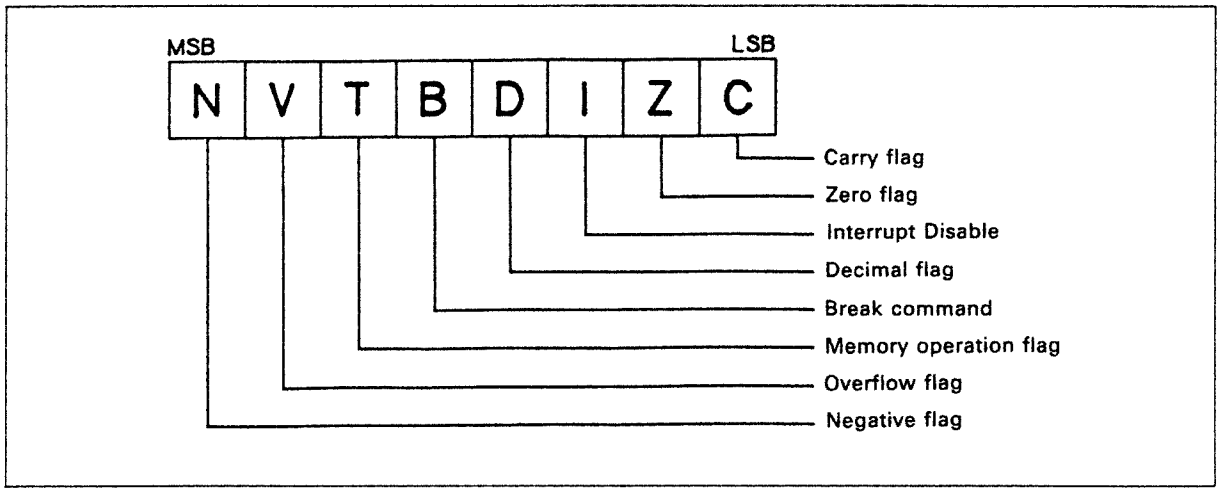


Fig. 2-1-1 Bit Organization of Status Register (P)

The status register, as its name implies, shows the status of the CPU. Each bit shows different status and is independent from each other. For instructions that affect these bits, see Section 6 INSTRUCTION SET SUMMARY.

The contents of the status register are automatically pushed into the stack when an interrupt occurs or a BRK instruction is executed. The RTI instruction causes the status register to be restored from the stack.

(1) Carry Flag (C)

The Carry flag (C) is set if a carry occurs out of the MSB of the result for ADC, or if a borrow occurs during SBC, CMP, CPX or CPY. Otherwise, it is cleared.

The Carry flag is also affected by a shift or rotate instruction.

The Carry flag is set by the SEC instruction, and reset by the CLC instruction.

(2) Zero Flag (Z)

The Zero flag (Z) is set if the result of an ALU operation equals zero or the data of a transfer instruction equals zero.

(3) Interrupt Disable (I)

The Interrupt Disable (I) is set when the system is reset, an interrupt occurs, or a BRK instruction is executed. It is set by the SEI instruction, and reset by the CLI instruction.

When the Interrupt Disable is "1", an interrupt by $\overline{IRQ1}$ or $\overline{IRQ2}$, or by the timer cannot occur.

(4) Decimal Flag (D)

The Decimal flag (D) is reset when the system is reset, an interrupt occurs, or a BRK instruction is executed. It is set by the SED instruction, and reset by the CLD instruction.

When the Decimal flag is "1", an ADC or an SBC instruction performs a decimal operation. In this case, the instruction cycle is supplemented by one additional cycle for decimal adjustment.

(5) Break Command (B)

The Break Command (B) is a status which is valid only when an interrupt occurs by $\overline{IRQ2}$ or a BRK instruction is executed.

When an $\overline{IRQ2}$ interrupt occurs or a BRK instruction is executed, the CPU reads the low byte vector address at logical address $FFF6_{16}$ and the high byte vector address at logical address $FFF7_{16}$, then executes a subroutine. In this case, the Break Command in the status register (P) to be saved into the stack is set to "0" for the $\overline{IRQ2}$ interrupt or to "1" for the BRK instruction.

The Break Command has no meaning except in the above cases, but is set to "1" if read with the PHP or the PLA instruction.

(6) Memory Operation Flag (T)

The Memory Operation flag (T) is set by the SET instruction. The flag affects the next instruction.

If the instruction following the SET is AND, OR, EOR or ADC, its operation is performed for the memory area on page 0 which is addressed indirectly by register X. If the instruction following the SET is any other than these four, the SET has no meaning.

The Memory Operation flag is set by the SET instruction, and reset in the fetch cycle of the next instruction.

The Memory Operation flag is saved into the stack when an interrupt occurs. Therefore, even if the interrupt handling routine is placed between the SET and the next instruction, the ALU instruction works as a memory operation instruction.

When the status register is read with a PHP or PLA instruction, the Memory Operation flag is set to "0". The flag saved in the stack is set to "1" only when a special interrupt occurs (see above).

(7) Overflow Flag (V)

When an operation with a sign is performed, the Overflow flag (V) is available. The HuC6280 uses the adder in the ALU to perform all kinds of add and subtract operations. In a subtract operation, the two's complement of the minuend is placed into the adder. If one of the two values that are placed into the adder is positive and the other negative, then the Overflow flag is reset. If both values are positive, the flag is set when bit 7 of the result of the add operation is "1", and reset when it is "0". If both values are negative, the flag is set when bit 7 of the result of the add operation is "0", and reset when it is "1".

The Overflow flag is reset by the CLV instruction.

When a BIT, TRB, TSB or TST instruction is executed, the data in bit 6 of memory is set in the Overflow flag.

(8) Negative Flag (N)

The Negative flag (N) is set or reset depending on bit 7 of the result of an ALU instruction. It is set if bit 7 is "1", and reset if it is "0".

When the BIT, TRB, TSB or TST instruction is executed, the data in bit 7 of the memory is set in the Negative flag.

2.1.6 Registers SH, DH and LH

The source high (SH), destination high (DH) or length high (LH) is a specialpurpose register which functions only when a block transfer instruction (TII–TDD) is executed. None of them can be read or written by any HuC6280 instruction.

The source high register is paired with register X as the low byte of a 16bit source address.

The destination high register is paired with register Y as the low byte of a 16-bit destination address.

The length high register is paired with the ACC as the low byte of a 16-bit downcounter which counts the length of a block to be transferred. Counting is on a byte basis.

- Operation with block transfer instructions

  At the beginning of a block data transfer instruction, the contents of the ACC, register X and register Y are saved in the stack. Then the source address, destination address and length of the target block are loaded to the (SH, X), (DH, Y) and (LH, ACC), respectively.

  The source address and the destination address used in the block transfer mode are specified for each instruction (TII–TDD) as described in Table 2-5-1. The transfer block length is specified by the length parameter. If length = 0000_{16}, 65536 bytes of data are transferred.

  While a block transfer instruction is executing, one byte of data is transferred from the source address to the destination address in six bus cycles. The number of cycles required for execution of a block transfer instruction is (17 + 6x), where x is the length.

  At the end of the block transfer instruction, the contents of the ACC, register X and register Y are returned from the stack.

  A block transfer instruction uses 3 bytes of stack as mentioned above.

Table 2-1-1 Source Address and Destination Address in Block Transfer Instructions

Mnemonic	Source address (SH, X)	Destination address (DH, Y)
TII	Post-increment	Post-increment
TIN	Post-increment	Fixed
TIA	Post-increment	① Post-increment ② Post-decrement } Alternated NOTE
TAI	① Post-increment ② Post-decrement } Alternated NOTE	Post-increment
TDD	Post-decrement	Post-decrement

NOTE: Post-increment precedes post-decrement.

2.2 Mapping Register (MPR)

The HuC6280 contains eight mapping registers (MPR0 – MPR7). A mapping register is an 8bit register which converts a 16bit logical address into a 21bit physical address.

As shown in Fig. 2-2-1, the mapping register (MPR) consists of eight registers (MPR0 – MPR7), an output selector flipflop, and an I/O control. In the figure,H7–H5refer to the highorder three bits of H-BUS. These three bits are used to select a mapping register (MPR0 – MPR7). They are replaced with 8 bits of data from the selected mapping register and are then converted into 21 bits of physical address.

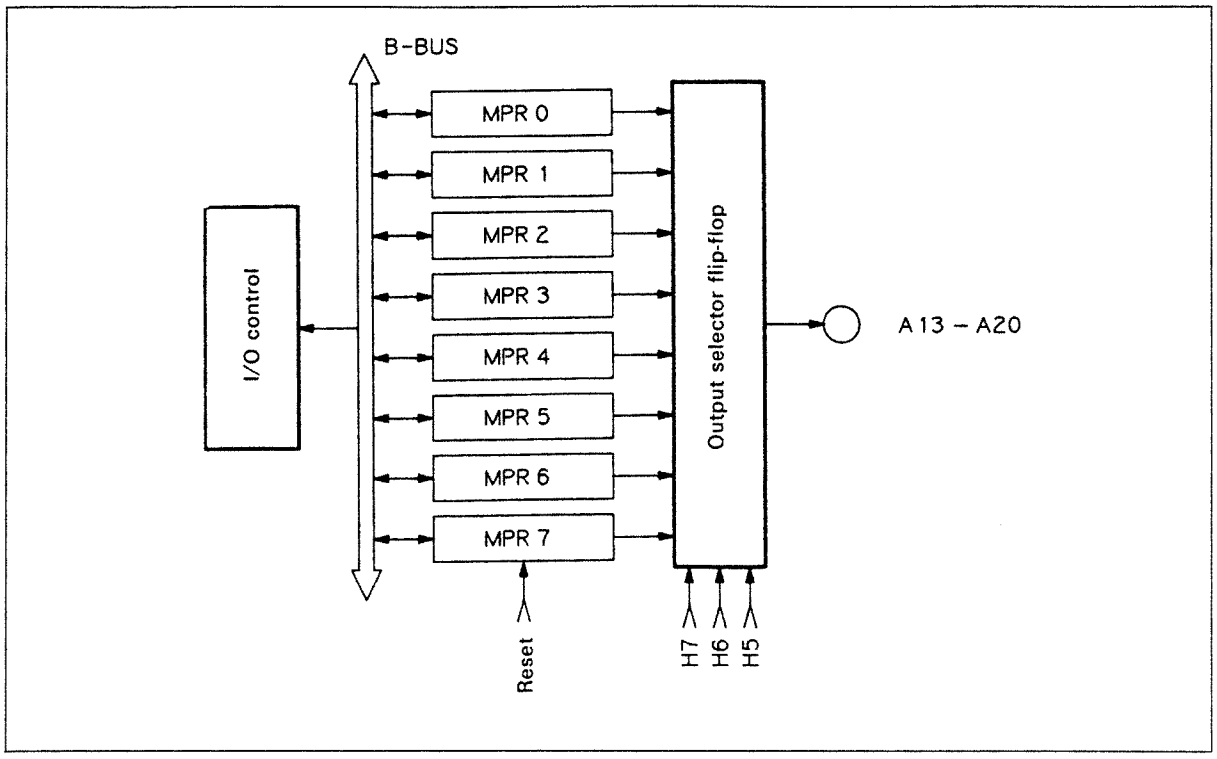


Fig. 2-2-1 Organization of Mapping Register (MPR)

The relation between the high-order 3 bits of logical address and mapping registers selected by them is listed in Table 2-2-1. The contents of a selected mapping register are output as a physical address to A13 – A20.

Table 2-2-1 Selection of Mapping Registers

H7	H6	H5		H7	H6	H5	
0	0	0	MPR0	1	0	0	MPR4
0	0	1	MPR1	1	0	1	MPR5
0	1	0	MPR2	1	1	0	MPR6
0	1	1	MPR3	1	1	1	MPR7

At a system reset, MPR7 is set to 0 ("00_{16}"). Then the HuC6280 reads the low byte address at logical address $FFFE_{16}$ and the high byte address at logical address $FFFF_{16}$, and starts. Since MPR7 = 00_{16}, the program reads the low byte address at physical address $001FFE_{16}$ and the high byte address at $001FFF_{16}$, and starts.
A system reset does not initialize the rest of the mapping registers (MPR0 – MPR6). The initial routine must be used to initialize these registers.
The TMAi (i = 0 – 7) instruction is used to read data from a mapping register. The TAMi (i = 0 – 7) instruction is used to write data to the mapping register. The TMAi or TAMi is a 2byte instruction, the second byte being used to specify the mapping register to be selected. The bit organization of the second byte is such that the bit corresponding to a mapping register number to be selected is set to "1" and the rest to "0". Table 2-2-2 shows assignment of the TMAi/TAMi bits to the mapping registers.

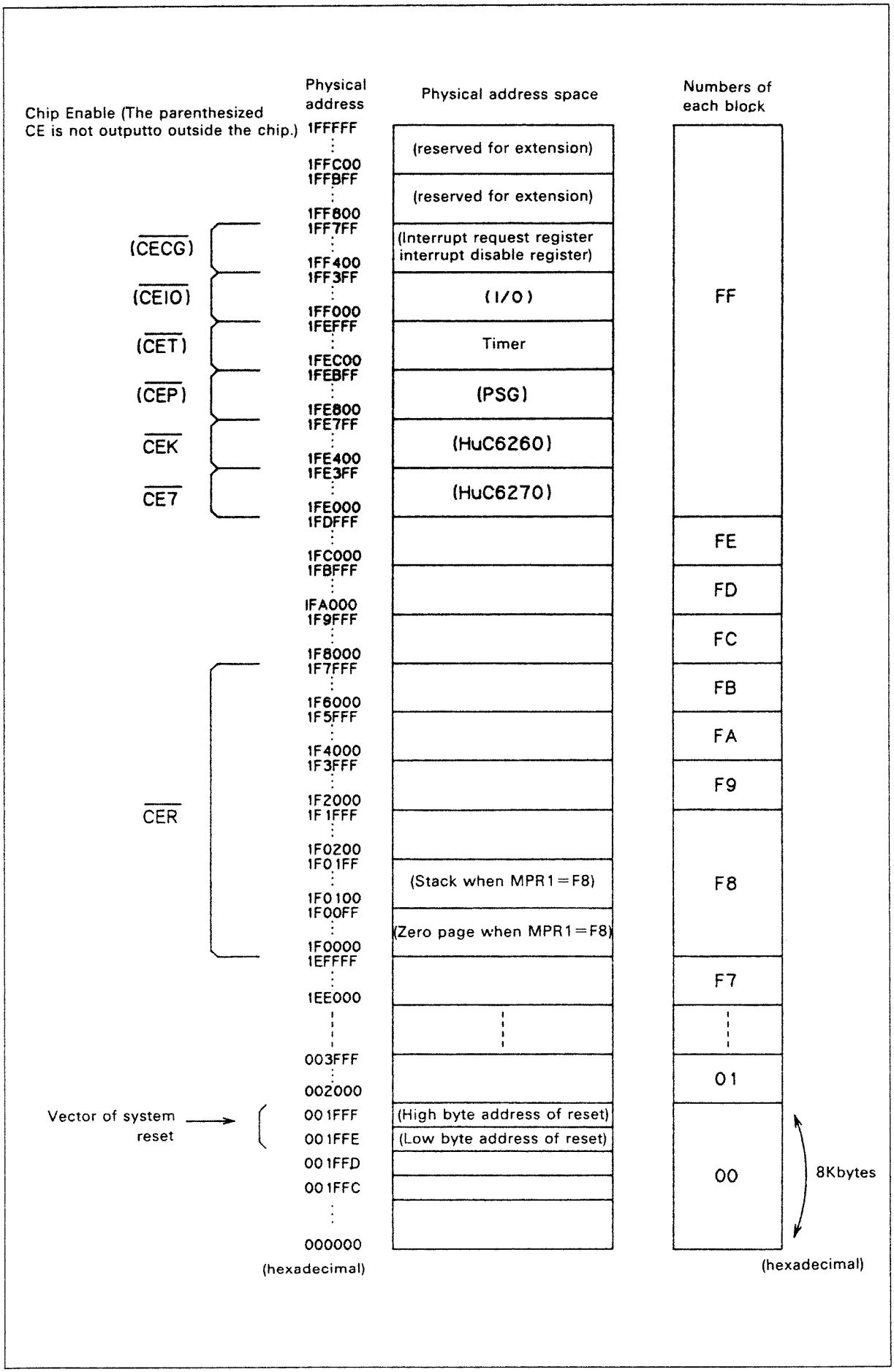
Table 2-2-2 Mapping Register Selection by TMAi/TAMi

Mapping register	Second byte (binary) MSB LSB
MPR 1	0 0 0 0 0 0 0 1
MPR 1	0 0 0 0 0 0 1 0
MPR 2	0 0 0 0 0 1 0 0
MPR 3	0 0 0 0 1 0 0 0
MPR 4	0 0 0 1 0 0 0 0
MPR 5	0 0 1 0 0 0 0 0
MPR 6	0 1 0 0 0 0 0 0
MPR 7	1 0 0 0 0 0 0 0

Address line A20 of the output selector flipflop outputs (A13 – A20) provides "H" level in the write cycle of an immediate data transfer instruction (ST0, ST1, ST2) to the HuC6270. Special code (shown in Table 2-2-3) is output to A0 and A1 in response to each immediate data transfer instruction in its write cycle.

Table 2-2-3 Address Output to HuC6270

IR \ Address	A1	A0
ST 0	0	0
ST 1	1	0
ST 2	1	1



Chip Enable Allocation

2.3 Memory Space

The HuC6280 has 64K bytes of logical address space and 2M bytes of physical address space. The relation between logical and physical addresses depends on the contents of the mapping registers MPR0 – MPR7. Fig. 2-8-1 gives an example of correspondence between the logical address space and the physical address space.

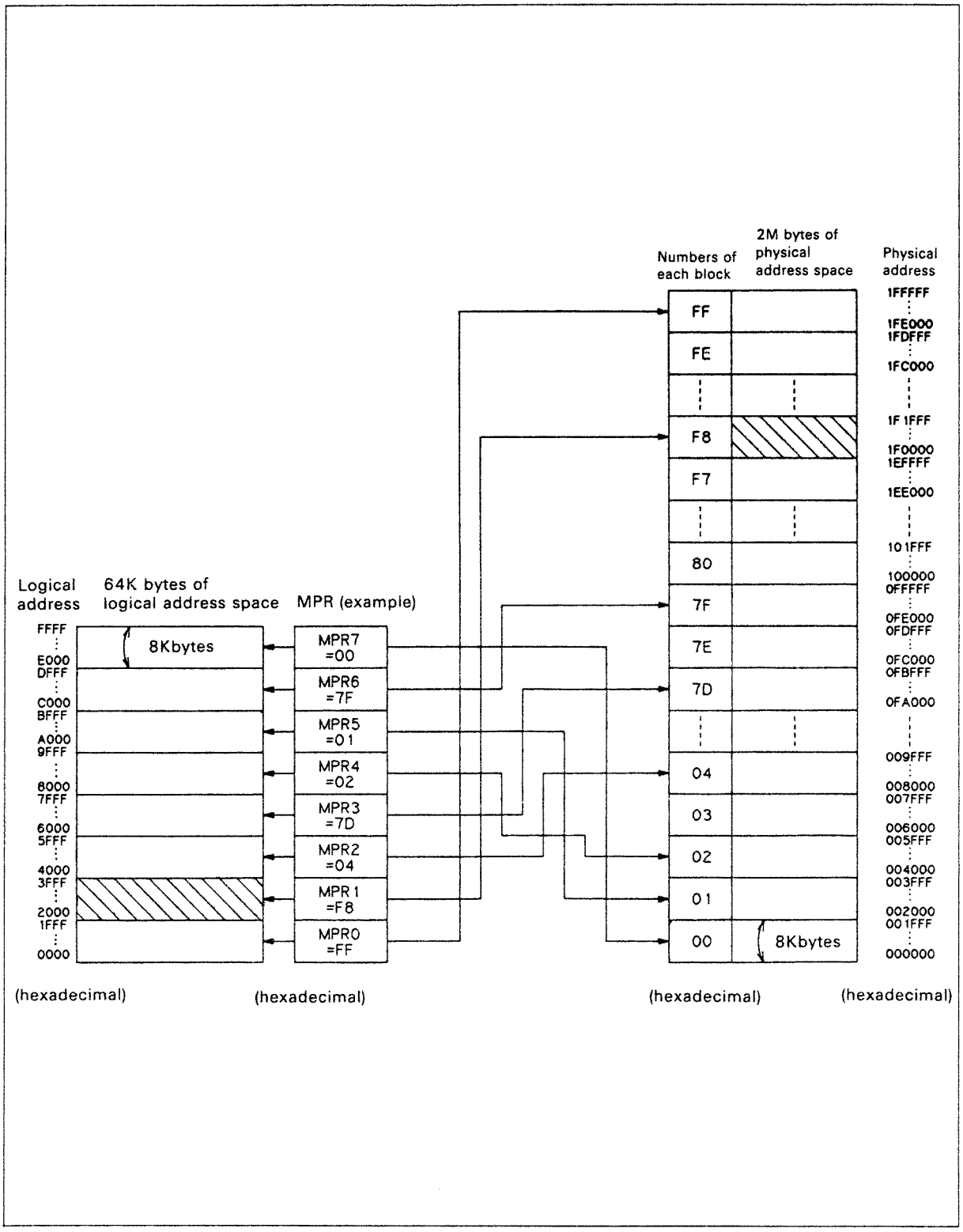


Fig. 2-3-1 Example of Correspondence between Logical

As shown in Fig. 2-3-1, the address space is organized in 8K-byte units (8K bytes = one unit of address space A0 – A12).

The physical address space consists of 256 units of 8K-byte blocks, which are numbered "00_{16}" to "FF_{16}" (low to high). Each block is selected by a mapping register (MPR0 – MPR7), and relocated in the logical address space.

Example: In Fig. 2-3-1, MPR1 =$F8_{16}$. Therefore, the 8K byte block at physical addresses from $1F0000_{16}$ to $1F1FFF_{16}$ is relocated at logical addresses from 2000_{16} to $3FFF_{16}$.

2.4 Interrupt and Break

The HuC6280 allows three external interrupts and two internal interrupts. The external interrupts are the nonmaskable interrupt ($\overline{NMI}$ input), interrupt request 1 ($\overline{IRQ1}$ input), and interrupt request 2 ($\overline{IRQ2}$ input). The internal interrupts are the timer interrupt and interrupt by software (BRK instruction).

These interrupt inputs are sampled in the last bus cycle of one instruction when it is executed. Fig. 2-9-1 shows the vector table and priority of the interrupts. System reset has higher priority than any of the interrupts.

The CPU reads the status of an $\overline{IRQ1}$ input, $\overline{IRQ2}$ input or timer interrupt from the interrupt request resgister into the ACC. The interrupt disable register allows these interrupts to be individually disabled. Moreover, Interrupt Disable (I) can disable all of the three interrupts at the same time.

Figs. 2-4-2 and 2-4-3 represent the organization of the interrupt request register, interrupt disable register and interrupt circuit, respectively.

2.4.1 Nonmaskable Interrupt ($\overline{NMI}$ Input)

Nonmaskable Interrupt ($\overline{NMI}$) is an external interrupt driven by edge sense. It cannot be masked by Interrupt Disable (I).

(1) Interrupt Caused by $\overline{NMI}$ Input

A transition of the $\overline{NMI}$ input from "H" to "L" level causes a nonmaskable interrupt. After the instruction in progress is completed, the CPU reads the low byte at logical address $FFFC_{16}$, and the high byte at logical address $FFFD_{16}$, and calls the interrupt handling subroutine.

(2) Operation of Interrupt Handling Subroutine In the subroutine call sequence, the contents of the program counter (PCH, PCL) and the status register (P) are pushed into the stack in that order (i.e., PCH, PCL and P). In the write cycle to the stack, the stack pointer (S) is postdecremented. The break command (B) in the status register to be pushed into the stack is set to "0".

The CPU sets Interrupt Disable (I) of the status register (P) and resets Decimal (D).

(3) Returning from Interrupt Handling Subroutine

When the RTI instruction is executed, the values of the PCH, PCL and P are returned from the stack.

Then the CPU restarts at the next address at which the interrupt handling subroutine was inserted.

● $\overline{\text{RESET}}$ and BRK instruction, and priority of interrupts

$\overline{\text{RESET}}$ > $\overline{\text{NMI}}$ > BRK instructions > TIMER > $\overline{\text{IRQ1}}$ > $\overline{\text{IRQ2}}$

● Vector table

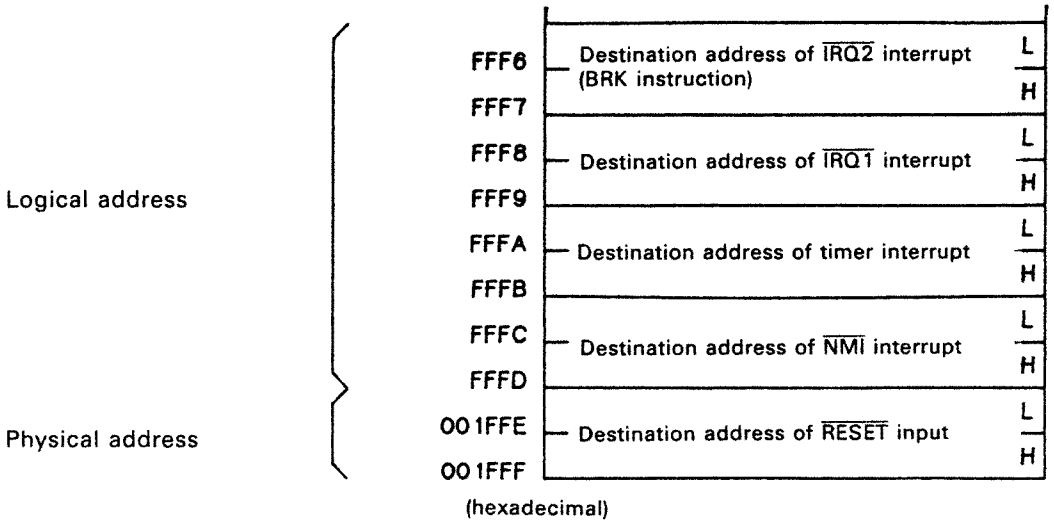


Fig. 2-4-1. Vector Table and Priority

2.4.2 Interrupt Requests 1 and 2 ($\overline{\text{IRQ1}}$ and $\overline{\text{IRQ2}}$)

Interrupt Request 1 ($\overline{\text{IRQ1}}$ input) and Interrupt Request 2 ($\overline{\text{IRQ2}}$ input) are external interrupts driven by level sense. These two are masked when Interrupt Disable (I) of the status register (P) is set.

(1) Interrupt Caused by $\overline{\text{IRQ1}}$ Input

An $\overline{\text{IRQ1}}$ interrupt occurs when the $\overline{\text{IRQ1}}$ input goes "L" with Interrupt Disable (I) reset and $\overline{\text{IRQ1}}$ Disable (IRQ1D) of the interrupt disable register at "0". When the instruction in progress is terminated, the CPU reads the low byte at logical address $FFF8_{16}$ and the high byte at logical address $FFF9_{16}$, and executes the interrupt handling subroutine.

(2) Interrupt Caused by $\overline{IRQ2}$ Input

An $\overline{IRQ2}$ interrupt occurs when the $\overline{IRQ2}$ input goes "L" with Interrupt Disable (I) reset and the $\overline{IRQ2}$ Disable (IRQ2D) of the interrupt disable register at "0". When the instruction in progress is terminated, the CPU reads the low byte at logical address $FFF6_{16}$ and the high byte at logical address $FFF7_{16}$, and executes the interrupt handling subroutine.

(3) Operation of Interrupt Handling Subroutine

In the subroutine call sequence, the values of the program counter (PCH, PCL) and the status register (P) are pushed into the stack in that order (i.e., PCH, PCL and P). In the write cycle to the stack, the stack pointer (S) is postdecremented. The break command (B) in the status register to be pushed into the stack is set to "0".

The CPU sets Interrupt Disable (I) of the status register and resets Decimal (D).

(4) Returning from Interrupt Handling Subroutine

When the RTI instruction is executed, the values of the PCH, PCL and P are returned from the stack. Then the CPU restarts at the next address at which the interrupt handling subroutine was inserted.

2.4.3 Timer Interrupt

A timer interrupt is an internal interrupt which is caused by a borrow of the timer. It is masked when Interrupt Disable (I) of the interrupt disable register is set.

(1) Conditions and Processing for Timer Interrupt

If a timer borrow occurs with Interrupt Disable (I) reset and TIQ Disable (TIQD) of the interrupt disable register at "0", Timer interrupt Request (TIQ) of the interrupt request register is set and a timer interrupt occurs. Then the CPU reads the low byte at logical address $FFFA_{16}$ and the high byte at logical address $FFFB_{16}$, and calls the interrupt handling subroutine.

(2) Operation of Interrupt Handling Subroutine

In the subroutine call sequence, the values of the program counter (PCH, PCL) and the status register (P) are pushed into the stack in that order (i.e., PCH, PCL and P). In the write cycle to the stack, the stack pointer (S) is postdecremented. The break command (B) in the status register to be pushed into the stack is set to "0".

The CPU sets Interrupt Disable (I) of the status register and resets Decimal (D).

(3) Returning from Interrupt Handling Subroutine

When the RTI instruction is executed, the values of the PCH, PCL and P are returned from the stack. Then the CPU restarts at the next address at which the interrupt handling subroutine was inserted.

(4) TIQ Reset

Timer Interrupt Request (TIQ) is reset by the execution of a write cycle to the interrupt request register.

2.4.4 Organization and Functions of Interrupt Request Register

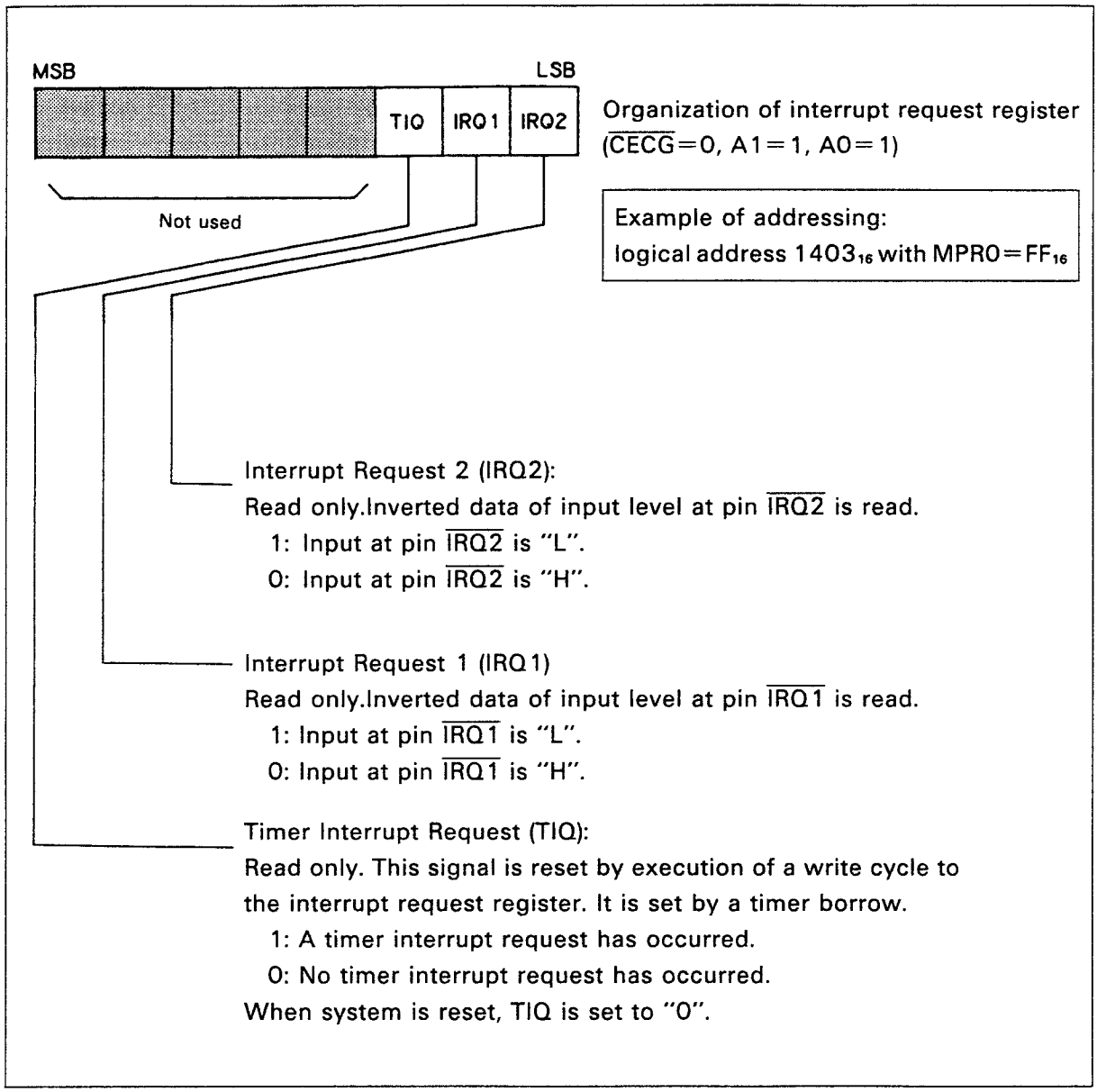


Fig. 2-4-2 Interrupt Request Register

Data can be read from or written to the interrupt request register at physical addresses from $1FF400_{16}$ to $1FF7FF_{16}$ where (A1, A0) = (1, 1).

2.4.5 Organization and Functions of Interrupt Disable Register

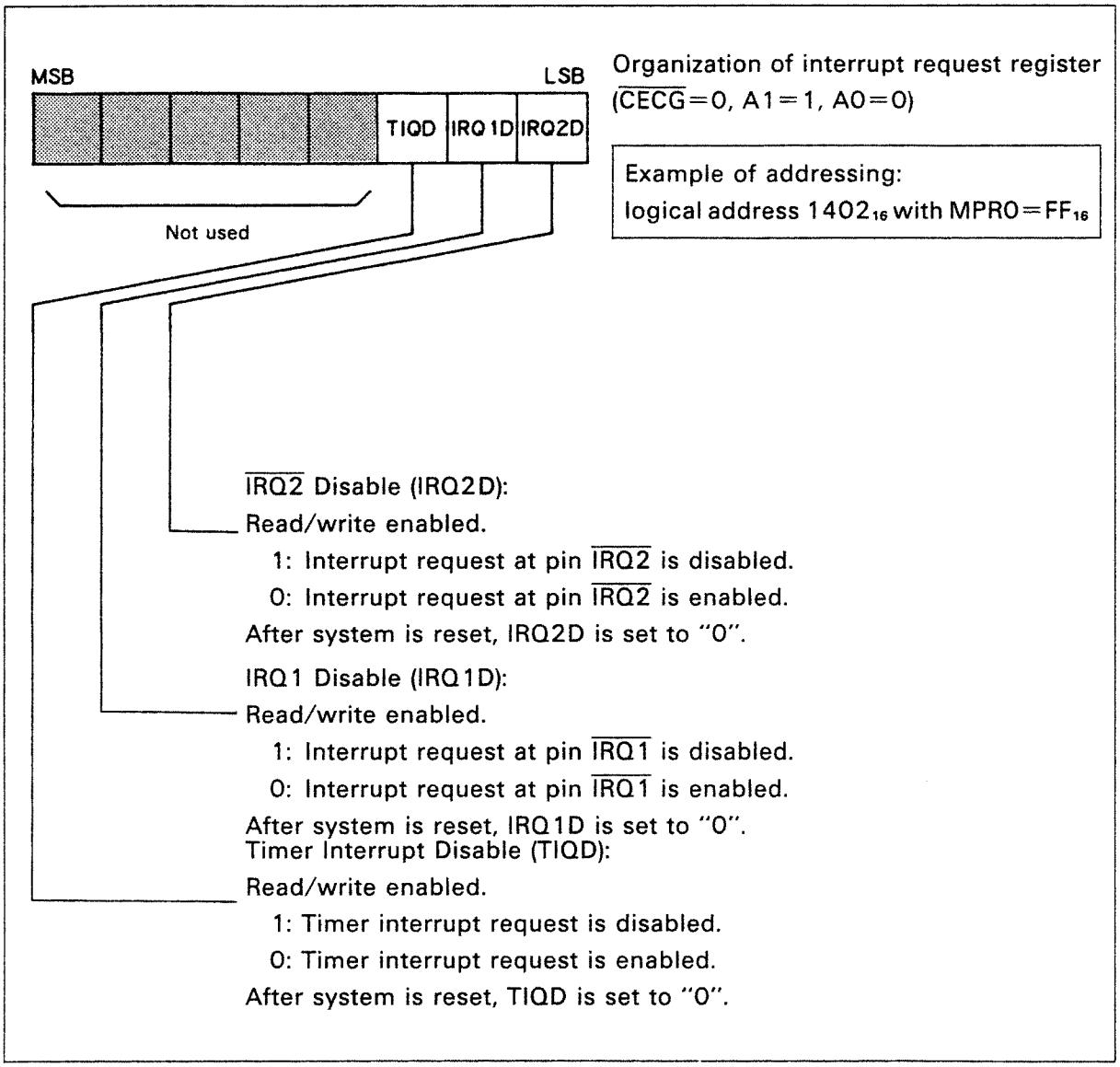


Fig. 2-4-3 Interrupt Disable Register

Data can be read from or written to the interrupt disable register at logical addresses from $1FF400_{16}$ to $1FF7FF_{16}$ where (A1, A0) = (1, 0).

2.4.7 Break (BRK)

Break (BRK) is a pseudointerrupt which is caused by software. It occurs unconditionally whenever a BRK instruction is executed.

(1) Interrupt Caused by BRK Instruction

When a BRK instruction is executed, the CPU reads the low byte at address $FFF6_{16}$ and the high byte at address $FFF7_{16}$, and calls the interrupt handling subroutine.

(2) Operation of Interrupt Handling Subroutine

In the subroutine call sequence, the contents of the program counter (PCH, PCL) and the status register (P) are pushed into the stack in that order (i.e., PCH, PCL and P). In the write cycle to the the stack, the stack pointer (S) is postdecremented.

The break command (B) in the status register to be pushed into the stack is set to "1".

The CPU sets Interrupt Disable (I) and resets Decimal (D).

The vector address which is read from memory is the same as that of Interrupt Request 2 ($\overline{IRQ2}$ input). Therefore, it is necessary to check, using the interrupt handling subroutine, the value of the break command saved in the stack to see whether the interrupt of interest was caused by a BRK instruction or Interrupt Request 2 ($\overline{IRQ2}$).

(3) Returning from Interrupt Handling Subroutine

If an interrupt is caused by a BRK instruction, the value of the program counter (PCH, PCL) to be pushed into the stack is (BRK + 2). When the RTI instruction is executed, the program returns to address (BRK + 2) and the instruction inserted at address (BRK + 1) is ignored. The program assumes the data at (BRK + 2) is an instruction code and restarts processing.

2.5 System Reset

(1) System Resetting Procedure

System can be reset by setting the $\overline{\text{RESET}}$ input pin to "L" level. Returning the pin to "H" level causes the system to start.

The "L" level applied to the pin must be a pulse which has a duration of at least 28 clock cycles with the clock input at OSC1 fully stabilized.

(2) Internal States after System Reset

When the system resumes after a reset, the program reads the low byte at physical address $001FFE_{16}$ and the high byte at physical address $001FFF_{16}$, and starts.

A system reset causes the following internal states:

- Interrupt Disable (I) is set.
- The Decimal flag (D) is reset.
- "00_{16}" is set in the MPR7 mapping register.
- The timer is stopped.
- The interrupt disable register is all reset.
- Timer Interrupt Request (TIQ) is reset.
- Low speed mode is set.
- "H" level is output to the output port (port O).
- The interrupt circuit is initialized.
- The Memory Operation flag (T) is reset.
- The ready state is cleared.
- "H" level is output to the $\overline{\text{RD}}/\overline{\text{WR}}$ pins.
- "L" level is output to the SYNC pin.
- Both the data bus and address bus provide invalid data.
- $\overline{\text{CE7}}$, $\overline{\text{CEK}}$, $\overline{\text{CER}}$ and A20 provide invalid data.
  Only one of these four pins produces "L" level.
- System clock is output to the SX pin.
- "L" level is output to the HSM pin.

2.6 I/O

The HuC6280 contains one 8bit input port (port K) and one 8bit output port (port O).

2.6.1 Port K

Port K consists of 8 bits numbered K0 – K7. Its input is of TTL level interface, with a pullup resistor. Executing a read cycle at physical addresses from $1FF000_{16}$ to $1FF3FF_{16}$ causes K0 – K7 to be read into the device.

2.6.2 Port O

Port O consists of 8 bits numbered O0 – O7, and has a latch. Its output is of a complementary type. When the system is reset, the port provides an "H" level output.
Executing a write cycle at physical addresses from $1FF000_{16}$ to $1FF3FF_{16}$ causes data to be output to O0 – O7.

Example of addressing to I/O port; Physical address 1000_{16} with MPR0 = FF_{16}

2.7 Timer

The HuC6280 has a 7bit timer.

(1) Organization of Timer

Fig. 2- 7-1 shows the organization of the timer. It consists of a 7bit downcounter, a 7bit reload register, a timer control register, and a prescaler.

- Downcounter

  The downcounter (binary) receives the output of the prescaler and counts down.

- Reload Register

  The contents of the reload register (7 bits) are loaded to the downcounter when the Start/Stop flagof the control register changes from "0" to "1" or when a borrow occurs in the downcounter.

- Timer Control Register

  The timer control register (1 bit) is the Timer Start/Stop flag, as shown in Fig. 2- 7-1.

- Prescaler

  The prescaler consists of ten frequency dividers each of which divides the input frequency by 2. The signal from OSC1 is divided by 3 to obtain a clock signal PPS3. PPS3 is further divided by 1,024 (decimal) and supplied to the downcounter.

(2) Writing Data to Reload Register

Executing a write cycle at physical addresses from $1FEC00_{16}$ to $1FEFFF_{16}$ causes bits 6 to 0 of write data to be loaded into the reload register.

(3) Writing Data to Timer Control Register

Executing a write cycle at physical addresses from $1FEC00_{16}$ to $1FEFFF_{16}$ causes bit 0 of write data to be loaded into the timer control register.

(4) Reading Data from Timer

Executing a read cycle at physical addresses from $1FEC00_{16}$ to $1FEFFF_{16}$ causes the contents of the downcounter to be loaded to bits 60.

(5) Timer Operation

- System reset

  Once the system has been reset, the data of both the reload register and the downcounter are invalid. The Timer Start/Stop flag of the timer control register is reset.

- Timer operation

  When a timer interval is set in the reload register and the Timer Start/Stop flag is set, the data in the reload register is loaded to the downcounter and the timer starts countingdown. If a borrow occurs in the downcounter, Timer Interrupt Request (TIQ) of the interrupt request register is set. The CPU is requested to handle a timer interrupt if Timer Interrupt Disable (TIQD) is "0" and Interrupt Disable (I) is "0". At the same time, the downcounter is loaded with the data of the reload register to have the timer continue countingdown. Once an interrupt request has occurred and has been handled, Timer Interrupt Request (TIQ) must be reset by executing a write cycle to the interrupt request register.

- Interval control

  The contents of the reload register can be updated while the timer is counting down. By doing so it is possible to control the interval of interrupts.

  When "0" is written in the Timer Start/Stop flag, the time stops and the prescaler is reset.

  If a 21.48 MHz clock is used, the prescaler output frequency (f) is given as follows:

  $f = 21.48\ \text{MHz} \div 3 \div 1024 = 6.992\ \text{KHz}$

  Then the timer can be used for generating a time interval of interrupt request from 143 μsec to 18.3 msec.

- **CAUTION:**

  When reading data from the downcounter, the count value can become invalid. It is important to read data twice and compare two values with each other to assure data.

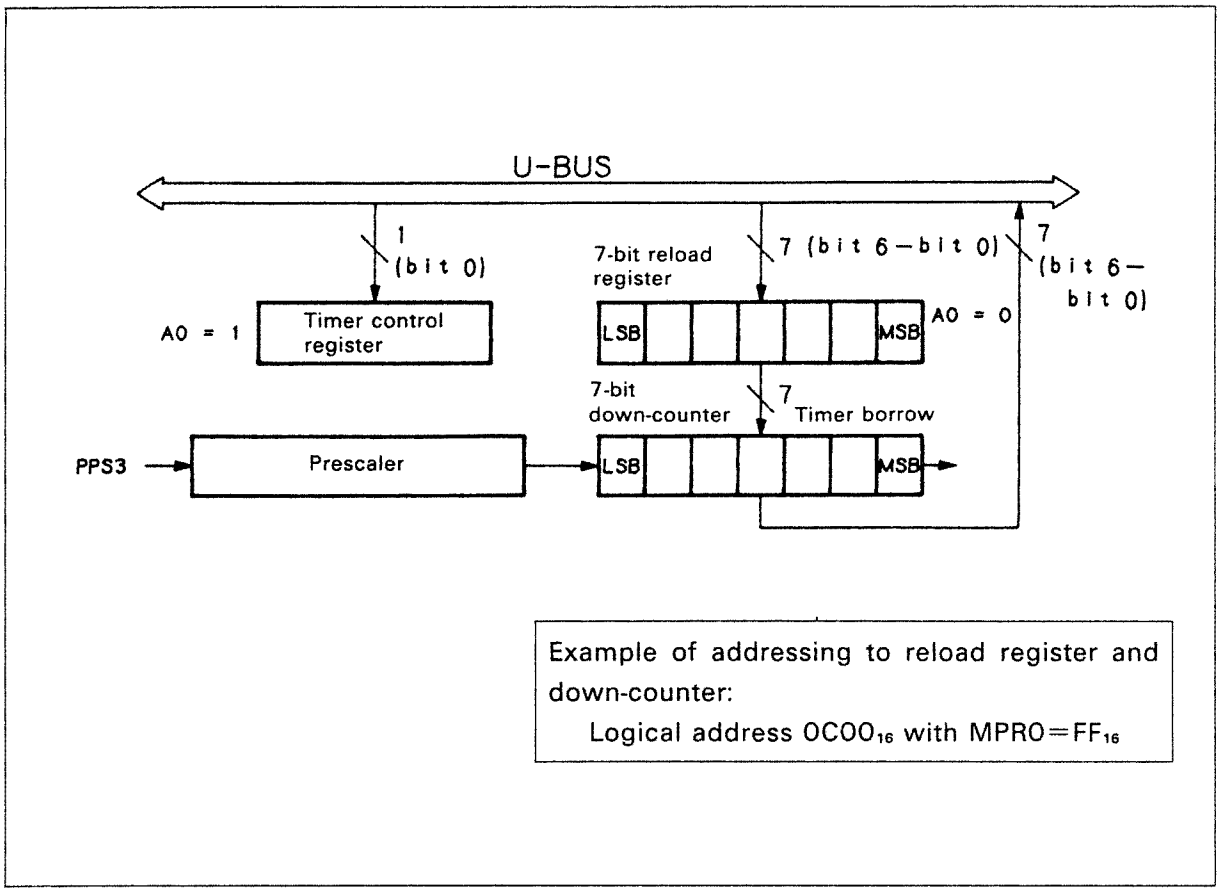


Fig. 2-7-1 Organization of Timer

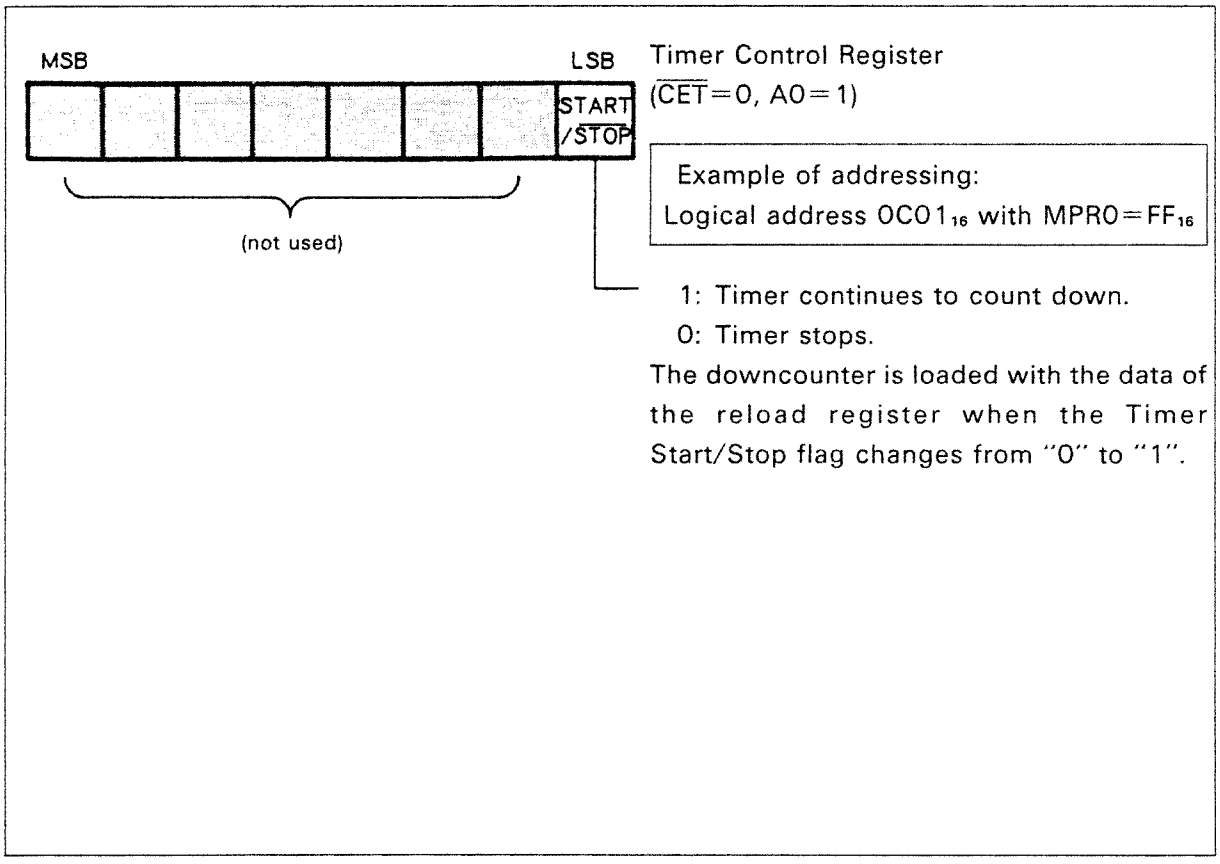


Fig. 2-7-2 Organization of Timer Control Register

2.8 Programmable Sound Generator (PSG)

The HuC6280 contains a programmable sound generator (PSG). For the operation of the PSG, see its manual. The following signals flow from the CPU to the PSG for interfacing:

- D0 – D7 (bidirectional data bus)
- $\overline{RD}$, $\overline{WR}$, $\overline{CEP}$ (PSG Chip Enable)
- A0 –A3 (4 bits of address)
- CLKP $\left(f = \frac{\text{clock (OSC1)}}{3}\right)$
- $\overline{RESET}$
- SX (output signal of system clock S3)

2.9 Device/Register Addresses

No.	Physical address (hexadecimal)	Example of addressing (hexadecimal)		Device name (Register name)	Organization and function of bit (hexadecimal)
		MPR	Logical address		
1	1FE000 ⋮ 1FE3FF	MPR0=FF	0000 0002 0003	HuC6270 (A1 – A0)	Logical address / Instruction for HuC6270 0000 ST 0 0002 ST 1 0003 ST 2
2	1FE400 ⋮ 1FE7FF	MPR0=FF	0400 ⋮ 0407	HuC6260 (A2 – A0)	———
3	1FE800 ⋮ 1FEBFF	MPR0=FF	0800 ⋮ 080F	PSG (A3 – A0)	———
4	1FEC00 ⋮ 1FEFFF (A0=1)	MPR0=FF	0C01	Timer control register (write only)	7 6 5 4 3 2 1 0 Reserved 1: Timer start 0: Timer stop
	1FEC00 ⋮ 1FEFFF (A0=0)	MPR0=FF	0C00	Write cycle: Timer reload register	7 6 5 4 3 2 1 0 Timer interval setting (7 bits) Reserved
				Read cycle: Timer downcounter	7 6 5 4 3 2 1 0 Timer downcounter (7 bits) Reserved
5	1FF000 ⋮ 1FF3FF	MPR0=FF	1000	Write cycle: port O	O7 – O0 Output to O7 – O0
				Read cycle: port K	Input from K7 – K0
6	1FF400 ⋮ 1FF7FF (A1=1, A0=0)	MPR0=FF	1402	Interrupt disable register (read/write enabled)	7 6 5 4 3 2 1 0 Reserved IRQ2D 1: Disable (bit 0) 0: Enable IRQ1D 1: Disable (bit 1) 0: Enable TIQD 1: Disable (bit 2) 0: Enable
	1FF400 ⋮ 1FF7FF (A1=1, A0=1)	MPR0=FF	1403	Interrupt request register (read; TIQ reset at write)	7 6 5 4 3 2 1 0 Reserved IRQ2 1: Interrupt request (bit0) 0: No interrupt request IRQ1 1: Interrupt request (bit1) 0: No interrupt request TIQ 1: Interrupt request (bit2) 0: No interrupt request
7	1F0000 ⋮ 1FF7FF	MPR1=F8 ⋮ FB	2000 ⋮ 3FFF	DATA MEMORY (RAM)	

3. INSTRUCTION MAP

IR7	IR6	IR5	IR4	IR3=0 IR2=0 IR1=0 IR0=0	IR3=0 IR2=0 IR1=0 IR0=1	IR3=0 IR2=0 IR1=1 IR0=1	IR3=0 IR2=0 IR1=1 IR0=0	IR3=0 IR2=1 IR1=1 IR0=0	IR3=0 IR2=1 IR1=1 IR0=1	IR3=0 IR2=1 IR1=0 IR0=1	IR3=0 IR2=1 IR1=0 IR0=0	IR3=1 IR2=1 IR1=0 IR0=0	IR3=1 IR2=1 IR1=0 IR0=1	IR3=1 IR2=1 IR1=1 IR0=1	IR3=1 IR2=1 IR1=1 IR0=0	IR3=1 IR2=0 IR1=0 IR0=0	IR3=1 IR2=0 IR1=0 IR0=1	IR3=1 IR2=0 IR1=1 IR0=1	IR3=1 IR2=0 IR1=1 IR0=0
0	0	0	0	BRK IMPLED 1 8	ORA (IND.X) 2 7	STO IMM 2 4	SXY IMPLED 1 3	ASL ZP 2 6	RMBO ZP 2 7	ORA ZP 2 4	TSB ZP 2 6	TSB ABS 3 7	ORA ABS 3 5	BBRO ZP 3 6	ASL ABS 3 7	ASL ACC 1 2		ORA IMM 2 2	PHP IMPLED 1 3
			1	BPL REL 2 2	ORA (IND).Y 2 7	ST1 IMM 2 4	ORA (IND) 2 7	ASL ZP.X 2 6	RMB1 ZP 2 7	ORA ZP.X 2 4	TRB ZP 2 6	TRB ABS 3 7	ORA ABS.X 3 5	BBR1 ZP 3 6	ASL ABS.X 3 7	INC ACC 1 2		ORA ABS.Y 3 5	CLC IMPLED 1 2
		1	1	BMI REL 2 2	AND (IND).Y 2 7		AND (IND) 2 7	ROL (IND) 2 7	ROL ZP.X 2 6	RMB3 ZP 2 7	AND ZP.X 2 4	BIT ABS.X 3 5	AND ABS.X 3 5	BBR3 ZP 3 6	ROL ABS.X 3 7	DEC ACC 1 2		AND ABS.Y 3 5	SEC IMPLED 1 2
			0	JSR ABS 3 7	AND (IND.X) 2 7	ST2 IMM 2 4	SAX IMPLED 1 3	ROL ZP 2 6	RMB2 ZP 2 7	AND ZP 2 4	BIT ZP 2 4	BIT ABS 3 5	AND ABS 3 5	BBR2 ZP 3 6	ROL ABS 3 7	ROL ACC 1 2		AND IMM 2 2	PLP IMPLED 1 4
	1	0	0	RTS IMPLED 1 7	ADC (IND.X) 2 7		CLA IMPLED 1 2	ROR ZP 2 6	RMB6 ZP 2 7	ADC ZP 2 4	STZ ZP 2 4	JMP (ABS) 3 7	ADS ABS 3 5	BBR6 ZP 3 6	ROR ABS 3 7	ROR ACC 1 2		ADC IMM 2 2	PLA IMPLED 1 4
			1	BVS REL 2 2	ADC (IND),Y 2 7	TII IMPLED 7 *	ADC (IND) 2 7	ROR ZP.X 2 6	RMB7 ZP 2 7	ADC ZP.X 2 4	STZ ZP.X 2 4	JMP (ABS.X) 3 7	ADC (ABS.X) 3 5	BBR7 ZP 3 6	ROR ABS.X 3 7	PLY IMPLED 1 4		ADC ABS.Y 3 5	SEI IMPLED 1 2
		1	1	BVC REL 2 2	EOR (IND). Y 2 7	TAMi IMPLED 2 5	EOR (IND) 2 7	LSR ZP.X 2 6	RMB5 ZP 2 7	EOR ZP.X 2 4	CSL IMPLED 1 3		EOR ABS.X 3 5	BBR5 ZP 3 6	LSR ABS.X 3 7	PHY IMPLED 1 3		EOR ABS.Y 3 5	CLI IMPLED 1 2
			0	RT1 IMPLED 1 7	EOR (IND.X) 2 7	TRAi IMPLED 2 4	SAY IMPLED 1 3	LSR ZP 2 6	RMB4 ZP 2 7	EOR ZP 2 4	BSR REL 2 8	JMP ABS 3 4	EOR ABS 3 5	BBR4 ZP 3 6	LSR ABS 3 7	LSR ACC 1 2		EOR IMM 2 2	PHA IMPLED 1 3
1	1	0	0	CPY IMM 2 2	CMP (IND.X) 2 7	TDD IMPLED 7 *	CLY IMPLED 1 2	DEC ZP 2 6	SMB4 ZP 2 7	CMP ZP 2 4	CPY ZP 2 4	CPY ABS 3 5	CMP ABS 3 5	BBS4 ZP 3 6	DEC ABS 3 7	DEX IMPLED 1 2		CMP IMM 2 2	INY IMPLED 1 2
			1	BNE REL 2 2	CMP (IND). Y 2 7	TIN IMPLED 7 *	CMP (IND) 2 7	DEC ZP.X 2 6	SMB5 ZP 2 7	CMP ZP.X 2 4	CSH IMPLED 1 3		CMP ABS.X 3 5	BBS5 ZP 3 6	DEC ABS.X 3 7	PHX IMPLED 1 3		CMP ABS.Y 3 5	CLD IMPLED 1 2
		1	1	BEQ REL 2 2	SBC (IND).Y 2 7	TAI IMPLED 7 *	SBC (IND) 2 7	INC ZP.X 2 6	SMB7 ZP 2 7	SBC ZP.X 2 4	SET IMPLED 1 2		SBC ABS.X 3 5	BBS7 ZP 3 6	INC ABS.X 3 7	PLX IMPLED 1 4		SBC ABS.Y 3 5	SED IMPLED 1 2
			0	CPX IMM 2 2	SBC (IND.X) 2 7	TIA IMPLED 7 *		INC ZP 2 6	SMB6 ZP 2 7	SBC ZP 2 4	CPX ZP 2 4	CPX ABS 3 5	SBC ABS 3 5	BBS6 ZP 3 6	INC ABS 3 7	NCP IMPLED 1 2		SBC IMM 2 2	INX IMPLED 1 2
	0	0	0	LDY IMM 2 2	LDA (IND.X) 2 7	TST IMM ZP.X 3 7	LDX IMM 2 2	LDX ZP 2 4	SMB2 ZP 2 7	LDA ZP 2 4	LDY ZP 2 4	LDY ABS 3 5	LDA ABS 3 5	BBS2 ZP 3 6	LDX ABS 3 5	TAX IMPLED 1 2		LDA IMM 2 2	TAY IMPLED 1 2
			1	BCS REL 2 2	LDA (IND), Y 2 7	TST IMM ABS.X 4 8	LDA (IND) 2 7	LDX ZP.Y 2 4	SMB3 ZP 2 7	LDA ZP 2 7	LDA ZP.X 2 4	LDY ABS.X 3 5	LDA ABS.X 3 5	BBS3 ZP 3 6	LDX ABS.Y 3 5	TSX IMPLED 1 2		LDA ABS.Y 3 5	CLV IMPLED 1 2
		0	1	BCC REL 2 2	STA (IND). Y 2 7	TST IMM ABS 4 8	STA (IND) 2 7	STX ZP.Y 2 4	SMB1 ZP 2 7	STA ZP.X 2 4	STY ZP.X 2 4	STZ ABS 3 5	STA ABS.X 3 5	BBS1 ZP 3 6	STZ ABS.X 3 5	TXS IMPLED 1 2		STA ABS.Y 3 5	TYA IMPLED 1 2
			0	BRA REL 2 4	STA (IND.X) 2 7	TST IMM ZP 3 7	CLX IMPLED 1 2	STX ZP 2 4	SMBO ZP 2 7	STA ZP 2 4	STY ZP 2 4	STY ABS 3 5	STA ABS 3 5	BBSO ZP 3 6	STX ABS 3 5	TXA IMPLED 1 2		BIT IMM 2 2	DY IMPLED 1 2

*17+16z