HD63B09, HD63C09
CMOS MPU (Micro Processing Unit)

Description

The HD6309 is the highest 8-bit microprocessor of HMCS6800 family, which is compatible with the conventional HD6809.

The HD6309 has hardware and software features which make it an ideal processor for higher level language execution or standard controller applications.

The HD6309 is complete CMOS device and its power dissipation is extremely low. Moreover, the SYNC and CWAI instruction makes low power application possible.

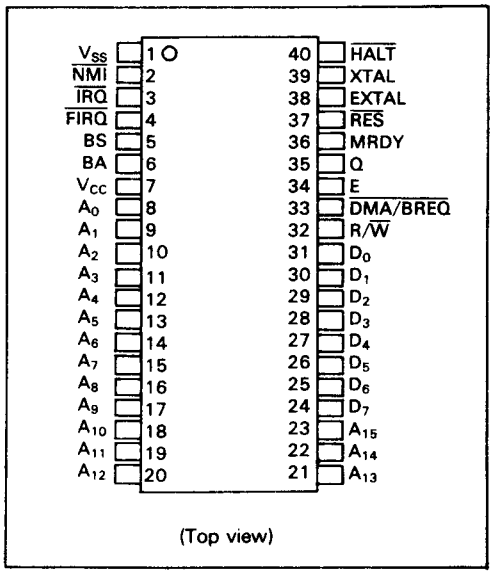
Features

- Hardware
  - −Interfaces with all HMCS6800 peripherals
  - −DMA transfer with no auto-refresh cycle
- Software: object code compatible with the HD6809
- Low power consumption mode (Sleep mode)
  - −SYNC state of SYNC Instruction
  - −WAIT state of CWAI Instruction
- On chip oscillator
- Wide operation range: f = 0.5 to 3 MHz (V_{CC} = 5 V ±10%)

Type of Products

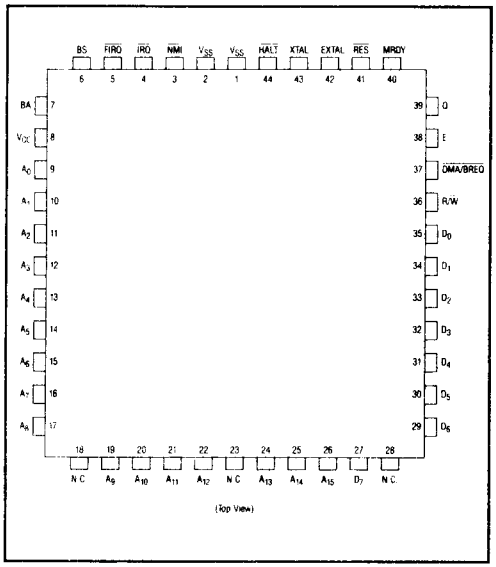
Type No.	Bus Timing
HD63B09	2.0 MHz
HD63C09	3.0 MHz

Pin Arrangement



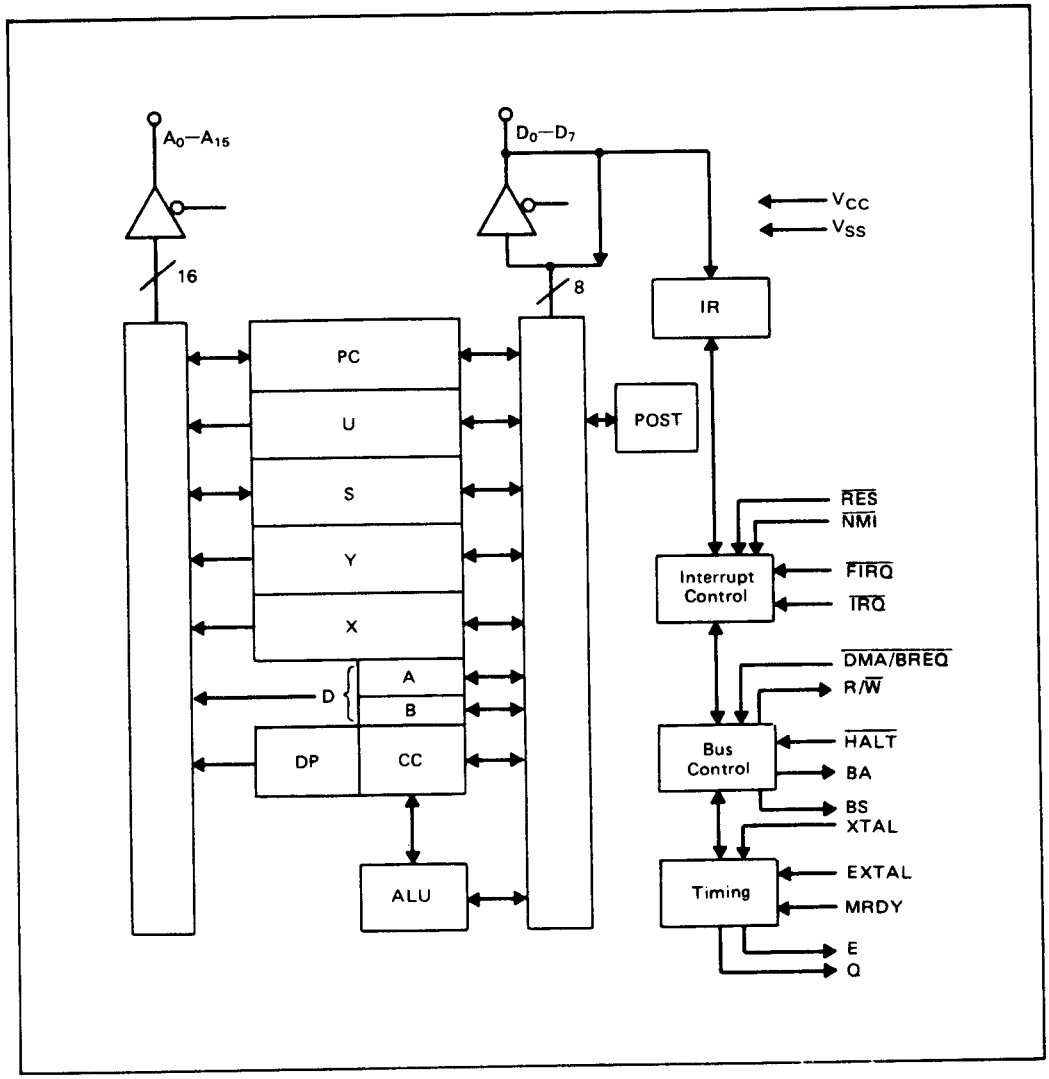
(Top view)

- PLCC package available



(Top View)

Block Diagram



2

Programming Model

As shown in figure 1, the HD6309 adds three registers to the set available in the HD6800. The added registers are a direct page register, the user stack pointer and a second index register.

Accumulators (A, B, D)

The A and B registers are general purpose accumulators which are used for arithmetic calculations and manipulation of data.

Certain instructions concatenate the A and B registers to form a single 16-bit accumulator. This is referred to as the D register. It is formed with the A register as the most significant byte.

Direct Page Register (DP)

The direct page register of the HD6309 serves to enhance the direct addressing mode. The contents of this register appears at the higher address outputs ($A_8 - A_{15}$) during direct addressing instruction execution. This allows the direct mode to be used at any place in memory, under program control. To ensure HD6800 compatibility, all bits of this register are cleared during processor reset.

Index Registers (X, Y)

The index registers are used in indexed mode addressing. The 16-bit address in this register takes part in the calculation of effective addresses. This address may be used to point to data directly or may be modified by an optional constant or register offset. In some indexed modes, the contents of the index register are incremented or decremented to point to the next item of tabular data. All four pointer registers (X, Y, U, S) may be used as index registers.

Stack Pointer (U, S)

The hardware stack pointer (S) is used automatically by the processor during subroutine calls and interrupts. The stack pointers of the HD6309 point to the top of the stack, in contrast to the HD6800 stack pointer, which pointed to the next free location on the stack. The user stack pointer (U) is controlled exclusively by the programmer thus allowing arguments to be passed to and from subroutines with ease. Both stack pointers have the same indexed mode addressing capabilities as the X and Y registers, but also support push and pull instructions. This allows the HD6309 to be used efficiently as a stack processor, greatly enhancing its ability to support higher level languages and modular programming.

Note: The stack pointers of the HD6309 point to the top of the stack, in contrast to the HD6800 stack pointer, which pointed to the next free location on stack.

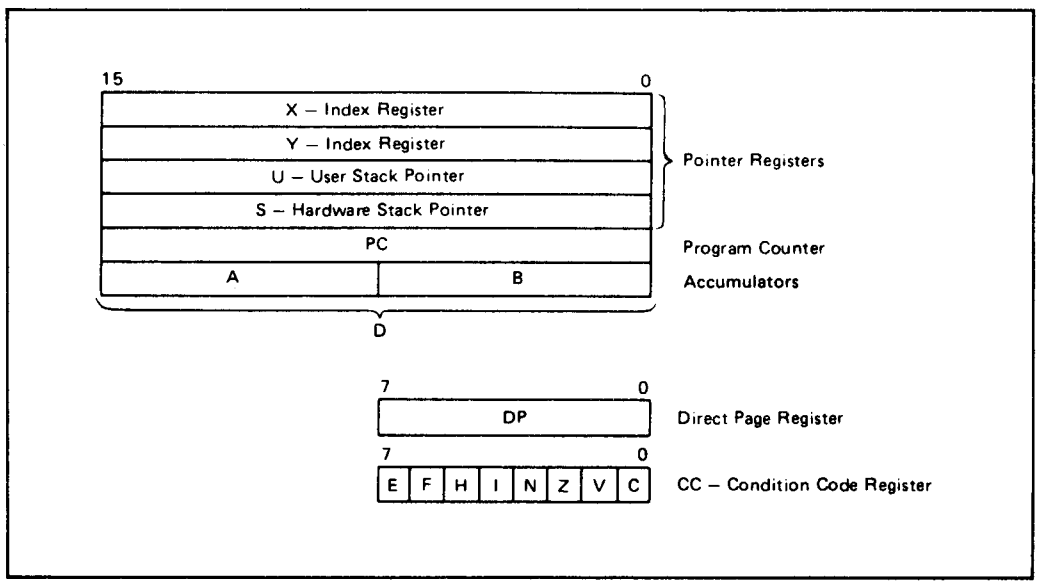


Figure 1. Programming Model of The Microprocessing Unit

Program Counter (PC)

The program counter is used by the processor to point to the address of the next instruction to be executed by the processor. Relative addressing is provided allowing the program counter to be used like an index register in some situations.

Condition Code Register (CC)

The condition code register defines the state of the processor at any given time. See figure 2.

Condition Code Register Description

Bit 0 (C)

Bit 0 is the carry flag. It is usually the carry from the binary ALU. C is also used to represent a 'borrow' from subtract-like instructions (CMP, NEG, SUB, SBC). Then, it is the complement of the carry from the binary ALU.

Bit 1 (V)

Bit 1 is the overflow flag. It is set to a one by an operation which causes a signed two's complement arithmetic overflow. This overflow is detected in an operation in which the carry from the MSB in the ALU does not match the carry from the MSB minus 1.

Bit 2 (Z)

Bit 2 is the zero flag. It is set to one if the result of the previous operation was identically zero.

Bit 3 (N)

Bit 3 is the negative flag. It contains exactly the value of the MSB of the result of the preceding operation. Thus, a negative two's-complement result will leave N set to one.

Bit 4 (I)

Bit 4 is the $\overline{IRQ}$ mask bit. The processor will not recognize interrupts from the $\overline{IRQ}$ line if this bit is set to one. $\overline{NMI}$, $\overline{FIRQ}$, $\overline{IRQ}$, $\overline{RES}$, and SWI all set I to one; SWI2 and SWI3 do not affect I.

Bit 5 (H)

Bit 5 is the half-carry bit. It is used to indicate a carry from bit 3 in the ALU as a result of an 8-bit addition only (ADC or ADD). This bit is used by the DAA instruction to perform a BCD decimal add adjust operation. The state of this flag is undefined in all subtract-like instructions.

Bit 6 (F)

Bit 6 is the $\overline{FIRQ}$ mask bit. The processor will not recognize interrupts from the $\overline{FIRQ}$ line if this bit is a one. $\overline{NMI}$, $\overline{FIRQ}$, SWI, and $\overline{RES}$ all set F to one. $\overline{IRQ}$, SWI2 and SWI3 do not affect F.

Bit 7 (E)

Bit 7 is the entire flag. Set to one, it indicates that the complete machine state (all the registers) was stacked, as opposed to the subset state (PC and CC). The E bit of the stacked CC is used on a return from interrupt (RTI) to determine the extent of the unstacking. Therefore, the current E left in the condition code register represents past action.

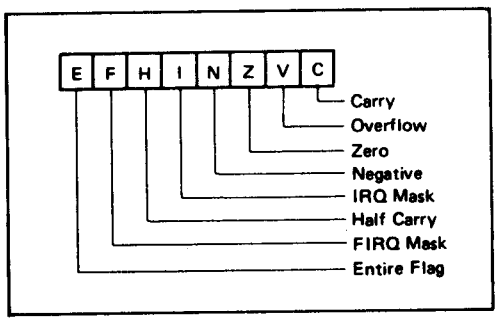


Figure 2. Condition Code Register Format

Signal Description

Power (Vss, V_{CC})

Two pins supply power to the part: Vss is ground or 0 volts, while V_{CC} is +5.0 V ±10%.

Address Bus ($A_0 - A_{15}$)

Sixteen pins output address information from the MPU onto the address bus. When the processor does not require the bus for a data transfer, it will output address $FFFF_{16}$, R/$\overline{W}$=high, and BS=low. This is a "dummy access" or $\overline{VMA}$ cycle (see figures 25 and 26). All address bus drivers are made high impedance when the bus available output (BA) is high. Each pin will drive one Schottky TTL load or four LS TTL loads, and typically 90 pF.

Data Bus ($D_0 - D_7$)

These eight pins provide communication with the system bi-directional data bus. Each pin will drive one Schottky TTL load or four LS TTL loads, and typically 130 pF.

Read/Write (R/$\overline{W}$)

This signal indicates the direction of data transfer on the data bus. A low indicates that the MPU is writing data onto the data bus. R/$\overline{W}$ is made high impedance when BA is high. Refer to figures 25 and 26.

Reset ($\overline{RES}$)

A low level on this Schmitt-trigger input for greater than one bus cycle will reset the MPU, as shown in figure 3. The reset vectors are fetched from locations $FFFE_{16}$ and $FFFF_{16}$ (table 2) when interrupt acknowledge is true, ($\overline{BA} \cdot BS = 1$). During initial power-on, the reset line should be held low until the clock oscillator is fully operational. See figure 4.

Because the HD6309 reset pin has a Schmitt-trigger input with a threshold voltage higher than that of standard peripherals, a simple R/C network may be used to reset the entire system. This higher

Table 1. Pin Description

Symbol	Pin No.	I/O	Function
Vss	1		Ground
$\overline{NMI}$	2	I	Non maskable interrupt
$\overline{IRQ}$	3	I	Interrupt request
$\overline{FIRQ}$	4	I	Fast interrupt request
BS, BA	5, 6	O	Bus status, Bus available
Vcc	7		+5 V power supply
$A_0 - A_{15}$	8–23	O	Address bus, bits 0-15
$D_7 - D_0$	24–31	I/O	Data bus, bits 0-7
R/$\overline{W}$	32	O	Read / Write output
$\overline{DMA/BREQ}$	33	I	DMA Bus request
E, Q	34, 35	O	Clock signal
MRDY	36	I	Memory ready
$\overline{RES}$	37	I	Reset input
EXTAL, XTAL	38, 39	I	Oscillator connection
$\overline{HALT}$	40	I	Halt input

threshold voltage ensures that all peripherals are out of the reset state before the processor.

Halt ($\overline{HALT}$)

A low level on this input pin will cause the MPU to stop running at the end of the present instruction and remain halted indefinitely without loss of data. When halted, the BA output is driven high indicating the buses are high impedance. BS is also high which indicates the processor is in the halt or bus grant state. While halted, the MPU will not respond to external realtime requests ($\overline{FIRQ}$, $\overline{IRQ}$) although $\overline{DMA/BREQ}$ will always be accepted, and $\overline{NMI}$ or $\overline{RES}$ will be latched for later response. During the halt state, Q and E continue to run normally. If the MPU is not running ($\overline{RES}$), a halted state (BA · BS = 1) can be achieved by pulling $\overline{HALT}$ low while $\overline{RES}$ is still low. See figure 5.

Bus Available, Bus Status (BA, BS)

The BA output is an indication of an internal control signal which makes the MOS buses of the MPU high impedance. This signal does not imply that the bus will be available for more than one cycle. When BA goes low, an additional dead cycle will elapse before the MPU acquires the bus.

The BS output signal, when decoded with BA, represents the MPU state.

Interrupt Acknowledge is indicated during both cycles of a hardware vector fetch ($\overline{RES}$, $\overline{NMI}$, $\overline{FIRQ}$, $\overline{IRQ}$, SWI, SWI2, SWI3). This signal, plus decoding of the lower four address lines, can provide the user with an indication of which interrupt level is being serviced and allow vectoring by device. See Table 2.

Sync Acknowledge is indicated while the MPU is waiting for external synchronization on an interrupt line.

Halt/Bus Grant is true when the HD6309 is in a halt or bus grant condition.

Non Maskable Interrupt ($\overline{NMI}$)

A negative edge on $\overline{NMI}$ requests that a nonmaskable interrupt sequence be generated. A nonmaskable interrupt cannot be inhibited by the program, and also has a higher priority than $\overline{FIRQ}$, $\overline{IRQ}$ or software interrupts. During recognition of an $\overline{NMI}$, the entire machine state is saved on the hardware stack. After reset, an $\overline{NMI}$ will not be recognized until the first program load of the hardware stack pointer (S). The pulse width of $\overline{NMI}$ low must be at least one E cycle. If the $\overline{NMI}$ input does not meet the minimum set up with respect to Q, the interrupt will not be recognized until the next cycle See figure 6.

2

Table 2. Memory Map for Interrupt Vectors

Memory Map for Vector Locations		Interrupt Vector Description
MS	**LS**	
FFFE	FFFF	$\overline{RES}$
FFFC	FFFD	$\overline{NMI}$
FFFA	FFFB	SWI
FFF8	FFF9	$\overline{IRQ}$
FFF6	FFF7	$\overline{FIRQ}$
FFF4	FFF5	SWI2
FFF2	FFF3	SWI3
FFF0	FFF1	Reserved

Table 3. MPU State Definition

BA	BS	MPU State
0	0	Normal (Running)
0	1	Interrupt or RESET Acknowledge
1	0	SYNC Acknowledge
1	1	HALT or Bus Grant

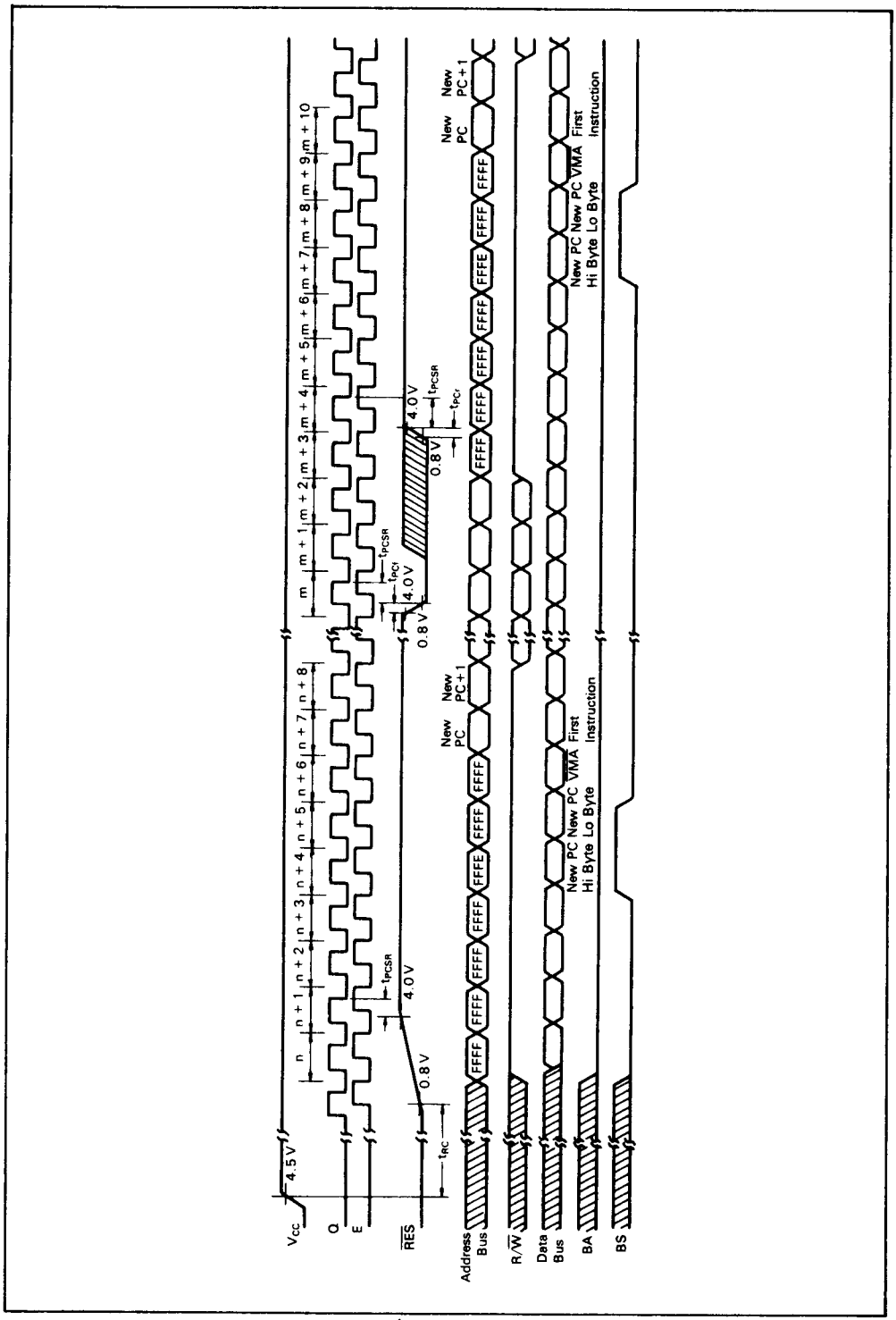
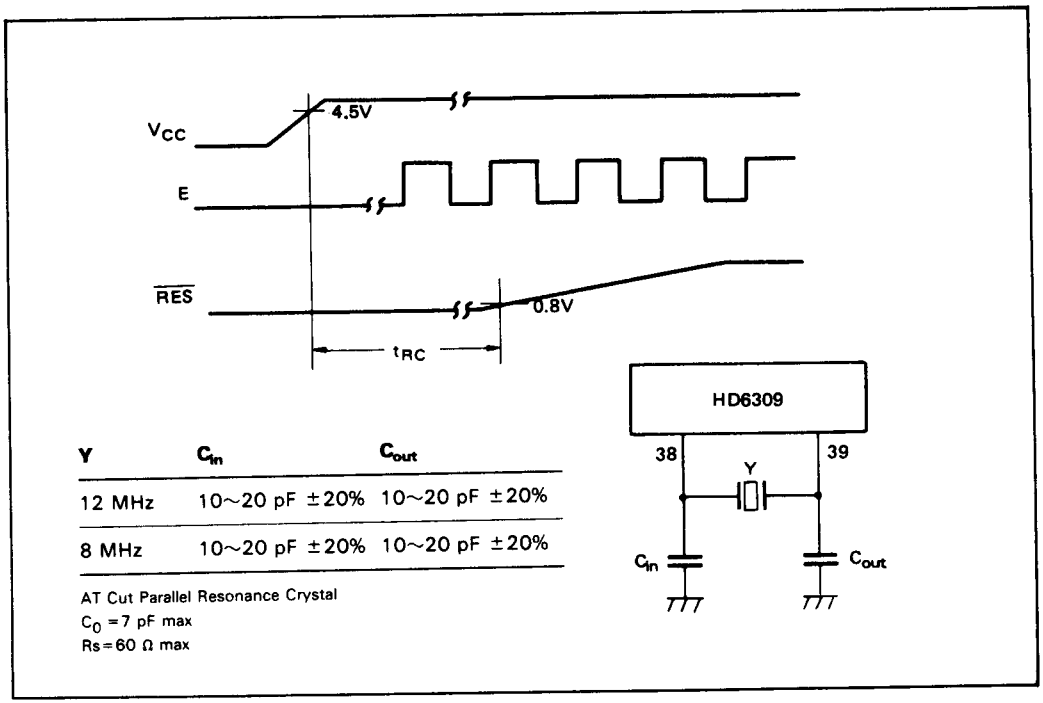


Figure 3. $\overline{RES}$ Timing



Y	C_{in}	C_{out}
12 MHz	10~20 pF ±20%	10~20 pF ±20%
8 MHz	10~20 pF ±20%	10~20 pF ±20%

AT Cut Parallel Resonance Crystal
C_0 = 7 pF max
Rs = 60 Ω max

Figure 4. Crystal Connections and Oscillator Start Up

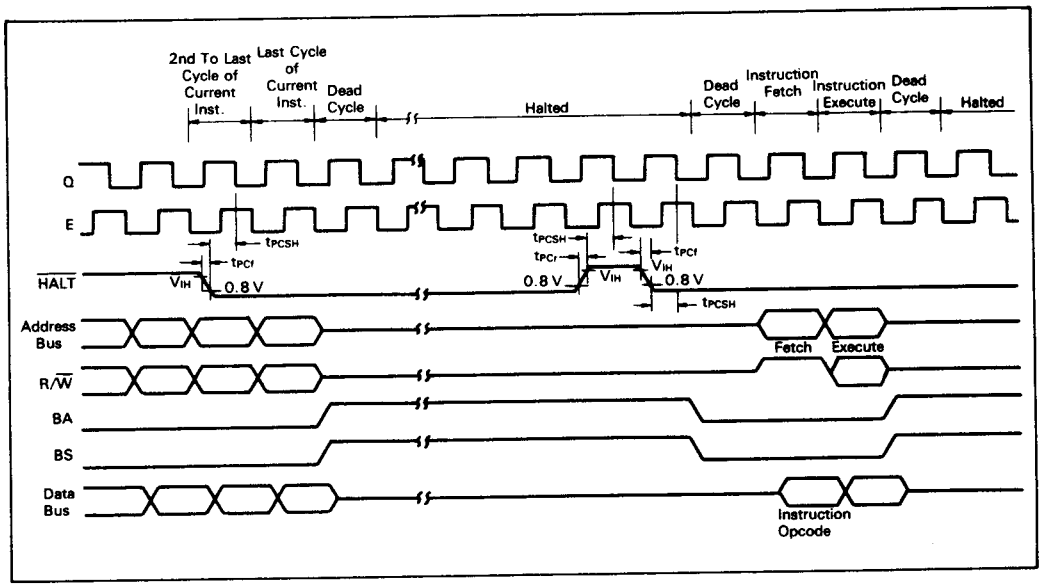


Figure 5. HALT and Single Instruction Execution for System Debug

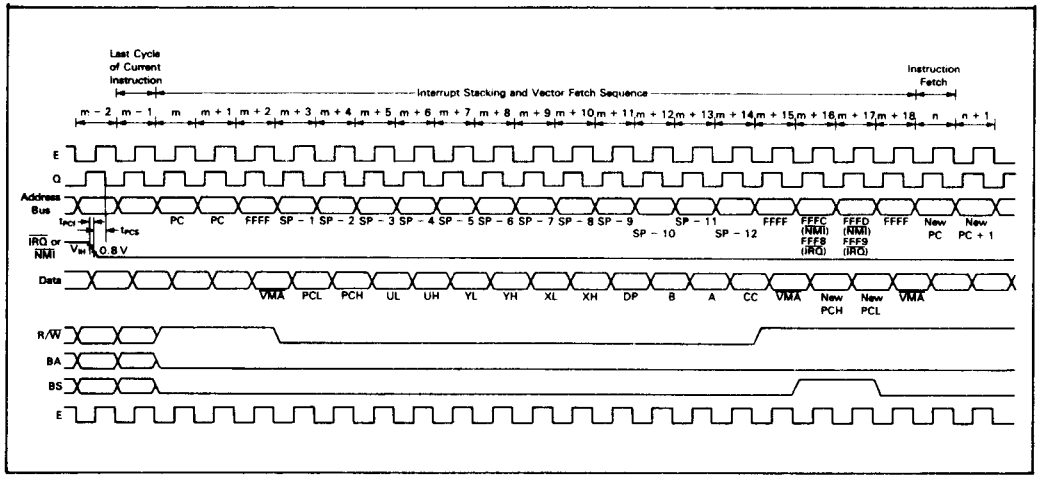


Figure 6. $\overline{IRQ}$ and $\overline{NMI}$ Interrupt Timing

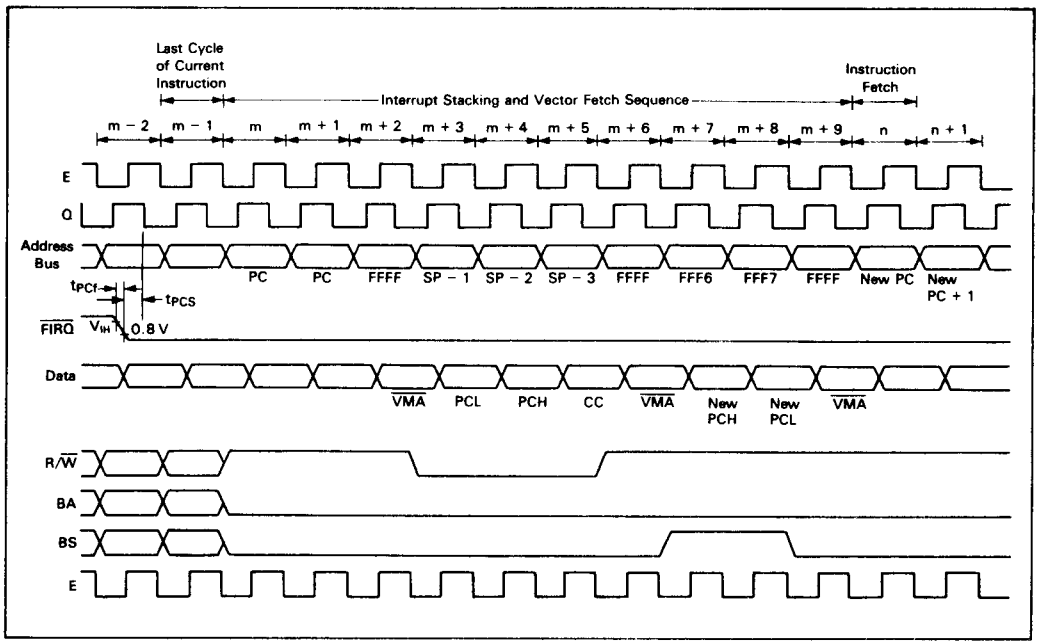


Figure 7. $\overline{FIRQ}$ Interrupt Timing

Fast Interrupt Request ($\overline{FIRQ}$)

A low level on $\overline{FIRQ}$ input will initiate a fast interrupt sequence provided its mask bit (F) in the CC is clear. This sequence has priority over the standard interrupt request ($\overline{IRQ}$). It is fast in the sense that it stacks only the contents of the condition code register and the program counter. The interrupt service routine should clear the source of the interrupt before doing an RTI. See figure 7.

Interrupt Request ($\overline{IRQ}$)

A low level input on $\overline{IRQ}$ will initiate an interrupt request sequence provided the mask bit (I) in the CC is clear. Since $\overline{IRQ}$ stacks the entire machine state it provides a slower response to interrupts than $\overline{FIRQ}$. $\overline{IRQ}$ also has a lower priority than $\overline{FIRQ}$. Again, the interrupt service routine should clear the source of the interrupt before doing an RTI. See figure 6.

Note: $\overline{NMI}$, $\overline{FIRQ}$, and $\overline{IRQ}$ requests are sampled on the falling edge of Q. One cycle is required for synchronization before these interrupts are recognized. The pending interrupt(s) will not be serviced until completion of the current instruction unless a SYNC or CWAI condition is present. If $\overline{IRQ}$ and $\overline{FIRQ}$ do not remain low until completion of the current instruction they may not be recognized. However, $\overline{NMI}$ is latched and need only remain low for one cycle.

XTAL, EXTAL

These two pins are connected with parallel resonant fundamental crystal, AT cut. Alternately, the pin EXTAL may be used as a TTL level input for external timing with XTAL floating. The crystal or external frequency is four times the bus frequency. See figure 4. Proper RF layout techniques should be observed in the layout of printed circuit boards.

Note for Board Design of the Oscillation Circuit: In designing the board, the following notes should be taken when the crystal oscillator is used. See figure 8.

1. Crystal oscillator and load capacity Cin, Cout must be placed near the LSI as much as possible. (Normal oscillation may be disturbed when external noise is induced to pin 38 and 39.)

2. Pin 38 and 39 signal line should be wired apart from other signal line as much as possible. Don't wire them in parallel with other lines. (Normal oscillation may be disturbed when E or Q signal feeds back to pin 38 and 39.)

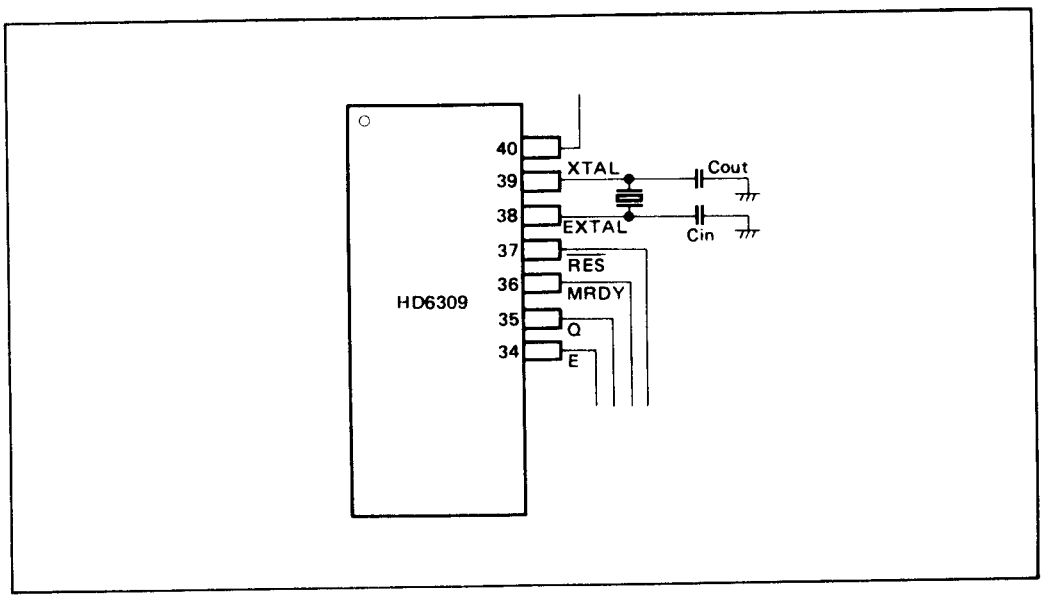


Figure 8. Board Design of the Oscillation Circuit

Designs to be Avoided: A signal line or a power source line must not cross or go near the oscillation circuit line as shown in figure 9 to prevent induction from these lines. The resistance between XTAL, EXTAL and other pins should be over 10 MΩ.

E, Q

E is similar to the HD6800 bus timing signal ϕ_2: Q is a quadrature clock signal which leads E. Q has no parallel on the HD6800. Data is latched on the falling edge of E. Timing for E and Q is shown in figure 10.

Memory Ready (MRDY)

This input control signal allows stretching of E and Q to extend data-access time. E and Q operate normally while MRDY is high. When MRDY is low, E and Q may be stretched in integral multiples of half (1/2) bus cycles, thus allowing interface to slow memories, as shown in figure 11. The maximum stretch is 5 microseconds.

During nonvalid memory access ($\overline{\text{VMA}}$ cycles) MRDY has no effect on stretching E and Q: this inhibits slowing the processor during "don't care" bus accesses. MRDY may also be used to stretch clocks (for slow memory) when bus control has been transferred to an external device (through the use of $\overline{\text{HALT}}$ and $\overline{\text{DMA/BREQ}}$).

MRDY also stretches E and Q during dead cycles.

DMA Bus Request ($\overline{\text{DMA/BREQ}}$)

The $\overline{\text{DMA/BREQ}}$ input provides a method of suspending execution and acquiring the MPU bus for another use, as shown in figure 12. Typical uses include DMA and dynamic memory refresh.

Transition of $\overline{\text{DMA/BREQ}}$ should occur during Q. A low level on this pin will stop instruction execution at the end of the current cycle. The MPU will acknowledge $\overline{\text{DMA/BREQ}}$ by setting BA and BS to high level. The HD6309 does not perform the auto-refresh executed in the HD6809. See figure 13.

Typically, the DMA controller will request to use the bus by asserting $\overline{\text{DMA/BREQ}}$ pin low on the leading edge of E. When the MPU replies by setting BA and BS to one, that cycle will be a dead cycle used to transfer bus mastership to the DMA controller.

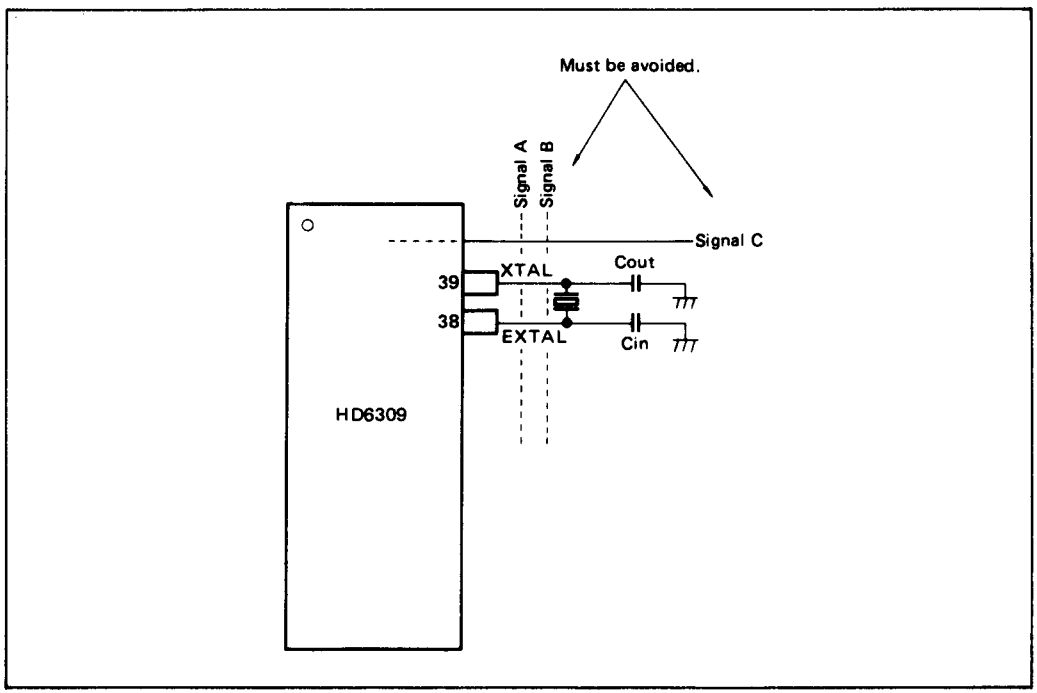


Figure 9. Example of Normal Oscillation may be Disturbed

False memory accesses may be prevented during dead cycles by developing a system $\overline{DMAVMA}$ signal which is low in any cycle when BA has changed.

When BA goes low (a result of $\overline{DMA/BREQ}$ = high), another dead cycle will elapse before the MPU accesses memory, to allow transfer of bus mastership without contention.

The $\overline{DMA/BREQ}$ input should be tied high during reset state.

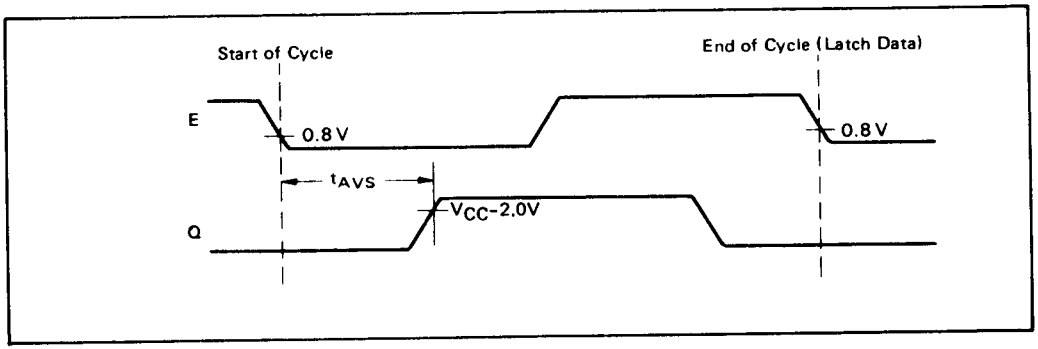


Figure 10. E/Q Relationship

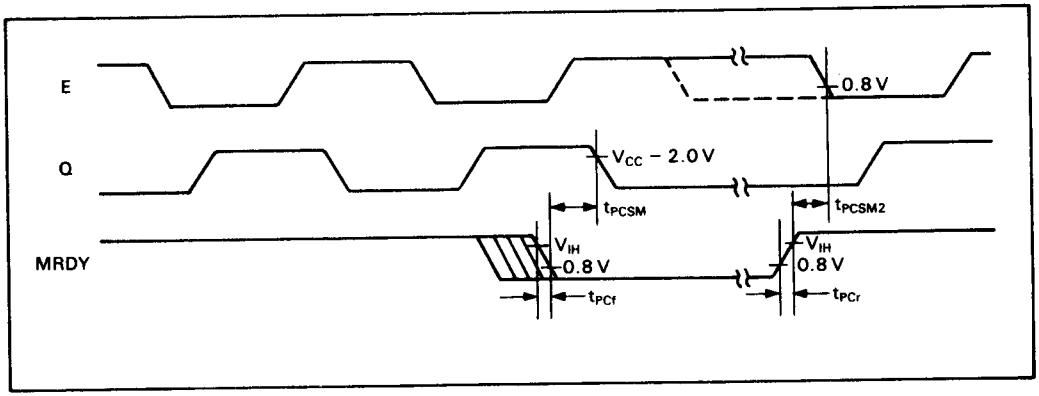


Figure 11. MRDY Clock Stretching

MPU Operation

During normal operation, the MPU fetches an instruction from memory and then executes the requested function. This sequence begins at $\overline{RES}$ and is repeated indefinitely unless altered by a special instruction or hardware occurrence. Software instructions that alter normal MPU operation are: SWI, SWI2, SWI3, CWAI, RTI and SYNC. An interrupt, $\overline{HALT}$ or $\overline{DMA/BREQ}$ can also alter the normal execution of instructions. Figure 14 illustrates the flow chart for the HD6309.

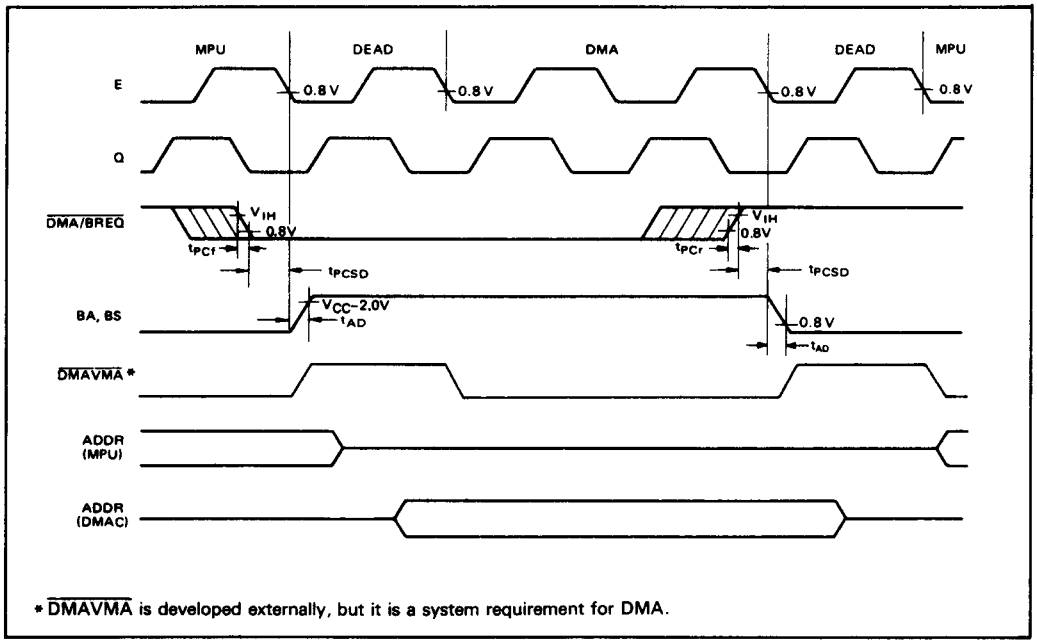


Figure 12. Typical DMA Timing

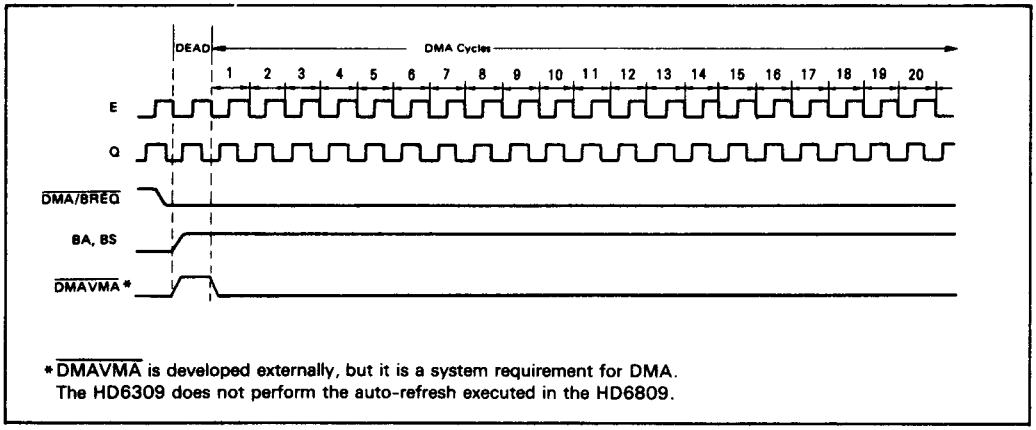
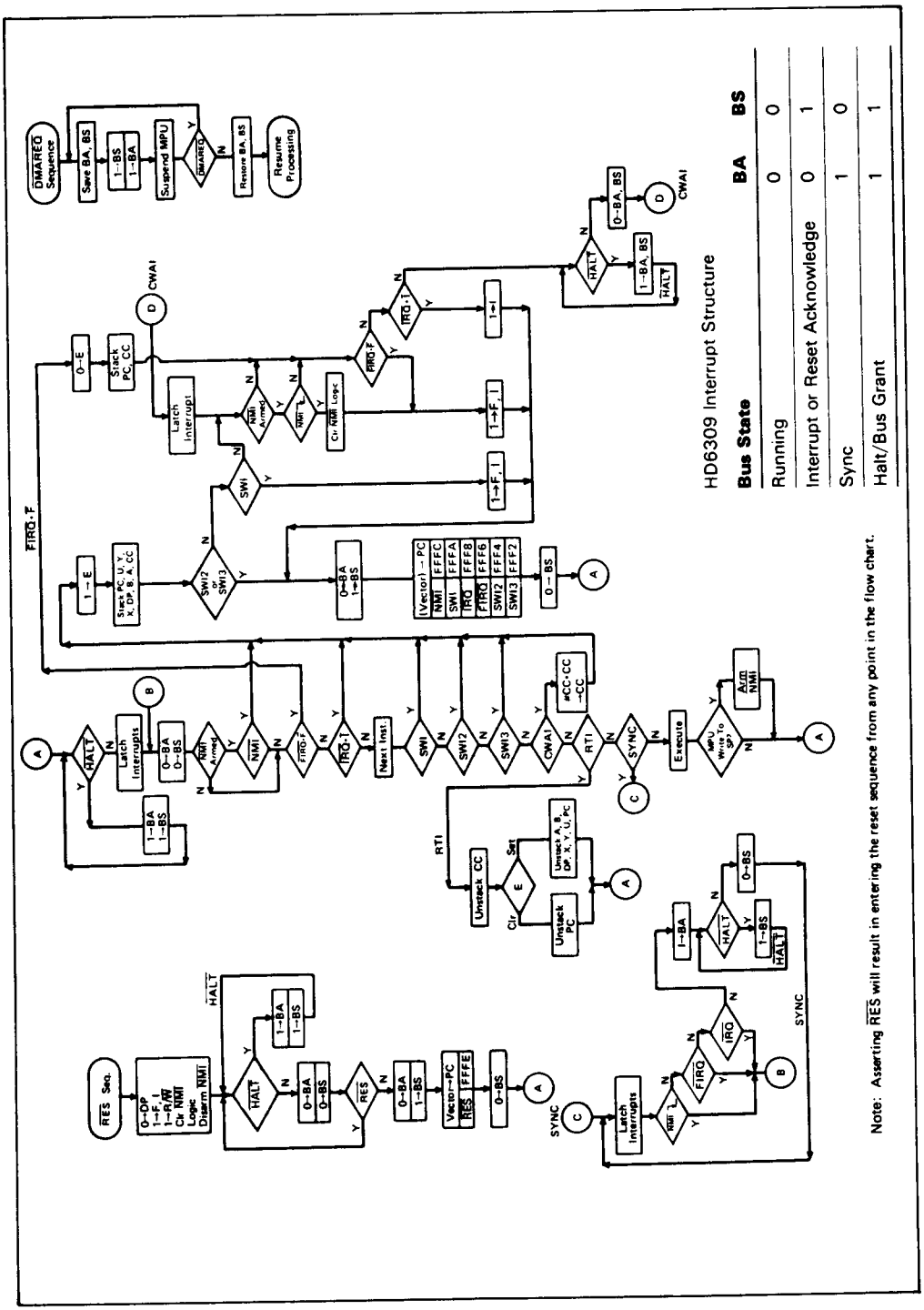


Figure 13. DMA Timing



Note: Asserting $\overline{RES}$ will result in entering the reset sequence from any point in the flow chart.

Figure 14. Flowchart for HD6309 Instruction

2

Addressing Modes

The basic instructions of any computer are greatly enhanced by the presence of powerful addressing modes. The HD6309 has the most complete set of addressing modes available on any microcomputer today. For example, the HD6309 has 59 basic instructions, however, it recognizes 1464 different variations of instructions and addressing modes. The addressing modes support modern programming techniques. The following addressing modes are available on the HD6309:

- Implied (includes accumulator)
- Immediate
- Extended
- Extended indirect
- Direct
- Register
- Indexed
  - Zero-offset
  - Constant offset
  - Accumulator offset
  - Auto increment/decrement
- Indexed indirect
- Relative
- Program counter relative

Implied (Includes Accumulator)

In this addressing mode, the opcode of the instruction contains all the address information necessary. Examples of implied addressing are: ABX, DAA, SWI, ASRA, and CLRB.

Immediate Addressing

In immediate addressing, the effective address of the data is the location immediately following the opcode (i.e., the data to be used in the instruction immediately follows the opcode of the instruction). The HD6309 uses both 8-and 16-bit immediate values depending on the size of the argument specified by the opcode. Examples of instructions with immediate addressing are:

```
LDA #$20
LDX #$F000
LDY #CAT
```

Note: # signifies immediate addressing, $ signifies hexadecimal value.

Extended Addressing

In extended addressing, the contents of the two bytes immediately following the opcode fully specify the 16-bit effective address used by the instruction. Note that the address generated by an extended instruction defines an absolute address and is not position independent. Examples of extended addressing include:

```
LDA   CAT
STX   MOUSE
LDD   $2000
```

Extended Indirect

As a special case of indexed addressing (discussed below), one level of indirection may be added to extended addressing. In extended indirect, the two bytes following the postbyte of an indexed instruction contain the address of the data.

```
LDA   [CAT]
LDX   [$FFFE]
STU   [DOG]
```

Direct Addressing

Direct addressing is similar to extended addressing except that only one byte of address follows the opcode. This byte specifies the lower 8 bits of the address to be used. The upper 8 bits of the address are supplied by the direct page register. Since only one byte of address is required in direct addressing, this mode requires less memory and executes faster than extended addressing. Of course, only 256 locations (one page) can be accessed without redefining the contents of the DP register. Since the DP register is set to $00 on reset, direct addressing on the HD6309 is compatible with direct addressing on the HD6800. Indirection is not allowed in direct addressing. Some examples of direct addressing are:

```
LDA     $30
SETDP   $10 (Assembler directive)
LDB     $1030
LDD     <CAT
```

Note: < is an assembler directive which forces direct addressing.

Register Addressing

Some opcodes are followed by a byte that defines a register or set of registers to be used by the instruction. This is called a postbyte. Some examples of register addressing are:

TFR	X,Y	Transfers X into Y
EXG	A, B	Exchanges A with B
PSHS	A, B, X, Y	Push Y, X, B, and A onto S
PULU	X, Y, D	Pull D, X, and Y from U

Indexed Addressing

In all indexed addressing, one of the pointer registers (X, Y, U, S, and sometimes PC) is used in a calculation of the effective address of the operand to be used by the instruction. Five basic types of indexing are available and are discussed below. The postbyte of an indexed instruction specifies the basic type and variation of the addressing mode as well as the pointer register to be used. Figure 15 lists the legal formats for the postbyte. Table 4 gives the assembler form and the number of cycles and bytes added to the basic values for indexed addressing for each variation.

Zero-Offset Indexed: In this mode, the selected pointer register contains the effective address of the data to be used by the instruction. This is the fastest indexing mode.

Examples are:

```
LDD     0, X
LDA     S
```

Post-byte Register Bit								Indexed Addressing Mode
7	6	5	4	3	2	1	0	
0	R	R	d	d	d	d	d	EA = ,R +5 Bit Offset
1	R	R	0	0	0	0	0	,R+
1	R	R	0/1	0	0	0	1	,R++
1	R	R	0	0	0	1	0	,−R
1	R	R	0/1	0	0	1	1	,−−R
1	R	R	0/1	0	1	0	0	EA = ,R + 0 Offset
1	R	R	0/1	0	1	0	1	EA = ,R + B Offset
1	R	R	0/1	0	1	1	0	EA = ,R + A Offset
1	R	R	0/1	1	0	0	0	EA = ,R + 8 Bit Offset
1	R	R	0/1	1	0	0	1	EA = ,R + 16 Bit Offset
1	R	R	0/1	1	0	1	1	EA = ,R + D Offset
1	×	×	0/1	1	1	0	0	EA = ,PC + 8 Bit Offset
1	×	×	0/1	1	1	0	1	EA = ,PC + 16 Bit Offset
1	0	0	1	1	1	1	1	EA = [, Address]

Addressing Mode Field

Indirect Field
(Sign bit when b7 = 0)
0····· Non Indirect
1····· Indirect

Register Field : RR
00 = X
01 = Y
10 = U
11 = S

d = Offset Bit
X = Don't Care

Figure 15. Indexed Addressing Postbyte Register Bit Assignments

Constant Offset Indexed: In this mode, a two's -complement offset and the contents of one of the pointer registers are added to form the effective address of the operand. The pointer register's initial content is unchanged by the addition.

Three sizes of offsets are available:

5-bit (−16 to +15)
8-bit (−128 to+127)
16-bit (−32768 to+32767)

The two's complement 5-bit offset is included in the postbyte and, therefore, is most efficient in use of bytes and cycles. The two's complement 8-bit offset is contained in a single byte following the postbyte. The two's complement 16-bit offset is in the two bytes following the postbyte. In most cases the programmer need not be concerned with the size of this offset since the assembler will select the optimal size automatically.

Examples of constant-offset indexing are:

```
LDA     23, X
LDX     −2, S
LDY     300, X
LDU     CAT, Y
```

Accumulator Offset Indexed: This mode is similar to constant offset indexed except that the two's-complement value in one of the accumulators (A, B or D) and the contents of one of the pointer registers are added to form

Table 4. Indexed Addressing Mode

		Non Indirect				Indirect			
Type	**Forms**	**Assembler Form**	**Postbyte OP Code**	**+ + ~ #**		**Assembler Form**	**Postbyte OP Code**	**+ + ~ #**	
Constant Offset From R (2's Complement Offsets)	No Offset	,R	1RR00100	0	0	[,R]	1RR10100	3	0
	5 Bit Offset	n,R	0RRnnnnn	1	0	defaults to 8-bit			
	8 Bit Offset	n,R	1RR01000	1	1	[n, R]	1RR11000	4	1
	16 Bit Offset	n,R	1RR01001	4	2	[n, R]	1RR11001	7	2
Accumulator Offset From R (2's Complement Offsets)	A Register Offset	A,R	1RR00110	1	0	[A, R]	1RR10110	4	0
	B Register Offset	B,R	1RR00101	1	0	[B, R]	1RR10101	4	0
	D Register Offset	D,R	1RR01011	4	0	[D, R]	1RR11011	7	0
Auto Increment/Decrement R	Increment By 1	,R+	1RR00000	2	0	not allowed			
	Increment By 2	,R++	1RR00001	3	0	[,R++]	1RR10001	6	0
	Decrement By 1	,−R	1RR00010	2	0	not allowed			
	Decrement By 2	,−−R	1RR00011	3	0	[,−−R]	1RR10011	6	0
Constant Offset From PC (2's Complement Offsets)	8 Bit Offset	n, PCR	1xx01100	1	1	[n, PCR]	1xx11100	4	1
	16 Bit Offset	n, PCR	1xx01101	5	2	[n, PCR]	1xx11101	8	2
Extended Indirect	16 Bit Address					[n]	10011111	5	2

R = X, Y, U or S
x = Don't Care

RR:
00=X
01=Y
10=U
11=S

+ and # indicate the number of additional cycles and bytes for the particular variation.

the effective address of the operand. The contents of both the accumulator and the pointer register are unchanged by the addition. The postbyte specifies which accumulator to use as an offset and no additional bytes are required. The advantage of an accumulator offset is that the value of the offset can be calculated by a program at run-time.

Some examples are:

```
LDA     B, Y
LDX     D, Y
LEAX    B, X
```

Auto Increment/Decrement Indexed: In the auto increment addressing mode, the pointer register contains the address of the operand. Then, after the pointer register is used it is incremented by one or two. This addressing mode is useful in stepping through tables, moving data, or for the creation of software stacks. In auto decrement, the pointer register is decremented prior to use as the address of the data. The use of auto decrement is similar to that of auto increment; but the tables, etc, are scanned from high to low addresses. The size of the increment/decrement can be either one or two to allow for tables of either 8-or 16-bit data to be accessed, selectable by the programmer. The pre-decrement, post-increment nature of these modes allow them to be used to create additional software stacks that behave identically to the U and S stacks.

Some examples of the auto increment/decrement addressing modes are:

```
LDA     ,X+
STD     ,Y++
LDB     ,-Y
LDX     ,--S
```

Care should be taken in performing operations on 16-bit pointer registers (X, Y, U, S) where the same register is used to calculate the effective address.

Consider the following instruction:

STX 0, X++ (X initialized to 0)

The desired result is to store a 0 in locations $0000 and $0001 then increment X to point to $0002. In reality, the following occurs:

```
0→temp      calculate the EA; temp is a
            holding register
X+2→X       perform autoincrement
X→(temp)    do store operation
```

Indexed Indirect

All of the indexing modes with the exception of auto increment/decrement by one, or a ±4-bit offset may have an additional level of indirection specified. In indirect addressing, the effective address is contained at the location specified by the contents of the index register plus any offset. In the example below, the A accumulator is loaded indirectly using an effective address calculated from the index register and an offset.

Before Execution:

A = ×× (don't care)
X = $F000

```
$0100     LDA [$10, X]     EA is now $F010

$F010     $F1              $F150 is now the
$F011     $50              new EA

$F150     $AA
```

After Execution:

A = $AA (Actual Data Loaded)
X = $F000

All modes of indexed indirect are included except those which are meaningless (e.g., auto increment/decrement by 1 indirect). Some examples of indexed indirect are:

```
LDA     [,X ]
LDD     [10,S]
LDA     [B,Y]
LDD     [,X++]
```

Relative Addressing

The byte(s) following the branch opcode is (are) treated as a signed offset which may be added to the program counter. If the branch condition is true then the calculated address (PC + signed offset) is loaded into the program counter. Program execution continues at the new location as indicated by the PC. Short (1 byte offset) and long (2 bytes offset) relative addressing modes are available. All of memory can be reached in long relative addressing as an effective address is interpreted modulo 2^{16}. Some examples of relative addressing are:

```
        BEQ     CAT     (short)
        BGT     DOG     (short)
CAT     LBEQ    RAT     (long)
DOG     LBGT    RABBIT  (long)
        .
        .
        .
RAT     NOP
RABBIT  NOP
```

Program Counter Relative

The PC can be used as the pointer register with 8 -or 16-bit signed offsets. As in relative addressing, the offset is added to the current PC to create the effective address. The effective address is then used as the address of the operand or data. Program counter relative addressing is used for writing position independent programs. Tables related to a particular routine will maintain the same relationship after the routine is moved, if referenced relative to the program counter. Examples are:

```
LDA     CAT, PCR
LEAX    TABLE, PCR
```

Since program counter relative is a type of indexing, an additional level of indirection is available.

```
LDA     [CAT, PCR]
LDU     [DOG, PCR]
```

HD6309 Instruction Set

The instruction set of the HD6309 is similar to that of the HD6800 and is upward compatible at the source code level. The number of opcodes has been reduced from 72 to 59, but because of the expanded architecture and additional addressing modes, the number of available opcodes (with different addressing modes) has risen from 197 to 1464.

Some of the instructions and addressing modes are described in detail below:

PSHU/PSHS

The push instructions can push onto either the hardware stack (S) or user stack (U) any single register, or set of registers with a single instruction.

PULU/PULS

The pull instructions have the same capability of the push instruction, in reverse order. The byte immediately following the push or pull opcode determines which register or registers are to be pushed or pulled. The actual PUSH/PULL sequence is fixed: each bit defines a unique register to push or pull, as shown in figure 16.

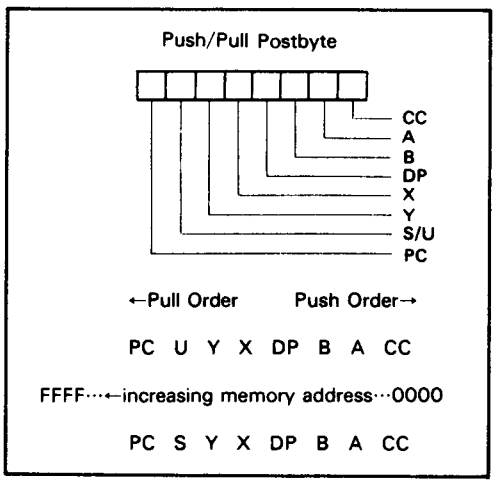


Figure 16. Push and Pull Order

TFR/EXG

Within the HD6309, any register may be transferred to or exchanged with another of like-size: i.e., 8-bit to 8-bit or 16-bit to 16-bit. Bits 4-7 of the postbyte define the source register, while bits 0-3 represent the destination register (figure 17). They are denoted as follows:

0000 – D	0101 – PC
0001 – X	1000 – A
0010 – Y	1001 – B
0011 – U	1010 – CC
0100 – S	1011 – DP

Note: All other combinations are undefined and invalid.

LEAX/LEAY/LEAU/LEAS

The LEA (load effective address) works by calculating the effective address used in an indexed instruction and stores that address value, rather than the data at that address, in a pointer register. This makes all the features of the internal addressing hardware available to the programmer. Some of the implications of this instruction are illustrated in table 5.

The LEA instruction also allows the user to access data in a position independent manner. For example:

```
LEAX  MSG1, PCR
LBSR  PDATA(Print message routine)


MSG1 FCC  'MESSAGE'
```

This sample program prints: 'MESSAGE'. By writing MSG1, PCR, the assembler computes the distance between the present address and MSG1. This result is placed as a constant into the LEAX instruction which will be indexed from the PC value at the time of execution. No matter where the code is located, when it is executed, the computed offset from the PC will put the absolute address of MSG1 into the X pointer register. This code is totally position independent.

The LEA instructions are very powerful and use an internal holding register (temp). Care must be exercised when using the LEA instructions with the autoincrement and autodecrement addressing modes due to the sequence of internal operations. The LEA internal sequence is outlined as follows:

LEAa ,b+	(any of the 16-bit pointer registers X, Y, U, or S may be substituted for a and b)
1. b→temp	(calculate the EA)
2. b + 1→b	(modify b, postincrement)
3. temp →a	(load a)
LEAa ,–b	
1. b – 1 →temp	(calculate EA with predecrement)
2. b – 1 →b	(modify b, predecrement)
3. temp →a	(load a)

Autoincrement-by-two and autodecrement-by-two instructions work similarly. Note that LEAX, X+ does not change X, however LEAX,–X does decrement X. LEAX 1, X should be used to increment X by one.

MUL

Multiplies the unsigned binary numbers in the A

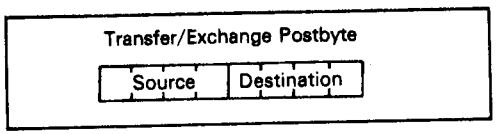


Figure 17. TFR/EXG Format

Table 5. LEA Examples

Instruction	Operation	Comment
LEAX 10, X	X+10→X	Adds 5-bit constant 10 to X
LEAX 500, X	X+500→X	Adds 16-bit constant 500 to X
LEAY A, Y	Y+A→Y	Adds 8-bit A accumulator to Y
LEAY D, Y	Y+D→Y	Adds 16-bit D accumulator to Y
LEAU-10, U	U-10→U	Subtracts 10 from U
LEAS-10, S	S-10→S	Used to reserve area on stack
LEAS 10, S	S+10→S	Used to 'clean up' stack
LEAX 5, S	S+5→X	Transfers as well as adds

and B accumulator and places the unsigned result into the 16-bit D accumulator. This unsigned multiply also allows multiple-precision multiplications.

Long And Short Relative Branches

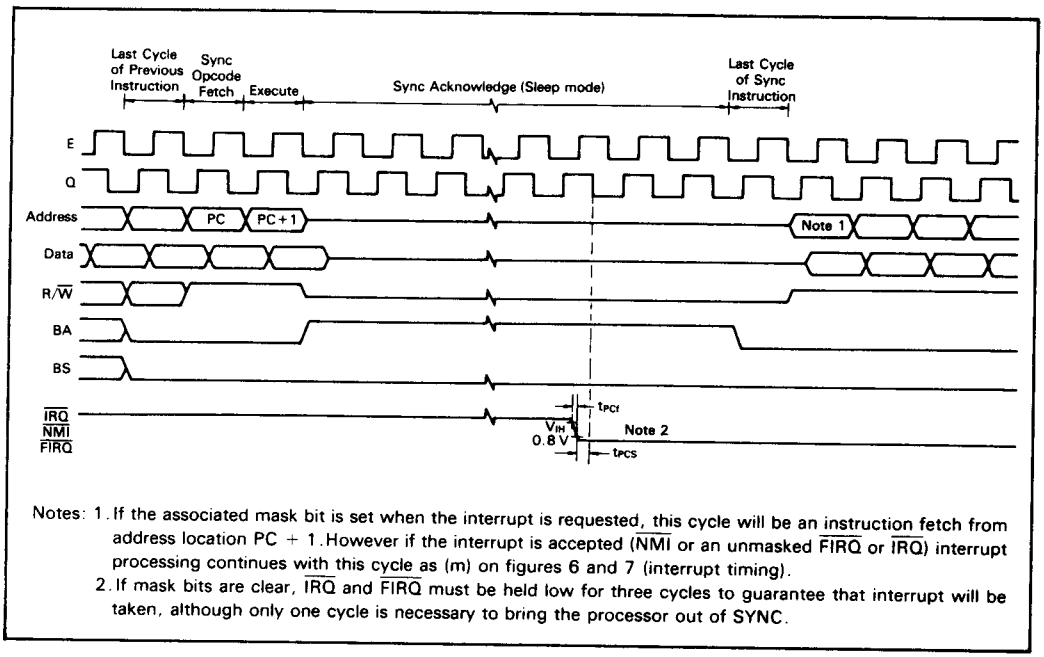
The HD6309 has the capability of program counter relative branching throughout the entire memory map. In this mode, if the branch is to be taken, the 8-or 16-bit signed offset is added to the value of the program counter to be used as the effective address. This allows the program to branch anywhere in the 64k memory map. Position independent code can be easily generated through the use of relative branching. Both short (8-bit) and long (16-bit) branches are available.

SYNC

After encountering a sync instruction, the MPU enters a sync state, stops processing instructions, and waits for an interrupt. If the pending interrupt is non-maskable ($\overline{NMI}$) or maskable ($\overline{FIRQ}$, $\overline{IRQ}$) with its mask bit (F or I) clear, the processor will clear the sync state and perform the normal interrupt stacking and service routine. Since $\overline{FIRQ}$ and $\overline{IRQ}$ are not edge-triggered, a low level with a minimum duration of three bus cycles is required to assure that the interrupt will be taken. If the pending interrupt is maskable ($\overline{FIRQ}$, $\overline{IRQ}$) with its mask bit (F or I) set, the processor will clear the sync state and continue processing by executing the next inline instruction. Figure 18 depicts sync timing.

Software Interrupt

A software interrupt instruction will cause an interrupt, and its associated vector fetch. These software interrupts are useful in operating system calls, software debugging, trace operations, memory mapping, and software development systems. Three levels of SWI are available on this HD6309, and are prioritized in the following order: SWI, SWI2, SWI3.



Notes: 1. If the associated mask bit is set when the interrupt is requested, this cycle will be an instruction fetch from address location PC + 1. However if the interrupt is accepted ($\overline{NMI}$ or an unmasked $\overline{FIRQ}$ or $\overline{IRQ}$) interrupt processing continues with this cycle as (m) on figures 6 and 7 (interrupt timing).
2. If mask bits are clear, $\overline{IRQ}$ and $\overline{FIRQ}$ must be held low for three cycles to guarantee that interrupt will be taken, although only one cycle is necessary to bring the processor out of SYNC.

Figure 18. Sync Timing

16-Bit Operation

The HD6309 has the capability of processing 16-bit data. These instructions include loads, stores, compares, adds, subtracts, transfers, exchanges, pushes and pulls.

Cycle-by-Cycle Operation

The address bus cycle-by-cycle performance chart illustrates the memory-access sequence corresponding to each possible instruction and address ing mode in the HD6309. Each instruction begins with an opcode fetch. While that opcode is being internally decoded, the next program byte is always fetched. (Most instructions will use the next byte, so this technique considerably speeds throughput.) Next, the operation of each opcode will follow the flow chart. $\overline{VMA}$ is an indication of $FFFF_{16}$ on the address bus, R/$\overline{W}$ = high and BS = low. The following examples illustrate the use of the chart : see figure 19.

Example 1: LBSR (Branch Taken)

Before Execution SP = F000

```
                          .
                          .
                          .
$8000                     LBSR    CAT
                          .
                          .
                          .
$A000          CAT        .
```

Cycle-by-Cycle Flow

Cycle #	Address	Data	R/$\overline{W}$	Description
1	8000	17	1	Opcode Fetch
2	8001	1F	1	Offset High Byte
3	8002	FD	1	Offset Low Byte
4	FFFF	*	1	$\overline{VMA}$ Cycle
5	FFFF	*	1	$\overline{VMA}$ Cycle
6	FFFF	*	1	$\overline{VMA}$ Cycle
7	FFFF	*	1	$\overline{VMA}$ Cycle
8	EFFF	03	0	Stack Low Order Byte of Return Address
9	EFFE	80	0	Stack High Order Byte of Return Address

Example 2: DEC (Extended)

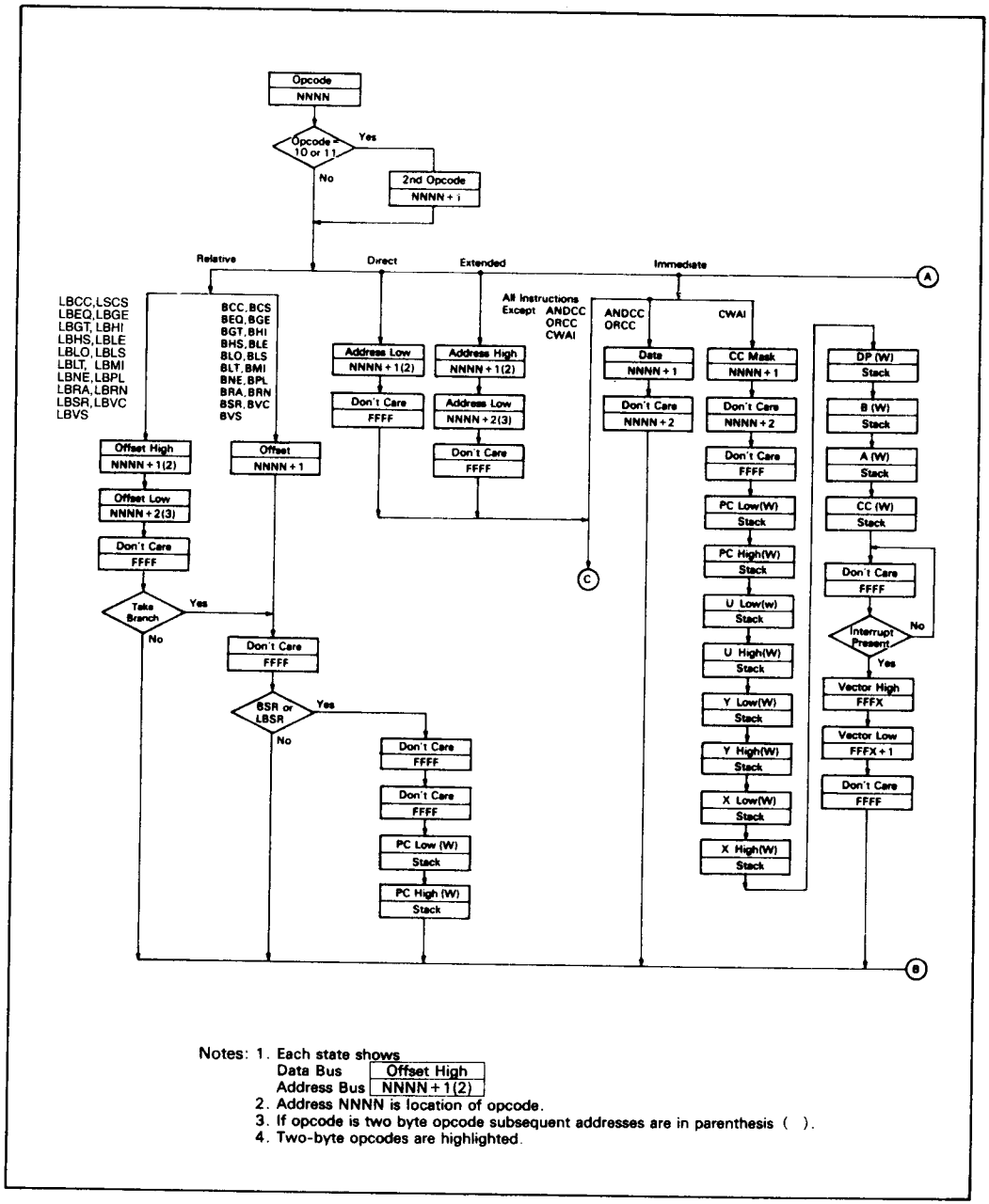
```
$8000          DEC          $A000
$A000          FCB          $80
```

Cycle-by-Cycle Flow

Cycle #	Address	Data	R/$\overline{W}$	Description
1	8000	7A	1	Opcode Fetch
2	8001	A0	1	Operand Address, High Byte
3	8002	00	1	Operand Address, Low Byte
4	FFFF	*	1	$\overline{VMA}$ Cycle
5	A000	80	1	Read the Data
6	FFFF	*	1	$\overline{VMA}$ Cycle
7	A000	7F	0	Store the Decremented Data

* The data bus has the data at that particular address.

2



Notes: 1. Each state shows
Data Bus | Offset High |
Address Bus | NNNN + 1(2) |
2. Address NNNN is location of opcode.
3. If opcode is two byte opcode subsequent addresses are in parenthesis ( ).
4. Two-byte opcodes are highlighted.

Figure 19. Cycle-by-Cycle Performance

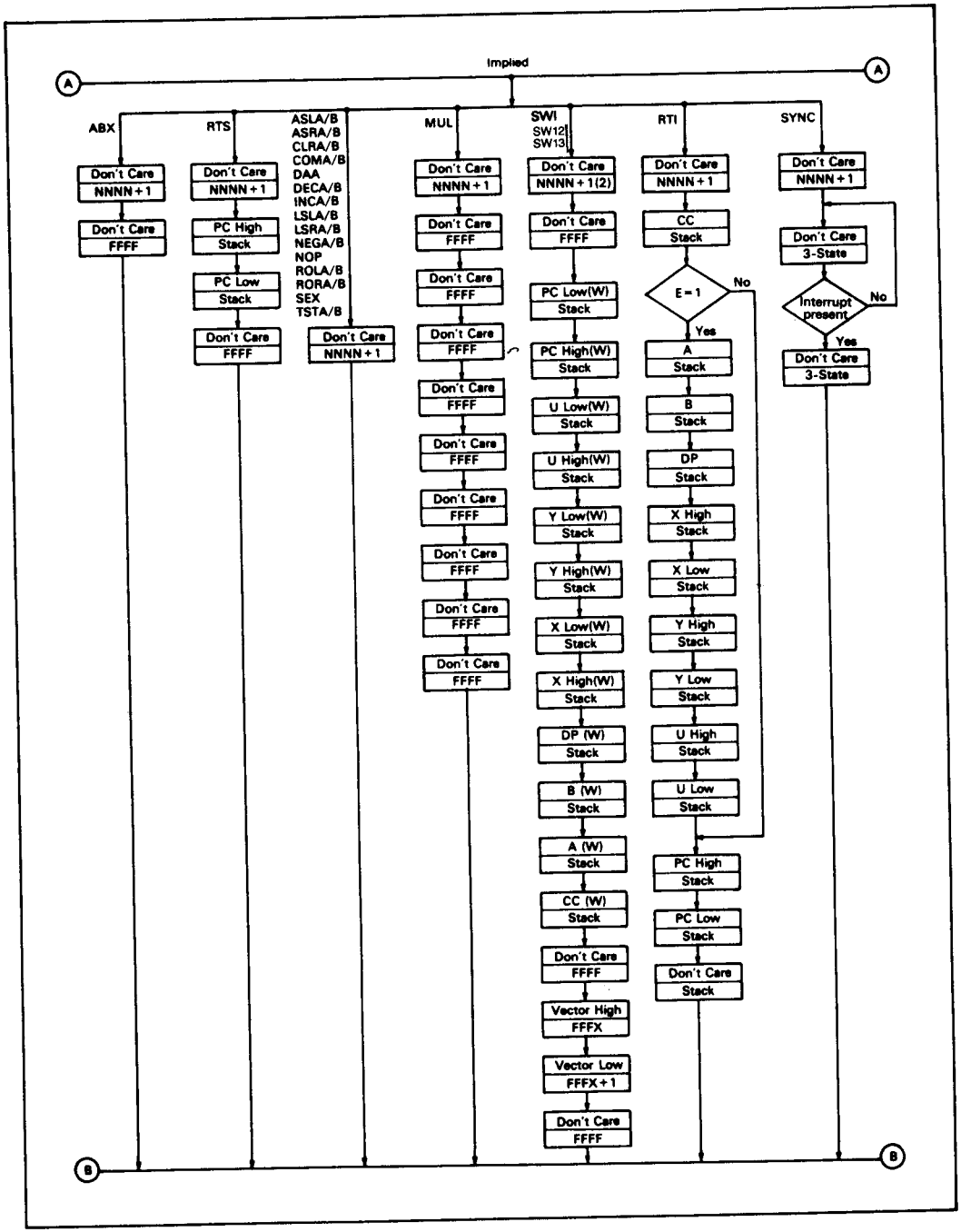


Figure 19. Cycle-by-Cycle Performance (Cont.)

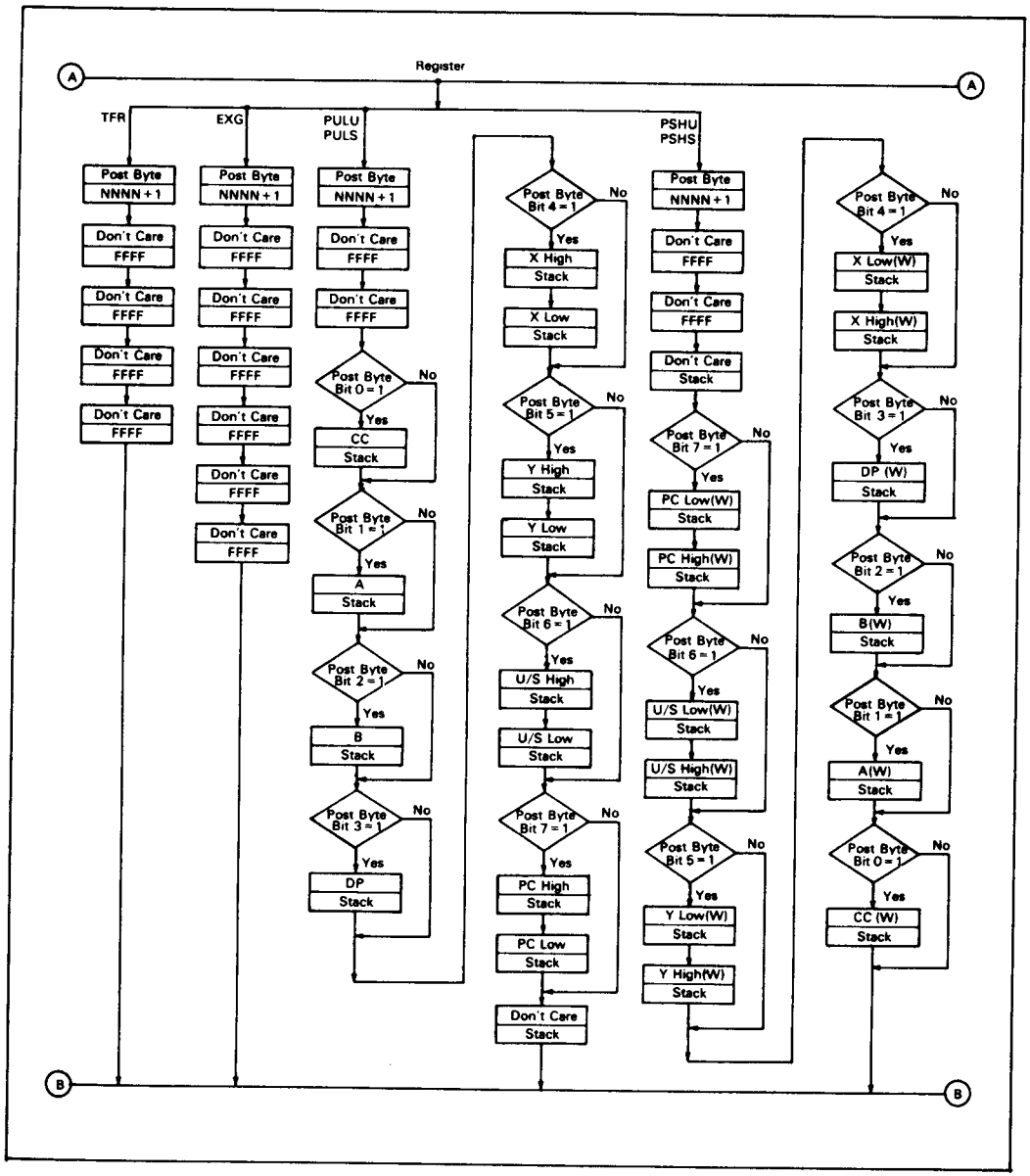


Figure 19. Cycle-by-Cycle Performance (Cont.)

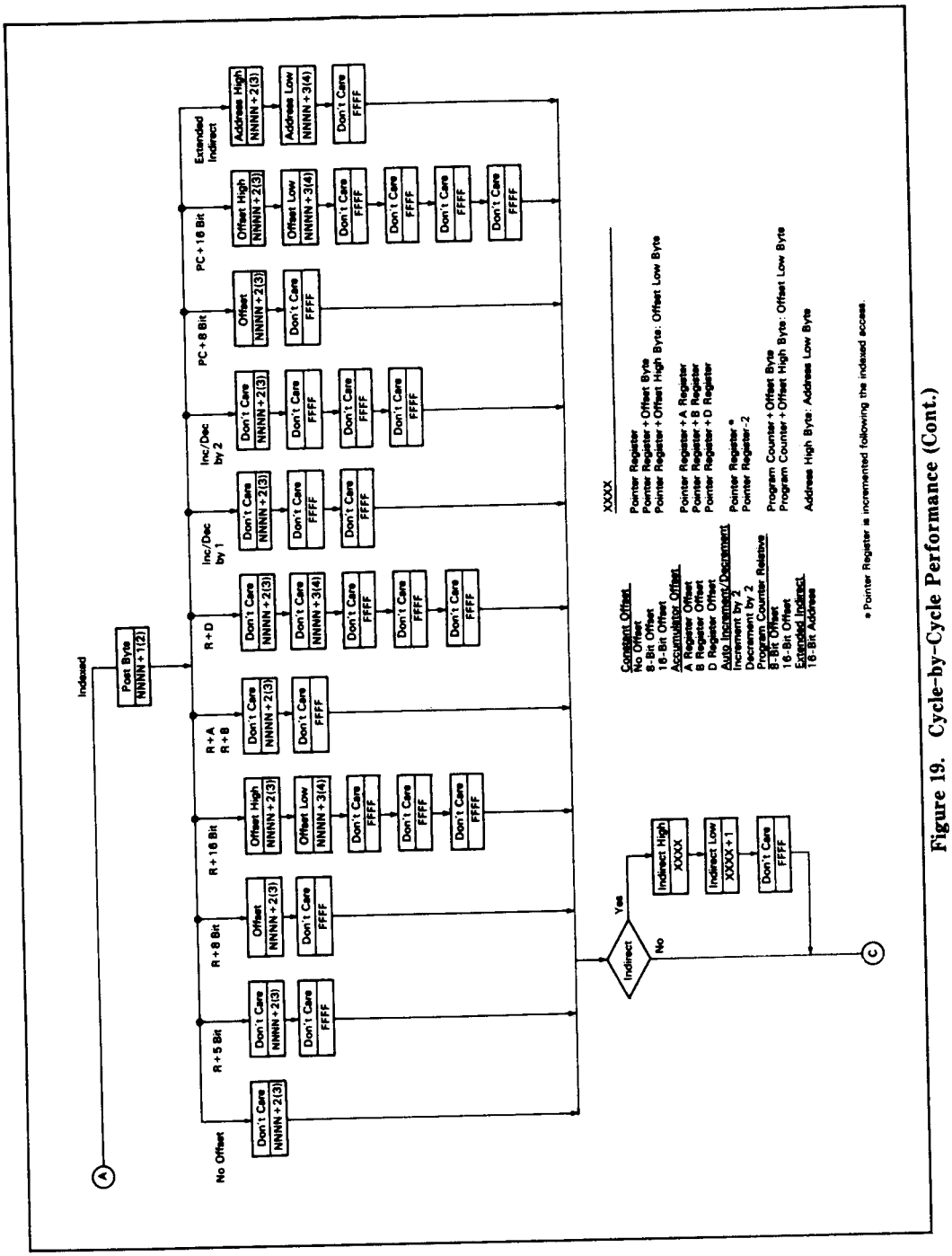


Figure 19. Cycle-by-Cycle Performance (Cont.)

2

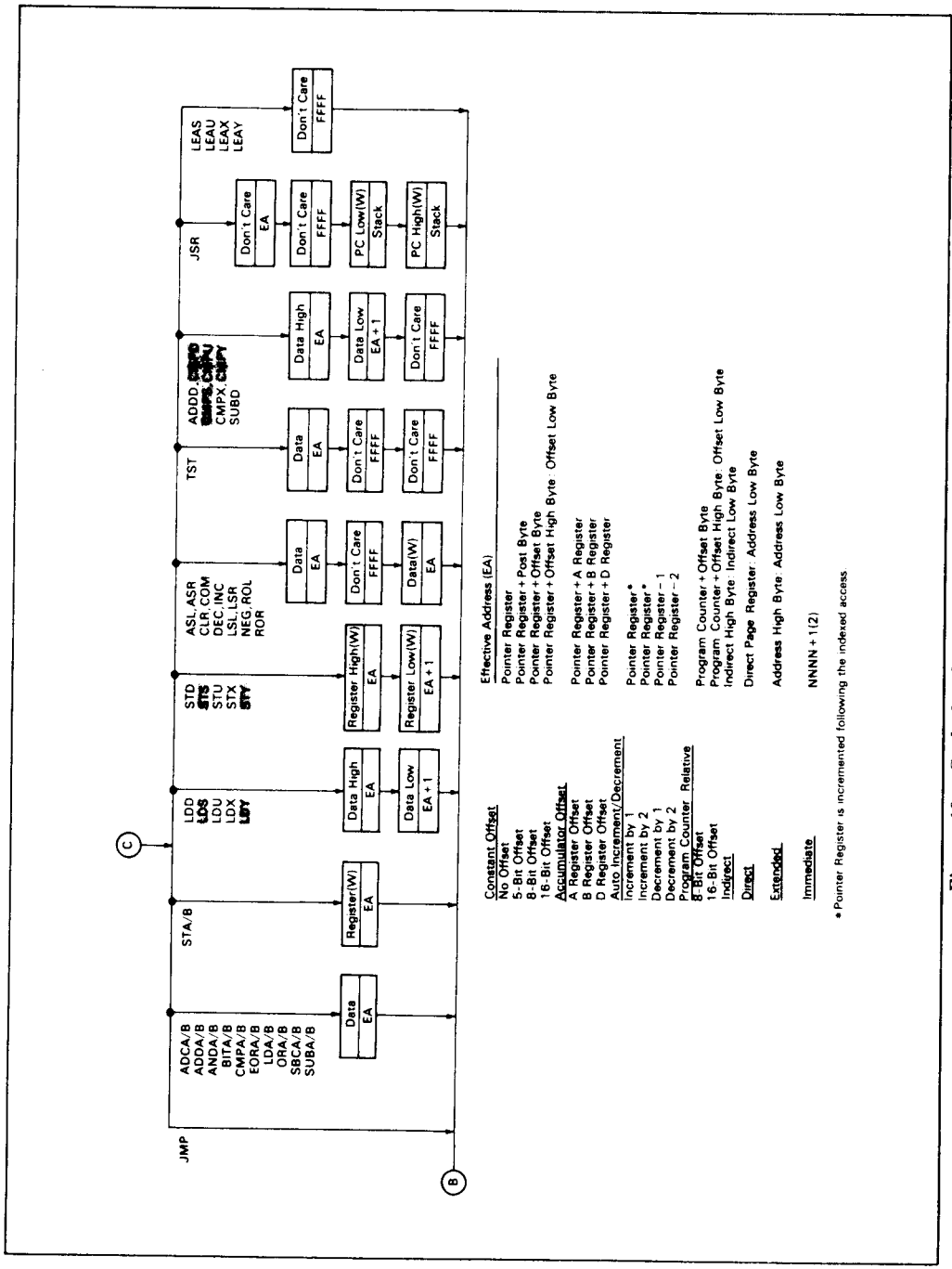


Figure 19. Cycle-by-Cycle Performance (Cont.)

Sleep Mode

During the interrupt wait period in the SYNC instruction (the sync state) and in the CWAI instruction (the wait state), MPU operation is halted and goes to the sleep mode. However, the state of I/O pins is the same as that of the HD6809 in this mode.

HD6309 Instruction set Tables

The instructions of the HD6309 have been broken down into five different categories. They are as follows:

- 8-Bit operation (table 6)
- 16-Bit operation (table 7)
- Index register/stack pointer instructions (table 8)
- Relative branches (long or short) (table 9)
- Miscellaneous instructions (table 10)

HD6309 instruction set tables and Hexadecimal Values of instructions are shown in table 11 and table 12.

Table 6. 8-Bit Accumulator and Memory Instructions

Mnemonic(s)	Operation
ADCA, ADCB	Add memory to accumulator with carry
ADDA, ADDB	Add memory to accumulator
ANDA, ANDB	AND memory with accumulator
ASL, ASLA, ASLB	Arithmetic shift of accumulator or memory left
ASR, ASRA, ASRB	Arithmetic shift of accumulator or memory right
BITA, BITB	Bit test memory with accumulator
CLR, CLRA, CLRB	Clear accumulator or memory location
CMPA, CMPB	Compare memory from accumulator
COM, COMA, COMB	Complement accumulator or memory location
DAA	Decimal adjust A accumulator
DEC, DECA, DECB	Decrement accumulator or memory location
EORA, EORB	Exclusive OR memory with accumulator
EXG R1, R2	Exchange R1 with R2 (R1, R2=A, B, CC, DP)
INC, INCA, INCB	Increment accumulator or memory location
LDA, LDB	Load accumulator from memory
LSL, LSLA, LSLB	Logical shift left accumulator or memory location
LSR, LSRA, LSRB	Logical shift right accumulator or memory location
MUL	Unsigned multiply (A×B→D)
NEG, NEGA, NEGB	Negate accumulator or memory
ORA, ORB	OR memory with accumulator
ROL, ROLA, ROLB	Rotate accumulator or memory left
ROR, RORA, RORB	Rotate accumulator or memory right
SBCA, SBCB	Subtract memory from accumulator with borrow
STA, STB	Store accumulator to memory
SUBA, SUBB	Subtract memory from accumulator
TST, TSTA, TSTB	Test accumulator or memory location
TFR R1, R2	Transfer R1 or R2 (R1, R2=A, B, CC, DP)

Note: A, B, CC or DP may be pushed to (pulled from) either stack with PSHS, PSHU (PULS, PULU) instructions.

Table 7. 16-Bit Accumulator and Memory Instructions

Mnemonic(s)	Operation
ADDD	Add memory to D accumulator
CMPD	Compare memory from D accumulator
EXG D, R	Exchange D with X, Y, S, U or PC
LDD	Load D accumulator from memory
SEX	Sign Extend B accumulator into A accumulator
STD	Store D accumulator to memory
SUBD	Subtract memory from D accumulator
TFR D, R	Transfer D to X, Y, S, U or PC
TFR R, D	Transfer X, Y, S, U or PC to D

Note: D may be pushed (pulled) to either stack with PSHS, PSHU (PULS, PULU) instructions.

Table 8. Index Register/Stack Pointer Instructions

Mnemonic(s)	Operation
CMPS, CMPU	Compare memory from stack pointer
CMPX, CMPY	Compare memory from index register
EXG R1, R2	Exchange D, X, Y, S, U or PC with D, X, Y, S, U or PC
LEAS, LEAU	Load effective address into stack pointer
LEAX, LEAY	Load effective address into index register
LDS, LDU	Load stack pointer from memory
LDX, LDY	Load index register from memory
PSHS	Push A, B, CC, DP, D, X, Y, U or PC onto hardware stack
PSHU	Push A, B, CC, DP, D, X, Y, S or PC onto user stack
PULS	Pull A, B, CC, DP, D, X, Y, U or PC from hardware stack
PULU	Pull A, B, CC, DP, D, X, Y, S or PC from user stack
STS, STU	Store stack pointer to memory
STX, STY	Store index register to memory
TFR R1, R2	Transfer D, X, Y, S, U or PC to D, X, Y, S, U or PC
ABX	Add B accumulator to X (unsigned)

Table 9. Branch Instructions

Mnemonic(s)	Operation
	Simple Branches
BEQ, LBEQ	Branch if equal
BNE, LBNE	Branch if not equal
BMI, LBMI	Branch if minus
BPL, LBPL	Branch if plus
BCS, LBCS	Branch if carry set
BCC, LBCC	Branch if carry clear
BVS, LBVS	Branch if overflow set
BVC, LBVC	Branch if overflow clear
	Signed Branches
BGT, LBGT	Branch if greater (signed)
BGE, LBGE	Branch if greater than or equal (signed)
BEQ, LBEQ	Branch if equal
BLE, LBLE	Branch if less than or equal (signed)
BLT, LBLT	Branch if less than (signed)
	Unsigned Branches
BHI, LBHI	Branch if higher (unsigned)
BHS, LBHS	Branch if higher or same (unsigned)
BEQ, LBEQ	Branch if equal
BLS, LBLS	Branch if lower or same (unsigned)
BLO, LBLO	Branch if lower (unsigned)
	Other Branches
BSR, LBSR	Branch to subroutine
BRA, LBRA	Branch always
BRN, LBRN	Branch never

Table 10. Miscellaneous Instructions

Mnemonic(s)	Operation
ANDCC	AND condition code register
CWAI	AND condition code register, then wait for interrupt
NOP	No operation
ORCC	OR condition code register
JMP	Jump
JSR	Jump to subroutine
RTI	Return from interrupt
RTS	Return from subroutine
SWI, SWI2, SWI3	Software interrupt (absolute indirect)
SYNC	Synchronize with interrupt line

Table 11. HD6309 Instruction Set Table

INSTRUCTIONS/ FORMS		IMP ACCM REG			DIRECT			EXTND			IMMED			INDEX①			RELATIVE			DESCRIPTION	7	6	5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~⑤	#		E	F	H	I	N	Z	V	C
ABX		3A	3	1																B+X→X (Unsigned)	●	●	●	●	●	●	●	●
ADC	ADCA				99	4	2	B9	5	3	89	2	2	A9	4+	2+				A+M+C→A	●	●	↕	●	↕	↕	↕	↕
	ADCB				D9	4	2	F9	5	3	C9	2	2	E9	4+	2+				B+M+C→B	●	●	↕	●	↕	↕	↕	↕
ADD	ADDA				9B	4	2	BB	5	3	8B	2	2	AB	4+	2+				A+M→A	●	●	↕	●	↕	↕	↕	↕
	ADDB				DB	4	2	FB	5	3	CB	2	2	EB	4+	2+				B+M→B	●	●	↕	●	↕	↕	↕	↕
	ADDD				D3	6	2	F3	7	3	C3	4	3	E3	6+	2+				D+M : M+1→D	●	●	●	●	↕	↕	↕	↕
AND	ANDA				94	4	2	B4	5	3	84	2	2	A4	4+	2+				A∧M→A	●	●	●	●	↕	↕	R	●
	ANDB				D4	4	2	F4	5	3	C4	2	2	E4	4+	2+				B∧M→B	●	●	●	●	↕	↕	R	●
	ANDCC										1C	3	2							CC∧IMM→CC	(			⑦				)
ASL	ASLA	48	2	1																A, B, M (C b7 b0)	●	●	⑧	●	↕	↕	↕	↕
	ASLB	58	2	1																	●	●	⑧	●	↕	↕	↕	↕
	ASL				08	6	2	78	7	3				68	6+	2+					●	●	⑧	●	↕	↕	↕	↕
ASR	ASRA	47	2	1																A, B, M (b7 b0 C)	●	●	⑧	●	↕	↕	●	↕
	ASRB	57	2	1																	●	●	⑧	●	↕	↕	●	↕
	ASR				07	6	2	77	7	3				67	6+	2+					●	●	⑧	●	↕	↕	●	↕
BCC	BCC																24	3	2	Branch C=0	●	●	●	●	●	●	●	●
	LBCC																10 24	5(6)	4	Long Branch C=0	●	●	●	●	●	●	●	●
BCS	BCS																25	3	2	Branch C=1	●	●	●	●	●	●	●	●
	LBCS																10 25	5(6)	4	Long Branch C=1	●	●	●	●	●	●	●	●
BEQ	BEQ																27	3	2	Branch Z=1	●	●	●	●	●	●	●	●
	LBEQ																10 27	5(6)	4	Long Branch Z=1	●	●	●	●	●	●	●	●
BGE	BGE																2C	3	2	Branch N⊕V=0	●	●	●	●	●	●	●	●
	LBGE																10 2C	5(6)	4	Long Branch N⊕V=0	●	●	●	●	●	●	●	●
BGT	BGT																2E	3	2	Branch Z∨(N⊕V)=0	●	●	●	●	●	●	●	●
	LBGT																10 2E	5(6)	4	Long Branch Z∨(N⊕V)=0	●	●	●	●	●	●	●	●
BHI	BHI																22	3	2	Branch C∨Z=0	●	●	●	●	●	●	●	●
	LBHI																10 22	5(6)	4	Long Branch C∨Z=0	●	●	●	●	●	●	●	●
BHS	BHS																24	3	2	Branch C=0	●	●	●	●	●	●	●	●
	LBHS																10 24	5(6)	4	Long Branch C=0	●	●	●	●	●	●	●	●
BIT	BITA				95	4	2	B5	5	3	85	2	2	A5	4+	2+				Bit Test A (M∧A)	●	●	●	●	↕	↕	R	●
	BITB				D5	4	2	F5	5	3	C5	2	2	E5	4+	2+				Bit Test B (M∧B)	●	●	●	●	↕	↕	R	●
BLE	BLE																2F	3	2	Branch Z∨(N⊕V)=1	●	●	●	●	●	●	●	●
	LBLE																10 2F	5(6)	4	Long Branch Z∨(N⊕V)=1	●	●	●	●	●	●	●	●
BLO	BLO																25	3	2	Branch C=1	●	●	●	●	●	●	●	●
	LBLO																10 25	5(6)	4	Long Branch C=1	●	●	●	●	●	●	●	●
BLS	BLS																23	3	2	Branch C∨Z=1	●	●	●	●	●	●	●	●
	LBLS																10 23	5(6)	4	Long Branch C∨Z=1	●	●	●	●	●	●	●	●
BLT	BLT																2D	3	2	Branch N⊕V=1	●	●	●	●	●	●	●	●
	LBLT																10 2D	5(6)	4	Long Branch N⊕V=1	●	●	●	●	●	●	●	●
BMI	BMI																2B	3	2	Branch N=1	●	●	●	●	●	●	●	●
	LBMI																10 2B	5(6)	4	Long Branch N=1	●	●	●	●	●	●	●	●

(Continued)

INSTRUCTIONS/ FORMS		IMP ACCM REG			DIRECT			EXTND			IMMED			INDEX①			RELATIVE			DESCRIPTION	7	6	5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~⑤	#		E	F	H	I	N	Z	V	C
BNE	BNE																26	3	2	Branch Z=0	●	●	●	●	●	●	●	●
	LBNE																10 26	5(6)	4	Long Branch Z=0	●	●	●	●	●	●	●	●
BPL	BPL																2A	3	2	Branch N=0	●	●	●	●	●	●	●	●
	LBPL																10 2A	5(6)	4	Long Branch N=0	●	●	●	●	●	●	●	●
BRA	BRA																20	3	2	Branch Always	●	●	●	●	●	●	●	●
	LBRA																16	5	3	Long Branch Always	●	●	●	●	●	●	●	●
BRN	BRN																21	3	2	Branch Never	●	●	●	●	●	●	●	●
	LBRN																10 21	5	4	Long Branch Never	●	●	●	●	●	●	●	●
BSR	BSR																8D	7	2	Branch to Subroutine	●	●	●	●	●	●	●	●
	LBSR																17	9	3	Long Branch to Subroutine	●	●	●	●	●	●	●	●
BVC	BVC																28	3	2	Branch V=0	●	●	●	●	●	●	●	●
	LBVC																10 28	5(6)	4	Long Branch V=0	●	●	●	●	●	●	●	●
BVS	BVS																29	3	2	Branch V=1	●	●	●	●	●	●	●	●
	LBVS																10 29	5(6)	4	Long Branch V=1	●	●	●	●	●	●	●	●
CLR	CLRA	4F	2	1																0→A	●	●	●	●	R	S	R	R
	CLRB	5F	2	1																0→B	●	●	●	●	R	S	R	R
	CLR				0F	6	2	7F	7	3				6F	6+	2+				0→M	●	●	●	●	R	S	R	R
CMP	CMPA				91	4	2	B1	5	3	81	2	2	A1	4+	2+				Compare M from A	●	●	⑧	●	↕	↕	↕	↕
	CMPB				D1	4	2	F1	5	3	C1	2	2	E1	4+	2+				Compare M from B	●	●	⑧	●	↕	↕	↕	↕
	CMPD				10 93	7	3	10 B3	8	4	10 83	5	4	10 A3	7+	3+				Compare M : M+1 from D	●	●	●	●	↕	↕	↕	↕
	CMPS				11 9C	7	3	11 BC	8	4	11 8C	5	4	11 AC	7+	3+				Compare M : M+1 from S	●	●	●	●	↕	↕	↕	↕
	CMPU				11 93	7	3	11 B3	8	4	11 83	5	4	11 A3	7+	3+				Compare M : M+1 from U	●	●	●	●	↕	↕	↕	↕
	CMPX				9C	6	2	BC	7	3	8C	4	3	AC	6+	2+				Compare M : M+1 from X	●	●	●	●	↕	↕	↕	↕
	CMPY				10 9C	7	3	10 BC	8	4	10 8C	5	4	10 AC	7+	3+				Compare M : M+1 from Y	●	●	●	●	↕	↕	↕	↕
COM	COMA	43	2	1																$\overline{A}$→A	●	●	●	●	↕	↕	R	S
	COMB	53	2	1																$\overline{B}$→B	●	●	●	●	↕	↕	R	S
	COM				03	6	2	73	7	3				63	6+	2+				$\overline{M}$→M	●	●	●	●	↕	↕	R	S
CWAI											3C	≥20	2							CC∧IMM→CC : Wait for Interrupt	S	(			⑦			)
DAA		19	2	1																Decimal Adjust A	●	●	●	●	↕	↕	⑧	↕
DEC	DECA	4A	2	1																A−1→A	●	●	●	●	↕	↕	↕	●
	DECB	5A	2	1																B−1→B	●	●	●	●	↕	↕	↕	●
	DEC				0A	6	2	7A	7	3				6A	6+	2+				M−1→M	●	●	●	●	↕	↕	↕	●
EOR	EORA				98	4	2	B8	5	3	88	2	2	A8	4+	2+				A⊕M→A	●	●	●	●	↕	↕	R	●
	EORB				D8	4	2	F8	5	3	C8	2	2	E8	4+	2+				B⊕M→B	●	●	●	●	↕	↕	R	●
EXG	R1, R2	1E	8	2																R1←R2②	(				⑩			)
INC	INCA	4C	2	1																A+1→A	●	●	●	●	↕	↕	↕	●
	INCB	5C	2	1																B+1→B	●	●	●	●	↕	↕	↕	●
	INC				0C	6	2	7C	7	3				6C	6+	2+				M+1→M	●	●	●	●	↕	↕	↕	●
JMP					0E	3	2	7E	4	3				6E	3+	2+				EA③→PC	●	●	●	●	●	●	●	●
JSR					9D	7	2	BD	8	3				AD	7+	2+				Jump to Subroutine	●	●	●	●	●	●	●	●

(Continued)

2

INSTRUCTIONS/ FORMS		IMP / ACCM REG			DIRECT			EXTND			IMMED			INDEX①			RELATIVE			DESCRIPTION	7	6	5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~⑤	#		E	F	H	I	N	Z	V	C
LD	LDA				96	4	2	B6	5	3	86	2	2	A6	4+	2+				M→A	●	●	●	●	↕	↕	R	●
	LDB				D6	4	2	F6	5	3	C6	2	2	E6	4+	2+				M→B	●	●	●	●	↕	↕	R	●
	LDD				DC	5	2	FC	6	3	CC	3	3	EC	5+	2+				M : M+1→D	●	●	●	●	↕	↕	R	●
	LDS				10	6	3	10	7	4	10	4	4	10	6+	3+				M : M+1→S	●	●	●	●	↕	↕	R	●
					DE			FE			CE			EE														
	LDU				DE	5	2	FE	6	3	CE	3	3	EE	5+	2+				M : M+1→U	●	●	●	●	↕	↕	R	●
	LDX				9E	5	2	BE	6	3	8E	3	3	AE	5+	2+				M : M+1→X	●	●	●	●	↕	↕	R	●
	LDY				10	6	3	10	7	4	10	4	4	10	6+	3+				M : M+1→Y	●	●	●	●	↕	↕	R	●
					9E			BE			8E			AE														
LEA	LEAS													32	4+	2+				EA③→S	●	●	●	●	●	●	●	●
	LEAU													33	4+	2+				EA③→U	●	●	●	●	●	●	●	●
	LEAX													30	4+	2+				EA③→X	●	●	●	●	●	↕	●	●
	LEAY													31	4+	2+				EA③→Y	●	●	●	●	●	↕	●	●
LSL	LSLA	48	2	1																A ⎫ B ⎬ M ⎭ C b7 b0	●	●	●	●	↕	↕	↕	↕
	LSLB	58	2	1																	●	●	●	●	↕	↕	↕	↕
	LSL				08	6	2	78	7	3				68	6+	2+					●	●	●	●	↕	↕	↕	↕
LSR	LSRA	44	2	1																A ⎫ B ⎬ M ⎭ 0 b7 b0 C	●	●	●	●	R	↕	●	↕
	LSRB	54	2	1																	●	●	●	●	R	↕	●	↕
	LSR				04	6	2	74	7	3				64	6+	2+					●	●	●	●	R	↕	●	↕
MUL		3D	11	1																A×B→D (Unsigned)	●	●	●	●	●	↕	●	⑨
NEG	NEGA	40	2	1																$\overline{A}$+1→A	●	●	⑧	●	↕	↕	↕	↕
	NEGB	50	2	1																$\overline{B}$+1→B	●	●	⑧	●	↕	↕	↕	↕
	NEG				00	6	2	70	7	3				60	6+	2+				$\overline{M}$+1→M	●	●	⑧	●	↕	↕	↕	↕
NOP		12	2	1																No Operation	●	●	●	●	●	●	●	●
OR	ORA				9A	4	2	BA	5	3	8A	2	2	AA	4+	2+				A∨M→A	●	●	●	●	↕	↕	R	●
	ORB				DA	4	2	FA	5	3	CA	2	2	EA	4+	2+				B∨M→B	●	●	●	●	↕	↕	R	●
	ORCC										1A	3	2							CC∨IMM→CC	(	―	―	⑦	―	―	―	)
PSH	PSHS	34	5+④	2																Push Registers on S Stack	●	●	●	●	●	●	●	●
	PSHU	36	5+④	2																Push Registers on U Stack	●	●	●	●	●	●	●	●
PUL	PULS	35	5+④	2																Pull Registers from S Stack	(	―	―	⑩	―	―	―	)
	PULU	37	5+④	2																Pull Registers from U Stack	(	―	―	⑩	―	―	―	)
ROL	ROLA	49	2	1																A ⎫ B ⎬ M ⎭ C b7 b0	●	●	●	●	↕	↕	↕	↕
	ROLB	59	2	1																	●	●	●	●	↕	↕	↕	↕
	ROL				09	6	2	79	7	3				69	6+	2+					●	●	●	●	↕	↕	↕	↕
ROR	RORA	46	2	1																A ⎫ B ⎬ M ⎭ C b7 b0	●	●	●	●	↕	↕	●	↕
	RORB	56	2	1																	●	●	●	●	↕	↕	●	↕
	ROR				06	6	2	76	7	3				66	6+	2+					●	●	●	●	↕	↕	●	↕
RTI		3B	6/15	1																Return from Interrupt	(	―	―	⑦	―	―	―	)
RTS		39	5	1																Return from Subroutine	●	●	●	●	●	●	●	●
SBC	SBCA				92	4	2	B2	5	3	82	2	2	A2	4+	2+				A−M−C→A	●	●	⑧	●	↕	↕	↕	↕
	SBCB				D2	4	2	F2	5	3	C2	2	2	E2	4+	2+				B−M−C→B	●	●	⑧	●	↕	↕	↕	↕
SEX		1D	2	1																Sign Extend B into A (Bのビット7=1 FF→A / Bのビット7=0 0→A)	●	●	●	●	↕	↕	●	●

(Continued)

INSTRUCTIONS/ FORMS		IMP ACCM REG			DIRECT			EXTND			IMMED			INDEX①			RELATIVE			DESCRIPTION	7	6	5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~⑤	#		E	F	H	I	N	Z	V	C
ST	STA				97	4	2	B7	5	3				A7	4+	2+				A→M	●	●	●	●	↕	↕	R	●
	STB				D7	4	2	F7	5	3				E7	4+	2+				B→M	●	●	●	●	↕	↕	R	●
	STD				DD	5	2	FD	6	3				ED	5+	2+				D→M : M+1	●	●	●	●	↕	↕	R	●
	STS				10	6	3	10	7	4				10	6+	3+				S→M : M+1	●	●	●	●	↕	↕	R	●
					DF			FF						EF														
	STU				DF	5	2	FF	6	3				EF	5+	2+				U→M : M+1	●	●	●	●	↕	↕	R	●
	STX				9F	5	2	BF	6	3				AF	5+	2+				X→M : M+1	●	●	●	●	↕	↕	R	●
	STY				10	6	3	10	7	4				10	6+	3+				Y→M : M+1	●	●	●	●	↕	↕	R	●
					9F			BF						AF														
SUB	SUBA				90	4	2	B0	5	3	80	2	2	A0	4+	2+				A−M→A	●	●	⑧	●	↕	↕	↕	↕
	SUBB				D0	4	2	F0	5	3	C0	2	2	E0	4+	2+				B−M→B	●	●	⑧	●	↕	↕	↕	↕
	SUBD				93	6	2	B3	7	3	83	4	3	A3	6+	2+				D−M : M+1→D	●	●	●	●	↕	↕	↕	↕
SWI	SWI⑥	3F	19	1																Software interrupt 1	S	S	●	S	●	●	●	●
	SWI2⑥	10	20	2																Software interrupt 2	S	●	●	●	●	●	●	●
		3F																										
	SWI3⑥	11	20	2																Software interrupt 3	S	●	●	●	●	●	●	●
		3F																										
SYNC		13	≧4	1																Synchronize to interrupt	●	●	●	●	●	●	●	●
TFR	R1,R2	1F	6	2																R1→R2②	(			⑩				)
TST	TSTA	4D	2	1																Test A	●	●	●	●	↕	↕	R	●
	TSTB	5D	2	1																Test B	●	●	●	●	↕	↕	R	●
	TST				0D	6	2	7D	7	3				6D	6+	2+				Test M	●	●	●	●	↕	↕	R	●

(NOTES)

① This column gives a base cycle and byte count. To obtain total count, and the values obtained from the INDEXED ADDRESSING MODES table.
② R1 and R2 may be any pair of 8 bit or any pair of 16 bit registers.
The 8 bit registers are: A, B, CC, DP
The 16 bit registers are: X, Y, U, S, D, PC
③ EA is the effective address.
④ The PSH and PUL instructions require 5 cycle plus 1 cycle for each byte pushed or pulled.
⑤ 5(6) means: 5 cycles if branch not taken, 6 cycles if taken.
⑥ SWI sets 1 and F bits. SW12 and SW13 do not affect I and F.
⑦ Conditions Codes set as a direct result of the instruction.
⑧ Value of half-carry flag is undefined.
⑨ Special Case—Carry set if b7 is SET.
⑩ Condition Codes set as a direct result of the instruction if CC is specified, and not affected otherwise.

LEGEND:

OP	Operation Code (Hexadecimal)	Z	Zero (byte)
~	Number of MPU Cycles	V	Overflow, 2's complement
#	Number of Program Bytes	C	Carry from bit 7
+	Arithmetic Plus	↕	Test and set if true, cleared otherwise
−	Arithmetic Minus	●	Not Affected
×	Multiply	CC	Condition Code Register
$\overline{M}$	Complement of M	:	Concatenation
→	Transfer Into	∨	Logical or
H	Half-carry (from bit 3)	∧	Logical and
N	Negative (sign bit)	⊻	Logical Exclusive or

2

Table 12. Hexadecimal Values of Machine Codes

OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#
00	NEG	Direct	6	2	30	LEAX	Indexed	4+	2+	60	NEG	Indexed	6+	2+
01	*	↑			31	LEAY	↑	4+	2+	61	*	↑		
02	*				32	LEAS	↓	4+	2+	62	*			
03	COM		6	2	33	LEAU	Indexed	4+	2+	63	COM		6+	2+
04	LSR		6	2	34	PSHS	Implied	5+	2	64	LSR		6+	2+
05	*				35	PULS	↑	5+	2	65	*			
06	ROR		6	2	36	PSHU		5+	2	66	ROR		6+	2+
07	ASR		6	2	37	PULU		5+	2	67	ASR		6+	2+
08	ASL, LSL		6	2	38	*				68	ASL, LSL		6+	2+
09	ROL		6	2	39	RTS		5	1	69	ROL		6+	2+
0A	DEC		6	2	3A	ABX		3	1	6A	DEC		6+	2+
0B	*				3B	RTI	Implied	6, 15	1	6B	*			
0C	INC		6	2	3C	CWAI	Immed	≥20	2	6C	INC		6+	2+
0D	TST		6	2	3D	MUL	Implied	11	1	6D	TST		6+	2+
0E	JMP	↓	3	2	3E	*				6E	JMP	↓	3+	2+
0F	CLR	Direct	6	2	3F	SWI	Implied	19	1	6F	CLR	Indexed	6+	2+
10	See	–	–	–	40	NEGA	Implied	2	1	70	NEG	Extended	7	3
11	Next Page	–	–	–	41	*	↑			71	*	↑		
12	NOP	Implied	2	1	42	*				72	*			
13	SYNC	Implied	≥4	1	43	COMA		2	1	73	COM		7	3
14	*				44	LSRA		2	1	74	LSR		7	3
15	*				45	*				75	*			
16	LBRA	Relative	5	3	46	RORA		2	1	76	ROR		7	3
17	LBSR	Relative	9	3	47	ASRA		2	1	77	ASR		7	3
18	*				48	ASLA, LSLA		2	1	78	ASL, LSL		7	3
19	DAA	Implied	2	1	49	ROLA		2	1	79	ROL		7	3
1A	ORCC	Immed	3	2	4A	DECA		2	1	7A	DEC		7	3
1B	*				4B	*				7B	*			
1C	ANDCC	Immed	3	2	4C	INCA		2	1	7C	INC		7	3
1D	SEX	Implied	2	1	4D	TSTA		2	1	7D	TST		7	3
1E	EXG	↕	8	2	4E	*	↓			7E	JMP	↓	4	3
1F	TFR	Implied	6	2	4F	CLRA	Implied	2	1	7F	CLR	Extended	7	3
20	BRA	Relative	3	2	50	NEGB	Implied	2	1	80	SUBA	Immed	2	2
21	BRN	↑	3	2	51	*	↑			81	CMPA	↑	2	2
22	BHI		3	2	52	*				82	SBCA		2	2
23	BLS		3	2	53	COMB		2	1	83	SUBD		4	3
24	BHS, BCC		3	2	54	LSRB		2	1	84	ANDA		2	2
25	BLO, BCS		3	2	55	*				85	BITA		2	2
26	BNE		3	2	56	RORB		2	1	86	LDA		2	2
27	BEQ		3	2	57	ASRB		2	1	87	*			
28	BVC		3	2	58	ASLB, LSLB		2	1	88	EORA		2	2
29	BVS		3	2	59	ROLB		2	1	89	ADCA		2	2
2A	BPL		3	2	5A	DECB		2	1	8A	ORA		2	2
2B	BMI		3	2	5B	*				8B	ADDA		2	2
2C	BGE		3	2	5C	INCB		2	1	8C	CMPX	Immed	4	3
2D	BLT		3	2	5D	TSTB		2	1	8D	BSR	Relative	7	2
2E	BGT		3	2	5E	*				8E	LDX	Immed	3	3
2F	BLE	Relative	3	2	5F	CLRB	Implied	2	1	8F	*			

Legend: ~ Number of MPU cycles (less possible push pull or indexed-mode cycles)
Number of program bytes
* Denotes unused opcode

Table 12. Hexadecimal Values of Machine Codes (Cont.)

OP	Mnem	Mode	~	#
90	SUBA	Direct	4	2
91	CMPA	↑	4	2
92	SBCA		4	2
93	SUBD		6	2
94	ANDA		4	2
95	BITA		4	2
96	LDA		4	2
197	STA		4	2
98	EORA		4	2
99	ADCA		4	2
9A	ORA		4	2
9B	ADDA		4	2
9C	CMPX		6	2
9D	JSR		7	2
9E	LDX	↓	5	2
9F	STX	Direct	5	2
A0	SUBA	Indexed	4+	2+
A1	CMPA	↑	4+	2+
A2	SBCA		4+	2+
A3	SUBD		6+	2+
A4	ANDA		4+	2+
A5	BITA		4+	2+
A6	LDA		4+	2+
A7	STA		4+	2+
A8	EORA		4+	2+
A9	ADCA		4+	2+
AA	ORA		4+	2+
AB	ADDA		4+	2+
AC	CMPX		6+	2+
AD	JSR		7+	2+
AE	LDX	↓	5+	2+
AF	STX	Indexed	5+	2+
B0	SUBA	Extended	5	3
B1	CMPA	↑	5	3
B2	SBCA		5	3
B3	SUBD		7	3
B4	ANDA		5	3
B5	BITA		5	3
B6	LDA		5	3
B7	STA		5	3
B8	EORA		5	3
B9	ADCA		5	3
BA	ORA		5	3
BB	ADDA		5	3
BC	CMPX		7	3
BD	JSR		8	3
BE	LDX	↓	6	3
BF	STX	Extended	6	3
C0	SUBB	Immed	2	2
C1	CMPB	↑	2	2
C2	SBCB		2	2
C3	ADDD		4	3
C4	ANDB		2	2
C5	BITB	Immed	2	2
C6	LDB	Immed	2	2
C7	*	↑		
C8	EORB		2	2
C9	ADCB		2	2
CA	ORB		2	2
CB	ADDB		2	2
CC	LDD		3	3
CD	*			
CE	LDU	Immed	3	3
CF	*			
D0	SUBB	Direct	4	2
D1	CMPB	↑	4	2
D2	SBCB		4	2
D3	ADDD		6	2
D4	ANDB		4	2
D5	BITB		4	2
D6	LDB		4	2
D7	STB		4	2
D8	EORB		4	2
D9	ADCB		4	2
DA	ORB		4	2
DB	ADDB		4	2
DC	LDD		5	2
DD	STD		5	2
DE	LDU	↓	5	2
DF	STU	Direct	5	2
E0	SUBB	Indexed	4+	2+
E1	CMPB	↑	4+	2+
E2	SBCB		4+	2+
E3	ADDD		6+	2+
E4	ANDB		4+	2+
E5	BITB		4+	2+
E6	LDB		4+	2+
E7	STB		4+	2+
E8	EORB		4+	2+
E9	ADCB		4+	2+
EA	ORB		4+	2+
EB	ADDB		4+	2+
EC	LDD		5+	2+
ED	STD		5+	2+
EE	LDU		5+	2+
EF	STU	Indexed	5+	2+
F0	SUBB	Extended	5	3
F1	CMPB	↑	5	3
F2	SBCB		5	3
F3	ADDD		7	3
F4	ANDB		5	3
F5	BITB		5	3
F6	LDB		5	3
F7	STB		5	3
F8	EORB		5	3
F9	ADCB		5	3
FA	ORB		5	3
FB	ADDB	Extended	5	3
FC	LDD	Extended	6	3
FD	STD	↑	6	3
FE	LDU		6	3
FF	STU	Extended	6	3
2 Bytes Opcode				
1021	LBRN	Relative	5	4
1022	LBHI	↑	5(6)	4
1023	LBLS		5(6)	4
1024	LBHS, LBCC		5(6)	4
1025	LBCS, LBLO		5(6)	4
1026	LBNE		5(6)	4
1027	LBEQ		5(6)	4
1028	LBVC		5(6)	4
1029	LBVS		5(6)	4
102A	LBPL		5(6)	4
102B	LBMI		5(6)	4
102C	LBGE		5(6)	4
102D	LBLT		5(6)	4
102E	LBGT	↓	5(6)	4
102F	LBLE	Relative	5(6)	4
103F	SWI2	Implied	20	2
1083	CMPD	Immed	5	4
108C	CMPY	↕	5	4
108E	LDY	Immed	4	4
1093	CMPD	Direct	7	3
109C	CMPY	↕	7	3
109E	LDY		6	3
109F	STY	Direct	6	3
10A3	CMPD	Indexed	7+	3+
10AC	CMPY	↕	7+	3+
10AE	LDY		6+	3+
10AF	STY	Indexed	6+	3+
10B3	CMPD	Extended	8	4
10BC	CMPY	↕	8	4
10BE	LDY		7	4
10BF	STY	Extended	7	4
10CE	LDS	Immed	4	4
10DE	LDS	Direct	6	3
10DF	STS	Direct	6	3
10EE	LDS	Indexed	6+	3+
10EF	STS	Indexed	6+	3+
10FE	LDS	Extended	7	4
10FF	STS	Extended	7	4
113F	SWI3	Implied	20	2
1183	CMPU	Immed	5	4
118C	CMPS	Immed	5	4
1193	CMPU	Direct	7	3
119C	CMPS	Direct	7	3
11A3	CMPU	Indexed	7+	3+
11AC	CMPS	Indexed	7+	3+
11B3	CMPU	Extended	8	4
11BC	CMPS	Extended	8	4

Note: All unused opcodes are both undefined and illegal.

Note for Use

Compatibility with NMOS MPU (HD6809)

The difference between HD6309 (CMOS) and HD6809 (NMOS) is shown in table 13.

Execution Sequence of CLR Instruction

Cycle-by-cycle flow of CLR instruction (direct, extended, indexed addressing mode) is shown below. In this sequence the contents of the memory location specified by the operand is read before writing 00 into it. Note that status flags, such as IRQ Flag, will be cleared by this extra data read operation when accessing the control/status register (sharing the same address between read and write) of peripheral devices.

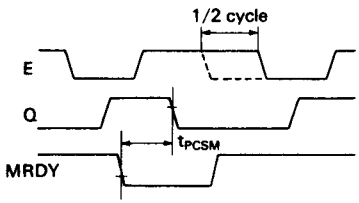
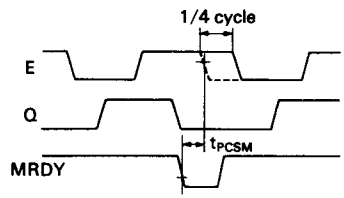
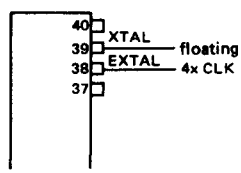
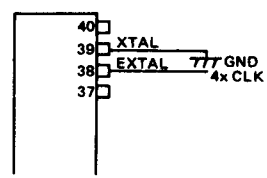
Example: CLR (Extended)

$8000	CLR	$A000
$A000	FCB	$80

Cycle #	Address	Data	R/$\overline{W}$	Description
1	8000	7F	1	Opcode Fetch
2	8001	A0	1	Operand Address, High Byte
3	8002	00	1	Operand Address, Low Byte
4	FFFF	*	1	$\overline{VMA}$ Cycle
5	A000	80	1	Read the Data
6	FFFF	*	1	$\overline{VMA}$ Cycle
7	A000	00	0	Store Fixed 00 into Specified Location

* The data bus has the data at that particular address.

Table 13. Difference between HD6309 and HD6809

Item		HD6309 (CMOS)	HD6809 (NMOS)
MRDY	Stretch Unit	integral multiples of half (1/2) bus cycles	integral multiples of quarter (1/4) bus cycles
			
	Stretch Time	5 μs max	10 μs max
$\overline{DMA/BREQ}$	Auto-refresh	None	Executed
External Clock Input		XTAL floating	XTAL grounded
			

Application Note for System Design

At the trailing edge of the address bus, the noise pulses may appeare on the output signals in HD6309.

Note the noise pulses and the following measures against them.

Noise Occurrence Condition: As shown in figure 20, the noise pulses which are 0.8 V or over may appear on E and Q clocks when the address bus changes from high to low.

If the address buses (A_0 -A_{15}, and R/$\overline{W}$) change from high to low, the transient current flows through the GND. The noise pulses are generated on the LSI's V_{SS} pins according to the current and to the impedance state of the GND wirings.

Figure 21 shows the noise voltage dependency on the each parameter.

Figure 23 shows the noise voltage dependency on the load capacitance of the address bus.

Note: The noise level should be carefully checked because it depends on the each parameter of actual application system.

Noise Reduction:

1. Control each parameter such as Cd, V_{CC}, Zg in figure 21, and the noise level is reduced to be allowable.
2. Insert a bypass capacitor between the V_{CC} and the GND of the HD6309.
3. Connect the CMOS buffer with noise margin to E and Q clocks.
4. Insert the damping registors to the address bus. That is effective for the noise level to reduce less than 0.8 V. The damping resistor is about 40-50 Ω on the higher byte of the address bus (A_{15} – A_8)and about 130-140 Ω on the lower byte of the address bus (A_7 – A_0), and R/$\overline{W}$ as shown in figure 22. Electrical characteristics do not change by inserting the damping resistors.

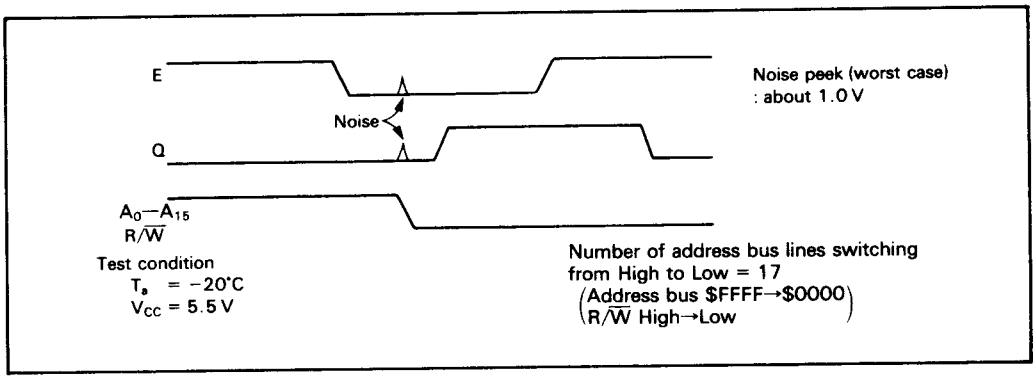


Figure 20. Noise at Address Bus Output Changing

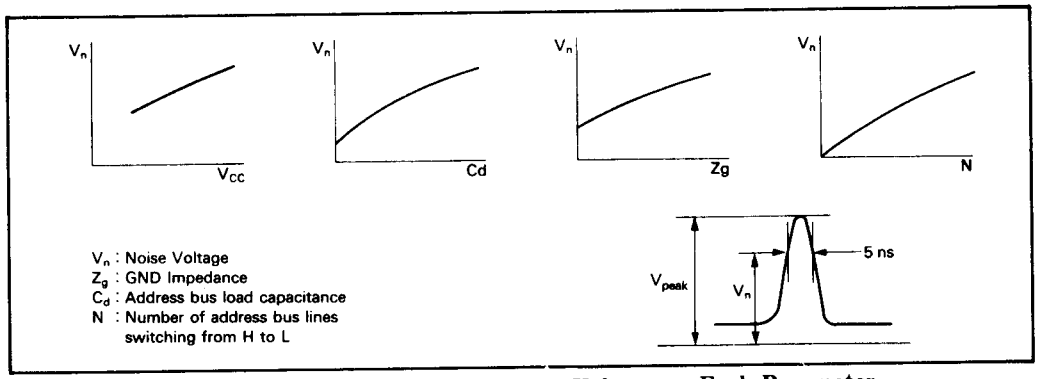


Figure 21. Dependency of the Noise Voltage on Each Parameter

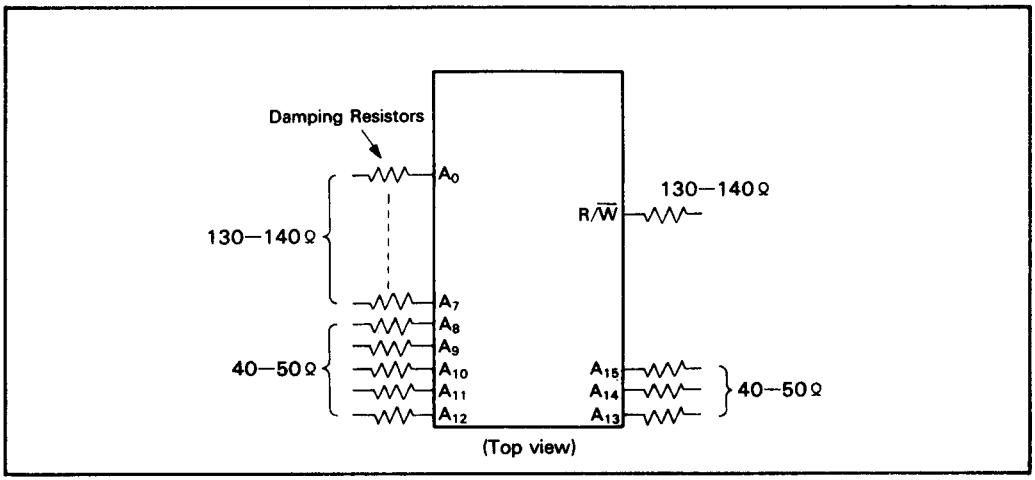


Figure 22. Connecting Damping Resistors to Address Bus

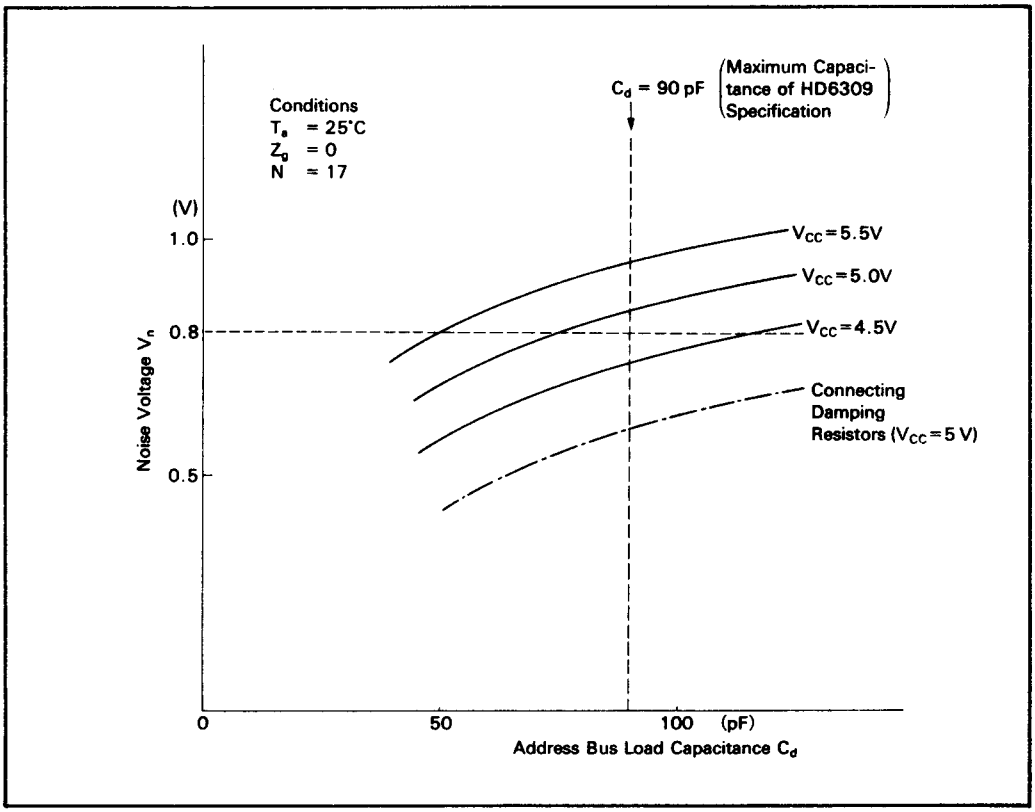


Figure 23. Dependency of the Noise Voltage on the Load Capacitance of the Address Bus

Absolute Maximum Ratings

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}[1]	−0.3 to +7.0	V
Input Voltage	V_{in}[1]	−0.3 to +7.0	V
Maximum Output Current	$\|I_o\|$[2]	5	mA
Maximum Total Output Current	$\|\Sigma I_o\|$[3]	100	mA
Operating Temperature	T_{opr}	−20 to +75	°C
Storage Temperature	T_{stg}	−55 to +150	°C

Notes: 1. ,With respect to V_{SS} (system GND)
2. Maximum output current is the maximum currents which can flow out from one output terminal and I/O common terminal (A_0-A_{15}, R/$\overline{W}$, D_0-D_7, BA, BS, Q, E).
3. Maximum total output current is the total sum of output currents which can flow out simultaneously from output terminals and I/O common terminals (A_0-A_{15}, R/$\overline{W}$, D_0-D_7, BA, BS, Q, E).
4. Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

2

Recommended Operating Conditions

Item		Symbol	Min	Typ	Max	Unit
Supply Voltage		V_{CC}[1]	4.5	5.0	5.5	V
Input Voltage	EXTAL	V_{IL}[1]	−0.3		0.6	V
	Other Inputs		−0.3		0.8	V
	$\overline{RES}$	V_{IH}[1]	V_{CC}−0.5		V_{CC}	V
	EXTAL		V_{CC}×0.7		V_{CC}	V
	Other Inputs		2.0		V_{CC}	V
Operating Temperature		T_{opr}	−20	25	75	°C

Note: 1. With respect to Vss (system GND)

Electrical Characteristics

DC Characteristics (V_{CC}=5.0 V ± 10%, V_{SS}=0 V, Ta=−20 to +75°C, unless otherwise noted.)

Item		Symbol	HD63B09 Min	Typ	Max	HD63C09 Min	Typ	Max	Unit	Test Condition
Input High Voltage	$\overline{RES}$	V_{IH}	V_{CC}−0.5		V_{CC}	V_{CC}−0.5		V_{CC}	V	
	EXTAL		V_{CC}×0.7		V_{CC}	V_{CC}×0.7		V_{CC}		
	Other Inputs		2.0		V_{CC}	2.0		V_{CC}		
Input Low Voltage	EXTAL	V_{IL}	−0.3		0.6	−0.3		0.6	V	
	Other Inputs		−0.3		0.8	−0.3		0.8		
Input Leakage Current	Except EXTAL, XTAL	I_{in}	−2.5		2.5	−2.5		2.5	μA	V_{in}=0 to V_{CC}, V_{CC}=max
Three State (Off State) Input Current	D_0-D_7	I_{TSI}	−10		10	−10		10	μA	V_{in}=0.4 to V_{CC}, V_{CC}=max
	A_0-A_{15}, R/$\overline{W}$		−10		10	−10		10		
Output High Voltage	D_0-D_7	V_{OH}	4.1			4.1			V	I_{LOAD}=−400μA
			V_{CC}−0.1			V_{CC}−0.1				I_{LOAD}≦−10μA
	A_0-A_{15}, R/$\overline{W}$, Q, E		4.1			4.1				I_{LOAD}=−400μA
			V_{CC}−0.1			V_{CC}−0.1				I_{LOAD}≦−10μA
	BA, BS		4.1			4.1				I_{LOAD}=−400μA
			V_{CC}−0.1			V_{CC}−0.1				I_{LOAD}≦−10μA
Output Low Voltage		V_{OL}			0.5			0.5	V	I_{LOAD}=2mA
Input Capacitance	D_0-D_7	C_{in}			15			15	pF	V_{in}=0V, T_a=25°C, f=1MHz
	Except D_0-D_7				10			10		
Output Capacitance	A_0-A_{15}, R/$\overline{W}$, BA, BS	C_{out}			12			12	pF	
Current Dissipation		I_{CC}			24			36	mA	Operating
					15			18		Sleeping

AC Characteristics (V_{CC}=5.0 V ± 10%, V_{SS}=0 V, Ta=−20 to +75°C, unless otherwise noted.)

Clock Timing

Item	Symbol	HD63B09 Min	HD63B09 Typ	HD63B09 Max	HD63C09 Min	HD63C09 Typ	HD63C09 Max	Unit	Test Condition
Frequency of Operation (Crystal External Input)	f_{XTAL}	2		8	2		12	MHz	Figs. 25, 26
Cycle Time	t_{cyc}	500		2000	333		2000	ns	
Total Up Time	t_{UT}	480			310			ns	
Processor Clock High	t_{PWEH}	220		5000	140		5000	ns	
Processor Clock Low	t_{PWEL}	210		1000	140		1000	ns	
E Rise and Fall Time	t_{Er}, t_{Ef}			20			20	ns	
E_{Low} to Q_{High} Time	t_{AVS}	100		140	70		100	ns	
Q Clock High	t_{PWQH}	220		1000	140		1000	ns	
Q Clock Low	t_{PWQL}	220		5000	140		5000	ns	
Q Rise and Fall Time	t_{Qr}, t_{Qf}			20			20	ns	
Q_{Low} to E_{Low} Time	t_{QE}	100			70			ns	

Bus Timing

Item		Symbol	HD63B09 Min	HD63B09 Typ	HD63B09 Max	HD63C09 Min	HD63C09 Typ	HD63C09 Max	Unit	Test Condition
Address Delay		t_{AD}			110			110	ns	Figs. 25, 26
Peripheral Read Access Time (t_{UT}-t_{AD}-t_{DSR}=t_{ACC})		t_{ACC}	330			160			ns	
Data Set Up Time (Read)		t_{DSR}	40			40			ns	
Input Data Hold Time		t_{DHR}	10			10			ns	
Address Hold Time	Ta=0 to +75°C	t_{AH}	20			20			ns	
	Ta=−20 to 0°C		10			10				
Data Delay Time (Write)		t_{DDW}			110			70	ns	
Output Hold Time	Ta=0 to +75°C	t_{DHW}	30			30			ns	
	Ta=−20 to 0°C		20			20				

2

Processor Control Timing

Item	Symbol	HD63B09 Min	HD63B09 Typ	HD63B09 Max	HD63C09 Min	HD63C09 Typ	HD63C09 Max	Unit	Test Condition
MRDY Set Up Time	t_{PCSM}	110			70			ns	Figs. 3 – 7
MRDY Set Up Time 2	t_{PCSM2}	240			160			ns	Figs. 11, 12
Interrupts Set Up Time	t_{PCS}	110			70			ns	
$\overline{HALT}$ Set Up Time	t_{PCSH}	110			70			ns	
$\overline{RES}$ Set Up Time	t_{PCSR}	110			110			ns	
$\overline{DMA/BREQ}$ Set Up Time	t_{PCSD}	110			70			ns	
Processor Control Rise and Fall Time	t_{PCr}, t_{PCf}			100			100	ns	
Crystal Oscillator Start Time	t_{RC}	20			20			ms	

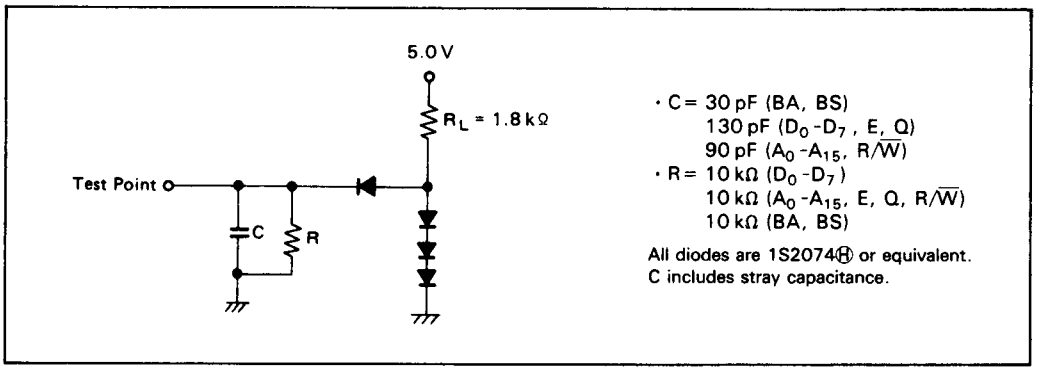


Figure 24. Bus Timing Test Load

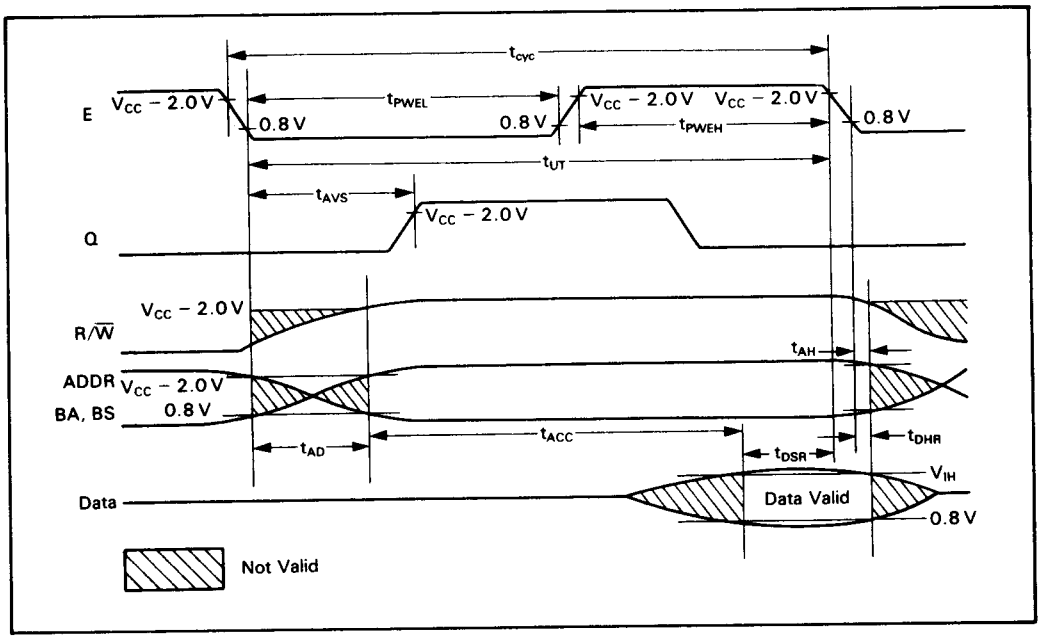


Figure 25. Read Data from Memory or Peripherals

2

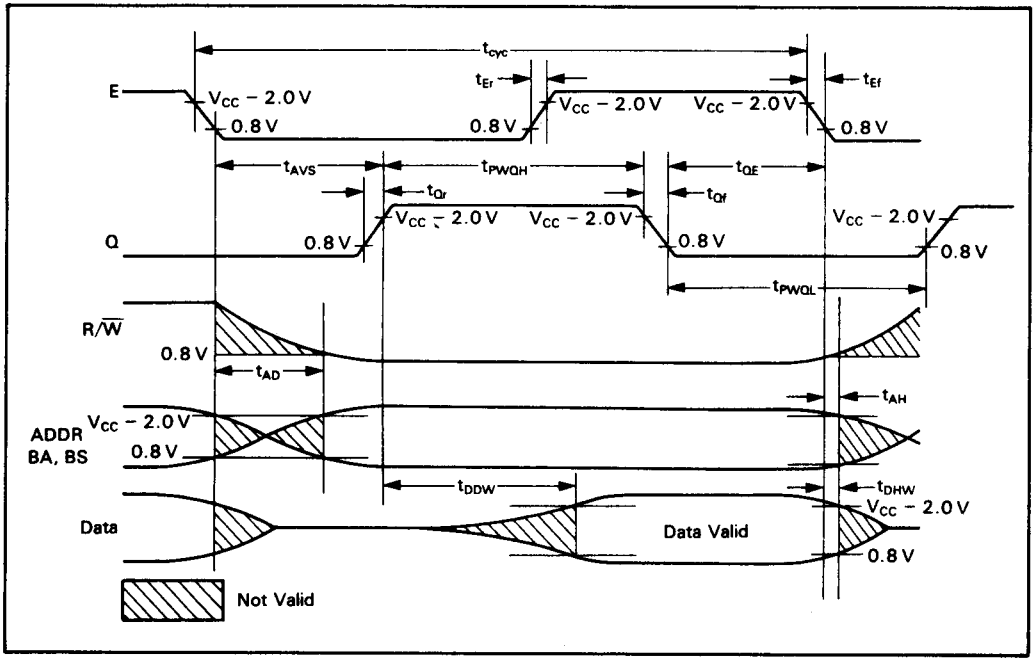


Figure 26. Write Data to Memory or Peripherals