

**8/16 BIT  
MULTI-CHIP  
MICROCOMPUTER  
DATA BOOK**



## TABLE OF CONTENTS

Quality and Reliability Commitment .....	3
Hitachi Microprocessor/Peripheral Cross Reference .....	4
New Hitachi Microprocessor Numbering System .....	5
HD6800, HD68A00, HD68B00 MPU (Micro Processing Unit) .....	6
HD6802 MPU (Microprocessor with Clock and RAM) .....	37
HD6809, HD68A09, HD68B09 MPU (Micro Processing Unit) .....	48
HD6821, HD68A21, HD68B21 PIA (Peripheral Interface Adapter) .....	77
HD6840, HD68A40, HD68B40 PTM (Programmable Timer Module) .....	97
HD6843S, HD68A43S FDC (Floppy Disk Controller) .....	110
HD6844, HD68A44 DMAC (Direct Memory Access Controller) .....	143
HD6845S, HD68A45S, HD68B45S CRTC (CRT Controller) .....	160
HD6846 COMBO (Combination ROM I/O Timer) .....	190
HD6850, HD68A50 ACIA (Asynchronous Communication Interface Adapter) .....	211
HD6852, HD68A52 SSDA (Synchronous Serial Data Adapter) .....	211
HD46508, HD46508-1 ADU (Analog Data Acquisition Unit)— <i>Preliminary</i> .....	234
HD68000 MPU (Micro Processing Unit)— <i>Preliminary</i> .....	253
APPENDIX .....	
Package Information .....	296
HMCS6800 Family Instructions .....	298
Memories .....	303
Linear IC & Interface Circuits .....	306
TTL HD74/HD74S/HD74LS Series .....	308



# An Unprecedented Commitment to Quality and Reliability . . .

As quality and reliability become increasingly important concerns, Hitachi continues to improve its efforts to provide the best possible product. The experience gained in shipping millions of microprocessors and peripheral LSIs for critical and demanding automotive and industrial applications is reflected in every product we sell. Each unit shipped receives 100% dynamic high-temperature burn-in, a quality assurance effort unparalleled in the semiconductor industry, and another reason why Hitachi is the Symbol of Semiconductor Quality, Worldwide.

## **QUALITY ASSURANCE FLOW FOR ASSEMBLY AND TEST (all microprocessor and microcomputer products):**

PROCESS	INSPECTION LEVEL	QC CRITERIA	REMARKS
1 Dicing	—	—	—
2 Chip Visual	100%	Visual	100x
3 Lot Acceptance	AQL = 0.25%*	Visual	100x
4 Die Attachment	—	—	Au-Si
5 Patrol Inspection	Once/Day/Machine	Visual	—
6 Wire Bonding	—	—	AI Ultrasonic
7 Patrol Inspection	Once/Day/Machine	Visual	
	Once/Week/Machine	Bond Dimension Bond Strength	
8 Visual Inspection	100%	Visual	20x
9 Lot Acceptance	AQL = 0.25%*	Visual	20x
10 Seal	—	—	A-Sn Alloy
11 Temperature Cycle	100%	—	-55 °C -25 °C -150 °C 10 Cycles
12 Hermeticity	100%	Fine and Gross	Hermetic Packages Only
13 Plating	—	—	Tin (Sn)
14 Lead Trim	—	—	—
15 Visual Inspection	100%	Visual	
16 Lot Acceptance	AQL = 0.25%	Visual	
17 Burn-in	100%	—	Dynamic Ta = 125 °C
18 Electrical Test	100%	DC, AC, Functional	Ta = 70 °C
19 Marking	—	—	
20 Electrical	100%	DC	
21 Visual Inspection	100%	External Visual	
22 Lot Acceptance	AQL = 0.25%*	Electrical	
	AQL = 0.65%	External Visual	

\*Combined DC, AC and functional.

# HITACHI MICROPROCESSOR/PERIPHERAL CROSS REFERENCE

Hitachi is in the process of converting many microprocessor part numbers to "industry standard" generic part numbers. A complete list showing both the "old" and "new" part numbers is shown in figure 1. The use of industry standard part numbers will greatly simplify the interface between Hitachi and our customers.

Beginning JULY 1, 1981, all orders should be entered using the "new" part numbers only.

Note that during the conversion process, product shipped by Hitachi will be marked 1 of 2 ways (see figure 2).

1) marked with the "old" Hitachi part number...

or

2) marked with a dual number ("old" and "new")

At the completion of the conversion (approximately JANUARY 1, 1982) all product will be shipped with the dual marking (2 above).

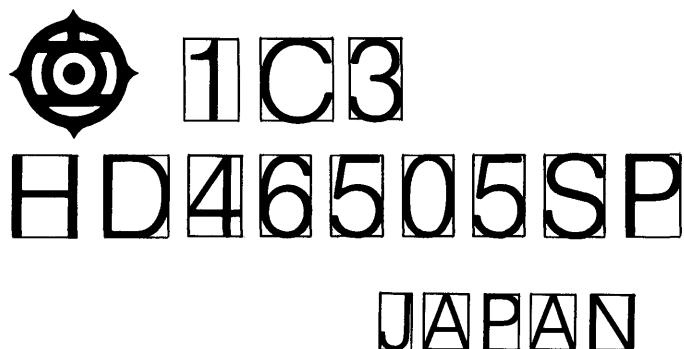
If this conversion plan poses problems, or you have any questions, please contact Hitachi Microprocessor Marketing.

Description	"old" HITACHI number	"new" HITACHI number	MOTOROLA number
16/32 bit microprocessing unit, 8 mhz .....	-----	HD68000-8	MC68000L
16/32 bit microprocessing unit, 6 mhz .....	-----	HD68000-6	MC68000L6
16/32 bit microprocessing unit, 4 mhz .....	-----	HD68000-4	MC68000L4
8/16 bit microprocessing unit, 1mhz .....	HD6809P	HD6809P	MC6809P
8/16 bit microprocessing unit, 1.5mhz .....	HD68A09P	HD68A09P	MC68A09P
8/16 bit microprocessing unit, 2mhz .....	HD68B09P	HD68B09P	MC68B09P
8 bit microprocessing unit, 1mhz .....	HD46800DP	HD6800P	MC6800P
8 bit microprocessing unit, 1.5mhz .....	HD468A00P	HD68A00P	MC68A00P
8 bit microprocessing unit, 2mhz .....	HD468B00P	HD68B00P	MC68B00P
8 bit microprocessing unit, 1mhz .....	HD46802SP	HD6802SP	MC6802P
with clock and 128 bytes RAM			
8 bit microprocessing unit, 1mhz .....	-----	HD6802WP	-----
with clock and 256 bytes RAM			
128 x 8 static RAM, 450ns access time .....	HM46810P	HD6810P	MC6810P
128 x 8 static RAM, 360ns access time .....	HM468A10P	HD68A10P	MC68A10P
Peripheral interface adapter, 1mhz .....	HD46821P	HD6821P	MC6821P
Peripheral interface adapter, 1.5mhz .....	HD468A21P	HD68A21P	MC68A21P
Peripheral interface adapter, 2mhz .....	HD468B21P	HD68B21P	MC68B21P
Programmable timer module, 1mhz .....	-----	HD6840P	MC6840P
Programmable timer module, 1.5mhz .....	-----	HD68A40P	MC68A40P
Programmable timer module, 2mhz .....	-----	HD68B40P	MC68B40P
Floppy disk controller, 1mhz .....	HD46503SP	HD6843SP	MC6843P
Floppy disk controller, 1.5mhz .....	HD46503SP-1	HD68A43SP	MC68A43P
8 bit DMA controller, 1mhz .....	HD46504RP	HD6844P	MC6844P
8 bit DMA controller, 1.5mhz .....	HD46504RP-1	HD68A44P	MC68A44P
8 bit DMA controller, 2mhz .....	HD46504RP-2	HD68B44P	MC68B44P
CRT controller, 1mhz .....	HD46505RP	HD6845RP	MC6845P
CRT controller, 1.5mhz .....	HD46505RP-1	HD68A45RP	MC68A45P
CRT controller, 2mhz .....	HD46505RP-2	HD68B45RP	MC68B45P
CRT controller (enhanced), 1 mhz .....	HD46505SP	HD6845SP	-----
CRT controller (enhanced), 1.5mhz .....	HD46505SP-1	HD68A45SP	-----
CRT controller (enhanced), 2mhz .....	HD46505SP-2	HD68B45SP	-----
ROM, I/O, Timer combo, 1mhz .....	-----	HD6846P	MC6846P
Asynchronous comm interface, 1mhz .....	HD46850P	HD6850P	MC6850P
Asynchronous comm interface, 1.5mhz .....	HD468A50P	HD68A50P	MC68A50P
Synchronous comm interface, 1mhz .....	HD46852P	HD6852P	MC6852P
Synchronous comm interface, 1.5mhz .....	HD468A52P	HD68A52P	MC68A52P
Analog data acquisition unit, 1mhz .....	HD46508P	HD46508P	-----
Analog data acquisition unit, 1.5mhz .....	HD46508P-1	HD46508P-1	-----
Analog data acquisition unit, 1mhz (enhanced)....	HD46508PA	HD46508PA	-----
Analog data acquisition unit, 1.5mhz (enhanced) ...	HD46508PA-1	HD46508PA-1	-----

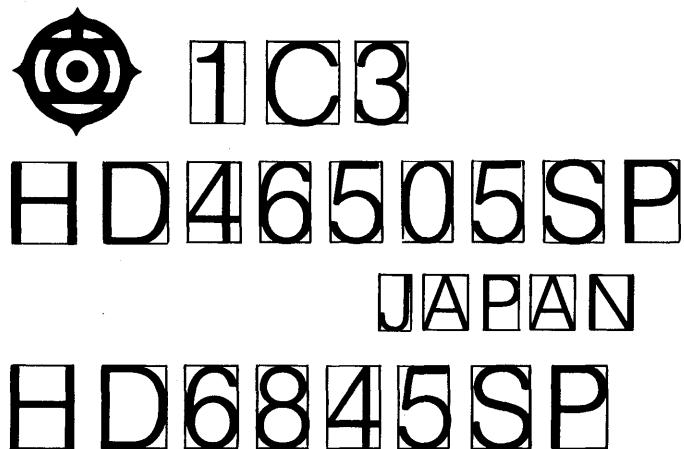
Figure 1. Hitachi Microprocessor/Peripheral Cross Reference

## **NEW HITACHI MICROPROCESSOR NUMBERING SYSTEM**

(a) Present marking



(b) New marking



**Figure 2.**

# HD6800, HD68A00, HD68B00

## MPU (Micro Processing Unit)

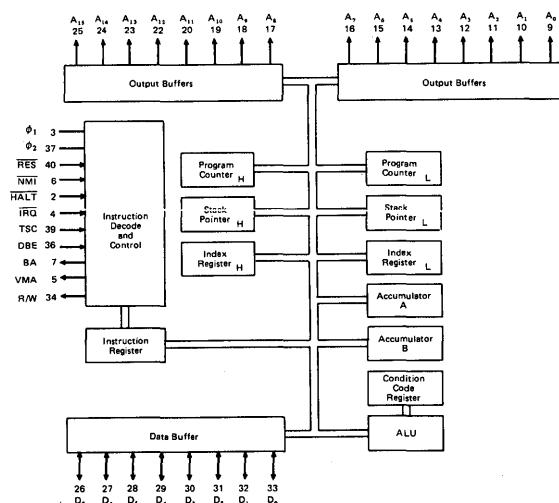
The HD6800 is a monolithic 8-bit microprocessor forming the central control function for Hitachi's HMCS6800 family. Compatible with TTL, the HD6800 as with all HMCS6800 system parts, requires only one 5V power supply, and no external TTL devices for bus interface. The HD68A00 and HD68B00 are high speed versions.

The HD6800 is capable of addressing 65K bytes of memory with its 16-bit address lines. The 8-bit data bus is bi-directional as well as 3-state, making direct memory addressing and multiprocessing applications realizable.

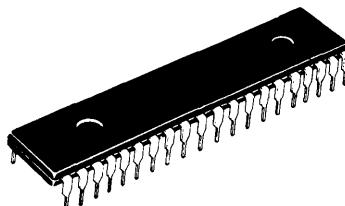
### ■ FEATURES

- Versatile 72 Instruction – Variable Length (1~3 Byte)
- Seven Addressing Modes – Direct, Relative, Immediate, Indexed, Extended, Implied and Accumulator
- Variable Length Stack
- Vectored Restart
- Maskable Interrupt
- Separate Non-Maskable Interrupt – Internal Registers Saved in Stack
- Six Internal Registers – Two Accumulators, Index Register, Program Counter, Stack Pointer and Condition Code Register
- Direct Memory Accessing (DMA) and Multiple Processor Capability
- Clock Rates as High as 2.0 MHz (HD6800 ... 1 MHz, HD68A00 ... 1.5 MHz, HD68B00 ... 2.0 MHz)
- Halt and Single Instruction Execution Capability
- Compatible with MC6800, MC68A00 and MC68B00

### ■ BLOCK DIAGRAM

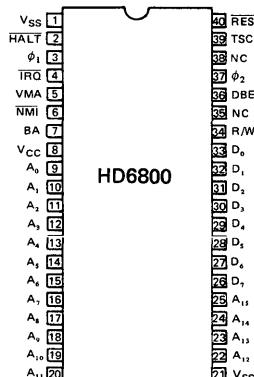


HD6800P, HD68A00P, HD68B00P



(DP-40)

### ■ PIN ARRANGEMENT



(Top View)

**HD6800, HD68A00, HD68B00**

**■ ABSOLUTE MAXIMUM RATINGS**

Item	Symbol	Value	Unit
Supply Voltage	$V_{CC}^*$	-0.3 ~ +7.0	V
Input Voltage	$V_{in}^*$	-0.3 ~ +7.0	V
Operating Temperature	$T_{opr}$	-20 ~ +75	°C
Storage Temperature	$T_{stg}$	-55 ~ +150	°C

\* With respect to  $V_{SS}$  (SYSTEM GND)

(NOTE) Permanent LSI damage may occur if maximum rating are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

**■ RECOMMENDED OPERATING CONDITION**

Item	Symbol	min	typ	max	Unit
Supply Voltage	$V_{CC}^*$	4.75	5	5.25	V
Input Voltage	$V_{IL}^*$	-0.3	—	0.8	V
	$V_{IH}^*$	2.0	—	$V_{CC}$	V
Operating Temperature	$T_{opr}$	-20	25	75	°C

\* With respect to  $V_{SS}$  (SYSTEM GND)

**■ ELECTRICAL CHARACTERISTICS**

● DC CHARACTERISTICS ( $V_{CC} = 5V \pm 5\%$ ,  $V_{SS} = 0V$ ,  $T_a = -20 \sim +75^\circ C$ , unless otherwise noted.)

Item	Symbol	Test Condition	min	typ*	max	Unit
Input "High" Voltage	$V_{IH}$	$V_{SS} + 2.0$	—	$V_{CC}$	—	V
Input "Low" Voltage	$V_{IL}$	$V_{SS} - 0.3$	—	$V_{SS} + 0.8$	—	V
Clock Input "High" Voltage	$\phi_1, \phi_2$	$V_{CC} - 0.6$	—	$V_{CC} + 0.3$	—	V
Clock Input "Low" Voltage	$V_{ILC}$	$V_{SS} - 0.3$	—	$V_{SS} + 0.4$	—	V
	$D_0 \sim D_7$	$I_{OH} = 205\mu A$	$V_{SS} + 2.4$	—	—	V
Output "High" Voltage	$A_0 \sim A_{15}, R/W$		$V_{SS} + 2.4$	—	—	V
	VMA		$V_{SS} + 2.4$	—	—	V
	BA		$V_{SS} + 2.4$	—	—	V
Output "Low" Voltage	$V_{OL}$	$I_{OL} = 1.6mA$	—	—	$V_{SS} + 0.4$	V
Input Leakage Current	Logic***	$V_{in} = 0 \sim 5.25V$ , All other pins are connected to GND	—	1.0	2.5	$\mu A$
	$\phi_1, \phi_2$		—	—	100	$\mu A$
Three-State (Off-state) Input Current	$D_0 \sim D_7$	$V_{in} = 0.4 \sim 2.4V$	—	2.0	10	$\mu A$
	$A_0 \sim A_{15}, R/W$		—	—	100	$\mu A$
Power Dissipation	$P_D$		—	0.5	1.0	W
Input Capacitance	Logic***	$V_{in} = 0V, T_a = 25^\circ C$ , $f = 1 MHz$	—	6.5	10	pF
	$D_0 \sim D_7$		—	10	12.5	pF
	$\phi_1$		—	25	35	pF
	$\phi_2$		—	45	70	pF
Output Capacitance	$A_0 \sim A_{15}, R/W$	$V_{in} = 0V, T_a = 25^\circ C$ , $f = 1 MHz$	—	—	12	pF

\*  $T_a = 25^\circ C$ ,  $V_{CC} = 5V$

\*\* All inputs except  $\phi_1$  and  $\phi_2$

\*\*\* All inputs except  $\phi_1$ ,  $\phi_2$  and  $D_0 \sim D_7$

## ● AC CHARACTERISTICS

### 1. TIMING CHARACTERISTICS OF CLOCK PULSE $\phi_1$ and $\phi_2$

Item	Symbol	Test Condition	HD6800			HD68A00			HD68B00			Unit	
			min	typ	max	min	typ	max	min	typ	max		
Frequency of Operation	f		0.1	—	1.0	0.1	—	1.5	0.1	—	2.0	MHz	
Cycle Time	$t_{cyc}$	Fig. 10	1,000	—	10	0.666	—	10	0.500	—	10	$\mu s$	
Clock Pulse Width	$\phi_1, \phi_2$	$PW_{CH1}, PW_{CH2}$	Fig. 10	400	—	4,500	230	—	4,500	180	—	4,500	ns
Rise and Fall Times	$\phi_1, \phi_2$	$t_r, t_f$	Fig. 10	—	—	100	—	—	100	—	—	100	ns
Delay Time (Clock Internal)	$t_d$	Fig. 10	—	—	4,500	—	—	4,500	—	—	4,500	ns	
Clock "High" Level Time	$t_{UT}$	Fig. 10	900	—	—	600	—	—	440	—	—	ns	

### 2. READ/WRITE CHARACTERISTICS

Item	Symbol	Test Condition	HD6800			HD68A00			HD68B00			Unit	
			min	typ	max	min	typ	max	min	typ	max		
Address Delay Time	$C=90pF$	$t_{AD1}$	Fig. 11, Fig. 12	—	—	270	—	—	180	—	—	150	ns
	$C=30pF$	$t_{AD2}$	Fig. 11, Fig. 12	—	—	250	—	—	165	—	—	135	ns
Data Setup Time (Read)	$t_{DSR}$	Fig. 11	100	—	—	60	—	—	40	—	—	—	ns
Peripheral Read Access Time $t_{acc} = t_{UT} - (t_{AD} + t_{DSR})$	$t_{acc}$	Fig. 11	—	—	530	—	—	360	—	—	250	ns	
Input Data Hold Time	$t_H$	Fig. 11	10	—	—	10	—	—	10	—	—	—	ns
Output Data Hold Time	$t_H$	Fig. 12	20	—	—	20	—	—	20	—	—	—	ns
Address Hold Time (Address, R/W, VMA)	$t_{AH}$	Fig. 11, Fig. 12	10	—	—	10	—	—	10	—	—	—	ns
Enable "High" Time for DBE Input	$t_{EH}$	Fig. 12	450	—	—	280	—	—	220	—	—	—	ns
Data Delay Time (Write)	$t_{DDW}$	Fig. 12	—	—	225	—	—	200	—	—	160	ns	
Data Bus Enable Down Time (During $\phi_1$ Up Time)	$t_{DBE}$	Fig. 12	150	—	—	120	—	—	75	—	—	—	ns
Data Bus Enable Delay Time	$t_{DBED}$	Fig. 12	300	—	—	250	—	—	180	—	—	—	ns
Data Bus Enable Rise and Fall Times	$t_{DBEr}$ $t_{DBEf}$	Fig. 12	—	—	25	—	—	25	—	—	25	ns	
Processor Control Setup Time	$t_{PCS}$		200	—	—	140	—	—	110	—	—	—	ns
Processor Control Rise and Fall Times	$t_{PCr}$ $t_{PCf}$		—	—	100	—	—	100	—	—	100	ns	
Bus Available Delay Time (BA)	$t_{BA}$		—	—	250	—	—	165	—	—	135	ns	
Three-State Delay Time	$t_{TSD}$		—	—	270	—	—	270	—	—	220	ns	

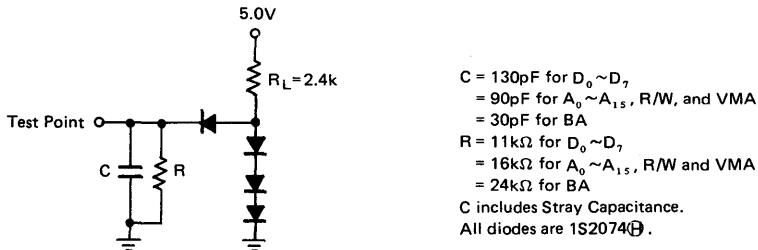


Figure 1 Bus Timing Test Load

---

**HD6800, HD68A00, HD68B00**

---

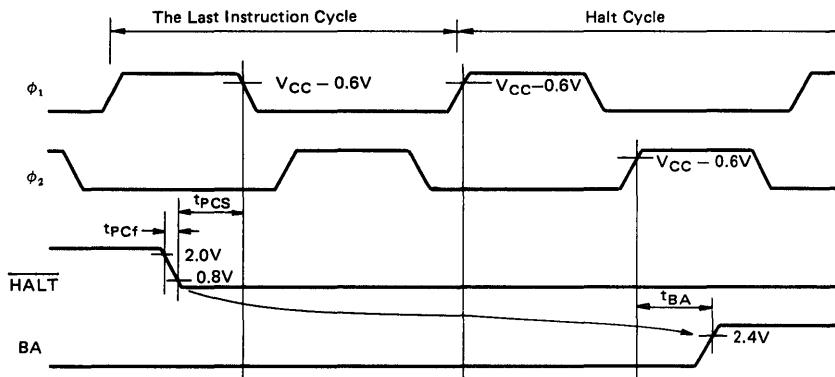


Figure 2 Timing of HALT and BA

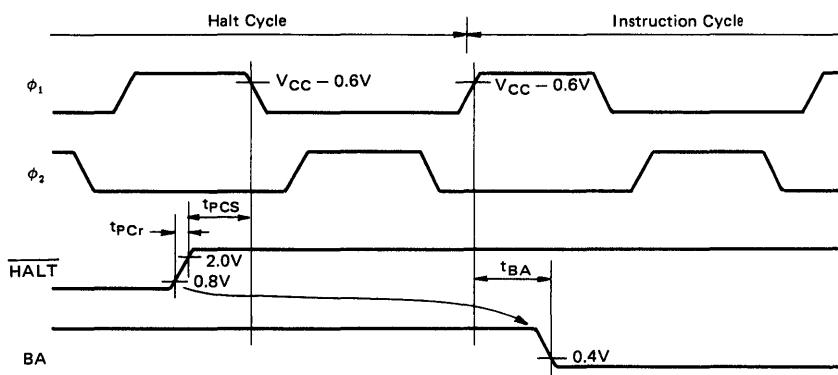


Figure 3 Timing of HALT and BA

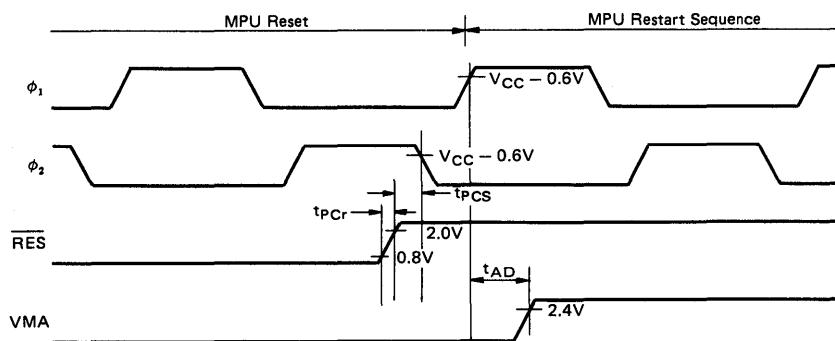
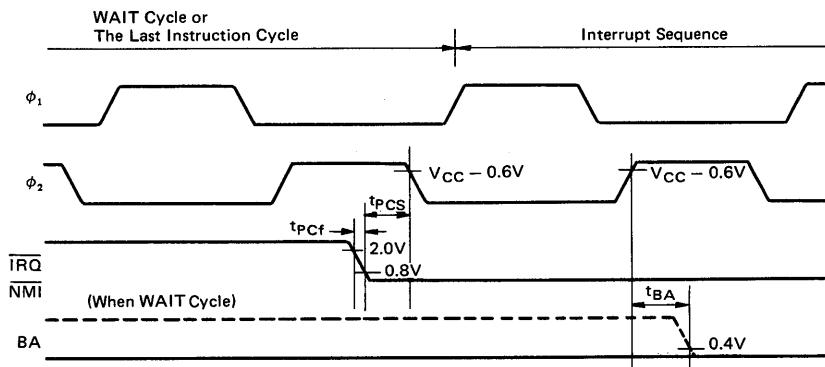
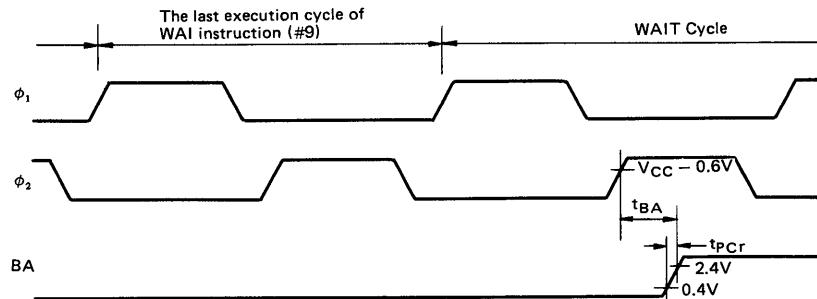


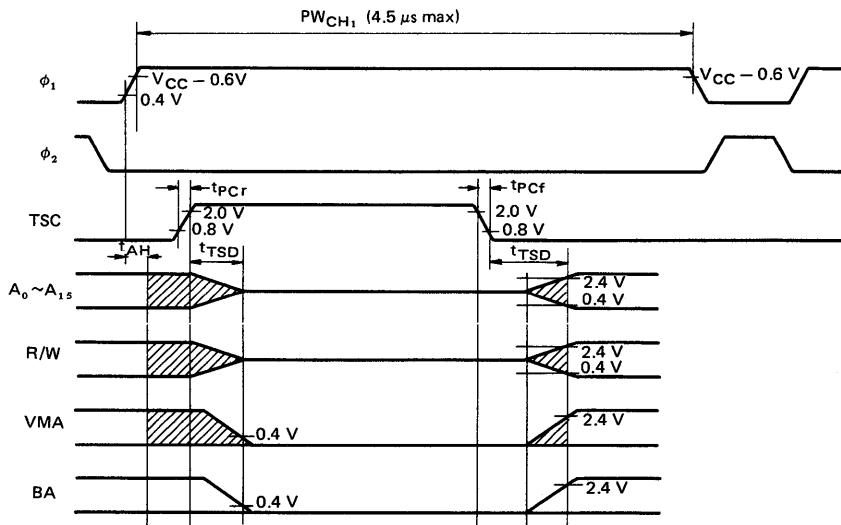
Figure 4 RES and MPU Restart Sequence



**Figure 5 IRQ and NMI Interrupt Timing**



**Figure 6 WAI Instruction and BA Timing**



**Figure 7 TSC Input and MPU Output**

## ■ MPU REGISTERS

The MPU provides several registers in Fig. 8, which is available for use by the programmer.

Each register is described below.

### ● Program Counter (PC)

The program counter is a two byte (16-bit) register that points to the current program address.

### ● Stack Pointer (SP)

The stack pointer is a two byte register that contains the address of the next available location in an external push-down/pop-up stack. This stack is normally a random access Read/Write memory that may have any location (address) that is convenient. In those applications that require storage of information in the stack when power is lost, the stack must be non-volatile.

### ● Index Register (IX)

The index register is a two byte register that is used to store data or a sixteen bit memory address for the Indexed mode of memory addressing.

### ● Accumulators (ACCA, ACCB)

The MPU contains two 8-bit accumulators that are used to hold operands and results from an arithmetic logic unit (ALU).

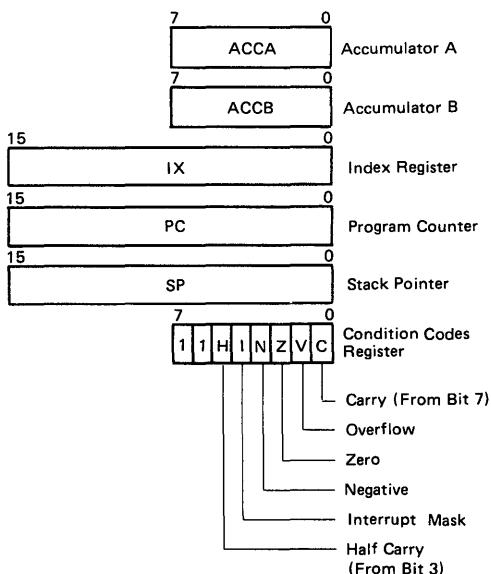


Figure 8 Programming Model of the Microprocessing Unit

### ● Condition Code Register (CCR)

The condition code register indicates the results of an Arithmetic Logic Unit operation: Negative (N), Zero (Z), Overflow (V), Carry from bit 7 (C), and half carry from bit 3(H). These bits of the Condition Code Register are used as testable conditions for the conditional branch instructions. Bit 4 is the interrupt mask bit (I). The unused bits of the Condition Code Register (b6 and b7) are "1". The detail block diagram of the microprossing unit is shown in Fig. 9.

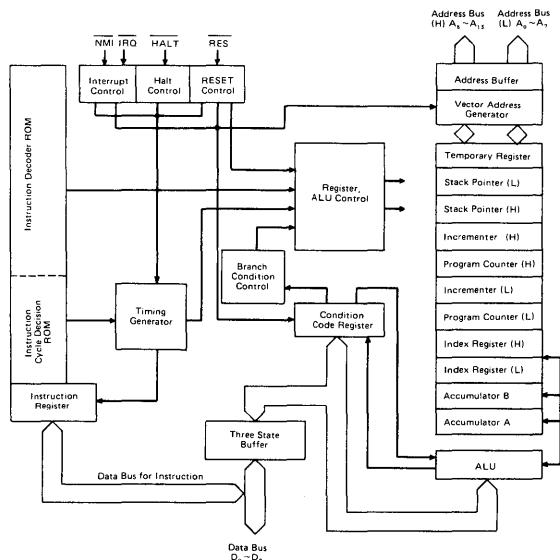


Figure 9 Internal Block Diagram of MPU

## ■ MPU SIGNAL DESCRIPTION

Proper operations of the MPU requires that certain control and timing signals (Fig. 9) be provided to accomplish specific functions. The functions of pins are explained in this section.

### ● Clock ( $\phi_1, \phi_2$ )

Two pins are used to provide the clock signals. A two-phase non-overlapping clock is provided as shown in Fig. 10.

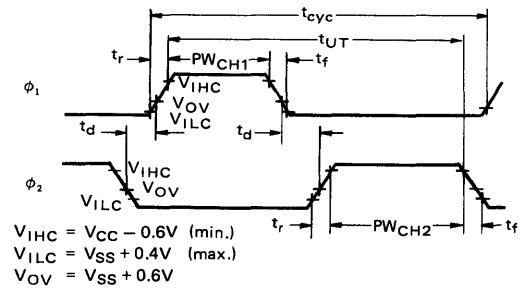


Figure 10 Clock Timing Waveform

### ● Address Bus ( $A_0 \sim A_{15}$ )

Sixteen pins are used for the address bus. The outputs are three-state bus drivers capable of driving one standard TTL load and 90pF. When the output is turned off, it is essentially an open circuit. This permits the MPU to be used in DMA applications. Putting TSC in its high state forces the Address bus to go into the three-state mode.

### ● Data Bus ( $D_0 \sim D_7$ )

Eight pins are used for the data bus. It is bidirectional, transferring data to and from the memory and peripheral devices. It also has three-state output buffers capable of driving one standard TTL load and 130pF. Data Bus is placed in the three-state mode when DBE is "Low."

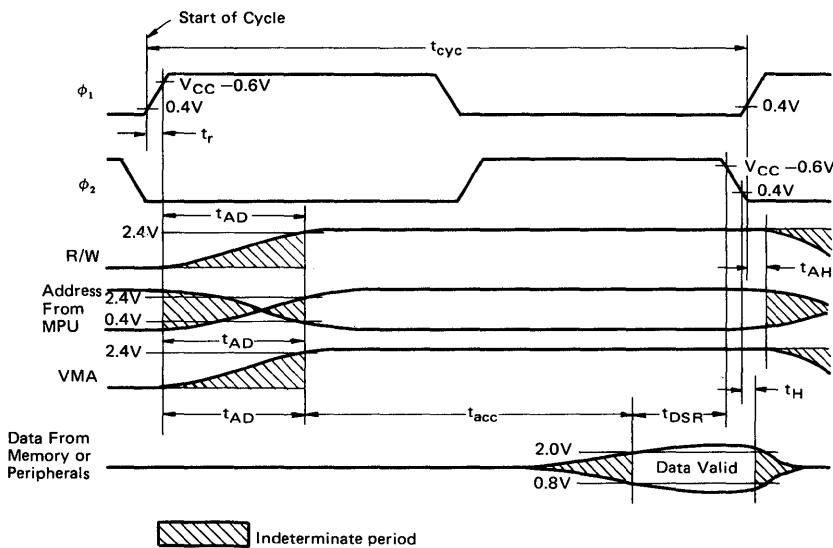


Figure 11 Read from Memory or Peripherals

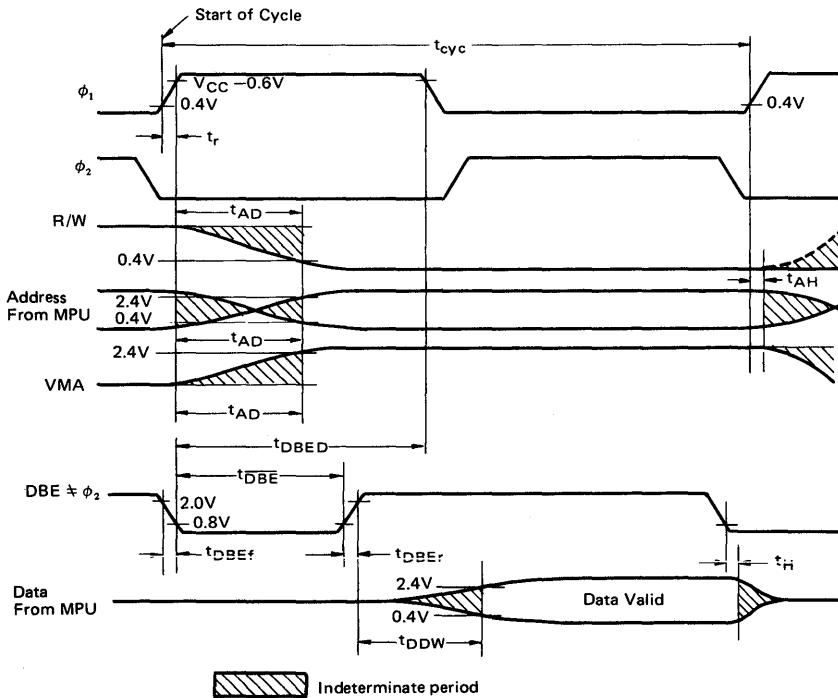


Figure 12 Write to Memory of Peripherals

**• Data Bus Enable (DBE)**

This input is the three-state control signal for the MPU data bus and will enable the bus drivers when in the "High" state; will make the bus driver off when in the "Low" state. This input is TTL compatible; however in normal operation, it would be driven by  $\phi_2$  clock. During an MPU read cycle, the data bus drivers will be disabled internally. When it is desired that another device control the data bus such as in Direct Memory Access (DMA) applications, DBE should be held "Low."

If additional data setup or hold time is required on an MPU write, the DBE down time can be decreased as shown in Fig. 13 ( $DBE \neq \phi_2$ ). The minimum down time for DBE is  $t_{DBE}$  as shown and must occur within  $\phi_1$  up time. As for the characteristic values in Fig. 12, refer to the table of electrical characteristics.

**• Bus Available (BA)**

The BA signal will normally be in the "Low" state. When activated, it will go to the "High" state indicating that the microprocessor has stopped and that the address bus is available. This will occur if the HALT line is in the "Low" state or the processor is in the WAIT state as a result of the execution of a WAIT instruction. At such time, all three-state output drivers will go to their off state and other outputs to their normally inactive level. The processor is removed from the WAIT state by the occurrence of a maskable (mask bit I = 0) or nonmaskable interrupt. This output is capable of driving one standard TTL load and 30pF. If TSC is in the "High" state, Bus Available will be "Low".

**• Read/Write (R/W)**

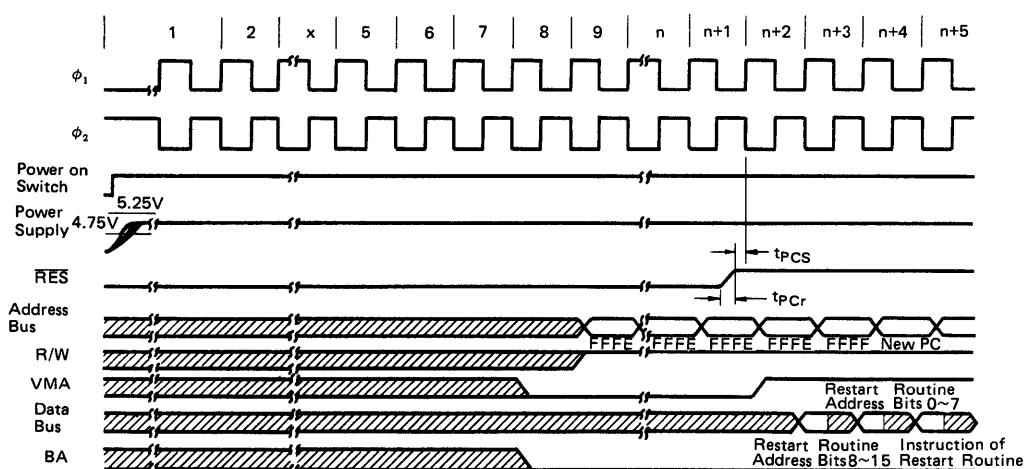
This TTL compatible output signals the peripherals and memory devices whether the MPU is in a Read ("High") or

Write ("Low") state. The normal standby state of this signal is Read ("High"). Three-State Control going "High" will turn R/W to the off (high impedance) state. Also, when the processor is halted, it will be in the off state. This output is capable of driving one standard TTL load and 90pF.

**• Reset (RES)**

The RES input is used to reset and start the MPU from a power down condition resulting from a power failure or initial start-up of the processor. This input can also be used to reinitialize the machine at any time after start-up.

If a "High" level is detected in this input, this will signal the MPU to begin the reset sequence. During the reset sequence, the contents of the last two locations (FFFF, FFFF) in memory will be loaded into the Program Counter to point to the beginning of the reset routine. During the reset routine, the interrupt mask bit is set and must be cleared under program control before the MPU can be interrupted by IRQ. While RES is "Low" (assuming a minimum of 8 clock cycles have occurred) the MPU output signals will be in the following states; VMA = "Low", BA = "Low", Data Bus = high impedance, R/W = "High" (read state), and the Address Bus will contain the reset address FFFE. Fig. 13 illustrates a power up sequence using the Reset control line. After the power supply reaches 4.75V, a minimum of eight clock cycles are required for the processor to stabilize in preparation for restarting. During these eight cycles, VMA will be in an indeterminate state so any devices that are enabled by VMA which could accept a false write during this time (such as a battery-backed RAM) must be disabled until VMA is forced "Low" after eight cycles. RES can go "High" asynchronously with the system clock any time after the eighth cycle.



= Indeterminate period

Figure 13 RES Timing

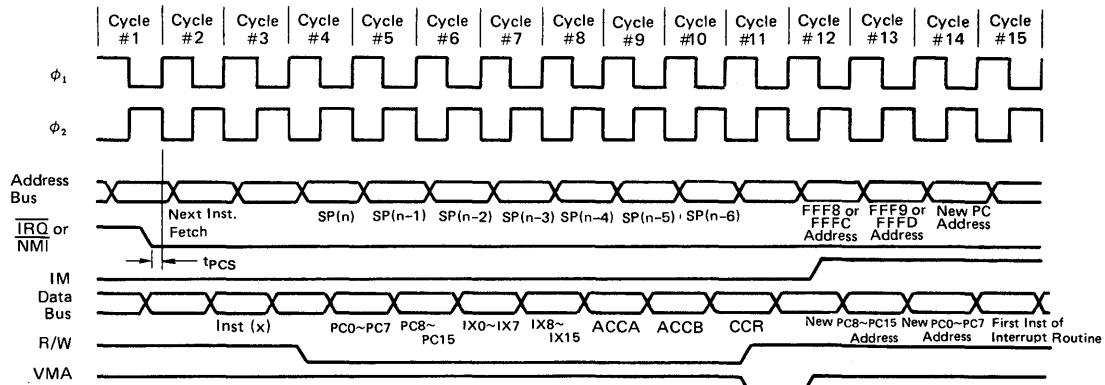
The Reset control line may also be used to reinitialize the MPU system at any time during its operation. This is accomplished by pulsing RES "Low" for the duration of a minimum of three complete  $\phi_2$  cycles. The RES pulse can be completely asynchronous with the MPU system clock and will be recognized during  $\phi_2$  if setup time  $t_{PCS}$  is met.

● **Interrupt Request (IRQ)**

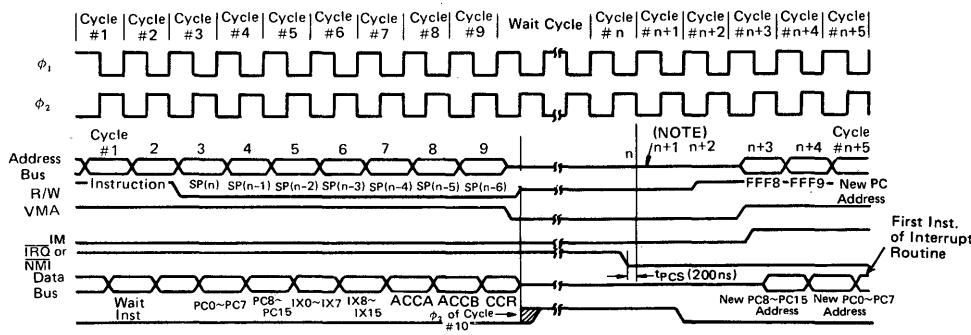
This level sensitive input requests that an interrupt sequence be generated within the machine. The processor will wait until it completes the current instruction that is being executed before it recognizes the request. If the interrupt mask bit in the Condition Code Register is not set, the machine will begin an interrupt sequence. The Index Register, Program Counter, Accumulators, and Condition Code Register are stored away on the stack.

Next the MPU will respond to the interrupt request by setting the interrupt mask bit "1" so that no further interrupts may occur. At the end of the cycle, a 16-bit address will be loaded that points to a vectoring address which is located in memory locations FFF8 and FFF9. An address loaded at these locations causes the MPU to branch to an interrupt routine in memory. Interrupt timing is shown in Fig. 14.

The HALT line must be in the "High" state for interrupts to be serviced. Interrupts will be latched internally while HALT is "Low". The IRQ has a high impedance pullup device internal to the chip; however a  $3k\Omega$  external resistor to  $V_{CC}$  should be used for wire-OR and optimum control of interrupts.



**Figure 14 Interrupt Timing**



(NOTE) Midrange waveform indicates high impedance state.

**Figure 15 WAI Instruction Timing**

● **Non-Maskable Interrupt (NMI) and Wait for Interrupt (WAI)**

The MPU is capable of handling two types of interrupts: maskable (IRQ) as described earlier, and non-maskable (NMI). IRQ is maskable by the interrupt mask in the Condition Code Register while NMI is not maskable. The handling of these interrupts by the MPU is the same except that each has its own vector address. The behavior of the MPU when interrupted is shown in Fig. 14 which details the MPU response to an interrupt while the MPU is executing the control program. The interrupt shown could be either IRQ or NMI and can be asynchronous with respect to  $\phi_2$ . The interrupt is shown going "Low" at time  $t_{PCS}$  in cycle #1 which precedes the first cycle of an instruction (OP code fetch). This instruction is not executed but instead the Program Counter (PC), Index Register (IX), Accumulators (ACCX), and the Condition Code Register (CCR) are pushed onto the stack.

The Interrupt Mask bit is set to prevent further interrupts. The address of the interrupt service routine is then fetched from FFFC, FFED for an NMI interrupt and from FFF8, FFF9 for an IRQ interrupt. Upon completion of the interrupt service routine, the execution of RTI will pull the PC, IX, ACCX, and CCR off of the stack; the Interrupt Mask bit is restored to its condition prior to interrupts. Fig. 15 is a similar interrupt sequence, except in this case, a WAIT instruction has been executed in preparation for the interrupt. This technique speeds up the MPU's response to the interrupt because the stacking of

the PC, IX, ACCX, and the CCR is already done.

While the MPU is waiting for the interrupt, Bus Available will go "High" indicating the following states of the control lines: VMA is "Low", and the Address Bus, R/W and Data Bus are all in the high impedance state. After the interrupt occurs, it is serviced as previously described.

**Table 1 Memory Map for Interrupt Vectors**

Vector		Description
MS	LS	
FFFE	FFFF	Restart
FFF0	FFF0	Non-maskable Interrupt
FFF1	FFF1	Software Interrupt
FFF8	FFF9	Interrupt Request

Refer to Figure 18 for program flow for Interrupts.

#### • Three State Control (TSC)

When the Three State Control (TSC) line is "High" level, the Address Bus and the R/W line are placed in a high impedance State. VMA and BA are forced "Low" when TSC = "High" to prevent false reads or writes on any device enabled by VMA. It is necessary to delay program execution while TSC is held "High". This is done by insuring that no transitions of  $\phi_1$  (or  $\phi_2$ ) occur during this period. (Logic levels of the clocks are irrelevant so long as they do not change.)

Since the MPU is a dynamic device, the  $\phi_1$  clock can be stopped for a maximum time PWCH1 without destroying data within the MPU. TSC then can be used in a short Direct Memory Access (DMA) application.

Fig. 16 shows the effect of TSC on the MPU. The Address Bus and R/W line will reach the high impedance state at  $t_{TSD}$  (three-state delay), with VMA being forced "Low". In this example, the Data Bus is also in the high impedance state while  $\phi_2$  is being held "Low" since DBE =  $\phi_2$ . At this point in time, a DMA transfer could occur on cycles #3 and #4. When TSC is returned "Low," the MPU address and R/W lines return to the bus. Because it is too late in cycle #5 to access memory, this cycle is dead and used for synchronization. Program execution resumes in cycle #6.

#### • Valid Memory Address (VMA)

This output indicates to peripheral devices that there is a valid address on the address bus. In normal operation, this signal should be utilized for enabling peripheral interfaces such as the PIA and ACIA. This signal is not three-state. One standard TTL load and 90pF may be directly driven by this active "High" signal.

#### • Halt (HALT)

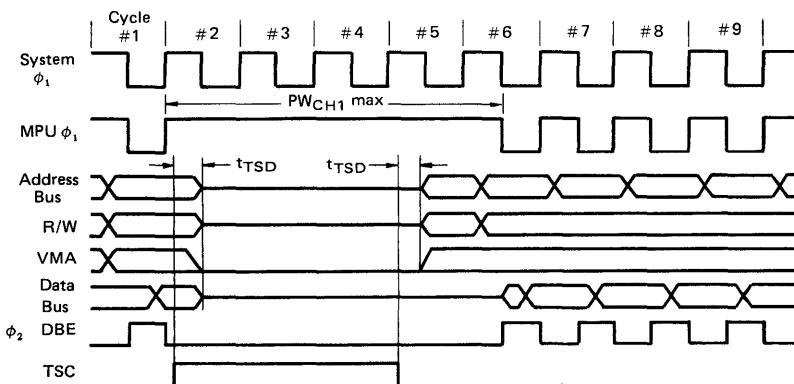
When this input is in the "Low" state, all activity in the machine will be halted. This input is level sensitive.

The HALT line provides an input to the MPU to allow control or program execution by an outside source. If HALT is "High", the MPU will execute the instructions; if it is "Low", the MPU will go to a halted or idle mode. A response signal, Bus Available (BA) provides an indication of the current MPU status. When BA is "Low", the MPU is in the process of executing the control program; if BA is "High", the MPU has halted and all internal activity has stopped.

When BA is "High", the Address Bus, Data Bus, and R/W line will be in a high impedance state, effectively removing the MPU from the system bus. VMA is forced "Low" so that the floating system bus will not activate any device on the bus that is enabled by VMA.

While the MPU is halted, all program activity is stopped, and if either an NMI or IRQ interrupt occurs, it will be latched into the MPU and acted on as soon as the MPU is taken out of the halted mode. If a RES command occurs while the MPU is halted, the following states occur: VMA = "Low", BA = "Low", Data Bus = high impedance, R/W = "High" (read state), and the Address Bus will contain address FFFE as long as RES is "Low". As soon as the HALT line goes "High", the MPU will go to locations FFFE and FFFF for the address of the reset routine.

Fig. 18 shows the timing relationships involved when halting the MPU. The instruction illustrated is a one byte, 2 cycle instruction such as CLRA. When HALT goes "Low", the MPU will halt after completing execution of the current instruction. The transition of HALT must occur  $t_{PCS}$  before the trailing edge of  $\phi_1$  of the last cycle of an instruction (point A of Fig. 18). HALT must not go "Low" any time later than the minimum  $t_{PCS}$  specified.



**Figure 16 TSC Control Timing**

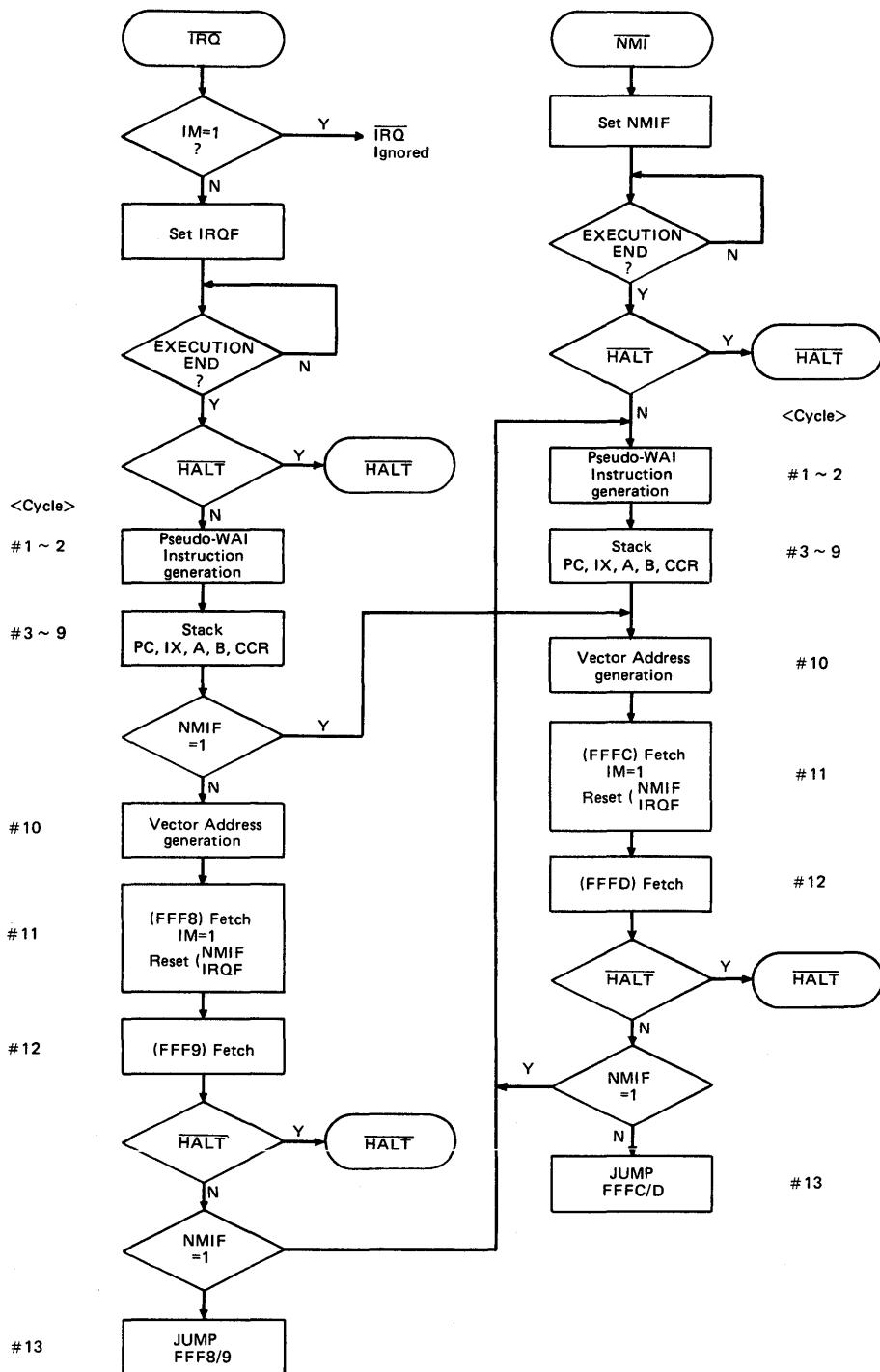
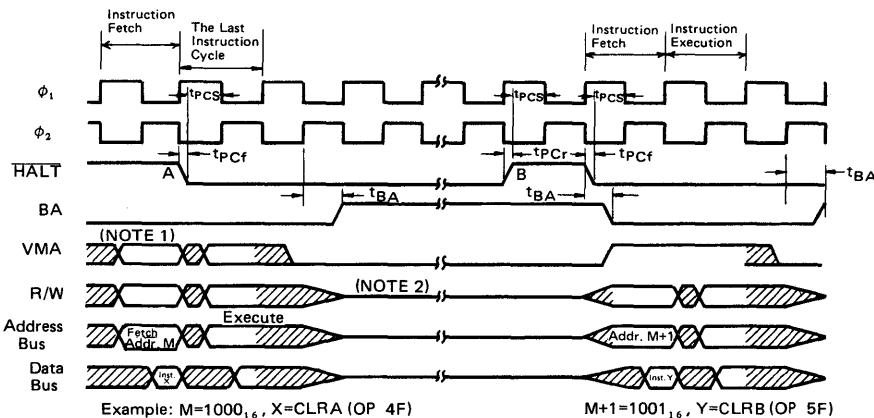


Figure 17 MPU Interrupt Flow Chart



(NOTE) 1. Oblique lines indicate indeterminate range of data.  
2. Midrange waveform indicates high impedance state.

Figure 18 HALT and Single Instruction Execution for System Debug

Table 2 Operation States of MPU and Signal Outputs (Except the Execution of Instruction)

Signals	Halt state	Reset state	Halt and Reset state	WAI state	TSC state
BA	"H"	"L"	"L"	"H"	"L"
VMA	"L"	"L"	"L"	"L"	"L"
R/W	"T"	"H"	"H"	"T"	"T"
A <sub>0</sub> ~ A <sub>15</sub>	"T"	(FFFE) <sub>16</sub>	(FFFE) <sub>16</sub>	"T"	"T"
D <sub>0</sub> ~ D <sub>7</sub>	"T"	"T"	"T"	"T"	-

"T" indicates high impedance state.

The fetch of the OP code by the MPU is the first cycle of the instruction. If HALT had not been "Low" at Point A but went "Low" during  $\phi_2$  of the cycle, the MPU would have halted after completion of the following instruction. BA will go "High" by time  $t_{BA}$  (bus available delay time) after the last instruction cycle. At this point in time, VMA is "Low" and R/W, Address Bus, and the Data Bus are in the high impedance state.

To debug programs it is advantageous to step through programs instruction by instruction. To do this, HALT must be brought "High" for one MPU cycle and then returned "Low" as shown at point B of Fig. 18. Again, the transitions of HALT must occur  $t_{PCr}$  before the trailing edge of  $\phi_1$ . BA will go "Low" at  $t_{BA}$  after the leading edge of the next  $\phi_1$ , indicating that the Address Bus, Data Bus, VMA and R/W lines are back on the bus. A single byte, 2 cycle instruction such as LSR is used for this example also. During the first cycle, the instruction Y is fetched from address M+1. BA returns "High" at  $t_{BA}$  on the last cycle of the instruction indicating the MPU is off the bus, if instruction Y had been three cycles, the width of the BA "Low" time would have been increased by one cycle.

Table 2 shows the relation between the state of MPU and signal outputs.

#### ■ MPU INSTRUCTION SET

This Section will provide a brief introduction and discuss their use in developing HD6800 MPU control programs. The HD6800 MPU has a set of 72 different executable source instructions. Included are binary and decimal arithmetic, logical, shift, rotate, load, store, conditional or unconditional branch, interrupt and stack manipulation instructions.

Each of the 72 executable instructions of the source language assembles into 1 to 3 bytes of machine code. The number of bytes depends on the particular instruction and on the addressing mode. (The addressing modes which are available for use with the various executive instructions are discussed later.)

The coding of the first (or only) byte corresponding to an executable instruction is sufficient to identify the instruction and the addressing mode. The hexadecimal equivalents of the binary codes, which result from the translation of the 72 instructions in all valid modes of addressing, are shown in Table 3. There are 197 valid machine codes, 59 of the 256 possible codes being unassigned.

When an instruction translates into two or three bytes of code, the second byte, or second and third bytes contain(s) an operand, an address, or information from which an address is obtained during execution.

Microprocessor instructions are often divided into three general classifications; (1) memory reference, so called because they operate on specific memory locations; (2) operating instructions that function without needing a memory reference; (3) I/O instructions for transferring data between the microprocessor and peripheral devices.

In many instances, the HD6800 MPU performs the same operation on both its internal accumulators and the external

memory locations. In addition, the HD6800 MPU allow the MPU to treat peripheral devices exactly like other memory locations, hence, no I/O instructions as such are required. Because of these features, other classifications are more suitable for introducing the HD6800's instruction set: (1) Accumulator and memory operations; (2) Program control operations; (3) Condition Code Register operations.

For Accumulator and Memory Operations, refer to Table 4.

**Table 3 Hexadecimal Values of Machine Codes**

LSB MSB	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	*	NOP (IMP)	*	*	*	*	TAP (IMP)	TPA (IMP)	INX (IMP)	DEX (IMP)	CLV (IMP)	SEV (IMP)	CLC (IMP)	SEC (IMP)	CLI (IMP)	SEI (IMP)
1	SBA (A, B)	CBA (A, B)	*	*	*	*	TAB (IMP)	TBA (IMP)	*	DAA (IMP)	*	ABA (IMP)	*	*	*	*
2	BRA (REL)	*	BHI (REL)	BLS (REL)	BCC (REL)	BCS (REL)	BNE (REL)	BEQ (REL)	BVC (REL)	BVS (REL)	BPL (REL)	BMI (REL)	BGE (REL)	BLT (REL)	BGT (REL)	BLE (REL)
3	TSX (IMP)	INS (IMP)	PUL (A)	PUL (B)	DES (IMP)	TXS (IMP)	PSH (A)	PSH (B)	*	RTS (IMP)	*	RTI (IMP)	*	*	WAI (IMP)	SWI (IMP)
4	NEG (A)	*	*	COM (A)	LSR (A)	*	ROR (A)	ASR (A)	ASL (A)	ROL (A)	DEC (A)	*	INC (A)	TST (A)	*	CLR (A)
5	NEG (B)	*	*	COM (B)	LSR (B)	*	ROR (B)	ASR (B)	ASL (B)	ROL (B)	DEC (B)	*	INC (B)	TST (B)	*	CLR (B)
6	NEG (IND)	*	*	COM (IND)	LSR (IND)	*	ROR (IND)	ASR (IND)	ASL (IND)	ROL (IND)	DEC (IND)	*	INC (IND)	TST (IND)	JMP (IND)	CLR (IND)
7	NEG (EXT)	*	*	COM (EXT)	LSR (EXT)	*	ROR (EXT)	ASR (EXT)	ASL (EXT)	ROL (EXT)	DEC (EXT)	*	INC (EXT)	TST (EXT)	JMP (EXT)	CLR (EXT)
8	SUB (IMM) <sup>(A)</sup>	CMP (IMM) <sup>(A)</sup>	SBC (IMM) <sup>(A)</sup>	*	AND (IMM) <sup>(A)</sup>	BIT (IMM) <sup>(A)</sup>	LDA (IMM) <sup>(A)</sup>	*	EOR (IMM) <sup>(A)</sup>	ADC (IMM) <sup>(A)</sup>	ORA (IMM) <sup>(A)</sup>	ADD (IMM)	CPX (IMM) <sup>(A)</sup>	BSR (REL)	LDS (IMM)	*
9	SUB (DIR) <sup>(A)</sup>	CMP (DIR) <sup>(A)</sup>	SBC (DIR) <sup>(A)</sup>	*	AND (DIR) <sup>(A)</sup>	BIT (DIR) <sup>(A)</sup>	LDA (DIR) <sup>(A)</sup>	STA (DIR) <sup>(A)</sup>	EOR (DIR) <sup>(A)</sup>	ADC (DIR) <sup>(A)</sup>	ORA (DIR) <sup>(A)</sup>	ADD (DIR)	CPX (DIR) <sup>(A)</sup>	*	LDS (DIR)	STS (DIR)
A	SUB (IND) <sup>(A)</sup>	CMP (IND) <sup>(A)</sup>	SBC (IND) <sup>(A)</sup>	*	AND (IND) <sup>(A)</sup>	BIT (IND) <sup>(A)</sup>	LDA (IND) <sup>(A)</sup>	STA (IND) <sup>(A)</sup>	EOR (IND) <sup>(A)</sup>	ADC (IND) <sup>(A)</sup>	ORA (IND) <sup>(A)</sup>	ADD (IND)	CPX (IND) <sup>(A)</sup>	JSR (IND)	LDS (IND)	STS (IND)
B	SUB (EXT) <sup>(A)</sup>	CMP (EXT) <sup>(A)</sup>	SBC (EXT) <sup>(A)</sup>	*	AND (EXT) <sup>(A)</sup>	BIT (EXT) <sup>(A)</sup>	LDA (EXT) <sup>(A)</sup>	STA (EXT) <sup>(A)</sup>	EOR (EXT) <sup>(A)</sup>	ADC (EXT) <sup>(A)</sup>	ORA (EXT) <sup>(A)</sup>	ADD (EXT)	CPX (EXT) <sup>(A)</sup>	JSR (EXT)	LDS (EXT)	STS (EXT)
C	SUB (IMM) <sup>(B)</sup>	CMP (IMM) <sup>(B)</sup>	SBC (IMM) <sup>(B)</sup>	*	AND (IMM) <sup>(B)</sup>	BIT (IMM) <sup>(B)</sup>	LDA (IMM) <sup>(B)</sup>	*	EOR (IMM) <sup>(B)</sup>	ADC (IMM) <sup>(B)</sup>	ORA (IMM) <sup>(B)</sup>	ADD (IMM)	*	*	LDX (IMM)	*
D	SUB (DIR) <sup>(B)</sup>	CMP (DIR) <sup>(B)</sup>	SBC (DIR) <sup>(B)</sup>	*	AND (DIR) <sup>(B)</sup>	BIT (DIR) <sup>(B)</sup>	LDA (DIR) <sup>(B)</sup>	STA (DIR) <sup>(B)</sup>	EOR (DIR) <sup>(B)</sup>	ADC (DIR) <sup>(B)</sup>	ORA (DIR) <sup>(B)</sup>	ADD (DIR)	*	*	LDX (DIR) <sup>(B)</sup>	STX (DIR)
E	SUB (IND) <sup>(B)</sup>	CMP (IND) <sup>(B)</sup>	SBC (IND) <sup>(B)</sup>	*	AND (IND) <sup>(B)</sup>	BIT (IND) <sup>(B)</sup>	LDA (IND) <sup>(B)</sup>	STA (IND) <sup>(B)</sup>	EOR (IND) <sup>(B)</sup>	ADC (IND) <sup>(B)</sup>	ORA (IND) <sup>(B)</sup>	ADD (IND)	*	*	LDX (IND)	STX (IND)
F	SUB (EXT) <sup>(B)</sup>	CMP (EXT) <sup>(B)</sup>	SBC (EXT) <sup>(B)</sup>	*	AND (EXT) <sup>(B)</sup>	BIT (EXT) <sup>(B)</sup>	LDA (EXT) <sup>(B)</sup>	STA (EXT) <sup>(B)</sup>	EOR (EXT) <sup>(B)</sup>	ADC (EXT) <sup>(B)</sup>	ORA (EXT) <sup>(B)</sup>	ADD (EXT)	*	*	LDX (EXT)	STX (EXT)

DIR = Direct Addressing Mode

EXT = Extended Addressing Mode

IMM = Immediate Addressing Mode

IND = Index Addressing Mode

IMP = Implied Addressing Mode

REL = Relative Addressing Mode

A = Accumulator A

B = Accumulator B

Table 4 Accumulator and Memory Operations

Operation	Mnemonic	Addressing Modes										Boolean/ Arithmetic Operation	Cond. Code Reg.					
		IMMED		DIRECT		INDEX		EXTND		IMPLIED			H	I	N	Z	V	
		OP ~ #	OP ~ #	OP ~ #	OP ~ #	OP ~ #	OP ~ #	OP ~ #	OP ~ #	OP ~ #	OP ~ #		5	4	3	2	1	0
Add	ADDA	8B	2	2	9B	3	2	AB	5	2	BB	4	3					
	ADDB	CB	2	2	DB	3	2	EB	5	2	FB	4	3					
Add Acmtr	ABA																	
Add with Carry	ADCA	89	2	2	99	3	2	A9	5	2	B9	4	3	1B	2	1	A + B → A	
	ADCB	C9	2	2	D9	3	2	E9	5	2	F9	4	3					
And	ANDA	84	2	2	94	3	2	A4	5	2	B4	4	3					
	ANDB	C4	2	2	D4	3	2	E4	5	2	F4	4	3					
Bit Test	BITA	85	2	2	95	3	2	A5	5	2	B5	4	3					
	BITB	C5	2	2	D5	3	2	E5	5	2	F5	4	3					
Clear	CLR																	
	CLRA																	
	CLRB																	
Compare	CMPA	81	2	2	91	3	2	A1	5	2	B1	4	3					
	CMPB	C1	2	2	D1	3	2	E1	5	2	F1	4	3					
Compare Acmtr	CBA																	
Complement, 1's	COM																	
	COMA																	
	COMB																	
Complement, 2's (Negate)	NEG																	
	NEGA																	
	NEGB																	
Decimal Adjust, A	DAA																	
Decrement	DEC																	
	DECA																	
	DEC B																	
Exclusive OR	EORA	88	2	2	98	3	2	A8	5	2	B8	4	3					
	EORB	C8	2	2	D8	3	2	E8	5	2	F8	4	3					
Increment	INC																	
	INCA																	
	INCB																	
Load Acmtr	LDAA	86	2	2	96	3	2	A6	5	2	B6	4	3					
	LDAB	C6	2	2	D6	3	2	E6	5	2	F6	4	3					
Or, Inclusive	ORAA	8A	2	2	9A	3	2	AA	5	2	BA	4	3					
	ORAB	CA	2	2	DA	3	2	EA	5	2	FA	4	3					
Push Data	PSHA																	
	PSHB																	
Pull Data	PULA																	
	PULB																	
Rotate Left	ROL																	
	ROLA																	
	ROLB																	
Rotate Right	ROR																	
	RORA																	
	RORB																	
Shift Left, Arithmetic	ASL																	
	ASLA																	
	ASLB																	
Shift Right, Arithmetic	ASR																	
	ASRA																	
	ASRB																	
Shift Right, Logic	LSR																	
	LSRA																	
	LSRB																	
Store Acmtr	STAA																	
	STAB																	
Subtract	SUBA	80	2	2	90	3	2	A0	5	2	B0	4	3					
	SUBB	CO	2	2	D0	3	2	E0	5	2	F0	4	3					
Subtract Acmtr	SBA																	
Subtr with Carry	SBCA	82	2	2	92	3	2	A2	5	2	B2	4	3					
	SBCB	C2	2	2	D2	3	2	E2	5	2	F2	4	3					
Transfer Acmtr	TAB																	
	TBA																	
Test Zero or Minus	TST																	
	TSTA																	
	TSTB																	

**LEGEND:**

OP Operation Code (Hexadecimal)

# Number of MPU Cycles

+ Number of Program Bytes

+ Arithmetic Plus

- Arithmetic Minus

• Boolean AND

Msp Contents of memory location pointed to by Stack Pointer

⊕ Boolean Inclusive OR

⊕ Boolean Exclusive OR

M Complement of M

→ Transfer Into

0 Bit = Zero

00 Byte = Zero

H Half-carry from bit 3

I Interrupt mask

N Negative (sign) bit

Z Zero (byte)

V Overflow, 2's complement

C Carry from bit 7

**CONDITION CODE SYMBOLS:**

R Reset Always

S Set Always

T Test and set if true, cleared otherwise

N Not Affected

(Note) Accumulator addressing mode instructions are included in the column for IMPLIED addressing.

**CONDITION CODE REGISTER NOTES:**

(Bit set if test is true and cleared otherwise)

① (Bit V) Test: Result = 10000000?

② (Bit C) Test: Result ≠ 00000000?

③ (Bit C) Test: Decimal value of most significant BCD Character greater than nine? (Not cleared if previously set.)

④ (Bit V) Test: Operand = 10000000 prior to execution?

⑤ (Bit V) Test: Operand = 01111111 prior to execution?

⑥ (Bit V) Test: Set equal to result of N⊕C after shift has occurred.

## ■ PROGRAM CONTROL OPERATIONS

Program Control operation can be subdivided into two categories: (1) Index Register/Stack Pointer instructions; (2) Jump and Branch operations.

### • Index Register/Stack Pointer Operations

The instructions for direct operation on the MPU's Index Register and Stack Pointer are summarized in Table 5. Decrement (DEX, DES), increment (INX, INS), load (LDX, LDS), and store (STX, STS) instructions are provided for both. The Compare instruction, CPX, can be used to compare the Index Register to a 16-bit value and update the Condition Code Register accordingly.

The TSX instruction causes the Index Register to be loaded with the address of the last data byte put onto the "stack". The TXS instruction loads the Stack Pointer with a value equal to one less than the current contents of the Index Register. This causes the next byte to be pulled from the "stack" to come from the location indicated by the Index Register. The utility of these two instructions can be clarified by describing the "stack" concept relative to the HMCS 6800 system.

The "stack" can be thought of as a sequential list of data stored in the MPU's read/write memory. The Stack Pointer contains a 16-bit memory address that is used to access the list from one end on a last-in-first-out (LIFO) basis in contrast to the random access mode used by the MPU's other addressing modes.

The HD6800 MPU instruction set and interrupt structure allow extensive use of the stack concept for efficient handling of data movement, subroutines and interrupts. The instructions can be used to establish one or more "stacks" anywhere in read/write memory. Stack length is limited only by the amount of memory that is made available.

Operation of the Stack Pointer with the Push and Pull instructions is illustrated in Figs. 19 and 20. The Push instruction (PSHA) causes the contents of the indicated accumulator (A in

this example) to be stored in memory at the location indicated by the Stack Pointer. The Stack Pointer is automatically decremented by one following the storage operation and is "pointing" to the next empty stack location.

The Pull instruction (PULA or PULB) causes the last byte stacked to be loaded into the appropriate accumulator. The Stack Pointer is automatically incremented by one just prior to the data transfer so that it will point to the last byte stacked rather than the next empty location. Note that the PULL instruction does not "remove" the data from memory; in the example, 1A is still in location (m+1) following execution of PULA. A subsequent PUSH instruction would overwrite that location with the new "pushed" data.

Execution of the Branch to Subroutine (BSR) and Jump to Subroutine (JSR) instructions cause a return address to be save on the stack as shown in Figs. 21 through 23. The stack is decremented after each byte of the return address is pushed onto the stack. For both of these instructions, the return address is the memory location following the bytes of code that correspond to the BSR and JSR instruction. The code required for BSR or JSR may be either two or three bytes, depending on whether the JSR is in the indexed (two bytes) or the extended (three bytes) addressing mode. Before it is stacked, the Program Counter is automatically incremented the correct number of times to be pointing at the location of the next instruction. The Return from Subroutine instruction, RTS, causes the return address to be retrieved and loaded into the Program Counter as shown in Fig. 24.

There are several operations that cause the status of the MPU to be saved on the stack. The Software Interrupt (SWI) and Wait for Interrupt (WAI) instructions as well as the maskable (IRQ) and non-maskable (NMI) hardware interrupts all cause the MPU's internal registers (except for the Stack Pointer itself) to be stacked as shown in Fig. 25. MPU status is restored by the Return from interrupt, RTI, as shown in Fig. 26.

Table 5 Index Register and Stack Pointer Instructions

Operation	Mnemonic	Addressing Modes												Boolean/ Arithmetic Operation	Cond. Code Reg.						
		IMMED			DIRECT			INDEX			EXTND				5	4	3	2	1	0	
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	H	I	N	Z	V
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	6	2	BC	5	3		(X <sub>H</sub> ) - (M), (X <sub>L</sub> ) - (M+1)	•	•	①	‡	②	•
Decrement Index Reg	DEX										09	4	1	X - 1 → X	•	•	•	•	‡	•	•
Decrement Stack Pntr	DES										34	4	1	SP - 1 → SP	•	•	•	•	•	•	•
Increment Index Reg	INX										08	4	1	X + 1 → X	•	•	•	•	‡	•	•
Increment Stack Pntr	INS										31	4	1	SP + 1 → SP	•	•	•	•	•	•	•
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	6	2	FE	5	3	M → X <sub>H</sub> , (M+1) → X <sub>L</sub>	•	•	•	③	‡	R	•
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	6	2	BE	5	3	M → SP <sub>H</sub> , (M+1) → SP <sub>L</sub>	•	•	•	③	‡	R	•
Store Index Reg	STX				DF	5	2	EF	7	2	FF	6	3	X <sub>H</sub> → M, X <sub>L</sub> → (M + 1)	•	•	•	③	‡	R	•
Store Stack Pntr	STS				9F	5	2	AF	7	2	BF	6	3	SP <sub>H</sub> → M, SP <sub>L</sub> → (M + 1)	•	•	•	③	‡	R	•
Index Reg → Stack Pntr	TXS										35	4	1	X - 1 → SP	•	•	•	•	•	•	•
Stack Pntr → Index Reg	TSX										30	4	1	SP + 1 → X	•	•	•	•	•	•	•

① (Bit N) Test: Sign bit of most significant (MS) byte of result = 1?

② (Bit V) Test: 2's complement overflow from subtraction of ms bytes?

③ (Bit N) Test: Result less than zero? (Bit 15 = 1)

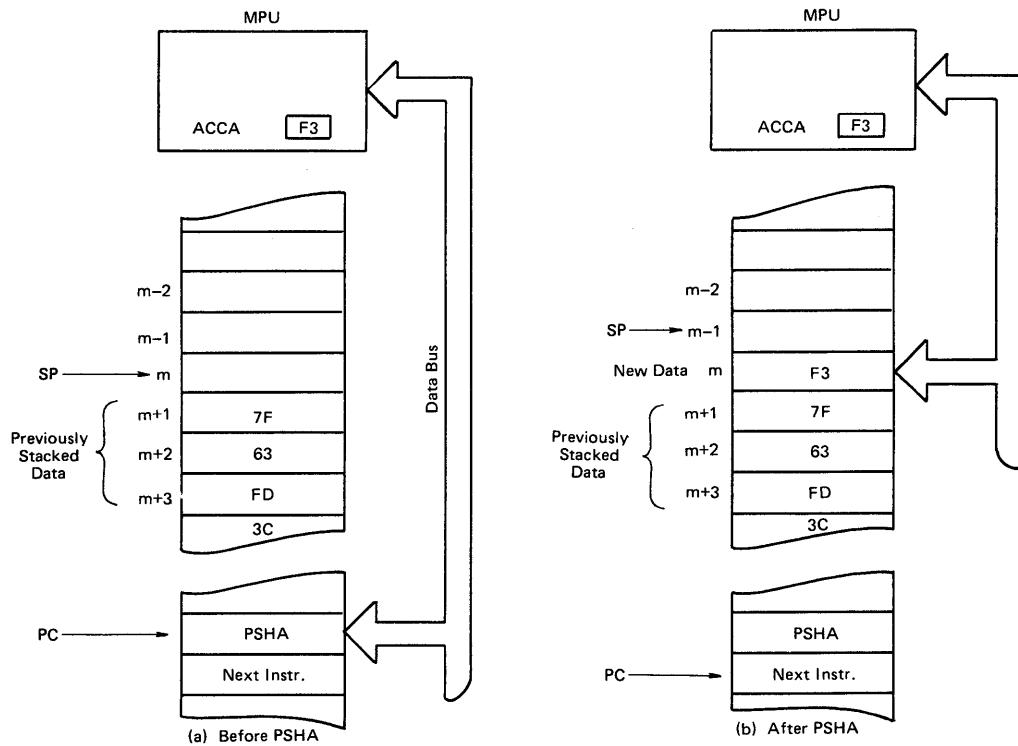


Figure 19 Stack Operation (Push Instruction)

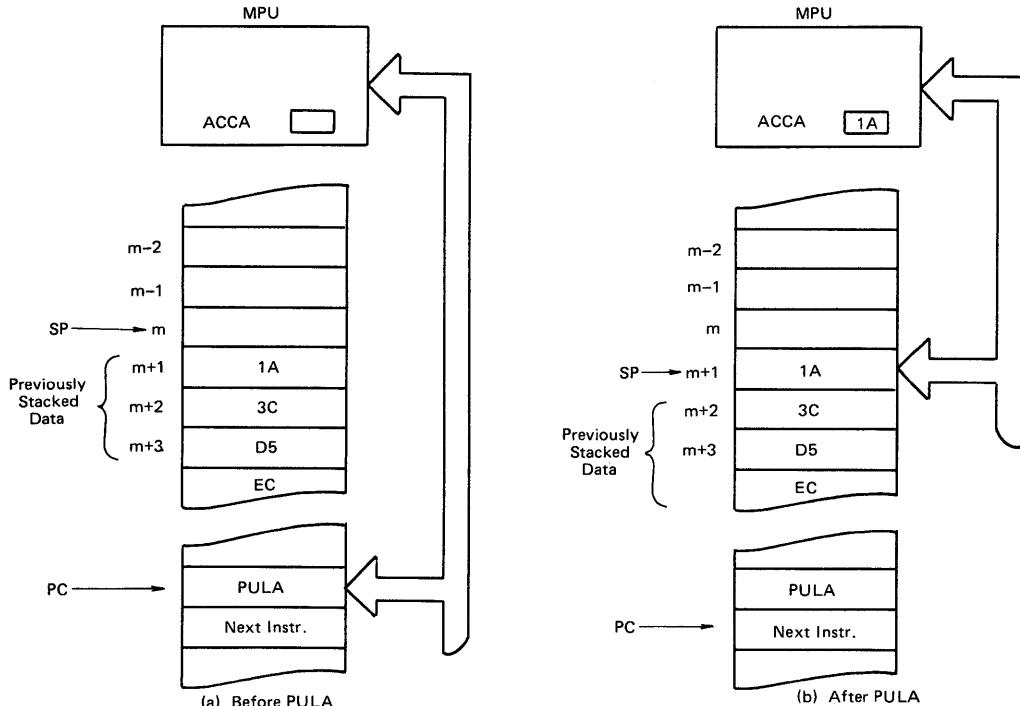
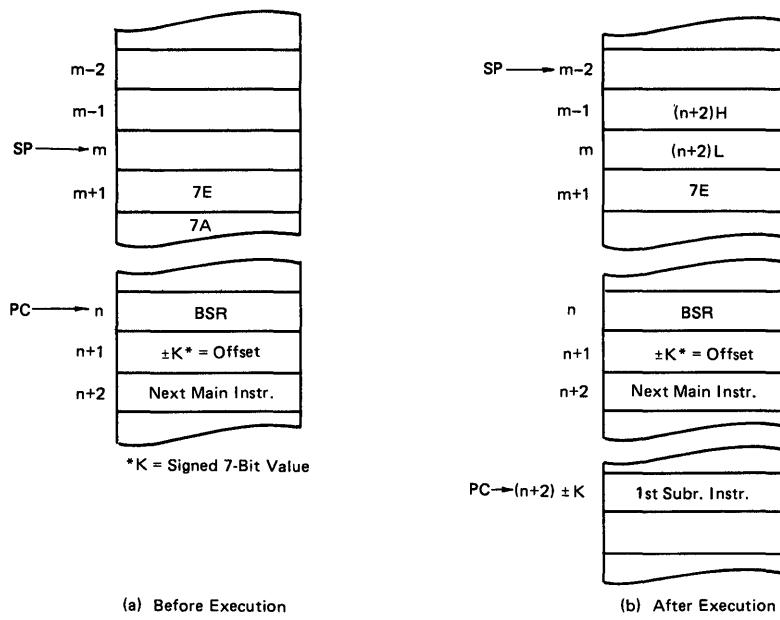
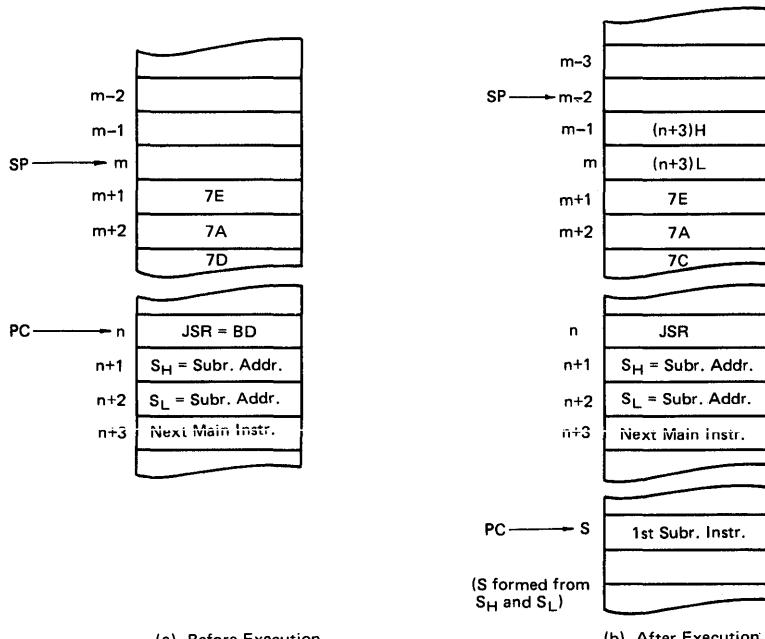


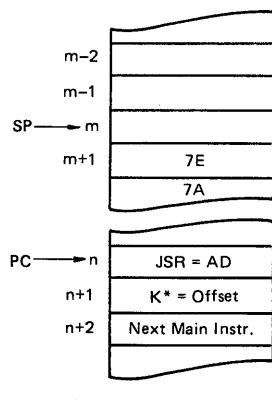
Figure 20 Stack Operation (Pull Instruction)



**Figure 21 Program Flow for BSR**

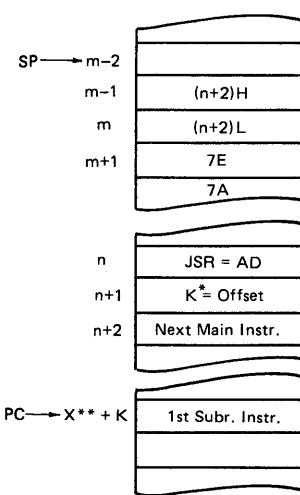


**Figure 22 Program Flow for JSR (Extended)**



\*K = 8-Bit Unsigned Value

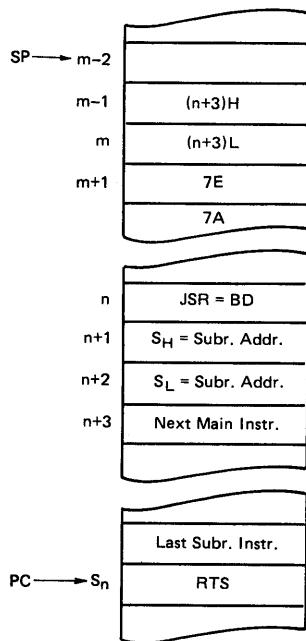
(a) Before Execution



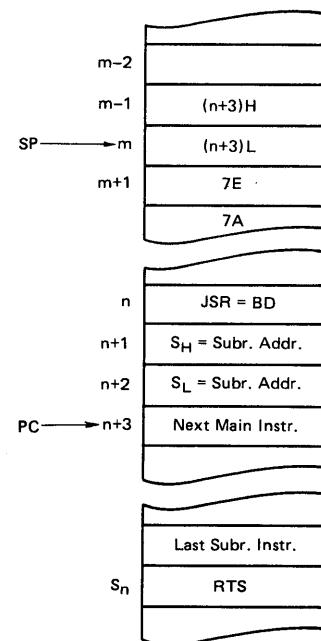
\*\*Contents of Index Register

(b) After Execution

**Figure 23 Program Flow for JSR (Indexed)**



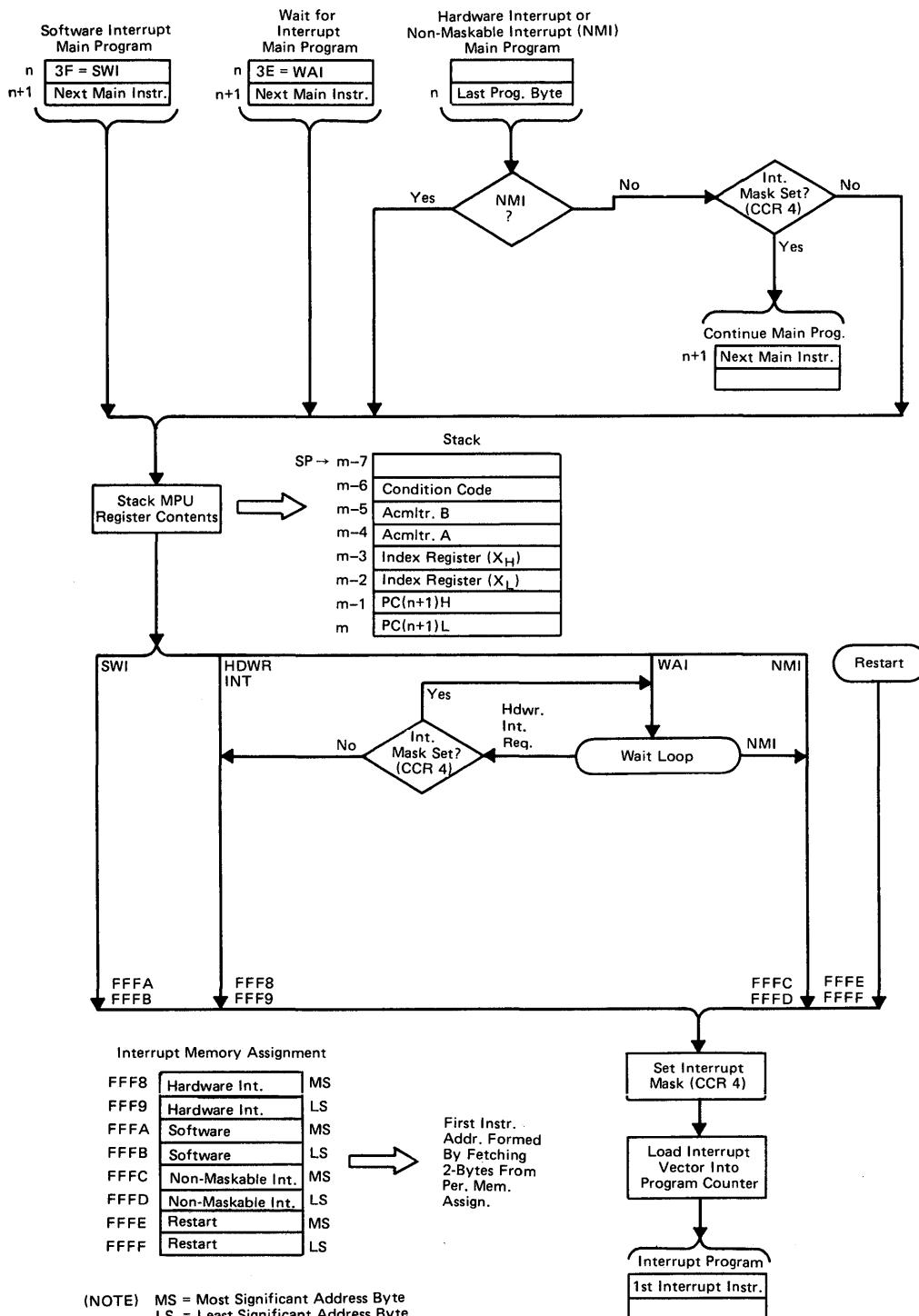
(a) Before Execution



(b) After Execution

**Figure 24 Program Flow for RTS**

## HD6800, HD68A00, HD68B00



**Figure 25 Program Flow for Interrupts**

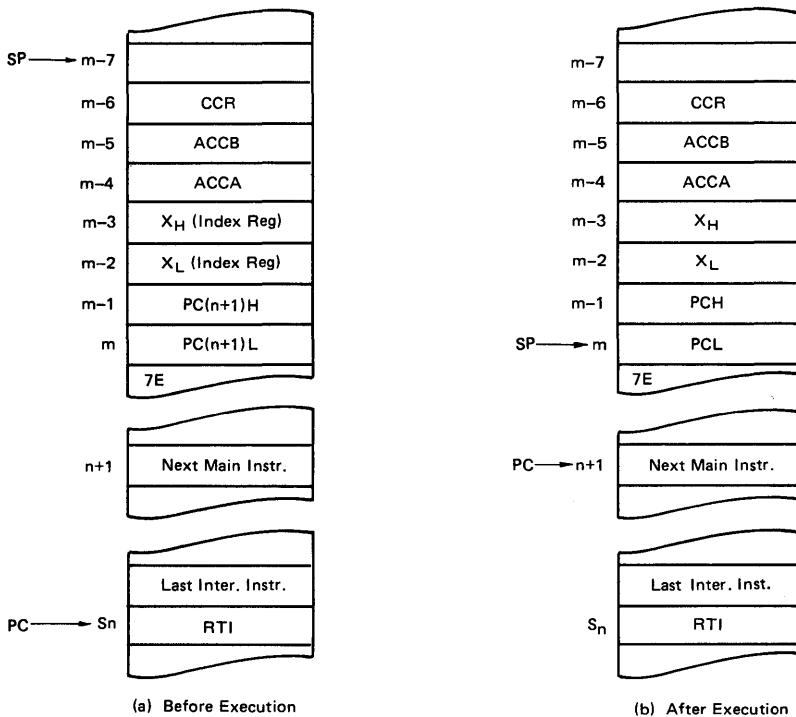


Figure 26 Program Flow for RTI

- **Jump and Branch Operation**

The Jump and Branch instructions are summarized in Table 6. These instructions are used to control the transfer of operation from one point to another in the control program.

The No Operation instruction, NOP, while included here, is a jump operation in a very limited sense. Its only effect is to increment the Program Counter by one. It is useful during program development as a "stand-in" for some other instruction that is to be determined during debug. It is also used for equalizing the execution time through alternate paths in a control program.

Execution of the Jump Instruction, JMP, and Branch Always, BRA, affects program flow as shown in Fig. 27. When the MPU encounters the Jump (Index) instruction, it adds the offset to the value in the Index Register and uses the result as the address of the next instruction to be executed. In the extended addressing mode, the address of the next instruction to be executed is fetched from the two locations immediately following the JMP instruction. The Branch Always (BRA) instruction is similar to the JMP (extended) instruction except that the relative addressing mode applies and the branch is limited to the range within -125 or +127 bytes of the branch instruction itself. The opcode for the BRA instruction requires one less byte than JMP (extended) but takes one more cycle to execute.

The effect on program flow for the Jump to Subroutine (JSR) and Branch to Subroutine (BSR) is shown in Figs. 21 through 23. Note that the Program Counter is properly in-

cremented to be pointing at the correct return address before it is stacked. Operation of the Branch to Subroutine and Jump to Subroutine (extended) instruction is similar except for the range. The BSR instruction requires less opcode than JSR (2 bytes versus 3 bytes) and also executes one cycle faster than JSR. The Return from Subroutine, RTS, is used at the end of a subroutine to return to the main program as indicated in Fig. 24.

The effect of executing the Software Interrupt, SWI, and the Wait for Interrupt, WAI, and their relationship to the hardware interrupts is shown in Fig. 25. SWI causes the MPU contents to be stacked and then fetches the starting address of the interrupt routine from the memory locations that respond to the addresses FFFA and FFFB. Note that as in the case of the subroutine instructions, the Program Counter is incremented to point at the correct return address before being stacked. The Return from Interrupt instruction, RTI, (Fig. 26) is used at the end of an interrupt routine to restore control to the main program. The SWI instruction is useful for inserting break points in the control program, that is, it can be used to stop operation and put the MPU registers in memory where they can be examined. The WAI instruction is used to decrease the time required to service a hardware interrupt; it stacks the MPU contents and then waits for the interrupt to occur, effectively removing the stacking time from a hardware interrupt sequence.

Table 6 JUMP/BRANCH Instruction

Operation	Mnemonic	Addressing Modes										Branch Test	Cond. Code Reg.						
		RELATIVE			INDEX			EXTND			IMPLIED			5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	H	I	N	Z	V	C
Branch Always	BRA	20	4	2										None	•	•	•	•	•
Branch If Carry Clear	BCC	24	4	2										C = 0	•	•	•	•	•
Branch If Carry Set	BCS	25	4	2										C = 1	•	•	•	•	•
Branch If = Zero	BEQ	27	4	2										Z = 1	•	•	•	•	•
Branch If $\geq$ Zero	BGE	2C	4	2										$N \oplus V = 0$	•	•	•	•	•
Branch If > Zero	BGT	2E	4	2										$Z + (N \oplus V) = 0$	•	•	•	•	•
Branch If Higher	BHI	22	4	2										$C + Z = 0$	•	•	•	•	•
Branch If $\leq$ Zero	BLE	2F	4	2										$Z + (N \oplus V) = 1$	•	•	•	•	•
Branch If Lower Or Same	BLS	23	4	2										$C + Z = 1$	•	•	•	•	•
Branch If $<$ Zero	BLT	2D	4	2										$N \oplus V = 1$	•	•	•	•	•
Branch If Minus	BMI	2B	4	2										$N = 1$	•	•	•	•	•
Branch If Not Equal Zero	BNE	26	4	2										$Z = 0$	•	•	•	•	•
Branch If Overflow Clear	BVC	28	4	2										$V = 0$	•	•	•	•	•
Branch If Overflow Set	BVS	29	4	2										$V = 1$	•	•	•	•	•
Branch If Plus	BPL	2A	4	2										$N = 0$	•	•	•	•	•
Branch To Subroutine	BSR	8D	8	2											•	•	•	•	•
Jump	JMP				6E	4	2	7E	3	3									
Jump To Subroutine	JSR				AD	8	2	BD	9	3				01	2	1			
No Operation	NOP													3B	10	1			
Return From Interrupt	RTI													39	5	1			
Return From Subroutine	RTS													3F	12	1			
Software Interrupt	SWI													3E	9	1			
Wait for Interrupt	WAI																		

(1) (All) Load Condition Code Register from Stack. (See Special Operations)

(2) (Bit I) Set when interrupt occurs. If previously set, a Non-Maskable interrupt is required to exit the wait state.

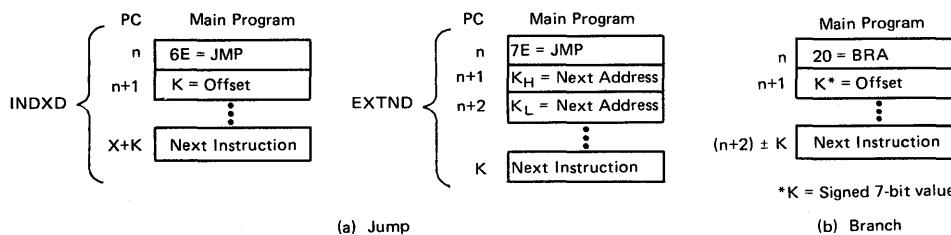


Figure 27 Program Flow for JUMP/BRANCH Instructions

BMI : N = 1 ;	BEQ : Z = 1 ;
BPL : N = 0 ;	BNE : Z = 0 ;
BVC : V = 0 ;	BCC : C = 0 ;
BVS : V = 1 ;	BCS : C = 1 ;
BHI : C + Z = 0 ;	BLT : N $\oplus$ V = 1 ;
BLS : C + Z = 1 ;	BGE : N $\oplus$ V = 0 ;
BLE : Z + (N $\oplus$ V) = 1 ;	
BGT : Z + (N $\oplus$ V) = 0 ;	

Figure 28 Conditional Branch Instructions

The conditional branch instructions, Fig. 28, consists of seven pairs of complementary instructions. They are used to test the results of the preceding operation and either continue with the next instruction in sequence (test fails) or cause a branch to another point in the program (test succeeds).

Four of the pairs are used for simple tests of status bits N, Z, V, and C:

1. Branch on Minus (BMI) and Branch On Plus (BPL) tests the sign bit, N, to determine if the previous result was negative or positive, respectively.
2. Branch On Equal (BEQ) and Branch On Not Equal (BNE) are used to test the zero status bit, Z, to determine whether or not the result of the previous operation was equal to "0". These two instructions are useful following a Compare (CMP) instruction to test for equality between an accumulator and the operand. They are also used following the Bit Test (BIT) to determine whether or not the same bit positions are set in an accumulator and the operand.

3. Branch On Overflow Clear (BVC) and Branch On Overflow Set (BVS) tests the state of the V bit to determine if the previous operation caused an arithmetic overflow.
4. Branch On Carry Clear (BCC) and Branch On Carry Set (BCS) tests the state of the C bit to determine if the previous operation caused a carry to occur. BCC and BCS are useful for testing relative magnitude when the values being tested are regarded as unsigned binary numbers, that is, the values are in the range "00" (lowest) of "FF" (highest). BCC following a comparison (CMP) will cause a branch if the (unsigned) value in the accumulator is higher than or the same as the value of the operand. Conversely, BCS will cause a branch if the accumulator value is lower than the operand.

The Fifth complementary pair, Branch On Higher (BHI) and Branch On Lower or Same (BLS) are in a sense complements to BCC and BCS. BHI tests for both C and Z = "0", if used following a CMP, it will cause a branch if the value in the accumulator is higher than the operand. Conversely, BLS will cause a branch if the unsigned binary value in the accumulator is lower than or the same as the operand.

The remaining two pairs are useful in testing results of operations in which the values are regarded as signed two's complement numbers. This differs from the unsigned binary case in the following sense: In unsigned, the orientation is higher or lower; in signed two's complement, the comparison is between larger or smaller where the range of values is between -128 and +127.

Branch On Less Than Zero (BLT) and Branch On Greater Than Or Equal Zero (BGE) test the status bits for  $N \oplus V = "1"$  and  $N \oplus V = "0"$ , respectively. BLT will always cause a branch following an operation in which two negative numbers were added. In addition, it will cause a branch following a CMP in which the value in the accumulator was negative and the operand was positive. BLT will never cause a branch following a CMP in which the accumulator value was positive and the operand negative. BGE, the complement to BLT, will cause a branch following operations in which two positive values were added or in which the result was "0".

The last pair, Branch On Less Than Or Equal Zero (BLE) and Branch On Greater Than Zero (BGTR) test the status bits for  $Z \oplus (N + V) = "1"$  and  $Z \oplus (N + V) = "0"$ , respectively. The action of BLE is identical to that for BLT except that a branch will also occur if the result of the previous result was "0". Conversely, BGTR is similar to BGE except that no branch will occur following a "0" result.

## ■ CONDITION CODE REGISTER OPERATIONS

The Condition Code Register (CCR) is a 6-bit register within the MPU that is useful in controlling program flow during system operation. The bits are defined in Fig. 29.

The instructions shown in Table 7 are available to the user for direct manipulation of the CCR. In addition, the MPU automatically sets or clears the appropriate status bits as many of the other instructions on the condition code register was indicated as they were introduced.

Systems which require an interrupt window to be opened under program control should use a CLI-NOP-SEI sequence rather than CLI-SEI.

b5	b4	b3	b2	b1	b0
H	I	N	Z	V	C

H = Half-carry; set whenever a carry from b3 to b4 of the result is generated by ADD, ABA, ADC; cleared if no b3 to b4 carry; not affected by other instructions.

I = Interrupt Mask; set by hardware of software interrupt or SEI instruction; cleared by CLI instruction. (Normally not used in arithmetic operations.) Restored to a "0" as a result of an RTI instruction if IM stored on the stacked is "0".

N = Negative; set if high order bit (b7) of result is set; cleared otherwise.

Z = Zero; set if result = "0"; cleared otherwise.

V = Overflow; set if there was arithmetic overflow as a result of the operation; cleared otherwise.

C = Carry; set if there was a carry from the most significant bit (b7) of the result; cleared otherwise.

Figure 29 Condition Code Register Bit Definition

## ■ ADDRESSING MODES

The MPU operates on 8-bit binary numbers presented to it via the Data Bus. A given number (byte) may represent either data or an instruction to be executed, depending on where it is encountered in the control program. The HD6800 MPU has 72 unique instructions, however, it recognizes and takes action on 197 of the 256 possibilities that can occur using an 8-bit word length. This larger number of instructions results from the fact that many of the executive instructions have more than one addressing mode.

Table 7 Condition Code Register Instructions

Operations	Mnemonic	Addressing Mode			Boolean Operation	Cond. Code Reg.						
		IMPLIED				5 4 3 2 1 0						
		OP	~	#		H	I	N	Z	V	C	
Clear Carry	CLC	0C	2	1	0 → C	•	•	•	•	•	R	
Clear Interrupt Mask	CLI	0E	2	1	0 → I	•	R	•	•	•	•	
Clear Overflow	CLV	0A	2	1	0 → V	•	•	•	•	R	•	
Set Carry	SEC	0D	2	1	1 → C	•	•	•	•	•	S	
Set Interrupt Mask	SEI	0F	2	1	1 → I	•	S	•	•	•	•	
Set Overflow	SEV	0B	2	1	1 → V	•	•	•	•	S	•	
Acmtr A → CCR	TAP	06	2	1	A → CCR	①						
CCR → Acmtr A	TPA	07	2	1	CCR → A	•	•	•	•	•	•	

R = Reset

S = Set

• = Not affected

① (ALL) Set according to the contents of Accumulator A.

## HD6800, HD68A00, HD68B00

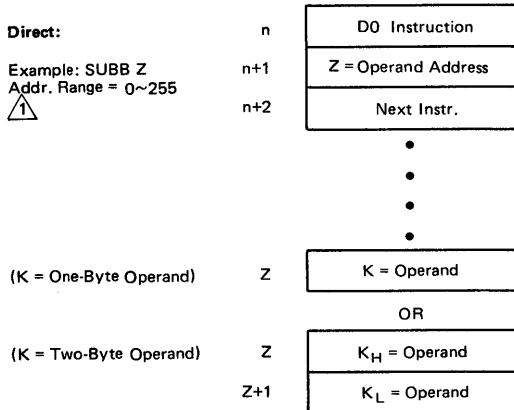
These addressing modes refer to the manner in which the program causes the MPU to obtain its instructions and data. The programmer must have a method for addressing the MPU's internal registers and all of the external memory locations.

Selection of the desired addressing mode is made by the user as the source statements are written. Translation into appropriate opcode then depends on the method used. If manual translation is used, the addressing mode is inherent in the opcode. For example, the Immediate, Direct, Indexed, and Extended modes may all be used with the ADD instruction. The proper mode is determined by selecting (hexadecimal notation) 8B, 9B, AB, or BB, respectively.

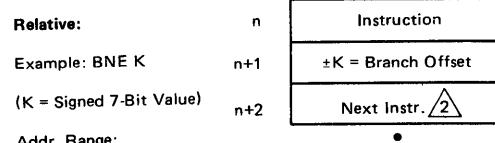
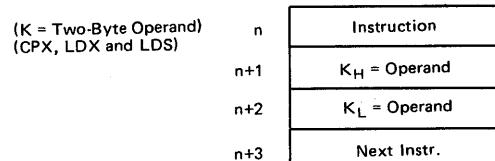
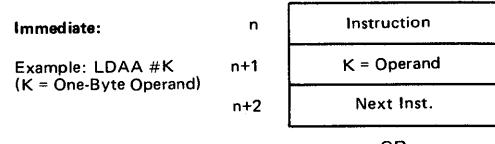
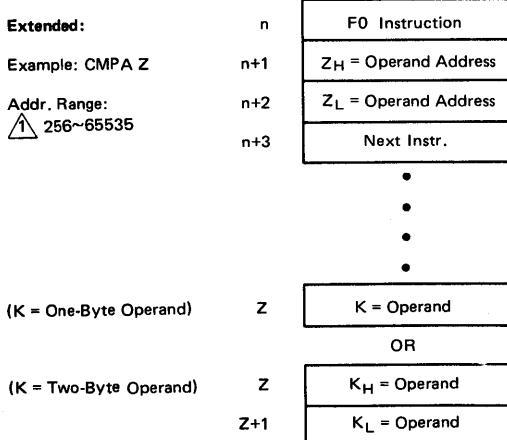
The source statement format includes adequate information for the selection if an assembler program is used to generate the opcode. For instance, the Immediate mode is selected by the

Assembler whenever it encounters the “#” symbol in the operand field. Similarly, an “X” in the operand field causes the Indexed mode to be selected. Only the Relative mode applies to the branch instructions, therefore, the mnemonic instruction itself is enough for the Assembler to determine addressing mode.

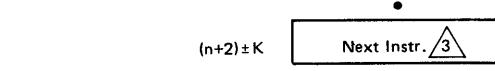
For the instructions that use both Direct and Extended modes, the Assembler selects the Direct mode if the operand value is in the range 0~255 and Extended otherwise. There are a number of instructions for which the Extended mode is valid but the Direct is not. For these instructions, the Assembler automatically selects the Extended mode even if the operand is in the 0~255 range. The addressing modes are summarized in Fig. 30.



 1 If Z ≤ 255, Assembler Select Direct Mode  
If Z > 255, Extended Mode is selected



Addr. Range:  
-125 to +129  
Relative to n.



 2 If Branch Test False,  3 If Branch Test True.

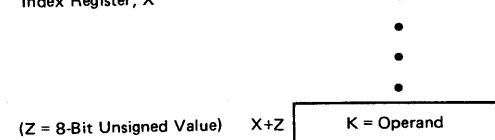
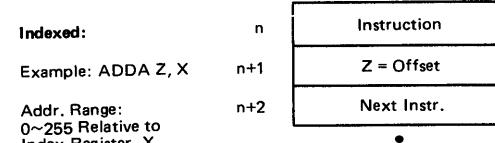


Figure 30 Addressing Mode Summary

- Inherent (Includes "Accumulator Addressing" Mode)**

The successive fields in a statement are normally separated by one or more spaces. An exception to this rule occurs for instructions that use dual addressing in the operand field and for instructions that must distinguish between the two accumulators. In these cases, A and B are "operands" but the space between them and the operator may be omitted. This is commonly done, resulting in apparent four character mnemonics for those instructions.

The addition instruction, ADD, provides an example of dual addressing in the operand fields;

Operator	Operand	Comment
ADDA	MEM12	ADD CONTENTS OF MEM12 TO ACCA
or ADDB	MEM12	ADD CONTENTS OF MEM12 TO ACCB

The example used earlier for the test instruction, TST, also applies to the accumulators and uses the "accumulator addressing mode" to designate which of the two accumulators is being tested:

Operator	Comment
TSTB	TEST CONTENTS OF ACCB
or TSTA	TEST CONTENTS OF ACCA

A number of the instructions either alone or together with an accumulator operand contain all of the address information that is required, that is, "inherent" in the instruction, itself. For instance, the instruction ABA causes the MPU to add the contents of accumulators A and B together and place the result in accumulator A. The instruction INC B, another example of "accumulator addressing", causes the contents of accumulator B to be increased by one. Similarly, INX, increment the Index Register, causes the contents of the Index Register to be increased by one.

Program flow for instructions of this type is illustrated in Figures 31 and 32. In these figures, the general case is shown on the left and a specific example is shown on the right. Numerical examples are in decimal notation. Instructions of this type require only one byte of opcode. Cycle-by-cycle operation of the inherent mode is shown in Table 8.

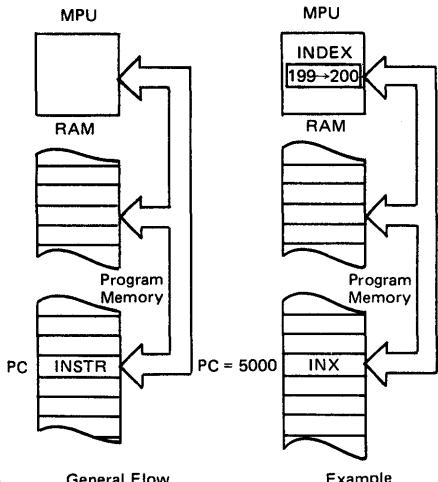


Figure 31 Inherent Addressing

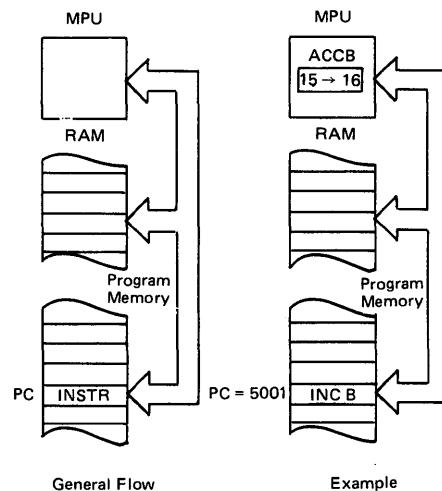


Figure 32 Accumulator Addressing

- Immediate Addressing Mode**

In the Immediate addressing mode, the operand is the value that is to be operated on. For instance, the instruction

Operator	Operand	Comment
LDA A	#25	LOAD 25 INTO ACCA

causes the MPU to "immediately load accumulator A with the value 25"; no further address reference is required. The Immediate mode is selected by preceding the operand value with the "#" symbol. Program flow for this addressing mode is illustrated in Fig. 33.

The operand format allows either properly defined symbols or numerical values. Except for the instructions CPX, LDX, and LDS, the operand may be any value in the range 0 ~ 255. Since Compare Index Register (CPX), Load Index Register (LDX), Load Stack Pointer (LDS), require 16-bit values, the immediate mode for these three instructions require two-byte operands.

Table 9 shows the cycle-by-cycle operation for the immediate addressing mode.

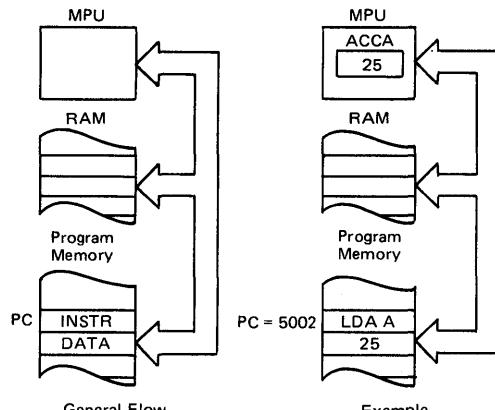


Figure 33 Immediate Addressing Mode

Table 8 Inherent Mode Cycle by Cycle Operation

Address Mode and Instructions	Cycle	Cycle #	VMA Line	Address Bus	R/W Line	Data Bus
ABA DAA SEC ASL DEC SEI ASR INC SEV CBA LSR TAB CLC NEG TAP CLI NOP TBA CLR ROL TPA CLV ROR TST COM SBA	2	1 2	1 1	Op Code Address Op Code Address + 1	1 1	Op Code Op Code of Next Instruction
DES DEX INS INX	4	1 2 3 4	1 1 0 0	Op Code Address Op Code Address + 1 Previous Register Contents New Register Contents	1 1 1 1	Op Code Op Code of Next Instruction Irrelevant Data (NOTE 1) Irrelevant Data (NOTE 1)
PSH	4	1 2 3 4	1 1 1 0	Op Code Address Op Code Address + 1 Stack Pointer Stack Pointer - 1	1 1 0 1	Op Code Op Code of Next Instruction Accumulator Data Accumulator Data
PUL	4	1 2 3 4	1 1 0 1	Op Code Address Op Code Address + 1 Stack Pointer Stack Pointer + 1	1 1 1 1	Op Code Op Code of Next Instruction Irrelevant Data (NOTE 1) Operand Data from Stack
TSX	4	1 2 3 4	1 1 0 0	Op Code Address Op Code Address + 1 Stack Pointer New Index Register	1 1 1 1	Op Code Op Code of Next Instruction Irrelevant Data (NOTE 1) Irrelevant Data (NOTE 1)
TXS	4	1 2 3 4	1 1 0 0	Op Code Address Op Code Address + 1 Index Register New Stack Pointer	1 1 1 1	Op Code Op Code of Next Instruction Irrelevant Data Irrelevant Data
RTS	5	1 2 3 4 5	1 1 0 1 1	Op Code Address Op Code Address + 1 Stack Pointer Stack Pointer + 1 Stack Pointer + 2	1 1 1 1 1	Op Code Irrelevant Data (NOTE 2) Irrelevant Data (NOTE 1) Address of Next Instruction (High Order Byte) Address of Next Instruction (Low Order Byte)
WAI	9	1 2 3 4 5 6 7 8 9	1 1 1 1 1 1 1 1 1	Op Code Address Op Code Address + 1 Stack Pointer Stack Pointer - 1 Stack Pointer - 2 Stack Pointer - 3 Stack Pointer - 4 Stack Pointer - 5 Stack Pointer - 6 (NOTE 3)	1 1 0 0 0 0 0 0 1	Op Code Op Code of Next Instruction Return Address (Low Order Byte) Return Address (High Order Byte) Index Register (Low Order Byte) Index Register (High Order Byte) Contents of Accumulator A Contents of Accumulator B Contents of Cond. Code Register
RTI	10	1 2 3 4 5 6 7 8 9 10	1 1 0 1 1 1 1 1 1 1	Op Code Address Op Code Address + 1 Stack Pointer Stack Pointer + 1 Stack Pointer + 2 Stack Pointer + 3 Stack Pointer + 4 Stack Pointer + 5 Stack Pointer + 6 Stack Pointer + 7	1 1 1 1 1 1 1 1 1 1	Op Code Irrelevant Data (NOTE 2) Irrelevant Data (NOTE 1) Contents of Cond. Code Register from Stack Contents of Accumulator B from Stack Contents of Accumulator A from Stack Index Register from Stack (High Order Byte) Index Register from Stack (Low Order Byte) Next Instruction Address from Stack (High Order Byte) Next Instruction Address from Stack (Low Order Byte)
SWI	12	1 2 3 4 5 6 7 8 9 10 11 12	1 1 1 1 1 1 1 1 1 0 1 1	Op Code Address Op Code Address + 1 Stack Pointer Stack Pointer - 1 Stack Pointer - 2 Stack Pointer - 3 Stack Pointer - 4 Stack Pointer - 5 Stack Pointer - 6 Stack Pointer - 7 Vector Address FFFA (Hex) Vector Address FFFB (Hex)	1 1 0 0 0 0 0 0 0 1 1 1	Op Code Irrelevant Data (NOTE 1) Return Address (Low Order Byte) Return Address (High Order Byte) Index Register (Low Order Byte) Index Register (High Order Byte) Contents of Accumulator A Contents of Accumulator B Contents of Cond. Code Register Irrelevant Data (NOTE 1) Address of Subroutine (High Order Byte) Address of Subroutine (Low Order Byte)

NOTE 1. If device which is addressed during this cycle uses VMA, then the Data Bus will go to the high impedance three-state condition.  
 Depending on bus capacitance, data from the previous cycle may be retained on the Data Bus.

NOTE 2. Data is ignored by the MPU.

NOTE 3. While the MPU is waiting for the interrupt, Bus Available will go "High" indicating the following states of the control lines: VMA is "Low"; Address Bus, R/W, and Data Bus are all in the high impedance state.

Table 9 Immediate Mode Cycle by Cycle Operation

Address Mode and Instructions	Cycle	Cycle #	VMA Line	Address Bus	R/W Line	Data Bus
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	2	1 2	1 1	Op Code Address Op Code Address + 1	1 1	Op Code Operand Data
CPX LDS LDX	3	1 2 3	1 1 1	Op Code Address Op Code Address + 1 Op Code Address + 2	1 1 1	Op Code Operand Data (High Order Byte) Operand Data (Low Order Byte)

- **Direct and Extended Addressing Modes**

In the Direct and Extended modes of addressing, the operand field of the source statement is the address of the value that is to be operated on. The Direct and Extended modes differ only in the range of memory locations to which they can direct the MPU. Direct addressing generates a single 8-bit operand and, hence, can address only memory locations 0 ~ 255; a two byte operand is generated for Extended addressing, enabling the MPU to reach the remaining memory locations, 256 ~ 65535. An example of Direct addressing and its effect on program flow is illustrated in Fig. 34.

Table 10 shows the cycle-by-cycle operations of this mode.

The MPU, after encountering the opcode for the instruction LDAA (Direct) at memory location 5004 (Program Counter = 5004), looks in the next location, 5005, for the address of the operand. It then sets the program counter equal to the value found there (100 in the example) and fetches the operand, in

this case a value to be loaded into accumulator A, from that location. For instructions requiring a two-byte operand such as LDX (Load the Index Register), the operand bytes would be retrieved from locations 100 and 101.

Extended addressing, Fig. 35, is similar except that a two-byte address is obtained from locations 5007 and 5008 after the LDAB (Extended) opcode shows up in location 5006. Extended addressing can be thought of as the "standard" addressing mode, that is, it is a method of reaching anyplace in memory. Direct addressing, since only one address byte is required, provides a faster method of processing data and generates fewer bytes of control code. In most applications, the direct addressing range, memory locations 0 ~ 255, are reserved for RAM. They are used for data buffering and temporary storage of system variables, the area in which faster addressing is of most value. Cycle-by-cycle operation is shown in Table 11 for Extended Addressing.

Table 10 Direct Mode Cycle by Cycle Operation

Address Mode and Instructions	Cycle	Cycle #	VMA Line	Address Bus	R/W Line	Data Bus
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	3	1 2 3	1 1 1	Op Code Address Op Code Address + 1 Address of Operand	1 1 1	Op Code Address of Operand Operand Data
CPX LDS LDX	4	1 2 3 4	1 1 1 1	Op Code Address Op Code Address + 1 Address of Operand Operand Address + 1	1 1 1 1	Op Code Address of Operand Operand Data (High Order Byte) Operand Data (Low Order Byte)
STA	4	1 2 3 4	1 1 0 1	Op Code Address Op Code Address + 1 Destination Address Destination Address	1 1 1 0	Op Code Destination Address Irrelevant Data (NOTE 1) Data from Accumulator
STS STX	5	1 2 3 4 5	1 1 0 1 1	Op Code Address Op Code Address + 1 Address of Operand Address of Operand Address of Operand + 1	1 1 1 0 0	Op Code Address of Operand Irrelevant Data (NOTE 1) Register Data (High Order Byte) Register Data (Low Order Byte)

NOTE 1. If device which is address during this cycle uses VMA, then the Data Bus will go to the high impedance three-state condition. Depending on bus capacitance, data from the previous cycle may be retained on the Data Bus.

Table 11 Extended Mode Cycle by Cycle

Address Mode and Instructions	Cycle	Cycle #	VMA Line	Address Bus	R/W Line	Data Bus
STS STX	6	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand (High Order Byte)
		3	1	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	0	Address of Operand	1	Irrelevant Data (NOTE 1)
		5	1	Address of Operand	0	Operand Data (High Order Byte)
		6	1	Address of Operand + 1	0	Operand Data (Low Order Byte)
JSR	9	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Subroutine (High Order Byte)
		3	1	Op Code Address + 2	1	Address of Subroutine (Low Order Byte)
		4	1	Subroutine Starting Address	1	Op Code of Next Instruction
		5	1	Stack Pointer	0	Return Address (Low Order Byte)
		6	1	Stack Pointer - 1	0	Return Address (High Order Byte)
		7	0	Stack Pointer - 2	1	Irrelevant Data (NOTE 1)
		8	0	Op Code Address + 2	1	Irrelevant Data (NOTE 1)
		9	1	Op Code Address + 2	1	Address of Subroutine (Low Order Byte)
JMP	3	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Jump Address (High Order Byte)
		3	1	Op Code Address + 2	1	Jump Address (Low Order Byte)
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand (High Order Byte)
		3	1	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	1	Address of Operand	1	Operand Data
CPX LDS LDX	5	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand (High Order Byte)
		3	1	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	1	Address of Operand	1	Operand Data (High Order Byte)
		5	1	Address of Operand + 1	1	Operand Data (Low Order Byte)
STA A STA B	5	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Destination Address (High Order Byte)
		3	1	Op Code Address + 2	1	Destination Address (Low Order Byte)
		4	0	Operand Destination Address	1	Irrelevant Data (NOTE 1)
		5	1	Operand Destination Address	0	Data from Accumulator
ASL LSR ASR NEG CLR ROL COM ROR DEC TST INC	6	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand (High Order Byte)
		3	1	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	1	Address of Operand	1	Current Operand Data
		5	0	Address of Operand	1	Irrelevant Data (NOTE 1)
		6	1/0 (NOTE 2)	Address of Operand	0	New Operand Data (NOTE 2)

NOTE 1. If device which is addressed during this cycle uses VMA, then the Data Bus will go to the high impedance three-state condition. Depending on bus capacitance, data from the previous cycle may be retained on the Data Bus.

NOTE 2. For TST, VMA = 0 and Operand data does not change.

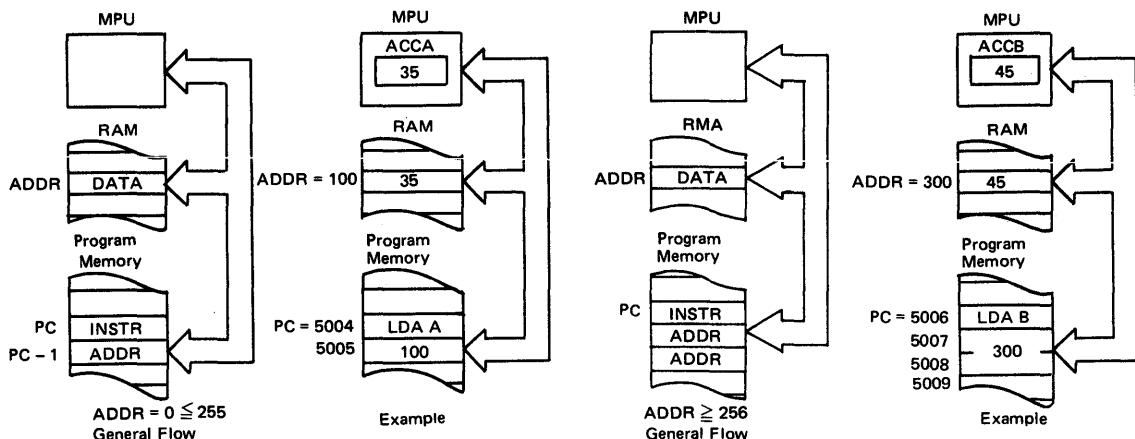


Figure 34 Direct Addressing Mode

Figure 35 Extended Addressing Mode

- **Relative Address Mode**

In both the Direct and Extended modes, the address obtained by the MPU is an absolute numerical address. The Relative addressing mode, implemented for the MPU's branch instructions, specifies a memory location relative to the Program Counter's current location. Branch instructions generate two bytes of machine code, one for the instruction opcode and one for the "relative" address (see Fig. 36). Since it is desirable to be able to branch in either direction, the 8-bit address byte is interpreted as a signed 7-bit value; the 8th bit of the operand is treated as a sign bit, "0" = plus and "1" = minus. The remaining seven bits represent the numerical value. This result in a relative addressing range of  $\pm 127$  with respect to the location of the branch instruction itself. However, the branch range is computed with respect to the next instruction that would be executed if the branch conditions are not satisfied. Since two bytes are generated, the next instruction is located at PC+2. If, D is defined as the address of the branch destination, the range is then;

$$\begin{aligned} &(PC+2) - 128 \leq D \leq (PC+2) + 127 \\ \text{or } &PC - 126 \leq D \leq PC + 129 \end{aligned}$$

that is, the destination of the branch instruction must be within -126 to +129 memory locations of the branch instruction itself. For transferring control beyond this range, the unconditional jump (JMP), jump to subroutine (JSR), and return from subroutine (RTS) are used.

In Fig. 36, when the MPU encounters the opcode for BEQ (Branch if result of last instruction was zero), it tests the Zero bit in the Condition Code Register. If that bit is "0", indicating a non-zero result, the MPU continues execution with the next instruction (in location 5010 in Fig. 36). If the previous result was zero, the branch condition is satisfied and the MPU adds the offset, 15 in this case, to PC+2 and branches to location 5025 for the next instruction.

The branch instructions allow the programmer to efficiently direct the MPU to one point or another in the control program depending on the outcome of test results. Since the control program is normally in read-only memory and cannot be changed, the relative address used in execution of branch instructions is a constant numerical value. Cycle-by-cycle operation is shown in Table 12 for relative addressing.

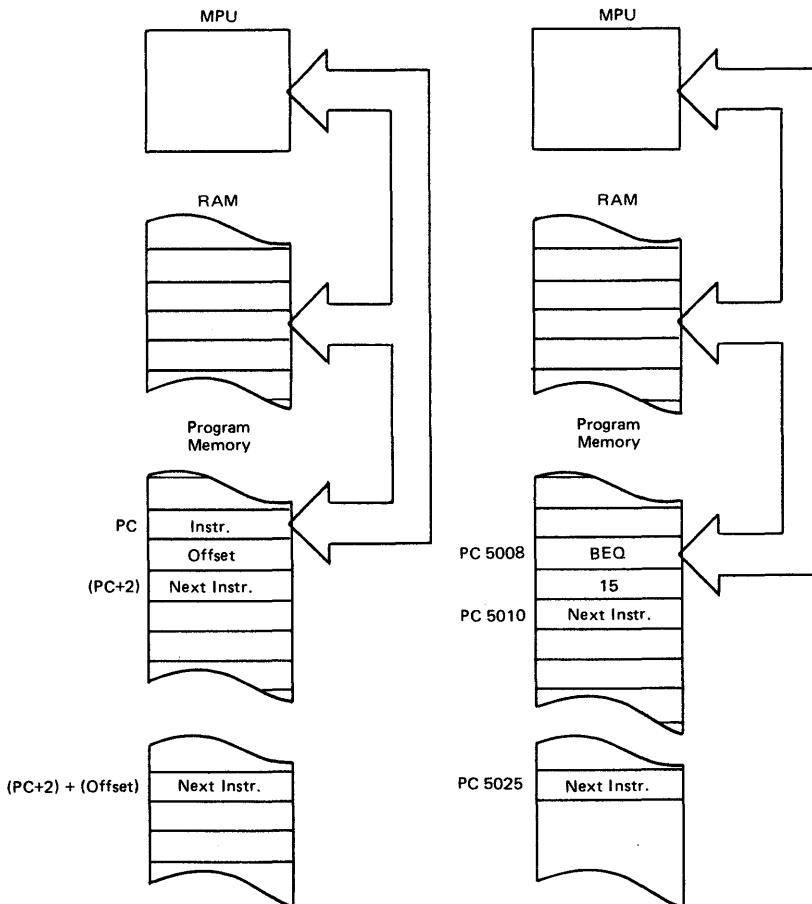


Figure 36 Relative Addressing Mode

Table 12 Relative Mode Cycle-by-Cycle Operation

Address Mode and Instructions	Cycle	Cycle #	VMA Line	Address Bus	R/W Line	Data Bus
BCC BHI BNE		1	1	Op Code Address	1	Op Code
BCS BLE BPL	4	2	1	Op Code Address + 1	1	Branch Offset
BEQ BLS BRA		3	0	Op Code Address + 2	1	Irrelevant Data (NOTE 1)
BGE BLT BVC		4	0	Branch Address	1	Irrelevant Data (NOTE 1)
BGT BMI BVS						
BSR		1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Branch Offset
		3	0	Return Address of Main Program	1	Irrelevant Data (NOTE 1)
	8	4	1	Stack Pointer	0	Return Address (Low Order Byte)
		5	1	Stack Pointer - 1	0	Return Address (High Order Byte)
		6	0	Stack Pointer - 2	1	Irrelevant Data (NOTE 1)
		7	0	Return Address of Main Program	1	Irrelevant Data (NOTE 1)
		8	0	Subroutine Address	1	Irrelevant Data (NOTE 1)

NOTE 1. If device which is addressed during this cycle uses VMA, then the Data Bus will go to the high impedance three-state condition. Depending on bus capacitance, data from the previous cycle may be retained on the Data Bus.

#### • Indexed Addressing Mode

With Indexed addressing the numerical address is variable and depend on the current contents of the Index Register. A source statement such as

Operator	Operand	Comment
STAA	X	PUT A IN INDEXED LOCATION

causes the MPU to store the contents of accumulator A in the memory location specified by the contents of the Index Register (recall that the label X is reserved to designate the Index Register). Since there are instructions for manipulating X during program execution (LDX, INX, DEX, etc.), the Indexed addressing mode provides a dynamic "on the fly" way to modify program activity.

The operand field can also contain a numerical value that will be automatically added to X during execution. This format is illustrated in Fig. 37.

When the MPU encounters the LDAB (Indexed) opcode in location 5006, it looks in the next memory location for the value to be added to X (5 in the example) and calculates the required address by adding 5 to the present Index Register value of 400. In the operand format, the offset may be represented by a label or a numerical value in the range 0 ~ 255 as in the example. In the earlier example, STAA X, the operand is equivalent to 0, X , that is, the "0" may be omitted when the desired address is equal to X. Table 13 shows the cycle-by-cycle operation for the Indexed Mode of Addressing.

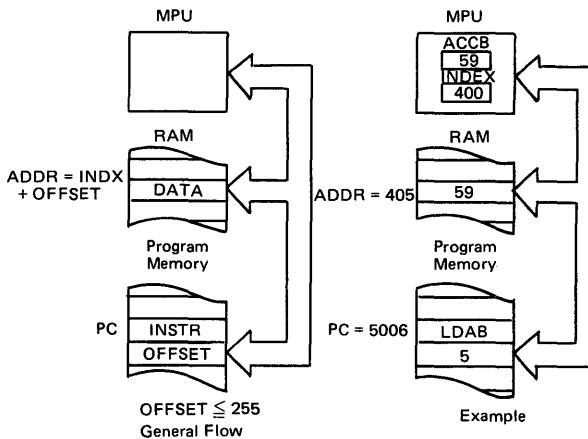


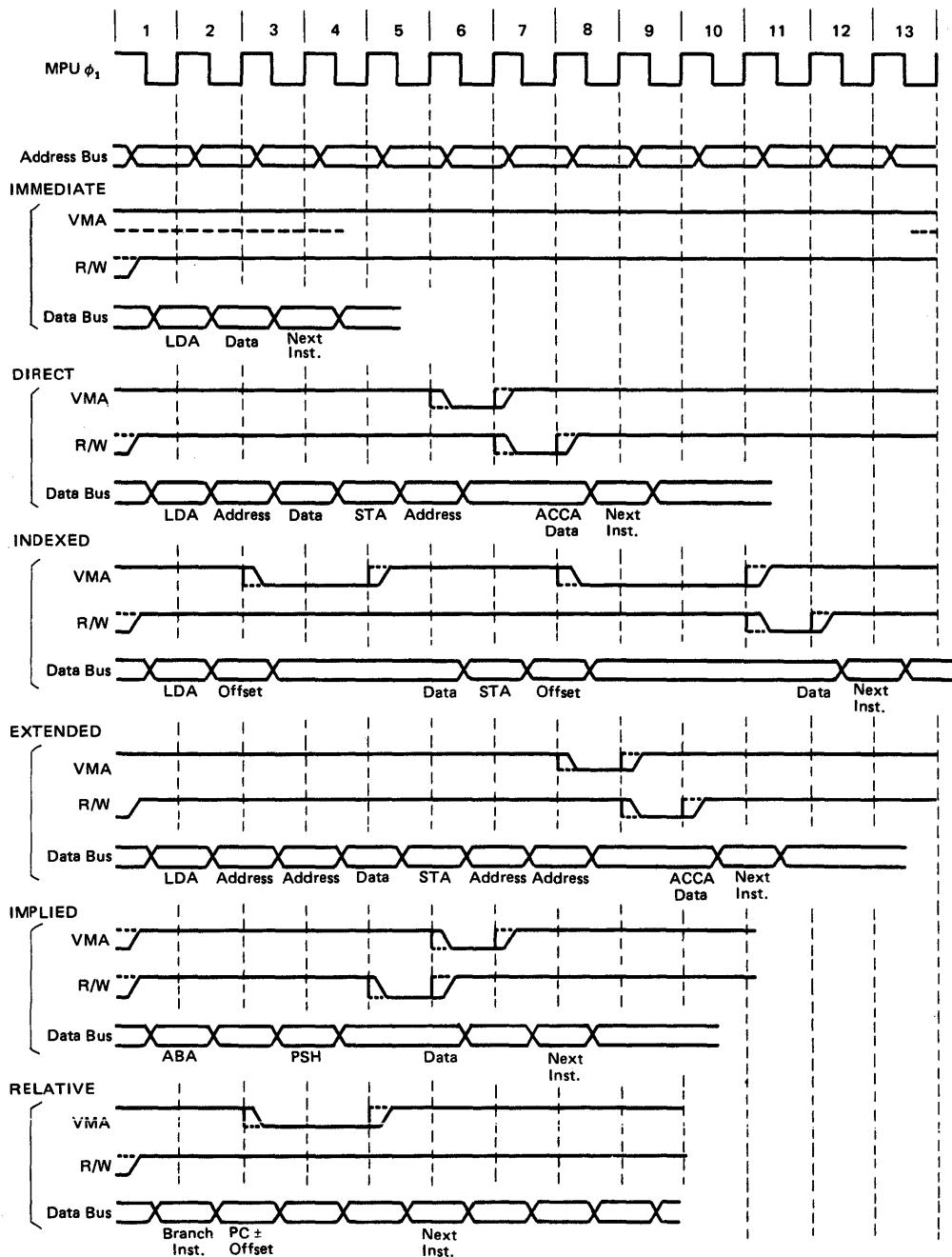
Figure 37 Indexed Addressing Mode

Table 13 Indexed Mode Cycle by Cycle

Address Mode and Instructions		Cycle	Cycle #	VMA Line	Address Bus	R/W Line	Data Bus
JMP		4	1	1	Op Code Address	1	Op Code
			2	1	Op Code Address + 1	1	Offset
			3	0	Index Register	1	Irrelevant Data (NOTE 1)
			4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (NOTE 1)
ADC	EOR	5	1	1	Op Code Address	1	Op Code
ADD	LDA		2	1	Op Code Address + 1	1	Offset
AND	ORA		3	0	Index Register	1	Irrelevant Data (NOTE 1)
BIT	SBC		4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (NOTE 1)
CMP	SUB		5	1	Index Register Plus Offset	1	Operand Data
CPX		6	1	1	Op Code Address	1	Op Code
LDS			2	1	Op Code Address + 1	1	Offset
LDX			3	0	Index Register	1	Irrelevant Data (NOTE 1)
			4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (NOTE 1)
			5	1	Index Register Plus Offset	1	Operand Data (High Order Byte)
			6	1	Index Register Plus Offset + 1	1	Operand Data (Low Order Byte)
STA		6	1	1	Op Code Address	1	Op Code
			2	1	Op Code Address + 1	1	Offset
			3	0	Index Register	1	Irrelevant Data (NOTE 1)
			4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (NOTE 1)
			5	0	Index Register Plus Offset	1	Irrelevant Data (NOTE 1)
			6	1	Index Register Plus Offset	0	Operand Data
ASL	LSR	7	1	1	Op Code Address	1	Op Code
ASR	NEG		2	1	Op Code Address + 1	1	Offset
CLR	ROL		3	0	Index Register	1	Irrelevant Data (NOTE 1)
COM	ROR		4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (NOTE 1)
DEC	TST		5	1	Index Register Plus Offset	1	Current Operand Data
INC			6	0	Index Register Plus Offset	1	Irrelevant Data (NOTE 1)
			7	1/0 (NOTE 2)	Index Register Plus Offset	0	New Operand Data (NOTE 2)
STS			1	1	Op Code Address	1	Op Code
STX		7	2	1	Op Code Address + 1	1	Offset
			3	0	Index Register	1	Irrelevant Data (NOTE 1)
			4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (NOTE 1)
			5	0	Index Register Plus Offset	1	Irrelevant Data (NOTE 1)
			6	1	Index Register Plus Offset	0	Operand Data (High Order Byte)
			7	1	Index Register Plus Offset + 1	0	Operand Data (Low Order Byte)
JSR		8	1	1	Op Code Address	1	Op Code
			2	1	Op Code Address + 1	1	Offset
			3	0	Index Register	1	Irrelevant Data (NOTE 1)
			4	1	Stack Pointer	0	Return Address (Low Order Byte)
			5	1	Stack Pointer - 1	0	Return Address (High Order Byte)
			6	0	Stack Pointer - 2	1	Irrelevant Data (NOTE 1)
			7	0	Index Register	1	Irrelevant Data (NOTE 1)
			8	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (NOTE 1)

NOTE 1. If Device which is addressed during this cycle uses VMA, then the Data Bus will go to the high impedance three-state condition.  
Depending on bus capacitance, data from the previous cycle may be retained on the Data Bus.

NOTE 2. For TST, VMA = 0 and Operand data does not change.



**Figure 38 Example of Execution Timing in Each Addressing Mode**

# HD6802

## MPU (Microprocessor with Clock and RAM)

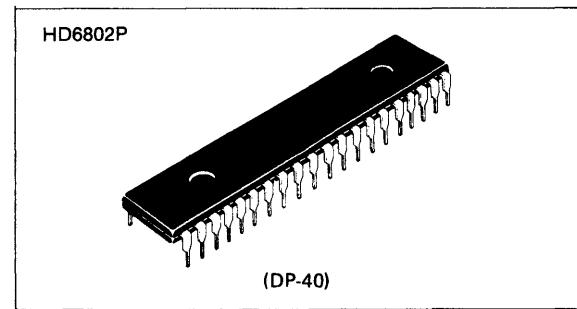
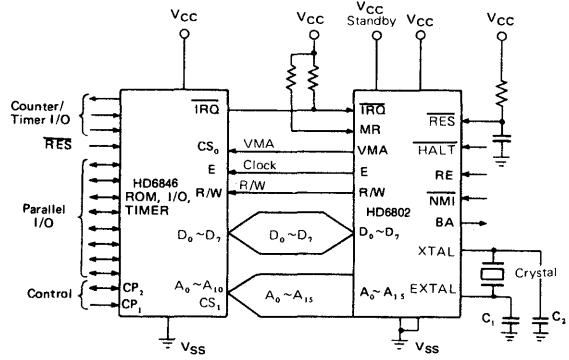
The HD6802 is a monolithic 8-bit microprocessor that contains all the registers and accumulators of the present HD6800 plus an internal clock oscillator and driver on the same chip. In addition, the HD6802 has 128 bytes of RAM on the chip located at hex addresses 0000 to 007F. The first 32 bytes of RAM, at hex addresses 0000 to 001F, may be retained in a low power mode by utilizing  $V_{CC}$  standby, thus facilitating memory retention during a power-down situation.

The HD6802 is completely software compatible with the HD6800 as well as the entire HMCS6800 family of parts. Hence, the HD6802 is expandable to 65K words.

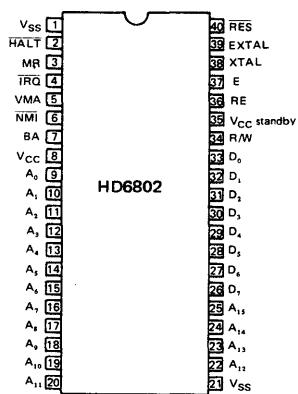
### ■ FEATURES

- On-Chip Clock Circuit
- 128  $\times$  8 Bit On-Chip RAM
- 32 Bytes of RAM are Retainable
- Software-Compatible with the HD6800
- Expandable to 65K words
- Standard TTL-Compatible Inputs and Outputs
- 8 Bit Word Size
- 16 Bit Memory Addressing
- Interrupt Capability
- Compatible with MC6802

### ■ MINIMUM SYSTEM



### ■ PIN ARRANGEMENT



(Top View)

### ■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	$V_{CC}^*$ $V_{CC}$ Standby*	-0.3 ~ +7.0	V
Input Voltage	$V_{in}^*$	-0.3 ~ +7.0	V
Operating Temperature	$T_{opr}$	-20 ~ +75	°C
Storage Temperature	$T_{stg}$	-55 ~ +150	°C

\* With respect to  $V_{SS}$  (SYSTEM GND)

(NOTE) 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.

### ■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol	min	typ	max	Unit
Supply Voltage	$V_{CC}^*$ $V_{CC}$ Standby*	4.75	5	5.25	V
Input Voltage	$V_{IL}^*$	-0.3	—	0.8	V
	$V_{IH}^*$	2.0	—	$V_{CC}$	V
Operation Temperature	$T_{opr}$	-20	25	75	°C

\* With respect to  $V_{SS}$  (SYSTEM GND)

### ■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS ( $V_{CC}=5.0V\pm5%$ ,  $V_{CC}$  Standby=5.0V±5%,  $V_{SS}=0V$ ,  $T_a=-20\sim+75^\circ C$ , unless otherwise noted.)

Item	Symbol	Test Condition	min	typ**	max	Unit	
Input "High" Voltage	Except $\overline{RES}$	$V_{IH}$	2.0	—	$V_{CC}$	V	
	$\overline{RES}$						
Input "Low" Voltage	Except $\overline{RES}$	*** $V_{IL}$	-0.3	—	0.8	V	
	$\overline{RES}$						
Output "High" Voltage	$D_0\sim D_7$ , E	$V_{OH}$	$I_{OH} = -205\mu A$ ,	2.4	—	V	
	$A_0\sim A_{15}$ , R/W, VMA		$I_{OH} = -145\mu A$ ,	2.4	—		
	BA		$I_{OH} = -100\mu A$	2.4	—		
Output "Low" Voltage	$V_{OL}$	$I_{OL} = 1.6mA$	—	—	0.4	V	
Three State (Off State) Input Current	$D_0\sim D_7$	$I_{TSI}$	$V_{in} = 0.4\sim 2.4V$	—	2.0	$\mu A$	
Input Leakage Current	Except $D_0\sim D_7$	$I_{in}^{****}$	$V_{in} = 0\sim 5.25V$	—	1.0	$\mu A$	
Power Dissipation	$P_D^*$			—	0.6	W	
Input Capacitance	$D_0\sim D_7$	$C_{in}$	$V_{in}=0V$ , $T_a=25^\circ C$ $f=1.0MHz$	—	10	12.5	pF
	Except $D_0\sim D_7$			—	6.5	10	
Output Capacitance	$A_0\sim A_{15}$ , R/W, BA VMA	$C_{out}$	$V_{in}=0V$ , $T_a=25^\circ C$ $f=1.0MHz$	—	—	12	pF

\* In power-down mode, maximum power dissipation is less than 42mW.

\*\*  $T_a=25^\circ C$ ,  $V_{CC}=5V$

\*\*\* As  $\overline{RES}$  input has hysteresis character, applied voltage up to 2.4V is regarded as "Low" level when it goes up from 0V.

\*\*\*\* Does not include Extal and Xtal, which are crystal inputs.

- AC CHARACTERISTICS ( $V_{CC}=5.0V \pm 5\%$ ,  $V_{CC\text{ Standby}}=5.0V \pm 5\%$ ,  $V_{SS}=0V$ ,  $T_a=-20\sim+75^\circ C$ , unless otherwise noted.)

### 1. CLOCK TIMING CHARACTERISTICS

Item	Symbol	Test Condition	min	typ	max	Unit
Frequency of Operation	Input Clock $\div 4$	f		0.1	—	1.0
	Crystal Frequency	$f_{XTAL}$		1.0	—	4.0
Cycle Time	$t_{cyc}$		1.0	—	10	$\mu s$
Clock Pulse Width	“High” Level	$PW_{\phi H}$	at 2.4V	450	—	4500
	“Low” Level	$PW_{\phi L}$	at 0.8V			
Clock Fall Time	$t_\phi$	0.8V – 2.4V	—	—	25	ns

### 2. READ/WRITE TIMING

Item	Symbol	Test Condition	min	typ	max	Unit
Address Delay	$t_{AD}$	Fig. 2, Fig. 3	—	—	270	ns
Peripheral Read Access Time	$t_{acc}$	Fig. 2	—	—	530	ns
Data Setup Time (Read)	$t_{DSR}$	Fig. 2	100	—	—	ns
Input Data Hold Time	$t_H$	Fig. 2	10	—	—	ns
Output Data Hold Time	$t_H$	Fig. 3	20	—	—	ns
Address Hold Time (Address, R/W, VMA)	$t_{AH}$	Fig. 2, Fig. 3	10	—	—	ns
Data Delay Time (Write)	$t_{DDW}$	Fig. 3	—	165	225	ns
Processor Controls	$t_{PCS}$	Fig. 7	200	—	—	ns
Processor Control Setup Time	$t_{PCr}$	Fig. 7, Fig. 8, Fig. 11	—	—	100	ns
Processor Control Rise and Fall Time (Measured at 0.8V and 2.0V)	$t_{PCf}$					

### 3. POWER DOWN SEQUENCE TIMING, POWER UP RESET TIMING AND MEMORY READY TIMING

Item	Symbol	Test Condition	min	typ	max	Unit
RAM Enable Reset Time (1)	$t_{RE1}$	Fig. 8	150	—	—	ns
RAM Enable Reset Time (2)	$t_{RE2}$	Fig. 8	E-3 cycles	—	—	
Reset Release Time	$t_{LRES}$	Fig. 7	20*	—	—	ms
RAM Enable Reset Time (3)	$t_{RE3}$	Fig. 7	0	—	—	ns
Memory Ready Setup Time	$t_{SMR}$	Fig. 11	300	—	—	ns
Memory Ready Hold Time	$t_{HMR}$	Fig. 11	0	—	200	ns

\* $t_{LRES} = 20$  msec min. for S type, 50 msec min. for R type.

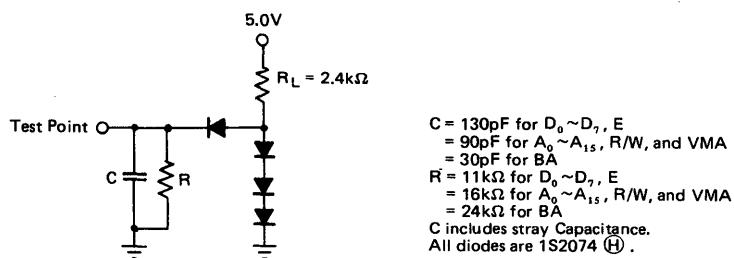
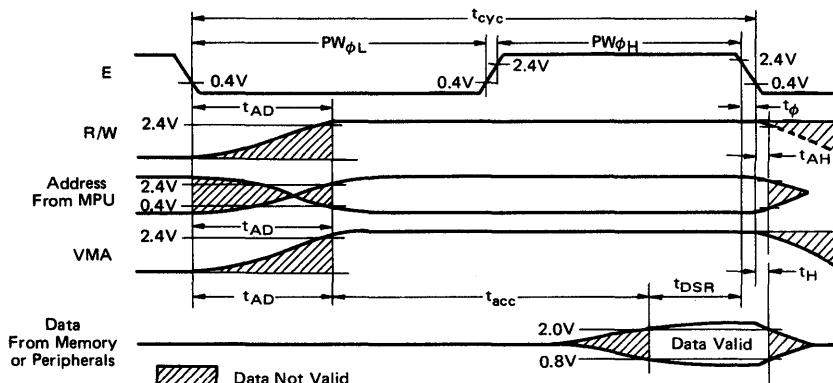
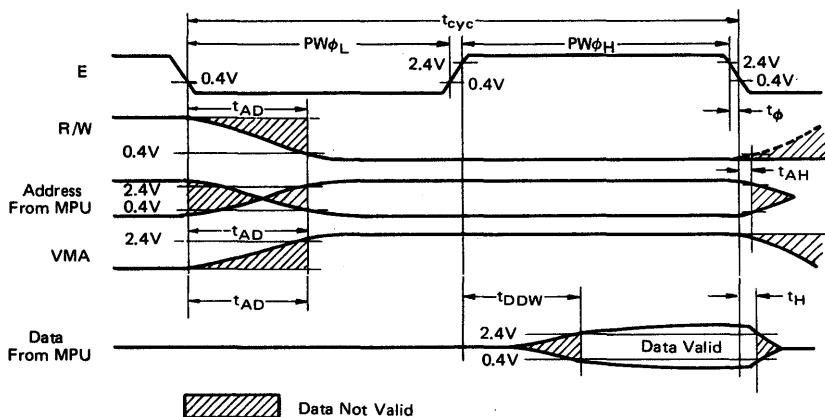


Figure 1 Bus Timing Test Load



**Figure 2** Read Data from Memory or Peripherals



**Figure 3 Write Data in Memory or Peripherals**

## ■ MPU REGISTERS

A general block diagram of the HD6802 is shown in Fig. 4. As shown, the number and configuration of the registers are the same as for the HD6800. The  $128 \times 8$  bit RAM has been added to the basic MPU. The first 32 bytes may be operated in a low power mode via a  $V_{CC}$  standby. These 32 bytes can be retained during power-up and power-down conditions via the RE signal.

The MPU has three 16-bit registers and three 8-bit registers available for use by the programmer (Fig. 5).

- **Program Counter (PC)**

The program counter is a two byte (16-bit) register that points to the current program address.

- **Stack Pointer (SP)**

The stack pointer is a two byte (16-bit) register that contains the address of the next available location in an external push-down/pop-up stack. This stack is normally a random access Read/Write memory that may have any location (address) that is convenient. In those applications that require storage of information in the stack when power is lost, the stack must be non-volatile.

- **Index Register (IX)**

The index register is a two byte register that is used to store data or a sixteen bit memory address for the Indexed mode of memory addressing.

- **Accumulators (ACCA, ACCB)**

The MPU contains two 8-bit accumulators that are used to hold operands and results from an arithmetic logic unit(ALU).

- **Condition Code Register (CCR)**

The condition code register indicates the results of an Arithmetic Logic Unit operation: Negative(N), Zero(Z), Overflow(V), Carry from bit7(C), and half carry from bit3(H). These bits of the Condition Code Register are used as testable conditions for the conditional branch instructions. Bit 4 is the interrupt mask bit(I). The used bits of the Condition Code Register (B6 and B7) are ones.

Fig. 6 shows the order of saving the microprocessor status within the stack.

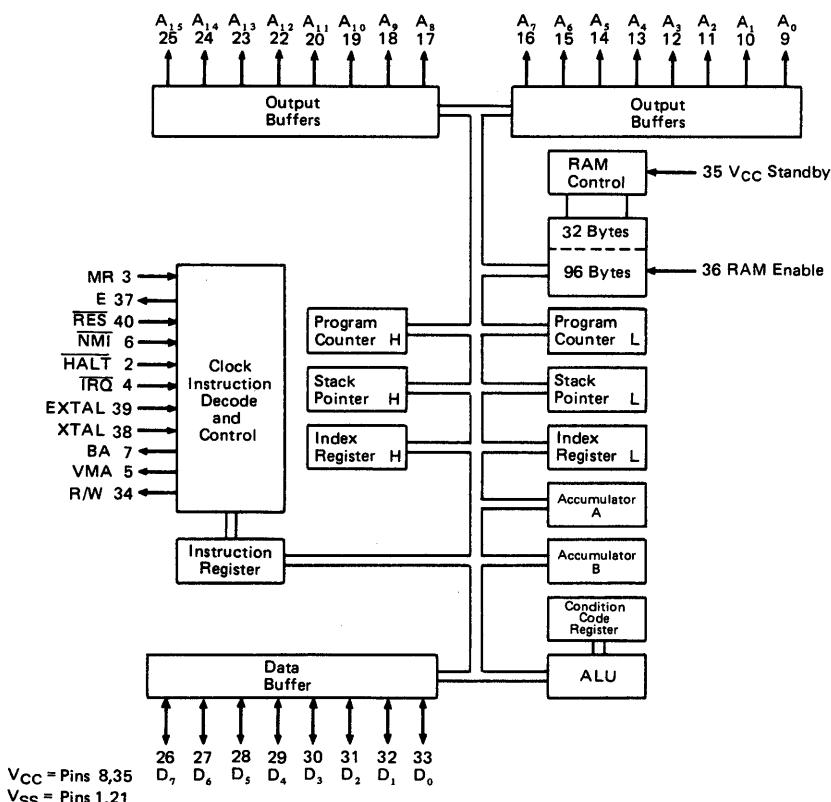


Figure 4 Expanded Block Diagram

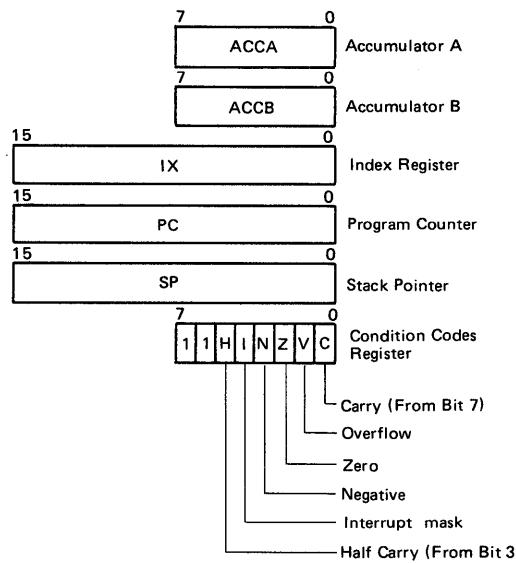


Figure 5 Programming Model of The Microprocessing Unit

SP = Stack Pointer  
 CC = Condition Codes (Also called the Processor Status Byte)  
 ACCB = Accumulator B  
 ACCA = Accumulator A  
 IXH = Index Register, Higher Order 8 Bits  
 IXL = Index Register, Lower Order 8 Bits  
 PCH = Program Counter, Higher Order 8 Bits  
 PCL = Program Counter, Lower Order 8 Bits

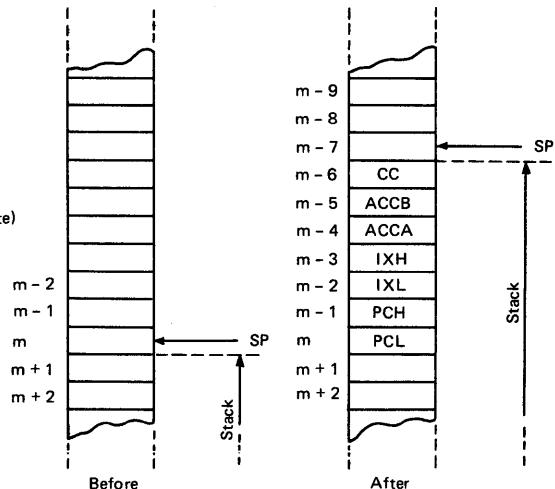


Figure 6 Saving The Status of The Microprocessor in The Stack

## ■ HD6802 MPU SIGNAL DESCRIPTION

Proper operation of the MPU requires that certain control and timing signals be provided to accomplish specific functions and that other signal lines be monitored to determine the state of the processor. These control and timing signals for the HD6802 are similar to those of the HD6800 except that TSC, DBE,  $\phi_1$ ,  $\phi_2$  input, and two unused pins have been eliminated, and the following signal and timing lines have been added.

### RAM Enable (RE)

Crystal Connections EXTAL and XTAL

### Memory Ready(MR)

$V_{CC}$  Standby

### Enable $\phi_2$ Output(E)

The following is a summary of the HD6802 MPU signals:

#### ● Address Bus ( $A_0 \sim A_{15}$ )

Sixteen pins are used for the address bus. The outputs are capable of driving one standard TTL load and 90pF.

#### ● Data Bus ( $D_0 \sim D_7$ )

Eight pins are used for the data bus. It is bidirectional, transferring data to and from the memory and peripheral devices. It also has three-state output buffers capable of driving one standard TTL load and 130pF.

Data Bus will be in the output mode when the internal RAM is accessed. This prohibits external data entering the MPU. It should be noted that the internal RAM is fully decoded from \$0000 to \$007F. External RAM at \$0000 to \$007F must be disabled when internal RAM is accessed.

#### ● HALT

When this input is in the "Low" state, all activity in the machine will be halted. This input is level sensitive.

In the halt mode, the machine will stop at the end of an instruction. Bus Available will be at a "High" state. Valid Memory Address will be at a "Low" state. The address bus will display the address of the next instruction.

To insure single instruction operation, transition of the HALT line must not occur during the last 250ns of E and the HALT line must go "High" for one Clock cycle.

HALT should be tied "High" if not used. This is good engineering design practice in general and necessary to insure proper operation of the part.

#### ● Read/Write (R/W)

This TTL compatible output signals the peripherals and memory devices whether the MPU is in a Read ("High") or Write ("Low") state. The normal standby state of this signal is Read ("High"). When the processor is halted, it will be in the logical one state.

This output is capable of driving one standard TTL load and 90pF.

#### ● Valid Memory Address (VMA)

This output indicates to peripheral devices that there is a valid address on the address bus. In normal operation, this signal should be utilized for enabling peripheral interfaces such as the PIA and ACIA. This signal is not three-state. One standard TTL load and 90pF may be directly driven by this active high signal.

#### ● Bus Available (BA)

The Bus Available signal will normally be in the "Low" state. When activated, it will go to the "High" state indicating that the microprocessor has stopped and that the address bus is available (but not in a three-state condition). This will occur if the HALT line is in the "Low" state or the processor is in the wait state as a result of the execution of a WAI instruction. At such time, all three-state output drivers will go to their off state and other

outputs to their normally inactive level.

The processor is removed from the wait state by the occurrence of a maskable (mask bit I=0) or nonmaskable interrupt. This output is capable of driving one standard TTL load and 30pF.

#### ● Interrupt Request (IRQ)

This level sensitive input requests that an interrupt sequence be generated within the machine. The processor will wait, until it completes the current instruction that is being executed before it recognizes the request. At that time, if the interrupt mask bit in the Condition Code Register is not set, the machine will begin an interrupt sequence. The Index Register, Program Counter, Accumulators, and Condition Code Register are stored away on the stack. Next the MPU will respond to the interrupt request by setting the interrupt mask bit high so that no further interrupts may occur. At the end of the cycle, a 16-bit address will be loaded that points to a vectoring address which is located in memory locations FFF8 and FFF9. An address loaded at these locations causes the MPU to branch to an interrupt routine in memory.

The HALT line must be in the "High" state for interrupts to be serviced. Interrupts will be latched internally while HALT is "Low".

A  $3k\Omega$  external register to  $V_{CC}$  should be used for wire-OR and optimum control of interrupts.

#### ● Reset (RES)

This input is used to reset and start the MPU from a power-down condition, resulting from a power failure or an initial start-up of the processor. When this line is "Low", the MPU is inactive and the information in the registers will be lost. If a "High" level is detected on the input, this will signal the MPU to begin the restart sequence. This will start execution of a routine to initialize the processor from its reset condition. All the higher order address lines will be forced "High". For the restart, the last two(FFFE, FFFF) locations in memory will be used to load the program that is addressed by the program counter. During the restart routine, the interrupt mask bit is set and must be reset before the MPU can be interrupted by IRQ. Power-up and reset timing and power-down sequences are shown in Fig. 7 and Fig. 8 respectively.

#### ● Non-Maskable Interrupt (NMI)

A low-going edge on this input requests that a non-mask-interrupt sequence be generated within the processor. As with the IRQ signal, the processor will complete the current instruction that is being executed before it recognizes the NMI signal. The interrupt mask bit in the Condition Code Register has no effect on NMI.

The Index Register, Program Counter, Accumulators, and Condition Code Register are stored away on the stack. At the end of the cycle, a 16-bit address will be loaded that points to a vectoring address which is located in memory locations FFEC and FFED. An address loaded at these locations causes the MPU to branch to a non-maskable interrupt routine in memory. A  $3k\Omega$  external resistor to  $V_{CC}$  should be used for wire-OR and optimum control of interrupts.

Inputs IRQ and NMI are hardware interrupt lines that are sampled when E is "High" and will start the interrupt routine on a "Low" E following the completion of an instruction. IRQ and NMI should be tied "High" if not used. This is good engineering design practice in general and necessary to insure proper operation of the part. Fig. 9 is a flowchart describing the major decision paths and interrupt vectors of the microprocessor. Table 1 gives the memory map for interrupt vectors.

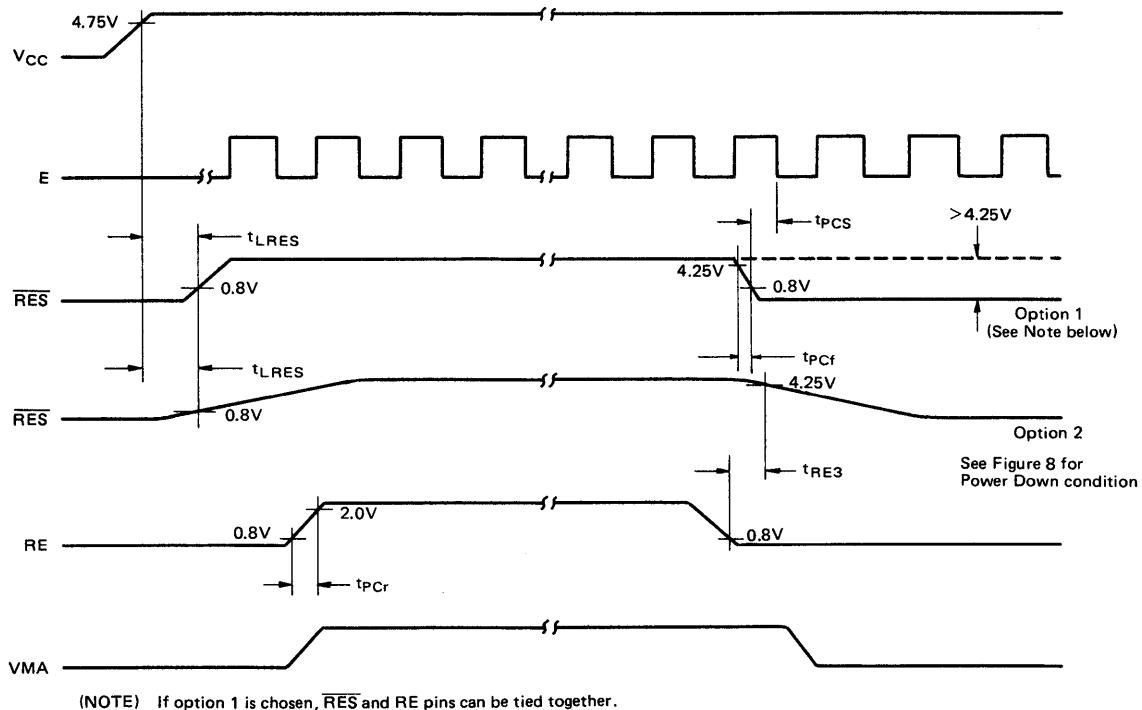


Figure 7 Power-up and Reset Timing

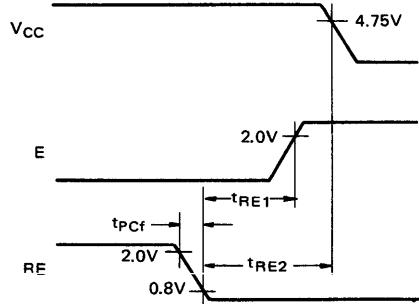


Figure 8 Power-down Sequence

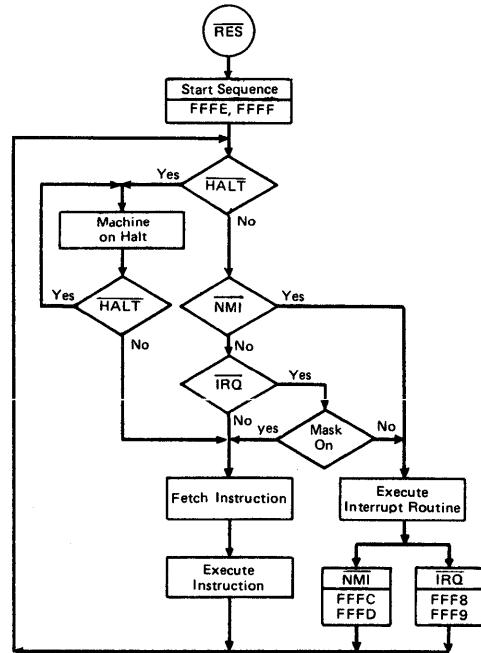


Figure 9 MPU Flow Chart

Table 1 Memory Map for Interrupt Vectors

MS	Vector	LS	Description
FFFE	FFFF		Restart ( $\overline{RES}$ )
FFF0	FFF0		Non-Maskable Interrupt ( $\overline{NMI}$ )
FFF1	FFF1		Software Interrupt ( $SWI$ )
FFF2	FFF2		Interrupt Request ( $IRQ$ )

- **RAM Enable (RE)**

A TTL-compatible RAM enable input controls the on-chip RAM of the HD6802. When placed in the "High" state, the on-chip memory is enabled to respond to the MPU controls. In the "Low" state, RAM is disabled. This pin may also be utilized to disable reading and writing the on-chip RAM during a power-down situation. RAM enable must be "Low" three cycles before  $V_{CC}$  goes below 4.75V during power-down.

RE should be tied to the correct "High" or "Low" state if not used. This is good engineering design practice in general and necessary to insure proper operation of the part.

- **EXTAL and XTAL**

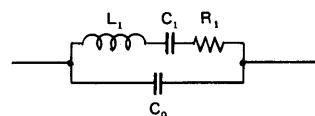
The HD6802 has an internal oscillator that may be crystal controlled. These connections are for a parallel resonant fundamental crystal (AT cut). A divide-by-four circuit has been added to the HD6802 so that a 4MHz crystal may be used in lieu of a 1MHz crystal for a more cost-effective system. Pin39 of the HD6802 may be driven externally by a TTL input signal if a separate clock is required. Pin38 is to be left open in this mode.

An RC network is not directly usable as a frequency source on pins 38 and 39. An RC network type TTL or CMOS oscillator will work well as long as the TTL or CMOS output drives the HD6802.

If an external clock is used, it may not be halted for more than  $4.5\mu s$ . The HD6802 is a dynamic part except for the internal RAM, and requires the external clock to retain information.

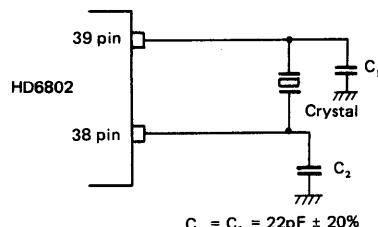
### Conditions for Crystal (4 MHz)

- AT Cut Parallel resonant
- $C_0 = 7 \text{ pF max.}$
- $R_1 = 80\Omega \text{ max.}$



Crystal Equivalent Circuit

### Recommended Oscillator (4MHz)



$$C_1 = C_2 = 22\text{pF} \pm 20\%$$

\*  $R_1=80\Omega$  max for S type,  $50\Omega$  max for R type. In addition  $56\text{k}\Omega$  resistance shall be put externally between 38 pin and 39 pin in parallel with Crystal for R type.

Figure 10 Crystal Oscillator

When using the crystal, see the note for Board Design of the Oscillation Circuit in HD6802.

- **Memory Ready (MR)**

MR is a TTL compatible input control signal which allows stretching of E. When MR is "High", E will be in normal operation. When MR is "Low", E may be stretched integral multiples of half periods, thus allowing interface to slow memories. Memory Ready timing is shown in Fig. 11.

MR should be tied "High" if not used. This is good engineering design practice in general and necessary to insure proper operation of the part. A maximum stretch is  $4.5\mu s$ .

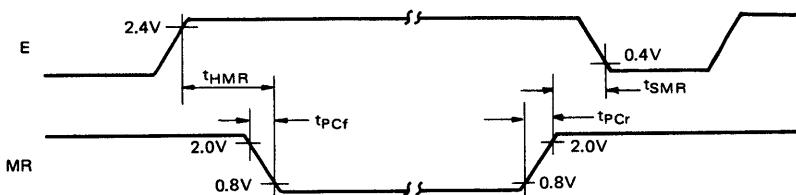


Figure 11 Memory Ready Control Function

- **Enable (E)**

This pin supplies the clock for the MPU and the rest of the system. This is a single phase, TTL compatible clock. This clock may be conditioned by a Memory Ready Signal. This is equivalent to  $\phi_2$  on the HD6800.

- **V<sub>CC</sub> Standby**

This pin supplies the dc voltage to the first 32 bytes of RAM as well as the RAM Enable (RE) control logic. Thus retention of data in this portion of the RAM on a power up, power-down, or standby condition is guaranteed at the range of 4.0 V to 5.25 V.

Maximum current drain at 5.25V is 8mA.

#### ■ MPU INSTRUCTION SET

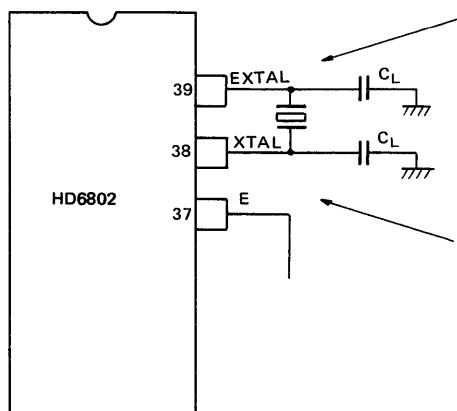
The HD6802 has a set of 72 different instructions. Included are binary and decimal arithmetic, logical, shift, rotate, load, store, conditional or unconditional branch, interrupt and stack manipulation instructions.

This instruction set is the same as that for the 6800MPU(HD6800 etc.) and is not explained again in this data sheet.

#### ■ NOTE FOR BOARD DESIGN OF THE OSCILLATION CIRCUIT IN HD6802

In designing the board, the following notes should be taken when the crystal oscillator is used.

56k $\Omega$  resistance shall be put externally between 38 pin and 39 pin in parallel with crystal for R type.

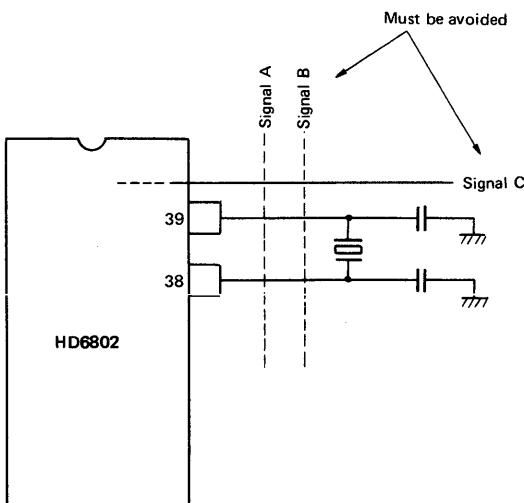


Crystal oscillator and load capacity  $C_L$  must be placed near the LSI as much as possible.

Normal oscillation may be disturbed when external noise is induced to pin 38 and 39.

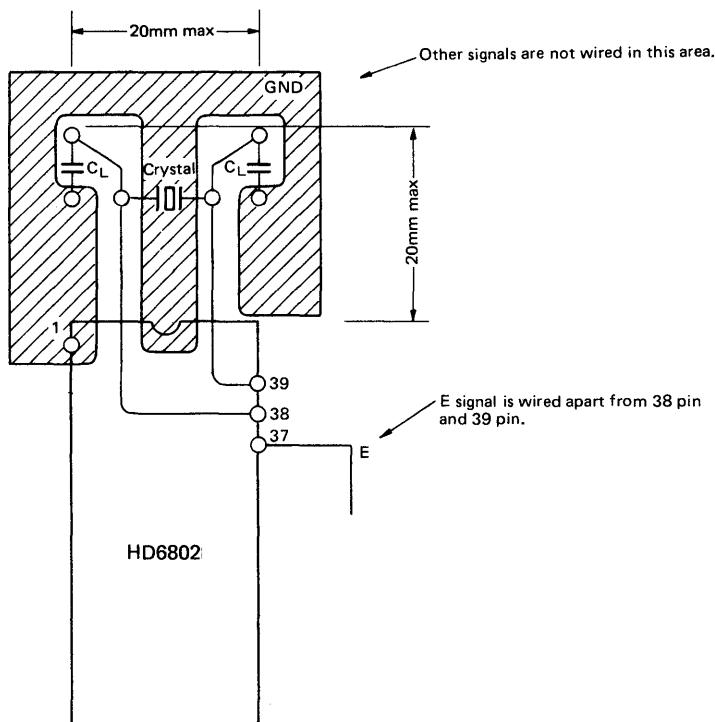
Pin 38 signal line should be wired apart from pin 37 signal line as much as possible. Don't wire them in parallel, or normal oscillation may be disturbed when E signal is feedbacked to XTAL.

The following design must be avoided.



A signal line or a power source line must not cross or go near the oscillation circuit line as shown in the left figure to prevent the induction from these lines and perform the correct oscillation. The resistance among XTAL, EXTAL and other pins should be over 10M $\Omega$ .

## Example of Board Design using the crystal oscillator



# HD6809, HD68A09, HD68B09

## MPU (Micro Processing Unit)

The HD6809 is a revolutionary high performance 8-bit microprocessor which supports modern programming techniques such as position independence, reentrancy, and modular programming.

This third-generation addition to the HMCS6800 family has major architectural improvements which include additional registers, instructions and addressing modes.

The basic instructions of any computer are greatly enhanced by the presence of powerful addressing modes. The HD6809 has the most complete set of addressing modes available on any 8-bit microprocessor today.

The HD6809 has hardware and software features which make it an ideal processor for higher level language execution or standard controller applications.

### HD46800D COMPATIBLE

- Hardware — Interfaces with All HMCS6800 Peripherals
- Software — Upward Source Code Compatible Instruction Set and Addressing Modes

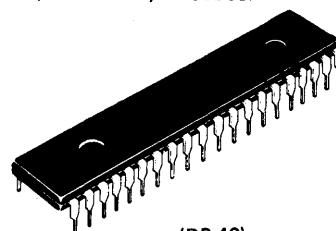
### ■ ARCHITECTURAL FEATURES

- Two 16-bit Index Registers
- Two 16-bit Indexable Stack Pointers
- Two 8-bit Accumulators can be Concatenated to Form One 16-Bit Accumulator
- Direct Page Register Allows Direct Addressing Throughout Memory

### ■ HARDWARE FEATURES

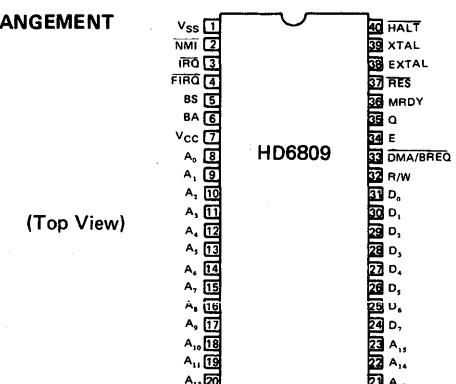
- On Chip Oscillator
- DMA/BREQ Allows DMA Operation or Memory Refresh
- Fast Interrupt Request Input Stacks Only Condition Code Register and Program Counter
- MRDY Input Extends Data Access Times for Use With Slow Memory
- Interrupt Acknowledge Output Allows Vectoring By Devices
- SYNC Acknowledge Output Allows for Synchronization to External Event
- Single Bus-Cycle RESET
- Single 5-Volt Supply Operation
- NMI Blocked After RESET Until After First Load of Stack Pointer
- Early Address Valid Allows Use With Slower Memories
- Early Write-Data for Dynamic Memories
- Compatible with MC6809, MC68A09 and MC68B09

HD6809P, HD68A09P, HD68B09P



(DP-40)

### ■ PIN ARRANGEMENT



### ■ SOFTWARE FEATURES

- 10 Addressing Modes
  - HMCS6800 Upward Compatible Addressing Modes
  - Direct Addressing Anywhere in Memory Map
  - Long Relative Branches
  - Program Counter Relative
  - True Indirect Addressing
  - Expanded Indexed Addressing

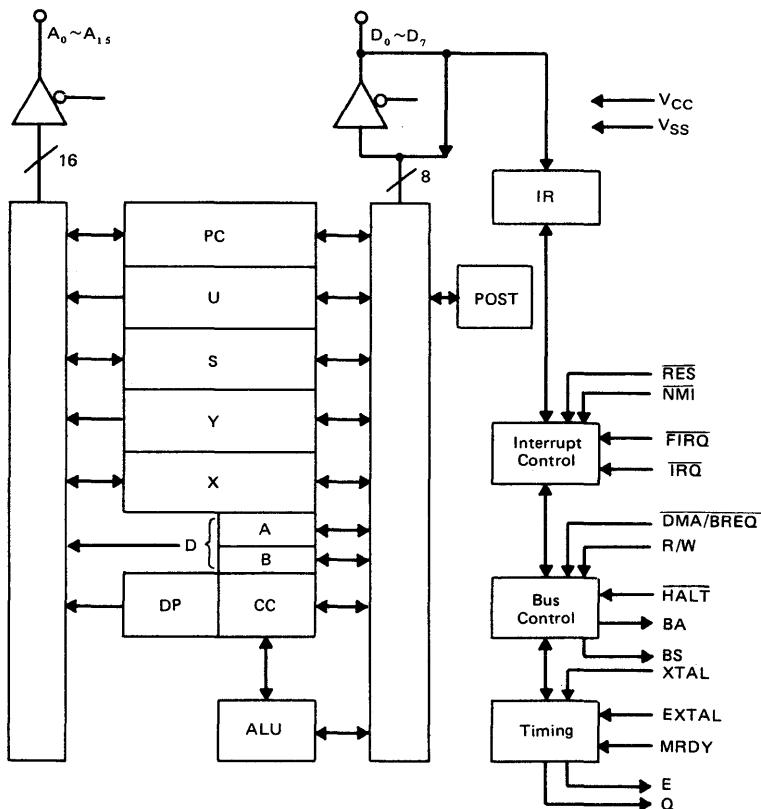
0, 5, 8, or 16-bit Constant Offsets

8, or 16-bit Accumulator Offsets

Auto-Increment/Decrement by 1 or 2

- Improved Stack Manipulation
- 1464 Instructions with Unique Addressing Modes
- 8 x 8 Unsigned Multiply
- 16-bit Arithmetic
- Transfer/Exchange All Registers
- Push/Pull Any Registers or Any Set of Registers
- Load Effective Address

**■ BLOCK DIAGRAM**



**■ ABSOLUTE MAXIMUM RATINGS**

Item	Symbol	Value	Unit
Supply Voltage	V <sub>CC</sub> *	-0.3 ~ +7.0	V
Input Voltage	V <sub>in</sub> *	-0.3 ~ +7.0	V
Operating Temperature	T <sub>opr</sub>	-20 ~ +75	°C
Storage Temperature	T <sub>stg</sub>	-55 ~ +75	°C

\* With respect to V<sub>SS</sub> (SYSTEM GND)

(NOTE) 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.

## ■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol	min	typ	max	Unit
Supply Voltage	$V_{CC}^*$	4.75	5	5.25	V
	$V_{IL}^*$	-0.3	-	0.8	V
Input Voltage	$V_{IH}^*$	Logic	2.0	-	$V_{CC}^*$
		$\bar{RES}$	4.0	-	$V_{CC}^*$
Operating Temperature	$T_{opr}$	-20	25	75	°C

\* With respect to  $V_{SS}$  (SYSTEM GND)

## ■ ELECTRICAL CHARACTERISTICS

- DC CHARACTERISTICS ( $V_{CC}=5V\pm5%$ ,  $V_{SS}=0$ ,  $T_a=-20\sim+75^\circ C$ , unless otherwise noted.)

Item	Symbol	Test Condition	min	typ*	max	Unit
Input "High" Voltage	Logic, EXTAL	$V_{IH}$	2.0	-	$V_{CC}$	V
	$\bar{RES}$		4.0	-	$V_{CC}$	
Input "Low" Voltage	Logic, EXTAL, $\bar{RES}$	$V_{IL}$	-0.3	-	0.8	V
Input Leakage Current	Logic	$I_{in}$	$V_{in}=0\sim5.25V$ , $V_{CC}$ max.		1.0	2.5 $\mu A$
Output "High" Voltage	$D_0\sim D_7$	$V_{OH}$	$I_{load}=205\mu A$ , $V_{CC}$ min.	2.4	-	V
	$A_0\sim A_{15}$ , R/W, Q, E		$I_{load}=145\mu A$ , $V_{CC}$ min.	2.4	-	
	BA, BS		$I_{load}=100\mu A$ , $V_{CC}$ min.	2.4	-	
Output "Low" Voltage	$V_{OL}$	$I_{load}=2.0mA$ , $V_{CC}$ min.		-	-	0.5 V
Power Dissipation	$P_D$			-	-	1.0 W
Input Capacitance	$D_0\sim D_7$	$C_{in}$	$V_{in}=0V$ , $T_a=25^\circ C$ , $f=1.0MHz$		10	15 $pF$
	Logic Inputs, EXTAL				7	10 $pF$
Output Capacitance	$A_0\sim A_{15}$ , R/W, BA, BS	$C_{out}$	$V_{in}=0V$ , $T_a=25^\circ C$ , $f=1.0MHz$		-	12 $pF$
Three-State (Off State) Input Current	$D_0\sim D_7$	$I_{TSI}$	$V_{in}=0.4\sim2.4V$ , $V_{CC}$ max.		2	10 $\mu A$
	$A_0\sim A_{15}$ , R/W				-	100 $\mu A$

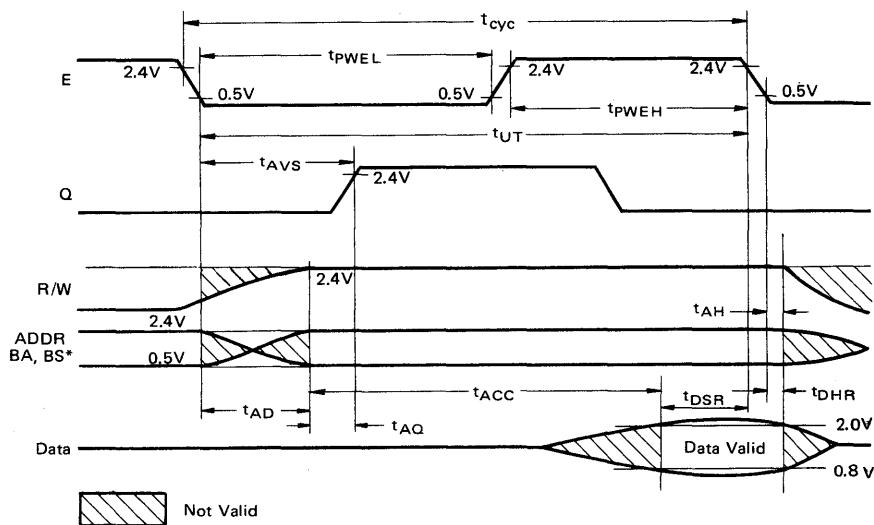
\*  $T_a=25^\circ C$ ,  $V_{CC}=5V$

## ● AC CHARACTERISTICS

### 1. READ/WRITE TIMING

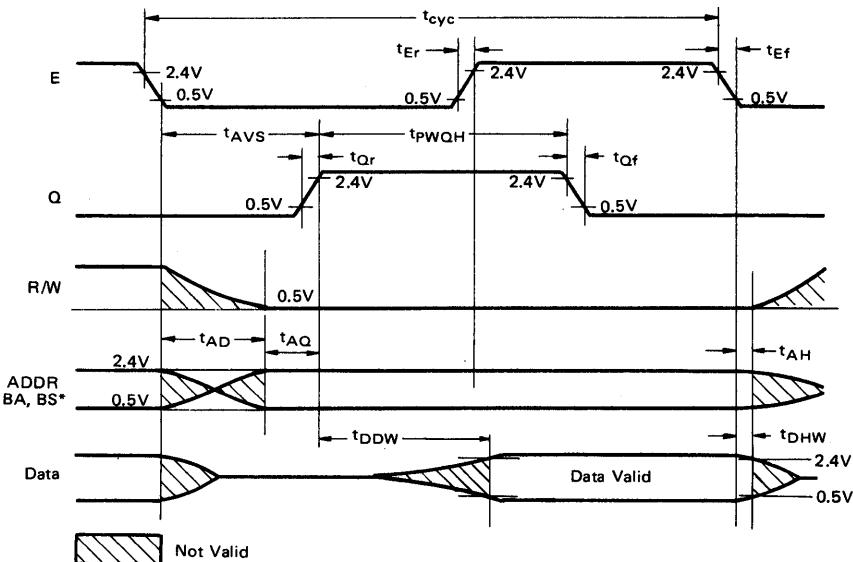
Item	Symbol	Test Condition	HD6809			HD68A09			HD68B09			Unit
			min	typ	max	min	typ	max	min	typ	max	
Frequency of Operation (Crystal or External Input)	$f_{XTAL}$		-	-	4	-	-	6	-	-	8	MHz
Cycle Time	$t_{cyc}$		1000	-	-	667	-	-	500	-	-	ns
Total Up Time	$t_{UT}$		975	-	-	640	-	-	480	-	-	ns
Peripheral Read Access Time ( $t_{UT}-t_{AD}-t_{DSR}=t_{ACC}$ )	$t_{ACC}$		695	-	-	440	-	-	330	-	-	ns
Data Set Up Time (Read)	$t_{DSR}$		80	-	-	60	-	-	40	-	-	ns
Input Data Hold Time	$t_{DHR}$		10	-	-	10	-	-	10	-	-	ns

Item	Symbol	Test Condition	HD6809			HD68A09			HD68B09			Unit
			min	typ	max	min	typ	max	min	typ	max	
Output Data Hold Time	$t_{DHW}$		20	—	—	20	—	—	20	—	—	ns
Address Hold Time (Address, R/W)	$t_{AH}$		10	—	—	10	—	—	10	—	—	ns
Address Delay	$t_{AD}$		—	—	200	—	—	140	—	—	110	ns
Data Delay Time (Write)	$t_{DDW}$		—	—	225	—	—	180	—	—	145	ns
E low to Q <sub>high</sub> Time	$t_{AVS}$		—	—	250	—	—	165	—	—	125	ns
Address Valid to Q <sub>high</sub>	$t_{AQ}$		50	—	—	25	—	—	15	—	—	ns
Processor Clock "Low"	$t_{PWEL}$		450	—	—	295	—	—	210	—	—	ns
Processor Clock "High"	$t_{PWEH}$		450	—	—	280	—	—	220	—	—	ns
MRDY Set Up Time	$t_{PCSM}$		125	—	—	125	—	—	125	—	—	ns
Interrupts Set Up Time	$t_{PCS}$		200	—	—	140	—	—	110	—	—	ns
HALT Set Up Time	$t_{PCSH}$		200	—	—	140	—	—	110	—	—	ns
RES Set Up Time	$t_{PCSR}$		200	—	—	140	—	—	110	—	—	ns
DMA/BREQ Set Up Time	$t_{PCSD}$		125	—	—	125	—	—	125	—	—	ns
Crystal Osc Start Time	$t_{RC}$		100	—	—	100	—	—	100	—	—	ms
E Rise and Fall Time	$t_{ER}, t_{EF}$		—	—	25	—	—	25	—	—	20	ns
Processor Control Rise/Fall	$t_{PCR}, t_{PCF}$		—	—	100	—	—	100	—	—	100	ns
Q Rise and Fall Time	$t_{QR}, t_{QF}$		—	—	25	—	—	25	—	—	20	ns
Q Clock "High"	$t_{PWQH}$		450	—	—	280	—	—	220	—	—	ns



\*Hold for BA, BS not specified

Figure 1 Read Data from Memory or Peripherals



\*Hold time for BA, BS not specified

Figure 2. Write Data to Memory or Peripherals

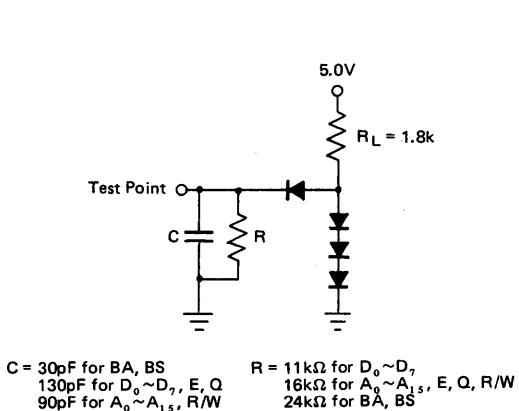


Figure 3 Bus Timing Test Load

#### ■ PROGRAMMING MODEL

As shown in Figure 4, the HD6809 adds three registers to the set available in the HD6800. The added registers include 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, and is formed with the A register as the most significant byte.

#### ● Direct Page Register (DP)

The Direct Page Register of the HD6809 serves to enhance the Direct Addressing Mode. The content of this register appears at the higher address outputs ( $A_8 \sim A_{15}$ ) during Direct Addressing Instruction execution. This allows the direct mode to be used at any place in memory, under program control. To ensure HD46800D compatibility, all bits of this register are cleared during Processor Reset.

#### ● Index Registers (X, Y)

The Index Registers are used in indexed mode of 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. During some indexed modes, the contents of the index register are incremented and decremented to point to the next item of tabular type data. All four pointer registers (X, Y, U, S) may be used as index registers.

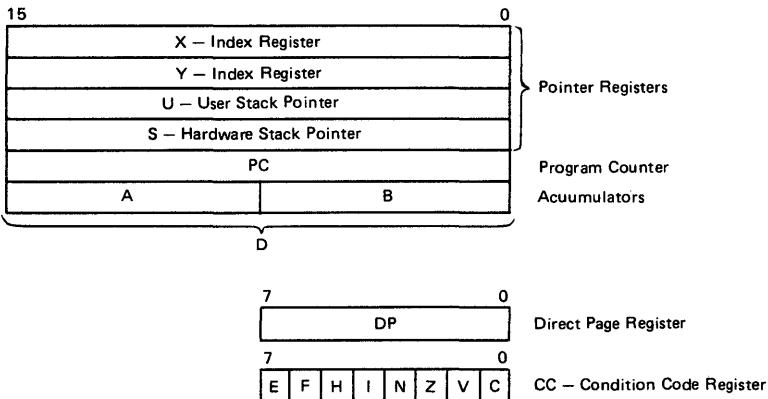


Figure 4 Programming Model of The Microprocessing Unit

#### ● 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 HD6809 point to the top of the stack, in contrast to the HD46800D 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 HD6809 to be used efficiently as a stack processor, greatly enhancing its ability to support higher level languages and modular programming.

#### ● Program Counter

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

The Condition Code Register defines the State of the Processor at any given time. See Fig. 5.

#### ■ CONDITION CODE REGISTER DESCRIPTION

##### ● Bit 0 (C)

Bit 0 is the carry flag, and is usually the carry from the binary ALU. C is also used to represent a ‘borrow’ from subtract like instructions (CMP, NEG, S<sub>UB</sub>, S<sub>BC</sub>) and is the complement of the carry from the binary ALU.

##### ● Bit 1 (V)

Bit 1 is the overflow flag, and 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-1.

##### ● Bit 2 (Z)

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

##### ● Bit 3 (N)

Bit 3 is the negative flag, which 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 a one.

##### ● Bit 4 (I)

Bit 4 is the IRQ mask bit. The processor will not recognize interrupts from the IRQ line if this bit is set to a one. NMI, FIRQ, IRQ, R<sub>ES</sub>, and SWI all are set I to a one; SWI2 and SWI3 do not affect I.

##### ● Bit 5 (H)

Bit 5 is the half-carry bit, and 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 FIRQ mask bit. The processor will not recognize interrupts from the FIRQ line if this bit is a one. NMI, FIRQ, SWI, and R<sub>ES</sub> all set F to a one. IRQ, SWI2 and SWI3 do not

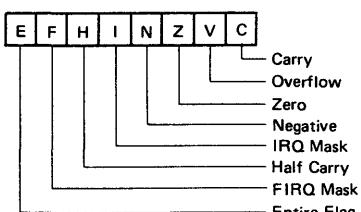


Figure 5 Condition Code Register Format

affect F.

#### • Bit 7 (E)

Bit 7 is the entire flag, and when set to a one 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.

### ■ SIGNAL DESCRIPTION

#### • Power ( $V_{SS}$ , $V_{CC}$ )

Two pins are used to supply power to the part:  $V_{SS}$  is ground or 0 volts, while  $V_{CC}$  is +5.0V ±5%.

#### • Address Bus ( $A_0 \sim A_{15}$ )

Sixteen pins are used to 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/W = "High", and BS = "Low"; this is a "dummy access" or  $\overline{VMA}$  cycle. Addresses are valid on the rising edge of Q (see Figs. 1 and 2). All address bus drivers are made high impedance when output Bus Available (BA) is "High". Each pin will drive one Schottky TTL load or four LS TTL loads, and typically 90 pF.

#### • Data Bus ( $D_0 \sim D_7$ )

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/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/W is made high impedance when BA is "High". R/W is valid on the rising edge of Q. Refer to Figs. 1 and 2.

#### • Reset ( $\overline{RES}$ )

A "Low" level on this Schmitt-trigger input for greater than one bus cycle will reset the MPU, as shown in Fig. 6. 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 Fig. 7.

Because the HD6809 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 threshold voltage ensures that all peripherals are out of the reset state before the Processor.

#### • 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 real-time 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 ( $\overline{RES}$ ,  $\overline{DMA/BREQ}$ ), a halted state ( $\overline{BA} \cdot BS=1$ ) can be achieved by pulling  $\overline{HALT}$

"Low" while  $\overline{RES}$  is still "Low". If  $\overline{DMA/BREQ}$  and  $\overline{HALT}$  are both pulled "Low", the processor will reach the last cycle of the instruction (by reverse cycle stealing) where the machine will then become halted. See Figs. 8 and 16.

#### • 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 (valid with leading edge of Q).

Table 1 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

**Interrupt Acknowledge** is indicated during both cycles of a hardware-vector-fetch ( $\overline{RES}$ ,  $\overline{NMI}$ ,  $\overline{FIRQ}$ ,  $\overline{IRQ}$ ,  $\overline{SWI}$ ,  $\overline{SWI2}$ ,  $\overline{SWI3}$ ). 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.

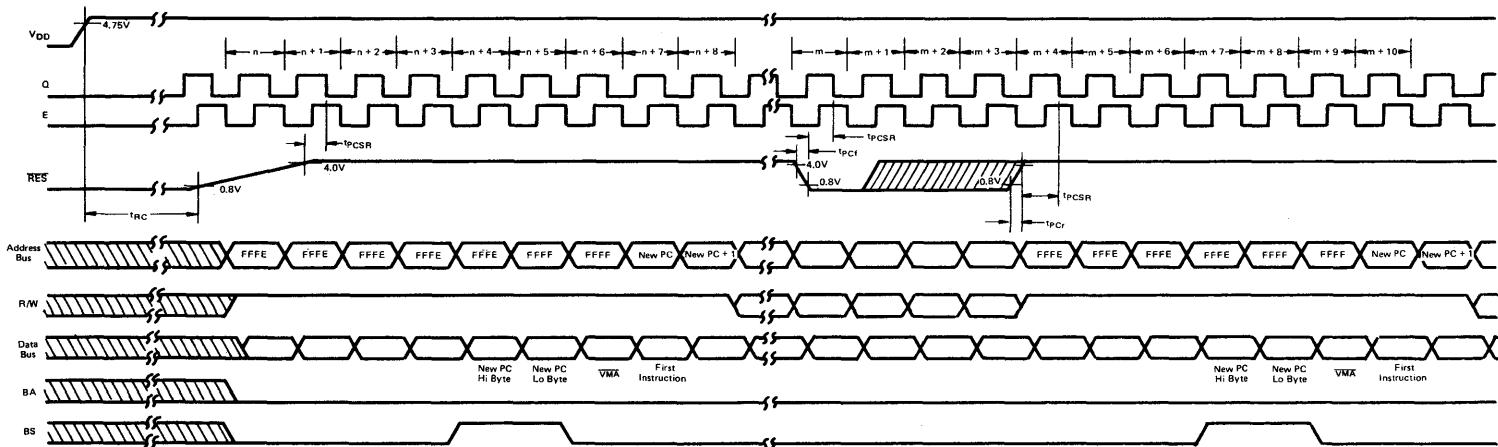
**Halt/Bus Grant** is true when the HD6809 is in a Halt or Bus Grant condition.

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	$\overline{SWI}$
FFF8	FFF9	$\overline{IRQ}$
FFF6	FFF7	$\overline{FIRQ}$
FFF4	FFF5	$\overline{SWI2}$
FFF2	FFF3	$\overline{SWI3}$
FFF0	FFF1	Reserved

#### • Non Maskable Interrupt ( $\overline{NMI}$ )\*

A negative edge on this input requests that a nonmaskable interrupt sequence be generated. A non-maskable 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 Fig. 9.

Figure 6  $\overline{RES}$  Timing

55

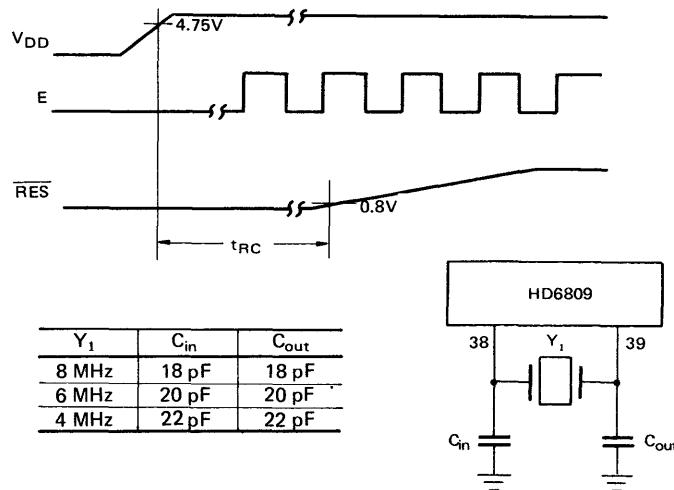


Figure 7 Crystal Connections and Oscillator Start Up

$Y_1$	$C_{in}$	$C_{out}$
8 MHz	18 pF	18 pF
6 MHz	20 pF	20 pF
4 MHz	22 pF	22 pF

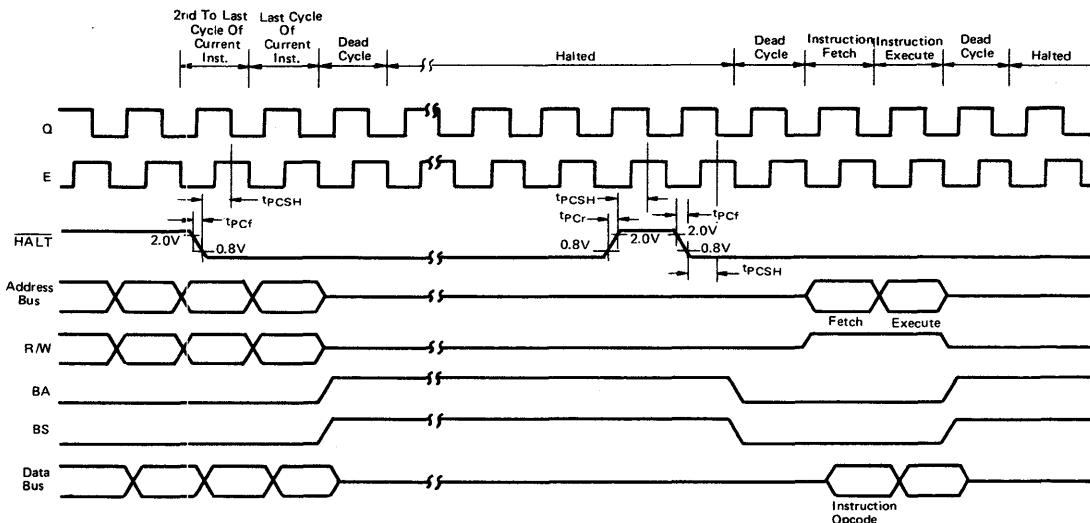


Figure 8 HALT and Single Instruction Execution for System Debug

56

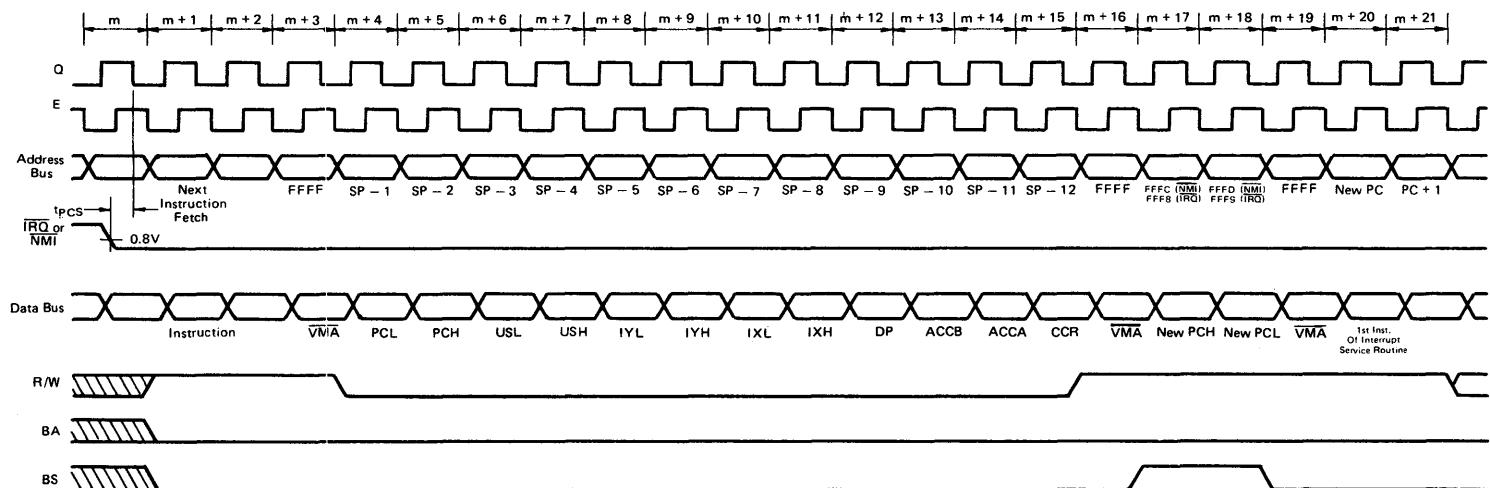


Figure 9 TRQ and NMI Interrupt Timing

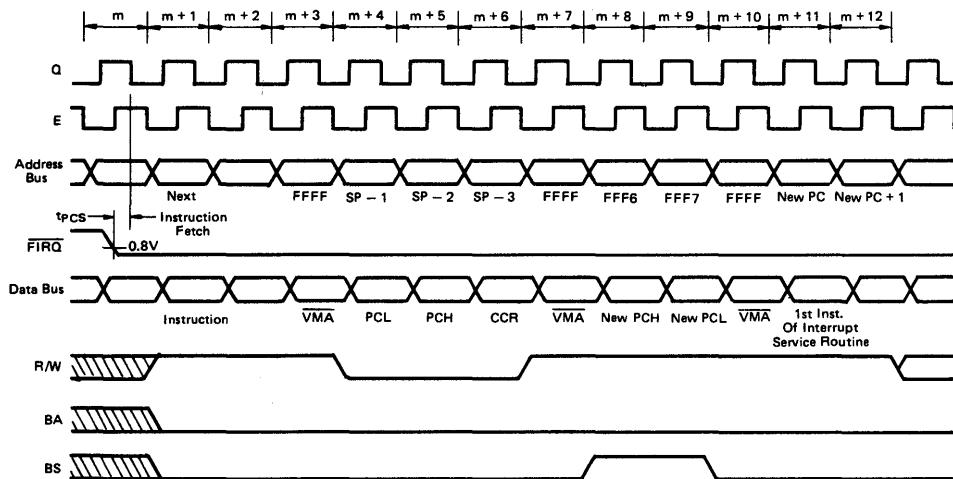


Figure 10 FIRQ Interrupt Timing

#### • Fast-Interrupt Request (FIRQ)\*

A “Low” level on this input pin 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 (IRQ), and 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 Fig. 10.

#### • Interrupt Request (IRQ)\*

A “Low” level input on this pin will initiate an interrupt Request sequence provided the mask bit (I) in the CC is clear. Since IRQ stacks the entire machine state it provides a slower response to interrupts than FIRQ. IRQ also has a lower priority than FIRQ. Again, the interrupt service routine should clear the source of the interrupt before doing an RTI. See Fig. 9.

\* NMI, FIRQ and IRQ requests are latched by the falling edge of every Q, except during cycle stealing operations (e.g., DMA) where only NMI is latched. From this point, a delay of at least one bus cycle will occur before the interrupt is serviced by MPU.

#### • XTAL, EXTAL

These inputs are used to connect the on-chip oscillator to an external parallel-resonant crystal. Alternately, the pin EXTAL may be used as a TTL level input for external timing by grounding XTAL. The crystal or external frequency is four times the bus frequency. See Fig. 7. Proper RF layout techniques should be observed in the layout of printed circuit boards.

#### • E, Q

E is similar to the HD6800 bus timing signal  $\phi_2$ ; Q is a quadrature clock signal which leads E. Q has no parallel on the HD6800. Addresses from the MPU will be valid with the leading edge of Q. Data is latched on the falling edge of E. Timing for E and Q is shown in Fig. 11.

#### • 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 quarter (1/4) bus cycles, thus allowing interface to slow memories, as shown in Fig. 12. A maximum

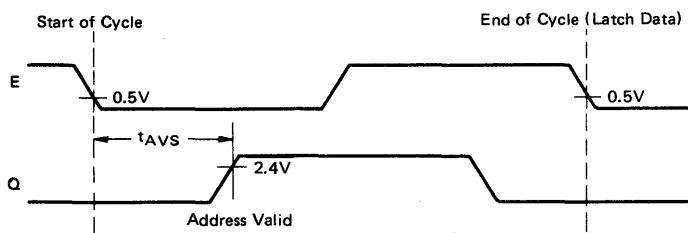


Figure 11 E/Q Relationship

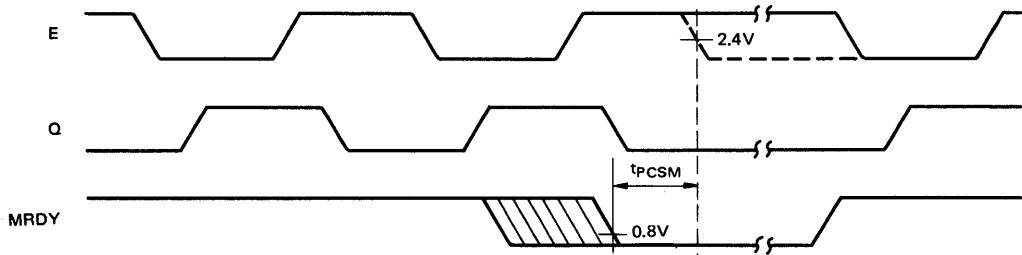


Figure 12 MRDY Timing

stretch is 10 microseconds. During nonvalid memory access (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 HALT and DMA/BREQ).

#### • DMA/BREQ

The DMA/BREQ input provides a method of suspending execution and acquiring the MPU bus for another use, as shown in Fig. 13. Typical uses include DMA and dynamic memory refresh.

Transition of 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 DMA/BREQ by setting BA and BS to a one. The requesting device will now have up to 15 bus cycles before the MPU retrieves the bus for self-refresh. Self-refresh requires one bus cycle with a leading

and trailing dead cycle. See Fig. 14.

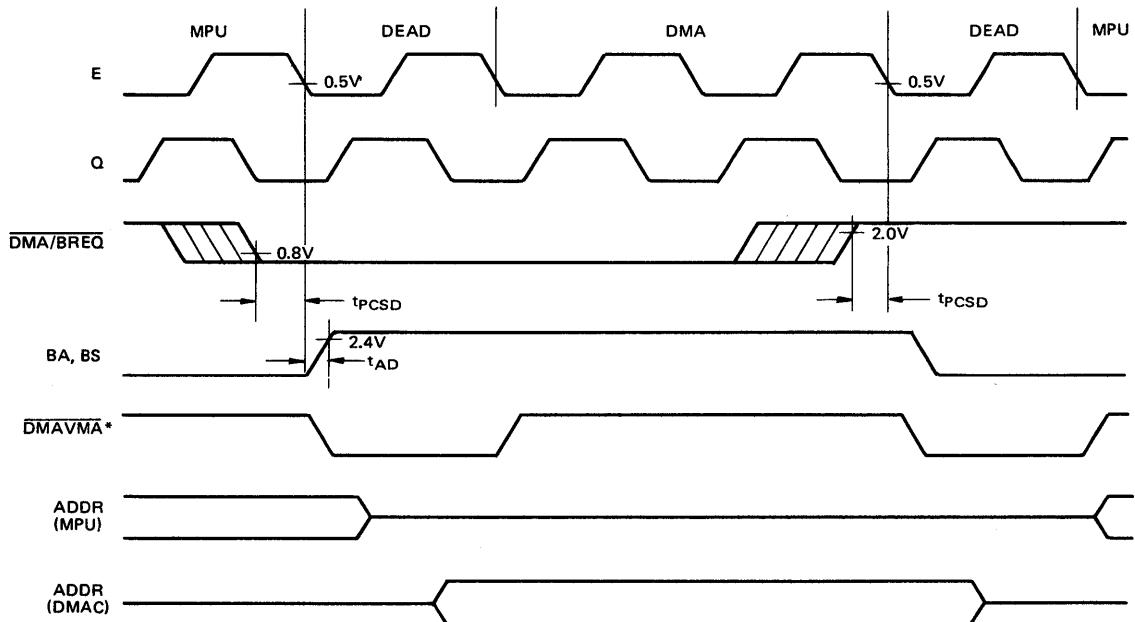
Typically, the DMA controller will request to use the bus by asserting DMA/BREQ pin "Low" on the leading edge of E. When the MPU replies by setting BA and BS to a one, that cycle will be a dead cycle used to transfer bus mastership to the DMA controller.

False memory accesses may be prevented during and dead cycles by developing a system DMAVMA signal which is "Low" in any cycle when BA has changed.

When BA goes "Low" (either as a result of DMA/BREQ = "High" or MPU self-refresh), the DMA device should be taken off the bus. Another dead cycle will elapse before the MPU accesses memory, to allow transfer of bus mastership without contention.

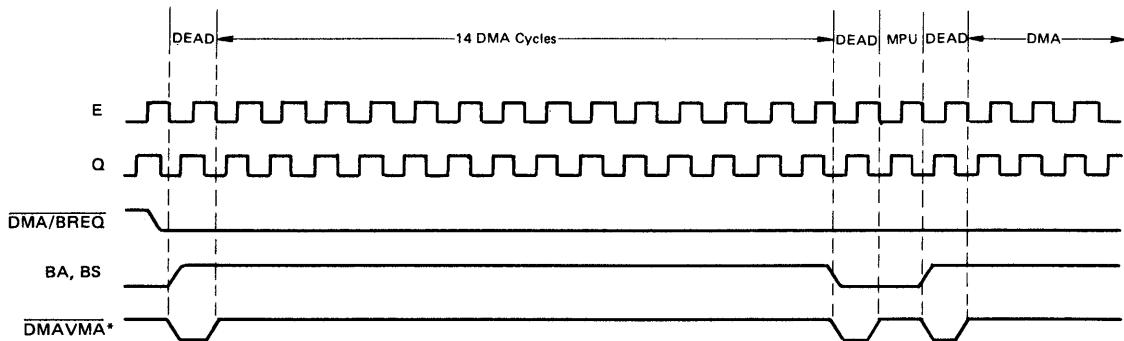
#### • MPU Operation

During normal operation, the MPU fetches an instruction from memory and then executes the requested function. This



\*DMAVMA is a signal which is developed externally, but is a system requirement for DMA.

Figure 13 Typical DMA Timing (&lt;14 Cycles)



\*DMAVMA is a signal which is developed externally, but is a system requirement for DMA.

Figure 14 Auto – Refresh DMA Timing  
(Reverse Cycle Stealing)

sequence begins at **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, HALT or DMA/BREQ can also alter the normal execution of instructions. Fig. 15 illustrates the flow chart for the HD6809.

#### ■ ADDRESSING MODES

The basic instructions of any computer are greatly enhanced by the presence of powerful addressing modes. The HD6809 has the most complete set of addressing modes available on any microcomputer today. For example, the HD6809 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 HD6809:

- Inherent (Includes Accumulator)
- Immediate
- Extended
- Extended Indirect
- Direct
- Register
- Indexed
- Zero-Offset
- Constant Offset
- Accumulator Offset
- Auto Increment/Decrement
- Indexed Indirect
- Relative
- Short/Long Relative Branching
- Program Counter Relative Addressing

#### ● Inherent (Includes Accumulator)

In this addressing mode, the opcode of the instruction contains all the address information necessary. Examples of Inherent 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 HD6809 uses both 8 and 16-bit immediate values depending on the size of 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
LDI $2000
```

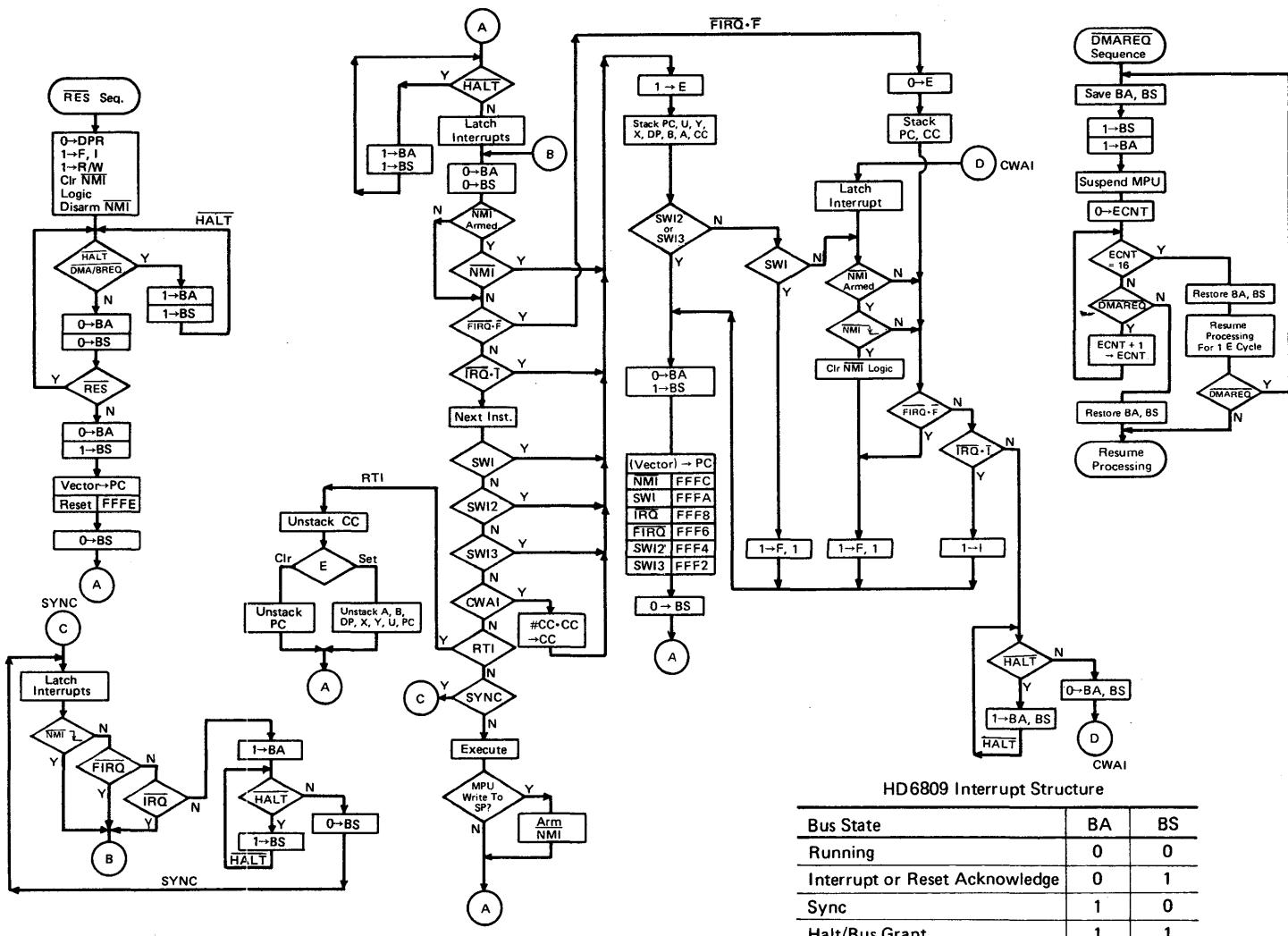
#### ● Extended Indirect

As a special case of indexed addressing (discussed below), "1" 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-bit of the address to be used. The upper 8-bit 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



(NOTE) Asserting RES will result in entering the reset sequence from any point in the flow chart.

Figure 15 Flowchart for HD6809 Instruction

accessed without redefining the contents of the DP register. Since the DP register is set to \$00 on Reset, direct addressing on the HD6809 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
LDI	<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. Fig. 16 lists the legal formats for the postbyte. Table 3 gives the assembler form and the number of cycles and bytes

Post-Byte Register Bit								Indexed Addressing Mode
7	6	5	4	3	2	1	0	
0	R	R	x	x	x	x	x	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 + ACCB Offset
1	R	R	0/1	0	1	1	0	EA = ,R + ACCA 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	x	x	0/1	1	1	0	0	EA = ,PC + 8 Bit Offset
1	x	x	0/1	1	1	0	1	EA = ,PC + 16 Bit Offset
1	R	R	1	1	1	1	1	EA = [,Address]

--	--	--

Addressing Mode Field

Indirect Field  
(Sign bit when b7 = 0)  
{0 ..... Nor Indirect  
{1 ..... Indirect

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

x = Don't Care

Figure 16 Index Addressing Postbyte Register Bit Assignments

Table 3 Indexed Addressing Mode

Type	Forms	Non Indirect				Indirect			
		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      RR:

x = Don't Care      00 = X  
01 = Y  
10 = U  
11 = S

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

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
```

#### 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 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 the "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 and is 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

#### ● 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 = XX (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]
LDU [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:

CAT	BEQ	CAT	(short)
DOG	BGT	DOG	(short)
	LBEQ	RAT	(long)
	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]

#### ■ HD6809 INSTRUCTION SET

The instruction set of the HD6809 is similar to that of the HD 6800 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 new instructions and addressing modes are described in detail below:

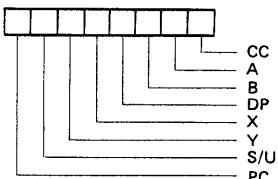
#### • PSHU/PSHS

The push instructions have the capability of pushing 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 below.

#### PUSH/PULL POST BYTE



← Pull Order      Push Order →

PC	U	Y	X	DP	B	A	CC
FFFF...	← increasing memory address .....0000						
PC	S	Y	X	DP	B	A	CC

#### • TFR/EXG

Within the HD6809, 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 postbyte define the source register, while bits 0-3 represent the destination register. Three 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.

#### TRANSFER/EXCHANGE POST BYTE

SOURCE	DESTINATION

#### • 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 4.

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.

Table 4 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 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

#### • MUL

Multiplies the unsigned binary numbers in the A 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 HD6809 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 (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. Fig. 17 depicts Sync timing.

**Software Interrupts**

A Software Interrupt is an instruction which 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 HD6809, and are prioritized in the following order: SWI, SWI2, SWI3.

**16-Bit Operation**

The HD6809 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 addressing mode in the HD6809. 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/W="High" and BS="Low". The following examples illustrate the use of the chart; see Fig. 18.

LBSR            (Branch taken)

Cycle # 1	opcode Fetch
2	opcode +
3	opcode +
4	<u>VMA</u>
5	<u>VMA</u>
6	ADDR
7	<u>VMA</u>
8	STACK (write)
9	STACK (write)

DEC            (Extended)

Cycle # 1	opcode Fetch
2	opcode +
3	opcode +
4	<u>VMA</u>
5	ADDR (read)
6	<u>VMA</u>
7	ADDR (write)

**■ HD6809 INSTRUCTION SET TABLES**

The instructions of the HD6809 have been broken down into five different categories. They are as follows:

8-Bit operation (Table 5)

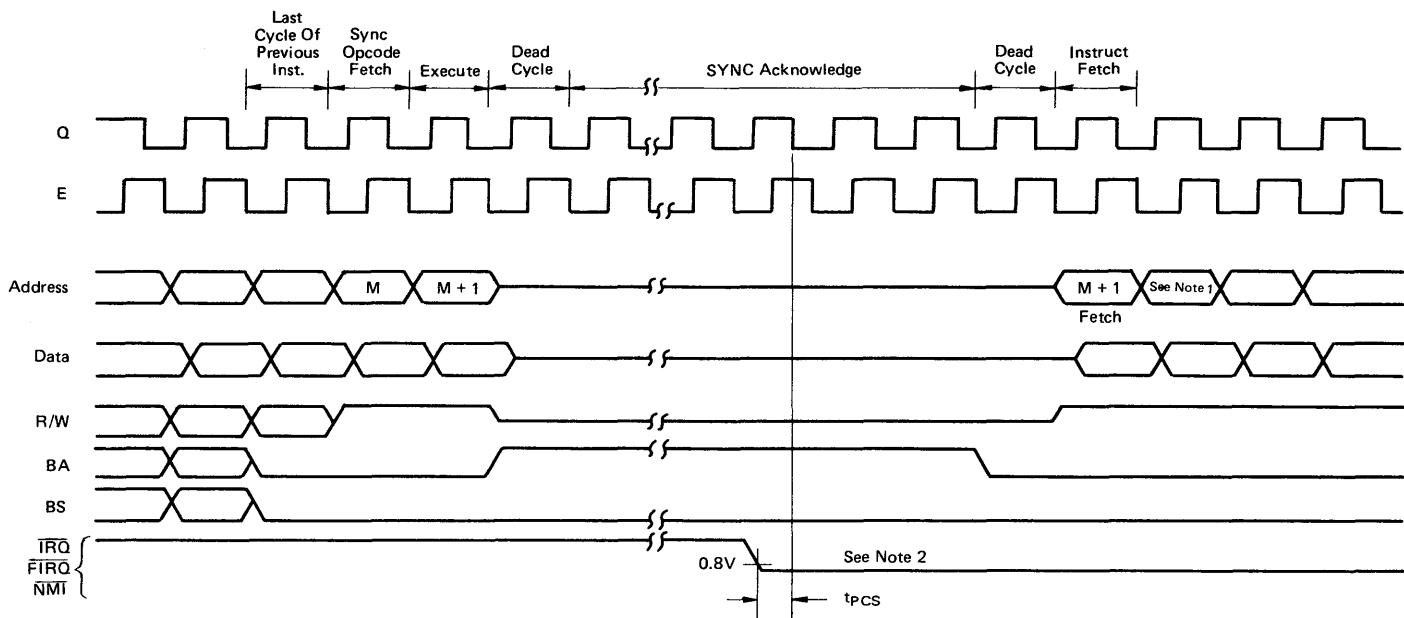
16-Bit operation (Table 6)

Index register/stack pointer instructions (Table 7)

Relative branches (long or short) (Table 8)

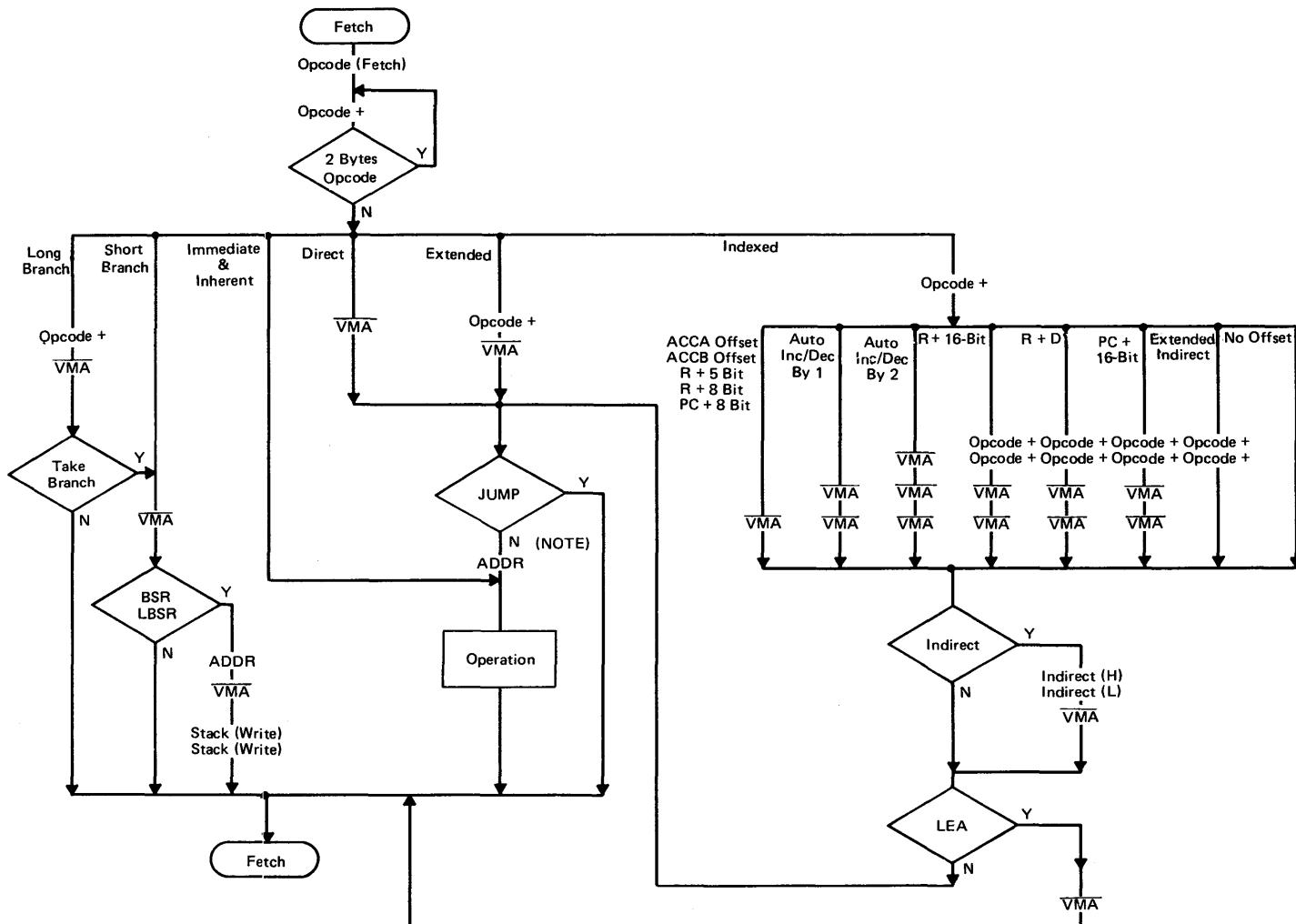
Miscellaneous instructions (Table 9)

HD6809 instruction set tables and Hexadecimal Values of instructions are shown in Table 10 and Table 11.



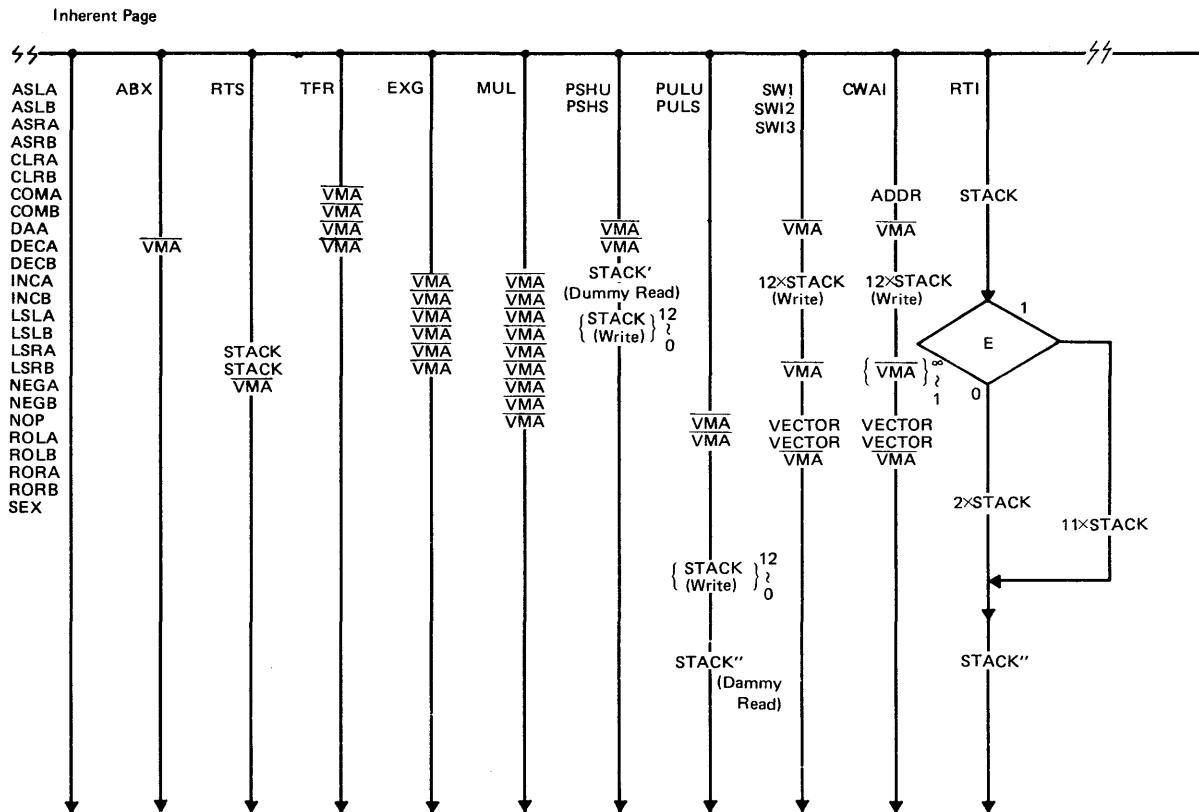
- (NOTE) 1. If the associated mask bit is set when the interrupt is requested, this cycle will continue the instruction fetched from the previous cycle. However, if the interrupt is accepted (NMI or an unmasked FIRQ or IRQ) the opcode address (placed on the bus from cycle M + 1) remains on the bus and interrupt processing continues with this cycle as (M + 2) on Figs. 9 and 10 (Interrupt Timing).  
 2. If mask bits are clear, IRQ and FIRQ must be held low for three cycles to guarantee interrupt to be taken, although only one cycle is necessary to bring the processor out of SYNC.

Figure 17 Sync Timing



(NOTE) Write operation during store instruction.

Figure 18 Address Bus Cycle-by-Cycle Performance



(NOTE) STACK': Address stored in stack pointer before execution.  
 STACK'': Address set to stack pointer as the result of the execution.

Figure 18 Address Bus Cycle-by-Cycle Performance (Continued)

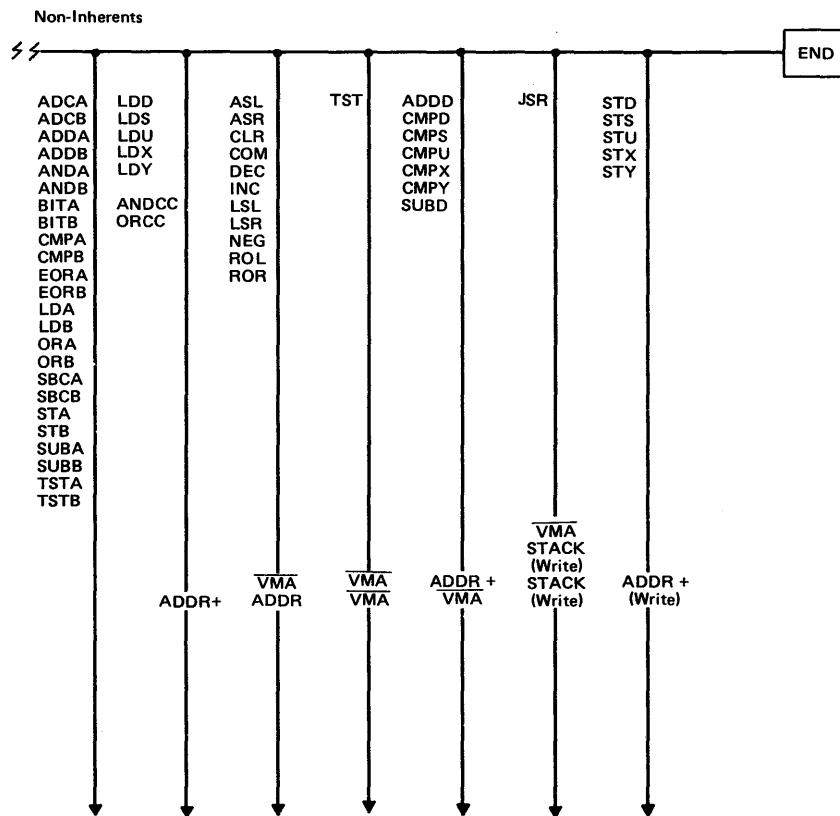


Figure 18 Address Bus Cycle-by-Cycle Performance (Continued)

Table 5 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 accumlutor 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 to 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 6 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 7 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 8 Branch Instructions

Mnemonic(s)	Operation
<b>SIMPLE BRANCHES</b>	
BEQ, LB EQ	Branch if equal
BNE, LB NE	Branch if not equal
BMI, LB MI	Branch if minus
BPL, LB PL	Branch if plus
BCS, LB CS	Branch if carry set
BCC, LB CC	Branch if carry clear
BVS, LB VS	Branch if overflow set
BVC, LB VC	Branch if overflow clear
<b>SIGNED BRANCHES</b>	
BGT, LB GT	Branch if greater (signed)
BGE, LB GE	Branch if greater than or equal (signed)
BEQ, LB EQ	Branch if equal
BLE, LB LE	Branch if less than or equal (signed)
BLT, LB LT	Branch if less than (signed)
<b>UNSIGNED BRANCHES</b>	
BHI, LB HI	Branch if higher (unsigned)
BHS, LB HS	Branch if higher or same (unsigned)
BEQ, LB EQ	Branch if equal
BLS, LB LS	Branch if lower or same (unsigned)
BLO, LB LO	Branch if lower (unsigned)
<b>OTHER BRANCHES</b>	
BSR, LB SR	Branch to subroutine
BRA, LB RA	Branch always
BRN, LB RN	Branch never

Table 9 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 10 HD6809 Instruction Set Table

INSTRUCTION/ FORMS		HD6809 ADDRESSING MODES														DESCRIPTION	5 3 2 1 0								
		INHERENT			DIRECT			EXTENDED			IMMEDIATE			INDEXED <sup>(1)</sup>				H	N	Z	V	C			
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~ <sup>(2)</sup>	#						
ABX		3A	3	1																B + X → X (UNSIGNED)	•	•	•	•	•
ADC	ADCA	D9	4	2	B9	5	3	89	2	2	A9	4+	2+							A + M + C → A	†	†	†	†	†
	ADCB				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	●	†	†	0	●
	ANDB	D4	4	2	F4	5	3	C4	2	2	E4	4+	2+							B ∧ M → B	●	†	†	0	●
	ANDCC							1C	3	2									CC ∧ IMM → CC	(	7)	)			
ASL	ASLA	48	2	1																	⑧	†	†	†	†
	ASLB	58	2	1																	⑧	†	†	†	†
	ASL																								
ASR	ASRA	47	2	1	08	6	2	78	7	3											⑧	†	†	●	†
	ASRB	57	2	1																	⑧	†	†	●	†
BCC	BCC				07	6	2	77	7	3										Branch C = 0	●	●	●	●	●
	LBCC																			Long Branch C = 0	●	●	●	●	●
BCS	BCS																			Branch C = 1	●	●	●	●	●
	LBCS																			Long Branch C = 1	●	●	●	●	●
BEQ	BEQ																			Branch Z = 1	●	●	●	●	●
	LBEQ																			Long Branch Z = 1	●	●	●	●	●
BGE	BGE																			Branch N ⊕ V=0	●	●	●	●	●
	LBGE																			Long Branch N ⊕ V=0	●	●	●	●	●
BGT	BGT																			Branch ZV(N⊕V)=0	●	●	●	●	●
	LBGT																			Long Branch ZV(N⊕V)=0	●	●	●	●	●
BHI	BHI																			Branch CVZ=0	●	●	●	●	●
	LBHI																			Long Branch CVZ=0	●	●	●	●	●
BHS	BHS																			Branch C=0	●	●	●	●	●
	LBHS																			Long Branch C=0	●	●	●	●	●
BIT	BITA																			Bit Test A (M ∧ A)	●	†	†	0	●
	BITB	95	4	2	B5	5	3	85	2	2	A5	4+	2+							Bit Test B (M ∧ B)	●	†	†	0	●
BLE	BLE	D5	4	2	F5	5	3	C5	2	2	E5	4+	2+							Branch ZV(N⊕V)=1	●	●	●	●	●
	LBLE																			Long Branch ZV(N⊕V)=1	●	●	●	●	●
BLO	BLO																			Branch C=1	●	●	●	●	●
	LBLO																			Long Branch C=1	●	●	●	●	●
BLS	BLS																			Branch CVZ=1	●	●	●	●	●
	LBLS																			Long Branch CVZ=1	●	●	●	●	●
BLT	BLT																			Branch N ⊕ V=1	●	●	●	●	●
	LBLT																			Long Branch N ⊕ V=1	●	●	●	●	●
BMI	BMI																			Branch N=1	●	●	●	●	●
	LBMI																			Long Branch N=1	●	●	●	●	●
BNE	BNE																			Branch Z = 0	●	●	●	●	●
	LBNE																			Long Branch Z = 0	●	●	●	●	●
BPL	BPL																			Branch N = 0	●	●	●	●	●
	LBPL																			Long Branch N = 0	●	●	●	●	●
BRA	BRA																			Branch Always	●	●	●	●	●
	LBRA																			Long Branch Always	●	●	●	●	●
BRN	BRN																			Branch Never	●	●	●	●	●
	LBRN																			Long Branch Never	●	●	●	●	●

(to be continued)

INSTRUCTION/ FORMS		HD6809 ADDRESSING MODES														DESCRIPTION	5 3 2 1 0								
		INHERENT			DIRECT			EXTENDED			IMMEDIATE			INDEXED <sup>①</sup>				H N Z V C							
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~ <sup>(5)</sup>	#	OP	~	#	H	N	Z
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	5(6)	4	Long Branch V = 0	•	•	•	•	•
BVS	BVS																29	3	2	Branch V = 1	•	•	•	•	•
	LBVS																10	5(6)	4	Long Branch V = 1	•	•	•	•	•
CLR	CLRA	4F	2	1	0F	6	2	7F	7	3						6F	6+	2+		0 → A	•	0	1	0	0
	CLRB	5F	2	1																0 → B	•	0	1	0	0
	CLR																		0 → M	•	0	1	0	0	
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	7	3	10	8	4	10	5	4	10	7+	3+						Compare M: M + 1 from D	•	↑	↑	↑	↑	
		93																	Compare M: M + 1 from S	•	↑	↑	↑	↑	
	CMPS	11	7	3	11	8	4	11	5	4	11	7+	3+						Compare M: M + 1 from U	•	↑	↑	↑	↑	
	CMPU	9C			BC			8C			AC								Compare M: M + 1 from X	•	↑	↑	↑	↑	
	CMPX	11	7	3	B3	8	4	11	5	4	A3	7+	3+						Compare M: M + 1 from Y	•	↑	↑	↑	↑	
	CMPY	9C	6	2	BC	7	3	8C	4	3	AC	6+	2+												
COM	COMA	43	2	1	03	6	2	73	7	3						63	6+	2+		Ā → A	•	↑	↑	0	1
	COMB	53	2	1															Ā → B	•	↑	↑	0	1	
	COM																	Ā → M	•	↑	↑	0	1		
CWAI		3C	20	2														CC ∧ IMM → CC (except 1→E)	←	⑦	→				
DAA		19	2	1														Wait for Interrupt							
DEC	DECA	4A	2	1	0A	6	2	7A	7	3						6A	6+	2+		Decimal Adjust A	•	↑	↑	⑧	↑
	DECB	5A	2	1															A → A	•	↑	↑	↑	•	
	DEC																	B → B	•	↑	↑	↑	•		
EOR	EORA	98	4	2	B8	5	3	88	2	2	A8	4+	2+					M → M	•	↑	↑	↑	•		
	EORB	D8	4	2	F8	5	3	C8	2	2	E8	4+	2+						A ⊕ M → A	•	↑	↑	0	•	
																		B ⊕ M → B	•	↑	↑	0	•		
EXG	R1, R2	1E	7	2														R1 ↔ R2 <sup>②</sup>	←	⑩	→				
INC	INCA	4C	2	1	0C	6	2	7C	7	3						6C	6+	2+		A + 1 → A	•	↑	↑	↑	•
	INCB	5C	2	1														B + 1 → B	•	↑	↑	↑	•		
	INC																	M + 1 → M	•	↑	↑	↑	•		
JMP		0E	3	2	7E	4	3									6E	3+	2+		EA <sup>③</sup> → PC	•	•	•	•	•
JSR		9D	7	2	BD	8	3									AD	7+	2+		Jump to Subroutine	•	•	•	•	•
LD	LDA	96	4	2	B6	5	3	86	2	2	A6	4+	2+					M → A	•	↑	↑	0	•		
	LDB	D6	4	2	F6	5	3	C6	2	2	E6	4+	2+					M → B	•	↑	↑	0	•		
	LDD	DC	5	2	FC	6	3	CC	3	3	EC	5+	2+					M: M + 1 → D	•	↑	↑	0	•		
	LDS	10	6	3	10	7	4	10	4	4	10	6+	3+					M: M + 1 → S	•	↑	↑	0	•		
	LDU	DE	5	2	FE	6	3	CE	3	3	EE	5+	2+					M: M + 1 → U	•	↑	↑	0	•		
	LDX	9E	5	2	BE	6	3	8E	3	3	AE	5+	2+					M: M + 1 → X	•	↑	↑	0	•		
	LDY	10	6	3	10	7	4	10	4	4	10	6+	3+					M: M + 1 → Y	•	↑	↑	0	•		
LEA	LEAS															32	4+	2+		EA <sup>④</sup> → S	•	•	•	•	•
	LEAU															33	4+	2+		EA <sup>④</sup> → U	•	•	•	•	•
	LEAX															30	4+	2+		EA <sup>④</sup> → X	•	•	•	•	•
	LEAY															31	4+	2+		EA <sup>④</sup> → Y	•	•	•	•	•
LSL	LSLA	48	2	1														A → B → M	•	↑	↑	↑	•		
	LSLB	58	2	1	08	6	2	78	7	3									•	↑	↑	↑	•		
	LSL																								
LSR	LSRA	44	2	1	04	6	2	74	7	3						64	6+	2+							
	LSRB	54	2	1																					
MUL		3D	11	1																					
NEG	NEGA	40	2	1	00	6	2	70	7	3						60	6+	2+		Ā + 1 → A	⑧	↑	↑	↑	↑
	NEGB	50	2	1															Ā + 1 → B	⑧	↑	↑	↑	↑	
	NEG																		M + 1 → M	⑧	↑	↑	↑	↑	
NOP		12	2	1															No Operation	•	•	•	•	•	

(to be continued)

INSTRUCTION/ FORMS			HD6809 ADDRESSING MODES														DESCRIPTION	5 3 2 1 0					
			INHERENT			DIRECT			EXTENDED			IMMEDIATE			INDEXED <sup>①</sup>				H	N	Z	V	C
OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#						
OR	ORA		9A	4	2	BA	5	3	8A	2	2	AA	4+	2+				A V M → A	•	†	†	0	•
	ORB		DA	4	2	FA	5	3	CA	2	2	EA	4+	2+				B V M → B	•	†	†	0	•
	ORCC		1A	3														CC V IMM → CC	(	—	(7)	—	)
PSH	PSHS	34	5+ <sup>④</sup>	2														Push Registers on S Stack	•	•	•	•	•
	PSHU	36	5+ <sup>④</sup>	2														Push Registers on U Stack	•	•	•	•	•
PUL	PULS	35	5+ <sup>④</sup>	2														Pull Registers from S Stack	(	—	(10)	—	)
	PULU	37	5+ <sup>④</sup>	2														Pull Registers from U Stack	(	—	(10)	—	)
ROL	ROLA	49	2	1														A	↓	↑	↑	↑	↑
	ROLB	59	2	1	09	6	2	79	7	3								B	↓	↑	↑	↑	↑
	ROL																	M	↓	↑	↑	↑	↑
ROR	RORA	46	2	1	06	6	2	76	7	3								A	↓	↑	↑	↑	↑
	RORB	56	2	1														B	↓	↑	↑	↑	↑
	ROR																	M	↓	↑	↑	↑	↑
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	•	†	↑	•	•
ST	STA		97	4	2	B7	5	3				A7	4+	2+				A → M	•	†	↑	0	•
	STB		D7	4	2	F7	5	3				E7	4+	2+				B → M	•	†	↑	0	•
	STD		DD	5	2	FD	6	3				ED	5+	2+				D → M: M + 1	•	†	↑	0	•
	STS		10	6	3	10	7	4				10	6+	3+				S → M: M + 1	•	†	↑	0	•
			DF			FF						EF						A → M: M + 1	•	†	↑	0	•
	STU		DF	5	2	FF	6	3				EF	5+	2+				X → M: M + 1	•	†	↑	0	•
	STX		9F	5	2	BF	6	3				AF	5+	2+				Y → M: M + 1	•	†	↑	0	•
	STY		10	6	3	10	7	4				10	6+	3+									
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	SWI <sup>⑤</sup>	3F	19	1														Software Interrupt1	•	•	•	•	•
	SWI2 <sup>⑥</sup>	10	20	2														Software Interrupt2	•	•	•	•	•
	SWI3 <sup>⑦</sup>	3F	11	20	2													Software Interrupt3	•	•	•	•	•
SYNC		13	≥2	1														Synchronize to Interrupt	•	•	•	•	•
TFR	R1, R2	1F	7	2														R1 → R2 <sup>⑧</sup>	(	—	(10)	—	)
TST	TSTA	4D	2	1	0D	6	2	7D	7	3								Test A	•	†	0	•	•
	TSTB	5D	2	1														Test B	•	†	0	•	•
	TST																	Test M	•	†	0	•	•

## (NOTES)

- ① This column gives a base cycle and byte count. To obtain total count, add 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(6) means: 5 cycles if branch not taken, 6 cycles if taken.
- ⑥ SWI sets 1 and F bits. SWI2 and SWI3 do not affect I and F.
- ⑦ Condition 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
M	Complement of M	:	Concatenation
→	Transfer Into	∨	Logical or
H	Half-carry (from bit 3)	∧	Logical and
N	Negative (sign bit)	⊕	Logical Exclusive or

Table 11 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	Inherent	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		6, 15	1	6B	*			
0C	INC		6	2	3C	CWAI		20	2	6C	INC		6+	2+
0D	TST		6	2	3D	MUL		11	1	6D	TST		6+	2+
0E	JMP		3	2	3E	*				6E	JMP		3+	2+
0F	CLR	Direct	6	2	3F	SWI	Inherent	19	1	6F	CLR	Indexed	6+	2+
10	See Next Page	—	—	—	40	NEGA	Inherent	2	1	70	NEG	Extended	7	3
11		—	—	—	41	*				71	*			
12	NOP	Inherent	2	1	42	*				72	*			
13	SYNC	Inherent	2	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	Inherent	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	Inherent	2	1	4D	TSTA		2	1	7D	TST		7	3
1E	EXG		8	2	4E	*				7E	JMP		4	3
1F	TFR	Inherent	6	2	4F	CLRA	Inherent	2	1	7F	CLR	Extended	7	3
20	BRA	Relative	3	2	50	NEGB	Inherent	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	ASRA		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	DEC B		2	1	8A	ORA		2	2
2B	BMI		3	2	5B	*				8B	ADDA		2	2
2C	BGE		3	2	5C	INC B		2	1	8C	CMPX	Immed	4	3
2D	BLT		3	2	5D	TST B		2	1	8D	BSR	Relative	7	2
2E	BGT		3	2	5E	*				8E	LDX	Immed	3	3
2F	BLE	Relative	3	2	5F	CLRB	Inherent	2	1	8F	*			

## LEGEND:

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

(to be continued)

**HD6809, HD68A09, HD68B09**

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

(NOTE): All unused opcodes are both undefined and illegal

HITACHI reserves the right to make changes to any products herein to improve functioning or design. Although the information in this document has been carefully reviewed and is believed to be reliable, HITACHI does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others.

# HD6821, HD68A21, HD68B21

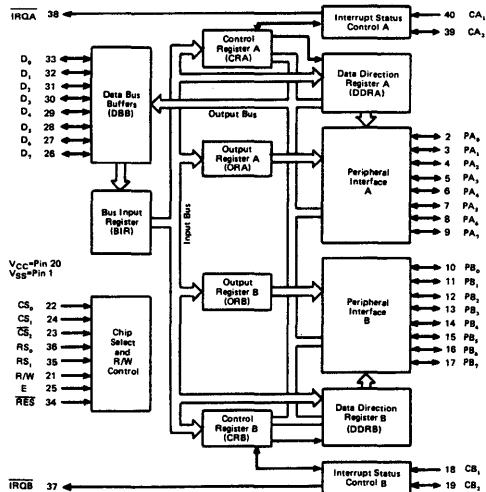
## PIA (Peripheral Interface Adapter)

The HD6821 Peripheral Interface Adapter provides the universal means of interfacing peripheral equipment to the HD6800 Microprocessing Unit(MPU). This device is capable of interfacing the MPU to peripherals through two 8-bit bi-directional peripheral data buses and four control lines. No external logic is required for interfacing to most peripheral devices.

### ■ FEATURES

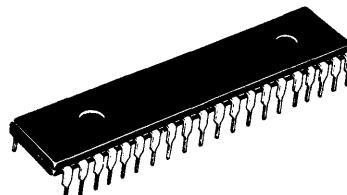
- Two Bi-directional 8-Bit Peripheral Data Bus for interface to Peripheral devices
- Two Programmable Control Registers
- Two Programmable Data Direction Registers
- Four Individually-Controlled Interrupt Input Lines: Two Usable as Peripheral Control Outputs
- Handshake Control Logic for Input and Output Peripheral Operation
- High-Impedance 3-State and Direct Transistor Drive Peripheral Lines
- Program Controlled Interrupt and Interrupt Disable Capability
- CMOS Drive Capability on Side A Peripheral Lines
- Two TTL Drive Capability on All A and B Side Buffers
- N Channel Silicon Gate MOS
- Compatible with MC6821, MC68A21 and MC68B21

### ■ BLOCK DIATRAM



The functional configuration of the PIA is programmed by the MPU during system initialization. Each of the peripheral data lines can be programmed to act as an input or output, and each of the four control/interrupt lines may be programmed for one of several control modes. This allows a high degree of flexibility in the over-all operation of the interface.

HD6821P, HD68A21P, HD68B21P



(DP-40)

### ■ PIN ARRANGEMENT

V <sub>SS</sub>	1	CA <sub>1</sub>
PA <sub>0</sub>	2	CA <sub>2</sub>
PA <sub>1</sub>	3	IRQA
PA <sub>2</sub>	4	IRQB
PA <sub>3</sub>	5	RS <sub>0</sub>
PA <sub>4</sub>	6	RS <sub>1</sub>
PA <sub>5</sub>	7	RES
PA <sub>6</sub>	8	D <sub>0</sub>
PA <sub>7</sub>	9	D <sub>1</sub>
PB <sub>0</sub>	10	D <sub>2</sub>
PB <sub>1</sub>	11	D <sub>3</sub>
PB <sub>2</sub>	12	D <sub>4</sub>
PB <sub>3</sub>	13	D <sub>5</sub>
PB <sub>4</sub>	14	D <sub>6</sub>
PB <sub>5</sub>	15	E
PB <sub>6</sub>	16	CS <sub>1</sub>
PB <sub>7</sub>	17	CS <sub>2</sub>
CB <sub>1</sub>	18	CS <sub>0</sub>
CB <sub>2</sub>	19	R/W
V <sub>CC</sub>	20	

(Top View)

**■ ABSOLUTE MAXIMUM RATINGS**

Item	Symbol	Value	Unit
Supply Voltage	$V_{CC}^*$	-0.3 ~ +7.0	V
Input Voltage	$V_{in}^*$	-0.3 ~ +7.0	V
Operating Temperature	$T_{opr}$	-20 ~ +75	°C
Storage Temperature	$T_{stg}$	-55 ~ +150	°C

\* With respect to  $V_{SS}$  (SYSTEM GND)

(NOTE) 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.

**■ RECOMMENDED OPERATING CONDITIONS**

Item	Symbol	min	typ	max	Unit
Supply Voltage	$V_{CC}^*$	4.75	5	5.25	V
Input Voltage	$V_{IL}^*$	-0.3	—	0.8	V
	$V_{IH}^*$	2.0	—	$V_{CC}$	
Operating Temperature	$T_{opr}$	-20	25	75	°C

\* With respect to  $V_{SS}$  (SYSTEM GND)**■ ELECTRICAL CHARACTERISTICS**

- DC CHARACTERISTICS ( $V_{CC}=5.0V \pm 5\%$ ,  $V_{SS}=0V$ ,  $T_a=-20 \sim +75^\circ C$ , unless otherwise noted.)

Item	Symbol	Test Condition	min	typ*	max	Unit
Input "High" Voltage	All Inputs	$V_{IH}$	2.0	—	$V_{CC}$	V
Input "Low" Voltage	All Inputs	$V_{IL}$	-0.3	—	0.8	V
Input Leakage Current	R/W, RES, RS <sub>0</sub> , RS <sub>1</sub> , CS <sub>0</sub> , CS <sub>1</sub> , CS <sub>2</sub> , CA <sub>1</sub> , CB <sub>1</sub> , E	$I_{in}$	$V_{in}=0 \sim 5.25V$	—	—	2.5 $\mu A$
Three-State (Off State) Input Current	D <sub>0</sub> ~D <sub>7</sub> , PB <sub>0</sub> ~PB <sub>7</sub> , CB <sub>2</sub>	$I_{TSI}$	$V_{in}=0.4 \sim 2.4V$	—	—	10 $\mu A$
Input "High" Current	PA <sub>0</sub> ~PA <sub>7</sub> , CA <sub>2</sub>	$I_{IH}$	$V_{IH}=2.4V$	-200	—	— $\mu A$
Input "Low" Current	PA <sub>0</sub> ~PA <sub>7</sub> , CA <sub>2</sub>	$I_{IL}$	$V_{IL}=0.4V$	—	—	-2.4 mA
Output "High" Voltage	D <sub>0</sub> ~D <sub>7</sub>	$V_{OH}$	$I_{OH}=205\mu A$	2.4	—	V
	PA <sub>0</sub> ~PA <sub>7</sub> , CA <sub>2</sub>		$I_{OH}=200\mu A$	2.4**	—	
	Other Outputs		$I_{OH}=10\mu A$	$V_{CC}-1.0$	—	
			$I_{OH}=200\mu A$	2.4	—	
Output "Low" Voltage	D <sub>0</sub> ~D <sub>7</sub> , IRQA, IRQB	$V_{OL}$	$I_{OL}=1.6mA$	—	—	V
	Other Outputs		$I_{OL}=1.6mA$	—	—	
			$I_{OL}=3.2mA$	—	—	
Output "High" Current	D <sub>0</sub> ~D <sub>7</sub>	$I_{OH}$	$V_{OH}=2.4V$	-205	—	$\mu A$
	Other Outputs		$V_{OH}=2.4V^{**}$	-200	—	
	PA <sub>0</sub> ~PA <sub>7</sub> , CA <sub>2</sub>		$V_{OH}=1.5V$	-1.0	—	
	PB <sub>0</sub> ~PB <sub>7</sub> , CB <sub>2</sub>		$V_{OH}=2.4V$	—	10	mA
Output Leakage Current (Off State)	IRQA, IRQB	$I_{LOH}$	$V_{OH}=2.4V$	—	—	$\mu A$
Power Dissipation	$P_D$		—	260	550	mW
Input Capacitance	PA <sub>0</sub> ~PA <sub>7</sub> , PB <sub>0</sub> ~PB <sub>7</sub> , CA <sub>2</sub> , CB <sub>2</sub> , D <sub>0</sub> ~D <sub>7</sub>	$C_{in}$	$V_{in}=0V$ , $T_a=25^\circ C$	—	—	12.5 pF
	R/W, RES, RS <sub>0</sub> , RS <sub>1</sub> , CS <sub>0</sub> , CS <sub>1</sub> , CS <sub>2</sub> , CA <sub>1</sub> , CB <sub>1</sub> , E		f=1.0MHz	—	—	10 pF
Output Capacitance	IRQA, IRQB	$C_{out}$	$V_{in}=0V$ , $T_a=25^\circ C$ , f=1.0MHz	—	—	10 pF

\*  $T_a=25^\circ C$ ,  $V_{CC}=5.0V$ \*\* HD468B21 ;  $V_{OH}=2.2V$  min (PA<sub>0</sub>~PA<sub>7</sub>, CA<sub>2</sub>)

## ● AC CHARACTERISTICS

1. PERIPHERAL TIMING ( $V_{CC}=5.0V \pm 5\%$ ,  $V_{SS}=0$ ,  $T_a=-20 \sim +75^\circ C$ , unless otherwise noted.)

Item	Symbol	Test Condition	HD6821		HD68A21		HD68B21		Unit	
			min	max	min	max	min	max		
Peripheral Data Setup Time	$t_{PDSU}$	Fig. 1	200	—	135	—	100	—	ns	
Peripheral Data Hold Time	$t_{PDH}$	Fig. 1	0	—	0	—	0	—	ns	
Delay Time, Enable negative transition to $CA_2$ , negative transition	Enable $\rightarrow CA_2$ , Negative	$t_{CA2}$	Fig. 2, Fig. 3	—	1.0	—	0.67	—	$\mu s$	
Delay Time, Enable negative transition to $CA_2$ , positive transition	Enable $\rightarrow CA_2$ , Positive	$t_{RS1}$	Fig. 2	—	1.0	—	0.67	—	$\mu s$	
Rise and Fall Times for $CA_1$ and $CA_2$ input signals	$CA_1, CA_2$	$t_r, t_f$	Fig. 3	—	1.0	—	1.0	—	$\mu s$	
Delay Time from $CA_1$ , active transition to $CA_2$ , positive transition	$CA_1 - CA_2$	$t_{RS2}$	Fig. 3	—	2.0	—	1.35	—	$\mu s$	
Delay Time, Enable negative transition to Peripheral Data Valid	Enable $\rightarrow$ Peripheral Data	$t_{PDW}$	Fig. 4, Fig. 5	—	1.0	—	0.67	—	$\mu s$	
Delay Time, Enable negative transition to Peripheral CMOS Data Valid	Enable $\rightarrow$ Peripheral Data $PA_0 \sim PA_7, CA_2$	$t_{CMOS}$	$V_{CC} - 30\% V_{CC}$ Fig. 4	—	2.0	—	1.35	—	$\mu s$	
Delay Time, Enable positive transition to $CB_2$ , negative position	Enable $\rightarrow CB_2$	$t_{CB2}$	Fig. 6, Fig. 7	—	1.0	—	0.67	—	$\mu s$	
Delay Time, Peripheral Data Valid to $CB_2$ negative transition	Peripheral Data $\rightarrow CB_2$	$t_{DC}$	Fig. 5	20	—	20	—	20	—	ns
Delay Time, Enable positive transition to $CB_2$ , positive transition	Enable $\rightarrow CB_2$	$t_{RS1}$	Fig. 6	—	1.0	—	0.67	—	$\mu s$	
Peripheral Control Output Pulse Width, $CA_2/CB_2$	$CA_2, CB_2$	$PW_{CT}$	Fig. 2, Fig. 6	550	—	550	—	500	—	ns
Rise and Fall Time for $CB_1$ and $CB_2$ input signals	$CB_1, CB_2$	$t_r, t_f$	Fig. 7	—	1.0	—	1.0	—	$\mu s$	
Delay Time, $CB_1$ active transition to $CB_2$ , positive transition	$CB_1 \rightarrow CB_2$	$t_{RS2}$	Fig. 7	—	2.0	—	1.35	—	$\mu s$	
Interrupt Release Time, $IRQA$ and $IRQB$	$IRQA, IRQB$	$t_{IR}$	Fig. 9	—	1.6	—	1.1	—	$\mu s$	
Interrupt Response Time	$IRQA, IRQB$	$t_{RS3}$	Fig. 8	—	1.0	—	1.0	—	$\mu s$	
Interrupt Input Pulse Width	$CA_1, CA_2, CB_1, CB_2$	$PWI$	Fig. 8	500**	—	500**	—	500**	—	ns
Reset "Low" Time	$RES^*$	$t_{RL}$	Fig. 10	1.0	—	0.66	—	0.5	—	$\mu s$

\* The Reset line must be "High" a minimum of  $1.0\mu s$  before addressing the PIA.

\*\* At least one Enable "High" pulse should be included in this period.

---

**HD6821, HD68A21, HD68B21**


---

**2. BUS TIMING**
**1) READ**

Item	Symbol	Test Condition	HD6821		HD68A21		HD68B21		Unit	
			min	max	min	max	min	max		
Enable Cycle Time	$t_{cycE}$	Fig. 11	1000	—	666	—	500	—	ns	
Enable Pulse Width, "High"	$PW_{EH}$	Fig. 11	450	—	280	—	220	—	ns	
Enable Pulse Width, "Low"	$PW_{EL}$	Fig. 11	430	—	280	—	210	—	ns	
Enable Pulse Rise and Fall Times	$t_{Er}, t_{Ef}$	Fig. 11	—	25	—	25	—	25	ns	
Setup Time	Address, R/W—Enable		$t_{AS}$	Fig. 13	140	—	140	—	70	—
Address Hold Time		$t_{AH}$	Fig. 13	10	—	10	—	10	—	ns
Data Delay Time		$t_{DDR}$	Fig. 13	—	320	—	220	—	180	ns
Data Hold Time		$t_{DHR}$	Fig. 13	10	—	10	—	10	—	ns

**2) WRITE**

Item	Symbol	Test Condition	HD6821		HD68A21		HD68B21		Unit		
			min	max	min	max	min	max			
Enable Cycle Time	$t_{cycE}$	Fig. 11	1000	—	666	—	500	—	ns		
Enable Pulse Width, "High"	$PW_{EH}$	Fig. 11	450	—	280	—	220	—	ns		
Enable Pulse Width, "Low"	$PW_{EL}$	Fig. 11	430	—	280	—	210	—	ns		
Enable Pulse Rise and Fall Times	$t_{Er}, t_{Ef}$	Fig. 11	—	25	—	25	—	25	ns		
Setup Time		$t_{AS}$	Fig. 12	140	—	140	—	70	—	ns	
Address Hold Time	Address, R/W—Enable		$t_{AH}$	Fig. 12	10	—	10	—	10	—	ns
Data Setup Time		$t_{DSW}$	Fig. 12	195	—	80	—	60	—	ns	
Data Hold Time		$t_{DHW}$	Fig. 12	10	—	10	—	10	—	ns	

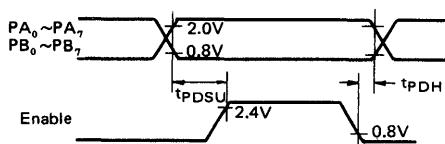


Figure 1 Peripheral Data Setup and Hold Times (Read Mode)

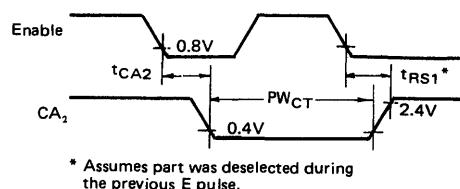


Figure 2 CA<sub>2</sub> Delay Time  
(Read Mode; CRA5=CRA3=1, CRA4=0)

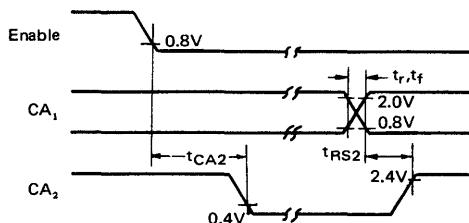


Figure 3 CA<sub>2</sub> Delay Time  
(Read Mode; CRA5=CRA3=CRA4=0)

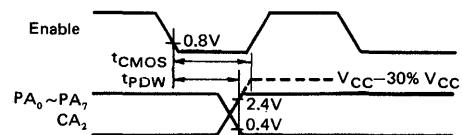


Figure 4 Peripheral CMOS Data Delay Times  
(Write Mode; CRA5=CRA3=1, CRA4=0)

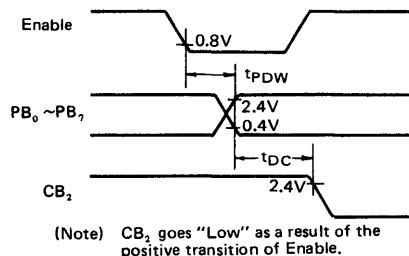
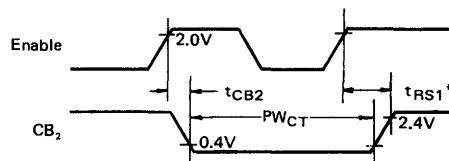
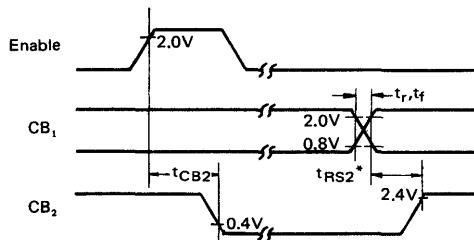


Figure 5 Peripheral Data and CB<sub>2</sub> Delay Times  
(Write Mode; CRB5=CRB3=1, CRB4=0)



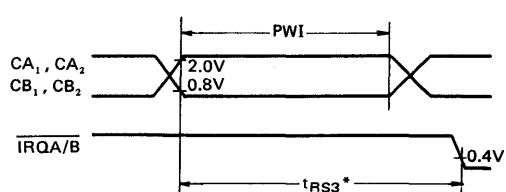
\* Assumes part was deselected during the previous E pulse.

Figure 6 CB<sub>2</sub> Delay Time  
(Write Mode; CRB5=CRB3=1, CRB4=0)



\* Assumes part was deselected during any previous E pulse.

Figure 7 CB<sub>2</sub> Delay Time  
(Write Mode; CRB5=1, CRB3=CRB4=0)



\* Assumes interrupt enable bits are set.

Figure 8 Interrupt Pulse Width and TRO Response

**HD6821, HD68A21, HD68B21**

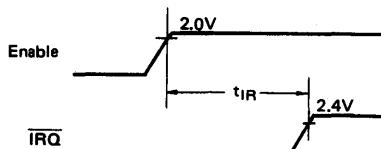
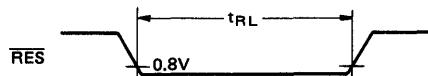


Figure 9  $\overline{\text{IRQ}}$  Release Time



\* The  $\overline{\text{RES}}$  line must be a  $V_{IH}$  for a minimum of 1.0  $\mu\text{s}$  before addressing the PIA.

Figure 10  $\overline{\text{RES}}$  Low Time

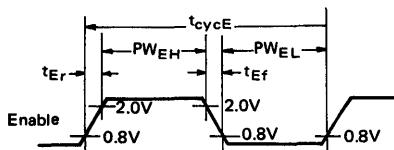


Figure 11 Enable Signal Characteristics

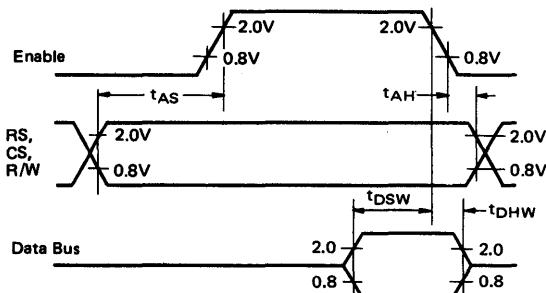


Figure 12 Bus Write Timing Characteristics  
(Write Information into PIA)

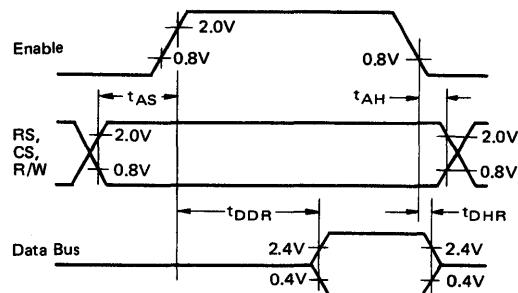
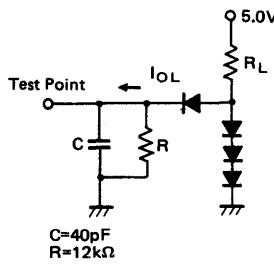


Figure 13 Bus Read Timing Characteristics  
(Read Information from PIA)

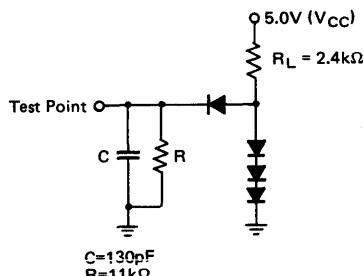
**LOAD A**  
( $\text{PA}_0 \sim \text{PA}_7$ ,  $\text{PB}_0 \sim \text{PB}_7$ ,  $\text{CA}_2$ ,  $\text{CB}_2$ )



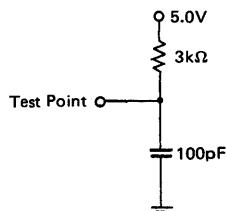
All diodes are IS2074 (H).

Adjust  $R_L$  so that  $I_{OL} = 3.2\text{mA}$ , then test  $V_{OL}$ .

**LOAD B**  
( $D_0 \sim D_7$ )



**LOAD C**  
( $\overline{\text{IRQ}}$  Only)



**LOAD D**

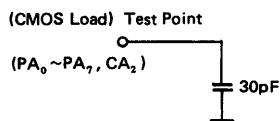


Figure 14 Bus Timing Test Loads

## ■ PIA INTERFACE SIGNALS FOR MPU

The PIA interfaces to the HD6800 MPU with an eight-bit bi-directional data bus, three chip select lines, two register select lines, two interrupt request lines, read/write line, enable line and reset line. These signals, in conjunction with the HD6800 VMA output, permit the MPU to have complete control over the PIA. VMA should be utilized in conjunction with an MPU address line into a chip select of the PIA.

- **PIA Bi-Directional Data ( $D_0 \sim D_7$ )**

The bi-directional data lines ( $D_0 \sim D_7$ ) allow the transfer of data between the MPU and the PIA. The data bus output drivers are three-state devices that remain in the high-impedance (off) state except when the MPU performs a PIA read operation. The R/W line is in the Read ("High") state when the PIA is selected for a Read operation.

- **PIA Enable (E)**

The enable pulse, E, is the only timing signal that is supplied to the PIA. Timing of all other signals is referenced to the leading and trailing edges of the E pulse. This signal will normally be a derivative of the HMCS6800 System  $\phi_2$  Clock. This signal must be continuous clock pulse.

- **PIA Read/Write (R/W)**

This signal is generated by the MPU to control the direction of data transfers on the Data Bus. A "Low" state on the PIA line enables the input buffers and data is transferred from the MPU to the PIA on the E signal if the device has been selected. A "High" on the R/W line sets up the PIA for a transfer of data to the bus. The PIA output buffers are enabled when the proper address and the enable pulse E are present.

- **Reset ( $\overline{RES}$ )**

The active "Low"  $\overline{RES}$  line is used to reset all register bits in the PIA to a logical zero "Low". This line can be used as a power-on reset and as a master reset during system operation.

- **PIA Chip Select ( $CS_0$ ,  $CS_1$  and  $CS_2$ )**

These three input signals are used to select the PIA.  $CS_0$  and  $CS_1$  must be "High" and  $CS_2$  must be "Low" for selection of the device. Data transfers are then performed under the control of the E and R/W signals. The chip select lines must be stable for the duration of the E pulse. The device is deselected when any of the chip selects are in the inactive state.

- **PIA Register Select ( $RS_0$  and  $RS_1$ )**

The two register select lines are used to select the various registers inside the PIA. These two lines are used in conjunction with internal Control Registers to select a particular register that is to be written or read.

The register and chip select lines should be stable for the duration of the E pulse while in the read or write cycle.

- **Interrupt Request ( $\overline{IRQA}$  and  $\overline{IRQB}$ )**

The active "Low" Interrupt Request lines ( $\overline{IRQA}$  and  $\overline{IRQB}$ ) act to interrupt the MPU either directly or through interrupt priority circuitry. These lines are "open drain" (no load device on the chip). This permits all interrupt request lines to be tied together in a wire-OR configuration.

Each  $\overline{IRQ}$  line has two internal interrupt flag bits that can cause the IRQ line to go "Low". Each flag bit is associated with a particular peripheral interrupt line. Also four interrupt enable bits are provided in the PIA which may be used to inhibit a particular interrupt from a peripheral device.

Servicing an interrupt by the MPU may be accomplished by a software routine that, on a prioritized basis, sequentially reads and tests the two control registers in each PIA for interrupt flag bits that are set.

The interrupt flags are cleared (zeroed) as a result of an MPU Read Peripheral Data Operation of the corresponding data register. After being cleared, the interrupt flag bit cannot be enabled to be set until the PIA is deselected during an E pulse. The E pulse is used to condition the interrupt control lines ( $CA_1$ ,  $CA_2$ ,  $CB_1$ ,  $CB_2$ ). When these lines are used as interrupt inputs at least one E pulse must occur from the inactive edge to the active edge of the interrupt input signal to condition the edge sense network. If the interrupt flag has been enabled and the edge sense circuit has been properly conditioned, the interrupt flag will be set on the next active transition of the interrupt input pin.

## ■ PIA PERIPHERAL INTERFACE LINES

The PIA provides two 8-bit bi-directional data buses and four interrupt/control lines for interfacing to peripheral devices.

- **Section A Peripheral Data ( $PA_0 \sim PA_7$ )**

Each of the peripheral data lines can be programmed to act as an input or output. This is accomplished by setting a "1" in the corresponding Data Direction Register bit for those lines which are to be outputs. A "0" in a bit of the Data Direction Register causes the corresponding peripheral data line to act as an input. During an MPU Read Peripheral Data Operation, the data on peripheral lines programmed to act as inputs appears directly on the corresponding MPU Data Bus lines.

The data in Output Register A will appear on the data lines that are programmed to be outputs. A logical "1" written into the register will cause a "High" on the corresponding data line while a "0" results in a "Low". Data in Output Register A may be read by an MPU "Read Peripheral Data A" operation when the corresponding lines are programmed as outputs. This data will be read properly if the voltage on the peripheral data lines is greater than 2.0 volts for a logic "1" output and less than 0.8 volt for a logic "0" output. Loading the output lines such that the voltage on these lines does not reach full voltage causes the data transferred into the MPU on a Read operation to differ from that contained in the respective bit of Output Register A.

- **Section B Peripheral Data ( $PB_0 \sim PB_7$ )**

The peripheral data lines in the B Section of the PIA can be programmed to act as either inputs or outputs in a similar manner to  $PA_0 \sim PA_7$ . However, the output buffers driving these lines differ from those driving lines  $PA_0 \sim PA_7$ . They have three-state capability, allowing them to enter a high impedance state when the peripheral data line is used as an input. In addition, data on the peripheral data lines  $PB_0 \sim PB_7$  will be read properly from those lines programmed as outputs even if the voltages are below 2.0 volts for a "High". As outputs, these lines are compatible with standard TTL and may also be used as a source of up to 2.5 milliampere (typ.) at 1.5 volts to directly drive the base of a transistor switch.

- **Interrupt Input ( $CA_1$  and  $CB_1$ )**

Peripheral Input lines  $CA_1$  and  $CB_1$  are input only lines that set the interrupt flags of the control registers. The active transition for these signals is also programmed by the two control registers.

- **Peripheral Control ( $CA_2$ )**

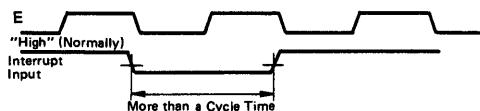
The peripheral control line  $CA_2$  can be programmed to act as an interrupt input or as a peripheral control output. As an output, this line is compatible with standard TTL. The function of this signal line is programmed with Control Register A.

- **Peripheral Control ( $CB_2$ )**

Peripheral Control line  $CB_2$  may also be programmed to act as an interrupt input or peripheral control output. As an input,

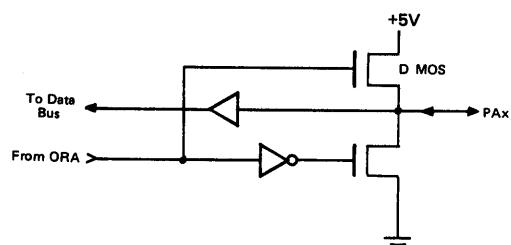
this line has "High" input impedance and is compatible with standard TTL. As an output it is compatible with standard TTL and may also be used as a source of up to 2.5 milliamper (typ) at 1.5 volts to directly drive the base of a transistor switch. This line is programmed by Control Register B.

- (NOTE)** 1. Interrupt inputs CA<sub>1</sub>, CA<sub>2</sub>, CB<sub>1</sub> and CB<sub>2</sub> shall be used at normal "High" level. When interrupt inputs are "Low" at reset (RES = "Low"), interrupt flags CRA6, CRA7, CRB6 and CRB7 may be set.  
 2. Pulse width of interrupt inputs CA<sub>1</sub>, CA<sub>2</sub>, CB<sub>1</sub> and CB<sub>2</sub> shall be greater than a E cycle time. In the case that "High" time of E signal is not contained in Interrupt pulse, an interrupt flag may not be set.

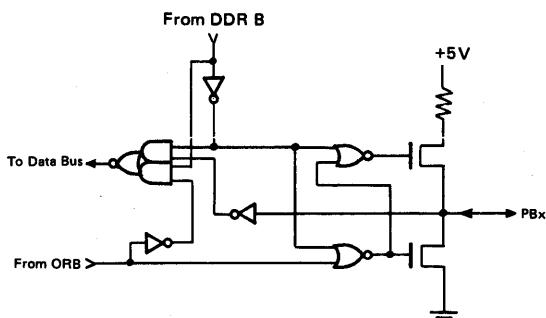


#### • The equivalent Circuit of the Lines on Peripheral side

The equivalent circuit of the lines on Peripheral side is shown in Fig. 15. The output circuits of A port is different from that of B port. When the port is used as input, the input is pullup to V<sub>CC</sub> side through load MOS in A port and B port becomes "Off" (high impedance).



(a) Section A



(b) Section B

Figure 15 Peripheral Data Bus

#### ■ INTERNAL CONTROLS

There are six locations within the PIA accessible to the MPU data bus: two Peripheral Registers, two Data Direction Registers, and two Control Registers. Selection of these locations is controlled by the RS<sub>0</sub> and RS<sub>1</sub> inputs together with bit 2 in the Control Register, as shown in Table 1.

Table 1 Internal Addressing

RS <sub>1</sub>	RS <sub>0</sub>	Control Register Bit		Location Selected
		CRA2	CRB2	
0	0	1	x	Peripheral Register A*
0	0	0	x	Data Direction Register A
0	1	x	x	Control Register A
1	0	x	1	Peripheral Register B*
1	0	x	0	Data Direction Register B
1	1	x	x	Control Register B

x = Don't Care

\* Peripheral interface register is a generic term containing peripheral data bus and output register.

#### • Initialization

A "Low" reset line has the effect of zeroing all PIA registers. This will set PA<sub>0</sub>~PA<sub>7</sub>, PB<sub>0</sub>~PB<sub>7</sub>, CA<sub>2</sub> and CB<sub>2</sub> as inputs, and all interrupts disabled. The PIA must be configured during the restart program which follows the reset.

Details of possible configurations of the Data Direction and Control Register are as follows.

#### • Data Direction Registers (DDRA and DDRB)

The two Data Direction Registers allow the MPU to control the direction of data through each corresponding peripheral data line. A Data Direction Register bit set at "0" configures the corresponding peripheral data line as an input; a "1" results in an output.

#### • Control Registers (CRA and CRB)

The two Control Registers (CRA and CRB) allow the MPU to control the operation of the four peripheral control lines CA<sub>1</sub>, CA<sub>2</sub>, CB<sub>1</sub> and CB<sub>2</sub>. In addition they allow the MPU to enable the interrupt lines and monitor the status of the interrupt flags. Bits 0 through 5 of the two registers may be written or read by the MPU when the proper chip select and register select signals are applied. Bits 6 and 7 of the two registers are read only and are modified by external interrupts occurring on control lines CA<sub>1</sub>, CA<sub>2</sub>, CB<sub>1</sub> or CB<sub>2</sub>. The format of the control words is shown in Table 2.

Table 2 Control Word Format

	7	6	5	4	3	2	1	0
CRA	IRQA1	IRQA2	CA <sub>2</sub> Control			DDRA Access	CA <sub>1</sub> Control	
CRB	IRQB1	IRQB2	CB <sub>2</sub> Control			DDR Access	CB <sub>1</sub> Control	

**Data Direction Access Control Bit (CRA2 and CRB2)**

Bit 2 in each Control register (CRA and CRB) allows selection of either a Peripheral Interface Register or the Data Direction Register when the proper register select signals are applied to RS<sub>0</sub> and RS<sub>1</sub>.

**Interrupt Flags (CRA6, CRA7, CRB6, and CRB7)**

The four interrupt flag bits are set by active transitions of signals on the four Interrupt and Peripheral Control lines when those lines are programmed to be inputs. These bits cannot be set directly from the MPU Data Bus and are reset indirectly by a Read Peripheral Data Operation on the appropriate section.

**Control of CA<sub>1</sub> and CB<sub>1</sub> Interrupt Lines (CRA0, CRB0, CRA1, and CRB1)**

The two lowest order bits of the control registers are used to control the interrupt input lines CA<sub>1</sub> and CB<sub>1</sub>. Bits CRA0 and

CRB0 are used to enable the MPU interrupt signals IRQA and IRQB, respectively. Bits CRA1 and CRB1 determine the active transition of the interrupt input signals CA<sub>1</sub> and CB<sub>1</sub> (Table 3)

**Control of CA<sub>2</sub> and CB<sub>2</sub> Peripheral Control Lines (CRA3, CRA4, CRA5, CRB3, CRB4, and CRB5)**

Bits 3, 4 and 5 of the two control registers are used to control the CA<sub>2</sub> and CB<sub>2</sub> Peripheral Control lines. These bits determine if the control lines will be an interrupt input or an output control signal. If bit CRA5 (CRB5) is "0" CA<sub>2</sub> (CB<sub>2</sub>) is an interrupt input line similar to CA<sub>1</sub> (CB<sub>1</sub>) (Table 4). When CRA5 (CRB5) is "1", CA<sub>2</sub> (CB<sub>2</sub>) becomes an output signal that may be used to control peripheral data transfers. When in the output mode, CA<sub>2</sub> and CB<sub>2</sub> have slightly different characteristics (Table 5 and 6).

Table 3 Control of Interrupt Inputs CA<sub>1</sub> and CB<sub>1</sub>

CRA1 (CRB1)	CRA0 (CRB0)	Interrupt Input CA <sub>1</sub> (CB <sub>1</sub> )	Interrupt Flag CRA7 (CRB7)	MPU Interrupt Request IRQA (IRQB)
0	0	↓ Active	Set "1" on ↓ of CA <sub>1</sub> (CB <sub>1</sub> )	Disabled – IRQ remains "High"
0	1	↓ Active	Set "1" on ↓ of CA <sub>1</sub> (CB <sub>1</sub> )	Goes "Low" when the inter- rupt flag bit CRA7 (CRB7) goes "1"
1	0	↑ Active	Set "1" on ↑ of CA <sub>1</sub> (CB <sub>1</sub> )	Disabled – IRQ remains "High"
1	1	↑ Active	Set "1" on ↑ of CA <sub>1</sub> (CB <sub>1</sub> )	Goes "Low" when the inter- rupt flag bit CRA7 (CRB7) goes "1".

- (Notes)
1. ↑ indicates positive transition ("Low" to "High")
  2. ↓ indicates negative transition ("High" to "Low")
  3. The Interrupt flag bit CRA7 is cleared by an MPU Read of the A Peripheral Register and CRB7 is cleared by an MPU Read of the B Peripheral Register.
  4. If CRA0 (CRB0) is "0" when an interrupt occurs (Interrupt disabled) and is later brought "1", IRQA (IRQB) occurs after CRA0 (CRB0) is written to a "1".

Table 4 Control of CA<sub>2</sub> and CB<sub>2</sub> as Interrupt Inputs – CRA5 (CRB5) is "0"

CRA5 (CRB5)	CRA4 (CRB4)	CRA3 (CRB3)	Interrupt Input CA <sub>2</sub> (CB <sub>2</sub> )	Interrupt Flag CRA6 (CRB6)	MPU Interrupt Request IRQA (IRQB)
0	0	0	↓ Active	Set "1" on ↓ of CA <sub>2</sub> (CB <sub>2</sub> )	Disabled – IRQ remains "High"
0	0	1	↓ Active	Set "1" on ↓ of CA <sub>2</sub> (CB <sub>2</sub> )	Goes "Low" when the inter- rupt flag bit CRA6 (CRB6) goes "1"
0	1	0	↑ Active	Set "1" on ↑ of CA <sub>2</sub> (CB <sub>2</sub> )	Disabled – IRQ remains "High"
0	1	1	↑ Active	Set "1" on ↑ of CA <sub>2</sub> (CB <sub>2</sub> )	Goes "Low" when the inter- rupt flag bit CRA6 (CRB6) goes "1"

- (Notes)
1. ↑ indicates positive transition ("Low" to "High")
  2. ↓ indicates negative transition ("High" to "Low")
  3. The interrupt flag bit CRA6 is cleared by an MPU Read of the A Peripheral Register and CRB6 is cleared by an MPU Read of the B Peripheral Register.
  4. If CRA3 (CRB3) is "0" when an interrupt occurs (Interrupt disabled) and is later brought "1", IRQA (IRQB) occurs after CRA3 (CRB3) is written to a "1".

Table 5 Control of CB<sub>2</sub> as an Output — CRB5 is "1"

CRB5	CRB4	CRB3	CB <sub>2</sub>	
			Cleared	Set
1	0	0	"Low" on the positive transition of the first E pulse after MPU Write "B" Data Register operation.	"High" when the interrupt flag bit CRB7 is set by an active transition of the CB <sub>1</sub> signal. (See Figure 16)
1	0	1	"Low" on the positive transition of the first E pulse after an MPU Write "B" Data Register operation.	"High" on the positive edge of the first "E" pulse following an "E" pulse which occurred while the part was deselected. (See Figure 16)
1	1	0	"Low" (The content of CRB3 is output on CB <sub>2</sub> )	
1	1	1	"High" (The content of CRB3 is output on CB <sub>2</sub> )	

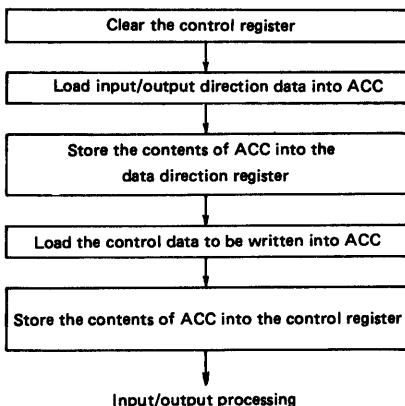
Table 6 Control of CA<sub>2</sub> as an Output — CRA5 is "1"

CRA5	CRA4	CRA3	CA <sub>2</sub>	
			Cleared	Set
1	0	0	"Low" on negative transition of E after an MPU Read "A" Data Operation.	"High" when the interrupt flag bit CRA7 is set by an active transition of the CA <sub>1</sub> signal. (See Figure 16)
1	0	1	"Low" on negative transition of E after an MPU Read "A" Data operation.	"High" on the negative edge of the first "E" pulse which occurs during a deselect. (See Figure 16)
1	1	0	"Low" (The content of CRA3 is output on CA <sub>2</sub> )	
1	1	1	"High" (The content of CRA3 is output on CA <sub>2</sub> )	

## ■ PIA OPERATION

### ● Initialization

When the external reset input  $\overline{RES}$  goes “Low”, all internal registers are cleared to “0”. Peripheral data port ( $PA_0 \sim PA_7$ ,  $PB_0 \sim PB_7$ ) is defined to be input and control lines ( $CA_1$ ,  $CA_2$ ,  $CB_1$  and  $CB_2$ ) are defined to be the interrupt input lines. PIA is also initialized by software sequence as follows.

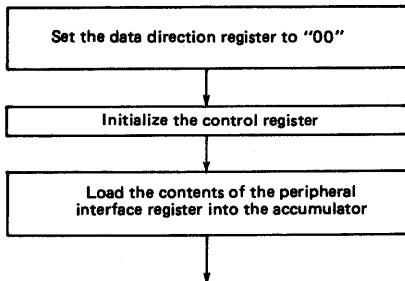


- Program the data direction register access bit of the control register to “0” to allow to access the data direction register.

- The data of the control line function is set into the accumulator, of which Data Direction Register Access Bit shall be programmed to “1”.
- Transfer the control data from the accumulator into the control register.

### ● Read/Write Operation Not Using Control Lines

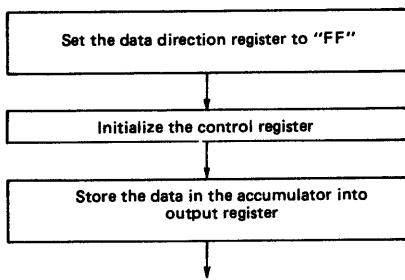
#### <Read Operation>



CLR	CRA
CLR	DDRA
LDAA	#\$04
STAA	CRA
-----	
LDAA	PIRA

- Clear the DDRA access bit of the control register to “0”.
- Clear all bits of the data direction register.
- Set DDRA access bit of the control register to “1” to allow to access the peripheral interface register.

#### <Write Operation>



CLR	CRA
LDAA	#\$FF
STAA	DDR <sub>B</sub>
-----	
LDAA	#\$04
STAA	CRB
-----	
LDAA	DATA
STAA	PIRB

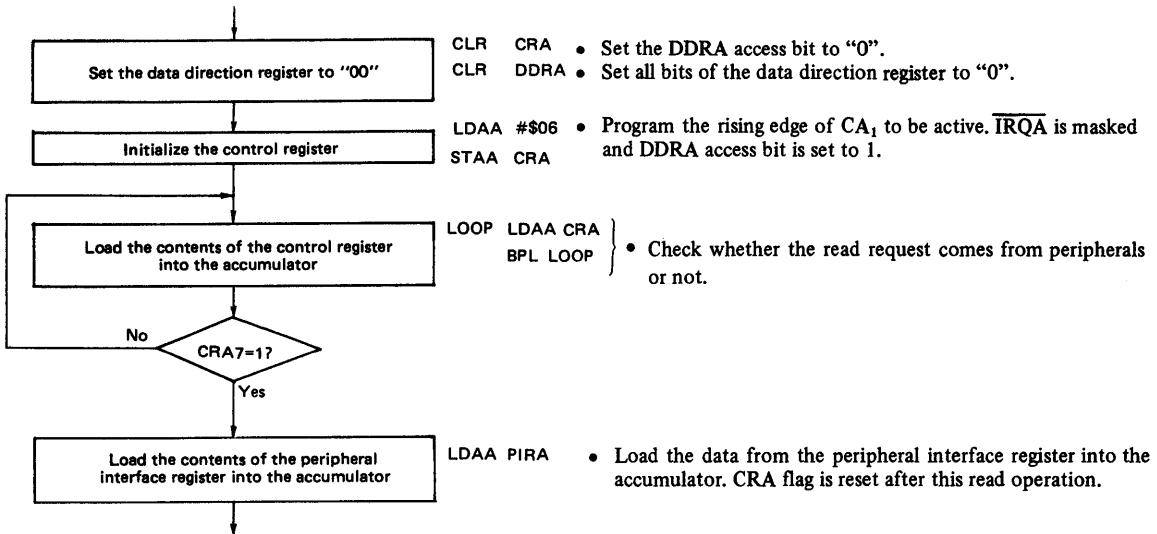
- Set DDRB access bit of the control register to “0”.
- Set all bits of the data direction register to “FF”.
- Set DDRB access bit of the control register to “1” to allow to access the peripheral interface register.
- Write the data into the peripheral interface register.

- **Read/Write Operating Using Control Lines**

Read/write request from peripherals shall be put into the control lines as an interrupt signal, and then MPU reads or writes after detecting interrupt request.

<Read>

The following case is that Port A is used and that the rising edge of CA<sub>1</sub> indicates the request for read from peripherals.

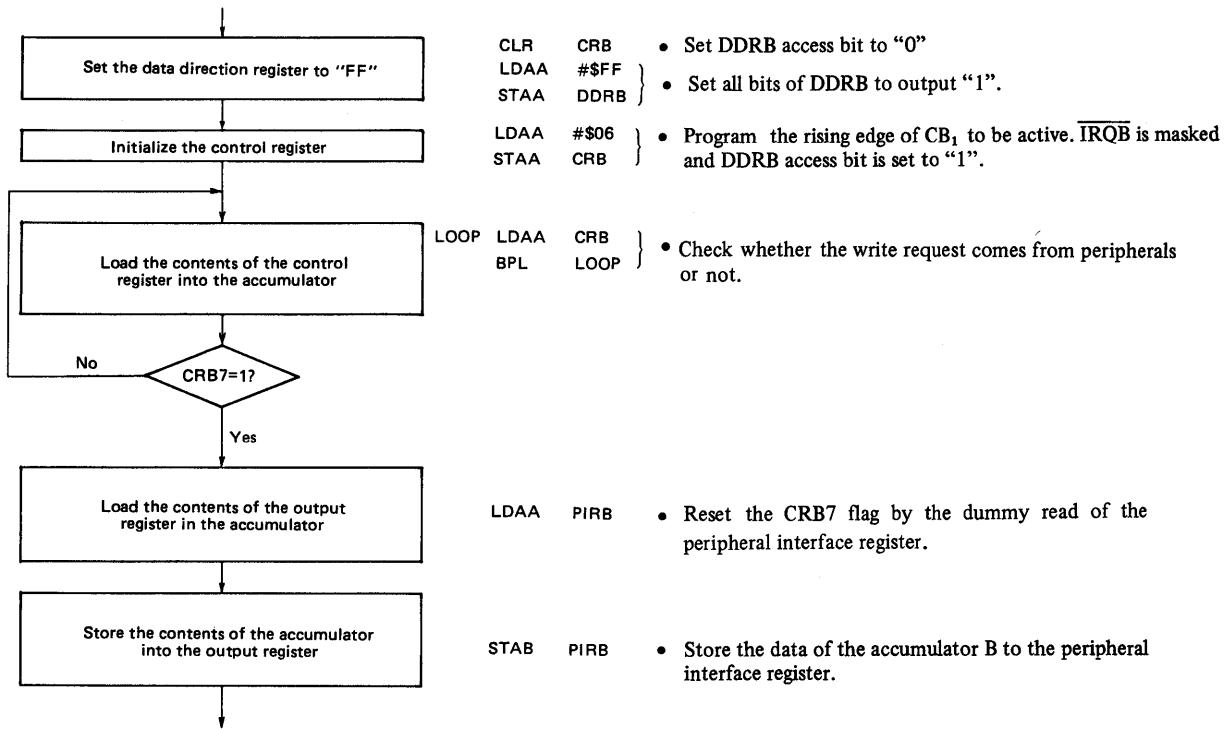


To read the peripheral data, the data is directly transferred to the data buses D<sub>0</sub>~D<sub>7</sub>, through PA<sub>0</sub>~PA<sub>7</sub>, or PB<sub>0</sub>~PB<sub>7</sub>, and they are not latched in the PIA. If necessary, the data should be held in the external latch until MPU completes reading it.

When initializing the control register, interrupt flag bit (CRA7, CRA6, CRB7, CRB6) cannot be written from MPU. If necessary the interrupt flag must be reset by dummy read of Peripheral Register A and B.

<Write>

Write operation using the interrupt signal is as follows. In this case, B port is used and interrupt request is input to CB<sub>1</sub>. And the IRQ flag is set at the rising edge of CB<sub>1</sub>.



Interrupt request flag bits (CRA7, CRA6, CRB7 and CRB6) cannot be written and they cannot be also reset by write operation to the peripheral interface register. So dummy read of peripheral interface register is needed to reset the flags.

To accept the next interrupt, it is essential to reset indirectly the interrupt flag by dummy read of peripheral interface register.

Software polling method mentioned above requires MPU to continuously monitor the control register to detect the read/write request from peripherals. So other programs cannot run at the same time. To avoid this problem, hardware interrupt may be used. The MPU is interrupted by  $\overline{\text{IRQA}}$  or  $\overline{\text{IRQB}}$  when the read/write request is occurred from peripherals and then MPU analyzes cause of the interrupt request during interrupt processing.

#### • Handshake Mode

The functions of CRA and CRB are similar but not identical in the hand-shake modes. Port A is used for read hand-shake operation and Port B is used for write hand-shake mode.

CA<sub>1</sub> and CB<sub>1</sub> are used for interrupt input requests and CA<sub>2</sub> and CB<sub>2</sub> are control outputs (answer) in hand-shake mode.

Fig. 16, Fig. 17 and Fig. 18 show the timing of hand-shake mode.

#### < Read Hand-shake Mode >

CRA5 = “1”, CRA4 = “0” and CRA3 = “0”

- (1) A peripheral device puts the 8-bit data on the peripheral data lines after the control output CA<sub>2</sub> goes “Low”.
- (2) The peripheral requests MPU to read the data by using CA<sub>1</sub> input.

(3) CRA7 flag is set and CA<sub>2</sub> becomes “High” (CA<sub>2</sub> automatically becomes “High” by the interrupt CA<sub>1</sub>). This indicates the peripheral to maintain the current data and not to transfer the next data.

- (4) MPU accepts the read request by  $\overline{\text{IRQA}}$  hardware interrupt or CRA read. Then MPU reads the peripheral register A.
- (5) CA<sub>2</sub> goes “Low” on the following edge of read Enable pulse. This informs that the peripheral can set the next data to port A.

#### < Write Hand-shake >

CRB5 = “1”, CRB4 = “0” and CRB3 = “0”

- (1) A peripheral device requests MPU to write the data by using CB<sub>1</sub> input. CB<sub>2</sub> output remains “High” until MPU write data to the peripheral interface register.
- (2) CRB7 flag is set and MPU accepts the write request.
- (3) MPU reads the peripheral interface register to reset CRB7 (dummy read).
- (4) Then MPU write data to the peripheral interface register. The data is output to port B through the output register.
- (5) CB<sub>2</sub> automatically becomes “Low” to tell the peripheral that new data is on port B.
- (6) The peripheral read the data on Port B peripheral data lines and set CB<sub>1</sub> to “Low” to tell MPU that the data on the peripheral data lines has been taken and that next data can be written to the peripheral interface register.

#### < Pulse mode >

CRA5 = “1”, CRA4 = “0” and CRA3 = “1”

CRB5 = “1”, CRB4 = “0” and CRB3 = “1”

This mode is shown in Figure 16, Figure 19 and Figure 20.

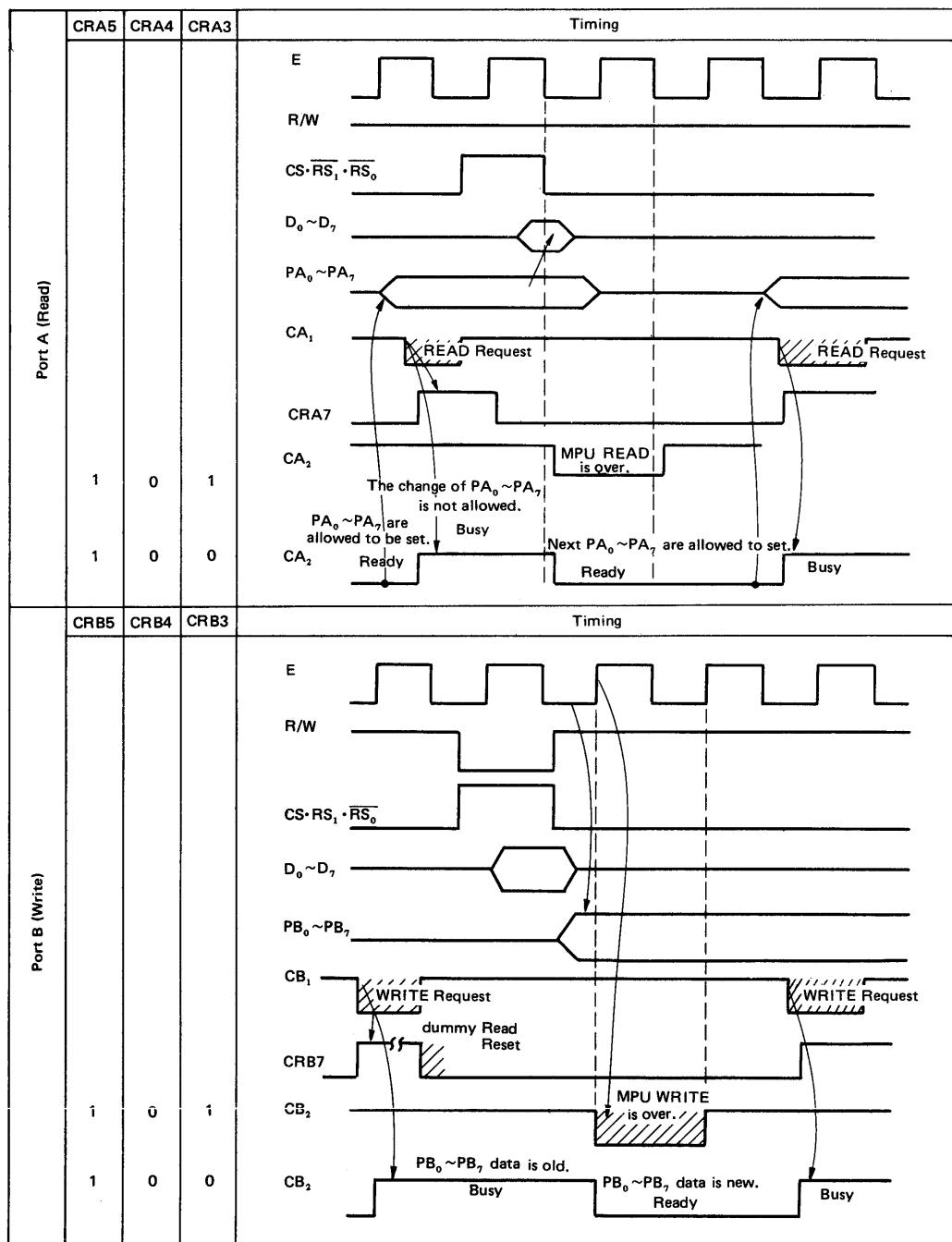
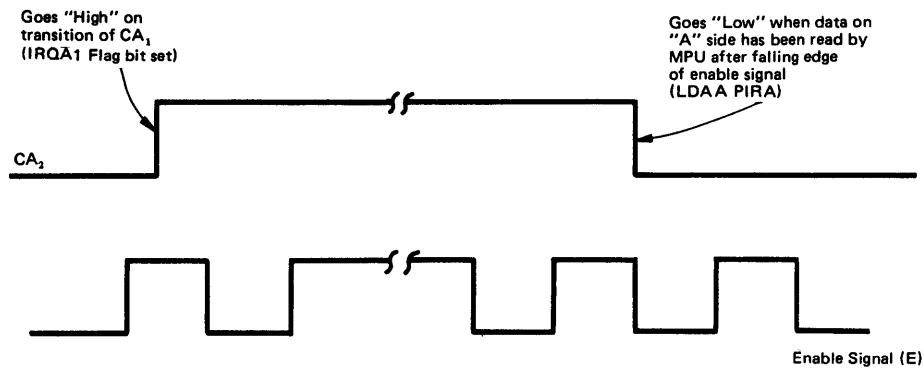


Figure 16 Timing of Hand-shake Mode and Pulse Mode



Handshaking with peripheral on 'A' side

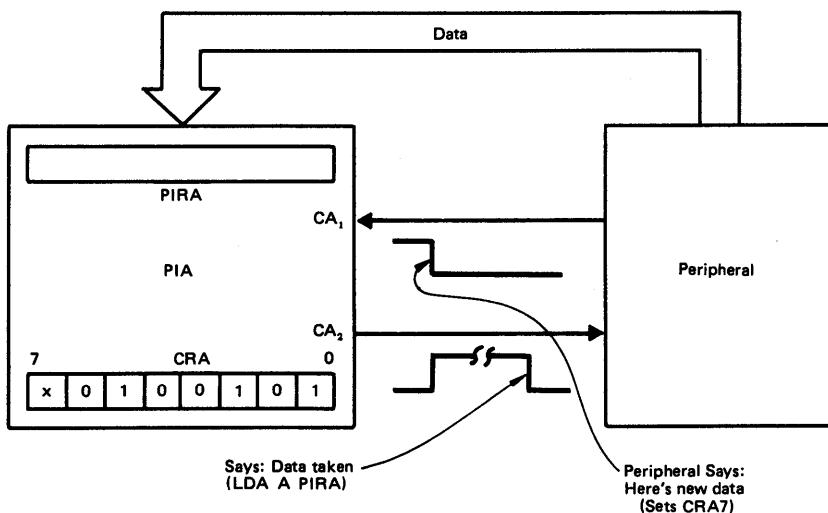


Figure 17 Bits 5, 4, 3 of CRA = 100 (Hand-shake Mode)

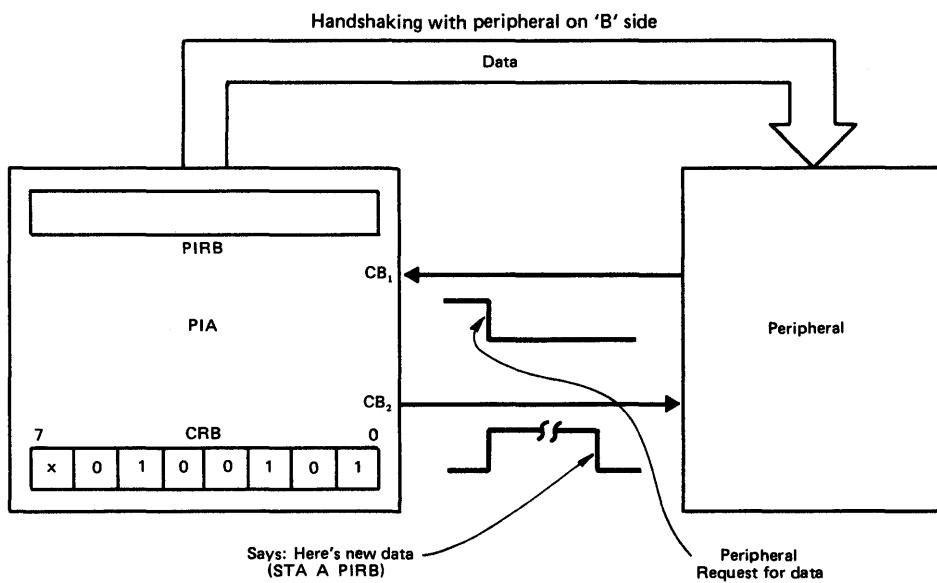
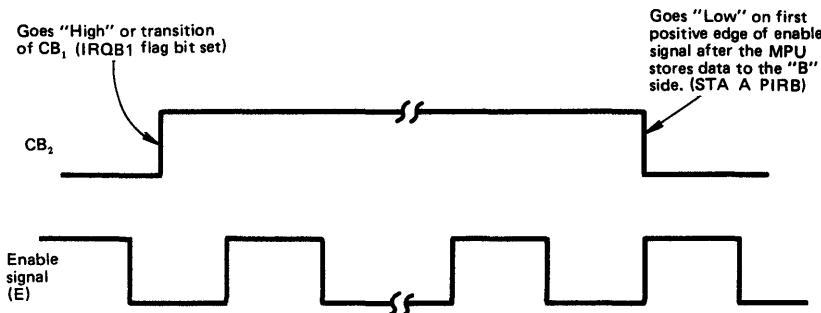


Figure 18 Bits 5, 4, 3 of CRB = 100 (Hand-shake Mode)

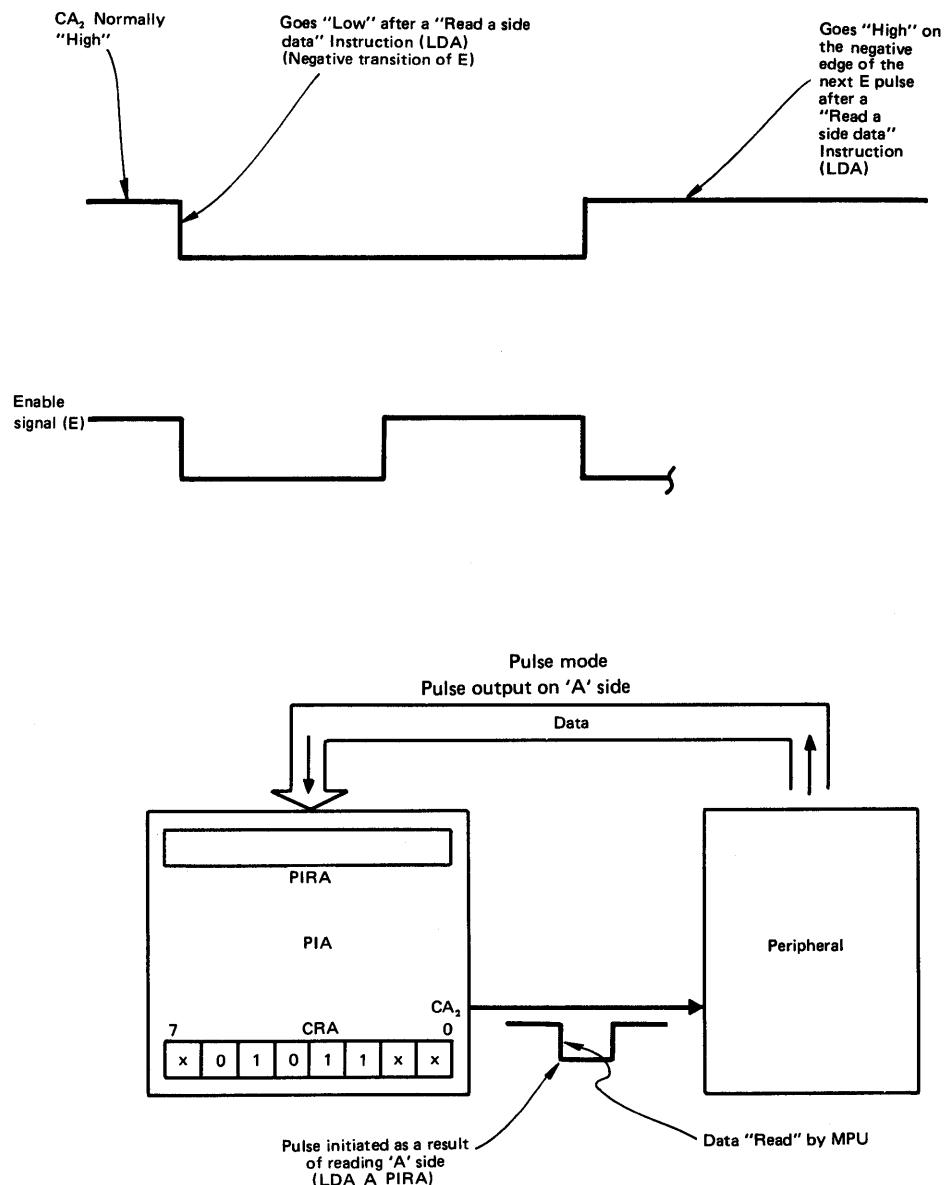


Figure 19 Bits 5, 4, 3 of CRA = 101 (Pulse Mode)

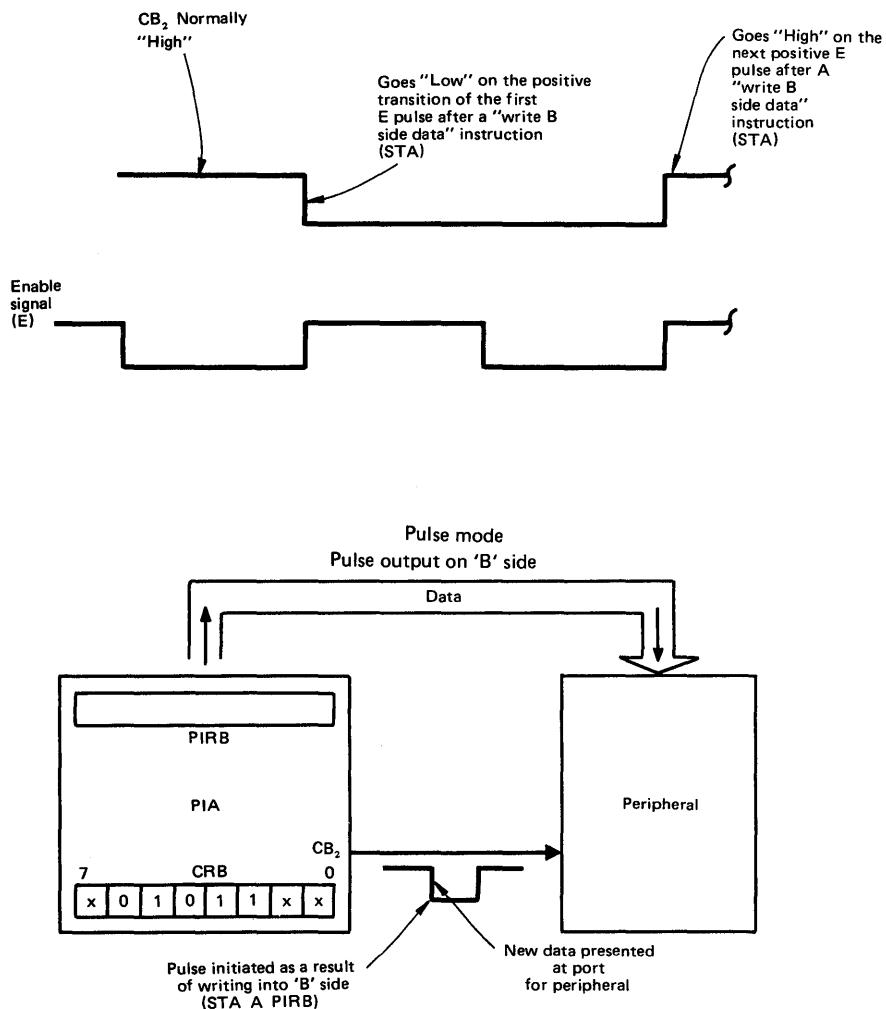


Figure 20 Bits 5, 4, 3 of CRB=101 (Pulse Mode)

## ■ SUMMARY OF CONTROL REGISTERS CRA AND CRB

Control registers CRA and CRB have total control of CA<sub>1</sub>, CA<sub>2</sub>, CB<sub>1</sub>, and CB<sub>2</sub> lines. The status of eight bits of the control registers may be read into the MPU. However, the MPU can only write into Bit 0 through Bit 5 (6 bits), since Bit 6 and Bit 7 are set only by CA<sub>1</sub>, CA<sub>2</sub>, CB<sub>1</sub>, or CB<sub>2</sub>.

### ● Addressing PIAs

Before addressing PIAs, the data direction (DDR) must first be loaded with the bit pattern that defines how each line is to function, i.e., as an input or an output. A logic "1" in the data direction register defines the corresponding line as an output while a logic "0" defines the corresponding line as an input. Since the DDR and the peripheral interface register have the same address, the control register bit 2 determines which register is being addressed. If Bit 2 in the control register is a logic "0", then the DDR is addressed. If Bit 2 in the control register is a logic "1", the peripheral interface register is addressed. Therefore, it is essential that the DDR be loaded first before setting Bit 2 of the control register.

### <Example>

Given a PIA with an address of 4004, 4005, 4006, and 4007. 4004 is the address of the A side peripheral interface register. 4005 is the address of the A side control register. 4006 is the address of the B side peripheral interface register. 4007 is the address of the B side control register. On the A side, Bits 0, 1, 2, and 3 will be defined as inputs, while Bits 4, 5, 6, and 7 will be used as outputs. On the B side, all lines will be used as outputs.

PIA1AD = 4004	(DDRA, PIRA)
PIA1AC = 4005	(CRA)
PIA1BD = 4006	(DDRB, PIRB)
PIA1BC = 4007	(CRB)

1. LDA A #11110000 (4 outputs, 4 inputs)
2. STA A PIA1AD (Loads A DDR)
3. LDA A #11111111 (All outputs)
4. STA A PIA1BD (Loads B DDR)
5. LDA A #00000100 (Sets Bit 2)
6. STA A PIA1AC (Bit 2 set in A control register)
7. STA A PIA1BC (Bit 2 set in B control register)

Statement 2 addresses the DDR, since the control register (Bit 2) has not been loaded. Statements 6 and 7 load the control registers with Bit 2 set, so addressing PIA1AD or PIA1BD accesses the peripheral interface register.

### ● PIA Programming Via The Index Register

The program shown in the previous section can be accomplished using the Index Register.

1. LDX #\$F004
2. STX PIA1AD \$F0→PIA1AD,\$04→PIA1AC
3. LDX #\$FF04
4. STX PIA1BD \$FF→PIA1BD,\$04→PIA1BC

Using the index register in this example has saved six bytes of program memory as compared to the program shown in the previous section.

### ● Active Low Outputs

When all the outputs of given PIA port are to be active "Low" (True  $\leq$  0.4 volts), the following procedure should be used.

- a) Set Bit 2 in the control register.
- b) Store all 1s (\$FF) in the peripheral interface register.
- c) Clear Bit 2 in the control register.
- d) Store all 1s (\$FF) in the data direction register.
- e) Store control word (Bit 2 = 1) in control register.

### <Example>

The B side of PIA1 is set up to have all active low outputs. CB<sub>1</sub> and CB<sub>2</sub> are set up to allow interrupts in the HAND-SHAKE MODE and CB<sub>1</sub> will respond to positive edges ("Low"-to-"High" transitions). Assume reset conditions. Addresses are set up and equated to the same labels as previous example.

1. LDA A #4 Set Bit 2 in PIA1BC (control register)
2. STA A PIA1BC
3. LDA B #\$FF All 1s in peripheral interface register
4. STA B PIA1BD
5. CLR PIA1BC Clear Bit 2
6. STA B PIA1BD All 1s in data direction register
7. LDA A #\$27
8. STA A PIA1BC 00100111→ control register

The above procedure is required in order to avoid outputs going "Low", to the active "Low" TRUE STATE, when all is stored to the data direction register as would be the case if the normal configuration procedure were followed.

### ● Interchanging RS<sub>0</sub> And RS<sub>1</sub>

Some system applications may require movement of 16 bits of data to or from the "outside world" via two PIA ports (A side + B side). When this is the case it is an advantage to interconnect RS<sub>1</sub> and RS<sub>0</sub> as follows.

RS<sub>0</sub> to A1 (Address Line A1)  
RS<sub>1</sub> to A0 (Address Line A0)

This will place the peripheral interface registers and control registers side by side in the memory map as follows.

Table Example Address

PIA1AD	\$4004	(DDRA, PIRA)
PIA1BD	\$4005	(DDRB, PIRB)
PIA1AC	\$4006	(CRA)
PIA1BC	\$4007	(CRB)

The index register or stackpointer may be used to move the 16-bit data in two 8-bit bytes with one instruction. As an example:

LDX PIA1AD PIA1AD→IX<sub>H</sub>; PIA1BD→IX<sub>L</sub>

### ● PIA — After Reset

When the RES (Reset Line) has been held "Low" for a minimum of one microsecond, all registers in the PIA will be cleared.

Because of the reset conditions, the PIA has been defined as

follows.

1. All I/O lines to the "outside world" have been defined as inputs.
2. CA<sub>1</sub>, CA<sub>2</sub>, CB<sub>1</sub>, and CB<sub>2</sub> have been defined as interrupt input lines that are negative edge sensitive.
3. All the interrupts on the control lines are masked. Setting of interrupt flag bits will not cause IRQA or IRQB to go "Low".

#### ■ SUMMARY OF CA<sub>1</sub>-CB<sub>1</sub> PROGRAMMING

Bits 1 and 0 of the respective control registers are used to program the interrupt input control lines CA<sub>1</sub> and CB<sub>1</sub>.

b1	b0	
0	0	b1 = Edge (0 = -, 1 = +)
0	1	b0 = Mask (0 = Mask, 1 = Allow)
1	0	
1	1	

Note that this is the same logic as Bits 4 and 3 for CA<sub>2</sub>-CB<sub>2</sub> when CA<sub>2</sub>-CB<sub>2</sub> are programmed as inputs.

#### ■ SUMMARY OF CA<sub>2</sub>-CB<sub>2</sub> PROGRAMMING

Bits 5, 4, and 3 of the control registers are used to program the operation of CA<sub>2</sub>-CB<sub>2</sub>.

	b5	b4	b3	
CA <sub>2</sub> - CB <sub>2</sub>	0	0(-)	0 (Mask)	CA <sub>2</sub> -CB <sub>2</sub> Input Mode
Input Mode	0	0(-)	1 (Allow)	b4 = Edge (0 = -, 1 = +)
	0	1(+) 0	(Mask)	b3 = Mask (0 = Mask, 1 = Allow)
	0	1(+) 1	1 (Allow)	

CA <sub>2</sub> - CB <sub>2</sub>	1	0	0	- Handshake Mode
Output Mode	1	0	1	- Pulse Mode
	1	1	0	} b3 Following Mode
	1	1	1	

#### I/O As Follow:

##### Control Lines:

- CA<sub>1</sub> — Positive Edge, Allow Interrupt
- CA<sub>2</sub> — Pulse Mode
- CB<sub>1</sub> — Negative Edge, Mask Interrupt
- CB<sub>2</sub> — Hand Shake Mode

#### Assume Reset Condition

PIA1AD  
PIA1AC  
PIA1BD  
PIA1BC

#### PIA Configuration Solution

```
LDA A #$BC      10111100
STA A PIA1AD    I/O to DDRA
LDA A #$FF      1111 1111
STA A PIA1BD    I/O to DDRB
LDA A #$2F      0010 1111
STA A PIA1AC    To "A" Control
LDA A #$24      0010 0100
STA A PIA1BC    To "B" Control
```

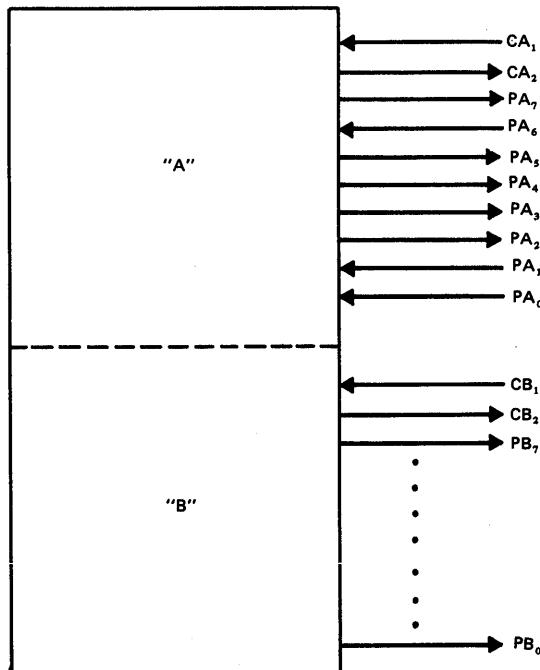


Figure 21 PIA Configuration Problem

# HD6840, HD68A40, HD68B40

## PTM (Programmable Timer Module)

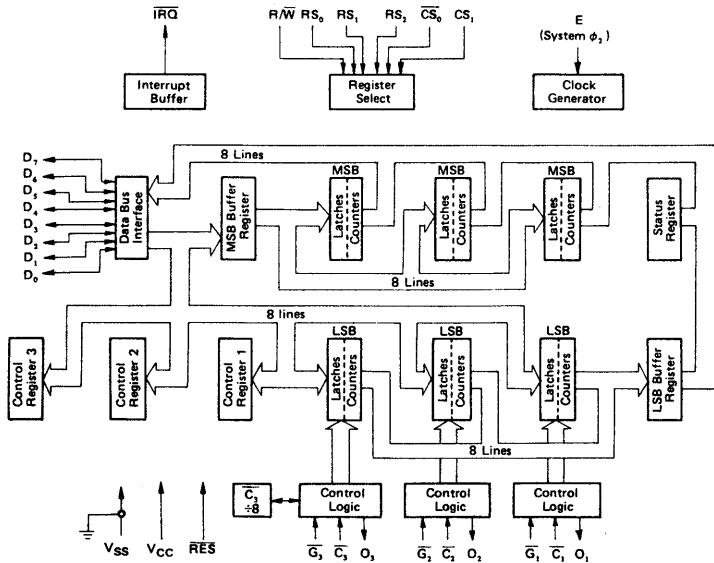
The HD6840 is a programmable subsystem component of the HMCS6800 family designed to provide variable system time intervals.

The HD6840 has three 16-bit binary counters, three corresponding control registers and a status register. These counters are under software control and may be used to cause system interrupts and/or generate output signals. The HD6840 may be utilized for such tasks as frequency measurements, event counting, interval measuring and similar tasks. The device may be used for square wave generation, gated delay signals, single pulses of controlled duration, and pulse width modulation as well as system interrupts.

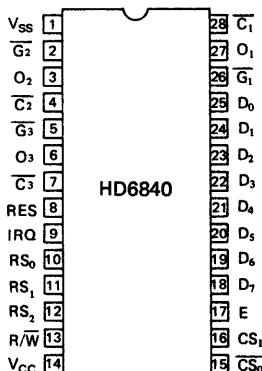
### ■ FEATURES

- Operates from a Single 5 Volts Power Supply
- Fully TTL Compatible
- Single System Clock Required (E)
- Selectable Prescaler on Timer 3 Capable of 4 MHz for the HD6840, 6 MHz for the HD68A40 and 8 MHz for the HD68B40
- Programmable Interrupts (IRQ) Output to MPU
- Readable Down Counter Indicates Counts to Go to Time-Out
- Selectable Gating for Frequency or Pulse-Width Comparison
- RES Input
- Three Asynchronous External Clock and Gate/Trigger Inputs Internally Synchronized
- Three Maskable Outputs

### ■ BLOCK DIAGRAM



### ■ PIN ARRANGEMENT



(Top View)

### ■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	$V_{CC}^*$	-0.3~+7.0	V
Input Voltage	$V_{IN}^*$	-0.3~+7.0	V
Operating Temperature	$T_{opr}$	-20~+75	°C
Storage Temperature	$T_{stg}$	-55~+150	°C

\* With respect to  $V_{SS}$  (SYSTEM GND)

[NOTE] 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.

### ■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol	min	typ	max	Unit
Supply Voltage	$V_{CC}^*$	4.75	5	5.25	V
Input Voltage	$V_{IL}^*$	-0.3	—	0.8	V
	$V_{IH}^*$	2.0	—	$V_{CC}$	V
Operating Temperature	$T_{opr}$	-20	25	75	°C

\* With respect to  $V_{SS}$  (SYSTEM GND)

### ■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS ( $V_{CC} = 5V \pm 5\%$ ,  $V_{SS} = 0V$ ,  $T_a = -20 \sim +75^\circ C$ , unless otherwise noted.)

Item	Symbol	Test Condition		min	typ*	max	Unit
Input "High" Voltage	$V_{IH}$			2.0	—	$V_{CC}$	V
Input "Low" Voltage	$V_{IL}$			-0.3	—	0.8	V
Input Leakage Current	$I_{in}$	$V_{in} = 0 \sim 5.25V$ (Except $D_0 \sim D_7$ )		-2.5	—	2.5	μA
Three-State Input Current (off-state)	$I_{TSI}$	$V_{in} = 0.4 \sim 2.4V$		-10	—	10	μA
		$V_{CC} = 5.25V$ ( $D_0 \sim D_7$ )					
Output "High" Voltage	$V_{OH}$	$I_{LOAD} = -205 \mu A$ ( $D_0 \sim D_7$ )		2.4	—	—	V
		$I_{LOAD} = -200 \mu A$ (Other Outputs)					
Output "Low" Voltage	$V_{OL}$	$I_{LOAD} = 1.6 mA$ ( $D_0 \sim D_7$ )		—	—	0.4	V
		$I_{LOAD} = 3.2 mA$ ( $O_1 \sim O_3$ , $\overline{IRQ}$ )					
Output Leakage Current (off-state)	$I_{LOH}$	$V_{OH} = 2.4V$ ( $\overline{IRQ}$ )		—	—	10	μA
Power Dissipation	$P_D$			—	330	550	mW
Input Capacitance	$C_{in}$	$V_{in} = 0V$	$D_0 \sim D_7$	—	—	12.5	pF
		$T_a = 25^\circ C$	Other Input	—	—	7.5	
Output Capacitance	$C_{out}$	$V_{in} = 0V$	$\overline{IRQ}$	—	—	5.0	pF
		$T_a = 25^\circ C$	$O_1, O_2, O_3$	—	—	10	

\*  $T_a = 25^\circ C$ ,  $V_{CC} = 5.0V$

- AC CHARACTERISTICS ( $V_{CC} = 5V \pm 5\%$ ,  $V_{SS} = 0V$ ,  $T_a = -20 \sim +75^\circ C$ , unless otherwise noted.)

### 1. MPU READ TIMING

Item	Symbol	Test Condition	HD6840			HD68A40			HD68B40			Unit
			min	typ	max	min	typ	max	min	typ	max	
Enable Cycle Time	$t_{cycE}$	Fig. 1	1.0	—	10	0.666	—	10	0.5	—	10	$\mu s$
Enable "High" Pulse Width	$PW_{EH}$		0.45	—	4.5	0.280	—	4.5	0.22	—	4.5	$\mu s$
Enable "Low" Pulse Width	$PW_{EL}$		0.43	—	—	0.280	—	—	0.21	—	—	$\mu s$
Enable Rise and Fall Time	$t_{ER}, t_{EF}$		—	—	25	—	—	25	—	—	25	ns
Address Set Up Time	$t_{AS}$		160	—	—	140	—	—	70	—	—	ns
Data Delay Time	$t_{DDR}$		—	—	320	—	—	220	—	—	180	ns
Data Hold Time	$t_H$		10	—	—	10	—	—	10	—	—	ns
Address Hold Time	$t_{AH}$		10	—	—	10	—	—	10	—	—	ns
Data Access Time	$t_{ACC}$		—	—	480	—	—	360	—	—	250	ns

### 2. MPU WRITE TIMING

Item	Symbol	Test Condition	HD6840			HD68A40			HD68B40			Unit
			min	typ	max	min	typ	max	min	typ	max	
Enable Cycle Time	$t_{cycE}$	Fig. 2	1.0	—	10	0.666	—	10	0.5	—	10	$\mu s$
Enable "High" Pulse Width	$PW_{EH}$		0.45	—	4.5	0.280	—	4.5	0.22	—	4.5	$\mu s$
Enable "Low" Pulse Width	$PW_{EL}$		0.43	—	—	0.280	—	—	0.21	—	—	$\mu s$
Enable Rise and Fall Time	$t_{ER}, t_{EF}$		—	—	25	—	—	25	—	—	25	ns
Address Set Up Time	$t_{AS}$		160	—	—	140	—	—	70	—	—	ns
Data Set Up Time	$t_{DSW}$		195	—	—	80	—	—	60	—	—	ns
Data Hold Time	$t_H$		10	—	—	10	—	—	10	—	—	ns
Address Hold Time	$t_{AH}$		10	—	—	10	—	—	10	—	—	ns

### 3. TIMING OF PTM SIGNAL

Item	Symbol	Test Condition	HD6840		HD68A40		HD68B40		Unit		
			min	max	min	max	min	max			
Input Rise and Fall Times	$\overline{C}_3, \overline{G}, RES$	$t_r, t_f$	Fig. 3, Fig. 4	—	1.0*	—	0.666*	—	0.5*	$\mu s$	
Input "Low" Pulse Width	$\overline{C}_3, \overline{G}, \overline{RES}$	$PW_L$	Fig. 3 (Asynchronous Mode)	$t_{cycE} + t_{SU} + t_{HD}$	—	$t_{cycE} + t_{SU} + t_{HD}$	—	$t_{cycE} + t_{SU} + t_{HD}$	—	ns	
Input "High" Pulse Width	$\overline{C}_3, \overline{G}$	$PW_H$	Fig. 4 (Asynchronous Mode)	$t_{cycE} + t_{SU} + t_{HD}$	—	$t_{cycE} + t_{SU} + t_{HD}$	—	$t_{cycE} + t_{SU} + t_{HD}$	—	ns	
Input Setup Time	$\overline{C}_3, \overline{G}, RES$	$t_{SU}$	Fig. 5 (Synchronous Mode)	200	—	120	—	75	—	ns	
				200	—	120	—	75	—		
Input Hold Time	$\overline{C}_3, \overline{G}, RES$	$t_{HD}$	Fig. 5 (Synchronous Mode)	50	—	50	—	50	—	ns	
				50	—	50	—	50	—		
Input Pulse Width	$\overline{C}_3$ ( $\div 8$ Pre-scaler Mode)	$PW_L, PW_H$	(Asynchronous Mode)	125	—	84	—	62.5	—	ns	
Output Delay Time	$O_1 \sim O_3$	$t_{co}$	$V_{OH}=2.4V$ , Load B	—	700	—	460	—	340	ns	
			$V_{OH}=2.4V$ , Load D	—	450	—	450	—	340	ns	
			$V_{OH}=0.7 \times V_{CC}$ , Load D	—	2.0	—	1.35	—	1.0	$\mu s$	
Interrupt Release Time			$t_{IR}$	Fig. 7	—	1.2	—	0.9	—	0.7	$\mu s$

\*  $t_r, t_f \leq t_{cycE}$

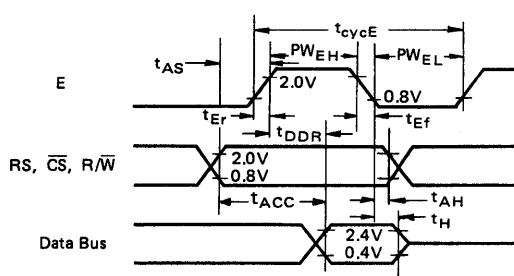


Figure 1 Bus Read Timing  
(Read Information from PTM)

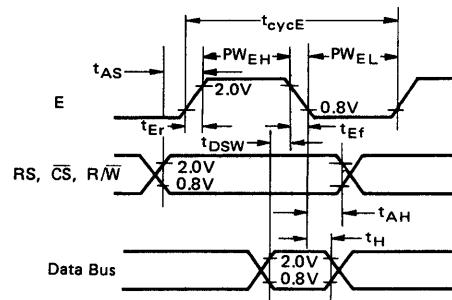


Figure 2 Bus Write Timing  
(Write Information into PTM)

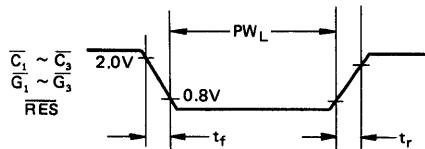


Figure 3 Input Pulse Width “Low”

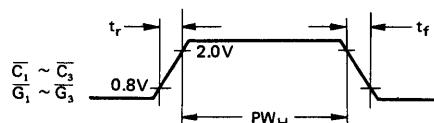


Figure 4 Input Pulse Width “High”

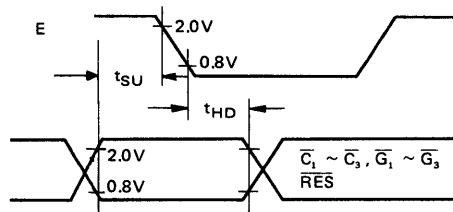


Figure 5 Input Setup and Hold Times

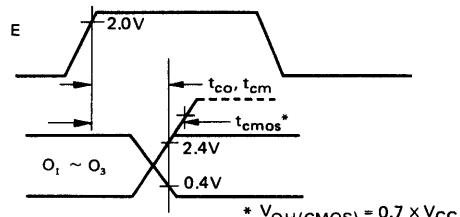


Figure 6 Output Delay  
\*  $V_{OH}(\text{CMOS}) = 0.7 \times V_{CC}$

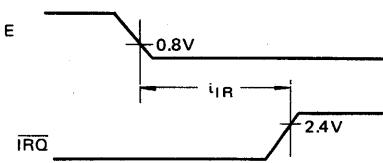


Figure 7 IRQ Release Time

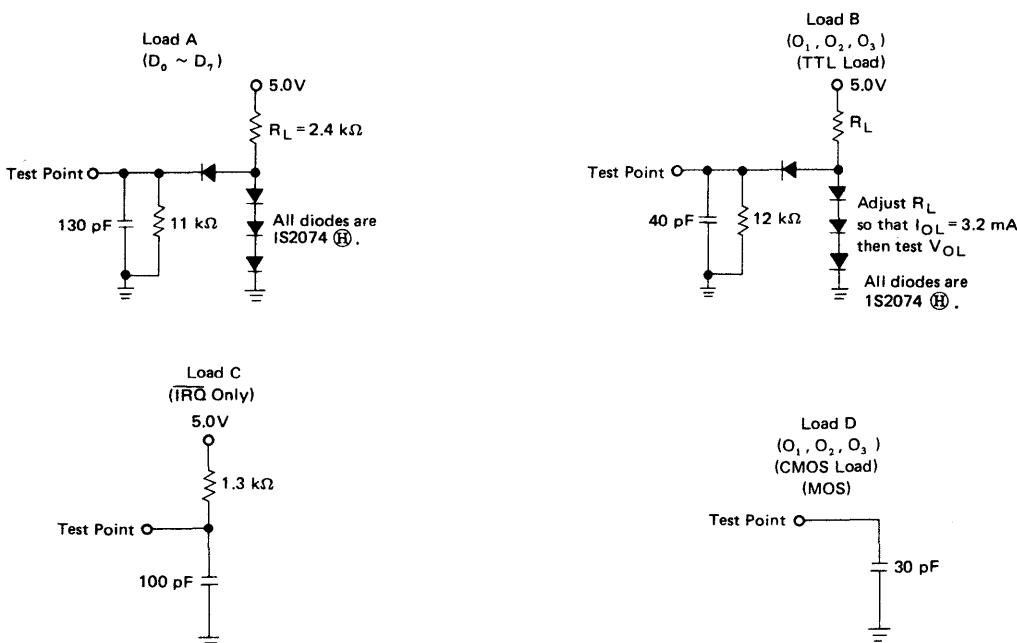


Figure 8 Bus Timing Test Loads

The three timers in the HD6840 may be independently programmed to operate in modes which fit a wide variety of applications. The device is fully bus compatible with HMCS6800 system and is accessed by load and store operations from the MPU in much the same manner as a memory device. In a typical application, a Timer will be loaded by first storing two bytes of data into an associated Counter Latch. This data is then transferred into the counter via a Counter Initialization cycle. The counter decrements on each subsequent clock period which may be an external clock or Enable (System  $\phi_2$ ) until one of several predetermined conditions causes it to halt or recycle. The timers are thus programmable, cyclic in nature, controllable by external inputs or the MPU program, and accessible by the MPU at any time.

#### ■ PTM INTERFACE SIGNALS FOR MPU

The Programmable Timer Module (PTM) interfaces to the HMCS6800 Bus with an eight-bit bidirectional data bus, two Chip Select lines, a Read/Write line, an Enable (System  $\phi_2$ ) line, an Interrupt Request line, an external Reset line, and three Register Select lines. These signals, in conjunction with the HD6800 VMA output, permit the

MPU to control the PTM. VMA should be utilized in conjunction with an MPU address line into a Chip Select of the PTM.

#### ● Bidirectional Data ( $D_0 \sim D_7$ )

The bidirectional data lines ( $D_0 \sim D_7$ ) allow the transfer of data between the MPU and PTM. The data bus output drivers are three-state devices which remain in the high-impedance (off) state except when the MPU performs a PTM read operation (Read/Write and Enable lines "High" and PTM Chip Selects activated).

#### ● Chip Select ( $\overline{CS}_0, CS_1$ )

These two signals are used to activate the Data Bus interface and allow transfer of data from the PTM. With  $\overline{CS}_0 = \text{"Low"}$  and  $CS_1 = \text{"High"}$ , the device is selected and data transfer will occur.

#### ● Read/Write ( $R/W$ )

This signal is generated by the MPU to control the direction of data transfer on the Data Bus. With the PTM selected, a "Low" state on the PTM R/W line enables the input buffers and data is transferred from the MPU to the

PTM on the trailing edge of the Enable (System  $\phi_2$ ) signal. Alternately, (under the same conditions) R/W = "High" and Enable "High" allows data in the PTM to be read by the MPU.

#### ● Enable (E)

This signal synchronizes data transfer between the MPU and the PTM. It also performs an equivalent synchronization function on the external clock, reset, and gate inputs of the PTM.

#### ● Interrupt Request (IRQ)

The active "Low" Interrupt Request signal is normally tied directly (or through priority interrupt circuitry) to the  $\overline{IRQ}$  input of the MPU. This is an "open drain" output (no load device on the chip) which permits other similar interrupt request lines to be tied together in a wire-OR configuration.

The  $\overline{IRQ}$  line is activated if, and only if, the Composite Interrupt Flag (Bit 7 of the Internal Status Register) is asserted. The conditions under which the  $\overline{IRQ}$  line is activated are discussed in conjunction with the Status Register.

#### ● External Reset (RES)

A "Low" level at this input is clocked into the PTM by the Enable (System  $\phi_2$ ) input. Two Enable pulses are required to synchronize and process the signal. The PTM then recognizes the active "Low" or inactive "High" on the third Enable pulse. If the RES signal is asynchronous,

an additional Enable period is required if setup times are not met. The RES input must be stable "High"/"Low" for the minimum time stated in the AC Characteristics.

Recognition of a "Low" level at this input by the PTM causes the following action to occur:

- All counter latches are preset to their maximal count values.
- All Control Register bits are cleared with the exception of CR10 (internal reset bit) which is set.
- All counters are preset to the contents of the latches.
- All counter outputs are reset and all counter clocks are disabled.
- All Status Register bits (interrupt flags) are cleared.

#### ● Register Select Lines ( $RS_0$ , $RS_1$ , $RS_2$ )

These inputs are used in conjunction with the R/W line to select the internal registers, counters and latches as shown in Table 1.

It has been previously stated that the PTM is accessed via MPU Load and Store operations in much the same manner as a memory device. The instructions available with the HMCS6800 family of MPUs which perform operations directly on memory should not be used when the PTM is accessed. These instructions actually fetch a byte from memory, perform an operation, then restore it to the same address location. Since the PTM used the R/W line as an additional register select input, the modified data may not be restored to the same register if these instructions are used.

Table 1 Register Selection

Register * Select Inputs			Operations	
$RS_2$	$RS_1$	$RS_0$	R/W = "Low"	R/W = "High"
L	L	L	CR20 = "0" Write Control Register #3 CR20 = "1" Write Control Register #1	No Operation
L	L	H	Write Control Register #2	Read Status Register
L	H	L	Write MSB Buffer Register	Read Timer #1 Counter
L	H	H	Write Timer #1 Latches	Read LSB Buffer Register
H	L	L	Write MSB Buffer Register	Read Timer #2 Counter
H	L	H	Write Timer #2 Latches	Read LSB Buffer Register
H	H	L	Write MSB Buffer Register	Read Timer #3 Counter
H	H	H	Write Timer #3 Latches	Read LSB Buffer Register

\* L; "Low" level, H; "High" level

#### ■ PTM ASYNCHRONOUS INPUT/OUTPUT SIGNALS

Each of the three timers within the PTM has external clock and gate inputs as well as a counter output line. The inputs are high impedance, TTL compatible lines and outputs are capable of driving two standard TTL loads.

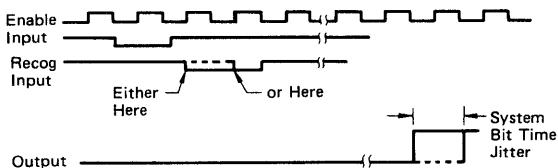
#### ● Clock Inputs ( $\overline{C}_1$ , $\overline{C}_2$ , $\overline{C}_3$ )

Input pins  $\overline{C}_1$ ,  $\overline{C}_2$ , and  $\overline{C}_3$  will accept asynchronous TTL voltage level signals to decrement Timers 1, 2, and 3, respectively. The "High" and "Low" levels of the external clocks must each be stable for at least one system clock period plus the sum of the setup and hold times for the inputs. The asynchronous clock rate can vary from dc to

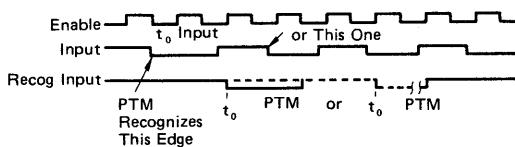
the limit imposed by Enable (System  $\phi_2$ ) Setup, and Hold time.

The external clock inputs are clocked in by Enable (System  $\phi_2$ ) pulses. Three Enable periods are used to synchronize and process the external clock. The fourth Enable pulse decrements the internal counter. This does not affect the input frequency, it merely creates a delay between a clock input transition and internal recognition of that transition by the PTM. All references to  $\overline{C}$  inputs in this document relate to internal recognition of the input transition. Note that a clock "High" or "Low" level which does not meet setup and hold time specifications may require an additional Enable pulse for recognition.

When observing recurring events, a lack of synchronization will result in "jitter" being observed on the output of the PTM when using asynchronous clocks and gate input signals. There are two types of jitter. "System jitter" is the result of the input signals being out of synchronization with the Enable (System  $\phi_2$ ), permitting signals with marginal setup and hold time to be recognized by either the bit time nearest the input transition or the subsequent bit time.



"Input jitter" can be as great as the time between input signal negative going transitions plus the system jitter, if the first transition is recognized during one system cycle, and not recognized the next cycle, or vice versa.



External clock input  $\overline{C}_3$  represents a special case when Timer #3 is programmed to utilize its optional  $\div 8$  prescaler mode. The maximum input frequency and allowable duty cycles for this case are specified under the AC Characteristics. The output of the  $\div 8$  prescaler is treated in the same manner as the previously discussed clock inputs. That is, it is clocked into the counter by Enable pulses, is recognized on the fourth Enable pulse (provided setup and hold time requirements are met), and must produce an output pulse at least as wide as the sum of an Enable period, setup, and hold times.

#### • Gate Inputs ( $\overline{G}_1$ , $\overline{G}_2$ , $\overline{G}_3$ )

Input pins  $\overline{G}_1$ ,  $\overline{G}_2$ , and  $\overline{G}_3$  accept asynchronous TTL-compatible signals which are used as triggers or clock gating functions to Timers 1, 2, and 3, respectively. The gating inputs are clocked into the PTM by the Enable (System  $\phi_2$ ) signal in the same manner as the previously discussed clock inputs. That is, a  $\overline{G}$  transition is recognized by the PTM on the fourth Enable pulse (provided setup and hold time requirements are met), and the "High" or "Low" levels of the  $\overline{G}$  input must be stable for at least one system clock period plus the sum of setup and hold times. All references to  $\overline{G}$  transition in this document relate to internal recognition of the input transition.

The Gate inputs of all timers directly affect the internal 16-bit counter. The operation of  $\overline{G}_3$  is therefore independent of the  $\div 8$  prescaler selection.

#### • Timer Outputs ( $O_1$ , $O_2$ , $O_3$ )

Timer outputs  $O_1$ ,  $O_2$ , and  $O_3$  are capable of driving up to two TTL loads and produce a defined output waveform for either Continuous or Single-Shot Timer modes. Output waveform definition is accomplished by selecting either Single 16-bit or Dual 8-bit operating modes. The single 16-bit mode will produce a square-wave output in the continuous timer mode and will produce a single pulse in the Single-Shot Timer mode. The Dual 8-bit mode will produce a variable duty cycle pulse in both the continuous and single shot Timer modes. "1" bit of each Control Register (CRX7) is used to enable the corresponding output. If this bit is cleared, the output will remain "Low" ( $V_{OL}$ ) regardless of the operating mode.

The Continuous and Single-Shot Timer Modes are the only ones for which output response is defined in this data sheet. Signals appear at the outputs (unless CRX7 = "0") during Frequency and Pulse Width comparison modes, but the actual waveform is not predictable in typical applications.

#### ■ CONTROL REGISTER

Three Write-Only registers in the HD6840 are used to modify timer operation to suit a variety of applications. Control Register #2 has a unique address space ( $RS_0$  = "High",  $RS_1$  = "Low",  $RS_2$  = "Low") and therefore may be written into at any time. The remaining Control Registers (#1 and #3) share the Address Space selected by a logic zero on all Register Select inputs. The least significant bit of Control Register #2 (CR20) is used as an additional addressing bit for Control Registers #1 and #3. Thus, with all Register Selects and R/W inputs at "Low" level, Control Register #1 will be written into if CR20 is a logic "1". Under the same conditions, Control Register #3 will be written into if CR20 is a logic "0". Control Register #3 can also be written into after a  $\overline{RES}$  "Low" condition has occurred, since all control register bits (except CR10) are cleared. Therefore, one may write in the sequence CR3, CR2, CR1.

The least significant bit of Control Register #1 is used as an Internal Reset bit. When this bit is a logic "0", all timers are allowed to operate in the modes prescribed by the remaining bits of the control registers. Writing a "1" into CR10 causes all counters to be preset with the contents of the corresponding counter latches, all counter clocks to be disabled, and the timer outputs and interrupt flags (Status Register) to be reset. Counter Latches and Control Registers are undisturbed by an Internal Reset and may be written into regardless of the state of CR10.

The least significant bit of Control Register #3 is used as a selector for a  $\div 8$  prescaler which is available with Timer #3 only. The prescaler, if selected, is effectively placed between the clock input circuitry and the input to Counter #3. It can therefore be used with either the internal clock (Enable) or an external clock source.

The functions depicted in the foregoing discussions are tabulated on the first row in Table 2 for ease of reference.

Table 2 Control Register Bits

CR10 Internal Reset Bit	CR20 Control Register Address Bit	CR30 Timer #3 Clock Control
"0" All timers allowed to operate "1" All timers held in preset state	"0" CR #3 may be written "1" CR #1 may be written	"0" T3 Clock is not prescaled "1" T3 Clock is prescaled by $\div 8$
CRX1*	Timer #X Clock Source "0" TX uses external clock source on CX input "1" TX uses Enable clock	
CRX2	Timer #X Counting Mode Control "0" TX configured for normal (16-bit) counting mode "1" TX configured for dual 8-bit counting mode	
CRX3 CRX4 CRX5	Timer #X Counter Mode and Interrupt Control (See Table 3)	
CRX6	Timer #X Interrupt Enable "0" Interrupt Flag masked on $\overline{IRQ}$ "1" Interrupt Flag enabled to $\overline{IRQ}$	
CRX7	Timer #X Counter Output Enable "0" TX Output masked on output OX "1" TX Output enabled on output OX	

\* Control Register for Timer 1, 2, or 3, Bit 1.

Control Register Bits CR10, CR20, and CR30 are unique in that each selects a different function. The remaining bits (1 through 7) of each Control Register select common functions, with a particular Control Register affecting only its corresponding timer. For example, Bit 1 of Control Register #1 (CR11) selects whether an internal or external clock source is to be used with Timer #1. Similarly, CR21 selects the clock source for Timer #2, and CR31 performs this function for Timer #3. The function of each bit of Control Register "X" can therefore be defined as shown in the remaining section of Table 2.

Control Register Bit 2 selects whether the binary information contained in the Counter Latches (and subsequently loaded into the Counter) is to be treated as a single 16-bit word or two 8-bit bytes. In the single 16-bit Counter Mode (CRX2 = "0") the counter will decrement to zero after N+1 enabled ( $\overline{G}$  = "Low") clock periods, where N is defined as the 16-bit number in the Counter Latches. With CRX2 = "1", a similar Time Out will occur after  $(L+1) \cdot (M+1)$  enabled clock periods, where L and M, respectively, refer to the LSB and MSB bytes in the Counter Latches.

Control Register Bits 3, 4, and 5 are explained in detail in the Timer Operating Mode section. Bit 6 is an interrupt mask bit which will be explained more fully in conjunction with the Status Register, and bit 7 is used to enable the corresponding Timer Output. A summary of the control register programming modes is shown in Table 3.

#### ■ STATUS REGISTER/INTERRUPT FLAGS

The HD6840 has an internal Read-Only Status Register which contains four Interrupt Flags. (The remaining four bits of the register are not used, and default to "0's when being read.) Bits 0, 1, and 2 are assigned to Timers 1, 2, and 3, respectively, as individual flag bits, while Bit 7 is a

Composite Interrupt Flag. This flag bit will be asserted if any of the individual flag bits is set while Bit 6 of the corresponding Control Register is at a logic "1". The conditions for asserting the Composite Interrupt Flag bit can therefore be expressed as:

$$INT = I_1 \cdot CR16 + I_2 \cdot CR26 + I_3 \cdot CR36$$

where INT = Composite Interrupt Flag (Bit 7)

$I_1$  = Timer #1 Interrupt Flag (Bit 0)

$I_2$  = Timer #2 Interrupt Flag (Bit 1)

$I_3$  = Timer #3 Interrupt Flag (Bit 2)

An interrupt flag is cleared by a Timer Reset condition, i.e., External  $\overline{RES}$  = "Low" or Internal Reset Bit (CR10) = "1". It will also be cleared by a Read Timer Counter Command provided that the Status Register has previously been read while the interrupt flag was set. This condition on the Read Status Register — Read Timer Counter (RS—RT) sequence is designed to prevent missing interrupts which might occur after the status register is read, but prior to reading the Timer Counter.

An Individual Interrupt Flag is also cleared by a Write Timer Latches (W) command or a Counter Initialization (CI) sequence, provided that W or CI affects the Timer corresponding to the individual Interrupt Flag.

#### ■ COUNTER LATCH INITIALIZATION

Each of the three independent timers consists of a 16-bit addressable counter and 16 bits of addressable latches. The counters are preset to the binary numbers stored in the latches. Counter initialization results in the transfer of the latch contents to the counter. See notes in Table 5 regarding the binary number N, L, or M placed into the Latches and their relationship to the output waveforms and counter Time-Outs.

Since the PTM data bus is 8-bits wide and the counters are 16-bits wide, a temporary register (MSB Buffer Register) is provided. This "write only" register is for the

Most Significant Byte of the desired latch data. Three addresses are provided for the MSB Buffer Register (as indicated in Table 1), but they all lead to the same Buffer. Data from the MSB Buffer will automatically be transferred into the Most Significant Byte of Timer #X when a Write Timer #X Latches Command is performed. So it can be seen that the HD6840 has been designed to allow transfer of two bytes of data into the counter latches provided that the MSB is transferred first.

In many applications, the source of the data will be an HMCS6800 MPU. It should be noted that the 16-bit store operations of the HMCS6800 microprocessors (STS and STX etc.) transfer data in the order required by the PTM. A Store Index Register Instruction, for example, results in the MSB of the X register being transferred to

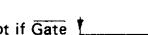
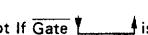
the selected address, then the LSB of the X register being written into the next higher location. Thus, either the index register or stack pointer may be transferred directly into a selected counter latch with a single instruction.

A logic "Low" at the RES input also initializes the counter latches. In this case, all latches will assume a maximum count of  $(65,536)_{10}$ . It is important to note that an Internal Reset (Bit 0 of Control Register 1 Set) has no effect on the counter latches.

#### ■ COUNTER INITIALIZATION

Counter Initialization is defined as the transfer of data from the latches to the counter with subsequent clearing of the Individual Interrupt Flag associated with the counter. Counter Initialization always occurs when a reset

Table 3 Control Register Programming

	Register 1	Register 2	Register 3
7   6   5   4   3   2   1   0	"0" All Timers Operate	Reg #3 May Be Written	T3 Clk ÷ 1
X   X   X   X   X   X   X   $\ddagger$	"1" All Timers Preset	Reg #1 May Be Written	T3 Clk ÷ 8
7   6   5   4   3   2   1   0	"0" External Clock ( $\bar{C}_X$ Input)		
X   X   X   X   X   X   $\ddagger$   X	"1" Internal Clock (Enable)		
7   6   5   4   3   2   1   0	"0" Normal (16-Bit) Count Mode		
X   X   X   X   X   $\ddagger$   X   X	"1" Dual 8-Bit Count Mode		
7   6   5   4   3   2   1   0	Continuous Operating Mode: Gate $\downarrow$ or Write to Latches or Reset Causes Counter Initialization		
X   X   0   0   0   X   X   X			
7   6   5   4   3   2   1   0	Frequency Comparison Mode: Interrupt if Gate  is < Counter Time Out		
X   X   0   0   1   X   X   X			
7   6   5   4   3   2   1   0	Continuous Operating Mode: Gate $\downarrow$ or Reset Causes Counter Initialization		
X   X   0   1   0   X   X   X			
7   6   5   4   3   2   1   0	Pulse Width Comparison Mode: Interrupt if Gate  is < Counter Time Out		
X   X   0   1   1   X   X   X			
7   6   5   4   3   2   1   0	Single Shot Mode: Gate $\downarrow$ or Write to Latches or Reset Causes Counter Initialization		
1   X   1   0   0   X   X   X			
7   6   5   4   3   2   1   0	Frequency Comparison Mode: Interrupt If Gate  is > Counter Time Out		
X   X   1   0   1   X   X   X			
7   6   5   4   3   2   1   0	Single Shot Mode: Gate $\downarrow$ or Reset Causes Counter Initialization		
1   X   1   1   0   X   X   X			
7   6   5   4   3   2   1   0	Pulse Width Comparison Mode: Interrupt If Gate  is > Counter Time Out		
X   X   1   1   1   X   X   X			
7   6   5   4   3   2   1   0	"0" Interrupt Flag Masked (IRQ)		
X   $\ddagger$   X   X   X   X   X   X	"1" Interrupt Flag Enabled (IRQ)		
7   6   5   4   3   2   1   0	"0" Timer Output Masked		
$\ddagger$   X   X   X   X   X   X   X	"1" Timer Output Enable		

(NOTE) Reset is Hardware or Software Reset (RES = "Low" or CR10 = "1").

condition (RES = "Low" or CR10 = "1") is recognized. It can also occur — depending on Timer Mode — with a Write Timer Latches command or recognition of a negative transition of the Gate input.

Counter recycling or re-initialization occurs when a negative transition of the clock input is recognized after the counter has reached an all-zero state. In this case, data is transferred from the Latches to the Counter.

#### ■ TIMER OPERATING MODES

The HD6840 has been designed to operate effectively in a wide variety of applications. This is accomplished by using three bits of each control register (CRX3, CRX4, and CRX5) to define different operating modes of the Timers. These modes are outlined in Table 4.

Table 4 Operating Modes

Control Register			Timer Operating Mode
CRX3	CRX4	CRX5	
0	*	0	Continuous
0	*	1	Single-Shot
1	0	*	Frequency Comparison
1	1	*	Pulse Width Comparison

\* Defines Additional Timer Functions.

In addition to the four timer modes in Table 4, the remaining control register bit is used to modify counter initialization and enabling or interrupt conditions.

#### ● Continuous Operating Mode (Table 5)

Any of the timers in the PTM may be programmed to operate in a continuous mode by writing "0's into bits 3

and 5 of the corresponding control register. Assuming that the timer output is enabled (CRX7 = "1"), either a square wave or a variable duty cycle waveform will be generated at the Timer Output, OX. The type of output is selected via Control Register Bit 2.

Either a Timer Reset (CR10 = "1" or External RES = "Low") condition or internal recognition of a negative transition of the Gate input results in Counter Initialization. A Write Timer Latches command can be selected as a Counter Initialization signal by clearing CRX4.

The counter is enabled by an absence of a Timer Reset condition and a "Low" level at the Gate input. The counter will then decrement on the first clock signal recognized during or after the counter initialization cycle. It continues to decrement on each clock signal so long as G remains "Low" and no reset condition exists. A Counter Time Out (the first clock after all counter bits = "0") results in the Individual Interrupt Flag being set and re-initialization of the counter.

A special condition exists for the dual 8-bit mode (CRX2 = "1") if L = "0". In this case, the counter will revert to a mode similar to the single 16-bit mode, except Time Out occurs after M+1 clock pulses. The output, if enabled, goes "Low" during the Counter Initialization cycle and reverses state at each Time Out. The counter remains cyclical (is re-initialized at each Time Out) and the Individual Interrupt Flag is set when Time Out occurs. If M = L = "0", the internal counters do not change, but the output toggles at a rate of 1/2 the clock frequency.

In the dual 8-bit mode (CRX2 = "1") [Refer to the example in Fig. 9] the MSB decrements once for every full countdown of the LSB + 1. When the LSB = "0", the

Table 5 Continuous Operating Modes

CONTINUOUS MODE (CRX3 = "0", CRX5 = "0")			
Control Register		Initialization/Output Waveforms	
CRX2	CRX4	Counter Initialization	*Timer Output (OX) (CRX7 = "1")
0	0	$\bar{G} \downarrow + W + R$	
	1	$\bar{G} \downarrow + R$	
1	0	$\bar{G} \downarrow + W + R$	
	1	$\bar{G} \downarrow + R$	

$\bar{G} \downarrow$  = Negative transition of Gate input.

W = Write Timer Latches Command.

R = Timer Reset (CR10 = "1" or External RES = "Low")

N = 16-Bit Number in Counter Latch.

L = 8-Bit Number in LSB Counter Latch.

M = 8-Bit Number in MSB Counter Latch.

T = Clock Input Negative Transitions to Counter.

$t_0$  = Counter Initialization Cycle.

TO = Counter Time Out (All Zero Condition).

\* All time intervals shown above assume the Gate ( $\bar{G}$ ) and Clock ( $\bar{C}$ ) signals are synchronized to Enable (System  $\phi_2$ ) with the specified setup and hold time requirements.

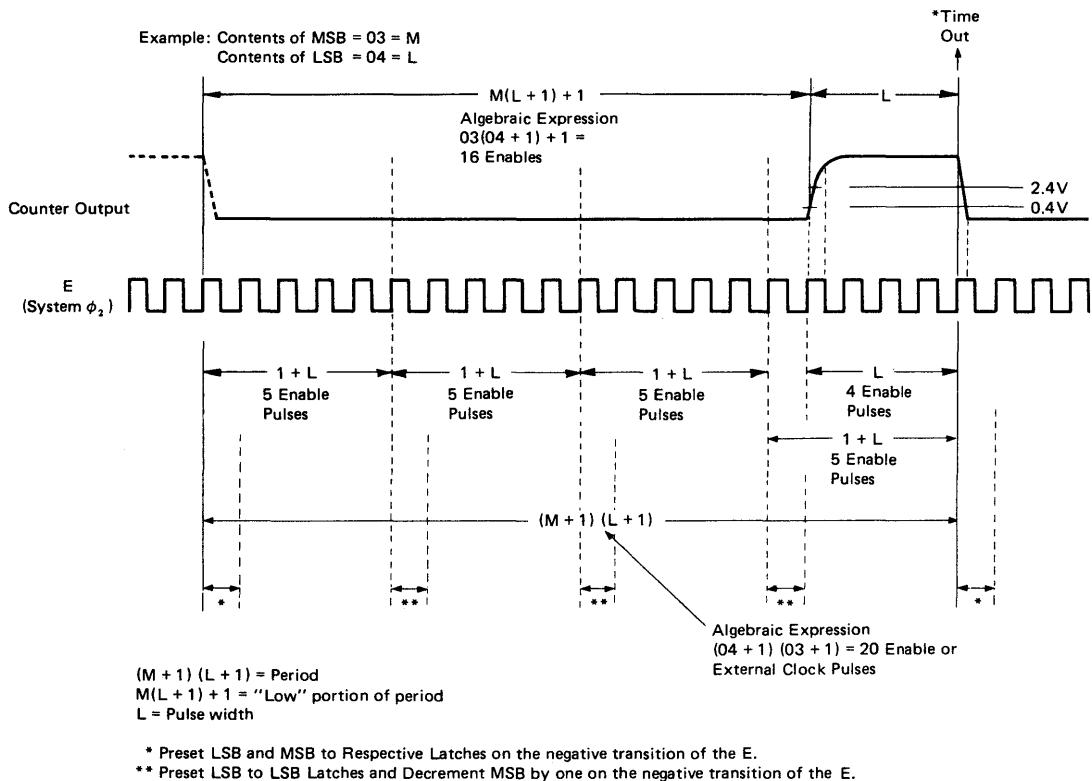


Figure 9 Timer Output Waveform Example  
(Continuous Dual 8-Bit Mode using Internal Enable)

MSB is unchanged; on the next clock pulse the LSB is reset to the count in the LSB Latches and the MSB is decremented by 1 (one). The output, if enabled, remains "Low" during and after initialization and will remain "Low" until the counter MSB is all "0's. The output will go "High" at the beginning of the next clock pulse. The output remains "High" until both the LSB and MSB of the counter are all "0's. At the beginning of the next clock pulse the defined Time Out (TO) will occur and the output will go "Low". In the normal 16-bit mode the period of the output of the example in Fig. 9 would span 1546 clock pulses as opposed to the 20 clock pulses using the Dual 8-bit mode.

The discussion of the Continuous Mode has assumed that the application requires an output signal. It should be noted that the Timer operates in the same manner with the output disabled (CRX7 = "0"). A Read Timer Counter command is valid regardless of the state of CRX7.

#### • Single-Shot Timer Mode

This mode is identical to the Continuous Mode with three exceptions. The first of these is obvious from the name — the output returns to a "Low" level after the initial Time Out and remains "Low" until another Counter Initialization cycle occurs. The waveforms availa-

ble are shown in Table 6.

As indicated in Table 6, the internal counting mechanism remains cyclical in the Single-Shot Mode. Each Time Out of the counter results in the setting of an Individual Interrupt Flag and re-initialization of the counter.

The second major difference between the Single-Shot and Continuous modes is that the internal counter enable is not dependent on the Gate input level remaining in the "Low" state for the Single-Shot mode.

Another special condition is introduced in the Single-Shot mode. If  $L = M = "0"$  (Dual 8-bit) or  $N = "0"$  (Single 16-bit), the output goes "Low" on the first clock received during or after Counter Initialization. The output remains "Low" until the Operating Mode is changed or nonzero data is written into the Counter Latches. Time Outs continue to occur at the end of each clock period.

The three differences between Single-Shot and Continuous Timer Modes can be summarized as attributes of the Single-Shot mode:

1. Output is enabled for only one pulse until it is reinitialized.
2. Counter Enable is independent of Gate.
3.  $L = M = "0"$  or  $N = "0"$  disables output.

Aside from these differences, the two modes are identical.

- Frequency Comparison or Period Measurement Mode (CRX3 = "1", CRX4 = "0")

The Frequency Comparison Mode with CRX5 = "1" is straightforward. If Time Out occurs prior to the first negative transition of the Gate input after a Counter Initialization cycle, an Individual Interrupt Flag is set. The counter is disabled, and a Counter Initialization cycle cannot begin until the interrupt flag is cleared and a negative transition on  $\overline{G}$  is detected.

Table 6 Single-Shot Operating Modes

Single-Shot Mode (CRX3 = "0", CRX7 = "1", CRX5 = "1")			
Control Register		Initialization/Output Waveforms	
CRX2	CRX4	Counter Initialization	Timer Output (OX)
0	0	$\overline{G} \downarrow + W + R$	
0	1	$\overline{G} \downarrow + R$	
1	0	$\overline{G} \downarrow + W + R$	
1	1	$\overline{G} \downarrow + R$	

Symbols are as defined in Table 5.

- Time Interval Modes

The Time Interval Modes are provided for those applications which require more flexibility of interrupt generation and Counter Initialization. Individual Interrupt Flags are set in these modes as a function of both Counter Time Out and transitions of the Gate input. Counter Initialization is also affected by Interrupt Flag status.

The counter does operate in either Single 16-bit or Dual 8-bit modes as programmed by CRX2. Other features of the Time Interval Modes are outlined in Table 7.

If CRX5 = "0", as shown in Table 7 and Table 8, an interrupt is generated if  $\overline{G}$  input returns "Low" prior to a Time Out. If Counter Time-Out occurs first, the counter is recycled and continues to decrement. A bit is set within the timer on the initial Time Out which precludes further individual interrupt generation until a new Counter Initialization cycle has been completed. When this internal bit is set, a negative transition of the  $\overline{G}$  input starts a new Counter Initialization cycle. (The condition of  $\overline{G} \downarrow \cdot TO$  is satisfied, since a Time Out has occurred and no individual Interrupt has been generated.)

Table 7 Time Interval Modes

CRX3 = "1"			
CRX4	CRX5	Application	Condition for Setting Individual Interrupt Flag
0	0	Frequency Comparison	Interrupt Generated if Gate Input Period (1/F) is less than Counter Time Out (TO)
0	1	Frequency Comparison	Interrupt Generated if Gate Input Period (1/F) is greater than Counter Time Out (TO)
1	0	Pulse Width Comparison	Interrupt Generated if Gate Input "Down Time" is less than Counter Time Out (TO)
1	1	Pulse Width Comparison	Interrupt Generated if Gate Input "Down Time" is greater than Counter Time Out (TO)

Any of the timers within the PTM may be programmed to compare the period of a pulse (giving the frequency after calculations) at the  $\overline{G}$  input with the time period required for Counter Time-Out. A negative transition of the  $\overline{G}$  input enables the counter and starts a Counter Initialization cycle — provided that other conditions as noted in Table 8 are satisfied. The counter decrements on each clock signal recognized during or after Counter Initialization until an interrupt is generated, a Write Timer Latches command is issued, or a Timer Reset condition occurs. It can be seen from Table 8 that an interrupt con-

dition will be generated if CRX5="0" and the period of the pulse (single pulse or measured separately repetitive pulses) at the  $\overline{G}$  input is less than the Counter Time Out period. If CRX5 = "1", an interrupt is generated if the reverse is true.

Assume now with CRX5 = "1" that a Counter Initialization has occurred and that the  $\overline{G}$  input has returned "Low" prior to Counter Time Out. Since there is no Individual Interrupt Flag generated, this automatically starts a new Counter Initialization Cycle. The process will continue with frequency comparison being performed

on each Gate input cycle until the mode is changed, or a cycle is determined to be above the predetermined limit.

- **Pulse Width Comparison Mode (CRX3 = "1", CRX4 = "1")**

This mode is similar to the Frequency Comparison Mode except for a positive, rather than negative, transition of the Gate input terminates the count. With CRX5 = "0", an Individual Interrupt Flag will be generated if the "Low" level pulse applied to the Gate input is less than

the time period required for Counter Time Out. With CRX5 = "1", the interrupt is generated when the reverse condition is true.

As can be seen in Table 9, a positive transition of the Gate input disables the counter. With CRX5 = "0", it is therefore possible to directly obtain the width of any pulse causing an interrupt. Similar data for other Time Interval Modes and conditions can be obtained, if two sections of the PTM are dedicated to the purpose.

Table 8 Frequency Comparison Mode

CRX3 = "1", CRX4 = "0"				
Control Reg Bit 5 (CRX5)	Counter Initialization	Counter Enable Flip-Flop Set (CE)	Counter Enable Flip-Flop Reset (CE)	Interrupt Flag Set (I)
0	$\overline{G} \downarrow \cdot \overline{T} \cdot (\overline{CE} + TO) + R$	$\overline{G} \downarrow \cdot W \cdot \overline{R} \cdot \overline{T}$	$W + R + I$	$\overline{G} \downarrow$ Before TO
1	$\overline{G} \downarrow \cdot \overline{T} + R$	$\overline{G} \downarrow \cdot W \cdot \overline{R} \cdot \overline{T}$	$W + R + I$	TO Before $\overline{G} \downarrow$

I represents the interrupt for a given timer.

Table 9 Pulse Width Comparison Mode

CRX3 = "1", CRX4 = "1"				
Control Reg Bit 5 (CRX5)	Counter Initialization	Counter Enable Flip-Flop Set (CE)	Counter Enable Flip-Flop Reset (CE)	Interrupt Flag Set (I)
0	$\overline{G} \downarrow \cdot \overline{T} + R$	$\overline{G} \downarrow \cdot W \cdot \overline{R} \cdot \overline{T}$	$W + R + I + G$	$\overline{G} \uparrow$ Before TO
1	$\overline{G} \downarrow \cdot \overline{T} + R$	$\overline{G} \downarrow \cdot W \cdot \overline{R} \cdot \overline{T}$	$W + R + I + G$	TO Before $\overline{G} \uparrow$

G = Level sensitive recognition of Gate input.

HITACHI reserves the right to make changes to any products herein to improve functioning or design. Although the information in this document has been carefully reviewed and is believed to be reliable, HITACHI does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights or others.

# HD6843S, HD68A43S

## FDC (Floppy Disk Controller)

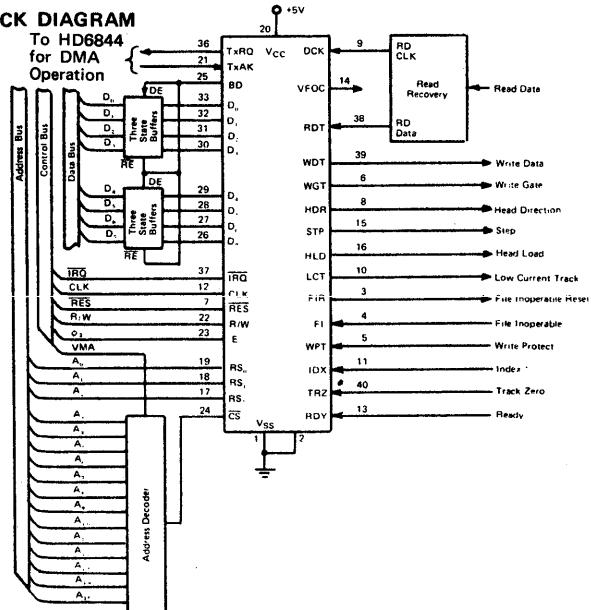
The HD6843SP Floppy Disk Controller performs the complex MPU/Floppy interface function. The FDC was designed to optimize the balance between the "Hardware/Software" in order to achieve integration of all key functions and maintain flexibility.

The FDC can interface a wide range of drives with a minimum of external hardware. Multiple drives can be controlled with the addition of external multiplexing rather than additional FDC's.

### ■ FEATURES

- Format compatible with IBM3740
- User Programmable read/write format
- Ten powerful macro-commands
- Macro End Interrupt allows parallel processing of MPU and FDC
- Controls multiple Floppies with external multiplexing
- Direct interface with HMCS6800
- Programmable seek and settling times enable operation with a wide range of Floppy drives
- Offers both Programmed Controlled I/O (PCIO) and DMA data transfer mode
- Free-Format read or write
- Single 5-volt power supply
- All registers directly accessible
- Compatible with MC6843\*

### ■ BLOCK DIAGRAM



### ■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V <sub>CC</sub> *	-0.3 ~ +7.0	V
Input Voltage	V <sub>in</sub> *	-0.3 ~ +7.0	V
Operating Temperature	T <sub>opr</sub>	-20 ~ +75	°C
Storage Temperature	T <sub>stg</sub>	-55 ~ +150	°C

\* With respect to V<sub>SS</sub> (SYSTEM GND)

(NOTE) 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.

### ■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol	min.	typ.	max.	Unit
Supply Voltage	V <sub>CC</sub> *	4.75	5.0	5.25	V
Input "High" Voltage	V <sub>IH</sub> *	2.0	—	V <sub>CC</sub>	V
Input "Low" Voltage	V <sub>IL</sub> *	-0.3	—	0.8	V
Operating Temperature	T <sub>opr</sub>	-20	25	75	°C

\* With respect to V<sub>SS</sub> (SYSTEM GND)

### ■ ELECTRICAL CHARACTERISTICS

#### ● DC CHARACTERISTICS (V<sub>CC</sub>=5V±5%, V<sub>SS</sub>=0V, T<sub>a</sub>=-20~+75°C, unless otherwise noted.)

Item	Symbol	Test Condition	min.	typ.*	max.	Unit
Input "High" Voltage	V <sub>IH</sub>		2.0	—	V <sub>CC</sub>	V
Input "Low" Voltage	V <sub>IL</sub>		-0.3	—	0.8	V
Input Leakage Current	I <sub>IN</sub>	V <sub>in</sub> =0~5.25V	—	1.0	2.5	μA
Output "High" Voltage	V <sub>OH</sub>	I <sub>OH</sub> =-205μA (D <sub>0</sub> ~D <sub>7</sub> ) I <sub>OH</sub> =-100μA (Others)	2.4	—	—	V
Output "Low" Voltage	V <sub>OL</sub>	I <sub>OL</sub> =3.2mA (I <sub>RO</sub> ) I <sub>OL</sub> =1.6mA (Others)	—	—	0.4	V
Three-state (off-state) Leakage Current	I <sub>TSI</sub>	V <sub>in</sub> =0.4~2.4V	—	2.0	10	μA
Output Leakage (off-state) Current (I <sub>RO</sub> )	I <sub>LOH</sub>	V <sub>OH</sub> =2.4V	—	1.0	10	μA
Power Dissipation	P <sub>D</sub>		—	600	1000	mW
Input Capacitance Other inputs	C <sub>in</sub>	V <sub>in</sub> =0V, T <sub>a</sub> =25°C f=1 MHz	—	—	12.5	pF
			—	—	10	pF
Output Capacitance	C <sub>out</sub>	V <sub>in</sub> =0V, T <sub>a</sub> =25°C f=1 MHz	—	—	10	pF

\* V<sub>CC</sub> = 5V, T<sub>a</sub> = 25°C

● AC CHARACTERISTICS ( $V_{CC}=5V\pm5\%$ ,  $V_{SS}=0V$ ,  $T_a=-20\sim+75^\circ C$ , unless otherwise noted.)

Item	Symbol	Test Condition	HD6843S			HD68A43S			Unit
			min.	typ.	max.	min.	typ.	max.	
CLK Cycle Time	$t_{cycC}$	Figure 1	—	1.0	—	—	1.0	—	μs
CLK Pulse Width, "High"	$PW_{HC}$	Figure 1	0.4	—	—	0.4	—	—	μs
CLK Pulse Width, "Low"	$PW_{LC}$	Figure 1	0.35	—	—	0.35	—	—	μs
Rise and Fall Time of CLK	$t_{Cr}, t_{Cf}$	Figure 1	—	—	25	—	—	25	ns
DCK Cycle Time	$t_{cycD}$	Figure 2	2.6	4.0	—	2.6	4.0	—	μs
DCK Pulse Width, "High"	$PW_{HD}$	Figure 2	1.3	1.95	—	1.3	1.95	—	μs
DCK Pulse Width, "Low"	$PW_{LD}$	Figure 2	1.3	1.95	—	1.3	1.95	—	μs
Rise and Fall Time of DCK	$t_{Dr}, t_{Df}$	Figure 2	—	—	25	—	—	25	ns
RDT Width, "High"	$t_{RDH}$	Figure 2	1.0	—	—	1.0	—	—	μs
RDT Width, "Low"	$t_{RDL}$	Figure 2	1.0	—	—	1.0	—	—	μs
RDT~DCK Delay Time 1	$t_{RDD1}$	Figure 2	0.15	—	1.70	0.15	—	1.70	μs
RDT~DCK Delay Time 2	$t_{RDD2}$	Figure 2	0.15	—	1.70	0.15	—	1.70	μs
IDX Pulse Width, "High"	$PW_{IDX}$	Figure 3	20.0	—	—	20.0	—	—	μs
FIR Delay Time	$t_{FIRD}$	Figure 4	—	—	450	—	—	450	ns
FIR Pulse Width, "High"	$PW_{FIR}$	Figure 4	200	—	—	200	—	—	ns
WDT Pulse Width, "High"	$PW_{WD}$	Figure 7	—	1.0	—	—	1.0	—	μs
WDT Cycle Time	$t_{cycW}$	Figure 7	—	2.0	—	—	2.0	—	μs
STP Pulse Width, "High"	$PW_{STP}$	Figure 5	—	32	—	—	32	—	μs
STP Cycle Time	$t_{cycS}^*$	Figure 5	1	—	15	1	—	15	ms
HLD Delay Time (HLD~STP)	$t_{HLDD}$	Figure 5	1	—	15	1	—	15	μs
HDR Set Up Time	$t_{HDRS}$	Figure 5	0	—	—	0	—	—	ns
HDR Hold Time	$t_{HDRH}$	Figure 5	32	—	—	32	—	—	μs
TxAK Set Up Time	$t_{AS3}$	Figure 10, 11	140	—	—	140	—	—	ns
TxAK Hold Time	$t_{AH3}$	Figure 10, 11	10	—	—	10	—	—	ns
TxRQ Release Time	$t_{TR}$	Figure 10, 11	—	—	450	—	—	240	ns
IRQ Release Time	$t_{IR}$	Figure 6	—	—	1.2	—	—	1.2	μs

\* Cycle Time of STP changes according to the program.

● BUS TIMING CHARACTERISTICS

1 READ OPERATION SEQUENCE

Item	Symbol	Test Condition	HD6843S			HD68A43S			Unit
			min.	typ.	max.	min.	typ.	max.	
Enable Cycle Time	$t_{cycE}$	Figure 8, 10	1.0	—	—	0.666	—	—	μs
Enable Pulse Width, "High"	$PW_{EH}$	Figure 8, 10	0.4	—	—	0.23	—	—	μs
Enable Pulse Width, "Low"	$PW_{EL}$	Figure 8, 10	0.4	—	—	0.23	—	—	μs
Rise and Fall Time of Enable Input	$t_{Er}, t_{Ef}$	Figure 8, 10	—	—	25	—	—	25	ns
Address Set Up Time	$t_{AS}$	Figure 8, 10	140	—	—	140	—	—	ns
Data Delay Time	$t_{DDR}$	Figure 8, 10	—	—	225	—	—	200	ns
Data Access Time	$t_{ACC}$	Figure 8, 10	—	—	365	—	—	340	ns
Data Hold Time	$t_H$	Figure 8, 10	10	—	—	10	—	—	ns
Address Hold Time	$t_{AH}$	Figure 8, 10	10	—	—	10	—	—	ns
Bus Direction Delay Time	$t_{DBD}$	Figure 8, 10	—	—	400	—	—	400	ns

## 2 WRITE OPERATION SEQUENCE

Item	Symbol	Test Condition	HD6843S			HD68A43S			Unit
			min.	typ.	max.	min.	typ.	max.	
Enable Cycle Time	$t_{cycE}$	Figure 9, 11	1.0	—	—	0.666	—	—	μs
Enable Pulse Width, "High"	$PW_{EH}$	Figure 9, 11	0.4	—	—	0.23	—	—	μs
Enable Pulse Width, "Low"	$PW_{EL}$	Figure 9, 11	0.4	—	—	0.23	—	—	μs
Rise and Fall Time of Enable Input	$t_{Er}, t_{Ef}$	Figure 9, 11	—	—	25	—	—	25	ns
Address Set Up Time	$t_{AS}$	Figure 9, 11	140	—	—	140	—	—	ns
Data Set Up Time	$t_{DSW}$	Figure 9, 11	100	—	—	60	—	—	ns
Data Hold Time	$t_H$	Figure 9, 11	10	—	—	10	—	—	ns
Address Hold Time	$t_{AH}$	Figure 9, 11	10	—	—	10	—	—	ns

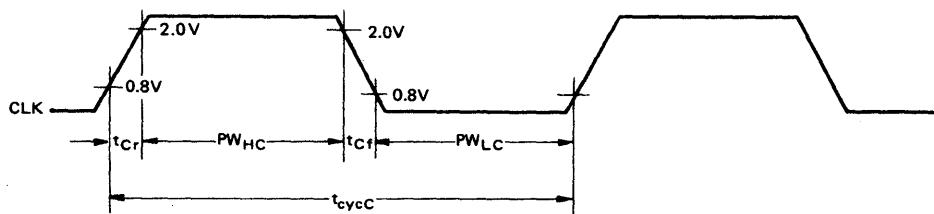


Figure 1 CLK Waveform

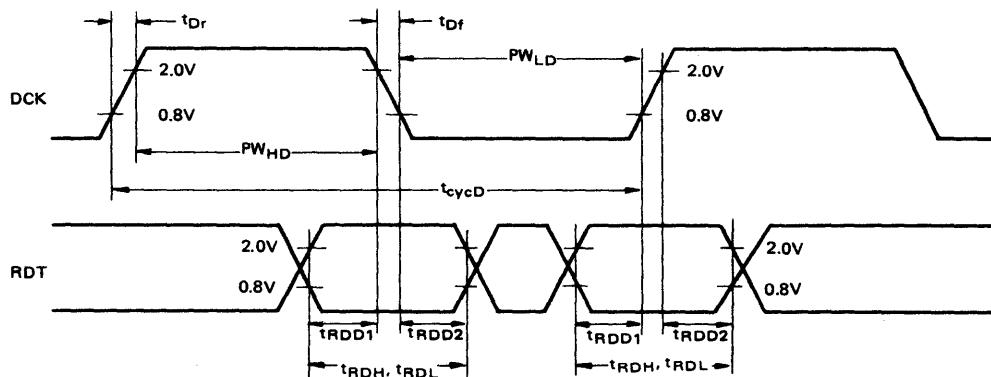


Figure 2 DCK, RDT Timing

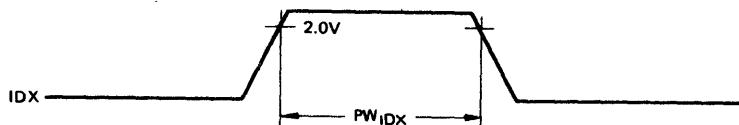


Figure 3 IDX Waveform

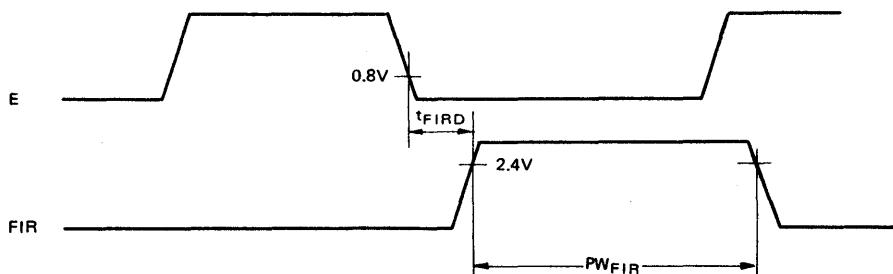


Figure 4 FIR Timing

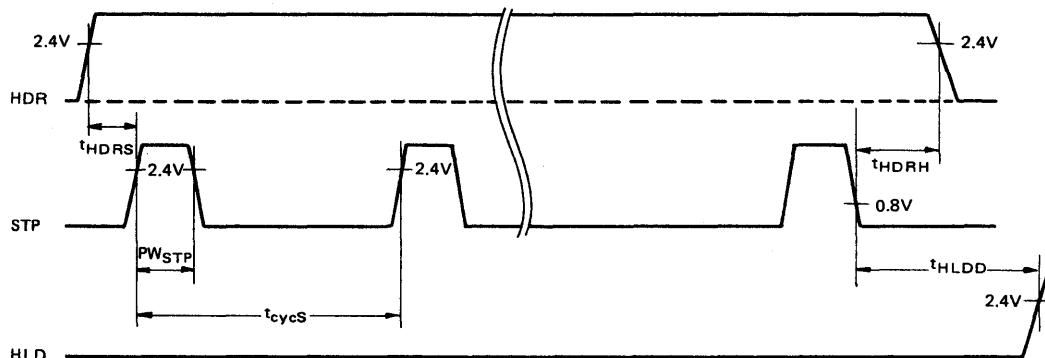


Figure 5 Seek Operation Sequence

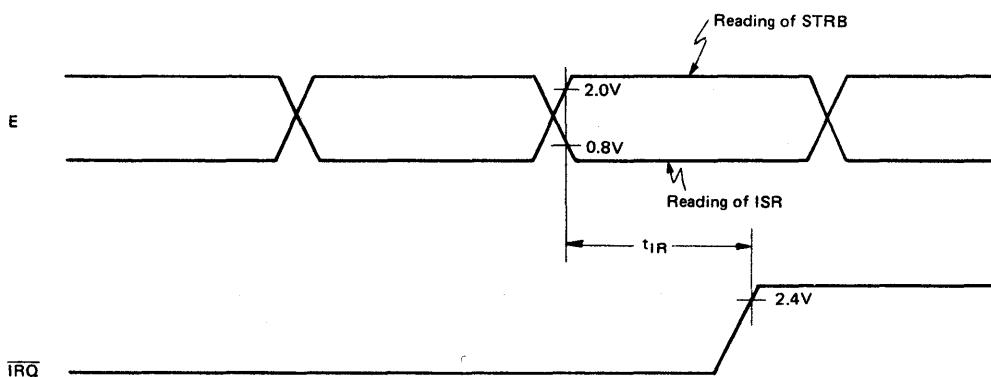


Figure 6  $\overline{iRQ}$  Release Timing

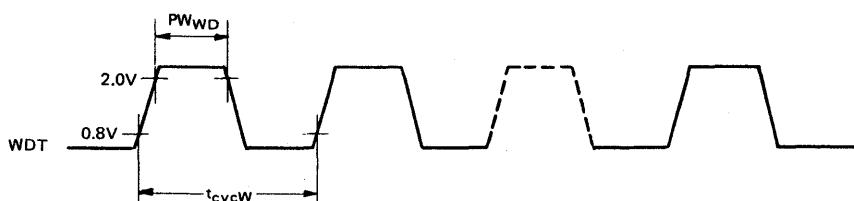


Figure 7 WDT Waveform

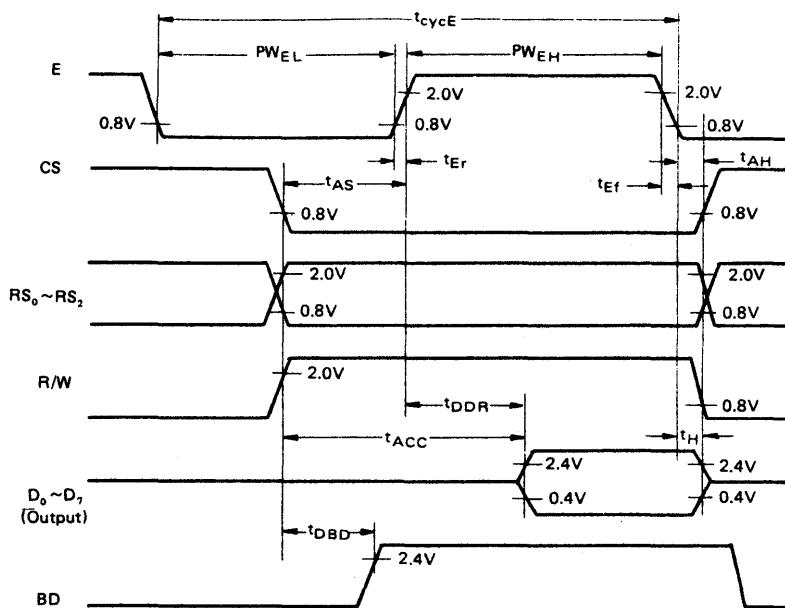


Figure 8 Read Timing

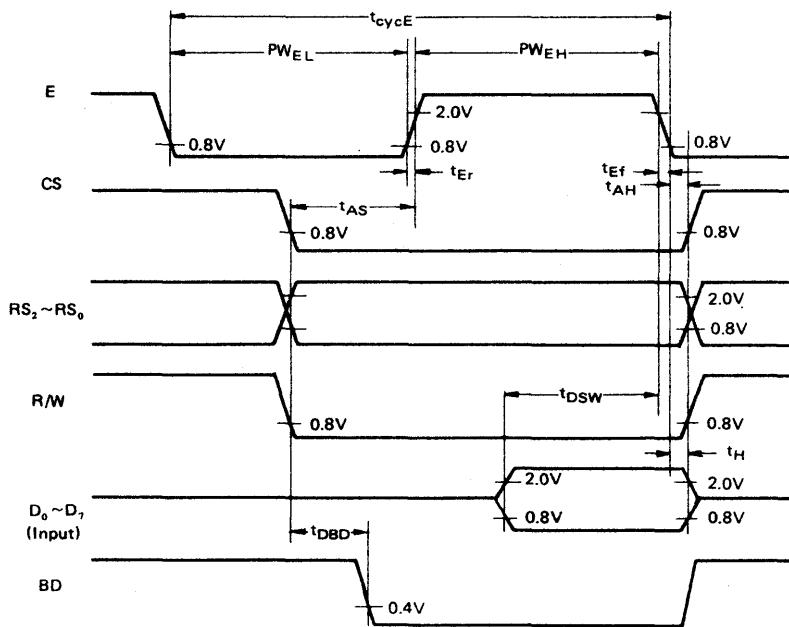


Figure 9 Write Timing

---

**HD6843S, HD68A43S**

---

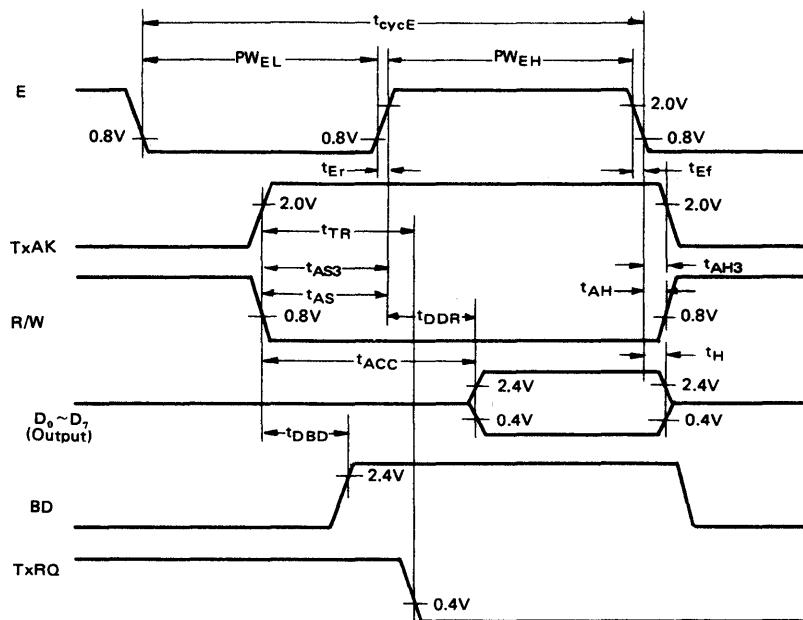


Figure 10 DMA Read Timing

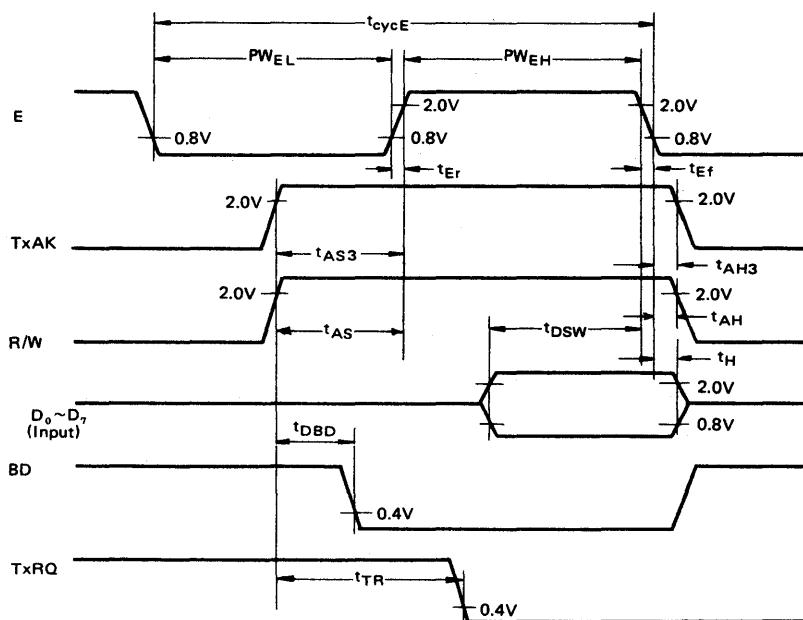


Figure 11 DMA Write Timing

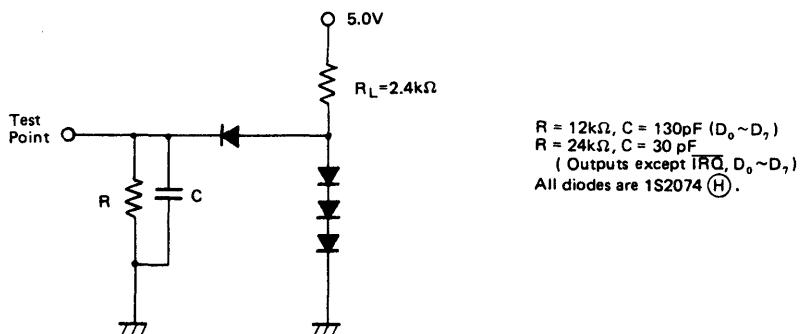
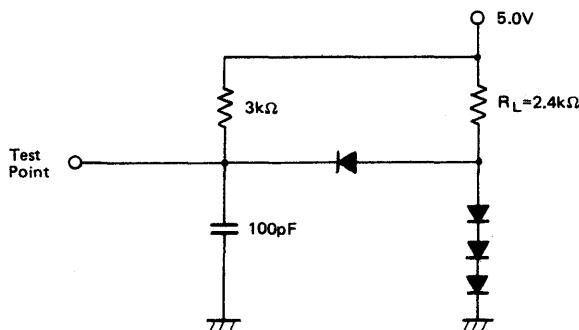
LOAD A (Except  $\overline{IRQ}$ )LOAD B ( $\overline{IRQ}$ )

Figure 12 Load Circuit

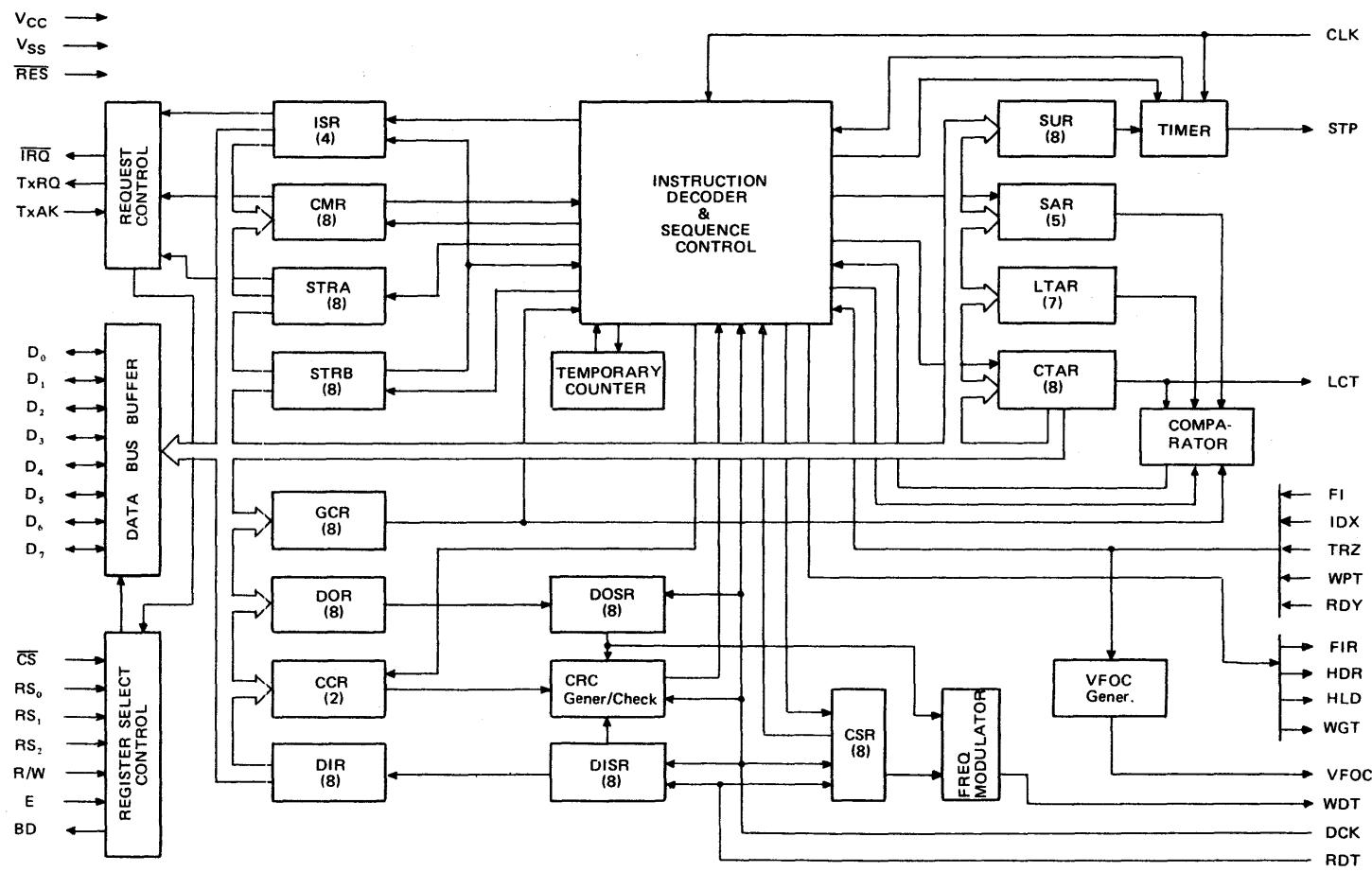


Figure 13 Block Diagram of the FDC

## ■ GENERAL DESCRIPTION

The HD6843SP FDC consists of four primary sections; the Register, Serializing, Bus Interface, and Control sections. The following explanation of these sections can be followed in the block diagram of Figure 13.

### ● Register Section

The register section consists of twelve user accessible registers used for controlling a floppy disk drive. All twelve are connected by the internal data bus to allow the processor access to them.

### Data Output Register (DOR)

The DOR is an 8-bit register which holds the data to be written onto the disk. The information is stored here by the bus interface.

### Data Input Register (DIR)

The data words read from the disk are stored in the 8-bit DIR until read by the bus interface.

### Current Track Address Register (CTAR)

CTAR is a 8-bit register containing the address of the track over which the R/W head is currently positioned.

### Command Register (CMR)

The macro commands are written to the 8-bit CMR to begin their execution.

### Interrupt Status Register (ISR)

The four bits of the ISR represent the four conditions that can cause an interrupt to occur.

### Set-Up Register (SUR)

Variable Seek and Settling times are programmed by the SUR. Four bits are used to program the track to track seek time and four bits are used to program the head settling time for the floppy disk drive used with the FDC.

### Status Register A (STRA)

The eight bits of STRA are used to indicate the state of the floppy disk interface.

### Sector Address Register (SAR)

SAR contains the five bit sector address associated with the current data transfer.

### Status Register B (STRB)

The eight error flags of STRB are used to signify error conditions detected by the FDC or generated by the floppy disk drive.

### General Count Register (GCR)

The seven bits of GCR contain the destination track address when a SEK (seek) macro command is being executed. If a multi-sector Read or Write macro command is being executed, GCR contains the number of sectors to be read or written.

### CRC Control Register (CCR)

The two bits of the CCR are used to enable the CRC and shift the CRC for the Free Format Commands.

### Logical Track Address Register (LTAR)

The seven bit track address used for read and write

operations is stored in the LTAR by the bus interface.

### ● Serializing Section

The serializing section handles the serial-to-parallel and parallel-to-serial conversions for Read/Write operations as well as CRC generation/checking and the generation/detection of the clock pattern. The Data Output Shift Register (DOSR), Data Input Shift Register (DISR), CRC Generator/Checker, and Clock Shift Register (CSR) comprise the serializing section of the FDC.

### ● Bus Interface

The Bus Interface section provides the timing and control logic that allows the FDC to operate with the 6800 bus, and is comprised of the Data Buffers, Request Control, and the Register Select circuitry.

### ● Control

The internal timing and control signals which sequence the FDC are derived from the macro instructions by the control section.

## ■ HD6843SP PIN DESCRIPTION

### ● Power Pins

V<sub>CC</sub>: +5 volt ( $\pm 5\%$ ) power input.

V<sub>SS</sub>: Power Supply Ground.

### ● Bus Pins

#### Reset (RES) Input

The RES input is used to initialize the FDC. When RES becomes "Low", the state of the outputs is defined by the table below:

Output	State of Output	Output	State of Output
FIR	"Low"	HLD	"Low"
WGT	"Low"	TxRO	"Low"
HDR	"Low"	IRQ	"High"
STP	"Low"	WDT	"Low"

Registers which are affected by RES are shown in Table 7.

#### Interrupt Request (IRQ) Output

The IRQ line is an open drain output that becomes a "Low" level (logic "0") when the FDC requests an interrupt. Interrupt requests are controlled by the interrupt enables in CMR (Command Register) with the function causing the interrupt shown in ISR (Interrupt Status Register).

#### Data Bus 0~Data Bus 7 (D<sub>0</sub>~D<sub>7</sub>) Bidirectional

The 8 bidirectional data lines allow the transfer of data between the FDC and the controlling system. The output buffers are three-state drivers that are enabled when the FDC is transferring data to the data bus.

#### Enable (E) Input

The E input to the FDC causes data transfers to occur between the FDC and the system controlling the FDC

(HMCS6800 MPU, DMA Controller, etc.) E must be a logic "1" ("High" level) for any transfer to be enabled on  $D_0 \sim D_7$ . The E input is normally connected to system  $\phi_2$ .

#### Chip Select ( $\overline{CS}$ ) Input

The  $\overline{CS}$  input in conjunction with the E input, is used to enable data transfers on  $D_0 \sim D_7$ . E must be a "High" level and  $\overline{CS}$  must be a "Low" level (logic "0") to enable the transfer. The TxAK input being a "High" level (logic "1") performs a similar function as  $\overline{CS}$  being a "Low" level.

#### Read/Write (R/W) Input

The R/W input is issued by the system controlling the FDC (HMCS6800 MPU, DMA Controller, etc.) to signify if a read or write operation is to be performed on the FDC. When TxAK is a "Low" level, R/W is used in conjunction with  $\overline{CS}$  and  $RS_0 \sim RS_2$  to determine which register is accessed by the bus as shown in Table 1. When TxAK is a "High" level, R/W is used to select either the DOR or DIR to the data bus (see description of TxAK input).

#### Register Select 0~Register Select 2 ( $RS_0 \sim RS_2$ ) Input

$RS_0 \sim RS_2$ , in conjunction with the R/W input, are used to select one of the user accessible registers in the FDC as shown in Table 1.

#### Transfer Request (TxRQ) Output

TxRQ is used in the DMA mode to request a data transfer from the DMAC. TxRQ is a "High" level if the FDC is in the DMA mode (CMR bit 5 is set) when a data transfer request occurs (STRA bit 0 is set). It is reset to a "Low" level (logic "0") when TxAK becomes a "High" level (logic "1"). Data transfer errors will occur if TxAK does not reset TxRQ before the next data transfer is required.

#### Transfer Acknowledge (TxAK) Input

TxAK is generated by the system controlling the FDC (HMCS6800 MPU, DMA Controller, etc.) and is a response to a TxRQ issued by the FDC. A "High" level (logic "1") on TxAK

causes the FDC to neglect the state of  $RS_0 \sim RS_2$  causing the FDC to select the DOR (Data Output Register) or DIR (Data Input Register) to the data bus ( $D_0 \sim D_7$ ) as shown in Table 2.  $\overline{CS} = "0"$  and  $TxAK = "1"$  cannot be permitted at the same time.

Table 2 Register Selection for DMA Transfers

TxAK	$RS_0 \sim RS_2$	$\overline{CS}$	R/W	Register Selected
1	x	1	1	DOR
1	x	1	0	DIR

This mode of operation is normally used for DMA (Direct Memory Access) transfer with the FDC.

When TxAK is a "Low" level the registers are selected by  $\overline{CS}$ , R/W and  $RS_0 \sim RS_2$  as shown in Table 1.

#### Bus Direction (BD) Output

The BD output is provided to control external bidirectional buffers on the data bus ( $D_0 \sim D_7$ ) as shown in Figure 14. Its polarity is shown by Table 3.

Table 3 Bus Direction (BD) States

TxAK	$\overline{CS}$	BD
1	1	R/W
0	1	0
0	0	R/W

(Operation of BD as defined by this chart allows the FDC to function with the DMA Controller HD6844P.)

Table 1 Address Codes for User Accessible Registers

TxAK	$\overline{CS}$	$RS_2$	$RS_1$	$RS_0$	R/W	Registers
0	0	0	0	0	0	DOR (Data Output Register)
					1	DIR (Data Input Register)
0	0	0	0	1	1/0	CTAR (Current Track Address Register)
					0	CMR (Command Register)
0	0	0	1	0	1	ISR (Interrupt Status Register)
					0	SUR (Set Up Register)
0	0	0	1	1	1	STRA (Status Register A)
					0	SAR (Sector Address Register)
0	0	1	0	0	1	STRB (Status Register B)
					0	GCR (General Count Register)
0	0	1	1	0	0	CCR (CRC Control Register)
					0	LTAR (Logical Track Address Register)

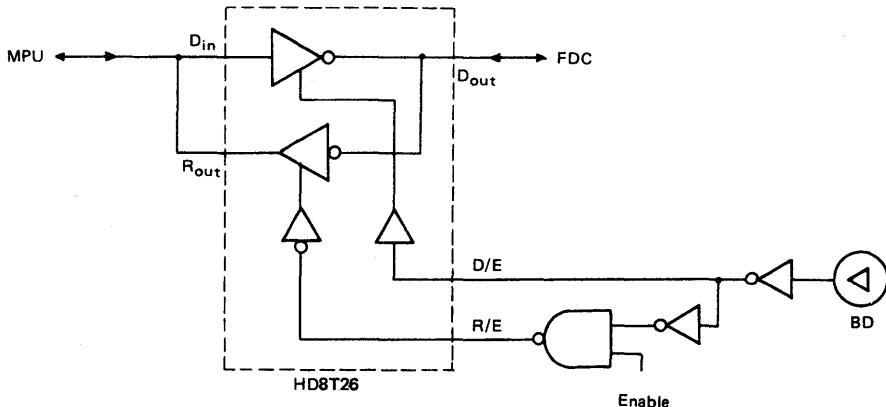


Figure 14 Bus Buffer Control

- I/O and Control Pins

#### Head Load (HLD) Output

HLD is used to notify the disk drive that the R/W head should be loaded (placed in contact with the media). When the FDC is ready for the head to load, HLD is a "High" level (logic "1"). A "Low" level (logic "0") HLD indicates the head should be unloaded.

#### Step (STP) Output

The STP output, in conjunction with HDR, is used to control head movement. A  $32\ \mu s$  wide positive (logic "1") pulse is generated on STP, to move the R/W head one track in the direction defined by the HDR output. The period of the STP signal is programmable by the SUR (Set-Up Register). The number of pulses generated on STP is the difference between the contents of the CTAR (Current Track Address Register), and the GCR (General Count Register) which contains the track address to which the head is to be moved.

#### Head Direction (HDR) Output

The HDR signal controls the direction of head movement. A "High" level (logic "1") signifies the head should step to the inside (toward the hub) of the disk. A "Low" level (logic "0") indicates the direction of head movement should be to the outside of the disk.

#### Low Current Track (LCT) Output

The LCT signal is used to control the level of write current used by the disk drive. LCT is a "Low" level (logic "0") when the write head is positioned over tracks 0~43. If it is over tracks 44~76, LCT is a "High" level (logic "1"). LCT is determined from the contents of the Current Track Address Register (CTAR).

#### Write Gate (WGT) Output

When a write operation is being performed, WGT is a logic "1" ("High" level). For a read operation, WGT is a "Low" level (logic "0").

#### File Inoperable Reset (FIR) Output

FIR is an output from the FDC to the floppy disk drive to reset it from an inoperable status. If the FI input is a "High" level, a pulse, of which width almost equals to E pulse "Low" width, is generated on the FIR output whenever Status Register

B is read.

#### File Inoperable (FI) Input

FI is an input to the FDC from the drive. A "High" level indicates the drive is in an inoperable state. Its current state can be examined by reading bit 5 of Status Register B (STRB).

#### Track Zero (TRZ) Input

The TRZ input is reflected by bit 3 of STRA (Status Register A). The TRZ input must be a "High" level (logic "1") when the R/W head of the drive is positioned over track zero. A logic "1" on this input inhibits step pulses during a Seek Track Zero command.

#### Index (IDX) Input

The index input is received from the floppy disk drive and is used to sense the index hole in the disk media. The IDX signal is used to initialize the internal FDC timing. The state of the IDX input is reflected by bit 6 of Status Register A (STRA). A "High" level (logic "1") is to indicate the index hole is under the index sensor. The index input is used to count the number of disk revolutions while searching for the address ID field (see description of STRB bit 3).

#### Ready (RDY) Input

The ready input is received from the disk drive and can be read as bit 2 of STRA (Status Register A). A "High" level (logic "1") indicates the drive is ready and allows the FDC to operate the drive.

#### Write Protect (WPT) Input

WPT is an input indicating when the media is Write Protected. A "High" level during an FDC write operation results in a Write Error (STRB bit 6) but the FDC continues to perform the write function. The state of the WPT input can be read by examining bit 4 of the Status Register A (STRA).

#### Clock (CLK) Input

The CLK input is used to generate various timing sequences internal to the FDC. The head settling, seek time, step pulse width and write data pulse width, etc., are generated from the CLK input signal. The CLK is 1 MHz frequency and the duty is 50%.

- Data Pins

**Data Clock (DCK) Input**

DCK is used to clock data from the drive into the FDC. It is generated from the read data received from the drive.

**Read Data (RDT) Input**

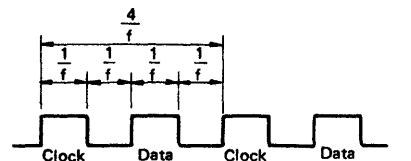
RDT is the serial data input from the drive. The data stream includes both the clock and data bits.

**Write Data (WDT) Output**

WDT is the double frequency modulated data output from the FDC. The time between clock bits is  $4/f$  where  $f$  is the frequency of the clock input. The pulse width for both clock and data is  $1/f$  (see Figure 15). For the normal clock frequency of 1 MHz the clock period is 4  $\mu$ s, the clock pulse width is 1  $\mu$ s and the data pulse width is 1  $\mu$ s. Figure 15 shows the relationship between the WDT output and the frequency of the CLK inputs.

**Variable Frequency Oscillator Control (VFOC) Output**

VFOC is used as a sync signal during system diagnostics. Waveforms are shown in Figure 16.



$f$  = Frequency of the CLK Input. To insure IBM3740 compatibility the clock frequency must be 1 MHz.

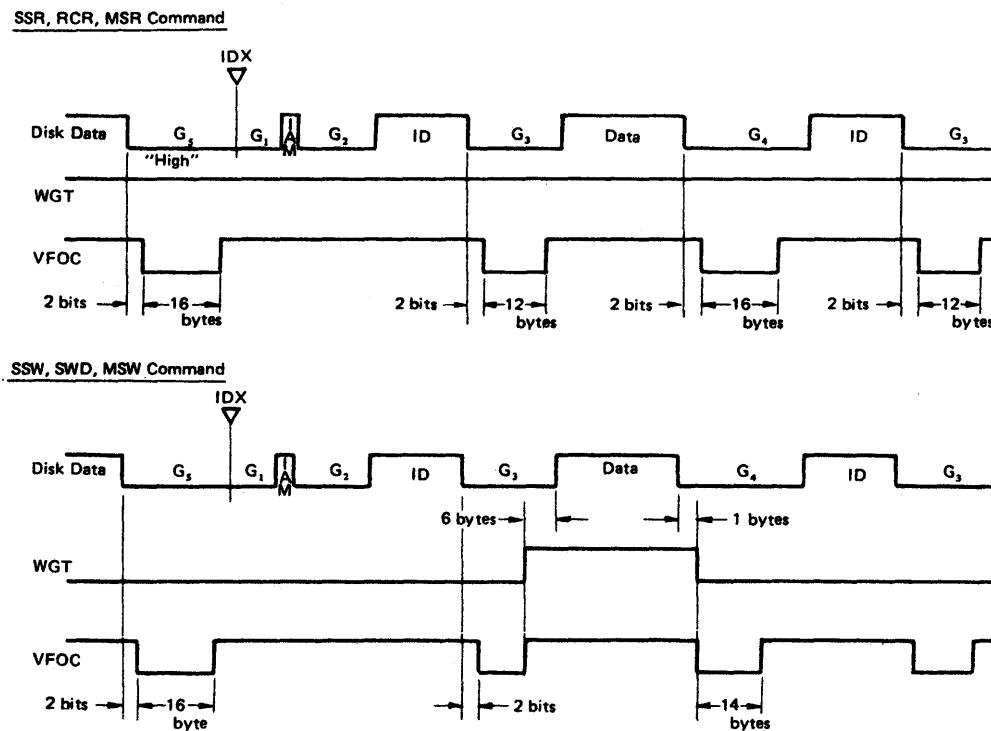
Figure 15 WDT Output Timing

- FORMAT

The format used by the HD6843SP, shown in Figure 18, compatible with the soft sector format of the IBM3740.

- MACRO COMMAND SET

The macro command set shown in Table 4 is discussed in the following paragraphs.



In FFW Command, VFOC becomes "High" when WGT is at "High" level.  
In FFR Command, VFOC remains "High".

Figure 16 Variable Frequency Oscillator Control Waveform  
(Relation Between WGT and VFOC)

## HD6843S, HD68A43S

SSW, SWD and MSW commands (Single Sector Write, Single Sector Write with Delete Data Mark, and Multi-Sector Write)

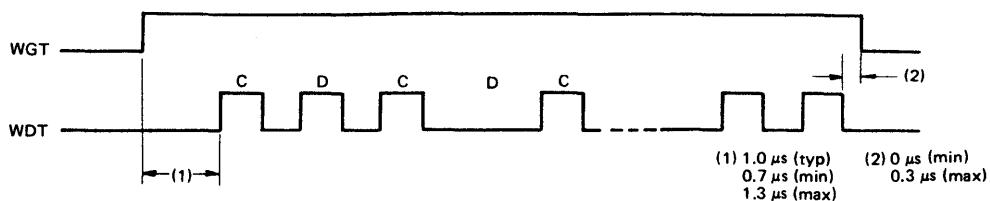


Figure 17 Write Data versus Write Gate Timing

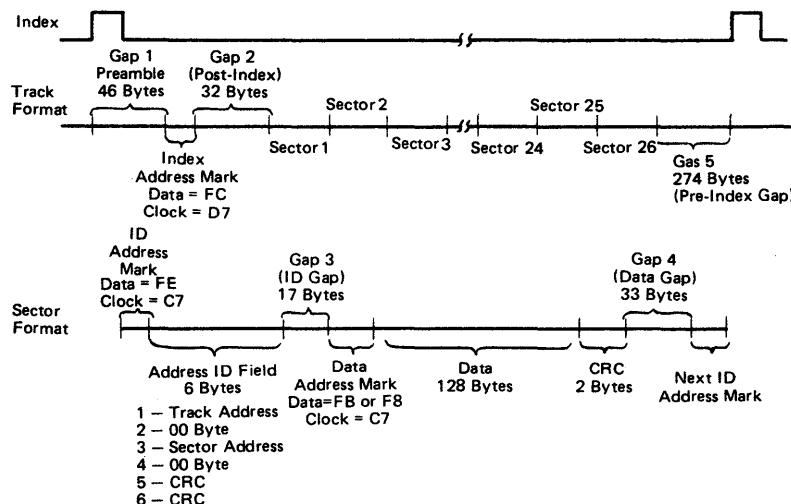


Figure 18 Soft Sector Format

Table 4 Macro Command Set

	Macro Command		CMR Bits				Hex Code
			Bit 3	Bit 2	Bit 1	Bit 0	
1	STZ	Seek Track Zero	0	0	1	0	2
2	SEK	Seek	0	0	1	1	3
3	SSR	Single Sector Read	0	1	0	0	4
4	SSW	Single Sector Write	0	1	0	1	5
5	RCR	Read CRC	0	1	1	0	6
6	SWD	Single Sector Write with Delete Data Mark	0	1	1	1	7
7	MSW	Multi Sector Write	1	1	0	1	D
8	MSR	Multi Sector Read	1	1	0	0	C
9	FFW	Free Format Write	1	0	1	1	B
10	FFR	Free Format Read	1	0	1	0	A

### • Seek Track Zero (STZ)

The STZ command causes the R/W head to be released from the surface of the disk (HLD is reset) and positioned above track 00. The FDC issues step pulses on the STP output until the TRZ input becomes a "High" level or until 82 pulses have been sent to the drive. When the TRZ input becomes "High", the step pulses are inhibited on the STP output but the FDC remains busy until all 82 have been generated internally.

If the TRZ input remains "Low" (logic "0") after all 82 pulses have been generated, the seek error flag (STRB bit 4) is set.

After all 82 pulses have been generated, the head is loaded (HLD becomes a "High"). After the settling time specified in the SUR has expired, the Seek Command End flag is set (ISR bit 1), Busy STRA7 is reset, CTAR and GCR are cleared. The head remains in contact with the disk. A command such as RCR (Read CRC) may be issued following a STZ if the head must be released.

### • Seek (SEK)

The SEK command is used to position the R/W head over the track on which a Read/Write operation is to be performed. The contents of the GCR are taken as the destination address and the content of the CTAR is the source address; therefore, the number of pulses (N) on the STP output are given by:

$$N = |(CTAR) - (GCR)|$$

HDR is a "High" for  $(GCR) > (CTAR)$  otherwise it is a "Low".

When a SEK command is issued, Busy is set, the head is raised from the disk, HDR is set as described above, and N number of pulses appear on the STP output. After the last step pulse is used, the head is placed in contact with the disk. Once the head settling time has expired, the Seek Command End flag (ISR bit 1) is set, Busy is reset, and the contents of the GCR are transferred to the CTAR.

### ■ SINGLE SECTOR READ/WRITE COMMANDS

The single sector Read/Write commands (SSR, RCR, SSW, and SWD) are used to Read/Write data from a single 128 byte sector on the disk. As shown in Figure 19 these types of instructions can be divided into two sections. The first section, which is common to all instructions, is the address search operation, while the second section is unique to the requirements of each instruction.

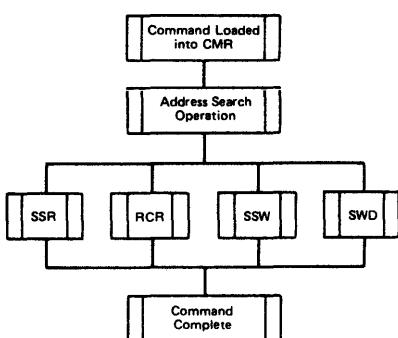


Figure 19 Basic Single Sector Command Flow Chart

### • Address Search Operation

The flow chart of Figure 20 shows the operation of the address search operation.

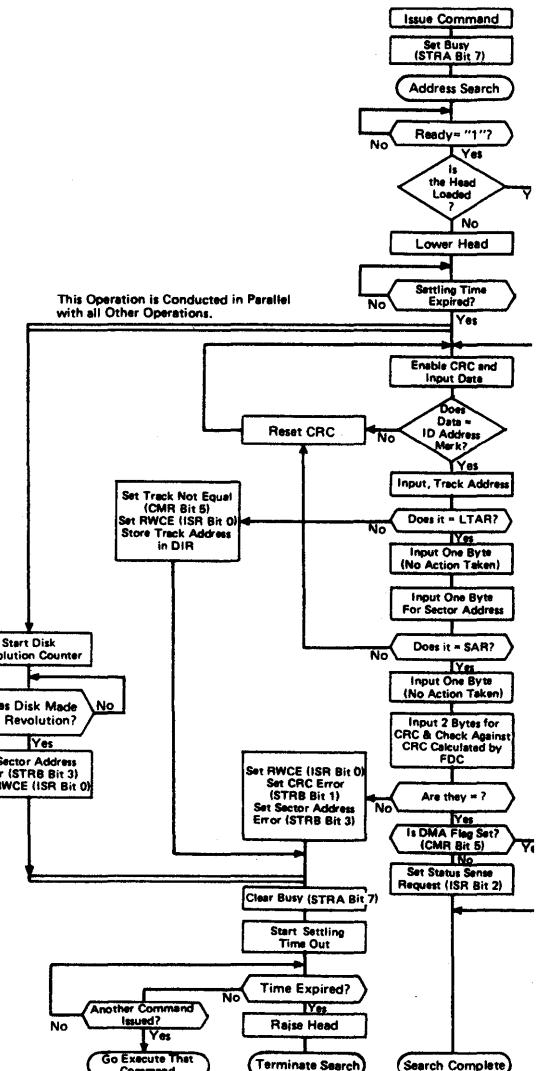


Figure 20 Operational Flow of the Address Search Sequence

### • Single Sector Read (SSR)

The single sector read command follows the address search procedure as defined in the previous flowchart. If the search is successful, status sense request is set and the operation continues as described by the flowchart of Figure 21.

### • Read CRC (RCR)

The RCR command is used to verify that correct data was written on a disk. The operation is the same as for the SSR.

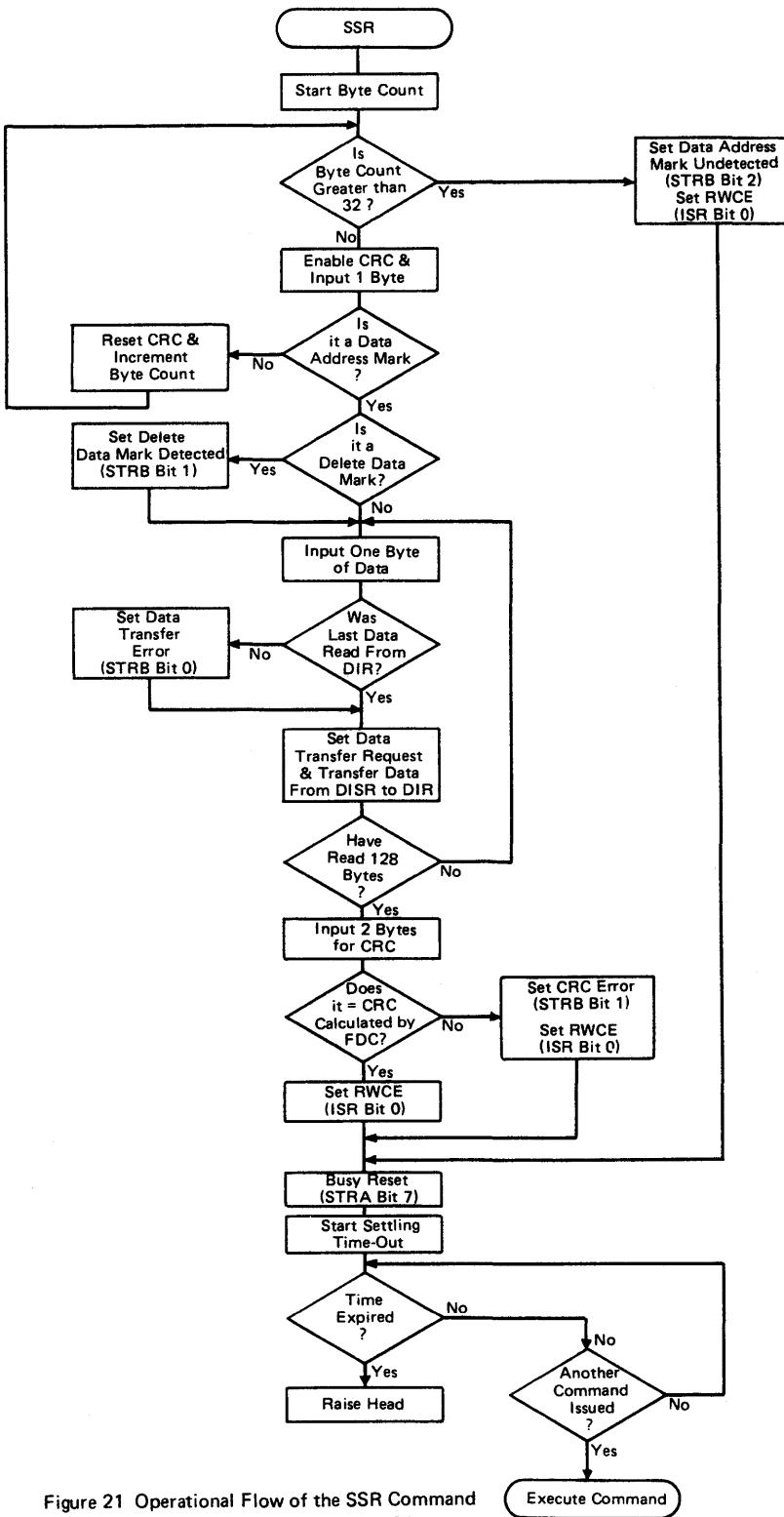


Figure 21 Operational Flow of the SSR Command

command with the exception that the data transfer request (STRA bit 0) is not set. The Status Sense Request interrupt can be disabled by using the DMA flag of CMR.

#### ● Single Sector Write (SSW)

Single sector write is used to write 128 bytes of data on the disk. After the command is issued, the address search is performed. The remainder of the instruction's operation is shown in Figure 22.

#### ● Single Sector Write with Delete Data Mark (SWD)

The operation flow of SWD is exactly like that of SSW. For SWD, the data pattern of the Data Address Mark becomes F instead of FB. The clock pattern remains C7.

#### ● Multi-Sector Commands (MSR/MSW)

MSR is used for sequential reading of one or more sectors. If S sectors are to be read, S - 1 must be written into the GC before the command is issued.

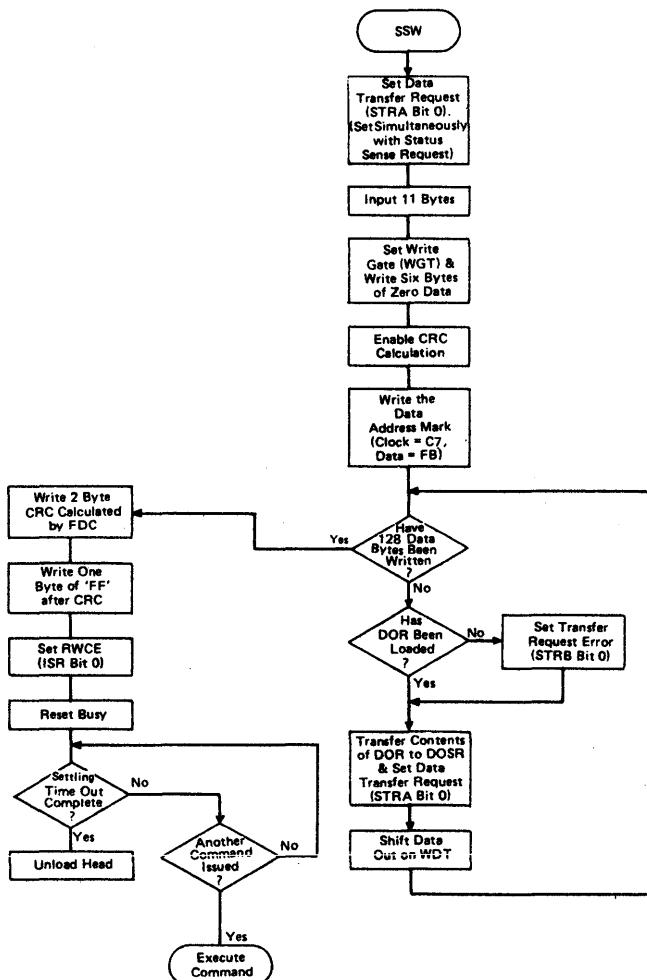


Figure 22 Operational Flow of the SSW Command

The basic operation for the MSR and MSW is the same as that for the SSR and SSW respectively. The basic operation begins with an address search operation, which is followed by a single sector read or write operation. This completes the operation on the first sector. The SAR is incremented, the GCR is decremented, and if no overflow is detected from the GCR (i.e., GCR become negative) the sequence is repeated until S number of sectors are read or written.

The completion of an MSR or MSW is like that of an SSR or SSW command. First RWCE is set and Busy is reset, after the settling time has expired, the head is released.

If a delete data mark is detected during an MSR command, STRA bit 1 (Delete Data Mark Detected) remains set throughout the commands operation.

When a multi-sector instruction is issued, the sum of the SAR and GCR must be less than 27. If  $SAR + GCR > 26$ , an address error (STRB bit 3 set) will occur after the contents of SAR becomes greater than 26.

#### • Free Format Write (FFW)

The FFW has two modes of operation which are selected by FWF (Free Format Write Flag) which is data bit 4 of the CMR.

When FWF = "0", the data bits of the DOR are written directly to the disk without first writing the preamble, address mark, etc. The contents of the DOR are FM modulated with a clock pattern of all ones.

If FWF = "1" the odd bits of the DOR are used as clock bits and even bits are used for data bits. In this mode, the DOSR clock is twice a normal write operation and one byte of DOR is one nibble (four bits of data) on the disk.

The two modes of the FFW command allow formatting a disk with either the IBM3470 format or a user defined format.

After the FFW command is loaded into the CMR, WGT becomes a "High" level, the contents of DOR are transferred to the DOSR, data transfer request (STRA bit 0) is set, and the serial bit pattern is shifted out on the WDT line. Therefore, DOR must be loaded before the FFW command is issued. Data from the DOR is continually transferred to the DOSR and shifted out on WDT until the CMR has been written with an all zero pattern. When CMR becomes zero, WGT becomes a "Low" level, but RWCE is not set and the R/W head is left in contact with the disk.

#### • Free Format Read (FFR)

FFR is used to input all data (including Address marks) from a disk. Once the FFR command is set into the CMR, the head is loaded and after the settling time has expired the serial data from the FDC is brought into the DISR. After 8 bits have accumulated, it is transferred to the DIR and Data Transfer Request (STRA bit 0) is set.

This operation continues until a zero pattern is stored in the CMR, terminating the FFR command. As in the case of the FFW command, RWCE is not set and the head remains in contact with the disk.

The first data that enters the DISR is not necessarily the first bit of a data word since the head may be lowered at any place on the disk. To prevent the FDC from remaining unsynchronized to the data, the FFR command will synchronize to an ID address mark (FE) or a Data Address mark (FB or F8) or an Index Address Mark (FC).

## ■ REGISTER DEFINITIONS

### • Data Output Register (DOR); Hex address 0, write only

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
8 Bits of Data Used for a Disk Write Operation							

When one of the four write macro commands (SSW, SWD, MSW, and FFW) is executed, the information contained in the DOR is loaded into the DOSR, and is shifted out on the WDT line using a double frequency (FM) format.

### • Data Input Register (DIR); Hex address 0, read only

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
8 Bits of Data Used for a Disk Read Operation							

One of the three read macro commands (SSR, MSR, FFR) executed, will cause the information on the RDT input to be clocked into the DISR. When 8 clock pulses have occurred, the 8 bits of information in the DISR are transferred to the DIR where it can be read by the bus interface.

### • Current Track Address (CTAR); Hex address 1, read/write

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Track Address of Current Head position							

The address of the track over which the R/W head is currently positioned is contained in the CTAR. At the end of a SEK command, the contents of the GCR are transferred to the CTAR. CTAR is cleared at the completion of a STZ command. CTAR is a read/write register so that the head position can be updated when several drives are connected to one FDC. Bit 7 is read as a "0".

### • Command Register (CMR); Hex address 2, write only

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3*	Bit 2*	Bit 1*	Bit 0*
Function Interrupt Mask	ISR3 Interrupt Mask	DMA Flag	FWF	Macro Command			

\*Bit 0 ~ 3 are cleared by RES.

The commands that control the FDC are loaded into the lower four bits of the CMR. Information that controls the data transfer mode and interrupt conditions are loaded into bits four through seven.

#### Bit 0~Bit 3: Macro Command

The Macro Command to be executed by the FDC is written to bits 0~3.

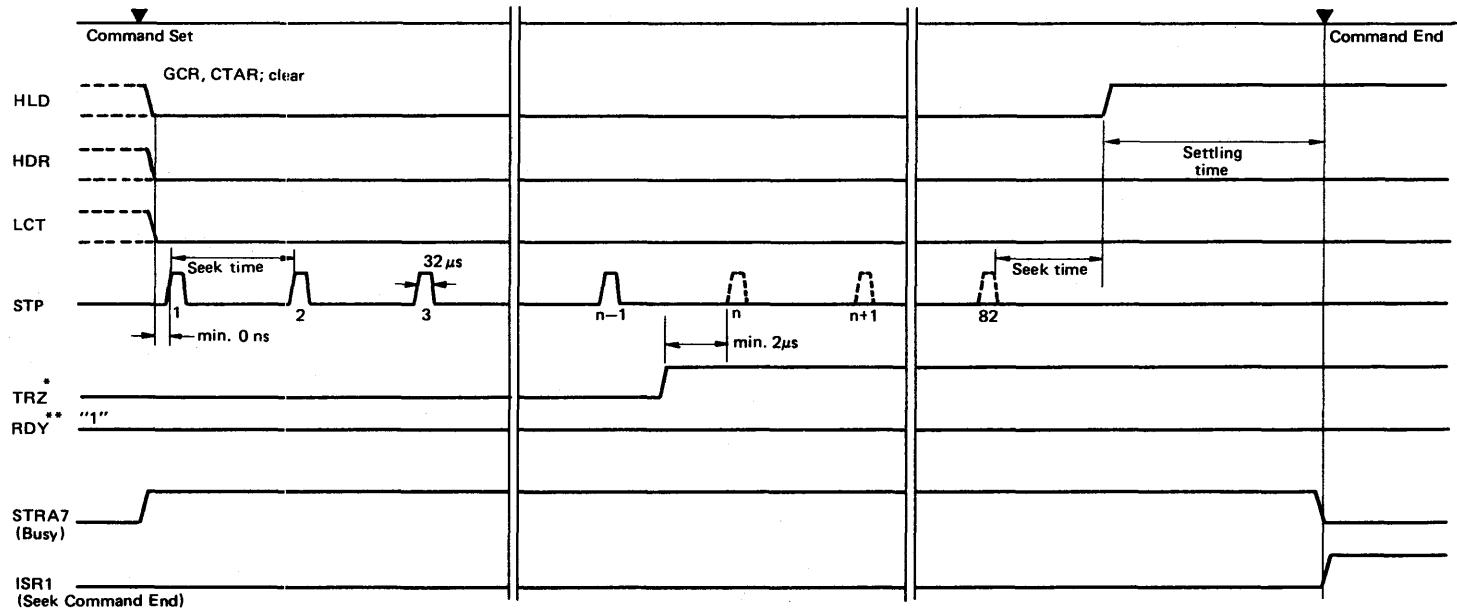
#### Bit 4: Free Format Write Flag (FWF)

If a Free Format Write command is issued, the state of bit 4 of the CMR determines what clock source will be used. The FWF is defined in the FFW (Free Format Write) command explanation.

#### Bit 5: DMA Flag

If bit 5 is a "1" the FDC is in the DMA mode. Bit 5 being a "1" inhibits setting of Status Sense Request (ISR bit 2) thereby preventing its associated interrupt. A logic "1" DMA flag also enables the TxRQ output allowing it to request DMA transfers when the Data Transfer Request flag (STRA bit 0) is set.

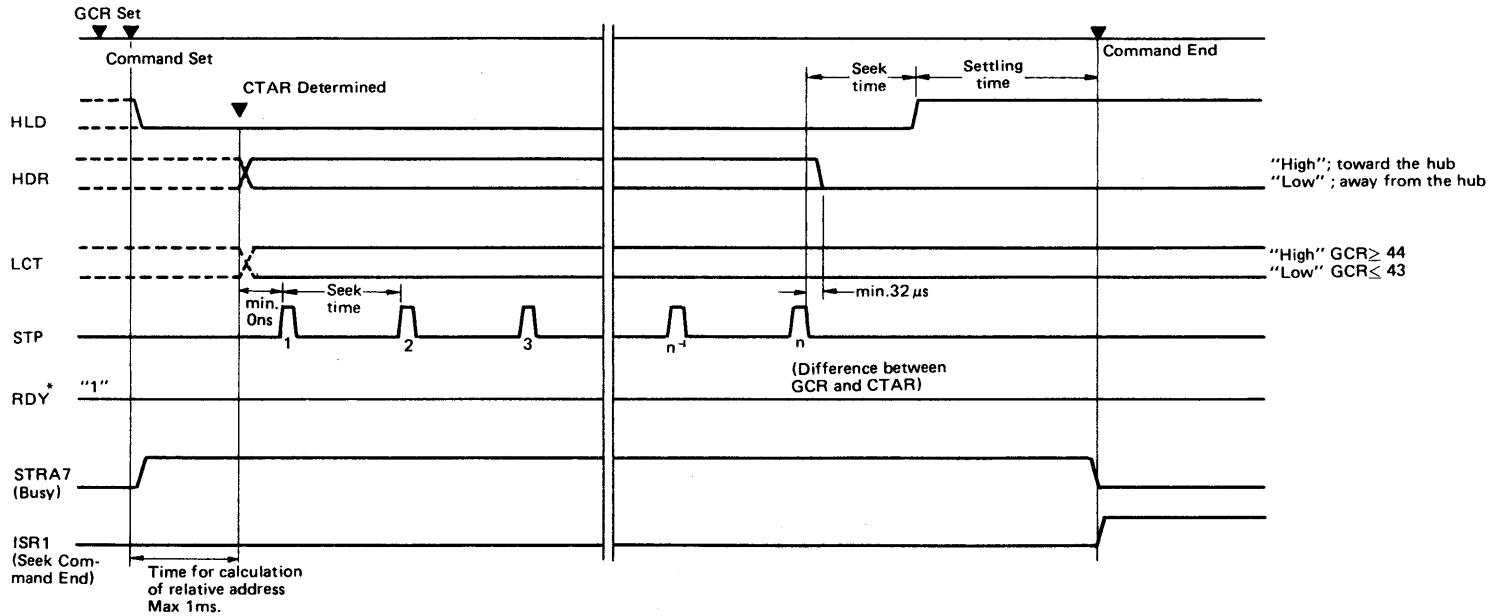
A logic "0" DMA flag indicates the program controlled I/O (PC I/O) mode.



\* STP output is masked when TRZ becomes "High". But if TRZ falls to "Low" again before 82 pulse outputs are all provided, STP output become available again from that time point.

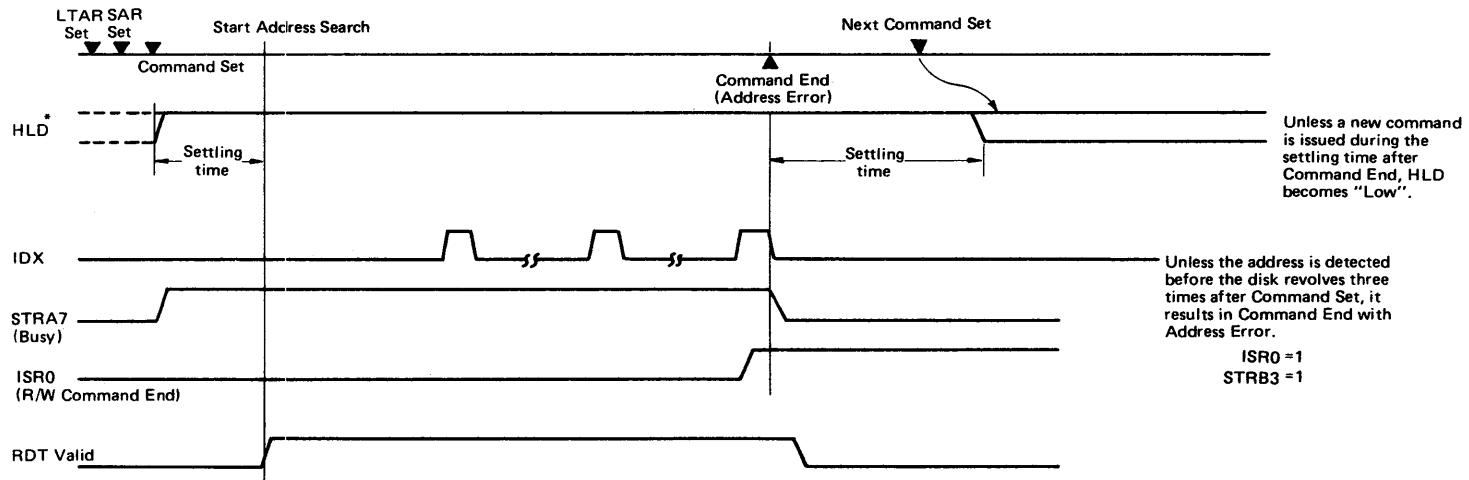
\*\* When RDY is "Low" with Command Set, the execution is postponed until RDY becomes "High".

Figure 23 Timing Sequence of STZ Command



\* When RDY is "Low" with Command Set, the execution is postponed until RDY becomes "High".

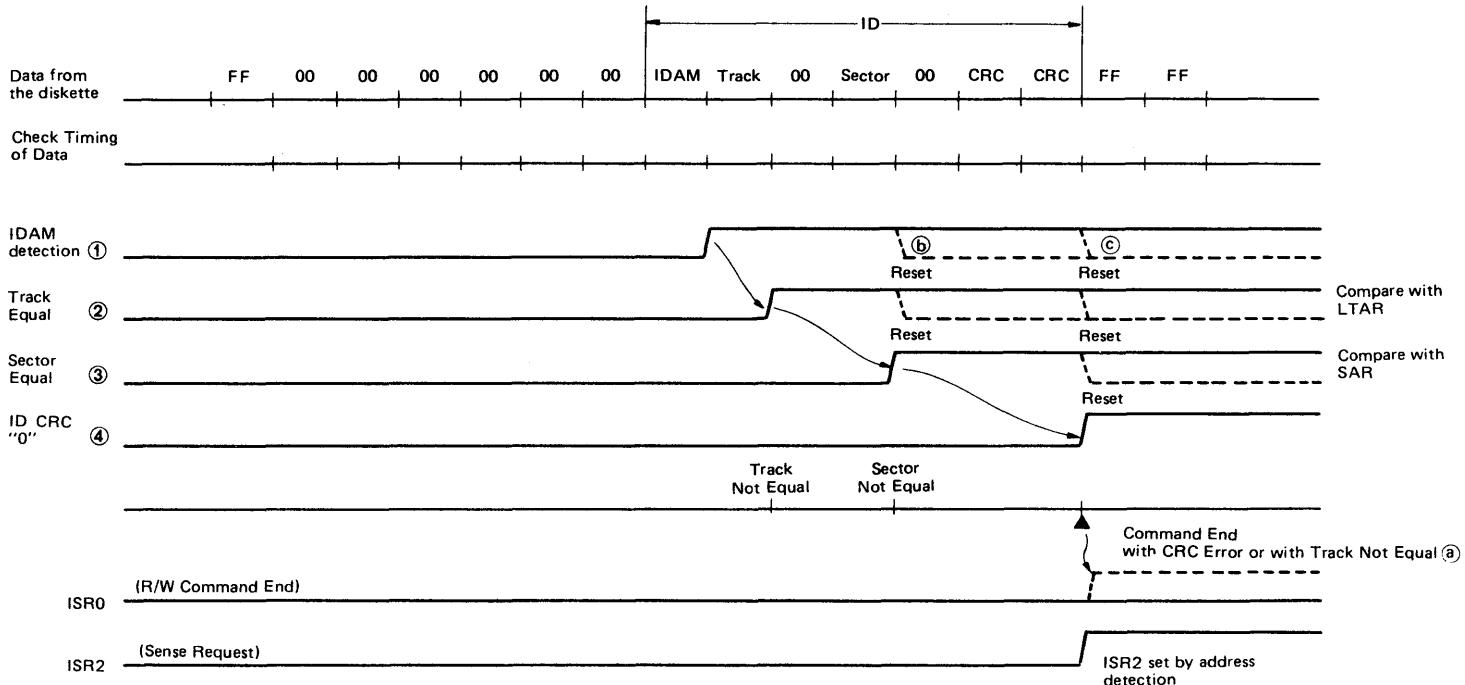
Figure 24 Timing Sequence of SEK Command



\* If HLD has already been "High" when the command is set, the FDC starts the address search immediately.

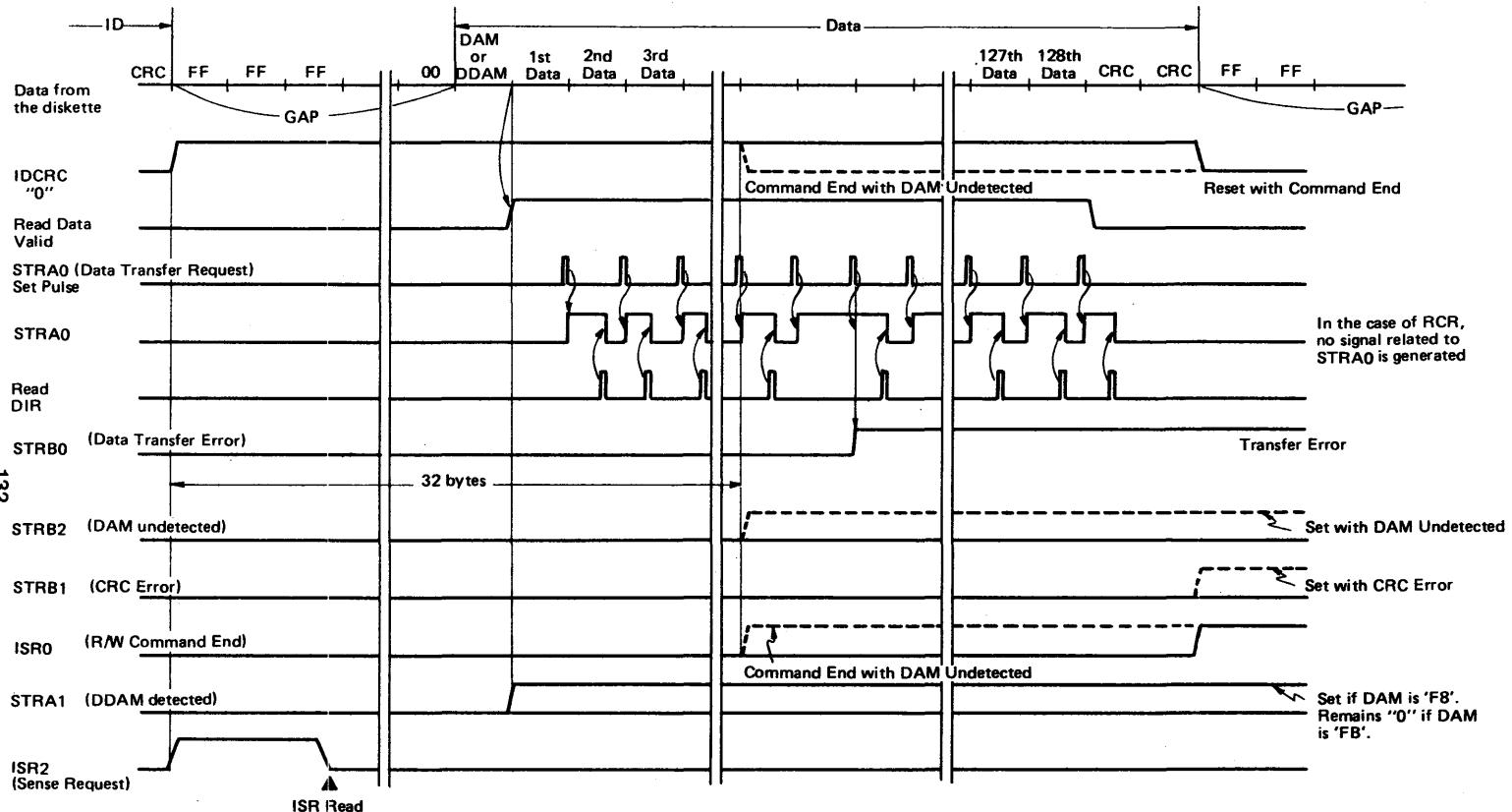
When RDY is "Low" with Command Set, the FDC waits for the execution until RDY becomes "High".

Figure 25 Timing Sequence of SSR, SSW, RCR, SWD, MSR, MSW Command  
(Relation with HLD and IDX)



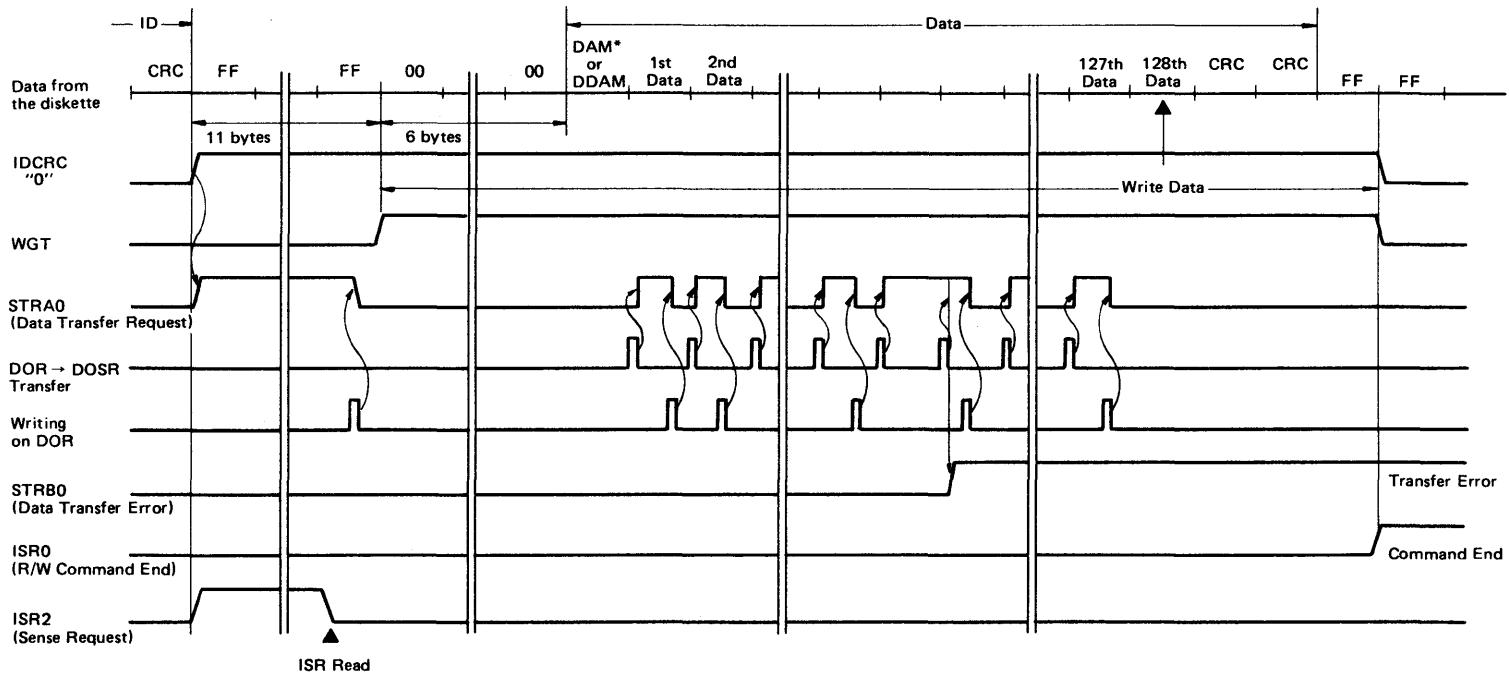
- (a) ; In the case of Track Not Equal, ② is not set and if CRC equals to the one calculated by FDC, STRA5 is set.  
 (b) ; In the case of Sector Not Equal, ③ is not set and ① & ② are reset to search the next IDAM.  
 (c) ; In the case of CRC Error, ④ is not set and ①, ② & ③ are reset. (ISR0: Set, STRB1: Set, STRB3: Set)  
 When ①, ②, ③, & ④ are all set, ISR2 is Set. These four signals are reset with Command End.  
 When ④ is "1", go to the data transfer routine.

Figure 26 Internal Timing Sequence of Address Search Routine



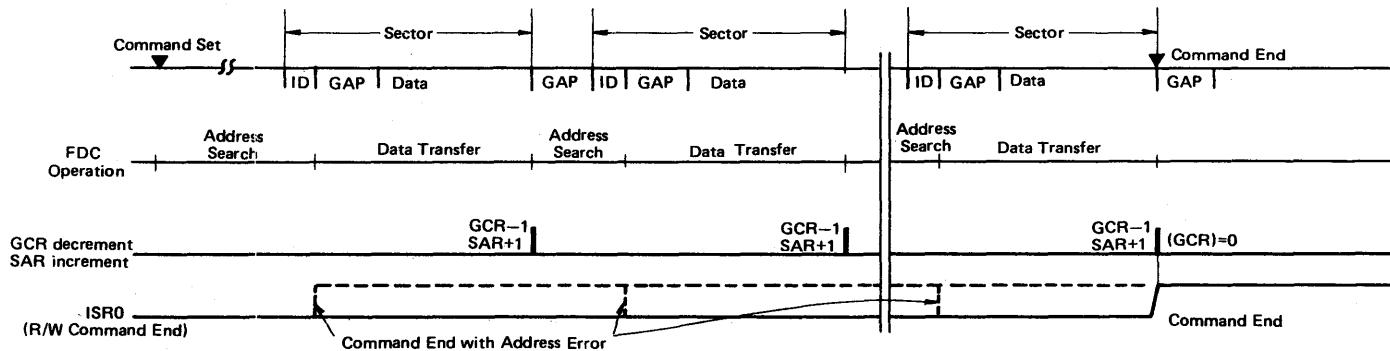
Unless DAM(FB) or DDAM(F8) is detected within 32 bytes after ID field has been detected, STRB2 is set to end the command.

Figure 27 Data Transfer Timing of SSR, RCR Command



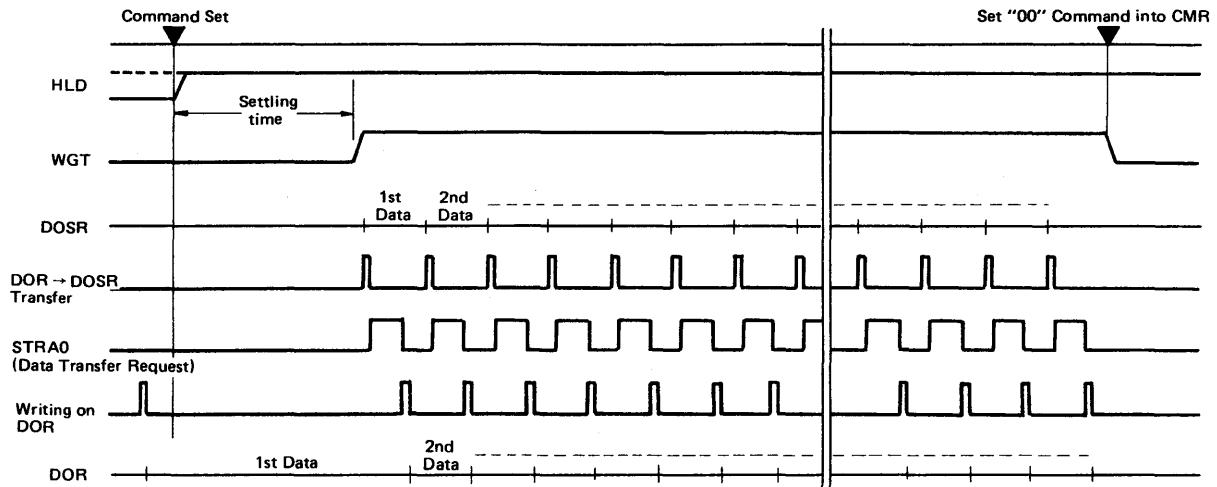
\* As Data Address Mark, SSW command writes 'FB' and SWD command writes 'F8'.

Figure 28 Data Transfer Timing of SSW, SWD Command



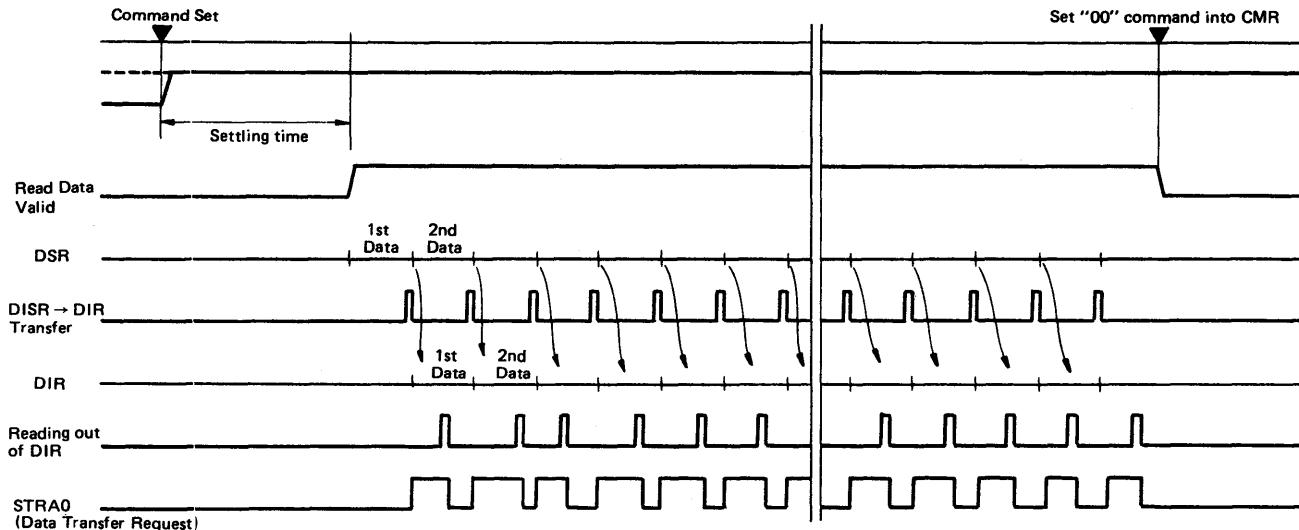
Address Search and Data Transfer in each sector is the same as those of SSR or SSW command.  
When Address Error occurs, it results in Command End. If an error relating to Data Transfer occurs,  
Error flag is set. But the command continues to be executed to shift into the next sector.

Figure 29 Timing Sequence of MSR, MSW Command



- The first one-byte data must be set into DOR before Command Set.
- If HLD has already been "High" when the command is set, WGT becomes "High" immediately.
- When '00' command is set into CMR, an interrupt of Command End is not generated.

Figure 30 Timing Sequence of FFW Command



If HLD has already been "High" when the command is set, Read operation starts immediately without waiting for the settling time.  
When "00" command is set into CMR, an interrupt of Command End is not generated.

Figure 31 Timing Sequence of FFR Command

**Bit 6: ISR3 Interrupt Mask**

CMR bit 6 (ISR3 Mask) is used to control the operation of ISR bit 3. A logic "1" in CMR bit 6 inhibits output of STRB-OR-Interrupt signal to  $\overline{IRQ}$ . If CMR bit 6 (ISR3 Mask) and CMR bit 7 are "0" STRB-OR-Interrupt signal will be output to  $\overline{IRQ}$ .

**Bit 7: Function Interrupt Mask**

When CMR bit 7 is a logic "1" all interrupts are inhibited.

Table 5

Causes of Interrupt	Command Register Masks That Affect Interrupts		
	CMR7 (Function Interrupt Mask)	CMR6 (ISR3 Mask)	CMR5 (DMA Flag)
ISR0 (Read write Command End)	M	X	X
ISR1 (Seek Command End)	M	X	X
ISR2 (Status Sense Request)	M	X	M
ISR3 (STRB-OR-Interrupt)	M	M	X

X = No effect

M = Bits that are used as masks

**• Interrupt Status Register (ISR); Hex address 2, read only**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2 *	Bit 1 *	Bit 0 *
Not Used (Read as "0")	STRB-OR	Status Sense Request	Seek Command End	Read Write Command End			

\* Cleared by  $\overline{RES}$

**Bit 0: Read Write Command End (RWCE)**

When an SSR, RCR, SSW, SWD, MSR or MSW Macro Command has completed execution, bit 0 becomes set (logic "1"). If the function interrupts are enabled (bit 7 of CMR is a logic "0"), the conclusion of a Macro Command's execution will cause an interrupt.

**Bit 1: Seek Command End (SCE)**

Seek Command End is set on SEK and STZ commands to indicate the head has been loaded and the settling time specified in SUR has expired. Since RWCE is not set for the SEK or STZ command, SCE can be used as an interrupt to signify the SEK or STZ command has finished. SCE is not set for any of the R/W commands.

**Bit 2: Status Sense Request**

For an SSR, SSW, SWD, MSR, or MSW Command, Status Sense Request indicates that the specified address ID field has been detected and verified by a CRC check. This is used as an early indication that data transfers will occur after 18 more byte

times. For MSR and MSW commands, it is set for each sector.

In the PC I/O mode, an interrupt occurs when Status Sense Request becomes a logic "1". In the DMA mode, (DMA flag of CMR is set) Status Sense Request is unchanged and does not generate an interrupt when the address ID field has been verified.

**Bit 3: STRB-OR**

STRB-OR is an "OR" of all of the bits of Status Register B.

$$\text{STRB-OR} = \text{STRB0} + \text{STRB1} + \text{STRB2} + \text{STRB3} +$$

$$\text{STRB4} + \text{STRB5} + \text{STRB6} + \text{STRB7}$$

$$\text{STRB-OR-Interrupt} = \text{STRB1} + \text{STRB2} + \text{STRB3} +$$

$$\text{STRB4} + \text{STRB5} + \text{STRB6} + \text{STRB7}$$

STRB-OR-Interrupt signal causes  $\overline{IRQ}$ . STRB-OR is read by Read ISR. STRB0 (Data Transfer Error) sets ISR Bit 3 but does not cause Interrupt.

ISR0, ISR1, and ISR2 are cleared when the Interrupt Status Register is read, but ISR3 is cleared only after Status Register B has been read except when FI input is "High".

**• Set-Up Register (SUR); Hex address 3, write only**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Track to Track Seek Time						Head Settling Time	

The SUR is not affected by a reset operation; therefore, once it is initialized, the information remains until power is removed from the FDC.

**Bit 0 ~ Bit 3: Head Settling Time**

The head settling time is used to generate a delay after the head is placed in contact with the disk. This allows the head to stop bouncing before any operations are performed. The delay is programmed by bits 0~3 and is specified by the equation:

$$\text{Delay} = \frac{4096}{f} \cdot B$$

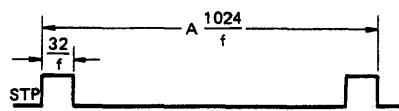
B = Number contained in bits 0~3 of SUR

f = Frequency of CLK input

For IBM3740 compatibility f = 1 MHz and the timing range is 4.096 ms for a "0001" to 61.44 ms for a "1111". A "0000" code prevents Settling Time complete from being set and the FDC must be Reset.

**Bit 4 ~ Bit 7: Track to Track Seek Time**

The frequency of STP is determined by bit 4~bit 7 of SUR as shown below.



A = Number specified in bits 4~7 of SUR.

f = Frequency of CLK input.

For IBM compatible operation, f is 1 MHz. This results in an STP pulse width of  $32 \mu s$  and an STP interval of 1.024 ms for a "0001" to 15.36 ms for a "1111".

- Status Register A (STRA); Hex address 3, read only**

Bit 7*	Bit 6	Bit 5*	Bit 4	Bit 3	Bit 2	Bit 1*	Bit 0*
Busy	Index	Track Not Equal	Write Protect	Track Zero	Drive Ready	Delete Data Mark Detected	Data Transfer Request

\* Cleared by RES

#### Bit 0: Data Transfer Request

For a write operation (SSW, SWD, MSW, FFW) the transfer request bit indicates that the DOR is ready to accept the next data word to be written on the disk. If data is not written into the DOR before the last data bit in the DOSR is shifted out to the WDT line; the data transfer error bit (bit 0 of STRB) will be set. After a write command has been issued, the first transfer request occurs simultaneously with the Status Sense Request. For a write operation, transfer request is reset after the DOR has been written from the data bus.

During a read operation (SSR, MSR, FFR) the transfer request bit signifies data from the DISR has been transferred to the DIR. The DIR must be read before the DISR is full again or the data transfer error bit (bit 0 of STRB) will be set. For read operations, transfer request is reset by a read of the DIR.

#### Bit 1: Delete Data Mark Detected

A Single Sector Read operation that detects a delete data code (F8) instead of a general data code (FB) as a Data Address Mark will set the Delete Data Mark Detected bit. For the MSR command, bit 1 is set the first time an "F8" code is found and remains set throughout the execution of the command. Bit 1 is reset whenever an SSR, SSW, SWD, MSR, MSW, or RCR command is issued.

#### Bit 2: Drive Ready

The Drive Ready bit indicates the state of the Ready input from the floppy disk drive. If a command is issued with Ready at logic "0", its execution will be inhibited until Ready becomes a logic "1". If ready becomes a "0" during the execution of a command the Hard Error Flag (STRB bit 7) is set.

#### Bit 3: Track Zero

The state of the Track Zero input from the floppy disk drive is reflected in this bit of STRA. A logic "1" on the Track Zero input inhibits step pulses during an STZ command.

#### Bit 4: Write Protect

The Write Protect input from the floppy disk drive is reflected by bit 4 of STRA. A "High" level (logic "1") on the WPT input during the execution of any write command results in a write error (bit 6 of STRB set).

#### Bit 5: Track Not Equal

If the track address read from the address ID field does not coincide with the address in the LTAR inspite of CRC matching the one calculated by FDC, the Track Not Equal bit is set. Track Not Equal applies to all non-free format read/write commands, and is reset after a non-free format read/write command is issued.

#### Bit 6: Index

The state of the index input appears in bit 6 of STRA. The index input is used to count the number of disk revolutions while the FDC is looking for the address ID field (see operation

of STRB bit 3) during the address search phase of a non-free format read/write command.

#### Bit 7: Busy

When Busy is a logic "1", the FDC is executing a command and no new commands can be issued. Busy should be confirmed to be "0" before reading ISR or, STRB as well as issuing a command.

- Sector Address Register (SAR); Hex address 4, write only**

Bit 7	Bit 6	Bit 5	Bit 4*	Bit 3*	Bit 2*	Bit 1*	Bit 0*
Not Used		5 Bit Sector Address					

\* Cleared by RES

Before a data transfer macro command (SSW, SWD, SSR, RCR, MSW, MSR) is issued, the address of the sector on which the operation is to be performed must be written into the SAR. The address in the sector address byte of an Address ID field of the disk is compared with the contents of the SAR. During an MSW or MSR command, the SAR is incremented after each sector is read or written. When execution is complete, the SAR contains the address of the last sector on which an operation was performed plus one.

- Status Register B (STRB); Hex address 4, read only**

Bit 7*	Bit 6*	Bit 5	Bit 4*	Bit 3*	Bit 2*	Bit 1*	Bit 0*
Hard Error	Write Error	File Inoperable	Seek Error	Sector Address Undetected	Data Mark Undetected	CRC Error	Data Transfer Error

\* Cleared by RES

The bits of the STRB represent possible error conditions that may occur during execution of macro commands. Whenever STRB is reset, ISR bit 3 is also reset.

#### Bit 0: Data Transfer Error

Data Transfer Error indicates an underflow or overflow of data. If a Write operation is being performed, it signifies that data was not presented to the DOR before the DOSR became empty. In this case, the current contents of the DOR are transferred to the DOSR and the write operation continues. The data transfer error remains set until STRB is read, and the data transfer request remains set until data is written into the DOR. The operation of the CRC is unchanged.

For read commands, a data transfer error indicates that data in the DIR was not read before the next data word from the disk was transferred to the DIR. The read operation continues until sufficient data has been read from the disk to satisfy the requirements of the command (128 bytes for SSR). The error indication remains set until STRB is read, and the transfer request remains set until data is read from the DIR.

#### Bit 1: CRC Error

A CRC error occurs when the CRC read from the disk does not match that calculated by the FDC on the data it reads from the disk. A CRC error can occur in two different situations; checking the address ID field, checking the data field.

If the CRC error occurs during the check of an address ID field, Sector Address Undetected (STRB bit 3) will also be indicated (see Table 6). A CRC error of a data field is indicated by a CRC Error and no Sector Address Undetected.

**Bit 2: Data Mark Undetected**

If a valid data mark is not detected in the data block of a sector, it is indicated by a Data Mark Undetected error.

**Bit 3: Sector Address Undetected**

The Sector Address Undetected bit can be set on two conditions; not finding the sector address and a CRC error on an address ID field.

If the disk makes three revolutions during an address search operation and the sector address specified in the sector address register is not found in any of the address ID fields, a Sector Address Undetected condition is indicated.

A CRC error that occurs on an address ID field will set bit 3 also. Table 6 shows how bits 1 and 3 are related.

**Table 6 Relationship of CRC Error and Sector Address Undetected**

CRC Error (STRB1)	Sector Address Undetected (STRB3)	Condition
0	0	No Error
0	1	Sector Address not Detected
1	0	CRC Error on a Data Field
1	1	CRC Error on Address ID Field

**Bit 4: Seek Error**

An STZ (Seek Track Zero) command that never receives a track zero indication on the track zero input will result in a Seek Error (see description of STZ command).

**Bit 5: File Inoperable**

The state of the File Inoperable input appears in bit 5. If the File Inoperable input is a "High" level, a pulse of width equals to Enable pulse width  $PW_{EL}$  is issued on the FIR output when STRB is read. FI is not latched but the input is gated to the bus when STRB is read.

**Bit 6: Write Error**

If the WPT input becomes a "High" level (logic "1") during the execution of a write command the Write Error bit is set.

**Bit 7: Hard Error**

If the Ready input becomes a "Low" level during the operation of a command (Busy is set), a Hard Error indication will result.

**• General Count Register (GCR); Hex address 5, write only**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Not Used	7 Bit Count for Track Number on SEK Command and Sector Count for MSR or MSW Command						

The GCR contains the destination track address for the R/W head on an SEK Macro Command. The contents of the GCR are transferred to the CTAR at the end of the SEK Command. For multi-sector read or write operations (MSR, MSW), the GCR contains the number of sectors to be read minus one. During the MSR or MSW execution the GCR is decremented after each sector is read or written.

**• CRC Control Register (CCR); Hex address 6, write only**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Not Used						Shift CRC	CRC Enable

The CCR information is used only in the free format commands; for all other commands this register is masked and has no function.

**Bit 0: CRC Enable**

During an FFW command, CRC Enable is set by software and CRC generation takes effect on the next transfer of data from DOR to DOSR (see figure 32). The CRC generation continues until Shift CRC (CCR bit 1) is set.

For an FFR command, CRC Enable is set by software and CRC generation takes effect on the next data read from DIR. The calculation continues for all data bytes read from DIR until CRC Enable is reset. The bytes read previous to resetting CRC Enable are considered the CRC information bytes and the CRC check is made against them.

**Bit 1: Shift CRC**

Bit 1 is valid only for the FFW command. After setting, it takes effect on the next transfer of data from DOR to DOSR (see Figure 33). Setting Shift CRC terminates the CRC calculation and causes the CRC calculated on all the data written into DOR up to the setting of bit 1, to be shifted out the WDT output. The CRC calculation will not include any data written to DOR after Shift CRC is set.

**• LTAR (Logical Track Address); Hex address 7, write only**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Not Used	7 Bit Logical Track Address						

When a read or write macro command (SSW, SWD, SSR, RCR, MSW, MSR) is issued, the address of the track on which the operation is to be performed must be written into the LTAR. The address in the track address byte of an Address ID field of the disk is compared with the contents of the LTAR. The contents of LTAR are not affected by the execution of any of the commands.

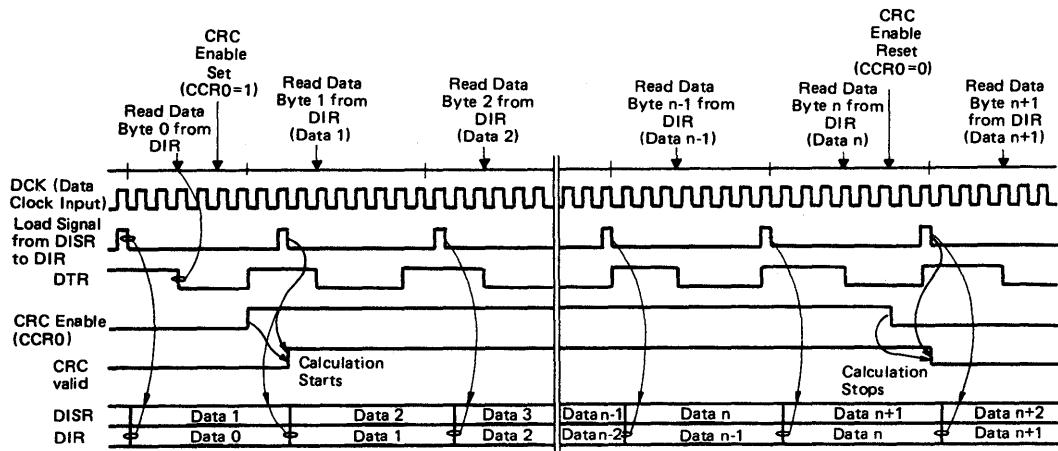


Figure 32 CCR Control Register Timing for an FFR Command (READ)

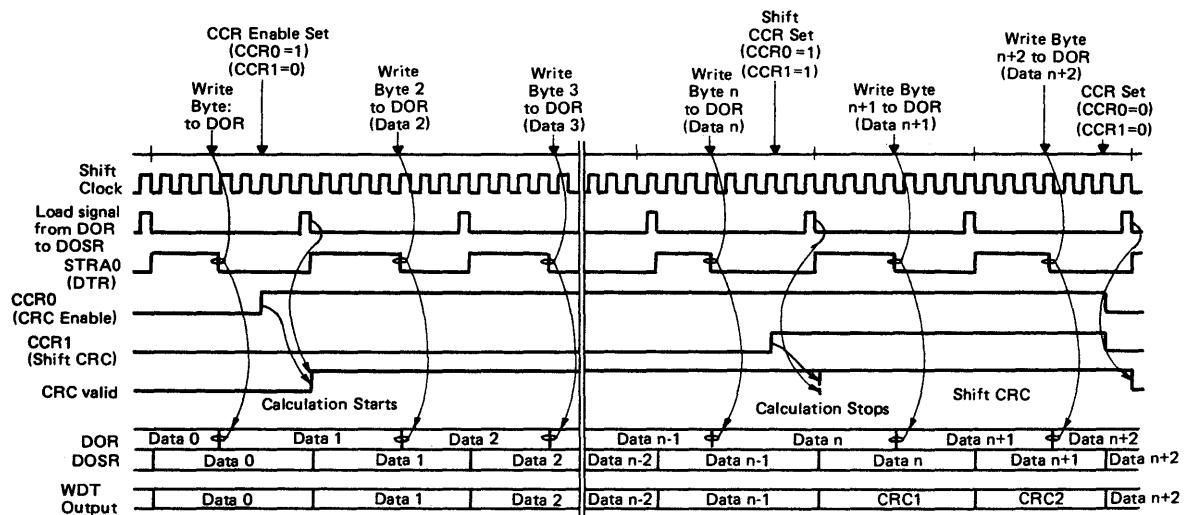


Figure 33 CCR Control Register Timing for an FFW Command (WRITE)

**HD6843S, HD68A43S**

**Table 7 Programming Reference Data**

Table 7 is a summary of the information in the data sheet and can be used as a reference when programming the HD6843S

Registers	Hex Address	R/W Mode	Data Bits											
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0				
DOOR	0	WO				8 Bits of Data Used for a Disk Write Operation								
DIR	0	RO	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0				
CTAR	1	R/W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Track Address of Current Head Position			
CMR	2	WO	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3 *	Bit 2 *	Bit 1 *	Bit 0 *				
			Function Interrupt Mask	ISR3 Interrupt Mask	DMA Flag	FWF	Macro Command							
ISR	2	RO	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2 *	Bit 1 *	Bit 0 *	Not Used	Status Sense Request	Seek Command End	Read Write Command End
SUR	3	WO	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Track to Track Seek Time			
			Bit 7 *	Bit 6	Bit 5 *	Bit 4	Bit 3	Bit 2	Bit 1 *	Bit 0 *				
STRA	3	RO	Busy	Index	Track Not Equal	Write Protect	Track Zero	Drive Ready	Delete Data Mark Detected	Data Transfer Request	Not Used	5 Bit Sector Address	Head Settling Time	
SAR	4	WO	Bit 7	Bit 6	Bit 5	Bit 4 *	Bit 3 *	Bit 2 *	Bit 1 *	Bit 0 *	Not Used			
			Bit 7 *	Bit 6 *	Bit 5	Bit 4 *	Bit 3 *	Bit 2 *	Bit 1 *	Bit 0 *	Hard Error	File Inoperable	Seek Error	
STRB	4	RO					Sector Address Undetected	Data Mark Undetected	CRC Error	Data Transfer Error				
GCR	5	WO	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Not Used	7 Bit Count for Track Number on SEK or Sector Count for MSR or MSW.		
CCR	6	WO	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Not Used	Shift CRC	CRC Enable	
LTAR	7	WO	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Not Used	7 Bit Logical Track Address		

RO – Read Only  
WO – Write Only  
R/W – Read/Write

\* Cleared by  $\overline{\text{RES}}$

## MACRO COMMANDS

Hex Code	Instruction	Hex Code	Instruction
2	STZ	A	FFR
3	SEK	B	FFW
4	SSR	C	MSR
5	SSW	D	MSW
6	RCR		
7	SWD		

Table 8 Error Condition, Command Execution, Interrupt, and Head Control

Error	Flag	Set Condition	Reset Condition	Command	Command Execution	Interrupt	Head Control
Track Not Equal	STRAS5	Track information of ID field is not equal to the content of LTAR.	Issuing of SSR, RCR, MSR, SSW, SWD or MSW Command	SSR, RCR, MSR SSW, SWD, MSW	The execution of a command is interrupted and R/W Command End (ISR0) is set.	Request (ISR0)	Unchanged**
Data Transfer Error	STRB0	Overrun or underflow during the data transfer	Reading of STRB	SSR, MSR, SSW, SWD, MSW, FFR FFW	Read/Write command continues to be executed.	No interrupt	Unchanged**
CRC Error	STRB1	CRC Error on ID field or Date field	Reading of STRB	SSR, RCR, MSR, SSW, SWD, MSW (FFR)	The execution of a command is interrupted and R/W Command End (ISR0) is set.	Request (ISR0, ISR3)	Unchanged**
Data Mark Undetected	STRB2	DAM or DDAM is undetected within 32 bytes after ID field has been detected.	Reading of STRB	SSR, RCR, MSR	The execution of a command is interrupted and R/W Command End (ISR0) is set.	Request (ISR0, ISR3)	Unchanged**
Sector Address Undetected	STRB3	(1) Sector Address of ID field is not equal to the content of SAR. (2) CRC Error on ID field	Reading of STRB after Busy (STRAT) is reset.	SSR, RCR, MSR SSW, SWD, MSW	The execution of a command is interrupted and R/W Command End (ISR0) is set.	Request (ISR0, ISR3)	Unchanged (Head remains loaded after settling time has expired.)
Seek Error	STRB4	TRZ signal remains "Low" level though eighty-two STP pulse outputs are provided in STZ command.	Reading of STRB	STZ	The execution of a command is interrupted and Seek Command End (ISR1) is set.	Request (ISR1, ISR3)	Unchanged
File Inoperable	STRB5	A "High" level input of FI terminal is reflected.	FI signal of the FDD is reset when "High" pulse output is provided by reading of STRB at FI="1".	All commands	The execution of a command is interrupted. If it is a Read/Write command, ISR0 is set. If it is a seek command, ISR1 is set.	Request (ISR0 or ISR1, ISR3)	Unload the head immediately (HLD="Low") Set WGT to "Low"
Write Error	STRB6	Write operation (WGT="High") is performed when the input of WPT terminal is "High" level.	Reading of STRB	SSW, SWD, MSW FFW	The execution of a command is interrupted and R/W Command End (ISR0) is set.	Request (ISR0, ISR3)	Unload the head immediately (HLD="Low" Set WGT to "Low")
Hard Error	STRB7	RDY input signal becomes "Low" level during the execution of a command (Busy="1").	Reading of STRB	All commands	The execution of a command is interrupted. If it is a Read/Write command, ISR0 is set. If it is a seek command, ISR1 is set.	Request (ISR0 or ISR1, ISR3)	Unload the head immediately (HLD="Low") Set WGT to "Low"
Not Ready during the idling	STRAS2	—	—	—	—	No interrupt	Unload the head immediately (HLD="Low")

\* These errors except STRB5 and STRA2 are reset by RES inputs.

\*\* Head is unloaded if the new command is not issued during the settling time after Read/Write command ends.

# HD6844P, HD68A44P

## DMAC (Direct Memory Access Controller)

The HD6844P Direct Memory Access Controller (DMAC) performs the function of transferring data directly between memory and peripheral device controllers. It controls the address and data buses in place of the MPU in bus organized systems such as the HMCS6800 Microprocessor System.

The bus interface of the HD6844P includes select, read/write, interrupt, transfer request/grant, and bus interface logic to allow the data transfer over an 8-bit bidirectional data bus. The functional configuration of the DMAC is programmed via the data bus. The internal structure provides for control and handling of four individual channels, each of which is separately configured. Programmable control registers provide control for the transfer location and length, individual channel control and transfer mode configuration, priority of servicing, data chaining, and interrupt control. Status and control lines provide control to the peripheral controllers.

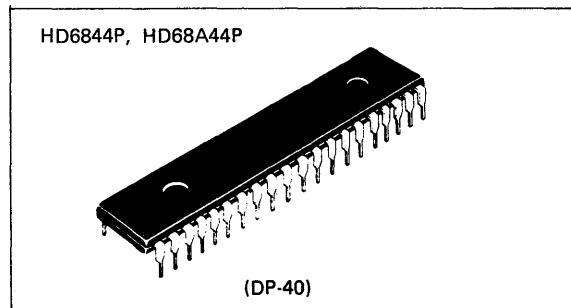
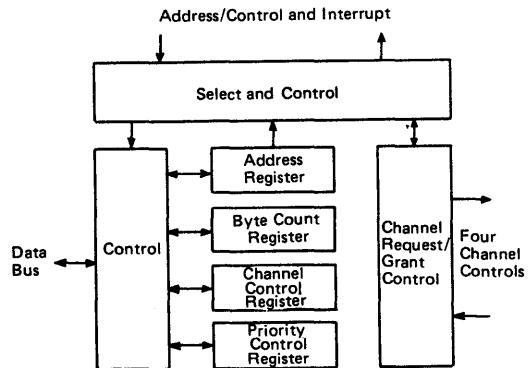
The mode of transfer for each channel can be programmed as cycle-stealing or a burst transfer mode.

Typical applications would be with the Floppy Disk Controller (FDC), etc..

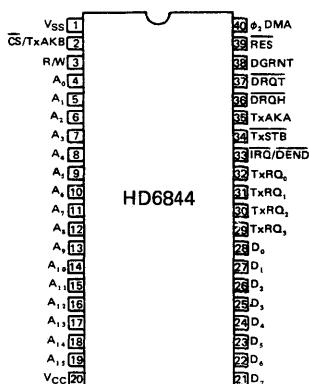
### ■ FEATURES

- Four DMA Channels, Each Having a 16-Bit Address Register and a 16-Bit Byte Count Register
- 1 M Byte/Sec (HD6844P), 1.5 M Byte/Sec (HD68A44P) Maximum Data Transfer Rate
- Selection of Fixed or Rotating Priority Service Control
- Separate Control Bits for Each Channel
- Data Chain Function
- Address Increment or Decrement Update
- Programmable Interrupts and DMA End to Peripheral Controllers
- Compatible with MC6844

### ■ BLOCK DIAGRAM



### ■ PIN ARRANGEMENT



(Top View)

### ■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V <sub>CC</sub> *	-0.3 ~ +7.0	V
Input Voltage	V <sub>in</sub> *	-0.3 ~ +7.0	V
Operating Temperature	T <sub>opr</sub>	-20 ~ +75	°C
Storage Temperature	T <sub>stg</sub>	-55 ~ +150	°C

\* With respect to V<sub>SS</sub> (SYSTEM GND)

(NOTE) 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.

### ■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol	min	typ	max	Unit
Power Supply Voltage	V <sub>CC</sub> *	4.75	5	5.25	V
Input Voltage	V <sub>IL</sub> *	-0.3	—	0.8	V
	V <sub>IH</sub> *	2.0	—	V <sub>CC</sub>	V
Operating Temperature	T <sub>opr</sub>	-20	25	75	°C

\* With respect to V<sub>SS</sub> (SYSTEM GND)

### ■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V<sub>CC</sub>=5V±5%, V<sub>SS</sub>=0V, T<sub>a</sub>=-20~+75°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ*	max	Unit
Input "High" Voltage	V <sub>IH</sub>			2.0	—	V <sub>CC</sub>	V
Input "Low" Voltage	V <sub>IL</sub>			-0.3	—	0.8	V
Input Leakage Current	TxRQ <sub>0~3</sub> , φ <sub>2</sub> DMA, RES, DGRNT	I <sub>in</sub>	V <sub>in</sub> =0~5.25V	—	—	2.5	μA
Three-State (off state) Leakage Current	A <sub>0</sub> ~A <sub>15</sub> , D <sub>0</sub> ~D <sub>7</sub> , R/W	I <sub>TSI</sub>	V <sub>in</sub> =0.4~2.4V	-10	—	10	μA
Output "High" Voltage	D <sub>0</sub> ~D <sub>7</sub>	V <sub>OH</sub>	I <sub>OH</sub> =-205μA	2.4	—	—	V
	A <sub>0</sub> ~A <sub>15</sub> , R/W		I <sub>OH</sub> =-145μA	2.4	—	—	
	All Other Outputs		I <sub>OH</sub> =-100μA	2.4	—	—	
Output "Low" Voltage	V <sub>OL</sub>		I <sub>OL</sub> =1.6mA	—	—	0.4	V
Source Current	CS/TxAKB	I <sub>CS</sub>	V <sub>in</sub> =0V, Fig. 10	—	10	16	mA
Power Dissipation	P <sub>D</sub>			—	500	1000	mW
Input Capacitance	φ <sub>2</sub> DMA	C <sub>in</sub>	V <sub>in</sub> =0V, T <sub>a</sub> =25°C f=1.0MHz	—	—	20	pF
	D <sub>0</sub> ~D <sub>7</sub> , CS, A <sub>0</sub> ~A <sub>4</sub> , R/W			—	—	12.5	
	TxRQ <sub>0~3</sub> , RES, DGRNT			—	—	10	
Output Capacitance	C <sub>out</sub>		V <sub>in</sub> =0V, T <sub>a</sub> =25°C, f=1MHz	—	—	12	pF

\* V<sub>CC</sub>=5.0V, T<sub>a</sub>=25°C

• AC CHARACTERISTICS (Load Condition Fig. 9)

1. CLOCK TIMING

Item	Symbol	Test Condition	HD6844P			HD68A44P			Unit	
			min	typ	max	min	typ	max		
$\phi_2$ DMA Cycle Time	$t_{\text{cy}\phi}$	Fig. 2	1,000	—	—	666	—	—	ns	
$\phi_2$ DMA Pulse Width	“High” Level “Low” Level	PW <sub>φH</sub> PW <sub>φL</sub>	Fig. 2	450	—	—	280	—	—	ns
$\phi_2$ DMA Rise and Fall Time	$t_{\phi r}, t_{\phi f}$	Fig. 2	—	—	25	—	—	25	ns	

2. DMA TIMING (Load Condition Fig. 9)

Item	Symbol	Test Condition	HD6844P			HD68A44P			Unit	
			min	typ	max	min	typ	max		
TxRQ Setup Time	$\phi_2$ DMA Rising Edge	$t_{TQS1}$	Fig. 3	120	—	—	120	—	—	ns
	$\phi_2$ DMA Falling Edge	$t_{TQS2}$		210	—	—	210	—	—	
TxRQ Hold Time	$\phi_2$ DMA Rising Edge	$t_{TQH1}$	Fig. 3	20	—	—	10	—	—	ns
	$\phi_2$ DMA Falling Edge	$t_{TQH2}$		20	—	—	10	—	—	
DGRNT Setup Time	DGRNT	$t_{DGS}$	Fig. 4	155	—	—	125	—	—	ns
DGRNT Hold Time	DGRNT	$t_{DGH}$		10	—	—	10	—	—	
Address Output Delay Time	$A_0 \sim A_{15}$ , R/W, TxSTB	$t_{AD}$	Fig. 6	—	—	270	—	—	180	ns
Address Output Hold Time	$A_0 \sim A_{15}$ , R/W TxSTB	$t_{AHO}$	Fig. 6	30	—	—	20	—	—	ns
				35	—	—	35	—	—	
Address Three-State Delay Time	$A_0 \sim A_{15}$ , R/W	$t_{ATSD}$	Fig. 7	—	—	270	—	—	270	ns
Address Three-State Recovery Time	$A_0 \sim A_{15}$ , R/W	$t_{ATSR}$	Fig. 7	—	—	270	—	—	270	ns
Delay Time	DRQH, DRQT	$t_{DQD}$	Fig. 5	—	—	375	—	—	250	ns
TxAK Delay Time	$\phi_2$ DMA Rising Edge	$t_{TKD1}$	Fig. 5	—	—	400	—	—	310	ns
	DGRNT Rising Edge	$t_{TKD2}$	Fig. 8	—	—	190	—	—	160	
IRQ/DEND Delay Time	$\phi_2$ DMA Falling Edge	$t_{DED1}$	Fig. 6	—	—	300	—	—	250	ns
	DGRNT Rising Edge	$t_{DED2}$	Fig. 8	—	—	190	—	—	160	

3. BUS TIMING

1) READ TIMING

Item	Symbol	Test Condition	HD6844P			HD68A44P			Unit	
			min	typ	max	min	typ	max		
Address Setup Time	$A_0 \sim A_4$ , R/W, CS	$t_{AS}$	Fig. 2	140	—	—	140	—	—	ns
Address Input Hold Time	$A_0 \sim A_4$ , R/W, CS	$t_{AHI}$		10	—	—	10	—	—	ns
Data Delay Time	$D_0 \sim D_7$	$t_{DDR}$		—	—	320	—	—	220	ns
Data Access Time	$D_0 \sim D_7$	$t_{ACC}$		—	—	460	—	—	360	ns
Data Output Hold Time	$D_0 \sim D_7$	$t_{DHR}$		10	—	—	10	—	—	ns

## 2) WRITE TIMING

Item	Symbol	Test Condition	HD68A44P			HD6844P			Unit
			min	typ	max	min	typ	max	
Address Setup Time	$A_0 \sim A_4, R/W, \bar{CS}$	Fig. 2	140	—	—	140	—	—	ns
Address Input Hold Time	$A_0 \sim A_4, R/W, \bar{CS}$		10	—	—	10	—	—	ns
Data Setup Time	$D_0 \sim D_7$		195	—	—	80	—	—	ns
Data Input Hold Time	$D_0 \sim D_7$		10	—	—	10	—	—	ns

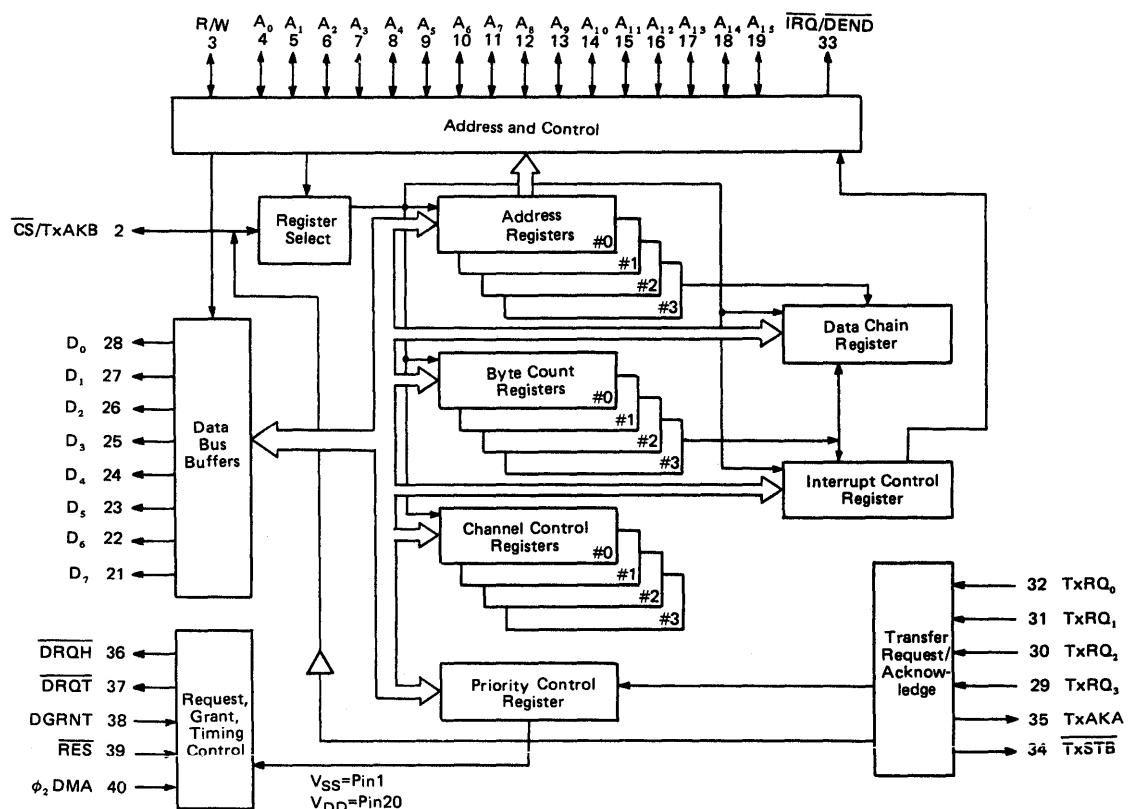


Figure 1 Expanded Block Diagram

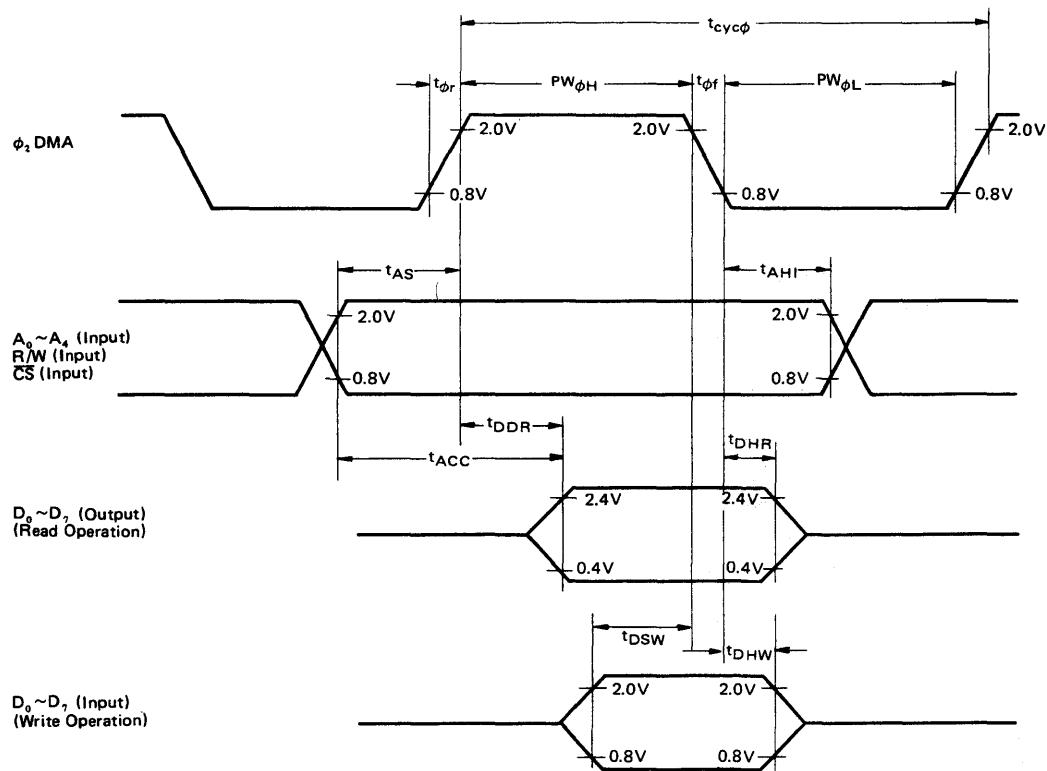


Figure 2 Read/Write Sequence

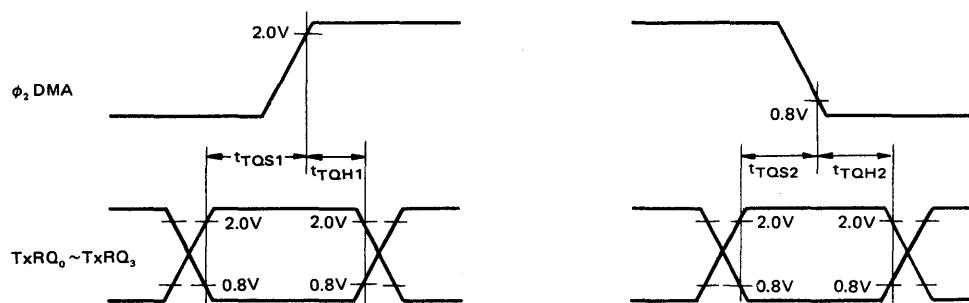


Figure 3 Timing of TxRQ Input

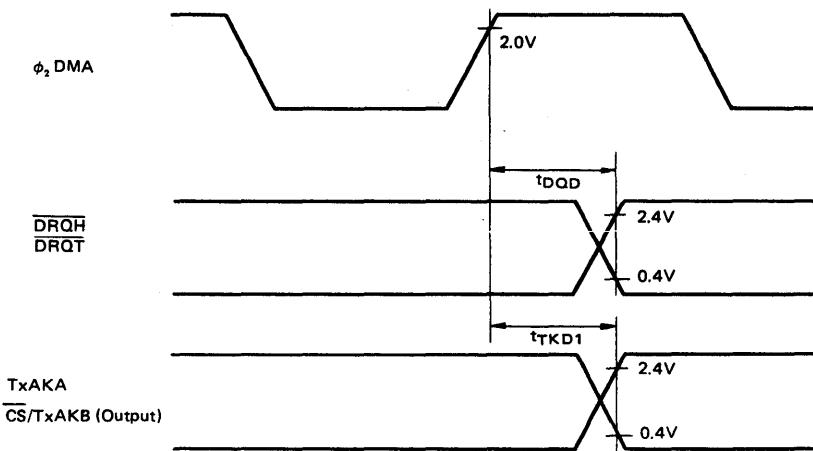
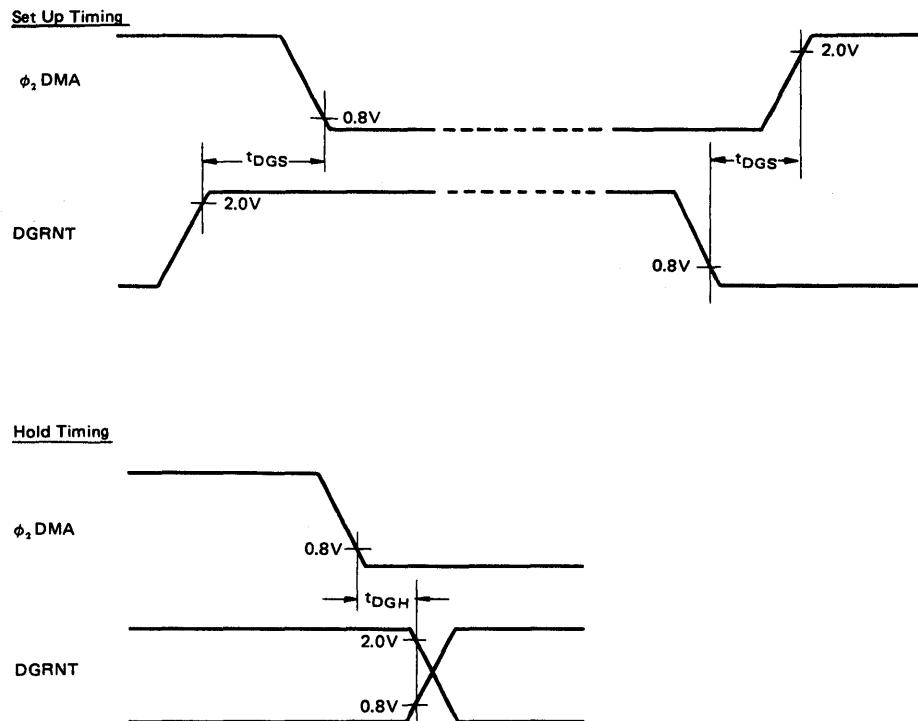


Figure 5 Timing of DRQH, DRQT, TxAK Outputs

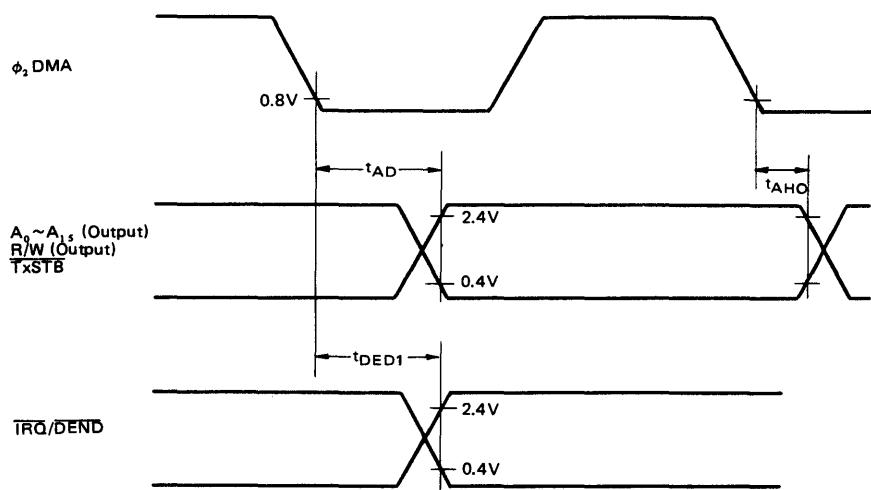
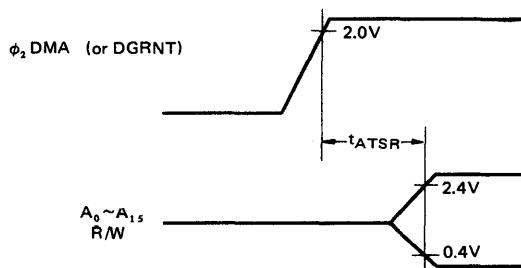


Figure 6 Timing of Address and  $\overline{\text{IRQ}}/\overline{\text{DEND}}$  Outputs

Recovery Time of Address Three-state



Delay Time of Address Three-state

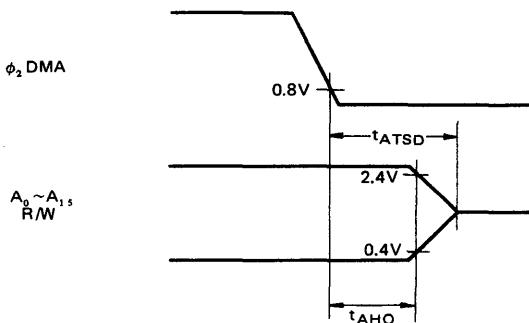


Figure 7 Timing of Address Three-state

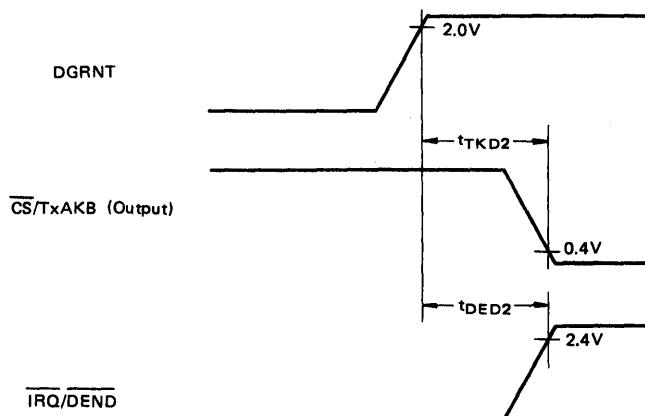


Figure 8 Timing of Synchronous DGRNT Output

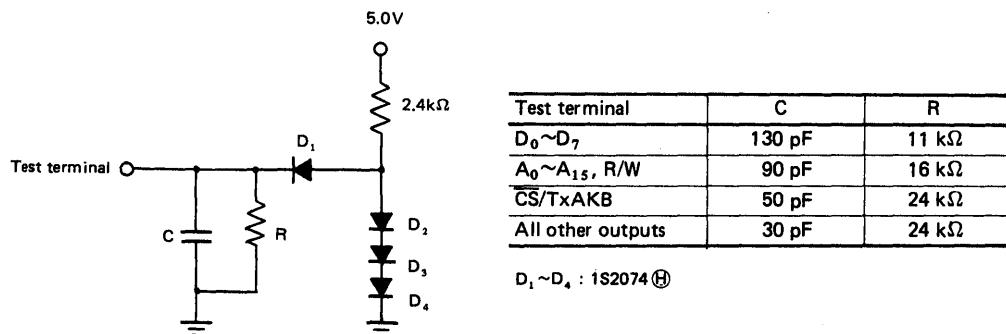
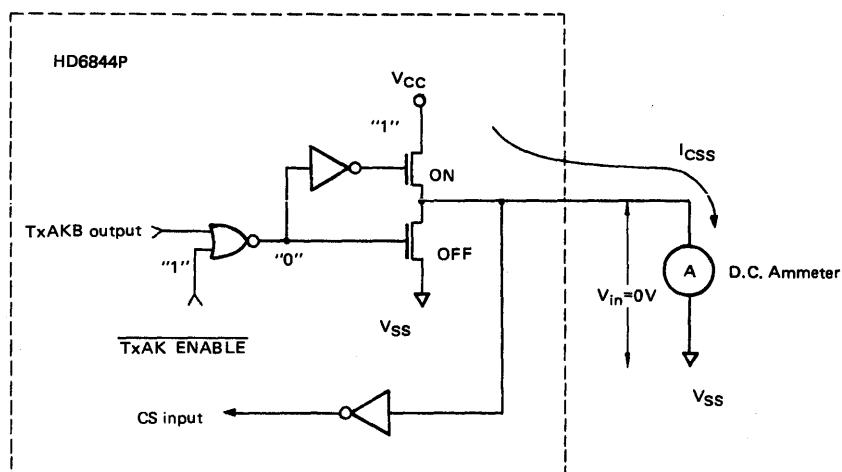


Figure 9 Load Circuit

Figure 10 Source Current Measurement Circuit for  $\overline{CS}/TxAKB$  Terminal

## ■ DEVICE OPERATION

The DMAC has fifteen addressable registers, eight of them are sixteen bits in length. Each channel has a separate Address Register and a Byte Count Register, each of which is sixteen bits. There are also four Channel Control Registers. The three General Control Registers common to all four channels are the Priority Control Register, the Interrupt Control Register, and the Data Chain Register.

To prepare a channel for DMA, the Address Registers must be loaded with the starting memory address and the Byte Count Register loaded with the number of bytes to be transferred. The bits in the Channel Control Register establish the direction of the transfer, the mode, and the address increment or decrement after each cycle. Each channel can be set for one of three transfer modes: Three-State Control (TSC) Steal, Halt Steal, or Halt Burst. Two read-only status bits in the Channel Control Register indicate when the channel is busy transferring data and when the DMA transfer is completed.

The Priority Control Register enables the transfer requests from the peripheral controllers and establishes either a fixed priority or rotating priority scheme of servicing these requests.

When the DMA transfer for a channel is complete (the Byte Count Register is zero), a DMA End signal is directed to the peripheral controller and an IRQ goes to the MPU. Enabling of these interrupts is done in the interrupt Control Register. The IRQ flag bit is read from this register.

Chaining of data transfers is controlled by the Data Chain Register. When enabled, the contents of the Address and Byte Count Registers for channel #3 are put into the registers of the channel selected for chaining when its Byte Count Register becomes zero. This allows for repetitively reading or writing a block of memory.

During the DMA mode, the DMAC controls the address bus and data bus for the system as well as providing the R/W line and a signal to be used as VMA. When a peripheral device controller desires a DMA transfer, it is requested by a Transfer Request. Assuming this request is enabled and meets the test of highest priority, the DMAC will issue a DMA Request. When the DMAC receives the DMA Grant, it gives a Transfer Acknowledge to the peripheral device controller, at which time the data is transferred. When the channel's Byte Count Register equals zero, the transfer is complete and a DMA End is given to the peripheral device controller, and an IRQ is given to the MPU.

### ● Initialization

During a power-on sequence, the DMAC is reset via the RES input. All registers, with the exception of the Address and Byte Count Registers, are set to a logic "0" state. This disables all requests and the Data Chain function while masking all interrupts. The Address, Byte Count, and Channel Control Registers must be programmed before the respective transfer request bit is enabled in the Priority Control Register.

### ● Transfer Modes

There are three ways in which a DMA transfer may be done. The one used is determined by the data transfer rate required, the number of channels attached, and the hardware complexity allowable. Refer to Figures 11 through 13.

Two of the modes, TSC Steal and Halt Steal, are done by cycle-stealing from the MPU. The Three-State Control (TSC) Steal mode is initiated by the DMAC bringing the DRQT line "Low". This line goes to the system clock driver which returns a "High" on DGRNT on the rising edge of the system  $\phi_1$  clock.

The DGRNT signal must cause the address control and data lines to go to the high impedance state. The DMAC now supplies the address from the Address Register of the channel requesting. It also supplies the R/W signal as determined from the Channel Control Register. After one byte is transferred, control is returned to the MPU. This method stretches the  $\phi_1$  and  $\phi_2$  clocks while the DMAC uses the memory.

The second method of cycle-stealing is the Halt Steal mode. This method actually halts the MPU instead of stretching the  $\phi_1$  clock for the transfer period. This mode is initiated by the DMAC bringing the DRQH line "Low". This line connects to the MPU HALT input. The MPU Bus Available (BA) line is the DGRNT input to the DMAC. While the MPU is halted, its Address Bus, Data Bus, and R/W are in the high impedance state. The DMAC now supplies the address and R/W line. After one byte is transferred, the HALT line is returned "High" and the MPU regains control. In this mode, the MPU stops internal activity and is removed from the system while the DMAC uses the memory.

The third mode of transfer is the Halt Burst mode. This mode is similar to the Halt Steal mode, except that the transfer does not stop with one byte. The MPU is halted while an entire block of data is transferred. When the channel's Byte Count Register equals zero, the transfer is complete and control is returned to the MPU. This mode gives the highest data transfer rate, at the expense of the MPU being inactive during the transfer period.

## ■ INPUT/OUTPUT FUNCTIONS

### ● DMAC Interface Signals for the MPU

The DMAC interfaces with the HMCS6800 MPU through the eight-bit bidirectional data bus, the CS line, five address lines, an IRQ line, the Read/Write line, and the RES line. These signals, in conjunction with the HMCS6800 VMA output, permit the MPU to have access to the DMAC. Four other lines associated with the MPU and the clock driver are the DRQT, DRQH, DGRNT, and the  $\phi_2$  DMA.

### Bidirectional Data ( $D_0 \sim D_7$ )

The Bidirectional Data lines ( $D_0 \sim D_7$ ) allow for data transfer between the DMAC and the MPU. The data bus output drivers are three-state devices that remain in the high impedance state except when the MPU performs DMAC read operations.

### Chip Select/Transfer Acknowledge B (CS/T x AKB)

This line is multiplexed, serving both as an input and an output. CS/TxAKB is an output in the four-channel mode during the DMA transfer. At all other times, it is a high impedance TTL compatible input used to address the DMAC. The DMAC is selected when CS/TxAKB is "Low". VMA must be used in generating this input to insure that false selects will not occur. Transfers of data to and from the DMAC are then performed under the control of the  $\phi_2$  DMA, Read/Write, and  $A_0 \sim A_4$  address lines. In the four-channel mode when TxAKB is needed, the CS gate must have an open-collector output (a pull-up resistor should not be used). In the two-channel mode, CS/TxAKB is always an input.

### Address Lines $A_0 \sim A_4$ ( $A_0 \sim A_4$ )

Address lines  $A_0 \sim A_4$  are both input and output lines. In the MPU mode, these are high impedance inputs used to address the DMAC registers. In the DMA mode, these lines are outputs which are set to the contents of the Address Register of the channel being processed.

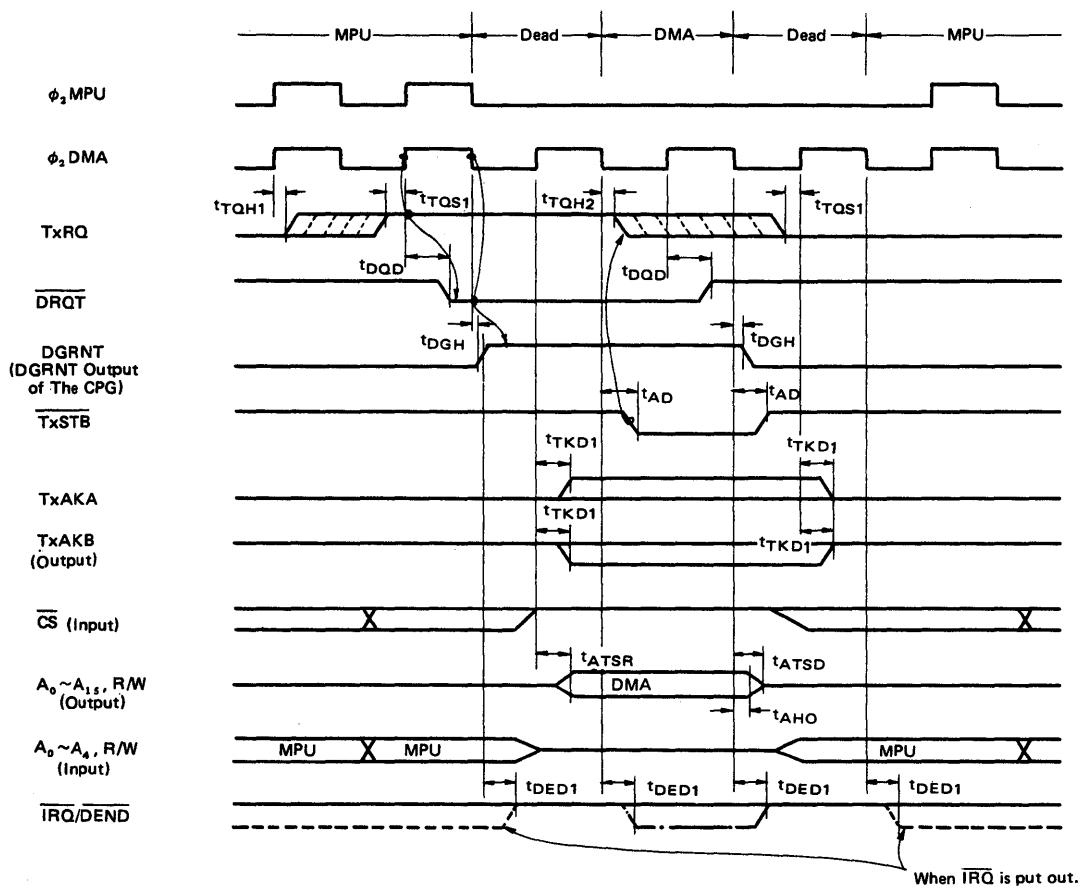
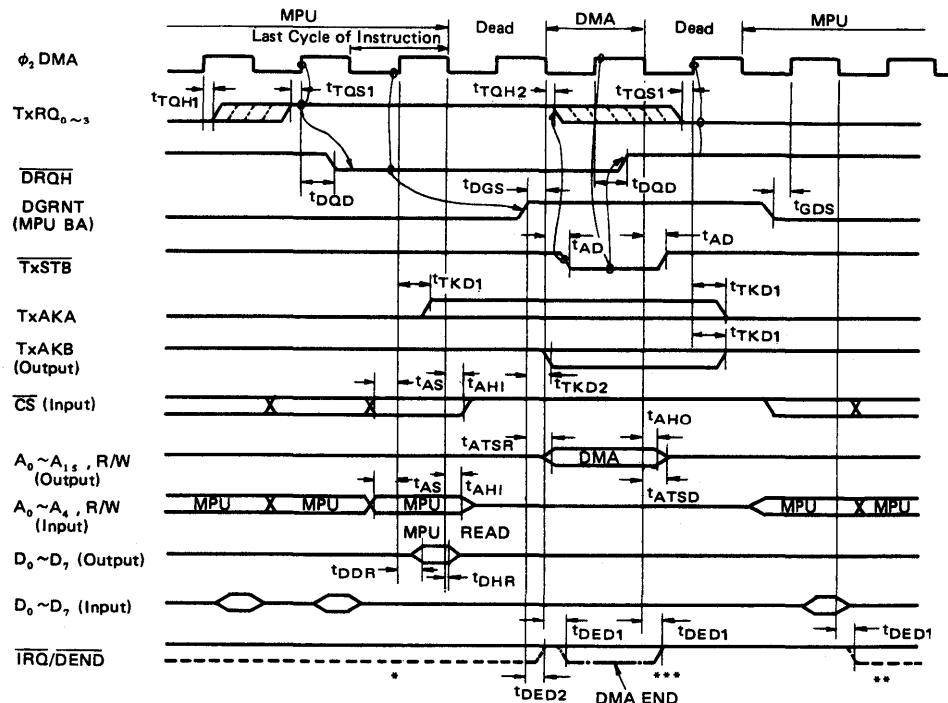


Figure 11 TSC Cycle Steal Mode



[NOTE] \* IRQ of another channel or its own IRQ (remaining)  
 \*\* Its own IRQ (put out) or its own IRQ (remaining) or IRQ of another channel  
 \*\*\* This is the last cycle of transfer.

Figure 12 HALT Cycle Steal Mode

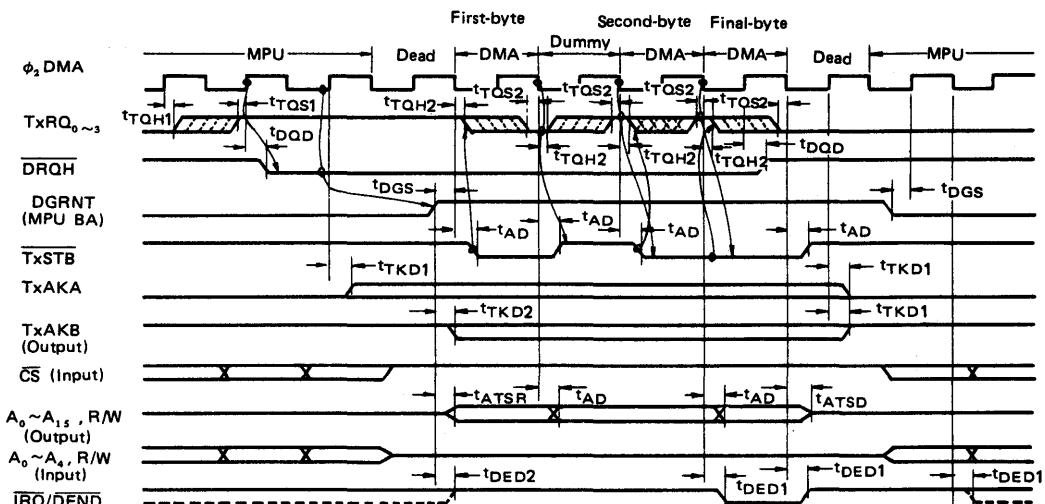


Figure 13 HALT Burst Mode

**Interrupt Request/DMA End (IRQ/DEND)**

IRQ/DEND is a TTL compatible, active "Low" output that is used to interrupt the MPU and to signal the peripheral controller that the data block transfer has ended. If the Interrupt has been enabled, the IRQ/DEND line will go "Low" after the last DMA cycle of a transfer. An open collector gate must be connected to DGRNT and IRQ/DEND to prevent false interrupts from the DEND signal when interrupts are not enabled. See Figures 12 and 18.

**Read/Write (R/W)**

Read/Write is a TTL compatible line that is a high impedance input in the MPU mode and an output in the DMA mode. In the MPU mode, it is used to control the direction of data flow through the DMAC's input/output data bus interface. When Read/Write is "High" (MPU read cycle) and the chip is selected, DMAC data output buffers are turned on and a selected register is read. When it is "Low", the DMAC output drivers are turned off and the MPU writes into a selected register.

In the DMA mode, Read/Write is an output to drive the memory and peripheral controllers. Its state is determined by bit 0 of the Channel Control Register for the channel being serviced. When Read/Write is "High", the memory is read and the data from the memory is written into the peripheral controller. When it is "Low", the peripheral controller is read and its data stored in the memory. In the DMA mode, the DMAC data buffers are off.

**Reset (RES)**

The RES input provides a means of resetting the DMAC from an external source. In the "Low" state, the RES input causes all registers, with the exception of the Address and Byte Count Registers, to be reset to the logic "0" state. This disables all transfer requests, masks all interrupts, disables the data chain function, and puts each Channel Control Register into the condition of memory write, Halt Steal transfer mode, and address increment.

**Transfer Signals to the MPU**

Two DMA request output lines and a DMA Grant input line, together with the system clock, synchronize the DMAC with the MPU system.

**DMA Request Three-State Control Steal (DRQT)**

This active "Low" output requests a DMA transfer for a channel configured for the TSC Steal transfer mode. This line is connected to the system clock driver, requesting a  $\phi_1$  clock stretch. It will remain in the "Low" state until the transfer has begun.

**DMA Request Halt (DRQH)**

This active "Low" output requests a DMA transfer for a channel programmed for the Halt Steal or Halt Burst mode transfer. This line is connected directly to the MPU HALT input and remains "Low" until the last byte has begun to be transferred.

**DMA Grant (DGRNT)**

This is a high impedance input to the DMAC, giving it control of the system busses. For the TSC Steal mode, the signal comes from the system clock drive circuit (DMA Grant), indicating that the clock is being stretched. For either of the Halt modes, this signal is the Bus Available from the MPU,

indicating that the MPU has halted and turned control of its busses over to the DMAC. For a design involving TSC Steal and Halt mode transfers, this input must be the OR of the clock driven DMA Grant and the MPU BA.

 **$\phi_2$  DMA**

Transferring in and out of the DMAC registers, sampling of channel request lines and gating of other control signals to the system is done internally in conjunction with the  $\phi_2$  DMA high impedance input. This input must be the system memory clock (non-stretched  $\phi_2$  clock).

**Transfer Signals From the Peripheral Controller****Transfer Request ( $TxRQ_0 \sim TxRQ_3$ )**

Each of the four channels has its own high impedance input request for transfer line. The peripheral controller requests a transfer by setting its TxRQ line "High" (a logic "1"). The lines are sampled according to the priority and enabling established in the Priority Control Register. In the Steal mode and the first byte of the Halt Burst mode, the TxRQ signals are tested on the positive edge of  $\phi_2$  DMA and the highest priority channel is strobed. Once strobed, the TxRQs are not tested until that channel's data transfer is finished. In the succeeding bytes of the Halt Burst mode transfer, the TxRQ is tested on the negative edge of  $\phi_2$  DMA, and data is transferred on the next  $\phi_2$  DMA cycle if TxRQ is "High".

**Transfer Signals to the Peripheral Controller**

Two encoded lines select the channel to be serviced. A strobe line acknowledges the request and performs the transfer. The DEND line signals to the peripheral controller that the DMA transfer is completed.

**Transfer Acknowledge A (TxAKA)**

The Transfer Acknowledge A (TxAKA) is a TTL compatible output used in conjunction with the CS/TxAKB line to select the channel to be strobed for transfer and to give the DMA End Signal. In the two-channel mode, only TxAKA is used to select channel 0 or channel 1, and CS/TxAKB is always an input.

**Chip Select/Transfer Acknowledge B (CS/TxAKB)**

In the DMA mode, this dual purpose line is encoded together with TxAKA to select the channel being serviced. Table 1 shows the encoding order.

Table 1 Encoding Order

CS/TxAKB	TxAKA	Channel #
0	0	0
0	1	1
1	0	2
1	1	3

**Transfer Strobe (TxSTB)**

The TxSTB causes acknowledgement to be given to the peripheral controller and transfers the data to or from the memory. This line is also intended to be the VMA signal for the system in the DMA mode. In a one-channel system, TxSTB may be inverted and run to the peripheral controller's Acknowledge input. In a two or four-channel system, TxSTB enables the decode of TxAKA and CS/TxAKB to select the device controller to be acknowledged.

**Interrupt Request/DMA End (IRQ/DEND)**

In the DMA mode, this dual purpose line is "Low" for the last byte of transfer, indicating a DMA End. This occurs when the Byte Count register decrements to zero.

This line, through the decode of TxAKA and CS/TxAKB, can be used to strobe a DMA End to each device controller.

**• Address Lines to the Memory****Address Lines ( $A_0 \sim A_{15}$ )**

These output lines are in the high impedance state during the MPU mode. In the DMA mode, these lines are outputs which are set to the contents of the Address Register of the channel being processed.

**■ THE DMAC REGISTERS**

All of the fifteen registers in the DMAC are Read/Write registers, although some of the bits are read only status bits.

**• Address Registers**

Each channel has its own individual 16-bit Address Register. Before a DMA transfer is begun, the starting address for the transfer must be loaded into the Address Register. Depending on the state of the Channel Control Register bit 3, the Address Register will be decremented or incremented after each byte of transfer.

**• Byte Count Registers**

Each channel also has its own Byte Count Register. Before the DMA transfer, this register must be loaded with the number of bytes to be transferred. Since it is 16 bits in length, the transfer can be up to 65,536 bytes of data. The Byte Count Register is decremented at the beginning of each DMA cycle.

**• Channel Control Registers**

The control of each channel's DMA transfer is programmed into its Channel Control Register. Bits 4 and 5 are unused.

**Read/Write (R/W), Bit 0**

The direction of the DMA transfer is controlled by this bit. When it is "1", the peripheral controller reads the memory. When it is "0", the transfer will be in the opposite direction, thus writing into the memory. The system R/W line is in the same state as this R/W bit in the DMA mode. The device controller must change the sense of its R/W input during the DMA mode.

**Burst/Steal, Bit 1**

This bit, along with bit 2, selects the mode of the DMA transfer. With bit 1 "1", the Burst mode is selected. A "0" selects the Steal mode. Table 2 shows the mode selection.

**TSC/Halt, Bit 2**

Bit 2 helps select the mode of DMA transfer. When this bit is "1", the TSC mode is selected. When "0", the Halt mode is

Table 2 Mode Select

Bit 2	Bit 1	DMA Transfer Mode
0	0	Halt Steal
0	1	Halt Burst
1	0	TSC Steal
1	1	(Illegal)

selected. Table 2 shows the mode selection. A TSC Burst mode is illegal for HMCS6800 family processors due to restrictions on φ1 clock stretching for these products.

**Address Up/Down, Bit 3**

Bit 3 controls the change in the Address Register for each DMA cycle. If this bit is "0", the Address Register will be incremented each time the Byte Count Register decrements. If the bit is "1", the Address Register will be decremented.

**Busy/Ready Flag, Bit 6**

The Busy/Ready Flag is a read only status bit that indicates a DMA transfer is in process on that channel. This bit goes "1" at the beginning of the transfer and remains so until the IRQ/DEND has been "Low" for one cycle (DMA End). The bit is then reset and the channel can again be configured for a transfer.

**DMA End Flag (DEND), Bit 7**

The DEND bit indicates a DMA block transfer has ended. This bit is set at the same time the Busy/Ready Flag is reset. The DEND bit is reset by the MPU reading the Channel Control Register. This bit causes an interrupt if enabled in the Interrupt Control Register.

**• Priority Control Register**

The enabling and prioritizing of the transfer requests (TxRQs) are done in the Priority Control Register. Bits 4 through 6 are unused.

**TxRQ Enable (TxENO~3), Bits 0~3**

The four channels are individually enabled by setting the respective TxEN bit "1". A "0" on any of these bits disables recognition of the transfer request for that channel. The bit number equals the channel number (i.e., bit 2 equals channel #2).

**Rotate Control, Bit 7**

The DMAC priority service routine is selected by this Rotate Control bit. When it is "0", the fixed mode is selected. Channel #0 has the highest priority, channel #1 the next highest, and so on down. When this bit is "1", a rotating routine is used. This routine states that initially it will be the same as in the fixed mode. But once a channel has been serviced, it moves to the lowest priority and those that were below it advance to the next highest priority.

**• Interrupt Control Register**

An interrupt is caused by a channel completing its DMA block transfer. DEND (Channel Control Register, Bit 7) flags this condition for each channel. Bits 4 through 6 are unused.

**IRQ Enable (IE0~3) Bits 0~3**

Each channel is separately enabled to cause the interrupt. A "1" enables an interrupt from the channel; a "0" masks the interrupt. The bit number equals the channel number (i.e., bit 2 equals channel #2).

**IRQ Flag, Bit 7**

This read only bit indicates an IRQ is requested of the MPU when it is "1". If the interrupt is enabled (IRQ Enable = "1") when a channel's DEND flag (Channel Control Register, bit 7) goes "1", the IRQ Flag bit also goes "High". It is reset by the MPU reading the Channel Control Register that caused the

interrupt.

#### • Data Chain Register

Repetitive reading or writing of a block of memory can be done in the data chain function. A DMA transfer cannot be active on channel #3 during the data chain. Bits 4 through 7 are unused.

#### Data Chain Enable (DCE), Bit 0

The data chain function is enabled when this bit is "1".

#### Data Chain Channel Selects A, B (DCA, DCB), Bits 1 and 2

The channel updated by data chaining is selected by bits 1 and 2, according to the order shown in Table 3.

The data chain function is performed by transferring the contents of Channel #3 Address and Byte Count Registers into the respective registers of the channel selected by bits 1 and 2. This transfer is done during the cycle of  $\phi_2$ DMA following the Byte Count Register having decremented to zero.

#### Channel Control Register

DEND	Bit 7 – Is set at end of DMA block transfer; reset by MPU reading the Channel Control Register.
Busy/Ready Flag	Bit 6 – Status bit set when in transfer: cleared after DMA End.
Address Up/Down	Bit 3 – "1" = decrement Address Register for each byte; "0" = increment.
TSC/Halt	Bit 2 – "1" = select TSC mode; "0" = Halt modes.
Burst/Steal	Bit 1 – "1" = select Burst mode; "0" = Steal modes.
R/W	Bit 0 – "1" = device controller reads memory; "0" = write into memory.

#### Priority Control Register

Rotate Control	Bit 7 – "1" = use rotate routine; "0" = fixed: 0, 1, 2, 3 priority.
TxRQ Enable 0~3	Bits 0~3 – "1" = enable transfer request for the channel; "0" = request disabled.

#### Interrupt Control Register

IRQ Flag	Bit 7 – This Flag is set by DENDs in Channel Control Registers when enabled; Reset by reading the Register that caused it to be set.
----------	--

Table 3 Channel Select

DCB Bit 2	DCA Bit 1	Channel #
0	0	0
0	1	1
1	0	2
1	1	(Illegal)

#### Two/Four Channel Select (2/4), Bit 3

The DMAC is configured to handle two or four channels by bit 3. This bit "1" selects the four-channel mode. In this mode, the CS/TxAKB becomes a chip select in the MPU mode and a Transfer Acknowledge B for the DMA mode.

With bit 3 "0", the two-channel mode is selected, and the CS/TxAKB line is always a chip select, both for the MPU and the DMA mode.

IRQ Enable 0~3 Bits 0~3 – "1" = enable IRQ by DEND for the channel; "0" = IRQ masked.

#### Data Chain Register

Two/Four Channel Bit 3 – "1" = four-channel mode; "0" = two-channel.

Data Chain Channel Select Bits 2 and 1 – Binary equivalent of channel to be updated by chaining.

Data Chain Enable Bit 0 – "1" = enable Data Chain function; "0" = disabled.

Preparation of a channel for a DMA transfer requires:

- 1) Load the starting address into the Address Register.
- 2) Load the number of bytes into the Byte Count Register.
- 3) Program the Channel Control Register for the transfer characteristics: direction (bit 0), mode (bits 1 and 2), and the address update (bit 3).

The channel is now configured. To enable the transfer request, set the appropriate enable bit (bits 0~3) of the Priority Control Register, as well as the Rotate Control bit.

If an interrupt on DMA End is desired, the enable bit (bits 0~3) of the Interrupt Control Register must be set.

If data chaining for the channel is necessary, it is programmed into the Data Chain Register and the appropriate data must be written into the Address and Byte Count Registers for channel #3.

Table 4 DMAC Programming Model

Register	Address (Hex)	Register Content							
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Channel Control	1x*	DMA End Flag (DEND)	Busy/Ready Flag	Not Used	Not Used	Address Up/Down	TSC/Halt	Burst/Steal	Read/Write (R/W)
Priority Control	14	Rotate Control	Not Used	Not Used	Not Used	TxRQ Enable #3 (TxEN3)	TxRQ Enable #2 (TxEN2)	TxRQ Enable #1 (TxEN1)	TxRQ Enable #0 (TxENO)
Interrupt Control	15	IRQ Flag	Not Used	Not Used	Not Used	IRQ Enable #3 (IE3)	IRQ Enable #2 (IE2)	IRQ Enable #1 (IE1)	IRQ Enable #0 (IE0)
Data Chain	16	Not Used	Not Used	Not Used	Not Used	Two/Four Channel Select (2/4)	Data Chain Channel Select B	Data Chain Channel Select A	Data Chain Enable

\* The x represents the binary equivalent of the channel desired.

Table 5 Maximum Transfer Speed & Response Time of the DMA when  $t_{cyc\phi}$  equals 1  $\mu$ sec.

Mode	Maximum Transfer Speed ( $\mu$ sec/byte)	Response Time ( $\mu$ sec)	
		maximum	minimum
HALT Cycle Steal	(executing time of one instruction) + 3	(executing time of one instruction) +3.5 - $t_{TQH1}$ 2 - $t_{TQH2}$	3.5 + $t_{TQH1}$ 1 + $t_{TQH2}$
HALT Burst			
since second byte			
TSC Cycle Steal	4	3.5 - $t_{TQH1}$	2.5 + $t_{TQH1}$

A comparison of the response times and maximum transfer rates is shown in Table 5. The values are shown for a system clock rate of 1 MHz.

The two 8-bit bytes that form the registers in Table 6 are placed in consecutive memory locations, making it very easy to use the MPU index register in programming them.

#### ■ SYSTEM DESCRIPTION

The hardware configuration of the DMA can be for a one, two, or four-channel system. These are shown in Figures 14 through 16.

If the peripheral device controllers do not use the DEND signal, the AND gates generating them are not needed. As mentioned previously, the open collector gate to IRQ is to prevent false interrupts from the DEND signal when interrupts are disabled. In the four-channel mode, the CS gate must be open collector so that CS/TxAKB can become an output during the DMA cycle.

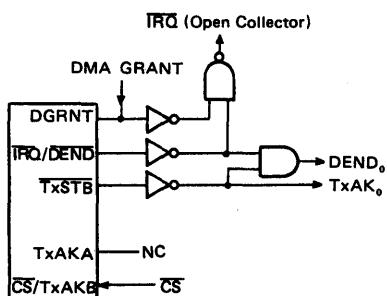


Figure 14 One Channel

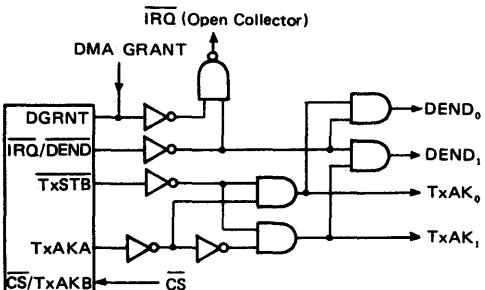


Figure 15 Two Channel

Fig. 17 shows an example of its minimum structure (1 channel, HALT mode, combination with FDC). Fig. 18 shows an example of its maximum structure. (but only one DMA is used.)

Table 6 Address and Byte Count Registers

Register	Channel	Address (Hex)
Address High	0	0
Address Low	0	1
Byte Count High	0	2
Byte Count Low	0	3
Address High	1	4
Address Low	1	5
Byte Count High	1	6
Byte Count Low	1	7
Address High	2	8
Address Low	2	9
Byte Count High	2	A
Byte Count Low	2	B
Address High	3	C
Address Low	3	D
Byte Count High	3	E
Byte Count Low	3	F

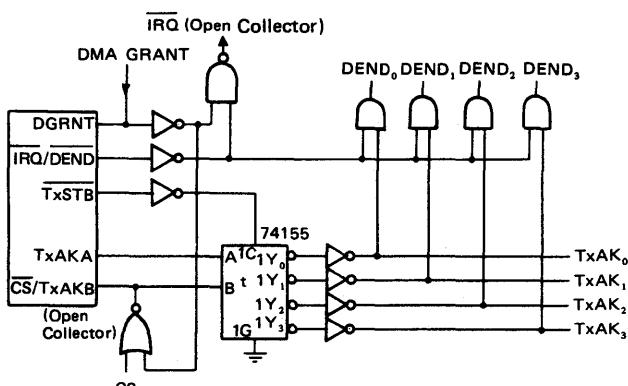


Figure 16 Four-Channel

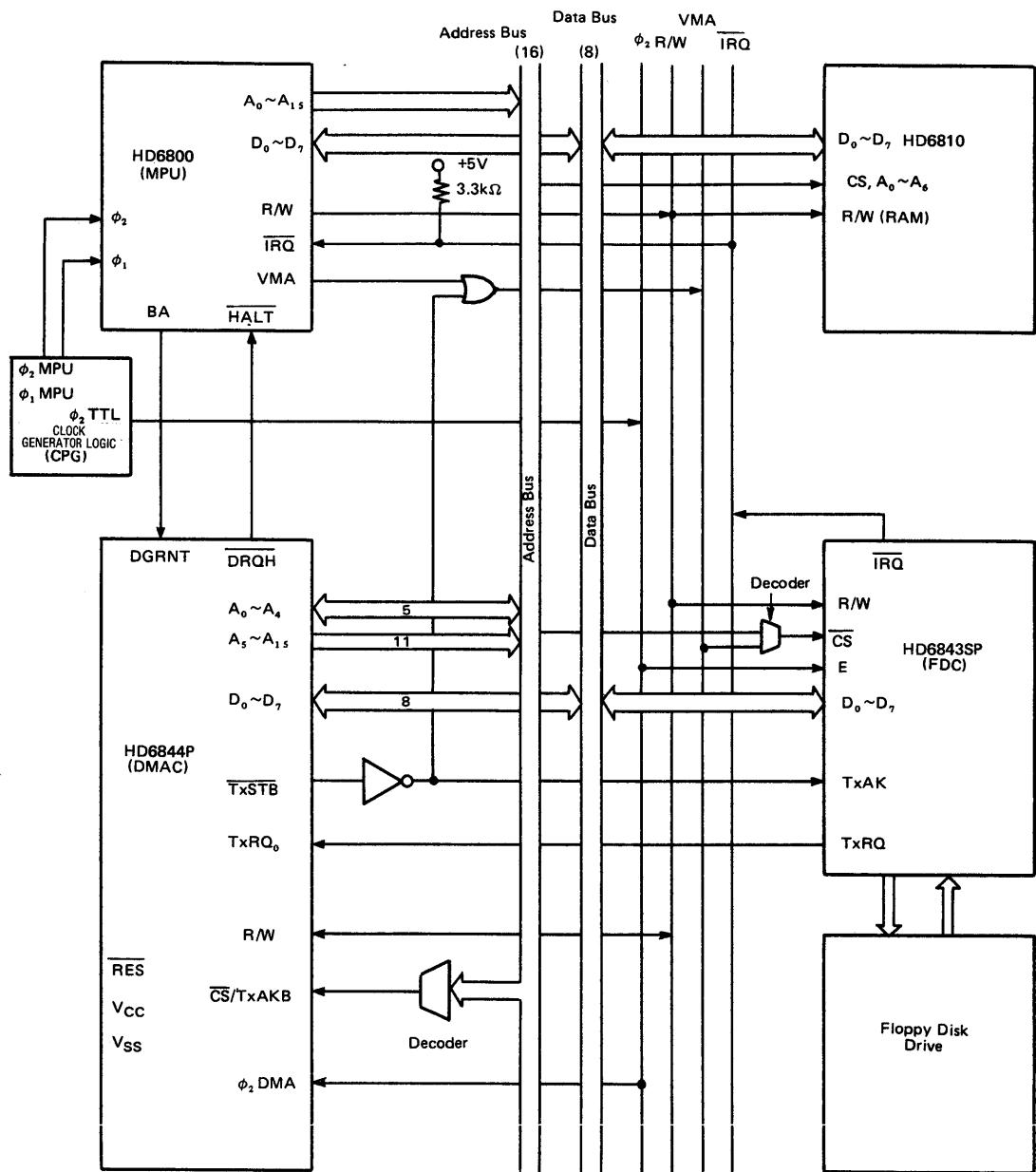


Figure 17 Example of DMA System Structure (1) (minimum)

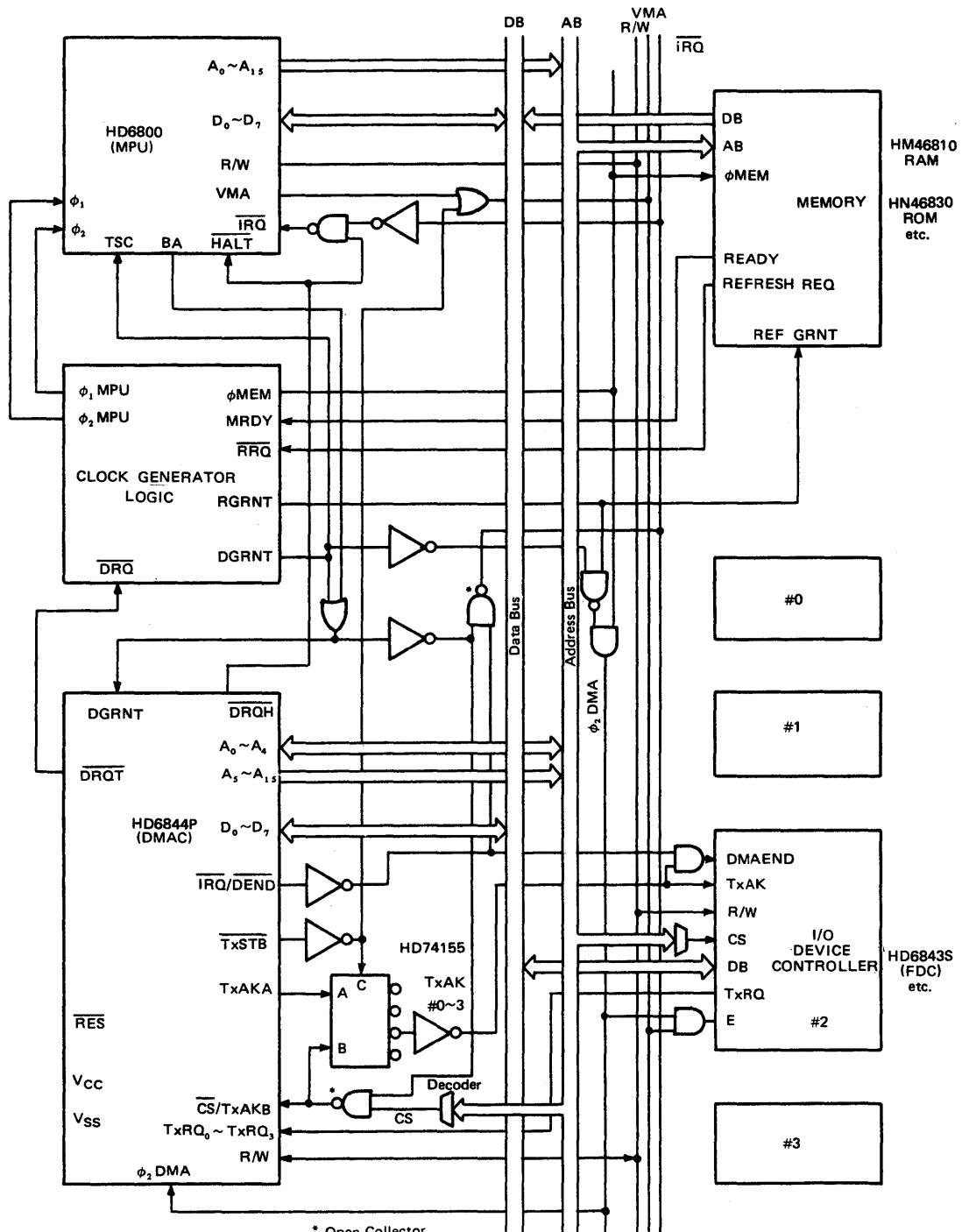


Figure 18 Example of DMA System Structure (2) (maximum)

# HD6845S, HD68A45S, HD68B45S

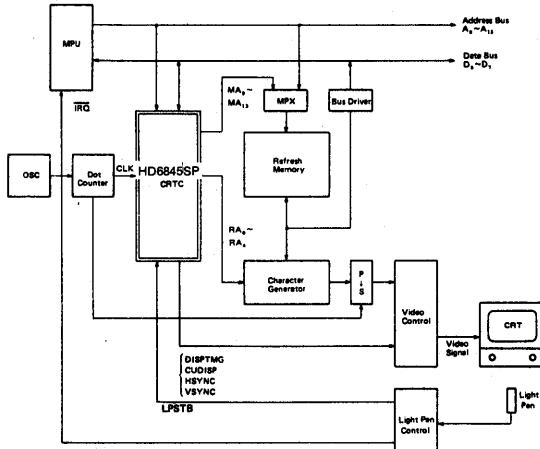
## CRTC (CRT Controller)

The CRTC is a LSI controller which is designed to provide an interface for microcomputers to raster scan type CRT displays. The CRTC belongs to the HMCS6800 LSI Family and has full compatibility with MPU in both data lines and control lines. Its primary function is to generate timing signal which is necessary for raster scan type CRT display according to the specification programmed by MPU. The CRTC is also designed as a programmable controller, so applicable to wide-range CRT display from small low-functioning character display up to raster type full graphic display as well as large high-functioning limited graphic display.

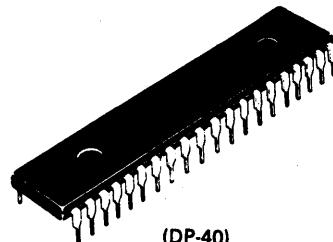
### ■ FEATURES

- Number of Displayed Characters on the Screen, Vertical Dot Format of One Character, Horizontal and Vertical Sync Signal, Display Timing Signal are Programmable
- 3.7 MHz High Speed Display Operation
- Line Buffer-less Refreshing
- 14-bit Refresh Memory Address Output (16k Words max. Access)
- Programmable Interlace/Non-interlace Scan Mode
- Built-in Cursor Control Function
- Programmable Cursor Height and its Blink
- Built-in Light Pen Detection Function
- Paging and Scrolling Capability
- TTL Compatible
- Single +5V Power Supply
- Upward compatible with MC6845

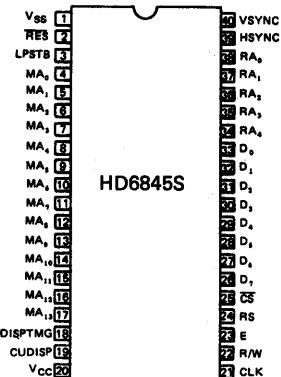
### ■ SYSTEM BLOCK DIAGRAM



HD6845SP, HD68A45SP, HD68B45SP



### ■ PIN ARRANGEMENT



(Top View)

### ■ ORDERING INFORMATION

CRTC	Bus Timing	CRT Display Timing
HD6845SP	1.0 MHz	
HD68A45SP	1.5 MHz	
HD68B45SP	2.0 MHz	3.7 MHz max.

**■ ABSOLUTE MAXIMUM RATINGS**

Item	Symbol	Value	Unit
Supply Voltage	$V_{CC}$ *	-0.3 ~ +7.0	V
Input Voltage	$V_{in}$ *	-0.3 ~ +7.0	V
Operating Temperature	$T_{opr}$	-20 ~ +75	°C
Storage Temperature	$T_{stg}$	-55 ~ +150	°C

\* With respect to  $V_{SS}$  (SYSTEM GND)

[NOTE] 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.

**■ RECOMMENDED OPERATING CONDITIONS**

Item	Symbol	min	typ	max	Unit
Supply Voltage	$V_{CC}$ *	4.75	5	5.25	V
Input Voltage	$V_{IL}$ *	-0.3	—	0.8	V
	$V_{IH}$ *	2.0	—	$V_{CC}$	V
Operating Temperature	$T_{opr}$	-20	25	75	°C

\* With respect to  $V_{SS}$  (SYSTEM GND)

**■ ELECTRICAL CHARACTERISTICS**

• DC CHARACTERISTICS ( $V_{CC} = 5V \pm 5\%$ ,  $V_{SS} = 0V$ ,  $T_a = -20 \sim +75^\circ C$ , unless otherwise noted.)

Item	Symbol	Test Condition		min	typ	max	Unit
Input "High" Voltage	$V_{IH}$			2.0	—	$V_{CC}$	V
Input "Low" Voltage	$V_{IL}$			-0.3	—	0.8	V
Input Leakage Current	$I_{in}$	$V_{in} = 0 \sim 5.25V$ (Except D <sub>0</sub> ~D <sub>7</sub> )		-2.5	—	2.5	μA
Three-State Input Current (off-state)	$I_{TSI}$	$V_{in} = 0.4 \sim 2.4V$ $V_{CC} = 5.25V$ (D <sub>0</sub> ~D <sub>7</sub> )		-10	—	10	μA
Output "High" Voltage	$V_{OH}$	$I_{LOAD} = -205 \mu A$ (D <sub>0</sub> ~D <sub>7</sub> )		2.4	—	—	V
Output "Low" Voltage	$V_{OL}$	$I_{LOAD} = -100 \mu A$ (Other Outputs)			—	—	V
Output "High" Current		$I_{LOAD} = 1.6 mA$		—	—	0.4	V
Input Capacitance	$C_{in}$	$V_{in} = 0$	D <sub>0</sub> ~ D <sub>7</sub>	—	—	12.5	pF
		T <sub>a</sub> = 25°C f = 1.0 MHz	Other Inputs	—	—	10.0	pF
Output Capacitance	$C_{out}$	$V_{in} = 0V$ , T <sub>a</sub> = 25°C, f = 1.0 MHz		—	—	10.0	pF
Power Dissipation	$P_D$			—	600	1000	mW

## HD6845S, HD68A45S, HD68B45S

- AC CHARACTERISTICS ( $V_{CC} = 5V \pm 5\%$ ,  $V_{SS} = 0V$ ,  $T_a = -20\sim+75^\circ C$ , unless otherwise noted.)

### 1. TIMING OF CRTC SIGNAL

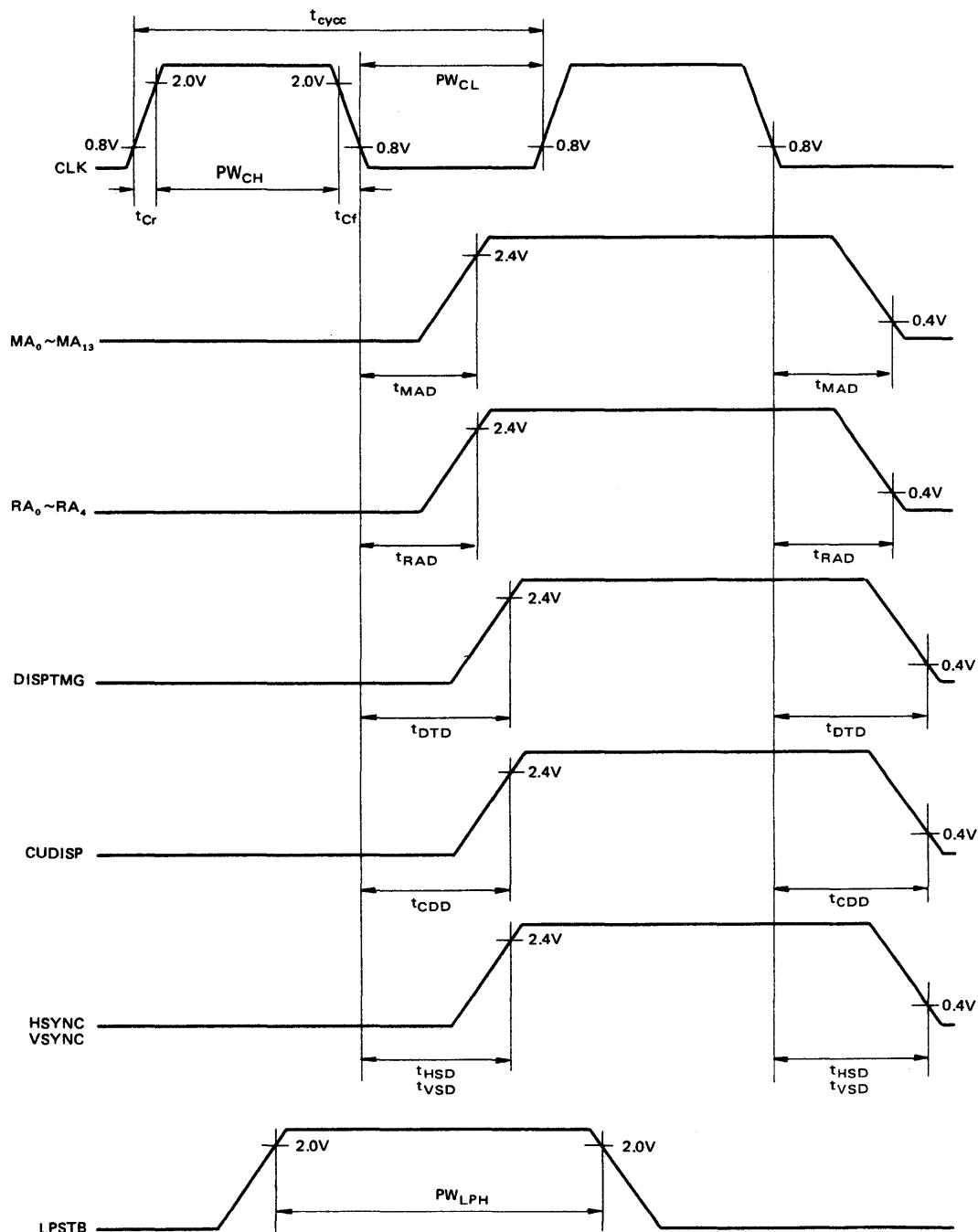
Item	Symbol	Test Condition	min	typ	max	Unit
Clock Cycle Time	$t_{cyc}$	Fig. 1	270	—	—	ns
Clock "High" Pulse Width	$PW_{CH}$		130	—	—	ns
Clock "Low" Pulse Width	$PW_{CL}$		130	—	—	ns
Rise and Fall Time for Clock Input	$t_{Cr}, t_{Cf}$		—	—	20	ns
Memory Address Delay Time	$t_{MAD}$		—	—	160	ns
Raster Address Delay Time	$t_{RAD}$		—	—	160	ns
DISPTMG Delay Time	$t_{DTD}$		—	—	250	ns
CUDISP Delay Time	$t_{CDD}$		—	—	250	ns
Horizontal Sync Delay Time	$t_{HSD}$		—	—	200	ns
Vertical Sync Delay Time	$t_{VSD}$		—	—	250	ns
Light Pen Strobe Pulse Width	$PW_{LPH}$	Fig. 2	60	—	—	ns
Light Pen Strobe	$t_{LPD1}$		—	—	70	ns
Uncertain Time of Acceptance	$t_{LPD2}$		—	—	0	ns

### 2. MPU READ TIMING

Item	Symbol	Test Condition	HD6845SP			HD68A45SP			HD68B45SP			Unit
			min	typ	max	min	typ	max	min	typ	max	
Enable Cycle Time	$t_{cycE}$	Fig. 3	1.0	—	—	0.666	—	—	0.5	—	—	μs
Enable "High" Pulse Width	$PW_{EH}$		0.45	—	—	0.280	—	—	0.22	—	—	μs
Enable "Low" Pulse Width	$PW_{EL}$		0.40	—	—	0.280	—	—	0.21	—	—	μs
Enable Rise and Fall Time	$t_{Er}, t_{Ef}$		—	—	25	—	—	25	—	—	25	ns
Address Set Up Time	$t_{AS}$		140	—	—	140	—	—	70	—	—	ns
Data Delay Time	$t_{DDR}$		—	—	320	—	—	220	—	—	180	ns
Data Hold Time	$t_H$		10	—	—	10	—	—	10	—	—	ns
Address Hold Time	$t_{AH}$		10	—	—	10	—	—	10	—	—	ns
Data Access Time	$t_{ACC}$		—	—	460	—	—	360	—	—	250	ns

### 3. MPU WRITE TIMING

Item	Symbol	Test Condition	HD6845SP			HD68A45SP			HD68B45SP			Unit
			min	typ	max	min	typ	max	min	typ	max	
Enable Cycle Time	$t_{cycE}$	Fig. 4	1.0	—	—	0.666	—	—	0.5	—	—	μs
Enable "High" Pulse Width	$PW_{EH}$		0.45	—	—	0.280	—	—	0.22	—	—	μs
Enable "Low" Pulse Width	$PW_{EL}$		0.40	—	—	0.280	—	—	0.21	—	—	μs
Enable Rise and Fall Time	$t_{Er}, t_{Ef}$		—	—	25	—	—	25	—	—	25	ns
Address Set Up Time	$t_{AS}$		140	—	—	140	—	—	70	—	—	ns
Data Set Up Time	$t_{DSW}$		195	—	—	80	—	—	60	—	—	ns
Data Hold Time	$t_H$		10	—	—	10	—	—	10	—	—	ns
Address Hold Time	$t_{AH}$		10	—	—	10	—	—	10	—	—	ns



This Figure shows the relation in time between CLK signal and each output signals. Output sequence is shown in Figs. 9 ~ 15.

Figure 1 Time Chart of the CRTC

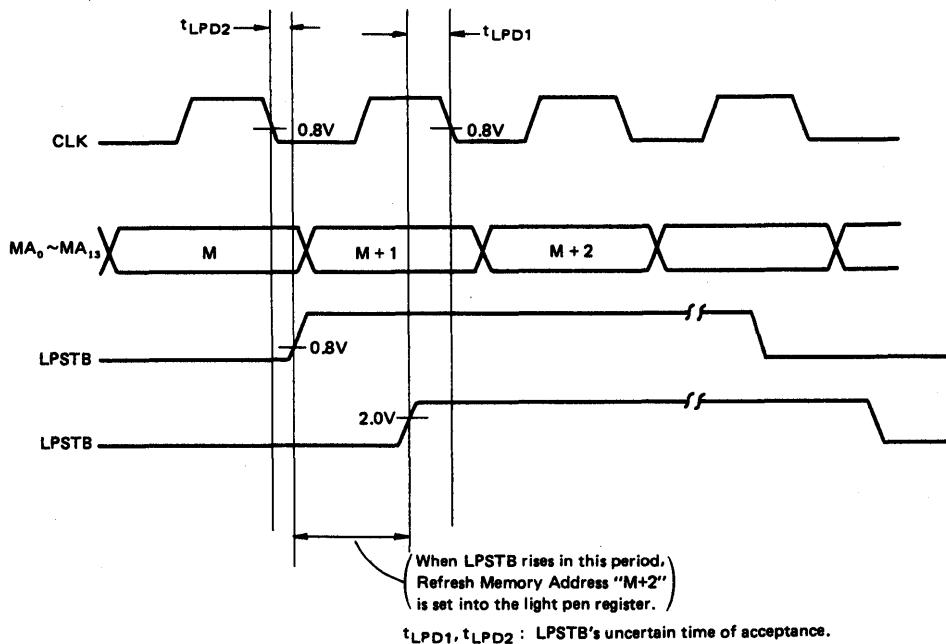


Figure 2 LPSTB Input Timing & Refresh Memory Address that is set into the light pen register.

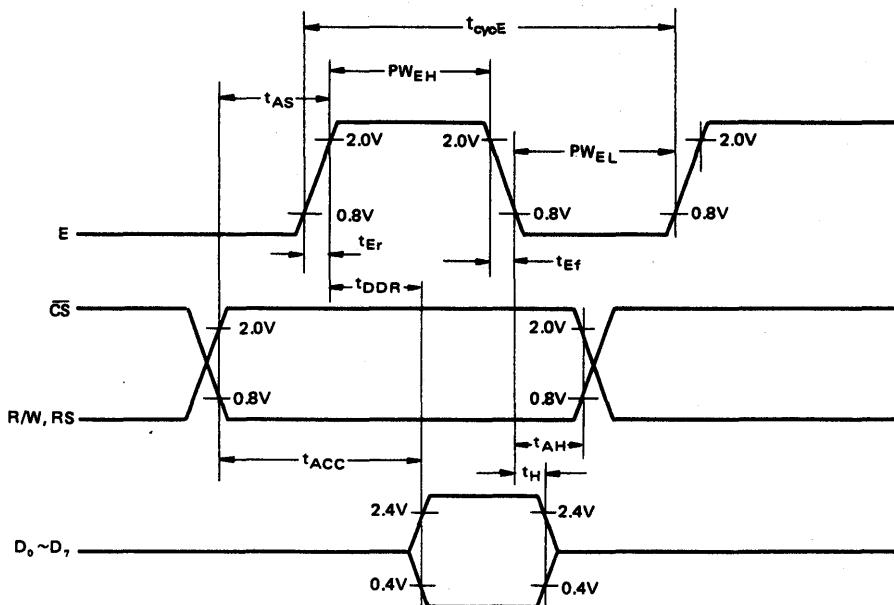


Figure 3 Read Sequence

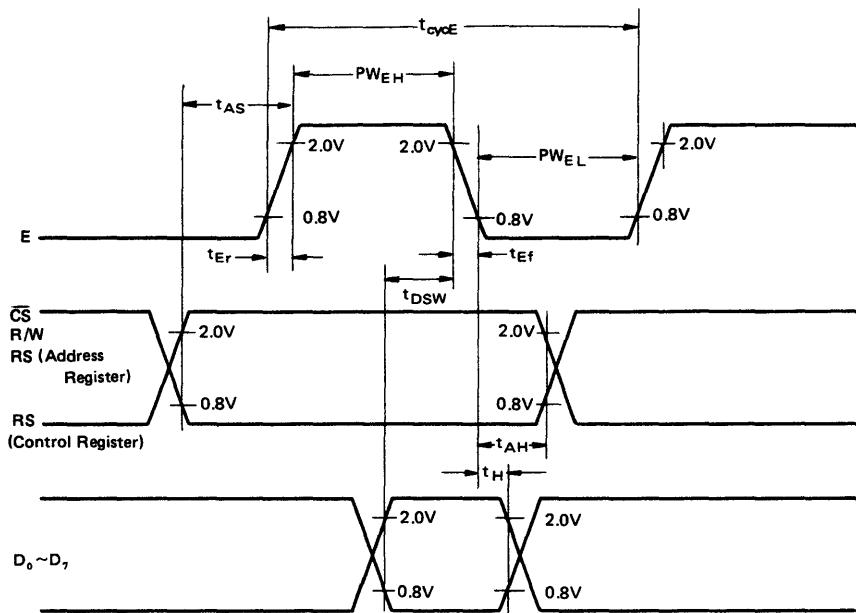


Figure 4 Write Sequence

## ■ SYSTEM DESCRIPTION

The CRTC is a LSI which is connected with MPU and CRT display device to control CRT display. The CRTC consists of internal register group, horizontal and vertical timing circuits, linear address generator, cursor control circuit, and light pen detection circuit. Horizontal and vertical timing circuit generate RA<sub>0</sub>~RA<sub>4</sub>, DISPTMG, HSYNC, and VSYNC. RA<sub>0</sub>~RA<sub>4</sub> are raster address signals and used as input signals for Character Generator. DISPTMG, HSYNC, and VSYNC signals are received by video control circuit. This horizontal and vertical timing circuit consists of internal counter and comparator circuit.

Linear address generator generates refresh memory address MA<sub>0</sub> ~MA<sub>13</sub> to be used for refreshing the screen. By these address signals, refresh memory is accessed periodically. As 14 refresh memory address signals are prepared, 16k words max are accessible. Moreover, the use of start address register enables paging and scrolling. Light pen detection circuit detects light pen position on the screen. When light pen strobe signal is received, light pen register memorizes linear address generated by linear address generator in order to memorize where light pen is on the screen. Cursor control circuit controls the position of cursor, its height, and its blink.

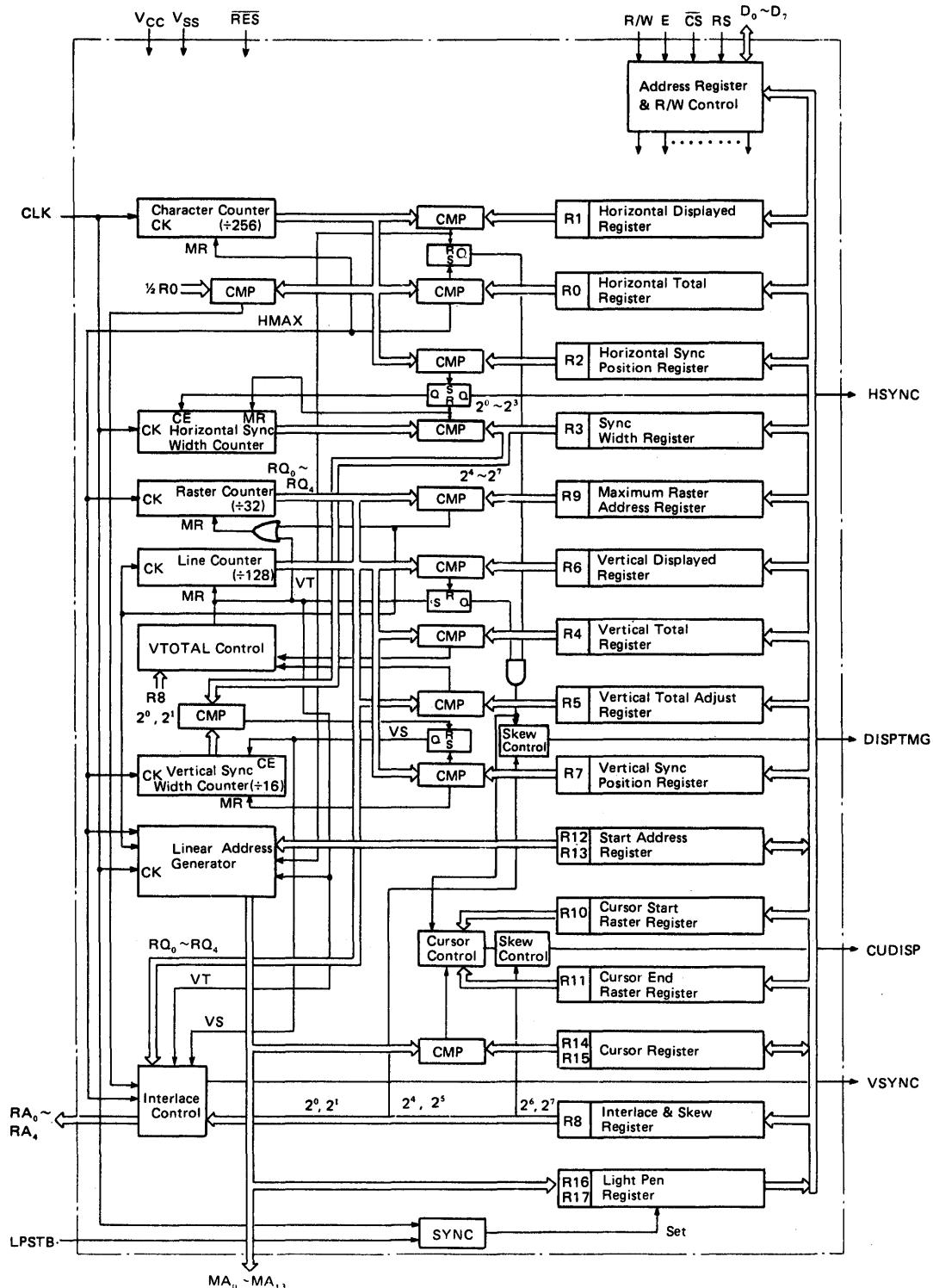


Figure 5 Internal Block Diagram of the CRTC

#### ■ FUNCTION OF SIGNAL LINE

The CRTC provides 13 interface signals to MPU and 25 interface signals to CRT display.

#### ● Interface Signals to MPU

##### Bi-directional Data Bus ( $D_0 \sim D_7$ )

Bi-directional data bus ( $D_0 \sim D_7$ ) are used for data transfer between the CRTC and MPU. The data bus outputs are 3-state buffers and remain in the high-impedance state except when MPU performs a CRTC read operation.

##### Read/Write (R/W)

R/W signal controls the direction of data transfer between the CRTC and MPU. When R/W is at "High" level, data of CRTC is transferred to MPU. When R/W is at "Low" level, data of MPU is transferred to CRTC.

##### Chip Select ( $\overline{CS}$ )

Chip Select signal ( $\overline{CS}$ ) is used to address the CRTC. When  $\overline{CS}$  is at "Low" level, it enables R/W operation to CRTC internal registers. Normally this signal is derived from decoded address signal of MPU under the condition that VMA signal of MPU is at "High" level.

##### Register Select (RS)

Register Select signal (RS) is used to select the address register and 18 control registers of the CRTC. When RS is at "Low" level, the address register is selected and when RS is at "High" level, control registers are selected. This signal is normally a derivative of the lowest bit (A0) of MPU address bus.

##### Enable (E)

Enable signal (E) is used as strobe signal in MPU R/W operation with the CRTC internal registers. This signal is normally a derivative of the HMCS6800 System  $\phi_2$  clock.

##### Reset ( $\overline{RES}$ )

Reset signal ( $\overline{RES}$ ) is an input signal used to reset the CRTC.

When  $\overline{RES}$  is at "Low" level, it forces the CRTC into the following status.

- 1) All the counters in the CRTC are cleared and the device stops the display operation.
- 2) All the outputs go down to "Low" level.
- 3) Control registers in the CRTC are not affected and remain unchanged.

This signal is different from other HMCS6800 family LSIs in the following functions and has restrictions for usage.

- 1)  $\overline{RES}$  signal has capability of reset function only when LPSTB is at "Low" level.
- 2) The CRTC starts the display operation immediately after  $\overline{RES}$  signal goes "High".

#### ● Interface Signals to CRT Display Device

##### Character Clock (CLK)

CLK is a standard clock input signal which defines character timing for the CRTC display operation. This signal is normally derived from the external high-speed dot timing logic.

##### Horizontal Sync (HSYNC)

HSYNC is an active "High" level signal which provides horizontal synchronization for display device.

##### Vertical Sync (VSYNC)

VSYNC is an active "High" level signal which provides vertical synchronization for display device.

##### Display Timing (DISPTMG)

DISPTMG is an active "High" level signal which defines the display period in horizontal and vertical raster scanning. It is necessary to enable video signal only when DISPTMG is at "High" level.

##### Refresh Memory Address ( $MA_0 \sim MA_{13}$ )

$MA_0 \sim MA_{13}$  are refresh memory address signals which are used to access to refresh memory in order to refresh the CRT screen periodically. These outputs enables 16k words max. refresh memory access. So, for instance, these are applicable up to 2000 characters/screen and 8-page system.

##### Raster Address ( $RA_0 \sim RA_4$ )

$RA_0 \sim RA_4$  are raster address signals which are used to select the raster of the character generator or graphic pattern generator etc.

##### Cursor Display (CUDISP)

CUDISP is an active "High" level video signal which is used to display the cursor on the CRT screen. This output is inhibited while DISPTMG is at "Low" level. Normally this output is mixed with video signal and provided to the CRT display device.

##### Light Pen Strobe (LPSTB)

LPSTB is an active "High" level input signal which accepts strobe pulse detected by the light pen and control circuit. When this signal is activated, the refresh memory address ( $MA_0 \sim MA_{13}$ ) which are shown in Fig. 2 are stored in the 14-bit light pen register. The stored refresh memory address need to be corrected in software, taking the delay time of the display device, light pen, and light pen control circuits into account.

## ■ REGISTER DESCRIPTION

Table 1 Internal Registers Assignment

CS	RS	Address Register 4 3 2 1 0	Register #	Register Name	Program Unit	READ	WRITE	Data Bit							
								7	6	5	4	3	2	1	0
1	X	X X X X X X			—	—	—								
0	0	X X X X X X	AR	Address Register	—	X	O								
0	1	0 0 0 0 0 0	R0	Horizontal Total *	Character	X	O								
0	1	0 0 0 0 0 1	R1	Horizontal Displayed	Character	X	O								
0	1	0 0 0 1 0	R2	Horizontal Sync * Position	Character	X	O								
0	1	0 0 0 1 1	R3	Sync Width	Vertical-Raster, Horizontal-Character	X	O	wv3	wv2	wv1	wv0	wh3	wh2	wh1	wh0
0	1	0 0 1 0 0	R4	Vertical Total *	Line	X	O								
0	1	0 0 1 0 1	R5	Vertical Total Adjust	Raster	X	O								
0	1	0 0 1 1 0	R6	Vertical Displayed	Line	X	O								
0	1	0 0 1 1 1	R7	Vertical Sync * Position	Line	X	O								
0	1	0 1 0 0 0	R8	Interlace & Skew	—	X	O	C1	C0	D1	D0			V	S
0	1	0 1 0 0 1	R9	Maximum Raster Address	Raster	X	O								
0	1	0 1 0 1 0	R10	Cursor Start Raster	Raster	X	O		B	P					
0	1	0 1 0 1 1	R11	Cursor End Raster	Raster	X	O								
0	1	0 1 1 0 0	R12	Start Address(H)	—	O	O								
0	1	0 1 1 0 1	R13	Start Address(L)	—	O	O								
0	1	0 1 1 1 0	R14	Cursor(H)	—	O	O								
0	1	0 1 1 1 1	R15	Cursor(L)	—	O	O								
0	1	1 0 0 0 0	R16	Light Pen(H)	—	O	X								
0	1	1 0 0 0 1	R17	Light Pen(L)	—	O	X								

- [NOTE]
- The Registers marked \*: (Written Value) = (Specified Value) - 1
  - Written Value of R9 is mentioned below.
    - Non-interlace Mode } (Written Value Nr) = (Specified Value) - 1
    - Interlace Sync Mode } (Written Value Nr) = (Specified Value) - 2
    - Interlace Sync & Video Mode (Written Value Nr) = (Specified Value) - 2
  - C0 and C1 specify skew of CUDISP output signal.
  - D0 and D1 specify skew of DISPTMG output signal.
  - When S is "1", V specifies video mode. S specifies the Interlace Sync Mode.
  - B specifies the cursor blink. P specifies the cursor blink period.
  - wv0~wv3 specify the pulse width of Vertical Sync Signal.
  - wh0~wh3 specify the pulse width of Horizontal Sync Signal.
  - R0 is ordinarily programmed to be odd number in interlace mode.
  - O; Yes, X; No

- **Address Register (AR)**

This is a 5-bit register used to select 18 internal control registers (R0~R17). Its contents are the address of one of 18 internal control registers. Programming the data from 18 to 31 produces no results. Access to R0~R17 requires, first of all, to write the address of corresponding control register into this register. When RS and CS are at "Low" level, this register is selected.

- **Horizontal Total Register (R0)**

This is a register used to program total number of horizontal characters per line including the retrace period. The data is 8-bit and its value should be programmed according to the specification of the CRT. When M is total number of characters, (M-1) shall be programmed to this register. When programming for interlace mode, M must be even.

- **Horizontal Displayed Register (R1)**

This is a register used to program the number of horizontal displayed characters per line. Data is 8-bit and any number that is smaller than that of horizontal total characters can be programmed.

- **Horizontal Sync Position Register (R2)**

This is a register used to program horizontal sync position as multiples of the character clock period. Data is 8-bit and any number that is lower than the horizontal total number can be programmed. When H is character number of horizontal Sync Position, (H-1) shall be programmed to this register. When programmed value of this register is increased, the display position on the CRT screen is shifted to the left. When programmed value is decreased, the position is shifted to the right. Therefore, the optimum horizontal position can be determined by this value.

- **Sync Width Register (R3)**

This is a register used to program the horizontal sync pulse width and the vertical sync pulse width. The horizontal sync pulse width is programmed in the lower 4-bit as multiples of the character clock period. "0" cannot be programmed. The vertical sync pulse width is programmed in higher 4-bit as multiples of the raster period. When "0" is programmed in higher 4-bit, 16 raster period (16H) is specified.

- **Vertical Total Register (R4)**

This is a register used to program total number of lines per frame including vertical retrace period. The data is within 7-bit and its value should be programmed according to the specification of the CRTC. When N is total number of lines, (N-1) shall be programmed to this register.

- **Vertical Total Adjust Register (R5)**

This is a register used to program the optimum number to adjust total number of rasters per field. This register enables to decide the number of vertical deflection frequency more strictly.

- **Vertical Displayed Register (R6)**

This is a register used to program the number of displayed character rows on the CRT screen. Data is 7-bit and any number that is smaller than that of vertical total characters can be programmed.

Table 2 Pulse Width of Vertical Sync Signal

VSW				Pulse Width
$2^7$	$2^6$	$2^5$	$2^4$	
0	0	0	0	16H
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

H; Raster period

Table 3 Pulse Width of Horizontal Sync Signal

HSW				Pulse Width
$2^3$	$2^2$	$2^1$	$2^0$	
0	0	0	0	-(Note)
0	0	0	1	1 CH
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

CH; Character clock period

(Note) HSW = "0" cannot be used.

- **Vertical Sync Position Register (R7)**

This is a register used to program the vertical sync position on the screen as multiples of the horizontal character line period. Data is 7-bit and any number that is equal to or less than vertical total characters can be programmed. When V is character number of vertical sync position, (V-1) shall be programmed to this register. When programmed value of this register is increased, the display position is shifted up. When programmed value is decreased, the position is shifted down. Therefore, the optimum vertical position may be determined by this value.

- **Interlace and Skew Register (R8)**

This is a register used to program raster scan mode and skew (delay) of CUDISP signal and DISPTMG signal.

**Interlace Mode Program Bit (V, S)**

Raster scan mode is programmed in the V, S bit.

Table 4 Interlace Mode ( $2^1, 2^0$ )

V	S	Raster Scan Mode
0	0	Non-interlace Mode
1	0	Interlace Sync Mode
0	1	Interlace Sync & Video Mode
1	1	

In the non-interlace mode, the rasters of even number field and odd number field are scanned duplicatedly. In the interlace sync mode, the rasters of odd number field are scanned in the middle of even number field. Then it is controlled to display the same character pattern in two fields. In the interlace sync and video mode, the raster scan method is the same as the interlace sync mode, but it is controlled to display different character pattern in two field.

**Skew Program Bit (C1, C0, D1, D0)**

These are used to program the skew (delay) of CUDISP signal and DISPTMG signal.

Skew of these two kinds of signals are programmed separately.

Table 5 DISPTMG Skew Bit ( $2^7, 2^6$ )

D1	D0	DISPTMG Signal
0	0	Non-skew
0	1	One-character skew
1	0	Two-character skew
1	1	Non-output

Table 6 Cursor Skew Bit ( $2^5, 2^4$ )

C1	C0	Cursor Skew
0	0	Non-skew
0	1	One-character skew
1	0	Two-character skew
1	1	Non-output

Skew function is used to delay the output timing of CUDISP and DISPTMG signals in LSI for the time to access refresh memory, character generator or pattern generator and to make the same phase with serial video signal.

- **Maximum Raster Address Register (R9)**

This is a register used to program maximum raster address within 5-bit. This register defines total number of rasters per character including space. This register is programmed as follows.

**Non-interlace Mode, Interlace Sync Mode**

When total number of rasters is RN, (RN-1) shall be programmed.

**Interlace Sync & Video Mode**

When total number of rasters is RN, (RN-2) shall be programmed.

This manual defines total number of rasters in non-interlace mode, interlace sync mode and interlace sync & video mode as follows:

**Non-interlace Mode**

0	Total Number of Rasters 5
1	Programmed Value Nr = 4
2	(The same as displayed)
3	total number of rasters
4	

**Raster Address**

**Interlace Sync Mode**

0	Total Number of Rasters 5
1	Programmed Value Nr = 4
2	1
3	2
4	3

In the interlace sync mode, total number of rasters in both the even and odd fields is ten. On programming, the half of it is defined as total number of rasters.

**Raster Address**

**Interlace Sync & Video Mode**

0	Total Number of Rasters 5
1	Programmed Value Nr = 3
2	3

Total number of rasters displayed in the even field and the odd field.

**Raster Address**

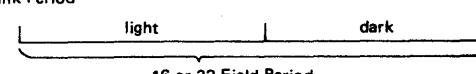
- **Cursor Start Raster Register (R10)**

This is a register used to program the cursor start raster address by lower 5-bit ( $2^0 \sim 2^4$ ) and the cursor display mode by higher 2-bit ( $2^5, 2^6$ ).

Table 7 Cursor Display Mode ( $2^6, 2^5$ )

B	P	Cursor Display Mode
0	0	Non-blink
0	1	Cursor Non-display
1	0	Blink, 16 Field Period
1	1	Blink, 32 Field Period

**Blink Period**



● **Cursor End Raster Register (R11)**

This is register used to program the cursor end raster address.

● **Start Address Register (R12, R13)**

These are used to program the first address of refresh memory to read out.

Paging and scrolling is easily performed using this register. This register can be read but the higher 2-bit ( $2^6, 2^7$ ) of R12 are always "0".

● **Cursor Register (R14, R15)**

These two read/write registers stores the cursor location. The higher 2-bit ( $2^6, 2^7$ ) of R14 are always "0".

● **Light Pen Register (R16, R17)**

These read only registers are used to catch the detection address of the light pen. The higher 2-bit ( $2^6, 2^7$ ) of R16 are always "0". Its value needs to be corrected by software because there is time delay from address output of the CRTC to signal input LPSTB pin of the CRTC in the process that raster is lit after address output and light pen detects it. Moreover, delay time shown in Fig. 2 needs to be taken into account.

**Restriction on Programming Internal Register**

- 1)  $0 < Nhd < Nht + 1 \leq 256$
- 2)  $0 < Nvd < Nvt + 1 \leq 128$
- 3)  $0 \leq Nhsp \leq Nht$
- 4)  $0 \leq Nvsp \leq Nvt^*$
- 5)  $0 \leq NCSTART \leq NCEND \leq Nr$  (Non-interlace, Interlace sync mode)
- 6)  $0 \leq NCSTART \leq NCEND \leq Nr + 1$  (Interlace sync & video mode)

6)  $2 \leq Nr \leq 30$

7)  $3 \leq Nht$  (Except non-interlace mode)

$5 \leq Nht$  (Non-interlace mode only)

\* In the interlace mode, pulse width is changed  $\pm\frac{1}{2}$  raster time when vertical sync signal extends over two fields.

**Notes for Use**

The method of directly using the value programmed in the internal register of LSI for controlling the CRT is adopted. Consequently, the display may flicker on the screen when the contents of the registers are changed from bus side asynchronously with the display operation.

**Cursor Register**

Writing into this register at frequent intervals for moving the cursor should be performed during horizontal and vertical retrace period.

**Start Address Register**

Writing into the start address register at frequent intervals for scrolling and paging should be performed during horizontal and vertical display period.

It is desirable to avoid programming other registers during display operation.

■ **OPERATION OF THE CRTC**

● **Time Chart of CRT Interface Signals**

The following example shows the display operation in which values of Table 8 are programmed to the CRTC internal registers. Fig. 6 shows the CRT screen format. Fig. 9 shows the time chart of signals output from the CRTC.

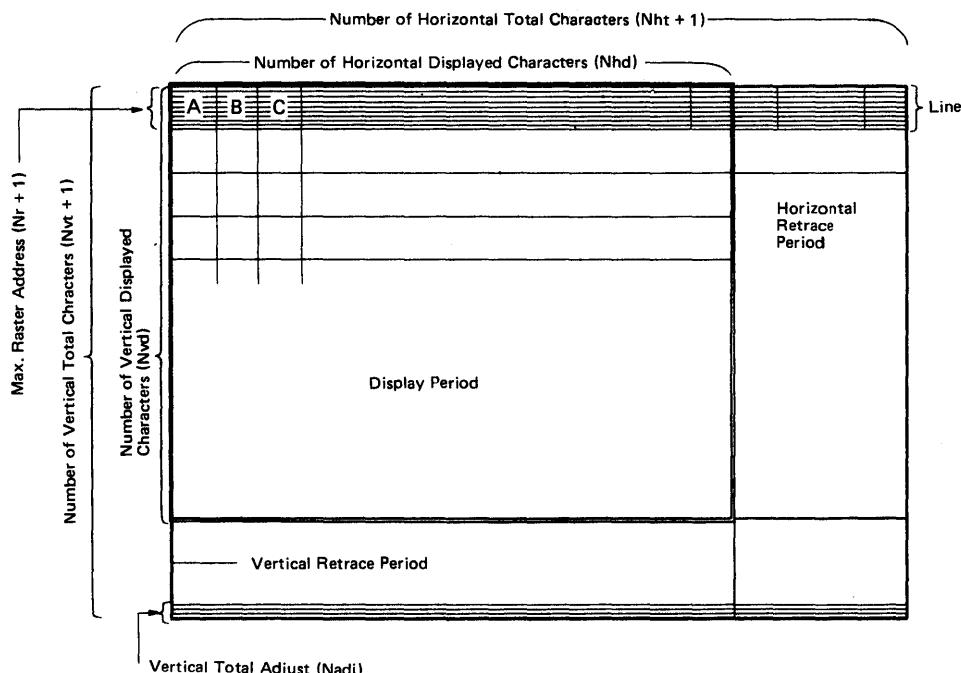


Figure 6 CRT screen Format

Table 8 Programmed Values into the Registers

Register	Register Name	Value	Register	Register Name	Value
R0	Horizontal Total	Nht	R9	Max. Raster Address	Nr
R1	Horizontal Displayed	Nhd	R10	Cursor Start Raster	
R2	Horizontal Sync Position	Nhsp	R11	Cursor End Raster	
R3	Sync Width	Nvsw, Nhsrw	R12	Start Address (H)	0
R4	Vertical Total	Nvt	R13	Start Address (L)	0
R5	Vertical Total Adjust	Nadj	R14	Cursor (H)	
R6	Vertical Displayed	Nvd	R15	Cursor (L)	
R7	Vertical Sync Position	Nvsp	R16	Light Pen (H)	
R8	Interlace & Skew		R17	Light Pen (L)	

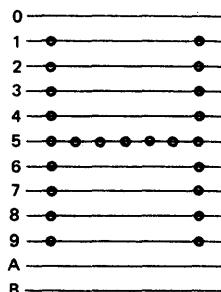
[NOTE] Nhd < Nht, Nvd < Nvt

The relation between values of Refresh Memory Address ( $MA_0 \sim MA_{13}$ ) and Raster Address ( $RA_0 \sim RA_4$ ) and the display position on the screen is shown in Fig. 15. Fig. 15 shows the case where the value of Start Address is 0.

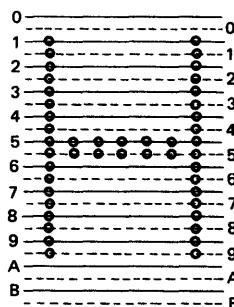
#### • Interlace Control

Fig. 7 shows an example where the same character is displayed in the non-interlace mode, interlace sync mode, and video mode.

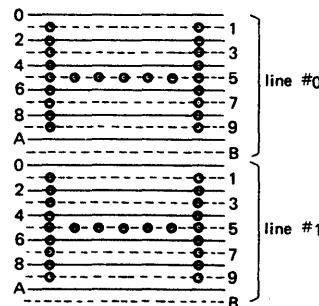
#### Non-interlace Mode Display



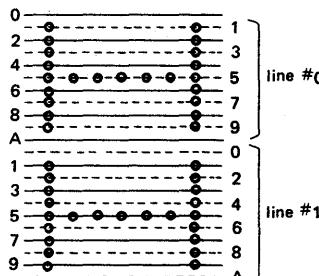
Non-interlace Mode



Interlace Sync Mode



Interlace Sync & Video Mode  
(Total number of rasters in a line is even.)



Interlace Sync & Video Mode  
(Total number of rasters in a line is odd.)

Figure 7 Example of Raster Scan Display

### Interlace Sync & Video Mode Display

In interlace sync & video mode, the output raster address when the number of rasters is even is different from that when the number of rasters is odd.

Table 9 The Output of Raster Address in  
Interlace Sync & Video Mode

Total Number of Rasters in a Line	Field	Even Field	Odd Field
Even	Even Address	Odd Address	
Odd	Even Line*	Even Address	Odd Address
	Odd Line*	Odd Address	Even Address

\* Internal line address begins from 0.

1) Total number of rasters in a line is even;

When number of rasters is programmed to be even, even raster address is output in the even field and odd raster address is output in the odd field.

2) Total number of rasters in a line is odd;

When total number of rasters is programmed to be odd, odd and even addresses are reversed according to the odd and even lines in each field. In this case, the difference in numbers of dots displayed between even field and odd field is usually smaller the case of 1). Then interlace can be displayed more stably.

[NOTE] The wide disparity of dots between number of dots between even field and odd field influences beam current of CRT. CRT, which has a stable high-voltage part, can make interlace display normal. On the contrary, CRT, which has unstable high-voltage part, moves deflection angle of beam current and also dots displayed in the even and odd fields may be shifted. Characters appears distorting on a border of the screen. So 2) programming has an effect to decrease such evil influences as mentioned above. Fig. 12 shows fine chart in each mode when interlace is performed.

### • Cursor Control

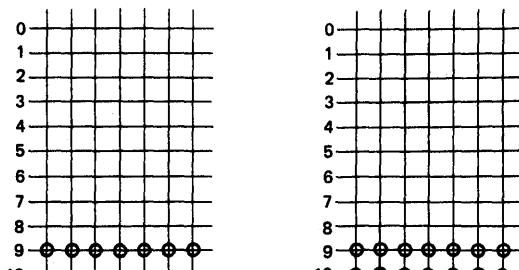
Fig. 8 shows the display patterns where each value is programmed to the cursor start raster register and the cursor end raster register. Programmed values to the cursor start raster register and the cursor end raster register need to be under the following condition.

Cursor Start Raster Register  $\leq$  Cursor End Raster Register  $\leq$  Maximum Raster Address Register.

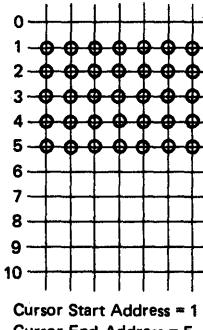
Time chart of CUDISP output signal is shown in Fig. 13 and Fig. 14.

### ■ INTERFACE TO DISPLAY CONTROL UNIT

Fig. 16 shows the interface between the CRTC and display control unit. Display control unit is mainly composed of Refresh Memory, Character Generator, and Video Control circuit. For refresh memory, 14 Memory Address line (0~16383) max are provided and for character generator, 5 Raster Address line (0~31) max are provided. For video control circuit, DISPTMG, CUDISP, HSYNC, and VSYNC signals are sent out. DISPTMG signal is used to control the blank period of video signal. CUDISP signal is used as video signal to display the cursor on the CRT screen. Moreover, HSYNC and VSYNC signals are used as drive signals respectively for CRT horizontal and vertical deflection circuits.



Cursor Start Address = 9  
Cursor End Address = 9  
Cursor Start Address = 9  
Cursor End Address = 10



Cursor Start Address = 1  
Cursor End Address = 5

Figure 8 Cursor Control

Outputs from video control circuit, (video signals and sync signals) are provided to CRT display unit to control the deflection and brightness of CRT, thus characters are displayed on the screen.

Fig. 17 shows detailed block diagram of display control unit. This shows how to use CUDISP and DISPTMG signals. CUDISP and DISPTMG signals should be used being latched at least one time at external flip-flop F1 and F2. Flip-flop F1 and F2 function to make one-character delay time so as to synchronize them with video signal from parallel-serial converter. High-speed D type flip-flop as TTL is used for this purpose. After being delayed at F1 and F2 DISPTMG signal is AND-ed with character video signal, and CUDISP signal is OR-ed with output from AND gate. By using this circuitry, blanking of horizontal and vertical retrace time is controlled. And cursor video is mixed with character video signal.

Fig. 17 shows the example in the case that both refresh memory and CG can be accessed for horizontal one character time. Time chart for this case is shown in Fig. 20. This method is used when a few character needed to be displayed in horizontal direction on the screen.

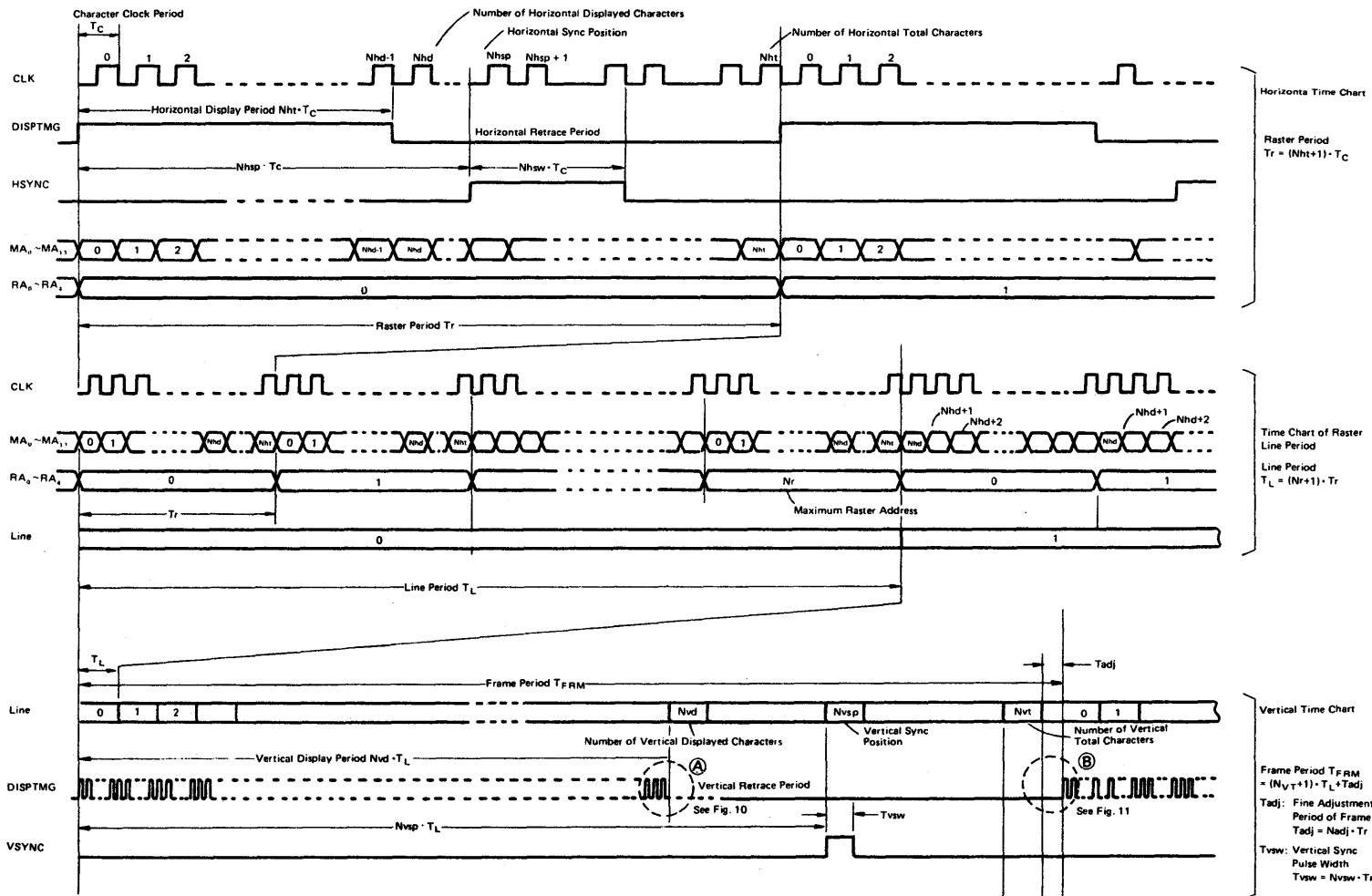


Figure 9 CRTC Time Chart

( Output waveform of horizontal & vertical display  
in the case where values shown in Table 8 are  
Programmed to each register. )

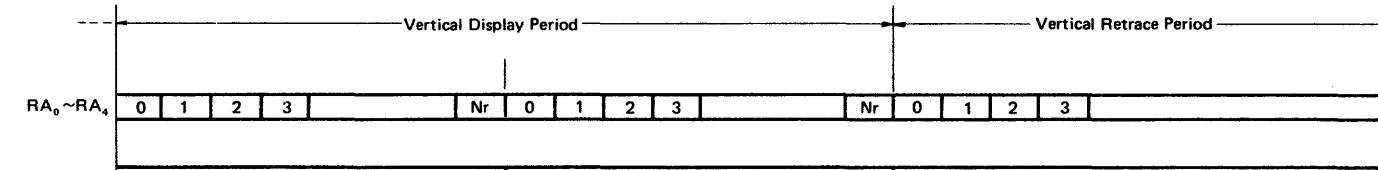


Figure 10 Switching from Vertical Display Period over to Vertical Retrace Period (Expansion of Fig. 9-Ⓐ)

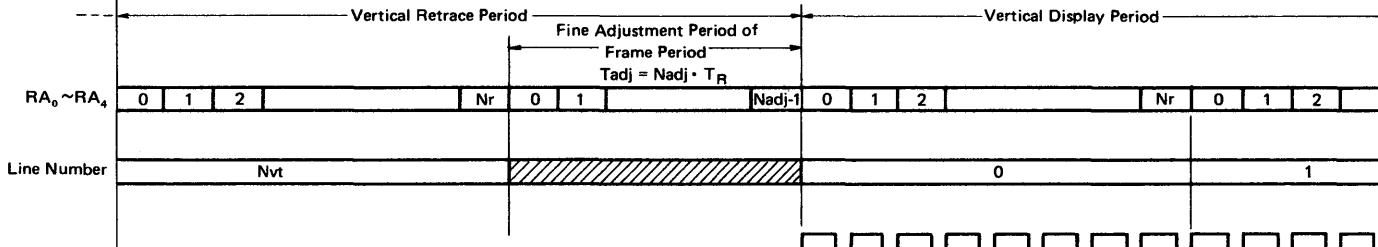


Figure 11 Fine Adjustment Period of Frame in Vertical Display  
(Expansion of Fig. 9-Ⓑ)

## HD6845S, HD68A45S, HD68B45S

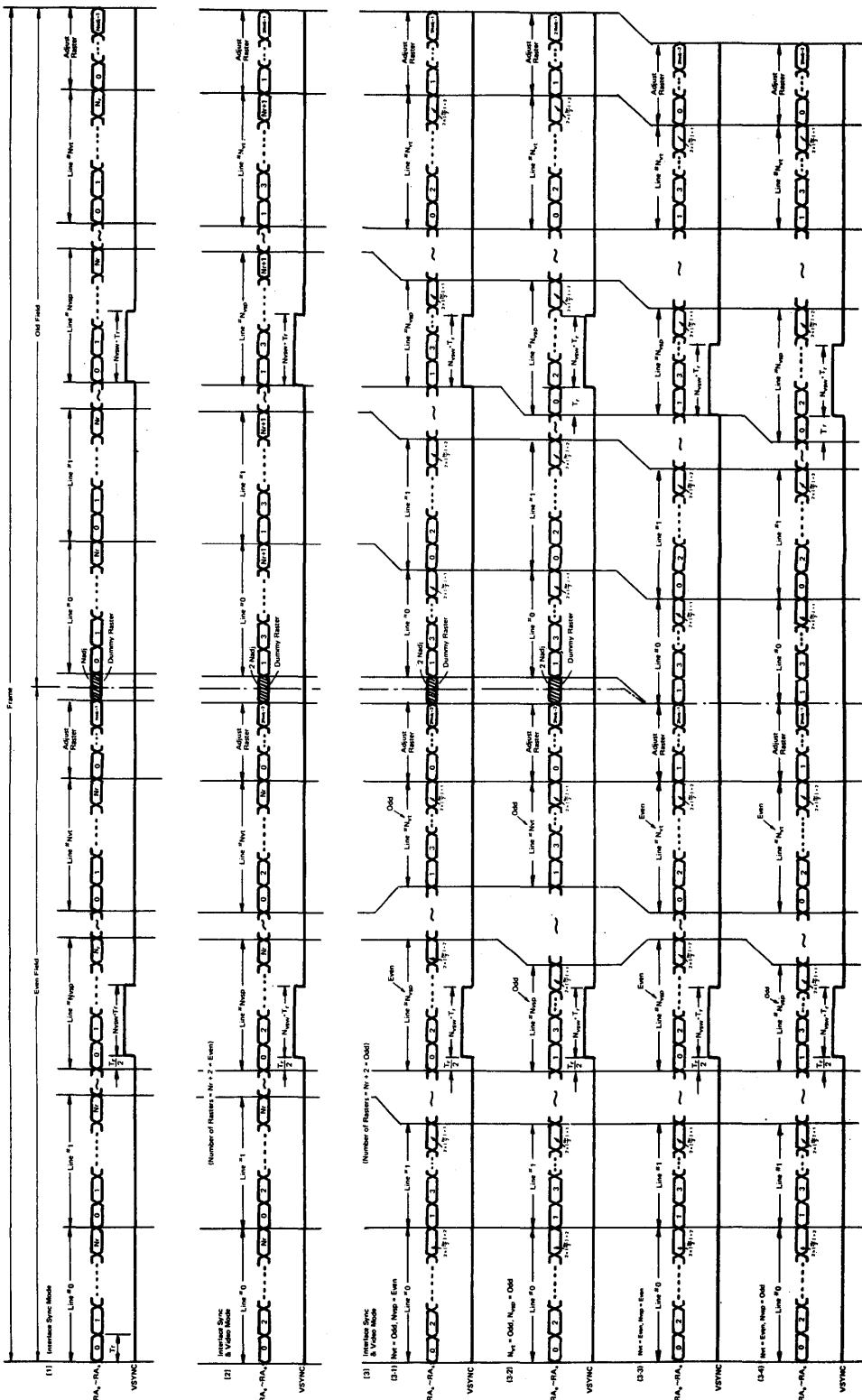


Figure 12 Interlace Control

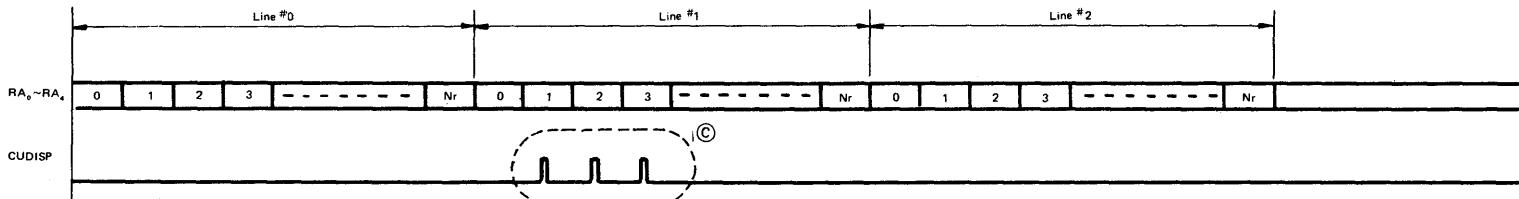
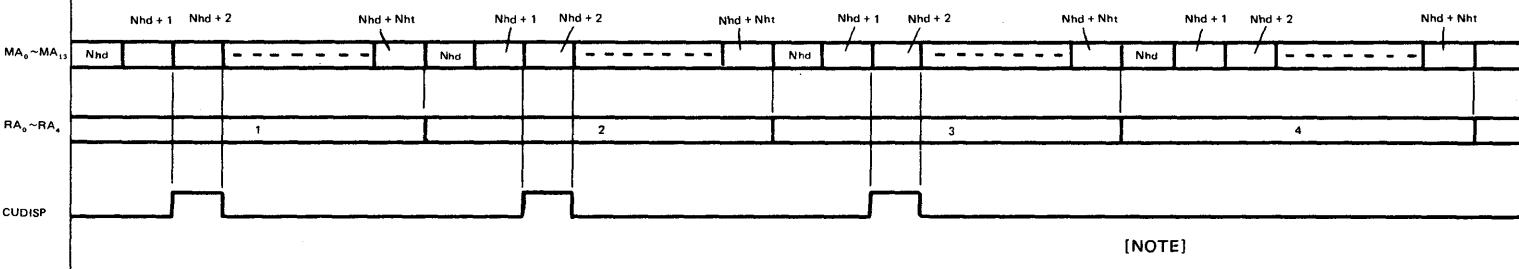


Figure 13 Relation between Line • Raster and CUDISP



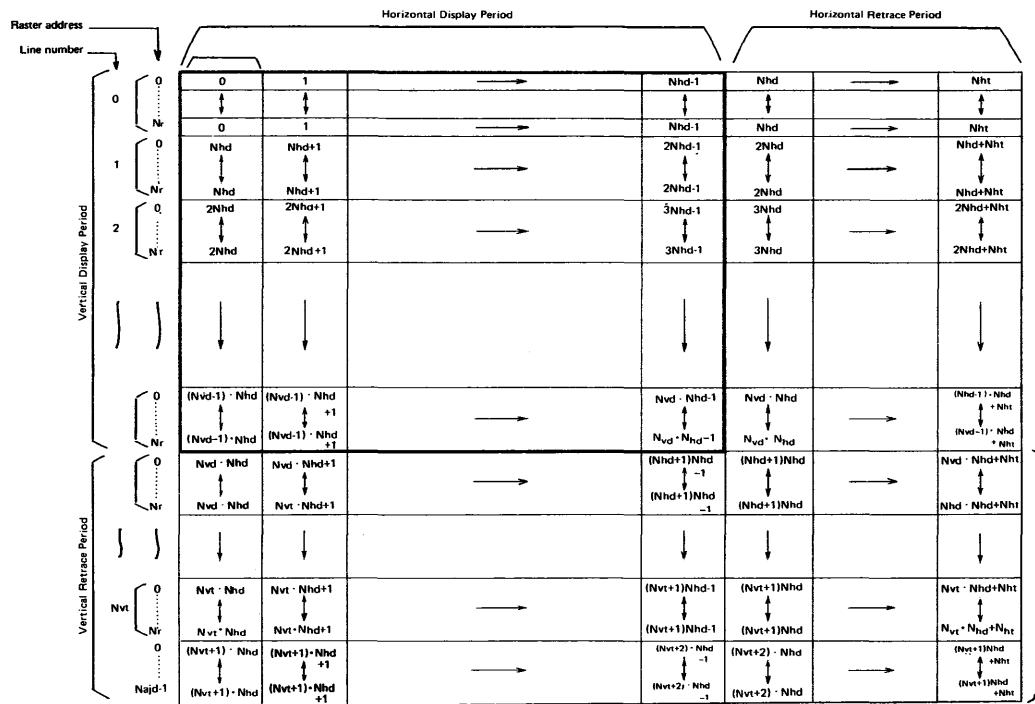
## [NOTE]

Cursor register = Nhd+2  
 Cursor Start  
 Raster Register = 1  
 Cursor End  
 Raster Register = 3

are Programmed in cursor display mode.

In blink mode, it is changed into display or non-display mode when field period is 16 or 32-time period.

Figure 14 CUDISP Output Timing (Exapnsion of Fig. 13- (C))



Valid refresh memory address (0~Nvd·Nrd-1) are shown within the thick-line square. Refresh memory address are provided even during horizontal and vertical retrace period. This is an example in the case where the programmed value of start address register is 0.

Figure 15 Refresh Memory Address (MA<sub>0</sub>~MA<sub>13</sub>)

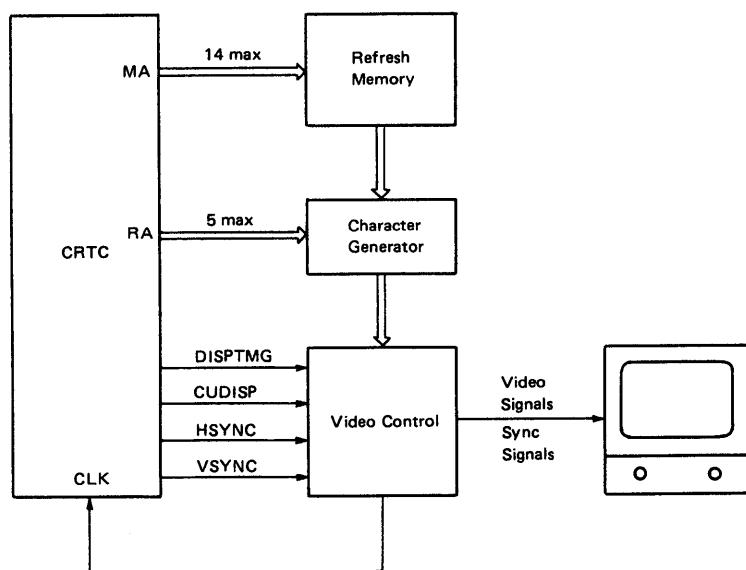


Figure 16 Interface to Display Control Unit

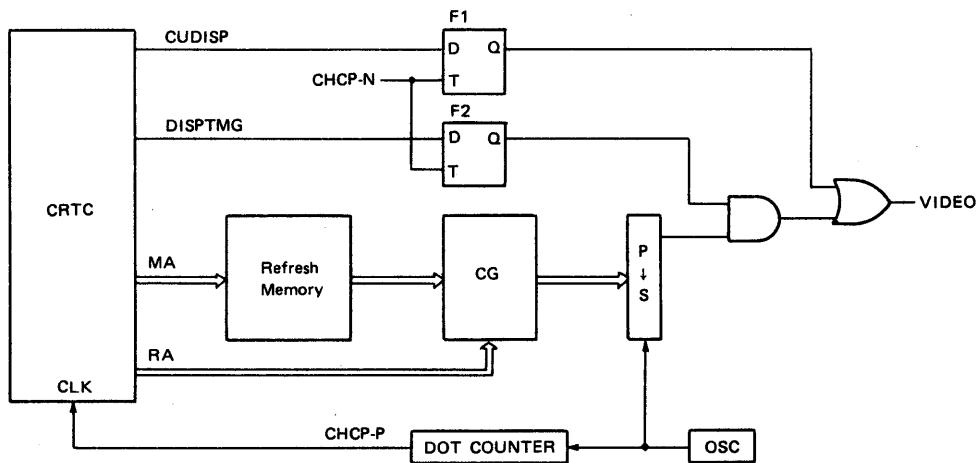


Figure 17 Display Control Unit (1)

When many characters are displayed in horizontal direction on the screen, and horizontal one-character time is so short that both refresh memory and CG cannot be accessed, the circuitry shown in Fig. 18 should be used. In this case refresh memory output shall be latched and CG shall be accessed at the next cycle. The time chart in this case is shown in Fig. 21. CUDISP and DISPTMG signals should be provided after being delayed by one-character time by using skew bit of interlace & skew register (R8). Moreover, when there are some

troubles about delay time of MA during horizontal one-character time on high-speed display operation, system shown in Fig.19 is adopted. The time chart in this case is shown in Fig.22. Character video signal is delayed for two-character time because each MA outputs and refresh memory outputs are latched, and they are made to be in phase with CUDISP and DISPTMG signals by delaying for two-character time. Table 10 shows the circuitry selection standard of display units.

Table 10 Circuitry Standard of Display Control Unit

Case	Relation among $t_{CH}$ , RM and CG	Block Diagram	Interlace & Skew Register Bit Programming			
			C1	C0	D1	D0
1	$t_{CH} > RM \text{ Access} + CG \text{ Access} + t_{MAD}$	Fig. 17	0	0	0	0
2	$RM \text{ Access} + CG \text{ Access} + t_{MAD} \geq t_{CH} > RM \text{ Access} + t_{MAD}$	Fig. 18	0	1	0	1
3	$RM \text{ Access} + t_{MAD} \geq t_{CH} > RM \text{ Access}$	Fig. 19	1	0	1	0

$t_{CH}$  : CHCP Period;  $t_{MAD}$  : MA Delay

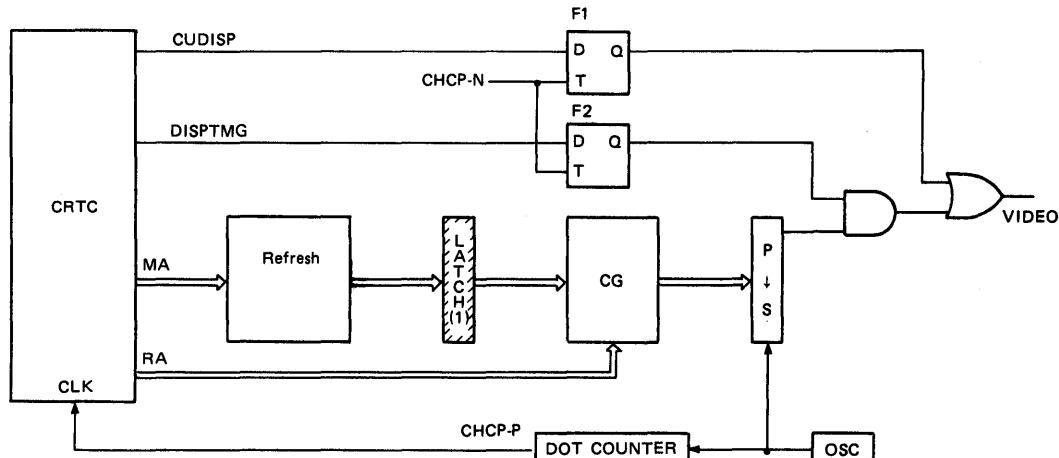


Figure 18 Display Control Unit (2)

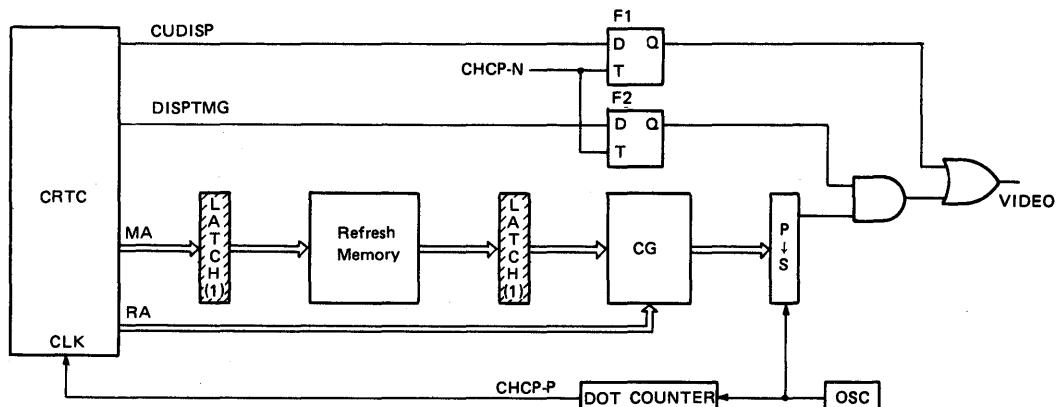


Figure 19 Display Control Unit (For high-speed display operation) (3)

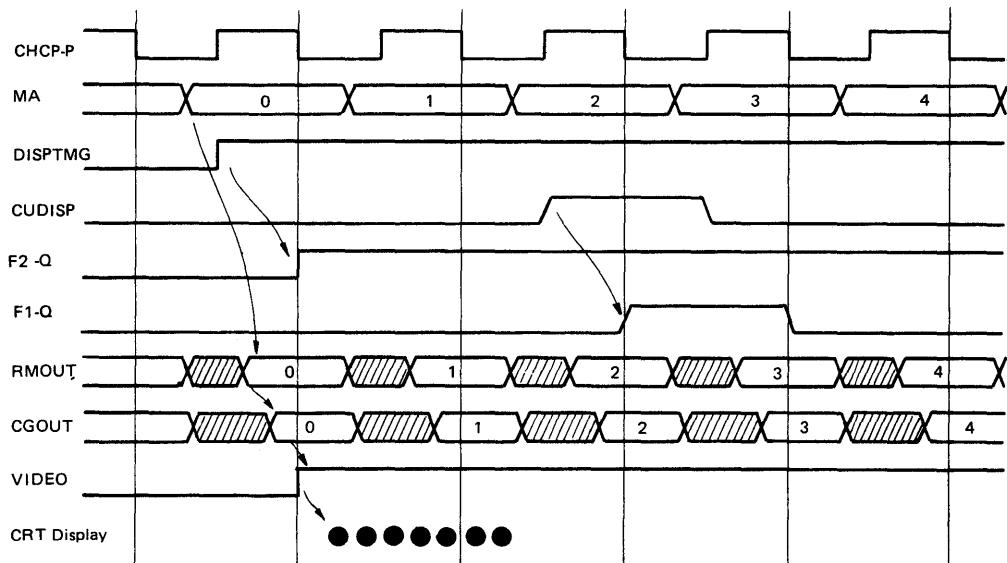


Figure 20 Time Chart of Display Control Unit (1)

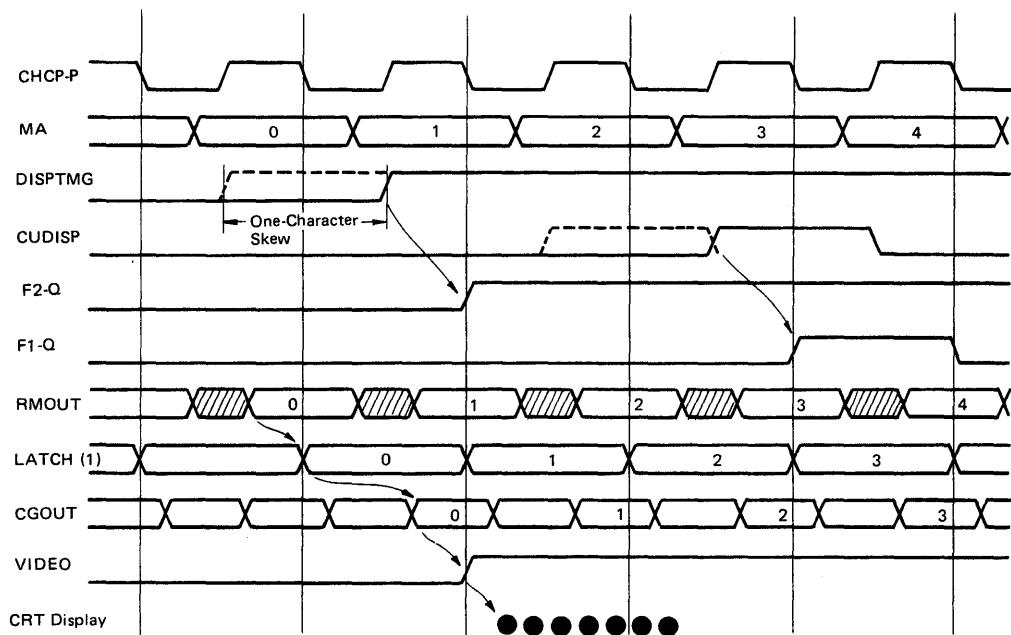


Figure 21 Time Chart of Display Control Unit (2)

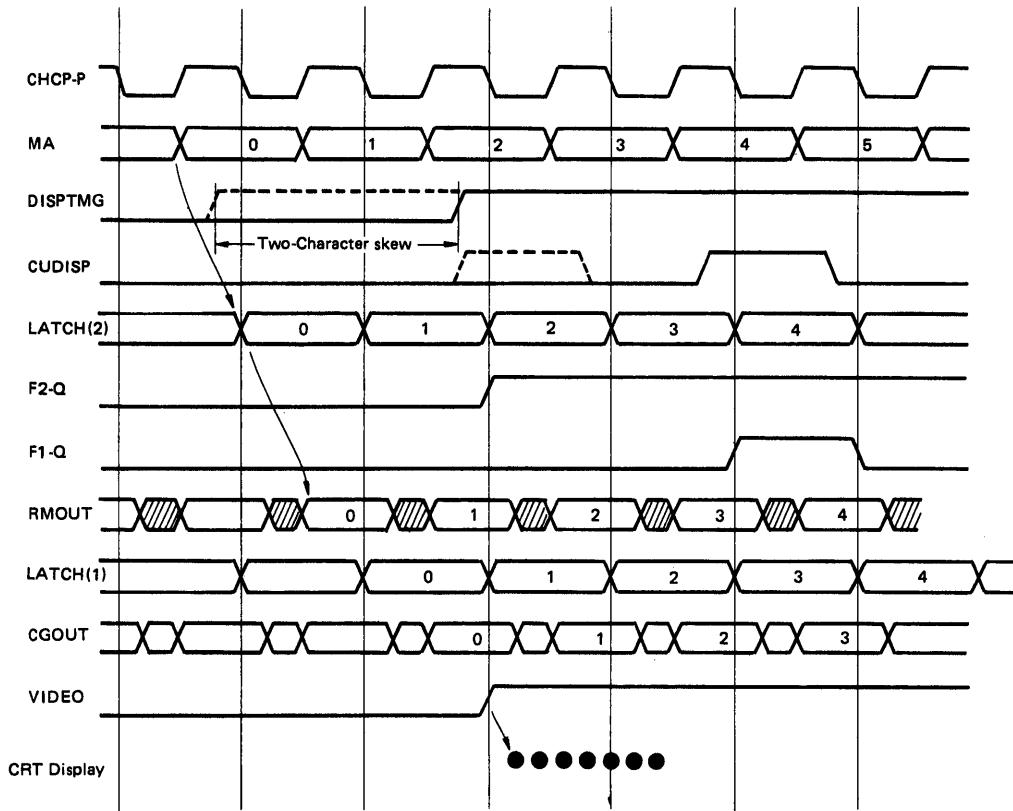


Figure 23 Time Chart of Display Unit (3)

#### ■ HOW TO DECIDE PARAMETERS SET ON THE CRTC

- How to Decide Parameters Based on Specification of CRT Display Unit (Monitor)

##### Number of Horizontal Total Characters

Horizontal deflection frequency  $f_h$  is given by specification of CRT display unit. Number of horizontal total characters is determined by the following equation.

$$f_h = \frac{1}{t_c (Nht + 1)}$$

where,

$t_c$  : Cycle Time of CLK (Character Clock)

$Nht$  : Programmed Value of Horizontal Total Register (R0)

##### Number of Vertical Total Characters

Vertical deflection frequency is given by specification of CRT display unit. Number of vertical Total characters is determined by the following equation.

- 1) Non-interlace Mode

$$Rt = (Nvt + 1)(Nr + 1) + Nadj$$

- 2) Interlace Sync Mode

$$Rt = (Nvt + 1)(Nr + 1) + Nadj + 0.5$$

- 3) Interlace Sync & Video Mode

$$Rt = \frac{(Nvt + 1)(Nr + 2) + 2Nadj}{2} \quad \dots \quad (a)$$

$$Rt = \frac{(Nvt + 1)(Nr + 2) + 2Nadj + 1}{2} \quad \dots \quad (b)$$

(a) is applied when both total numbers of vertical characters ( $Nvt + 1$ ) and that of rasters in a line ( $Nr + 2$ ) are odd.

(b) is applied when total number of rasters ( $Nr + 2$ ) is even, or when ( $Nr + 2$ ) is odd and total number of vertical characters ( $Nvt + 1$ ) is even.

where,

$Rt$  : Number of Total Rasters per frame  
(Including retrace period)

$Nvt$  : Programmed Value of Vertical Total Register (R4)

$Nr$  : Programmed Value of Maximum Raster Address Register (R9)

$Nadj$  : Programmed Value of Vertical Total Adjust Register (R5)

##### Horizontal Sync Pulse Width

Horizontal sync pulse width is programmed to low order 4-bit of horizontal sync width register (R3) in unit of horizontal character time. Programmed value can be selected within from 1 to 15.

### Horizontal Sync Position

As shown in Fig. 24, horizontal sync position is normally selected to be in the middle of horizontal blank period. But there are some cases where its optimum sync position is not located in the middle of horizontal blank period according to specification of CRT. Therefore, horizontal sync position should be determined by specification of CRT. Horizontal sync pulse position is programmed in unit of horizontal character time.

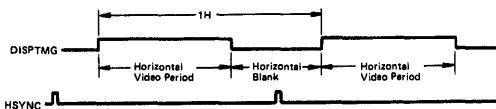


Figure 24 Time Chart of HSYNC

### Vertical Sync Pulse Width

Vertical Sync Pulse Width is programmed to high order 4-bit of vertical sync pulse width register (R3) in unit of raster period. Programmed value can be selected within from 1 to 16.

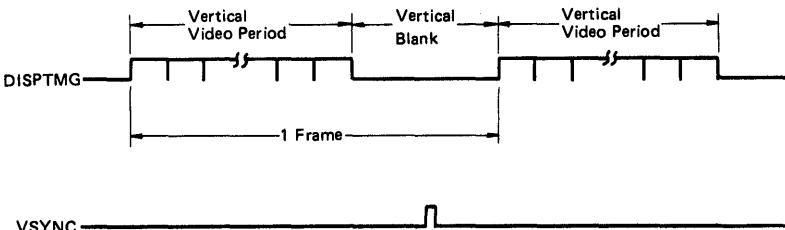


Figure 25 Time Chart of VSYNC

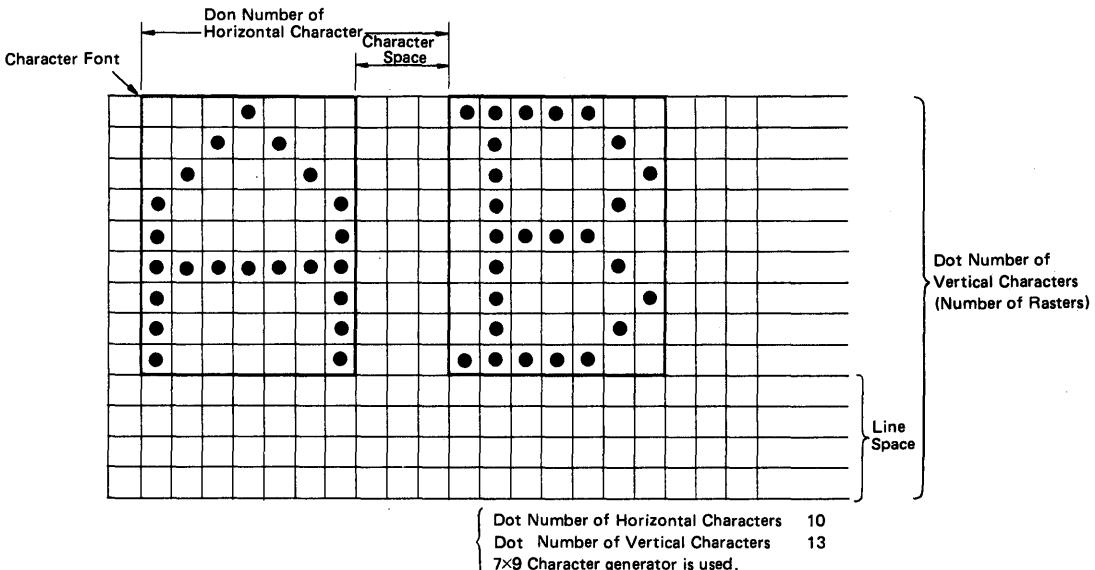


Figure 26 Dot Number of Horizontal and Vertical Characters

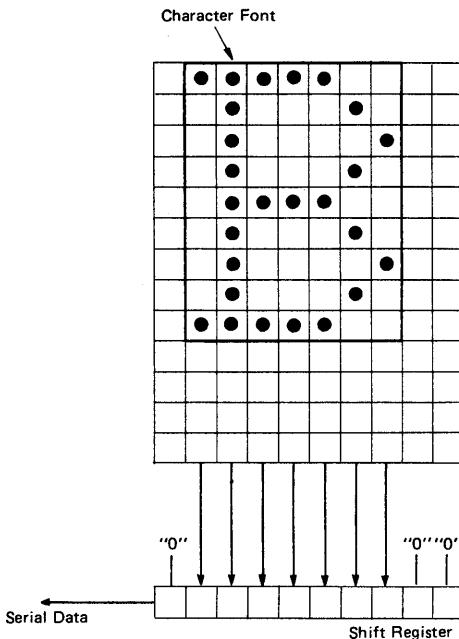


Figure 27 How to Make Character Space

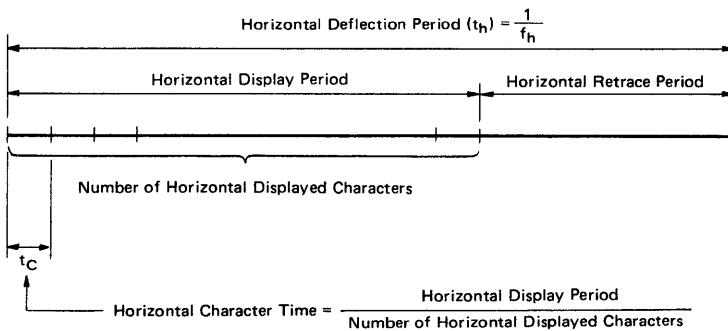


Figure 28 Number of Horizontal Displayed Characters

value of R9, dot number of characters (vertical) is ( $N_r + 1$ ).

#### Number of Horizontal Displayed Characters

Number of horizontal displayed characters is programmed to horizontal displayed register (R1) of the CRTC. Programmed value is based on screen format. Horizontal display period, which is given by specification of horizontal deflection frequency and horizontal retrace period of CRT display unit, determines horizontal character time, being divided by number of horizontal displayed characters. Moreover, its cycle time and access time which are necessary for CRT display system are determined by horizontal character time.

#### Number of Vertical Displayed Characters

Number of vertical displayed characters is programmed to vertical displayed register (R6). Programmed value is based on screen format. As specification of vertical deflection frequency of CRT determines number of total rasters (Rt) including verti-

cal retrace period and the relation between number of vertical displayed character and total number of rasters on a screen is as mentioned above, CRT which is suitable for desired screen format should be selected.

For optimum screen format, it is necessary to adjust number of rasters per line, number of vertical displayed characters, and total adjust raster (Nadj) within specification of vertical deflection frequency.

#### Scan Mode

The CRTC can program three-scan modes shown in Table 11 to interlace mode register (R8). An example of character display in each scan mode is shown in Fig. 7.

Table 11 Program of Scan Mode

$2^1$	$2^0$	Scan Mode	Main Usage
0	0	Non-interlace	Normal Display of Characters & Figures
0	1	Interlace Sync	Fine Display of Characters & Figures
1	1	Interlace Sync & Video	Display of Many Characters & Figures Without Using High-resolution CRT

[NOTE] In the interlace mode, the number of times per sec. in raster scanning on one spot on the screen is half as many as that in non-interlace mode. Therefore, when persistence of luminescence is short, flickering may happen. It is necessary to select optimum scan mode for the system, taking characteristics of CRT, raster scan speed, and number of displayed characters and figures into account.

#### Cursor Display Method

Cursor start raster register and cursor end raster register (R10, R11) enable programming the display modes shown in Table 7 and display patterns shown in Fig. 8. Therefore, it is possible to change the method of cursor display dynamically according to the system conditions as well as to realize the cursor display that meets the system requirements.

#### Start Address

Start address resistors (R12, R13) give an offset to the address of refresh memory to read out. This enables paging and scrolling easily.

#### Cursor Register

Cursor registers (R14, R15) enable programming the cursor display position on the screen. As for cursor address, it is not X, Y address but linear address that is programmed.

#### ■ EXAMPLES OF APPLIED CIRCUIT OF THE CRTC

Fig. 30 shows an example of application of the CRTC to monochrome character display. Its specification is shown in Table 12. Moreover, specification of CRT display unit is shown in Table 13 and initializing values for the CRTC are shown in Table 14.

Table 12 Specification of Applied Circuit

Item	Specification																
Character Format	$5 \times 7$ Dot																
Character Space	Horizontal : 3 Dot Vertical : 5 Dot																
One Character Time	$1 \mu s$																
Number of Displayed Characters	40 characters $\times$ 16 lines = 640 characters																
Access Method to Refresh Memory	Synchronous Method (DISPTMG Read)																
Refresh Memory	$2^{15} \text{ } 2^{14} : 2^{13} \text{ } 2^{12} \text{ } 2^{11} \text{ } 2^{10} \text{ } 2^9 \text{ } 2^8 \text{ } 2^7 \text{ } 2^6 \text{ } 2^5 \text{ } 2^4 \text{ } 2^3 \text{ } 2^2 \text{ } 2^1 \text{ } 2^0$																
Address Map	Refresh Memory	0	0	0	0	0	0	*	*	*	*	*	*	*	*	*	*
	CRTC Address Register	0	0	0	1	0	0	x	x	x	x	x	x	x	x	x	0
	CRTC Control Register	0	0	0	1	0	0	x	x	x	x	x	x	x	x	x	1
	$\times \dots$ don't care, * ... 0 or 1																
Synchronization Method	HVSYNC Method																

Table 13 Specification of Character Display

Item	Specification
Scan Mode	Non-interlace
Horizontal Deflection Frequency	15.625 kHz
Vertical Deflection Frequency	60.1 Hz
Dot Frequency	8 MHz
Character Dot (Horizontal $\times$ Vertical)	8 $\times$ 12 (Character Font 5 $\times$ 9)
Number of Displayed Characters (Row $\times$ Line)	40 $\times$ 16
HSYNC Width	4 $\mu s$
VSYNC Width	3 H
Cursor Display	Raster 9 ~ 10, Blink 16 Field Period
Paging, Scrolling	Not used

Table 14 Initializing Values for Character Display

Register	Name	Symbol	Initializing Value Hex (Decimal)
R0	Horizontal Total	Nht	3F (63)
R1	Horizontal Displayed	Nhd	28 (40)
R2	Horizontal Sync Position	Nhsp	34 (52)
R3	Sync Width	Nvsw, Nhsrw	34
R4	Vertical Total	Nvt	14 (20)
R5	Vertical Total Adjust	Nadj	08 (8)
R6	Vertical Displayed	Nvd	10 (16)
R7	Vertical Sync Position	Nvsp	13 (19)
R8	Interlace & Skew		00
R9	Maximum Raster Address	Nr	0B (11)
R10	Cursor Start Raster	B, P, NCSTART	49
R11	Cursor End Raster	NCEND	0A (10)
R12	Start Address (H)		00 (0)
R13	Start Address (L)		00 (0)
R14	Cursor (H)		00 (0)
R15	Cursor (L)		00 (0)

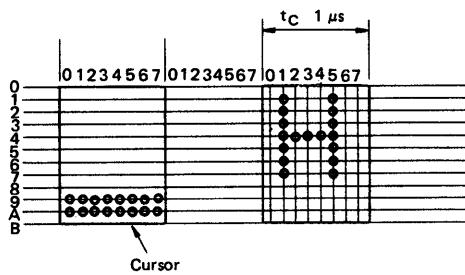


Figure 29 Non-interlace Display (Example)

## HD6845S, HD68A45S, HD68B45S

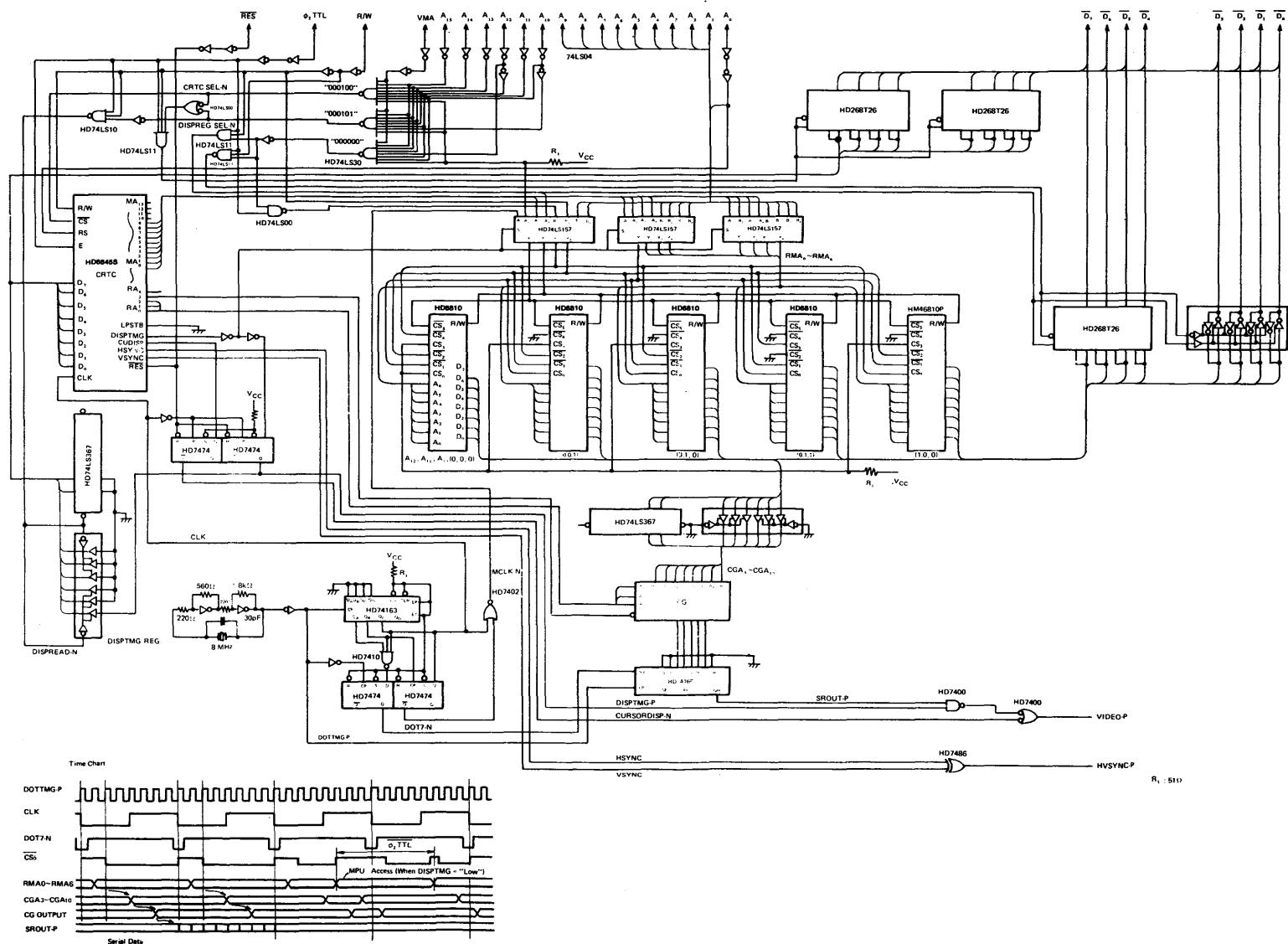
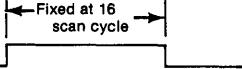
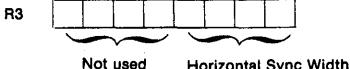
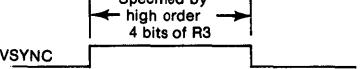
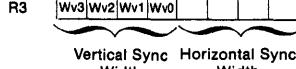
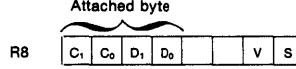
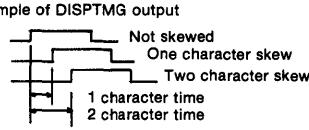


Figure 30 Example of Applied Circuit of the CRTC (Monochrome Character Display)

## Differences between the HD6845R (Motorola MC6845 Compatible) and the HD6845S (Enhanced)

No.	Functional Difference	HD6845R	HD6845S
1	Interface Sync & Video Mode Display	<p>Programming Method of number of vertical characters</p> <p>Character line address</p> <p>In HD6845R, number of characters is vertically programmed in units of two lines, as illustrated above. (Number of vertical total characters, Number of vertical displayed characters, Vertical Sync Position)</p> <p>Example of above figure . . .</p> <p>Programmed number into Vertical Displayed Register = 5</p>	<p>Character line address</p> <p>In HD6845S, number of characters is vertically programmed in unit of one line, as illustrated above. (Number of vertical total characters, Number of vertical displayed characters, Vertical Sync Position)</p> <p>Example of above figure . . .</p> <p>Programmed number into Vertical Displayed Register = 10</p>
	Number of raster per character line	<p>Only even number can be specified.</p> <p>Number of raster = 10 scanline (specified)</p> <p>However, number which is programmed into register is calculated as follows.</p> <p>Programmed number (Nr) = (Number specified) - 1</p>	<p>Both even number and odd number can be specified.</p> <p>When number of raster per character line is EVEN</p> <p>Number of raster = 10 scan line (specified)</p> <p>When number of raster per character line is ODD</p> <p>Number of raster = 9 scan line (specified)</p> <p>However, number which is programmed into register is calculated as follows.</p> <p>Programmed number (NR) = (Number specified) - 2</p>
	Cursor Display	<p>Cursor is displayed in either EVEN field or ODD field.</p>	<p>Cursor is displayed in both EVEN field and ODD field.</p>

No.	Functional Difference	HD6845R	HD6845S
2	Vertical Sync Pulse Width (VSYNC output)	Fixed at 16 raster scan cycle (16H)   	Programmable (1 - 16 raster scan cycle)   
3	SKEW Function	Not included  	SKEW capability is included in DISPTMG, CUDISP signals.   
4	Start Address Register	Write Only	Read or Write
5	RESET Signal (RES)	MA <sub>0</sub> ~ MA <sub>13</sub> Output RA <sub>0</sub> ~ RA <sub>4</sub> Output Other Outputs -----  Output signals of MA <sub>0</sub> ~ MA <sub>13</sub> , RA <sub>0</sub> ~ RA <sub>4</sub> , synchronized with CLK "LOW" level, go to "LOW" level, after RES has gone to "LOW". Other outputs go to "LOW" immediately after RES has gone to "LOW" level.	MA <sup>°</sup> ~ MA <sub>13</sub> Output, RA <sub>0</sub> ~ RA <sub>4</sub> Output Other Outputs -----  Output signals of MA <sub>0</sub> ~ MA <sub>13</sub> , RA <sub>0</sub> ~ RA <sub>4</sub> and others go to "LOW" level immediately after RES has gone to "LOW" level.

**AC Characteristic Differences between HD6845R (Motorola MC6845 Compatible) and HD6845S (Enhanced)**

No.	Characteristic Difference	Symbol	HD46505R			HD46505S			Unit
			min.	typ.	max.	min.	typ.	max.	
1	Clock Cycle Time	t <sub>cyc</sub>	330	—	—	270	—	—	ns
2	Clock Pulse Width "High"	PW <sub>CH</sub>	150	—	—	130	—	—	ns
3	Clock Pulse Width "Low"	PW <sub>CL</sub>	150	—	—	130	—	—	ns
4	Rise and Fall Time for Clock Input	T <sub>CR</sub> , T <sub>CF</sub>	—	—	15	—	—	20	ns
5	Horizontal Sync Delay Time	T <sub>HSD</sub>	—	—	250	—	—	200	ns
6	Light Pan Strobe Pulse Width	PW <sub>LPH</sub>	80	—	—	60	—	—	ns
7	Light Pan Strobe, Uncertain Time of Acceptance	T <sub>LPD1</sub>	—	—	80	—	—	70	ns
		T <sub>LPD2</sub>	—	—	10	—	—	0	ns

# HD6846

## COMBO (Combination ROM I/O Timer)

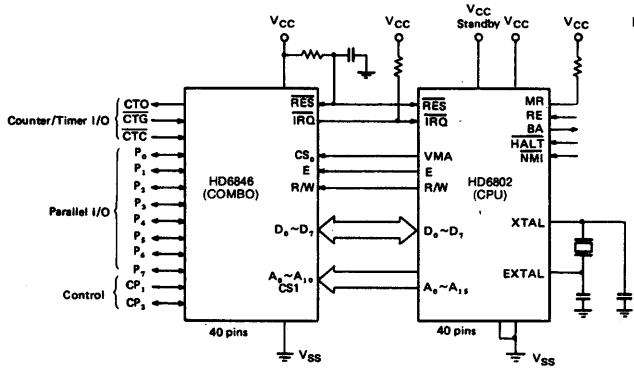
The HD6846 combination chip provides the means, in conjunction with the HD6802, to develop a basic 2-chip microcomputer system. The HD6846 consists of 2048 bytes of mask-programmable ROM, an 8-bit bidirectional data port with control lines, and a 16-bit programmable timer-counter.

This device is capable of interfacing with the HD6802 (basic HD6800, clock and 128 bytes of RAM) as well as the HD6800 if desired. No external logic is required to interface with most peripheral devices.

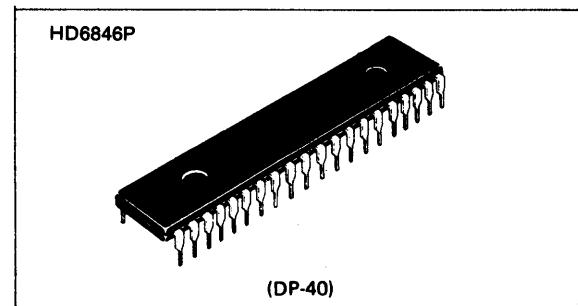
### ■ FEATURES

- 2048 8-Bit Bytes of Mask-Programmable ROM
- 8-Bit Bidirectional Data Port for Parallel Interface plus Two Control Lines
- Programmable Interval Timer-Counter Functions
- Programmable I/O Peripheral Data, Control and Direction Registers
- Compatible with the Complete HMCS6800 Microcomputer Product Family
- TTL-Compatible Data and Peripheral Lines
- Single 5-Volt Power Supply
- Compatible with MC6846

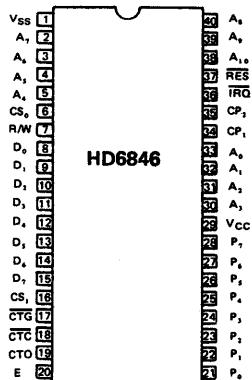
### ■ TYPICAL MICROCOMPUTER



This is a block diagram of a typical cost effective microcomputer. The MPU is the center of the microcomputer system and is shown in a minimum system interfacing with a ROM combination chip. It is not intended that this system be limited to this function but that it be expandable with other parts in the HMCS6800 Microcomputer family.



### ■ PIN ARRANGEMENT



(Top View)

## ■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	$V_{CC}^*$	-0.3 ~ +7.0	V
Input Voltage	$V_{in}^*$	-0.3 ~ +7.0	V
Operating Temperature	$T_{opr}$	-20 ~ +75	°C
Storage Temperature	$T_{stg}$	-55 ~ +150	°C

\* With respect to  $V_{SS}$  (SYSTEM GND)

(NOTE) 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.

## ■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol	min	typ	max	Unit
Supply Voltage	$V_{CC}^*$	4.75	5.0	5.25	V
Input Voltage	$V_{IL}^*$	-0.3	—	0.8	V
	$V_{IH}^*$	2.0	—	$V_{CC}$	V

\* With respect to  $V_{SS}$  (SYSTEM GND)

## ■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS ( $V_{CC}=5.0V \pm 5\%$ ,  $V_{SS}=0V$ ,  $T_a=-20 \sim +75^\circ C$ , unless otherwise noted.)

Item	Symbol	Test Condition	min.	typ.	max.	Unit
Input "High" Voltage	$V_{IH}$		2.0	—	$V_{CC}$	V
Input "Low" Voltage	$V_{IL}$		-0.3	—	0.8	V
Clock Overshoot/Ubershoot	Input "High" Level	$V_{OS}$	$V_{CC} - 0.5$	—	$V_{CC} + 0.5$	V
	Input "Low" Level		$V_{CC} - 0.5$	—	$V_{CC} + 0.5$	
Input Leakage Current	$I_{in}$	$V_{in} = 0 \sim 5.25V$	—	—	2.5	$\mu A$
Three-State (Off State) Input Current	$D_0 \sim D_7, P_0 \sim P_7, CP_2$	$I_{TSI}$	$V_{in} = 0.4 \sim 2.4V$	—	—	$\mu A$
Output "High" Voltage	$D_0 \sim D_7$	$V_{OH}$	$I_{OH} = -205\mu A$	2.4	—	V
	$CP_2, P_0 \sim P_7$		$I_{OH} = -200\mu A$	2.4	—	V
	CTO		$I_{OH} = -200\mu A$	2.4	—	V
Output "Low" Voltage	$D_0 \sim D_7$	$V_{OL}$	$I_{OL} = 1.6mA$	—	—	0.4 V
	Other Outputs		$I_{OL} = 3.2mA$	—	—	0.4 V
Output "High" Current (Sourcing)	$D_0 \sim D_7$	$I_{OH}$	$V_{OH} = 2.4V$	-205	—	$\mu A$
	$CTO, CP_2, P_0 \sim P_7$		$V_{OH} = 2.4V$	-200	—	$\mu A$
Output "High" Current (Sourcing) (the current for driving other than TTL, e.g., Darlington Base)	$CP_2, P_0 \sim P_7$	$I_{OH}$	$V_{OH} = 1.5V$	-1.0	—	—10 mA
Output "Low" Current (Sinking)	$D_0 \sim D_7$	$I_{OL}$	$V_{OL} = 0.4V$	1.6	—	—
	Other Outputs		$V_{OL} = 0.4V$	3.2	—	—
Output Leakage Current (Off State)	$IRQ$	$I_{LOH}$	$V_{OH} = 2.4V$	—	—	10 $\mu A$
Power Dissipation	$P_D$			—	—	800 mW
Capacitance	E	$C_{in}$	$V_{CC} = 0V$ $V_{in} = 0V$ $T_a = 25^\circ C$ $f = 1MHz$	—	—	20 pF
	$D_0 \sim D_7$	$C_{in}$		—	—	12.5 pF
	$P_0 \sim P_7, CP_2, CTO$	$C_{out}$		—	—	10 pF
	$A_0 \sim A_{10}, R/W$	$C_{in}$		—	—	7.5 pF
	$RES, CS_0, CS_1, CP_1, CTG$	$C_{in}$		—	—	10 pF
	$IRQ$	$C_{out}$		—	—	7.5 pF

● AC CHARACTERISTICS ( $V_{CC}=5.0V \pm 5\%$ ,  $V_{SS}=0V$ ,  $T_a=-20^{\circ}C \sim +75^{\circ}C$ , unless otherwise noted.)

1. BUS TIMING

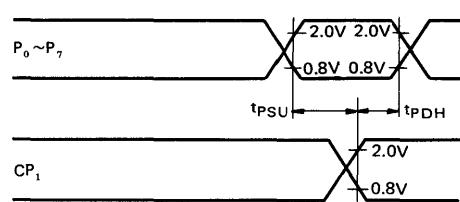
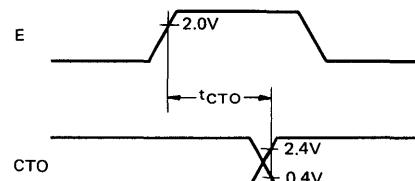
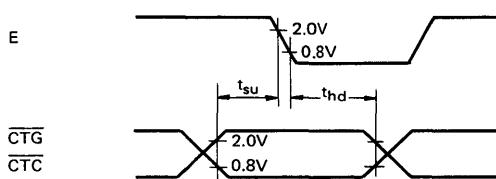
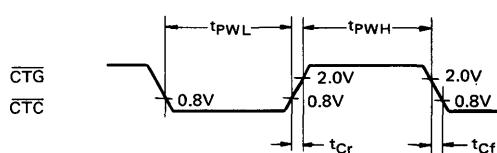
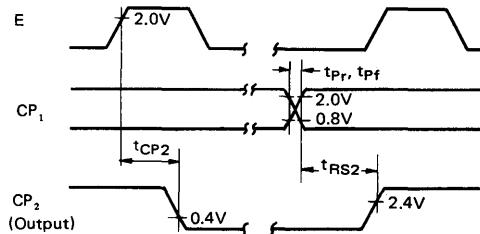
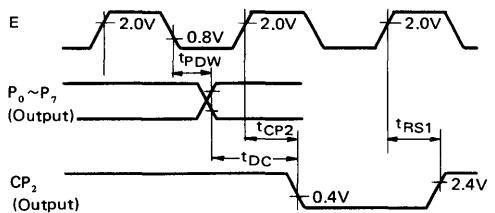
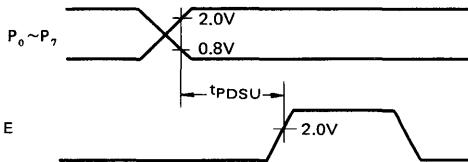
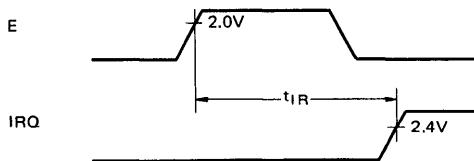
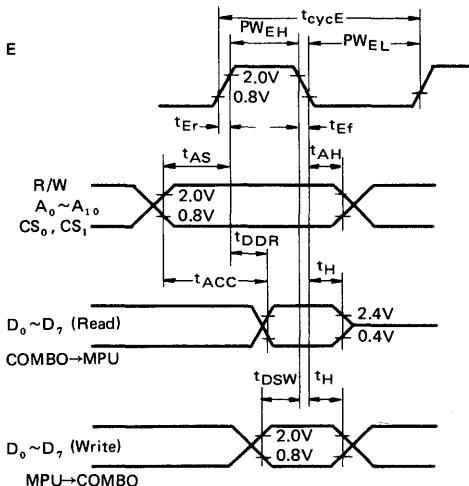
Item	Symbol	Test Condition	min	typ	max	Unit
Enable Cycle Time	$t_{cycE}$	Fig. 1	1.0	—	10	$\mu s$
Enable Pulse Width, "Low"	$PW_{EL}$		430	—	4500	ns
Enable Pulse Width, "High"	$PW_{EH}$		430	—	4500	ns
Address Set Up Time	$t_{AS}$		140	—	—	ns
Data Delay Time	$t_{DDR}$		—	—	320	ns
Data Hold Time	$t_H$		10	—	—	ns
Address Hold Time	$t_{AH}$		10	—	—	ns
Enable Rise and Fall Time	$t_{Ef}, t_{Er}$		—	—	25	ns
Data Set Up Time	$t_{DSW}$		195	—	—	ns
Reset "Low" Time	$t_{RL}$		2	—	—	$\mu s$
Interrupt Release Time	$t_{IR}$	Fig. 2	—	—	1.6	$\mu s$

2. PALLAREL PERIPHERAL I/O LINE TIMING

Item	Symbol	Test Condition	min	typ	max	Unit
Peripheral Data Setup Time	$t_{PDSU}$	Fig. 3	200	—	—	ns
Rise and Fall Times $CP_1, CP_2$	$t_{Pr}, t_{pf}$	Fig. 5	—	—	1.0	$\mu s$
Delay Time E to $CP_2$ Fall	$t_{CP2}$	Fig. 4	—	—	1.0	$\mu s$
Delay Time I/O Data $CP_2$ Fall	$t_{DC}$		20	—	—	ns
Delay Time E to $CP_2$ Rise	$t_{RS1}$		—	—	1.0	$\mu s$
Delay Time $CP_1$ to $CP_2$ Rise	$t_{RS2}$	Fig. 5	—	—	2.0	$\mu s$
Peripheral Data Delay	$t_{PDW}$	Fig. 4	—	—	1.0	$\mu s$
Peripheral Data Setup Time for Latch	$t_{PSU}$	Fig. 9	100	—	—	ns
Peripheral Data Hold Time for Latch	$t_{PDH}$		15	—	—	ns

3. TIMER/COUNTER LINE TIMING

Item	Symbol	Test Condition	min	typ	max	Unit
CTC, CTG Rise and Fall Time	$t_{Cr}, t_{Cf}$	Fig. 6	—	—	100	ns
CTC, CTG Pulse Width, "High" (Asynchronous Mode)	$t_{PWH}$		$t_{cycE} + 250$	—	—	ns
CTC, CTG Pulse Width, "Low" (Asynchronous Mode)	$t_{PWL}$		$t_{cycE} + 250$	—	—	ns
CTC, CTG Setup Time (Synchronous Mode)	$t_{su}$	Fig. 7	200	—	—	ns
CTC, CTG Hold Time (Synchronous Mode)	$t_{hd}$		50	—	—	ns
CTO Delay Time	$t_{CTO}$	Fig. 8	—	—	1.0	$\mu s$



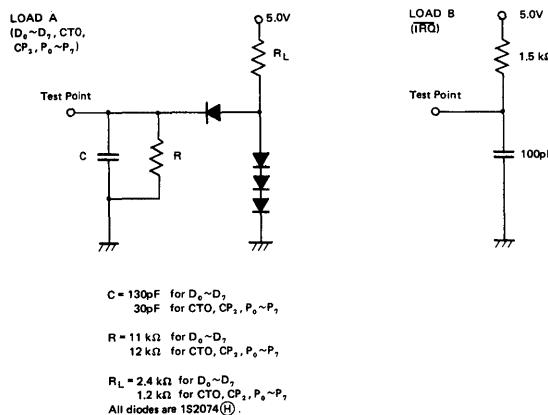


Figure 10 Bus Timing Test Loads

## ■ GENERAL DESCRIPTION

The HD6846 combination chip may be partitioned into three functional operating sections: programmed storage, timer-counter functions, and a parallel I/O port.

### ● Programmed Storage

The mask-programmable ROM section is similar to other ROM products of the HMCS6800 family. The ROM is organized in a 2048 by 8-bit array to provide read only storage for a minimum microcomputer system. Two mask-programmable chip selects are available for user definition.

Address inputs A<sub>0</sub> ~ A<sub>10</sub> allow any of the 2048 bytes of ROM to be uniquely addressed. Bidirectional data lines (D<sub>0</sub> ~ D<sub>7</sub>) allow the transfer of data between the MPU and the HD 6846.

### ● Timer-Counter Functions

Under software control this 16-bit binary counter may be programmed to count events, measure frequencies, time intervals, or similar tasks. Internal registers associated with the I/O functions may be selected with A<sub>0</sub>, A<sub>1</sub> and A<sub>2</sub>. It may also be used for square wave generation, single pulses of controlled duration, and gated signals. Interrupts may be generated from a number of conditions selectable by software programming.

The timer/counter control register allows control of the interrupt enable, output enable, selection of an internal or external clock source, a  $\div 8$  prescaler, and operating mode. Input pin CTC (counter-timer clock) will accept an asynchronous clock pulse to decrement the internal register for the counter-timer. If the divide-by-8 prescaler is used, the maximum clock rate can be four times the master clock frequency with an absolute maximum of 4 MHz. Gate input (CTG) accepts an asynchronous TTL-compatible signal which may be used as a trigger or gating function to the counter-timer. A counter-time output (CTO) is also available and is under software control being dependent on the timer control register, the gate input and the clock source.

### ● Parallel I/O Port

The parallel bidirectional I/O port has functional operations characteristics similar to the B port on the HD6821 PIA. This includes 8 bidirectional data lines and two handshake control signals. The control and operation of these lines are completely software programmable.

The interrupt input (CP<sub>1</sub>) will set the interrupt flag CSR1 in the composite status register. The peripheral control (CP<sub>2</sub>) may be programmed to act as an interrupt input (set CSR2) or as a peripheral control output.

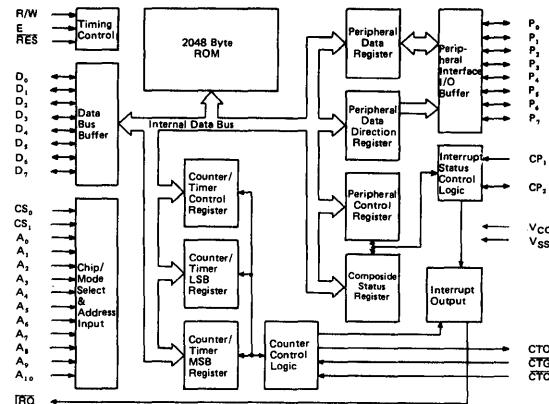


Figure 11 Combination ROM I/O Timer (COMBO)

Basic Block Diagram

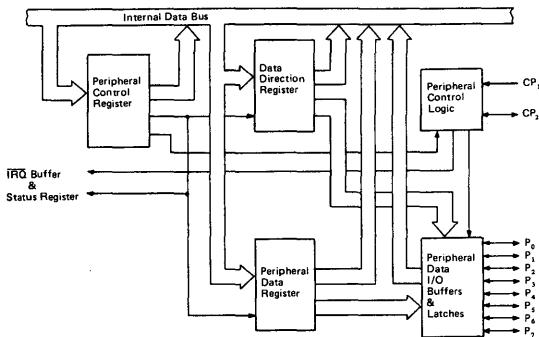


Figure 12 Parallel I/O Port Block Diagram

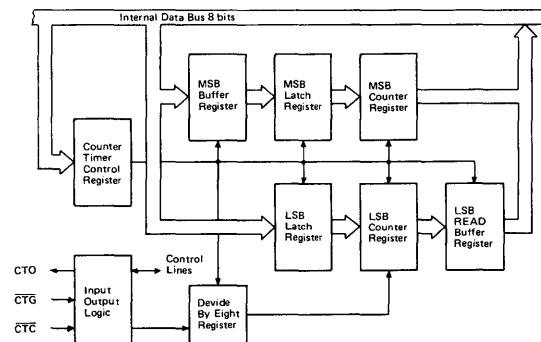


Figure 13 Timer/Counter Block Diagram

## SIGNAL DESCRIPTION

### Bus Interface

The HD6846 interfaces to the HMCS6800 Bus via an eight bit bidirectional data bus, two Chip Select lines, a Read/Write line, and eleven address lines. These signals, in conjunction with the HMCS6800 VMA output, permit the MPU to control the HD6846.

### Bidirectional Data Bus (D<sub>0</sub>~D<sub>7</sub>)

The bidirectional data lines (D<sub>0</sub> ~ D<sub>7</sub>) allow the transfer of data between the MPU and the HD6846. The data bus output drivers are three-state devices which remain in the high-impedance (Off) state except when the MPU performs an HD6846 register or ROM read (R/W = 1 and I/O Registers or ROM selected).

### Chip Select (CS<sub>0</sub>, CS<sub>1</sub>)

The CS<sub>0</sub> and CS<sub>1</sub> inputs are used to select the ROM or I/O timer of the HD6846. They are mask programmed to be active "High" or active "Low" as chosen by the user.

### Address Inputs (A<sub>0</sub> ~ A<sub>10</sub>)

The Address Inputs allow any of the 2048 bytes of ROM to be uniquely selected when the circuit is operating in the ROM mode. In the I/O-Timer mode, address inputs A<sub>0</sub>, A<sub>1</sub>, and A<sub>2</sub> select the proper I/O Register, while A<sub>3</sub> through A<sub>10</sub> (together with CS<sub>0</sub> and CS<sub>1</sub>) can be used as additional qualifiers in the I/O Select circuitry. (See the section on I/O-Timer Select for additional details.)

### Reset (RES)

The active "Low" state of the RES input is used to initialize all register bits in the I/O section of the device to their proper values. (See the section on Initialization for Reset conditions for timer and peripheral registers.)

### Enable (E)

This signal synchronizes data transfer between the MPU and the HD6846. It also performs an equivalent synchronization function on the external clock, reset, and gate inputs of the HD6846 Timer section.

### Read/Write (R/W)

This signal is generated by the MPU and is used to control the direction of data transfer on the bidirectional data pins. A "Low" level on the R/W input enables the HD6846 input buffers and data is transferred to the circuit during the E pulse when the part has been selected. A "High" level on the R/W input enables the output buffers and data is transferred to the MPU during E when the part is selected.

### Interrupt Request (IRQ)

The active "Low" IRQ output acts to interrupt the MPU through logic included on the HD6846. This output utilizes an open drain configuration and permits other interrupt request outputs from other circuits to be connected in a wire-OR configuration.

### Peripheral Data (P<sub>0</sub>~P<sub>7</sub>)

The peripheral data lines can be individually programmed as either inputs or outputs via the Data Direction Register. When programmed as outputs, these lines will drive two standard TTL

loads (3.2 mA). They are also capable of sourcing up to 1.0 mA at 1.5 Volts (Logic "1" output.)

When programmed as inputs, the output drivers associated with these lines enter a three-state (high impedance) mode. Since there is no internal pull-up for these lines, they represent a maximum 10 $\mu$ A load to the circuitry driving them – regardless of logic state.

A logic zero at the RES input forces the peripheral data lines to the input configuration by clearing the Data Direction Register. This allows the system designer to preclude the possibility of having a peripheral data output connected to an external driver output during power-up sequence.

#### • Interrupt Input (CP<sub>1</sub>)

Peripheral input line CP<sub>1</sub> is an input-only that sets the Interrupt Flags of the Composite Status register. The active transition for this signal is programmed by the peripheral control register for the parallel port. CP<sub>1</sub> may also act as a strobe for the peripheral data register when it is used as an input latch. Details for programming CP<sub>1</sub> are in the section on the parallel peripheral port.

#### • Peripheral Control (CP<sub>2</sub>)

Peripheral Control line CP<sub>2</sub> may be programmed to act as an interrupt input or Peripheral Control output. As an input, this line has high impedance and is compatible with standard TTL voltage levels. As an output, it is also TTL compatible and may be used as a source of 1 mA at 1.5 V to directly drive the base of a Darlington transistor switch. This line is programmed by the Peripheral Control Register.

#### • Counter Timer Output (CTO)

The Counter Timer Output is software programmable by selected bits in the timer/counter control register. The mode of operation is dependent on the Timer control register, the gate input, and the clock source. The output is TTL compatible.

#### • External Clock Input (CTC)

Input pin CTC will accept asynchronous TTL voltage level signals to be used as a clock to decrement the Timer. The "High" and "Low" levels of the external clock must be stable for at least one system clock period plus the sum of the setup and hold times for the inputs. The asynchronous clock rate can vary from dc to the limit imposed by System E, setup, and hold times.

The external clock input is clocked in by Enable pulses. Three Enable periods are used to synchronize and process the external clock. The fourth Enable pulse decrements the internal counter. This does not affect the input frequency; it merely creates a delay between a clock input transition and internal recognition of that transition by the HD6846. All references to

CTC inputs in this document relate to internal recognition of the input transition. Note that a clock transition which does not meet setup and hold time specifications may require an additional Enable pulse for recognition.

When observing recurring events, a lack of synchronization will result in either "System jitter" or "Input jitter" being observed on the output of the HD6846 when using an asynchronous clock and gate input signal. "System jitter" is the result of the input signals being out of synchronization with Enable, permitting signals with marginal set-up and hold time to be recognized by either the bit time nearest the input transition or subsequent bit time. "Input jitter" can be as great as the time between the negative going transitions of the input signal plus the system jitter if the first transition is recognized during one system cycle, and not recognized the next cycle or vice-versa.

#### • Gate Inputs (CTG)

The input pin CTG accepts an asynchronous TTL-compatible signal which is used as a trigger or a clock gating function to the Timer. The gating input is clocked into the HD6846 by the Enable signal in the same manner as the previously discussed clock inputs. That is, a CTG transition is recognized on the fourth Enable pulse (provided setup and hold time requirements are met), and the "High" or "Low" levels of the CTG input must be stable for at least one system clock period plus the sum of setup and hold times. All references to CTG transition in this document relate to internal recognition of the input transition.

The CTG input of the timer directly affects the internal 16-bit counter. The operation of CTG is therefore independent of the ÷8 prescaler selection.

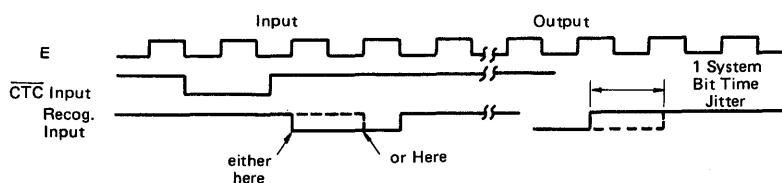
### ■ FUNCTIONAL SELECT CIRCUITRY

#### • I/O-Timer Select Circuitry

CS<sub>0</sub> and CS<sub>1</sub> are user programmable. Any of the four binary combinations of CS<sub>0</sub> and CS<sub>1</sub> can be used to select the ROM. Likewise, any other combination can be used to select the I/O-Timer. In addition, several address lines are used as qualifiers for the I/O-Timer. Specifically, A<sub>3</sub> = A<sub>4</sub> = A<sub>5</sub> = logical "0". A<sub>6</sub> can be programmed to a "1", "0", or don't care. A<sub>7</sub> = A<sub>8</sub> = A<sub>9</sub> = A<sub>10</sub> = don't care or one line only may be programmed to a logical "1". Figure 14 outlines in diagrammatic form the available chip select options.

#### • Internal Addressing

Seven I/O Register locations within the HD6846 are accessible to the MPU data bus. Selection of these registers is controlled by A<sub>0</sub>, A<sub>1</sub>, and A<sub>2</sub> (as shown in Table 1) provided the I/O timer is selected. The combination status register is Read-only; all other Registers are Read and Write.



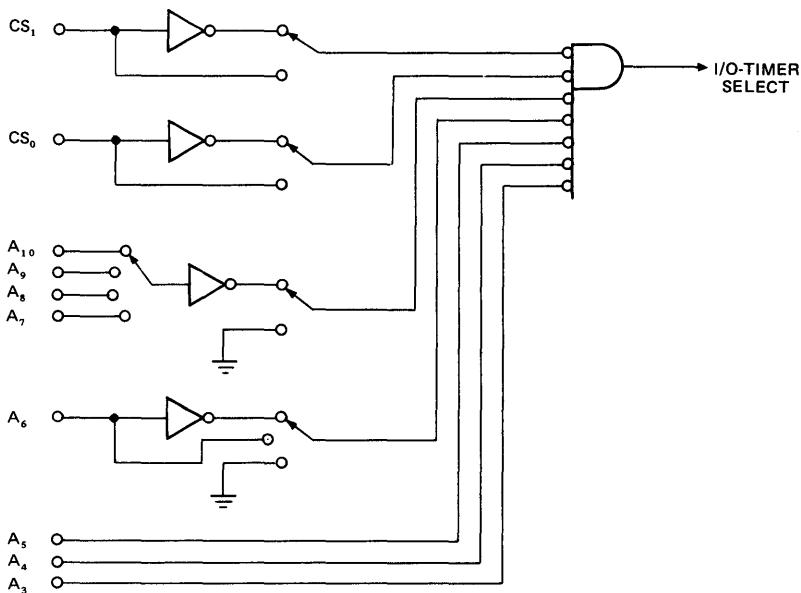


Figure 14 I/O-Timer Select Circuitry

Table 1 Internal Register Addresses

REGISTER SELECTED	A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>
Combination Status Register	0	0	0
Peripheral Control Register	0	0	1
Data Direction Register	0	1	0
Peripheral Data Register	0	1	1
Combination Status Register	1	0	0
Timer Control Register	1	0	1
Timer MSB Register	1	1	0
Timer LSB Register	1	1	1
ROM Address	x	x	x

### Initialization

When the RES input has accepted a "Low" signal, all registers are initialized to the reset state. The data direction and peripheral data registers are cleared. The Peripheral Control Register is cleared except for bit 7 (the RES bit). This forces the parallel port to the input mode with Interrupts disabled. To remove the RES condition from the parallel port, a "0" must be written into the Peripheral Control Register bit 7 (PCR7).

The counter latches are preset to their maximal count, the Timer control register bits are reset to zero except for Bit 0 (TCR0 is set), the counter output is cleared, and the counter clock disabled. This state forces the timer counter to remain in an inactive state. The combination status register is cleared of all interrupt flags. During timer initialization, the reset bit (CCR0) must be cleared.

### ROM

The Mask Programmable ROM section is similar in operation to other ROM products of the HMCS6800 Microprocessor family. The ROM is organized as 2048 words of 8-bits to provide read-only storage for a minimum microcomputer system. The ROM is active when selected by the unique

combination of the chip select inputs.

### ROM Select

The active levels of CS<sub>0</sub> and CS<sub>1</sub> for ROM and I/O select are a user programmable option. Either CS<sub>0</sub> or CS<sub>1</sub> may be programmed active "High" or active "Low", but different codes must be used for ROM or I/O select. CS<sub>0</sub> and CS<sub>1</sub> are mask programmed simultaneously with the ROM pattern. The ROM Select Circuitry is shown in Figure 15.

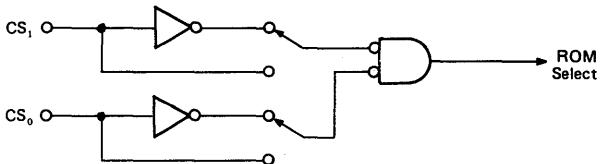


Figure 15 ROM Select Circuitry

### TIMER OPERATION

The Timer may be programmed to operate in modes which fit a wide variety of applications. The device is fully bus compatible with the HMCS6800 system, and is accessed by Load and Store operations from the MPU.

In a typical application, the timer will be loaded by storing two bytes of data into the counter latch. This data is then transferred into the counter during a Counter Initialization cycle. The counter decrements on each subsequent clock cycle (which may be Enable or an external clock) until one of several predetermined conditions causes it to halt or recycle. Thus the timer is programmable, cyclic in nature, controllable by external inputs or MPU program, and accessible to the MPU at any time.

### ● Counter Latch Initialization

The Timer consists of a 16-bit addressable counter and two 8-bit addressable latches. The function of the latches is to store a binary equivalent of the desired count value minus one. Counter initialization results in the transfer of the latch contents of the counter. It should be noted that data transfer to the counters is always accomplished via the latches. Thus, the counter latches may be accurately described as a 16-bit "counter initialization data" storage register.

In some modes of operation, the initialization of the latches will cause simultaneous counter initialization (i.e. immediate transfer of the new latch data into the counters). It is, therefore, necessary to insure that all 16-bit of the latches are updated simultaneously. Since the HD6846 data bus is 8-bit wide, a temporary register (MSB Buffer Register) is provided for in the Most Significant Byte of the desired latch data. This is a "write-only" register selected via address lines  $A_0$ ,  $A_1$ , and  $A_2$ . Data is transferred directly from the data bus to the MSB Buffer when the chip is selected, R/W is "Low", and the timer MSB register is selected ( $A_0 = "0"$ ,  $A_1 = A_2 = "1"$ ).

The lower 8-bit of the counter latch can also be referred to as a "write-only" register. Data Bus information will be transferred directly to the LSB of a counter latch when the chip is selected, R/W is "Low" and the Timer LSB Register is selected ( $A_0 = A_1 = A_2 = "1"$ ). Data from the MSB Buffer will automatically be transferred into the Most Significant Byte of the counter latches simultaneously with the transfer of the Data Bus information to the Least Significant Byte of the Counter Latch. For brevity, the conditions for this operation will be referred to henceforth as a "Write Timer Latches Command."

The HD6846 has been designed to allow transfer of two bytes of data into the counter latches from any source, provided the MSB is transferred first. In many applications, the source of data will be an HMCS6800 MPU. It should therefore be noted that the 16-bit store operations of the HMCS6800 family microprocessors (STS and STX) transfer data in the order required by the HD6846. A Store Index Register instruction, for example, results in the MSB of the index register being transferred to the selected address, then the LSB of the index register being written into the next higher location. Thus, either

the index register or stack pointer may be transferred directly into a selected counter latch with a single instruction.

A logic zero at the  $\bar{RES}$  input also initializes the counter latches. All latches will assume maximum count (65,535 values). It is important to note that an internal reset (Bit zero of the Timer/Control Register Set) has no effect on the counter latches.

### ● Counter Initialization

Counter Initialization is defined as the transfer of data from the latches to the counter with attendant clearing of the Individual Interrupt Flag associated with the counter. Counter Initialization always occurs when a reset condition (external  $\bar{RES} = "0"$  or  $TCR0 = "1"$ ) is recognized. It can also occur (dependent on The Timer Mode) with a Write Timer Latches command or recognition of a negative transition of the  $\bar{CTC}$  input.

Counter recycling or reinitialization occurs when a clock input is recognized after the counter has reached an all-zero state. In this case, data is transferred from the Latches to the Counter, but the Interrupt Flag is unaffected.

### ● Timer Control Register

The Timer Control register (see Table 2) in the HD6846 is used to modify timer operation to suit a variety of application. The Timer Control Register has a unique address space ( $A_0 = "1"$ ,  $A_1 = "0"$ ,  $A_2 = "J"$ ) and therefore may be written into at any time. The least significant bit of the Control Register is used as an Internal Reset bit. When this bit is a logic zero, all timer bits are allowed to operate in the modes prescribed by the remaining bits of the timer control register.

Writing "1" into Timer Control Register Bit 0 (TCR0) causes the counter to be preset with the contents of the counter latches; all counter clocks are disabled, and the timer output and interrupt flag (Status Register) are reset. The Counter Latch and Timer/Control Register are undisturbed by an Internal Reset and may be written into regardless of the state of TCR0.

Timer Control Register Bit 1 (TCR1) is used to select the clock source. When  $TCR1 = "0"$ , the external clock input  $\bar{CTC}$  is selected, and when  $TCR1 = "1"$ , the timer uses Enable.

Table 2 Format for Timer/Counter Control Register

CONTROL REGISTER BIT	STATE	BIT DEFINITION	STATE DEFINITION
TCR0	0	Internal Reset	Timer Enabled
	1		Timer in Preset State
TCR1	0	Clock Source	Timer uses External Clock ( $\bar{CTC}$ )
	1		Timer uses $\phi 2$ System Clock
TCR2	0	$\div 8$ Prescaler Enabler	Clock is not Prescaled
	1		Clock is prescaled by $\div 8$ Counter
TCR3 TCR4 TCR5	x x x	Operating Mode Selection	See Table 3
TCR6	0	Timer Interrupt Enable	$\bar{IRQ}$ Masked from Timer
	1		$\bar{IRQ}$ Enabled from Timer
TCR7	0	Timer Output Enable	Counter Output (CTO) Set "LOW"
	1		Counter Output Enabled

Table 3 Counter/Timer Operation Modes

Mode	TCR3	TCR4	TCR5	Counter Initialization "CI"	Counter Enable "CE"	Counter Clock "CC"	Interrupt Flag	
							Set	Clear
Continuous Mode	0	0	0	$\bar{G} \downarrow +W+R$	$(\bar{G}=\text{Low}) \cdot \bar{R}$	$CE \cdot C$	TO	RS-RT or CI
	0	1	0	$\bar{G} \downarrow +R$	$(\bar{G}=\text{Low}) \cdot \bar{R}$	$CE \cdot C$	TO	RS-RT or CI
Cascaded Single Shot Mode	0	0	1	$\bar{G} \downarrow +W+R$	$\bar{R}$	$CE \cdot C$	TO	RS-RT or CI
Normal Single Shot Mode	0	1	1	$\bar{G} \downarrow +R$	$\bar{R}$	$CE \cdot C$	TO	RS-RT or CI
Frequency Comparison Mode	1	0	0	$(\bar{C}E+TOF \cdot CE) \cdot \bar{G} \downarrow +R$	$CE \text{ set} = \bar{G} \downarrow \cdot \bar{W} \cdot \bar{R} \cdot T$ $CE \text{ reset} = W+R+I$	$CE \cdot C$	$\bar{G} \downarrow \text{ before TO}$	RS-RT or CI or W
	1	0	1	$\bar{G} \downarrow \cdot \bar{T}+R$	$CE \text{ set} = \bar{G} \downarrow \cdot \bar{W} \cdot \bar{R} \cdot T$ $CE \text{ reset} = W+R+I$	$CE \cdot C$	$\bar{G} \downarrow \text{ before TO}$	RS-RT or CI or W
Pulse Width Comparison Mode	1	1	0	$\bar{G} \downarrow \cdot \bar{T}+R$	$CE \text{ set} = \bar{G} \downarrow \cdot \bar{W} \cdot \bar{R} \cdot T \cdot G$ $CE \text{ reset} = W+R+I+(\bar{G}=\text{High})$	$CE \cdot C$	$\bar{G} \uparrow \text{ before TO}$	RS-RT or CI or W
	1	1	1	$\bar{G} \downarrow \cdot \bar{T}+R$	$CE \text{ set} = \bar{G} \downarrow \cdot \bar{W} \cdot \bar{R} \cdot T \cdot G$ $CE \text{ reset} = W+R+I+(\bar{G}=\text{High})$	$CE \cdot C$	$\bar{G} \uparrow \text{ before TO}$	RS-RT or CI or W

R = External  $\bar{RES}$  or Internal Reset TCR0

W = Write Timer Latch

I = Interrupt Flag

G = CTG

C = Clock selected in the internal register

 $\bar{G} \downarrow$  = Negative transition of CTG signal $G \uparrow$  = Positive transition of CTG signalRS-RT = Read Operation of Timer Counter after the read of Status Register  
(Normal operation to clear the interrupt)

CI = Counter Initialization (Internal Signal)

TOF = Time Out Flag (Set by CI•TO, Reset by CI)

TO = Counter Time Out

Timer Control Register Bit 2 (TCR2) enables the  $\div 8$  prescaler (TCR2 = "1"). In this mode, the clock frequency is divided by eight before being applied to the counter. When TCR2 = "0" Enable is applied directly to the counter.

TCR3, 4, 5 select the Timer Operating Mode, and are discussed in the next section.

Timer Control Register Bit 6 (TCR6) is used to mask or enable the Timer Interrupt Request. When TCR6 = "0", the Interrupt Flag is masked from the timer. When TCR6 = "1", the Interrupt Flag is enabled into Bit 7 of the Composite Status Register (Composite IRQ Bit), which appears on the  $\bar{IRQ}$  output pin.

Timer Control Register Bit Seven (TCR7) has a special function when the timer is in the Cascaded Single Shot mode. (This function is explained in detail in the section describing the mode.) In all other modes, TCR7 merely acts as an output enable bit. If TCR7 = "0", the Counter Timer Output ("TO") is forced "Low". Writing a logic one into TCR7 enables CTO.

#### • Timer Operating Modes

The HD6846 has been designed to operate effectively in a wide variety of applications. This is accomplished by using three bits of the control register (TCR3, TCR4, and TCR5) to define different operating modes of the Timer, outlined in Table 3.

#### • Continuous Operating Mode (TCR3 = 0, TCR5 = 0)

The timer may be programmed to operate in a continuous counting mode by writing zeros into bits 3 and 5 of the timer control register. Assuming that the timer output is enabled (TCR7 = "1"), a square wave will be generated at the Timer Output CTO (See Table 4).

Table 4 Continuous Operating Modes

CONTINUOUS MODE (TCR3=0, TCR7=1, TCR5=0)			
CONTROL REGISTER		INITIALIZATION/OUTPUT WAVEFORMS	
TCR4	Counter	CTO	
0	Initialization $\bar{G} \downarrow +W+R$	$\bar{(N+1)}(T) \rightarrow$	$\bar{(N+1)}(T) \rightarrow$
1	$\bar{G} \downarrow +R$	$\bar{(N+1)}(T) \rightarrow$	$V_{OH}$
		$\bar{to} \quad \bar{TO^*} \quad \bar{TO^*} \quad \bar{TO^*}$	$V_{OL}$

 $\bar{G} \downarrow$  = Negative Transition CTG Input.

W = Write Timer Latches Command.

R = Timer Reset (TCR0=1 or External  $\bar{RES}=0$ )

N = 16 Bit Number in Counter Latch.

T = Period of Clock Input to Counter.

to = Counter Initialization Cycle.

TO = Counter Time Out (All Zero Condition.)

\* Point at which an interrupt may occur.

Either a Timer Reset (TCR0 = "1" or External  $\bar{RES} = 0$ ) condition or internal recognition of a negative transition of the CTG input results in Counter Initialization. A Write Timer Latches command can be selected as a Counter Initialization signal by clearing TCR4.

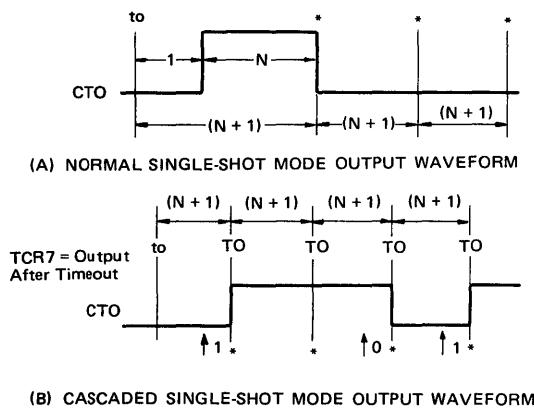
The discussion of the Continuous Mode has assumed the application requires an output signal. It should be noted the Timer operates in the same manner with the output disabled (TCR7 = "0"). A Read Timer Counter command is valid regardless of the state of TCR7.

- Normal Single-Shot Timer Mode (TCR3 = 0, TCR4 = 1, TCR5 = 1)

This mode is identical to the Continuous Mode with two exceptions. The first of these is obvious from the name – the output returns to a “Low” level after the initial Time Out and remains “Low” until another Counter Initialization cycle occurs. The output waveform (CTO) is shown in Figure 16.

As indicated in Figure 16, the internal counting mechanism remains cyclical in the Single-Shot Mode. Each Time Out of the counter results in the setting of an Individual Interrupt Flag and re-initialization of the counter.

The second major difference between the Single-Shot and Continuous modes is that the internal counter enable is not dependent on the  $\overline{\text{CTG}}$  input level remaining in the “Low” state for the Single-Shot mode. Aside from these differences, the two modes are identical.



\*Point at which an interrupt may occur.

(NOTE) All time intervals shown above assume the Gate ( $\overline{\text{CTG}}$ ) and Clock (CTC) signals are synchronized to Enable with the specified setup and hold time requirements.

Figure 16 Single-Shot Modes

- Cascaded Single-shot Mode (TCR3=0, TCR4=0, TCR5=1)

This mode is identical to the single-shot mode with two exceptions. First, the output waveform does not return to a “Low” level and remain “Low” after timeout. Instead, the output level remains at its initialized level until it is re-programmed and changed by timeout. The output level may be changed at any timeout or may have any number of timeouts between changes.

The second difference is the method used to change the output level. Timer Control Register Bit 7 (TCR7) has a special function in this mode. The timer output (CTO) is equal to TCR7 clocked by timeout. At every timeout, the content of TCR7 is clocked to and held at the CTO output. Thus, output pulses of length greater than one timer cycle can be generated by cascading timer cycles and counting timeouts with a software program (See Figure 16).

An interrupt is generated at each timeout. To cascade timer cycles, the MPU would need an interrupt routine to: 1) count each timeout and determine when to change TCR7; 2) write into TCR7 the state corresponding to the next desired state of the output waveform (only necessary during the last timer cycle before the output is to change state); and 3) clear the interrupt flag by reading the combination status register followed by Read Timer MSB. It is also possible, if desired, to change the length of the timer cycle by reinitializing the timer latches. This allows more flexibility for obtaining desired times.

- Time Interval Modes (TCR3 = 1)

The Time Interval Modes are provided for applications requiring more flexibility of interrupt generation and Counter Initialization. The Interrupt Flag is set in these modes as a function of both Counter Time Out and transitions of the CTG input. Counter Initialization is also affected by Interrupt Flag status. The output signal is not defined in any of these modes. Other features of the Time Interval Modes are outlined in Table 5.

- Frequency Comparison Mode (TCR3 = 1, TCR4 = 0)

The timer within the HD6846 may be programmed to compare the period of a pulse (giving the frequency after calculations) at the CTG input with the time period required for Counter Time Out. A negative transition of the CTG input enables the counter and starts a Counter Initialization cycle – provided that other conditions as noted in Table 3 are satisfied. The counter decrements on each clock signal recognized during or after Counter Initialization until an Interrupt is generated, a

Table 5 Time Interval Modes

TCR3 = 1			
TCR4	TCR5	APPLICATION	CONDITION FOR SETTING INDIVIDUAL INTERRUPT FLAG
0	0	Frequency Comparison	Interrupt Generated if $\overline{\text{CTG}}$ Input Period (1/F) is Less Than Counter Time Out (TO).
0	1	Frequency Comparison	Interrupt Generated if $\overline{\text{CTG}}$ Input Period (1/F) is Greater Than Counter Time Out (TO).
1	0	Pulse Width Comparison	Interrupt Generated if $\overline{\text{CTG}}$ Input “Down Time” is Less Than Counter Time Out (TO).
1	1	Pulse Width Comparison	Interrupt Generated if $\overline{\text{CTG}}$ Input “Down Time” is Greater Than Counter Time Out (TO).

Write Timer Latches command is issued, or a Timer Reset condition occurs. It can be seen from Table 3 that an interrupt condition will be generated if TCR5 = "0" and the period of the pulse (single pulse or measured separately repetitive pulses) at the CTG input is less than the Counter Time Out period. If TCR5 = "1", an interrupt is generated if the reverse is true.

Assume now with TCR5 = "1" that a Counter Initialization has occurred and that the  $\overline{CTG}$  input has returned "Low" prior to Counter Time Out. Since there is no Individual Interrupt Flag generated, this automatically starts a new Counter Initialization Cycle. The process will continue with frequency comparison being performed on each  $\overline{CTG}$  input cycle until the mode is changed, or a cycle is determined to be above the predetermined limit.

#### • Pulse Width Comparison Mode (TCR3 = 1, TCR4 = 1)

This mode is similar to the Frequency Comparison Mode except for the limiting factor being a positive, rather than negative, transition of the  $\overline{CTG}$  input. With TCR5 = "0", an Individual Interrupt Flag will be generated if the zero level pulse applied to the  $\overline{CTG}$  input is less than the time period required for Counter Time Out. With TCR5 = "1", the interrupt is generated when the reverse condition is true.

As can be seen in Table 3, a positive transition of the  $\overline{CTG}$  input disables the counter. With TCR5 = "0", it is therefore possible to directly obtain the width of any pulse causing an interrupt.

#### • Composite Status Register

The Composite Status Register (CSR) is a read-only register which is shared by the Timer and the Peripheral Data Port of the HD6846. Three individual interrupt flags in the register are set directly via the appropriate conditions in the timer or peripheral port. The composite interrupt flag — and the  $\overline{IRQ}$  Output — respond to these individual interrupts only if corresponding enable bits are set in the appropriate Control Registers. (See Figure 17.) The sequence of assertion is not detected. Setting TCR6 while CSR0 is "High" will cause CSR7 to be set, for example.

The Composite Interrupt Flag (CSR7) is clear only if all enabled Individual Interrupt Flags are clear. The conditions for

clearing CSR1 and CSR2 are detailed in a later section. The Timer Interrupt Flag (CSR0) is cleared under the following conditions:

- 1) Timer Reset — Internal Reset Bit (TCR0) = "1" or External RES = "0".
- 2) Any Counter Initialization condition.
- 3) A Write Timer Latches command if Time Interval modes (TCR3 = "1") are being used.
- 4) A Read Timer Counter command, provided this is preceded by a Read Composite Status Register while CSRO is set. This latter condition prevents missing an Interrupt Request generated after reading the Status Register and prior to reading the counter.

The remaining bits of the Composite Status Register (CSR3~CSR6) are unused. They default to a logic zero when read.

#### ■ I/O OPERATION

##### • Parallel Peripheral Port

The peripheral port of the HD6846 contains 8 Peripheral Data lines ( $P_0 \sim P_7$ ), two Peripheral Control lines ( $CP_1$  and  $CP_2$ ), a Data Direction Register, a Peripheral Data Register, and a Peripheral Control Register. The port also directly affects two bits (CSR1 and CSR2) of the Composite Status Register.

The Peripheral Port is similar to the "B" side of a PIA (HD46821) with the following exceptions:

- 1) All registers are directly accessible in the HD6846 Data Direction and Peripheral Data in the HD6821 are located at the same address, with Bit Two of the Control Register used for register selection.
- 2) Peripheral Control Register Bit Two (PRC2) of the HD6846 is used to select an optional input latch function. This option is not available with HD6821 PIA's.
- 3) Interrupt Flags are located in the HD6846 composite status register rather than Bits 6 and 7 of the Control Register as used in the HD6821.
- 4) Interrupt Flags are cleared in the HD6821 by reading data from the Peripheral Data Register. HD6846 Interrupt Flags are cleared by either reading or writing to the Peripheral Data Register — provided that this sequence is followed a) Flag Set, b) Read Composite Status Register, c) Read/Write Peripheral Data Register is followed.

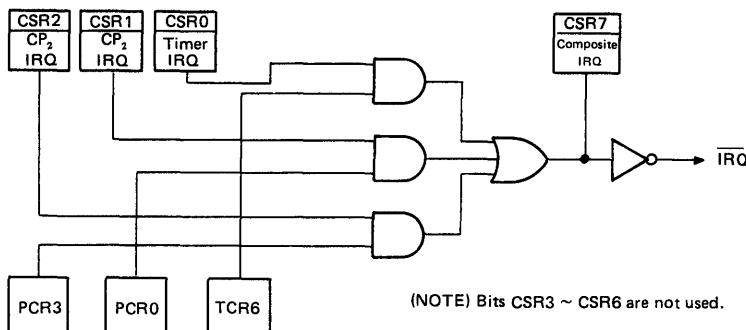


Figure 17 Composite Status Register & Associated Logic

- 5) Bit 6 of the HD6846 Peripheral Control Register is not used. Bit 7 (PCR7) is an Internal Reset Bit not available on the HD6821.
- 6) The Peripheral Data lines (and CP<sub>2</sub>) of the HD6846 features internal current limiting which allows them to directly drive the base of Darlington NPN transistors.

#### • Data Direction Register

The MPU can write directly to this eight-bit register to configure the Peripheral Data lines as either inputs or outputs. A particular bit within the register (DDR<sub>n</sub>) is used to control the corresponding Peripheral Data line (P<sub>n</sub>). With DDR<sub>n</sub> = "0", P<sub>n</sub> becomes an input; if DDR<sub>n</sub> = "1", P<sub>n</sub> is an output. As an example, writing Hex \$0F into the Data Direction Register results in P<sub>0</sub> thru P<sub>3</sub> becoming outputs and P<sub>4</sub> thru P<sub>7</sub> being inputs. Hex \$55 in the Data Direction Register results in alternate outputs and inputs at the parallel port.

#### • Peripheral Data Register

This eight-bit register is used for transferring data between the peripheral data port and the MPU. Any bit corresponding to an output line will be used to drive the output buffer associated with that line. Data in these output bits is normally provided by an MPU Write function. (Input bits – those associated with input lines – are unchanged by a Write Command.) Any input bit will reflect the state of the associated input line if the input latch function is deselected. If the Control Register is programmed to provide input latching, the input bit will retain the state at the time CP<sub>1</sub> was activated until the Peripheral Data Register is read by the MPU.

#### • Peripheral Control Register

This eight-bit register is used to control the reset function as well as for selection of optional functions of the two peripheral control lines (CP<sub>1</sub> and CP<sub>2</sub>). The Peripheral Control Register functions are outlined in Table 6.

#### • Peripheral Port Reset (PCR7)

Bit 7 of the Peripheral Control Register (PCR7) may be used to initialize the peripheral section of the HD6846. When this bit is set "High", the peripheral data register, the peripheral data direction register, and the interrupt flags associated with the peripheral port (CSR1 & CSR2) are all cleared. Other bits in the peripheral control register are not affected by PCR7.

PCR7 is set by either a logic zero at the External RES input or under program control by writing a "1" into the location. In any case, PCR7 may be cleared only by writing a zero into the location while RES is "High". The bit must be cleared to activate the port.

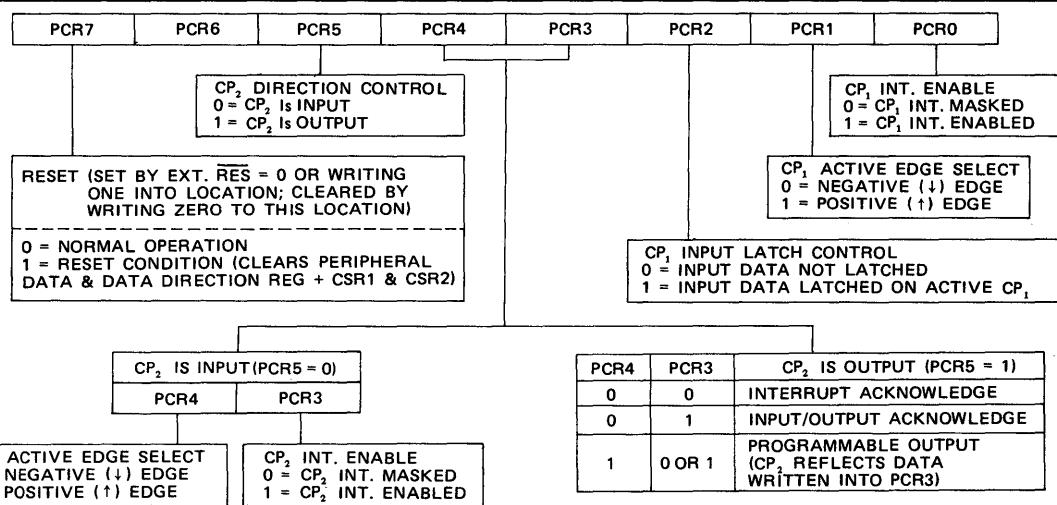
#### • Control of CP<sub>1</sub> Peripheral Control Line

CP<sub>1</sub> may be used an interrupt request to the HD6846, as a strobe to allow latching of input data, or both. In any case, the input can be programmed to be activated by either a positive or negative transition of the signal. These options are selected via Control Register Bits PCR0, RCR1 & PCR2.

Control Register Bit 0 (PCR0) is used to enable the interrupt transfer circuitry of the HD6846. Regardless of the state of PCR0, and active transition of CP<sub>1</sub> causes the Composite Status Register Bit One (CSR1) to be set. If PCR0 = "1", this interrupt will be reflected in the Composite Interrupt Flag (CSR7), and thus at the IRQ output. CSR1 is cleared by a Peripheral Port Reset condition or by either reading or writing to the peripheral data register after the Composite Status Register is read. The latter alternative is conditional – CSR1 must have been a logic one when the Composite Status Register was last read. This precludes inadvertent clearing of interrupt flags generated between the time the Status Register is read and the manipulation of peripheral data.

Control Register Bit One (PCR1) is used to select the edge which activates CP<sub>1</sub>. When PCR1 = "0", CP<sub>1</sub> is active on negative transitions ("High" to "Low"). "Low" to "High" transitions are sensed by CP<sub>1</sub> when PCR1 = "1".

Table 6 Peripheral Control Register Format (Expanded)



In addition to its use as an interrupt input, CP<sub>1</sub> can be used as a strobe to capture input data in an internal latch. This option is selected by writing a one into Peripheral Control Register Bit Two (PCR2). In operating, the data at the pins designated by the Data Direction Register as inputs will be captured by an active transition of CP<sub>1</sub>. An MPU Read of the Peripheral Data Register will result in the captured data being transferred to the MPU — and it also releases the latch to allow capture of new data. Note that successive active transitions with no Read Peripheral Data Command between does not update the input latch. Also, it should be noted that use of the input latch function (which can be deselected by writing a zero into PCR2) has no effect on output data. It also does not affect Interrupt function of CP<sub>1</sub>.

#### ● Control of CP<sub>2</sub> Peripheral Control Line

CP<sub>2</sub> may be used as an input by writing a zero into PCR5. In this configuration, CP<sub>2</sub> becomes a dual of CP<sub>1</sub> in regard to generation of interrupts. An active transition (as selected by PCR4) causes Bit Two of the Composite Status Register to be set. PCR3 is then used to select whether the CP<sub>2</sub> transition is to cause CSR7 to be set — and thereby cause IRQ to go “Low”. CP<sub>2</sub> has no effect on the input latch function of the HD6846.

Writing a one into PCR5 causes CP<sub>2</sub> to function as an output. PCR4 then determines whether CP<sub>2</sub> is to be used in a handshake or programmable output mode. With PCR4 = “1”, CP<sub>2</sub> will merely reflect the data written into PCR3. Since this can readily be changed under program control, this mode allows CP<sub>2</sub> to be a programmable output line in much the same

manner as those lines selected as outputs by the Data Direction Register.

The handshaking mode (PCR5 = “1”, PCR4 = “0”) allows CP<sub>2</sub> to perform one of two functions as selected by PCR3. With PCR3 = “1”, CP<sub>2</sub> will go “Low” on the first Enable positive transition after a Read or Write to the Peripheral Data Register. This Input/Output Acknowledge signal is released (returns “High”) on the next positive transition of the Enable signal.

In the Interrupt Acknowledge mode (PCR5 = “1”, PCR4 = PCR3 = “0”), CP<sub>2</sub> is set when CSR1 is set by an active transition of CP<sub>1</sub>. It is released (goes “Low”) on the first positive transition of Enable after CSR1 has been cleared via an MPU Read or Write to the Peripheral Data Register. (Note that the previously described conditions for clearing CSR1 still apply.)

#### ● Restart Sequence

A typical restart sequence for the HD6846 will include initialization of both the Peripheral Control & Data Direction Registers of the parallel port. It is necessary to set up the Peripheral Control Register first, since PCR7 = “0” is a condition for writing data into the Data Direction Register. (A logic zero at the external RES input automatically sets PCR7.)

#### ● Summary

The HD6846 has several optional modes of operation which allow it to be used in a variety of applications. The following tables are provided for reference in selecting these modes.

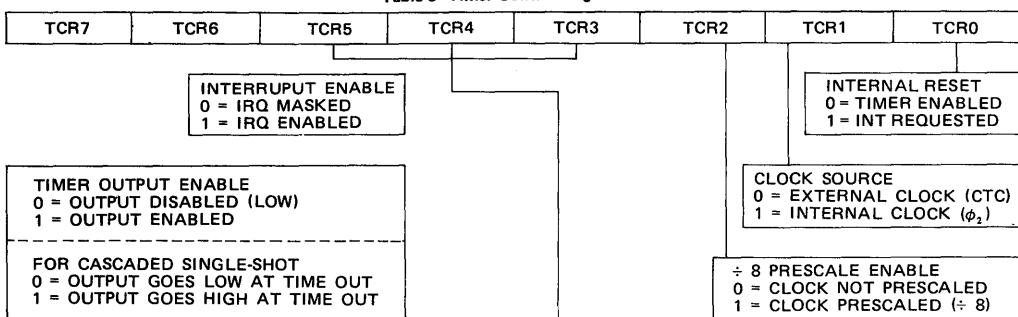
Table 7 HD6846 Internal Register Addresses

R/W	A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>	REGISTER SELECTED
R	0	0	0	Combination Status Register
R/W	0	0	1	Peripheral Control Register
R/W	0	1	0	Data Direction Register
R/W	0	1	1	Peripheral Data Register
R	1	0	0	Combination Status Register
R/W	1	0	1	Timer Control Register
R/W	1	1	0	Timer MSB Register
R/W	1	1	1	Timer LSB Register
R	x	x	x	ROM Address

Table 8 Composite Status Register

CSR7	CSR3~CSR6 NOT USED DEFAULT TO ZERO WHEN READ	CSR2	CSR1	CSR0
COMPOSITE INTERRUPT FLAG 0 = NO ENABLED INTERRUPT FLAG SET 1 = ONE OR MORE ENABLED INTERRUPT FLAGS SET*		CP <sub>2</sub> INTERRUPT FLAG 0 = NO INT REQ. 1 = INT REQUESTED		TIMER INTERRUPT FLAG 0 = NO INT REQ. 1 = INT REQUESTED
INVERSE OF THIS BIT APPEARS AT IRQ OUTPUT				
*STATUS OF THIS BIT CAN BE EXPRESSED AS: CSR7 = CSR0·TCR6 + CSR1·PCR0 + CSR2·PCR3				CP <sub>1</sub> INTERRUPT 0 = NO INT REQ. 1 = INT REQUESTED

Table 9 Timer Control Register



TCR3	TCR4	TCR5	TIME OPERATING MODE	COUNTER INITIALIZATION	INTERRUPT FLAG SET
0	0	0	CONTINUOUS	$\overline{CTG} \downarrow + W + R$	TO
0	0	1	CASCADED SINGLE SHOT	$\overline{CTG} \downarrow + W + R$	TO
0	1	0	CONTINUOUS	$\overline{CTG} \downarrow + R$	TO
0	1	1	NORMAL SINGLE SHOT	$CTG \downarrow + R$	TO
1	0	0	FREQUENCY COMPARISON	$CTG \downarrow \cdot \bar{T} \cdot (CE + TOF \cdot CE) + R$	$CTG \downarrow$ BEFORE TO
1	0	1		$CTG \downarrow \cdot \bar{T} + R$	TO BEFORE $CTG \downarrow$
1	1	0	PULSE WIDTH COMPARISON	$CTG \downarrow \cdot \bar{T} + R$	$CTG \downarrow$ BEFORE TO
1	1	1		$CTG \downarrow \cdot \bar{T} + R$	TO BEFORE $CTG \downarrow$

R = RESET CONDITION

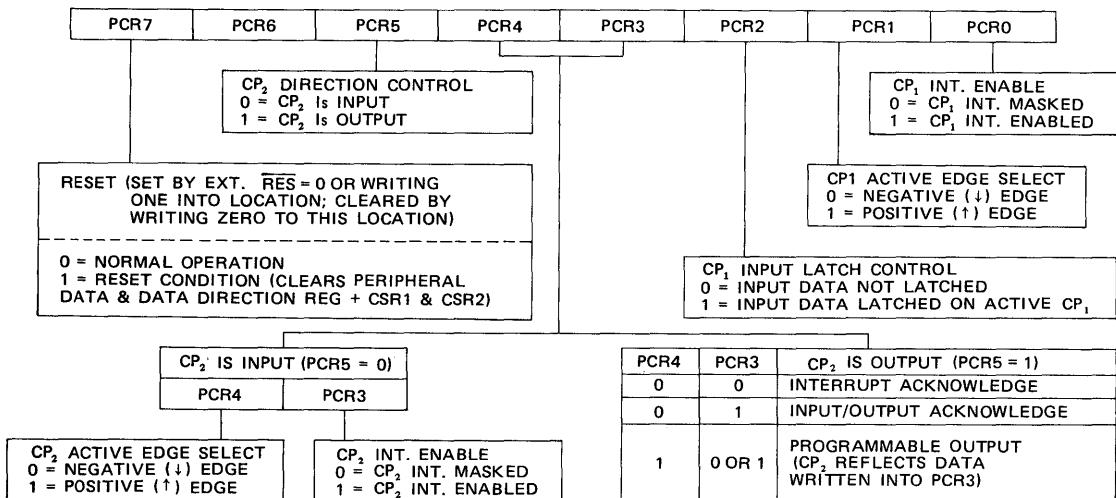
W = WRITE TIMER LATCHES

TO = COUNTER TIME OUT

CE = COUNTER ENABLE

 $\overline{CTG} \downarrow$  = NEG TRANSISTION OF PIN 17 $CTG \uparrow$  = POS TRANSITION OF PIN 17 $\bar{T}$  = INTERRUPT FLAG (CSR0) = 0

Table 10 Peripheral Control Register



### ■ CUSTOM PROGRAMMING

By the programming of a single photomask for the HD6846, the customer may specify the content of the memory and the method of enabling the outputs.

Information on the general options of the HD6846 should be submitted on an Organizational Data form such as that

ORGANIZATIONAL DATA HD6846 COMBINATION ROM-I/O-TIMER																																																																																															
Customer:																																																																																															
Company _____		Hitachi Use Only:																																																																																													
Part No. _____		Quote:																																																																																													
Originator _____		Part No.: _____																																																																																													
Phone No. _____		Specif. No.: _____																																																																																													
Enable Options: (ROM ENABLE MUST DIFFER FROM I/O-TIMER)																																																																																															
<table border="1"> <tr><td>CS<sub>0</sub></td><td>1</td><td>0</td><td>1</td><td>0</td></tr> <tr><td><input type="checkbox"/></td><td><input type="checkbox"/></td><td><input type="checkbox"/></td><td><input type="checkbox"/></td></tr> <tr><td>CS<sub>1</sub></td><td>1</td><td>0</td><td>1</td><td>0</td></tr> <tr><td><input type="checkbox"/></td><td><input type="checkbox"/></td><td><input type="checkbox"/></td><td><input type="checkbox"/></td></tr> <tr><td colspan="4">ROM SECTION</td><td colspan="4">I/O-TIMER SECTION</td></tr> </table>		CS <sub>0</sub>	1	0	1	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CS <sub>1</sub>	1	0	1	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	ROM SECTION				I/O-TIMER SECTION				<table border="1"> <tr> <th colspan="10">CHECK ONE COLUMN ONLY</th> </tr> <tr> <td colspan="4">I/O-TIMER SELECT</td> <td>A<sub>6</sub></td> <td>A<sub>10</sub></td> <td>1</td> <td>x</td> <td>x</td> <td>x</td> </tr> <tr> <td colspan="4"></td> <td>1</td> <td>x</td> <td>x</td> <td>1</td> <td>x</td> <td>x</td> </tr> <tr> <td colspan="4"></td> <td>0</td> <td>x</td> <td>x</td> <td>x</td> <td>1</td> <td>x</td> </tr> <tr> <td colspan="4"></td> <td>x</td> <td>x</td> <td>x</td> <td>x</td> <td>x</td> <td>1</td> </tr> <tr> <td colspan="4"></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </table>								CHECK ONE COLUMN ONLY										I/O-TIMER SELECT				A <sub>6</sub>	A <sub>10</sub>	1	x	x	x					1	x	x	1	x	x					0	x	x	x	1	x					x	x	x	x	x	1										
CS <sub>0</sub>	1	0	1	0																																																																																											
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																																																																																												
CS <sub>1</sub>	1	0	1	0																																																																																											
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																																																																																												
ROM SECTION				I/O-TIMER SECTION																																																																																											
CHECK ONE COLUMN ONLY																																																																																															
I/O-TIMER SELECT				A <sub>6</sub>	A <sub>10</sub>	1	x	x	x																																																																																						
				1	x	x	1	x	x																																																																																						
				0	x	x	x	1	x																																																																																						
				x	x	x	x	x	1																																																																																						
$1 \geq 2.0V$ $0 \leq 0.8V$ x = NOT USED																																																																																															

Figure 18 Format for Programming General Options

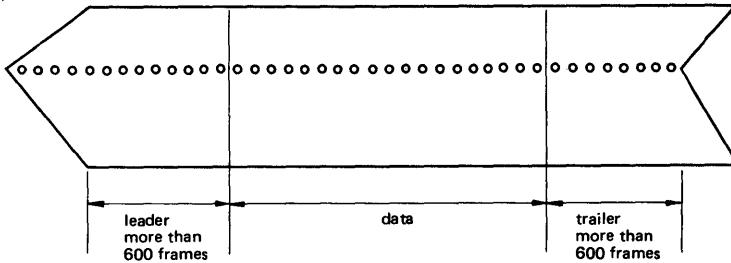
COMPANY		ENGINEER		SECTION		DATE	
CUSTOMERS P/N (if you need)				TYPE NO. OF ROM			
<input type="checkbox"/>							
DATA FORMAT							
1. HMCS6800 load module format		2. BNPF format					
coding media <input type="checkbox"/> 1. paper tape <input type="checkbox"/> 2. IBM 80 column card		total bytes of data (decimal)  initial ROM address (decimal)  parity (for paper tape) <input type="checkbox"/> 1. even <input type="checkbox"/> 2. odd <input type="checkbox"/> 3. none  total number of cards					
for HITACHI reference only		designed  <hr/> approved					
ref. No.	+						
mask ROM No.							
processed data							
approved data							

Figure 19 Confirmation sheet of specification for HD6846 series ROM

### ■ PAPER TAPE

- 1) Any one inch width tape usually available in market can be used but tape in black color is recommended.
- 2) Both leader and trailer have more than 600 frames.

(Example)



- 3) One file data of each chip shall be contained in one reel of paper tape. One file data shall not be divided into more than two reels.

### 4) Parity

Parity shall be indicated in "Confirmation sheet of specification". Parity forms are grouped;

- (1) With parity EVEN or ODD
- (2) Without parity

- 5) 8-bit ASCII code shall be used.

### ■ CARD

- 1) Use IBM 80 column card.
- 2) Use EBCDIC code.
- 3) Card format is as follows;
- 4) Total number of cards shall be written in "Confirmation sheet of specification".

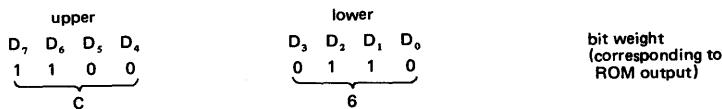
column	contents
1 to 71	Free format of data column
72	Blank
73 to 80	Sequential card number, not free format. Least significant digit of decimal sequential number is located in column 80. No alphabet letters. Any sequential number more than 1 can be used.

### ■ DATA FORMAT

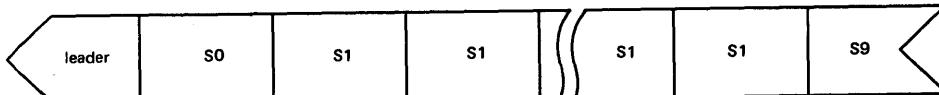
#### ● HMCS6800 LOAD MODULE FORMAT

- This is object format obtained from HMCS6800 assembler.
- 1) 8-bit code is divided into upper and lower 4 bits and transformed into hexadecimal number.

(Example) Binary number if 1100 0010 is transformed as follows.

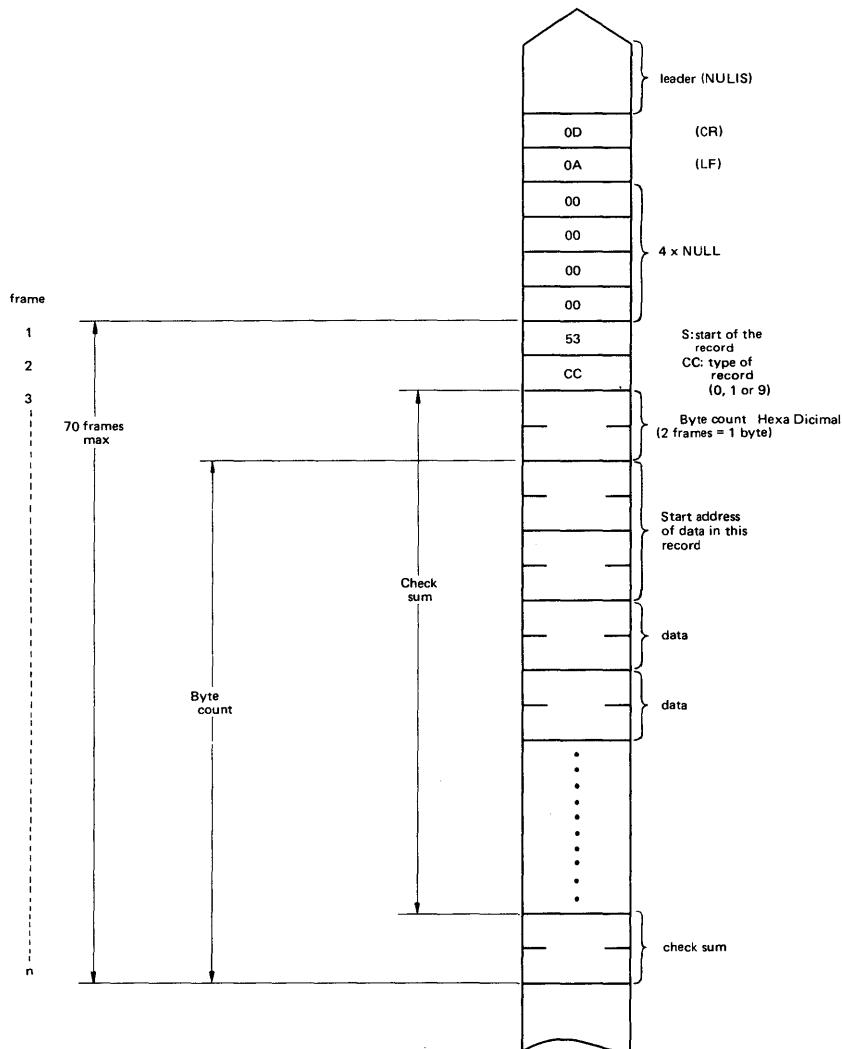
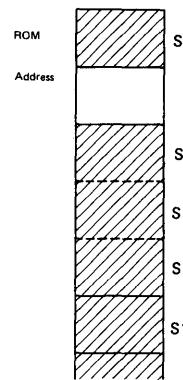


- 2) Load module structure of paper tape is shown as an example.



S0 is the header record, S1 is data record and S9 is end of file record. Each data record corresponds to each ROM data as shown below. Continuous memory address shall be divided into several records due to limitation of maximum frame number (70 frames = 35 bytes) in one record.

S0, S1 or S9 is distinguished by CC following start of the record S.



Check sum is complement of 1 for sum of each 8-bit.

## 3) Example of load module format

	frame	CC=30 header record	CC=31 data record	CC=39 end of file record	
1	start of record	53	S	53	S
2	type of record	30	0	31	1
3	byte count	30	06	31	39
4		36		31	30
5				36	33
6	start address of			31	30
7	data in this			31	30
8	record		0000	30	30
9				30	30
10	data	30		1100	0000
		30			
		30			
		30			
n	data	34	48-H	39	46
		38		38	43
		34	44-D	30	
		34		32	
		35	52-R		
		32			
	check	32	2B	41	A8
	sum	42	(check sum)	38	(check sum)

Check sum of header record above is complement of 1 of  $(06 + 00 + 00 + 48 + 44 + 52)_{16}$  i.e., 2B.

The start address of data record is incremented for each one byte data, then is compared to the next address in data record and is checked to be sequential or not.

When it is not sequential, hexadecimal 00 is filled as data for that address automatically.

A example of type out of paper tape in HMCS6800 load module format is shown below.

```
header record ...S00600004844522B
data record .....S113F0007EF5587EF7897EFAA77EF9C07EF9C47E24
data record .....S112F010FA657EFA8B7EFAA07EF9DC7EFA247E06
end of file record ....S9030000FC
```

4) Four types of data of ROM code are able to be processed. In any case, header record before data record is needed and so as end of file record after data record.

(a) No vacancy in ROM

Data record is filled with full ROM record of one chip. Therefore address is sequential. Initial ROM address in "Confirmation sheet of specification" is 0.

(b) Vacancy in former part of ROM

Desired initial address shall be filled in initial ROM address column in "Confirmation sheet of specification". Data of 00 are filled automatically for vacant address.

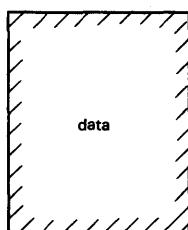
(c) Vacancy in the middle of ROM

Data of 00 are filled in for vacant address. Initial ROM address for data I is 0 and desired initial address for data II shall be described in "Confirmation sheet of specification".

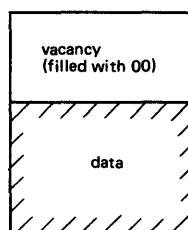
(d) Vacancy in later part of ROM

When end of file record is read out, data of 00 are filled in thereafter.

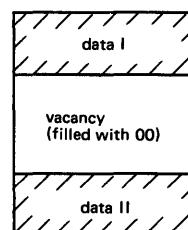
ROM



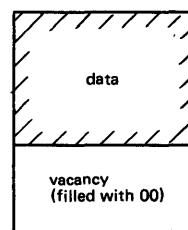
(a)



(b)

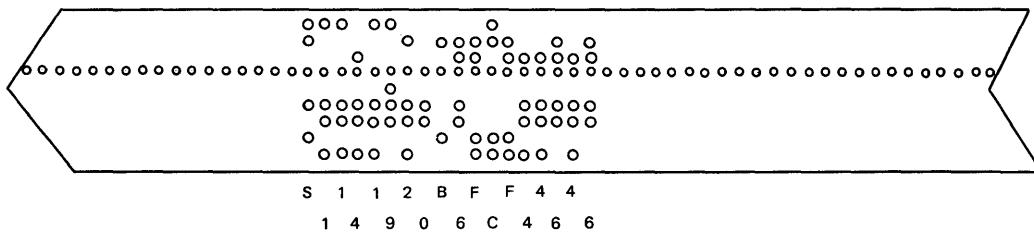


(c)



(d)

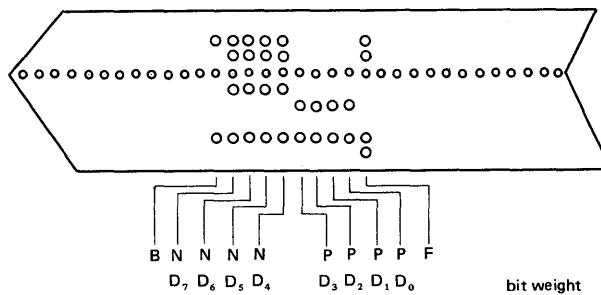
(Example) Paper tape whose data record is S1141920B6FC .....



- **BNPF format**

- 1) Each word is expressed as BNPF slice which begins word opening mark B, has 8 character bit contents shown by P or N and finishes with end mark F.

(Example) 0F in hexadecimal code is expressed as shown below (paper tape).



- 2) Any contents between F of the first slice and B of next slice are disregarded.
- 3) Bit pattern (BNPF) slice for all ROM address shall be indicated. Initial ROM address in "Confirmation sheet of specification" is, therefore always 0 for BNPF.

B ..... shows beginning of the word  
 N ..... shows 0 of one bit data  
 P ..... shows 1 of one bit data  
 F ..... shows end of the word

- Note 1) X can be used except for P and N for indication of word contents of BNPF slice. This X means that bit can be either P or N (don't care). X shall be determined by HITACHI for testing and shall be

Note 2) informed to the customer in confirmation table. Expression of B\*nF can be used for indicating that the same contents of foregoing slice are applicable from this word to following n words.

For example, when B\*4F is indicated at 10th word position, the contents of 9th word are repeated for 10, 11, 12 and 13th word.

(Content of X is not always repeated even in this case.)

Note 3) n is greater than 1 and less than final address of ROM. When vacancy of ROM exists, combination of Note 1) and Note 2) is useful.

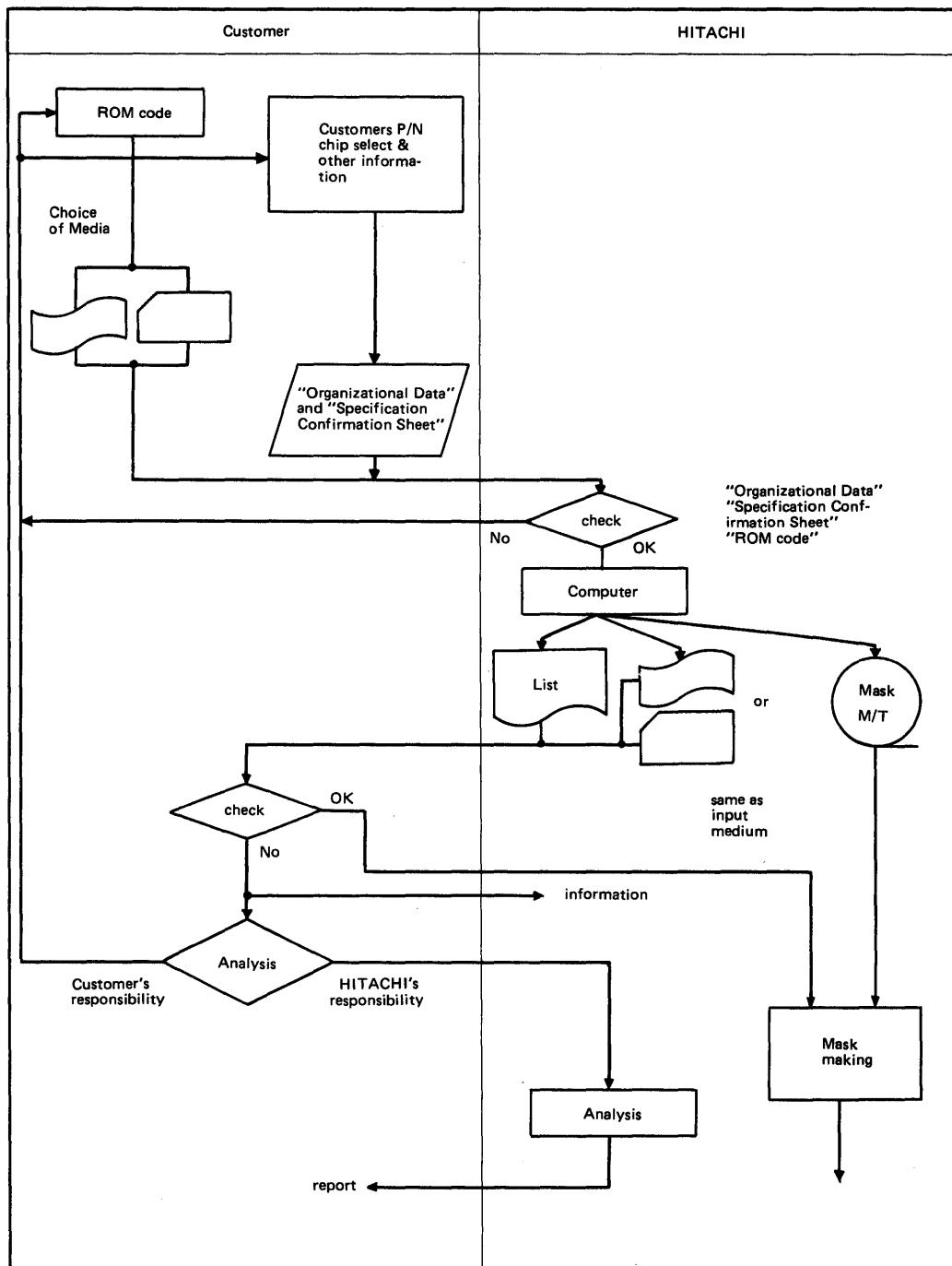


Figure 20 Flow chart of Mask ROM Development

# HD6850, HD68A50

## ACIA (Asynchronous Communication Interface Adapter)

The HD6850 Asynchronous Communications Interface Adapter provides the data formatting and control to interface serial asynchronous data communications information to bus organized systems such as the HMCS6800 Microprocessing Unit.

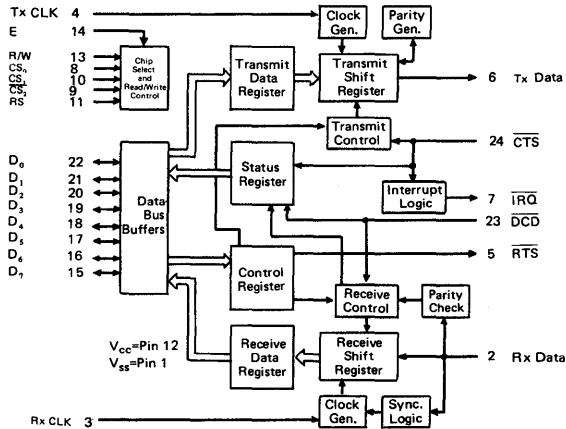
The bus interface of the HD6850 includes select, enable, read/write, interrupt and bus interface logic to allow data transfer over an 8-bit bi-directional data bus. The parallel data of the bus system is serially transmitted and received by the asynchronous data interface, with proper formatting and error checking.

The functional configuration of the ACIA is programmed via the data bus during system initialization. A programmable Control Register provides variable word lengths, clock division ratios, transmit control, receive control, and interrupt control. For peripheral or modem operation three control lines are provided.

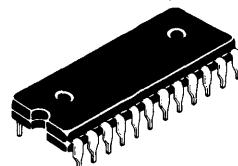
### ■ FEATURES

- Serial/Parallel Conversion of Data
- Eight and Nine-bit Transmission
- Insertion and Deleting of Start and Stop Bit
- Optional Even and Odd Parity
- Parity, Overrun and Framing Error Checking
- Peripheral/Modem Control Functions (Clear to Send CTS, Request to Send RTS, Data Carrier Detect DCD)
- Optional  $\div 1$ ,  $\div 16$ , and  $\div 64$  Clock Modes
- Up to 500kbps Transmission
- Programmable Control Register
- N-channel Silicon Gate Process
- Compatible with MC6850 and MC68A50

### ■ BLOCK DIAGRAM

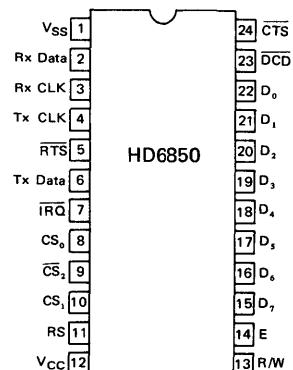


HD6850P, HD68A50P



(DP-24)

### ■ PIN ARRANGEMENT



(Top View)

### ■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	$V_{CC}^*$	-0.3 ~ +7.0	V
Input Voltage	$V_{IN}^*$	-0.3 ~ +7.0	V
Operating Temperature	$T_{opr}$	-20 ~ +75	°C
Storage Temperature	$T_{stg}$	-55 ~ +150	°C

\* With respect to  $V_{SS}$  (SYSTEM GND)

(NOTE) 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.

### ■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol	min	typ	max	Unit
Supply Voltage	$V_{CC}^*$	4.75	5	5.25	V
Input Voltage	$V_{IL}^*$	-0.3	—	0.8	V
	$V_{IH}^*$	2.0	—	$V_{CC}$	V
Operating Temperature	$T_{opr}$	-20	25	75	°C

\* With respect to  $V_{SS}$  (SYSTEM GND)

### ■ ELECTRICAL CHARACTERISTICS

• DC CHARACTERISTICS ( $V_{CC}=5V\pm 5\%$ ,  $V_{SS}=0V$ ,  $T_a=-20\sim+75^\circ C$ , unless otherwise noted.)

Item	Symbol	Test Condition	min	typ*	max	Unit
Input "High" Voltage	All Inputs	$V_{IH}$	2.0	—	$V_{CC}$	V
Input "Low" Voltage	All Inputs	$V_{IL}$	-0.3	—	0.8	V
Input Leakage Current	R/W, CS <sub>0</sub> , CS <sub>1</sub> , CS <sub>2</sub> , E	$I_{in}$	$V_{in}=0\sim 5.25V$	—	—	$\mu A$
Three-State (Off State) Input Current	D <sub>0</sub> ~D <sub>7</sub>	$I_{TSI}$	$V_{in}=0.4\sim 2.4V$	—	—	$\mu A$
Output "High" Voltage	D <sub>0</sub> ~D <sub>7</sub>	$V_{OH}$	$I_{OH}=-20\mu A$ , Enable Pulse Width $\leq 25\mu s$	2.4	—	—
	TxDATA, RTS		$I_{OH}=-10\mu A$ , Enable Pulse Width $\leq 25\mu s$	2.4	—	—
Output "Low" Voltage	All outputs	$V_{OL}$	$I_{OL}=1.6mA$ , Enable Pulse Width $\leq 25\mu s$	—	—	0.4
Output Leakage Current (Off State)	IRQ	$I_{LOH}$	$V_{OH}=2.4V$	—	—	10
Power Dissipation	$P_D$		—	300	525	mW
Input Capacitance	D <sub>0</sub> ~D <sub>7</sub>	$C_{in}$	$V_{in}=0V$ , $T_a=25^\circ C$ , $f=1.0MHz$	—	—	12.5
	E, TxCLK, RxCLK, R/W, RS, RxData, CS <sub>0</sub> , CS <sub>1</sub> , CS <sub>2</sub> , CTS, DCD			—	—	7.5
Output Capacitance	RTS, TxDATA	$C_{out}$	$V_{in}=0V$ , $T_a=25^\circ C$ , $f=1.0MHz$	—	—	10
	IRQ			—	—	5.0

\*  $T_a=25^\circ C$ ,  $V_{CC}=5V$

**● AC CHARACTERISTICS**

**1. TIMING OF DATA TRANSMISSION**

Item	Symbol	Test Condition	min	typ	max	Unit
Minimum Clock Pulse Width	$\text{PW}_{\text{CL}}$	Fig. 1	600	—	—	ns
	$\text{PW}_{\text{CH}}$	Fig. 2	600	—	—	ns
Clock Frequency	$f_c$	—	—	—	500	kHz
		—	—	—	800	
Clock-to-Data Delay for Transmitter	$t_{\text{TDD}}$	Fig. 3	—	—	1.0	$\mu\text{s}$
Receive Data Setup Time	$t_{\text{RDSU}}$	Fig. 4	500	—	—	ns
Receive Data Hold Time	$t_{\text{RDH}}$	Fig. 5	500	—	—	ns
IRQ Release Time	$t_{\text{IR}}$	Fig. 6	—	—	1.2	$\mu\text{s}$
RTS Delay Time	$t_{\text{RTS}}$	Fig. 6	—	—	1.0	$\mu\text{s}$
Rise Time and Fall Time	Except E	$t_r, t_f$	—	—	1.0*	$\mu\text{s}$

\* 1.0  $\mu\text{s}$  or 10% of the pulse width, whichever is smaller.

**2. BUS TIMING CHARACTERISTICS**

**1) READ**

Item	Symbol	Test Condition	HD6850			HD68A50			Unit
			min	typ	max	min	typ	max	
Enable Cycle Time	$t_{\text{cycE}}$	Fig. 7	1.0	—	—	0.666	—	—	$\mu\text{s}$
Enable "High" Pulse Width	$\text{PW}_{\text{EH}}$	Fig. 7	0.45	—	25	0.28	—	25	$\mu\text{s}$
Enable "Low" Pulse Width	$\text{PW}_{\text{EL}}$	Fig. 7	0.43	—	—	0.28	—	—	$\mu\text{s}$
Setup Time, Address and R/W valid to Enable positive transition	$t_{\text{AS}}$	Fig. 7	140	—	—	140	—	—	ns
Data Delay Time	$t_{\text{DDR}}$	Fig. 7	—	—	320	—	—	220	ns
Data Hold Time	$t_H$	Fig. 7	10	—	—	10	—	—	ns
Address Hold Time	$t_{\text{AH}}$	Fig. 7	10	—	—	10	—	—	ns
Rise and Fall Time for Enable Input	$t_{\text{Er}}, t_{\text{Ef}}$	Fig. 7	—	—	25	—	—	25	ns

**2) WRITE**

Item	Symbol	Test Condition	HD6850			HD68A50			Unit
			min	typ	max	min	typ	max	
Enable Cycle Time	$t_{\text{cycE}}$	Fig. 8	1.0	—	—	0.666	—	—	$\mu\text{s}$
Enable "High" Pulse Width	$\text{PW}_{\text{EH}}$	Fig. 8	0.45	—	25	0.28	—	25	$\mu\text{s}$
Enable "Low" Pulse Width	$\text{PW}_{\text{EL}}$	Fig. 8	0.43	—	—	0.28	—	—	$\mu\text{s}$
Setup Time, Address and R/W valid to Enable positive transition	$t_{\text{AS}}$	Fig. 8	140	—	—	140	—	—	ns
Data Setup Time	$t_{\text{DSW}}$	Fig. 8	195	—	—	80	—	—	ns
Data Hold Time	$t_H$	Fig. 8	10	—	—	10	—	—	ns
Address Hold Time	$t_{\text{AH}}$	Fig. 8	10	—	—	10	—	—	ns
Rise and Fall Time for Enable Input	$t_{\text{Er}}, t_{\text{Ef}}$	Fig. 8	—	—	25	—	—	25	ns

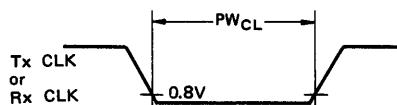


Figure 1 Clock Pulse Width, "Low" State

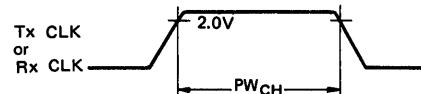


Figure 2 Clock Pulse Width, "High" State

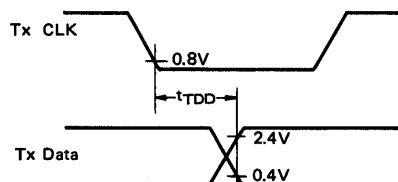


Figure 3 Transmit Data Output Delay

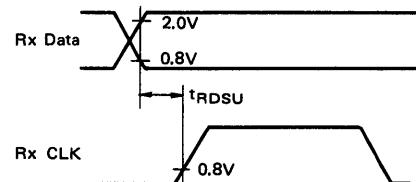


Figure 4 Receive Data Setup Time ( $\div 1$  Mode)

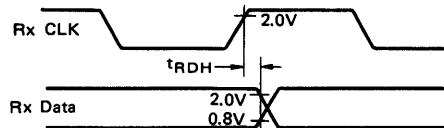


Figure 5 Receive Data Hold Time ( $\div 1$  Mode)

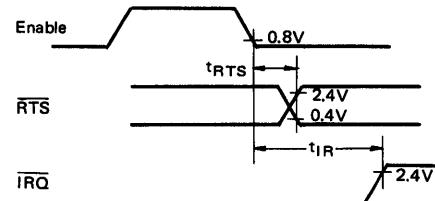


Figure 6  $\overline{\text{RTS}}$  Delay and  $\overline{\text{IRQ}}$  Release Time

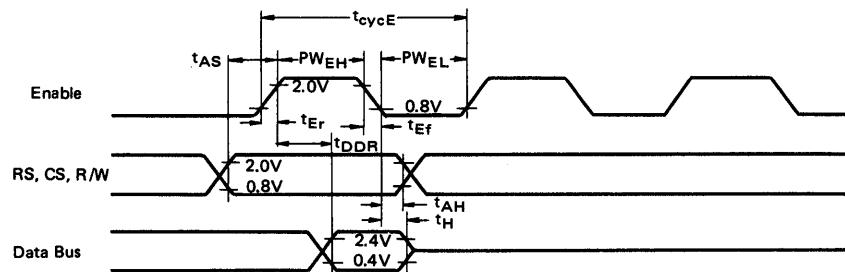


Figure 7 Bus Read Timing Characteristics  
(Read information from ACIA)

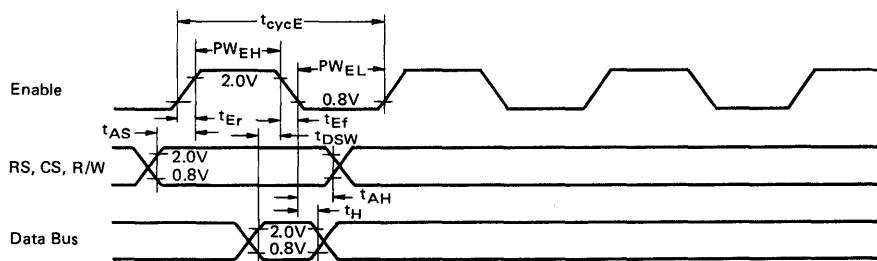
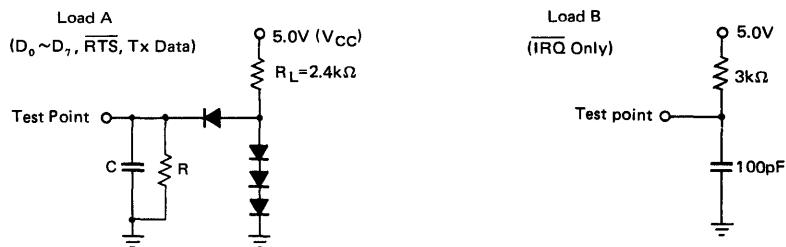


Figure 8 Bus Write Timing Characteristics  
(Write information into ACIA)



C = 130pF for  $D_0 \sim D_7$ ,  
= 30pF for RTS and Tx Data  
All diodes are 1S2074 (H) or Equiv.

R = 11kΩ for  $D_0 \sim D_7$ ,  
= 24kΩ for RTS and Tx Data

Figure 9 Bus Timing Test Loads

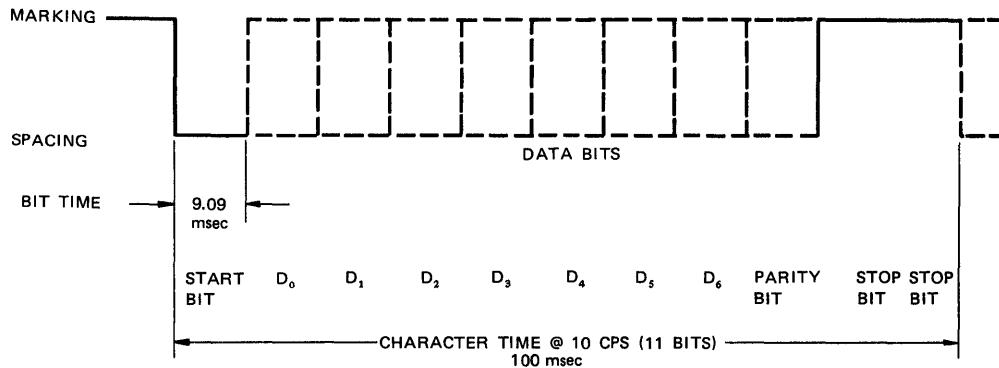


Figure 10 110 Baud Serial ASCII Data Timing

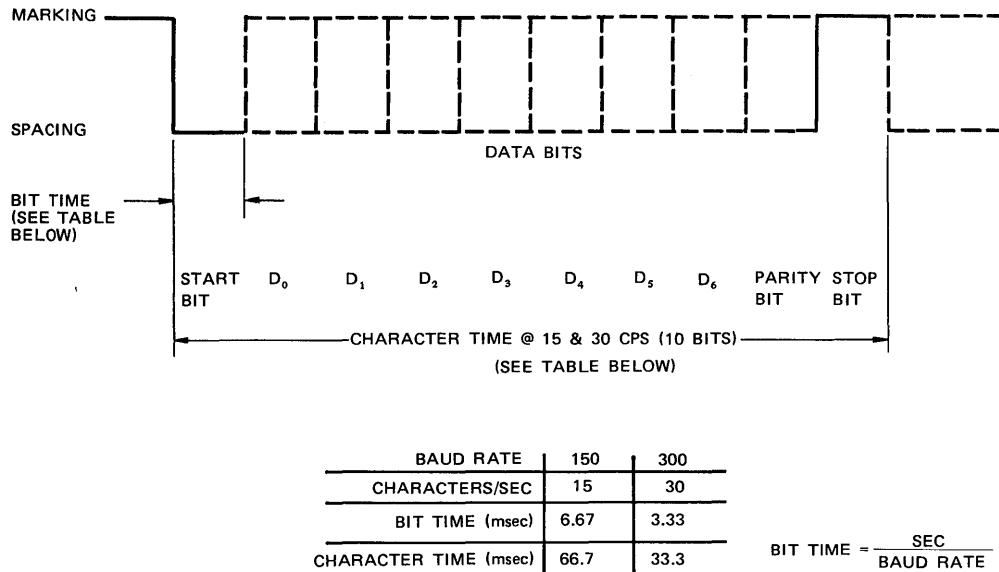
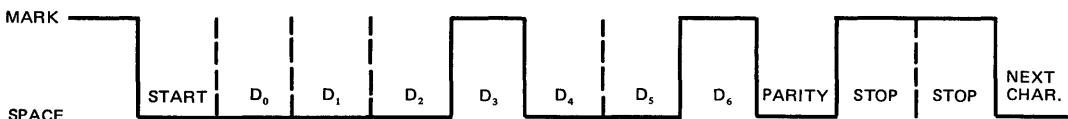


Figure 11 150 &amp; 300 Baud Serial ASCII Data Timing

Figure 12 Send a 7 Bit ASCII Char. "H" Even Parity  
— 2 Stop Bits H = 48<sub>16</sub> = 1001000<sub>2</sub>

## ■ DATA OF ACIA

HD6850 is an interface adapter which controls transmission and reception of Asynchronous serial data. Some examples of serial data are shown in Figs. 10 ~ 12.

## ■ INTERNAL STRUCTURE OF ACIA

HD6850(ACIA) provides the following; 8-bit Bi-directional Data Buses ( $D_0 \sim D_7$ ), Receive Data Input (Rx Data), Transmit Data Output (Tx Data), three Chip Selects ( $CS_0$ ,  $CS_1$ ,  $\overline{CS}_2$ ), Register Select Input (RS), Two Control Input (Read/Write (R/W), Enable(E)), Interrupt Request Output( $\overline{IRQ}$ ), Clear-to-Send (CTS) to control the modem, Request-to-Send ( $\overline{RTS}$ ), Data Carrier Detect(DCD) and Clock Inputs(Tx CLK, Rx CLK) used for synchronization of received and transmitted data. This ACIA also provides four registers; Status Register, Control Register, Receive Register and Transmit Register.

24-pin dual-in-line type package is used for the ACIA. Internal Structure of ACIA is illustrated in Fig. 13.

## ■ ACIA OPERATION

### • Master Reset

The master reset (CR0, CR1) should be set during system initialization to insure the reset condition and prepare for programming the ACIA functional configuration when the communications channel is required. Control bits CR5 and CR6 should also be programmed to define the state of  $\overline{RTS}$  whenever master reset is utilized. After master resetting the ACIA, the programmable Control Register can be set for a number of options such as variable clock divider ratios, variable word length, one or two stop bits, parity (even, odd, or none), etc.

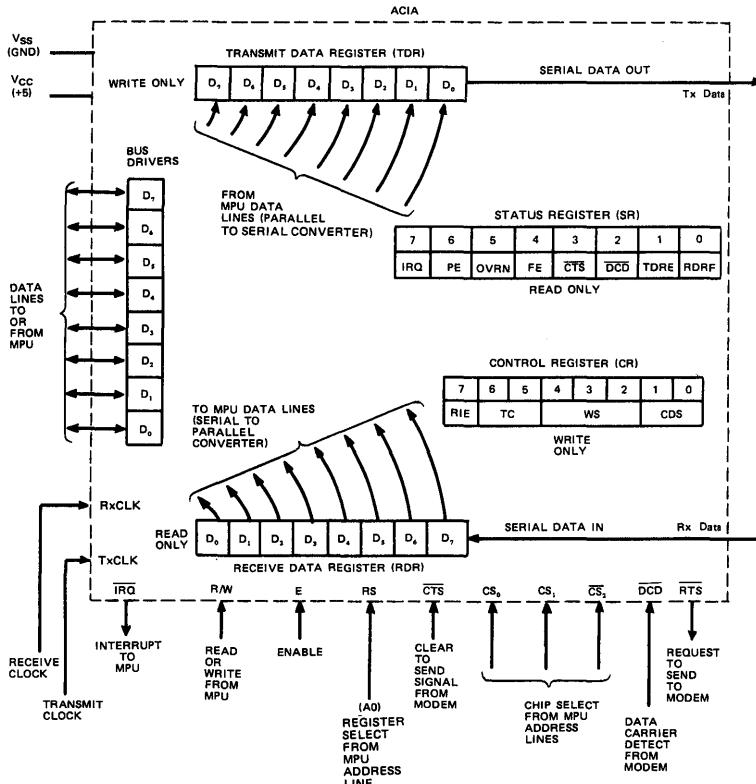


Figure 13 Internal Structure of ACIA

#### • Transmit

A typical transmitting sequence consists of reading the ACIA Status Register either as a result of an interrupt or in the ACIA's turn in a polling sequence. A character may be written into the Transmit Data Register if the status read operation has indicated that the Transmit Data Register is empty. This character is transferred to a Shift Register where it is serialized and transmitted from the Transmit Data output preceded by a start bit and followed by one or two stop bits. Internal parity (odd or even) can be optionally added to the character and will occur between the last data bit and the first stop bit. After the first character is written in the Data Register, the Status Register can be read again to check for a Transmit Data Register Empty condition and current peripheral status. If the register is empty, another character can be loaded for transmission even though the first character is in the process of being transmitted (because of double buffering). The second character will be automatically transferred into the Shift Register when the first character transmission is completed. This sequence continues until all the characters have been transmitted.

#### • Receive

Data is received from a peripheral by means of the Receive Data input. A divide-by-one clock ratio is provided for an externally synchronized clock (to its data) while the divide-by-16 and 64 ratios are provided for internal synchronization. Bit synchronization in the divide-by-16 and 64 modes is initiated by

the detection of the leading mark-space transition of the start bit. False start bit detection capability insures that a full half bit of a start bit has been received before the internal clock is synchronized to the bit time. As a character is being received, parity (odd or even) will be checked and the error indication will be available in the Status Register along with framing error, overrun error, and Receive Data Register full. In a typical receiving sequence, the Status Register is read to determine if a character has been received from a peripheral. If the Receiver Data Register is full, the character is placed on the 8-bit ACIA bus when a Read Data command is received from the MPU. When parity has been selected for an 8-bit word (7 bits plus parity), the receiver strips the parity bit ( $D_7 = "0"$ ) so that data alone is transferred to the MPU. This feature reduces MPU programming. The Status Register can continue to be read again to determine when another character is available in the Receive Data Register. The receiver is also double buffered so that a character can be read from the data register as another character is being received in the Shift register. The above sequence continues until all characters have been received.

#### ■ ACIA INTERNAL REGISTERS

The ACIA provides four registers; Transmit Data Register (TDR), Receive Data Register(RDR), Control Register(CR) and Status Register(SR). The content of each of the registers is summarized in Table 1.

Table 1 Definition of ACIA Register Contents

Buffer Address	****	RS=1 • R/W=0	RS=1 • R/W=1	RS=0 • R/W=0	RS=0 • R/W=1
Data Bus	Transmit Data Register	Receiver Data Register	Control Register	Status Register	
	(Write Only)	(Read Only)	(Write Only)	(Read Only)	
0	Data Bit 0*	Data Bit 0	Counter Divide Select (CR0)	Rx Data Reg. Full (RDRF)	
1	Data Bit 1	Data Bit 1	Counter Divide Select (CR1)	Tx Data Reg. Empty (TDRE)	
2	Data Bit 2	Data Bit 2	Word Select 1 (CR2)	Data Carrier Detect (DCD)	
3	Data Bit 3	Data Bit 3	Word Select 2 (CR3)	Clear to Send (CTS)	
4	Data Bit 4	Data Bit 4	Word Select 3 (CR4)	Framing Error (FE)	
5	Data Bit 5	Data Bit 5	Tx Control 1 (CR5)	Overrun (OVRN)	
6	Data Bit 6	Data Bit 6	Tx Control 2 (CR6)	Parity Error (PE)	
7	Data Bit 7***	Data Bit 7**	Rx Interrupt Enable (CR7)	Interrupt Request (IRQ)	

\* Leading bit = LSB = Bit 0

\*\* Data bit will be zero in 7-bit plus parity modes.

\*\*\* Data bit is "don't care" in 7-bit plus parity modes.

\*\*\*\* 1 ... "High" level, 0 ... "Low" level

#### • Transmit Data Register (TDR)

Data is written in the Transmit Data Register during the negative transition of the enable (E) when the ACIA has been addressed and RS • R/W is selected. Writing data into the register causes the Transmit Data Register Empty bit in the Status Register to go "0". Data can then be transmitted. If the transmitter is idling and no character is being transmitted, then the transfer will take place within one bit time of the trailing edge of the Write command. If a character is being transmitted, the new data character will commence as soon as the previous character is complete. The transfer of data causes the Transmit Data Register Empty (TDRE) bit to indicate empty.

#### • Receive Data Register (RDR)

Data is automatically transferred to the empty Receive Data Register (RDR) from the receiver deserializer (a shift register) upon receiving a complete character. This event causes the Receive Data Register Full bit (RDRF) on the status buffer to go "1" (full). Data may then be read through the bus by addressing the ACIA and R/W "High" when the ACIA is enabled. The non-destructive read cycle causes the RDRF bit to be cleared to empty although the data is retained in the RDR. The status is maintained by RDRF as to whether or not the data is current. When the Receive Data Register is full, the automatic transfer of data from the Receiver Shift Register to the Data Register is inhibited and the RDR contents remain valid with its current status stored in the Status Register.

#### • Control Register

The ACIA Control Register consists of eight bits of write-only buffer that are selected when RS and R/W are "Low". This register controls the function of the receiver, transmitter,

interrupt enables, and the Request-to-Send (RTS) peripheral/modem control output.

#### Counter Divide Select Bits (CR0 and CR1)

The Counter Divide Select Bits (CR0 and CR1) determine the divide ratios utilized in both the transmitter and receiver section of the ACIA. Additionally, these bits are used to provide a master reset for the ACIA which clears the Status Register (except for external conditions on CTS and DCD) and initializes both the receiver and transmitter. Master reset does not affect other Control Register bits. Note that after power-on or a power fail/restart, these bits must be set "1" to reset the ACIA. After resetting, the clock divide ratio may be selected. These counter select bits provide for the following clock divide ratios:

Table 2 Function of Counter Devide Select Bit

CR1	CR0	Function
0	0	÷1
0	1	÷16
1	0	÷64
1	1	Master Reset

#### Word Select Bits (CR2, CR3, and CR4)

The Word Select bits are used to select word length, parity, and the number of stop bits. The encoding format is as follows:

Table 3 Function of Word Select Bit

CR4	CR3	CR2	Function
0	0	0	7 Bits + Even Parity + 2 Stop Bits
0	0	1	7 Bits + Odd Parity + 2 Stop Bits
0	1	0	7 Bits + Even Parity + 1 Stop Bit
0	1	1	7 Bits + Odd Parity + 1 Stop Bit
1	0	0	8 Bits + 2 Stop Bits
1	0	1	8 Bits + 1 Stop Bit
1	1	0	8 Bits + Even Parity + 1 Stop Bit
1	1	1	8 Bits + Odd Parity + 1 Stop Bit

Word length, Parity Select, and Stop Bit changes are not buffered and therefore become effective immediately.

#### Transmitter Control Bits (CR5 and CR6)

Two Transmitter Control bits provide for the control of the interrupt from the Transmit Data Register Empty condition, the Request-to-Send (RTS) output, and the transmission of a Break level (space). The following encoding format is used:

Table 4 Function of Transmitter Control-Bit

CR6	CR5	Function
0	0	RTS = "Low", Transmitting Interrupt Disabled.
0	1	RTS = "Low", Transmitting Interrupt Enabled.
1	0	RTS = "High", Transmitting Interrupt Disabled.
1	1	RTS = "Low", Transmits a Break level on the Transmit Data Output. Transmitting Interrupt Disabled.

#### Receive Interrupt Enable Bit (CR7)

The following interrupts will be enabled by a "1" in bit position 7 of the Control Register (CR7): Receive Data Register Full, Overrun, or a "Low" to "High" transition on the Data Carrier Detect (DCD) signal line.

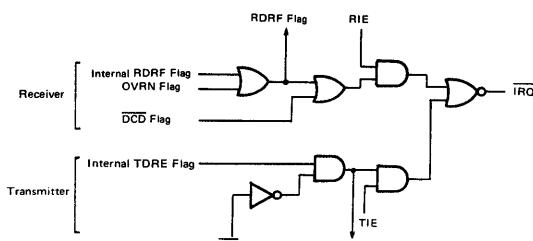


Fig. 14 IRQ Internal Circuit

#### ● Status Register

Information on the status of the ACIA is available to the MPU by reading the ACIA Status Register. This read-only register is selected when RS is "Low" and R/W is "High". Information stored in this register indicates the status of the Transmit Data Register, the Receive Data Register and error logic, and the peripheral/modem status inputs of the ACIA.

#### Receive Data Register Full (RDRF), Bit 0

RDRF indicates that received data has been transferred to the Receive Data Register. RDRF is cleared after an MPU read of the Receive Data Register or by a master reset. The cleared or empty state indicates that the contents of the Receive Data Register are not current. Data Carrier Detect (DCD) being "High" also causes RDRF to indicate empty.

#### Transmit Data Register Empty (TDRE), Bit 1

The Transmit Data Register Empty bit being set "1" indicates that the Transmit Data Register contents have been transferred and that new data may be entered. The "0" state indicates that the register is full and that transmission of a new character has not begun since the last write data command.

#### Data Carrier Detect (DCD), Bit 2

The DCD bit will be "1" when the DCD input from a modem has gone "High" to indicate that a carrier is not present. This bit going "1" causes an Interrupt Request to be generated when the Receive Interrupt Enable is set. It remains "1" after the DCD input is returned "Low" until cleared by first reading the Status Register and then the Data Register or until a master reset occurs. If the DCD input remains "High" after read status and read data or master reset has occurred, the interrupt is cleared, the DCD status bit remains "1" and will follow the DCD input.

#### Clear-to-Send (CTS), Bit 3

The CTS bit indicates the state of the CTS input from a modem. A "Low" CTS indicates that there is a CTS from the modem. In the "High" state, the Transmit Data Register Empty bit is inhibited and the CTS status bit will be "1". Master reset does not affect the Clear-to-Send Status bit.

#### Framing Error (FE), Bit 4

FE indicates that the received character is improperly framed by a start and a stop bit and is detected by the absence of the 1st stop bit. This error indicates a synchronization error, faulty transmission, or a break condition. The FE flag is set or reset during the receive data transfer time. Therefore, this error indicator is present throughout the time that the associated character is available.

#### Receiver Overrun (OVRN), Bit 5

Overrun is an error flag that indicates that one or more characters in the data stream were lost. That is, a character or a number of characters were received but not read from the Receive Data Register (RDR) prior to subsequent characters being received. The overrun condition begins at the midpoint of the last bit of the second character received in succession without a read of the RDR having occurred. The overrun does not occur in the Status Register until the valid character prior to Overrun has been read. The RDRF bit remains set until the Overrun is reset. Character synchronization is maintained during the Overrun condition. The Overrun indication is reset after the reading of data from the Receive Data Register or by a Master Reset.

**Parity Error (PE), Bit 6**

The PE flag indicates that the number of "1"s (highs) in the character does not agree with the preselected odd or even parity. Odd parity is defined to be when the total number of ones is odd. The parity error indication will be present as long as the data character is in the RDR. If no parity is selected, then both the transmitter parity generator output and the receiver parity check results are inhibited.

**Interrupt Request (IRQ), Bit 7**

The IRQ bit indicates the state of the  $\overline{IRQ}$  output. Any interrupt condition with its applicable enable will be indicated in this status bit. Anytime the  $\overline{IRQ}$  output is "Low" the IRQ bit will be "1" to indicate the interrupt or service request status. IRQ is cleared by a read operation to the Receive Data Register or a write operation to the Transmit Data Register.

**■ SIGNAL FUNCTIONS****● Interface Signal for MPU****Bi-Directional Data Bus ( $D_0 \sim D_7$ )**

The bi-directional data bus ( $D_0 \sim D_7$ ) allow for data transfer between the ACIA and the MPU. The data bus output drivers are three-state devices that remain in the high impedance (off) state except when the MPU performs an ACIA read operation.

**Enable (E)**

The Enable signal, E, is a high impedance TTL compatible input that enables the bus input/output data buffers and clocks data to and from the ACIA. This signal will normally be a derivative of the HMCS6800  $\phi_2$  Clock.

**Read/Write (R/W)**

The R/W line is a high impedance input that is TTL compatible and is used to control the direction of data flow through the ACIA's input/output data bus interface. When R/W is "High" (MPU Read cycle), ACIA output drivers are turned on and a selected register is read. When it is "Low", the ACIA output drivers are turned off and the MPU writes into a selected register. Therefore, the R/W signal is used to select read-only or write-only registers within the ACIA.

**Chip Select ( $CS_0, CS_1, \overline{CS}_2$ )**

These three high impedance TTL compatible input lines are used to address the ACIA. The ACIA is selected when  $CS_0$  and  $CS_1$  are "High" and  $\overline{CS}_2$  is "Low". Transfers of data to and from the ACIA are then performed under the control of the Enable signal, Read/Write, and Register Select.

**Register Select (RS)**

The RS line is a high impedance input that is TTL compatible. A "High" level is used to select the Transmit/Receive Data Registers and a "Low" level the Control/Status Registers. The R/W signal line is used in conjunction with Register Select to select the read-only or write-only register in each register pair.

**Interrupt Request ( $\overline{IRQ}$ )**

$\overline{IRQ}$  is a TTL compatible, open-drain (no internal pullup), active "Low" output that is used to interrupt the MPU. The  $\overline{IRQ}$  output remains "Low" as long as the cause of the interrupt is present and the appropriate interrupt enable within the ACIA is set.

**Clock Inputs**

Separate high impedance TTL compatible inputs are provided for clocking of transmitted and received data. Clock frequencies of 1, 16 or 64 times the data rate may be selected.

**Transmit Clock (Tx CLK)**

The Tx CLK input is used for the clocking of transmitted data. The transmitter initiates data on the negative transition of the clock.

**Receive Clock (Rx CLK)**

The Rx CLK input is used for synchronization of received data. (In the  $\div 1$  mode, the clock and data must be synchronized externally.) The receiver samples the data on the positive transition of the clock.

**● Serial Input/Output Lines****Receive Data (Rx Data)**

The Rx Data line is a high impedance TTL compatible input through which data is received in a serial format. Synchronization with a clock for detection of data is accomplished internally when clock rates of 16 or 64 times the bit rate are used. Data rates are in the range of 0 to 500 kbps when external synchronization is utilized.

**Transmit Data (Tx Data)**

The Tx Data output line transfers serial data to a modem or other peripheral. Data rates in the range of 0 to 500 kbps when external synchronization is utilized.

**Modem Control**

The ACIA includes several functions that permit limited control of a peripheral or modem. The functions included are  $\overline{CTS}$ ,  $\overline{RTS}$  and  $\overline{DCD}$ .

**Clear-to-Send (CTS)**

This high impedance TTL compatible input provides automatic control of the transmitting end of a communications link via the modem  $\overline{CTS}$  active "Low" output by inhibiting the Transmit Data Register Empty (TDRE) status bit.

**Request-to-Send (RTS)**

The  $\overline{RTS}$  output enables the MPU to control a peripheral or modem via the data bus. The  $\overline{RTS}$  output corresponds to the state of the Control Register bits CR5 and CR6. When CR6=0 or both CR5 and CR6=1, the  $\overline{RTS}$  output is "Low" (the active state). This output can also be used for Data Terminal Ready (DTR).

**Data Carrier Detect ( $\overline{DCD}$ )**

This high impedance TTL compatible input provides automatic control, such as in the receiving end of a communications link by means of a modem  $\overline{DCD}$  output. The  $\overline{DCD}$  input inhibits and initializes the receiver section of the ACIA when "High". A "Low" to "High" transition of the  $\overline{DCD}$  initiates an interrupt to the MPU to indicate the occurrence of a loss of carrier when the Receive Interrupt Enable bit is set.

# **HD6852, HD68A52**

# **SSDA (Synchronous Serial Data Adapter)**

The HD6852 Synchronous Serial Data Adapter provides a bi-directional serial interface for synchronous data information interchange. It contains interface logic for simultaneously transmitting and receiving standard synchronous communications characters in bus organized systems such as the HMCS6800 Microprocessor systems.

The bus interface of the HD6852 includes select, enable, read/write, interrupt, and bus interface logic to allow data transfer over an 8-bit bi-directional data bus. The parallel data of the bus system is serially transmitted and received by the synchronous data interface with synchronization, fill character insertion/deletion, and error checking. The functional configuration of the SSDA is programmed via the data bus during system initialization.

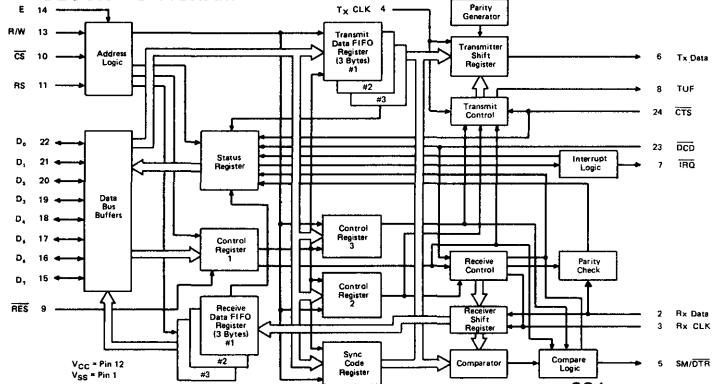
Programmable control registers provide control for variable word length, transmit control, receive control, synchronization control and interrupt control. Status, timing and control lines provide peripheral or modem control.

Typical applications include data communications terminals, floppy disk controllers, cassette or cartridge tape controllers and numerical control systems.

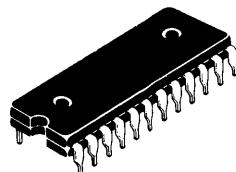
## ■ FEATURES

- Programmable Interrupts from Transmitter, Receiver, and Error Detection Logic
  - Character Synchronization on One or Two Sync Codes
  - External Synchronization Available for Parallel-Serial Operation
  - Programmable Sync Code Register
  - Up to 600kbps Transmitter
  - Peripheral/Modem Control Functions
  - Three Bytes of FIFO Buffering on Both Transmit and Receive
  - 6, 7, or 8 Bit Data Transmission
  - Optional Even and Odd Parity
  - Parity, Overrun, and Underflow Status
  - Compatible with MC6852 and MC68A52

## ■ BLOCK DIAGRAM

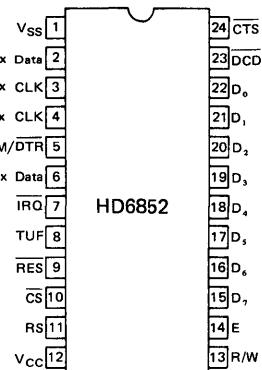


HD6852P, HD68A52P



(DP-24)

#### ■ PIN ARRANGEMENT



(Top View)

## ■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V <sub>CC</sub> *	-0.3 ~ +7.0	V
Input Voltage	V <sub>in</sub> *	-0.3 ~ +7.0	V
Operating Temperature	T <sub>opr</sub>	-20 ~ +75	°C
Storage Temperature	T <sub>stg</sub>	-55 ~ +150	°C

\* With respect to V<sub>SS</sub> (SYSTEM GND)

(NOTE) 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.

## ■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol	min	typ	max	Unit
Supply Voltage	V <sub>CC</sub> *	4.75	5	5.25	V
Input Voltage	V <sub>IL</sub> *	-0.3	—	0.8	V
	V <sub>IH</sub> *	2.0	—	V <sub>CC</sub>	V
Operating Temperature	T <sub>opr</sub>	-20	25	75	°C

\* With respect to V<sub>SS</sub> (SYSTEM GND)

## ■ ELECTRICAL CHARACTERISTICS

- DC CHARACTERISTICS (V<sub>CC</sub> = 5V ± 5%, V<sub>SS</sub> = 0V, Ta = -20~+75°C, unless otherwise noted.)

Item	Symbol	Test Condition	min	typ*	max	Unit	
Input "High" Voltage	V <sub>IH</sub>	—	2.0	—	—	V	
Input "Low" Voltage	V <sub>IL</sub>	—	—	—	0.8	V	
Output "High" Voltage	D <sub>0</sub> ~D <sub>7</sub>	V <sub>OH</sub>	I <sub>OH</sub> = -205 μA, PW <sub>EH</sub> , PW <sub>EL</sub> ≤ 25 μs	2.4	—	V	
	Tx Data DTR, TUF	V <sub>OH</sub>	I <sub>OH</sub> = -100 μA, PW <sub>EH</sub> , PW <sub>EL</sub> ≤ 25 μs	2.4	—	V	
Output "Low" Voltage	V <sub>OL</sub>	I <sub>OL</sub> = 1.6 mA, PW <sub>EH</sub> , PW <sub>EL</sub> ≤ 25 μs	—	—	0.4	V	
Input Leakage Current	I <sub>in</sub>	V <sub>in</sub> = 0 ~ 5.25V	—	—	2.5	μA	
Three-State Input Current (Off State)	D <sub>0</sub> ~D <sub>7</sub>	I <sub>TSI</sub>	V <sub>in</sub> = 0.4 ~ 2.4V, V <sub>CC</sub> = 5.25V	—	—	10	μA
Output Leakage Current (Off State)	IRQ	I <sub>LOH</sub>	V <sub>OH</sub> = 2.4V	—	—	10	μA
Power Dissipation	P <sub>D</sub>		—	300	525	mW	
Input Capacitance	D <sub>0</sub> ~D <sub>7</sub>	C <sub>in</sub>	—	—	12.5	pF	
	RxDATA, RxCLK, TxCLK, RES, CS, RS, R/W, E, DCD, CTS		V <sub>in</sub> = 0V, Ta = 25°C f = 1 MHz	—	—		
Output Capacitance	TxDATA, DTR, TUF,	C <sub>out</sub>	V <sub>in</sub> = 0V, Ta = 25°C	—	—	10	pF
	IRQ		f = 1 MHz	—	—	5.0	

\* Ta = 25°C, V<sub>CC</sub> = 5V

● AC CHARACTERISTICS ( $V_{CC}=5V\pm5\%$ ,  $V_{SS}=0V$ ,  $T_a=-20\sim+75^\circ C$ , unless otherwise noted.)

1. TIMING OF THE DATA TRANSFER

Item	Symbol	Test Condition	HD6852			HD68A52			Unit
			min	typ	max	min	typ	max	
Clock "Low" Pulse Width	$PW_{CL}$	Fig. 1	700	—	—	400	—	—	ns
Clock "High" Pulse Width	$PW_{CH}$	Fig. 2	700	—	—	400	—	—	ns
Clock Frequency	$f_C$		—	—	600	—	—	1,000	kHz
Receive Data Setup Time	$t_{RDSD}$	Fig. 3,7	350	—	—	200	—	—	ns
Receive Data Hold Time	$t_{RDH}$	Fig. 3	350	—	—	200	—	—	ns
Sync Match Delay Time	$t_{SM}$	Fig. 3	—	—	1.0	—	—	0.666	$\mu s$
Clock-to-Data Delay for Transmitter	$t_{TDD}$	Fig. 4,6	—	—	1.0	—	—	0.666	$\mu s$
Transmitter Underflow	$t_{TUF}$	Fig. 4	—	—	1.0	—	—	0.666	$\mu s$
DTR Delay Time	$t_{DTR}$	Fig. 5	—	—	1.0	—	—	0.666	$\mu s$
IRQ Release Time	$t_{IR}$	Fig. 5	—	—	1.2	—	—	0.8	$\mu s$
RES Pulse Width	$t_{RES}$		1.0	—	—	0.666	—	—	$\mu s$
CTS Setup Time	$t_{CTS}$	Fig. 6	200	—	—	150	—	—	ns
DCD Setup Time	$t_{DCD}$	Fig. 7	500	—	—	350	—	—	ns
Input Rise and Fall Times(Except E)	$t_r, t_f$	0.8V to 2.0V	—	—	1.0*	—	—	1.0*	$\mu s$

\* 1.0 $\mu$  or 10% of the pulse width, whichever is smaller.

2. BUS TIMING

1) READ

Item	Symbol	Test Condition	HD6852		HD68A52		Unit
			min	max	min	max	
Enable Cycle Time	$t_{cycE}$	Fig. 8	1.0	—	0.666	—	$\mu s$
Enable "High" Pulse Width	$PW_{EH}$		0.45	25	0.28	25	$\mu s$
Enable "Low" Pulse Width	$PW_{EL}$		0.43	—	0.28	—	$\mu s$
Setup Time, Address and R/W valid to Enable positive transition	$t_{AS}$		140	—	140	—	ns
Data Delay Time	$t_{DDR}$		—	320	—	220	ns
Data Hold Time	$t_H$		10	—	10	—	ns
Address Hold Time	$t_{AH}$		10	80	10	80	ns
Rise and Fall Time for Enable input	$t_{Er}, t_{Ef}$		—	25	—	25	ns

2) WRITE

Item	Symbol	Test Condition	HD6852		HD68A52		Unit
			min	max	min	max	
Enable Cycle Time	$t_{cycE}$	Fig. 9	1.0	—	0.666	—	$\mu s$
Enable Pulse Width, "High"	$PW_{EH}$		0.45	25	0.28	25	$\mu s$
Enable Pulse Width, "Low"	$PW_{EL}$		0.43	—	0.28	—	$\mu s$
Setup Time, Address and R/W valid to Enable positive transition	$t_{AS}$		140	—	140	—	ns
Data Setup Time	$t_{DSW}$		195	—	80	—	ns
Data Hold Time	$t_H$		10	—	10	—	ns
Address Hold Time	$t_{AH}$		10	80	10	80	ns
Rise and Fall Time for Enable input	$t_{Er}, t_{Ef}$		—	25	—	25	ns

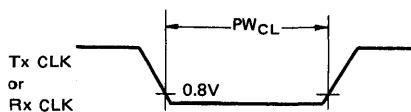


Figure 1 Clock Pulse Width ("Low" level)

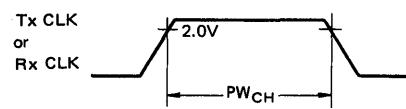


Figure 2 Clock Pulse Width ("High" level)

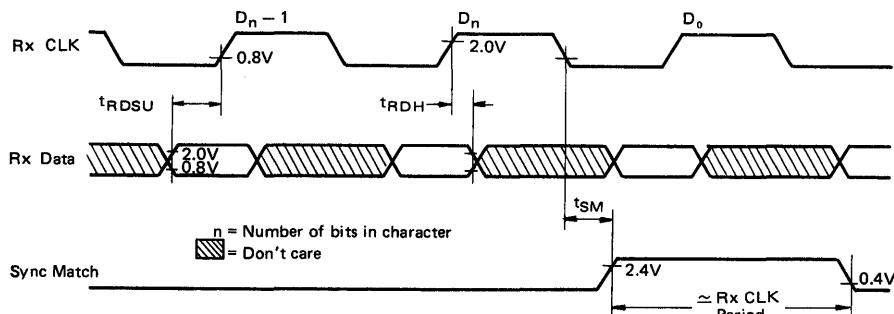


Figure 3 Receive Data Setup and Hold Times and Sync Match Delay Time

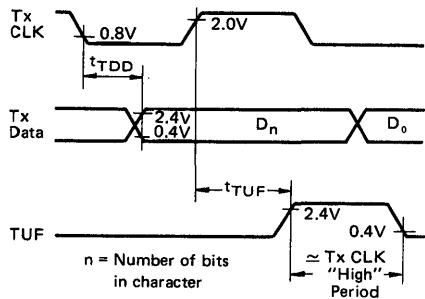


Figure 4 Transmit Data Output Delay and Transmitter Underflow Delay Time

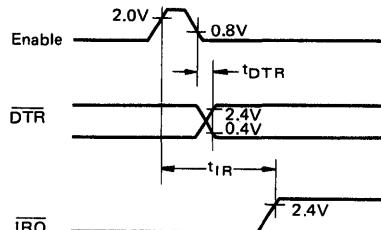


Figure 5  $\overline{DTR}$  and  $\overline{IRQ}$  Release Time

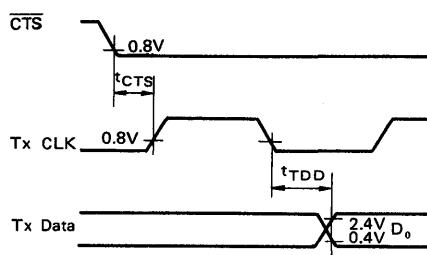


Figure 6  $\overline{CTS}$  Setup Time

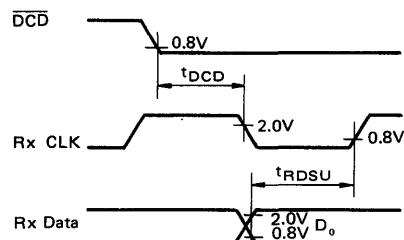


Figure 7  $\overline{DCD}$  Setup Time

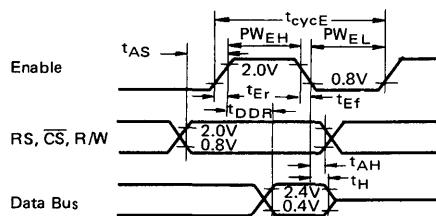


Figure 8 Bus Read Timing Characteristics  
(Read information from SSDA)

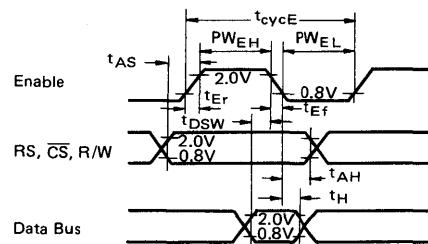
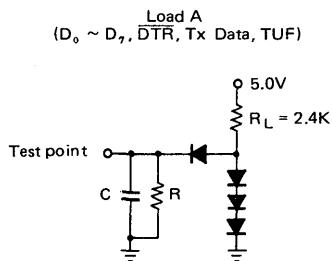


Figure 9 Bus Write Timing Characteristics  
(Write information into SSDA)



$C=130\text{pF}$  for  $D_0 \sim D_7$ ,  
 $=30\text{pF}$  for  $\overline{DTR}$ , Tx Data, and TUF  
All diodes are 1S2074 (⊕) or equiv.  
 $R=11\text{k}\Omega$  for  $D_0 \sim D_7$ ,  
 $=24\text{k}\Omega$  for  $\overline{DTR}$ , Tx Data, and TUF

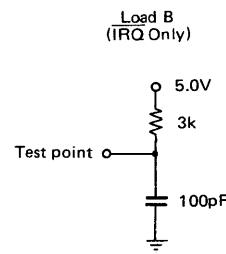


Figure 10 Test Loads

## ■ DEVICE OPERATION

At the bus interface, the SSDA appears as two addressable memory locations. Internally, there are seven registers: two read-only and five write-only registers. The read-only registers are Status and Receive Data; the write-only registers are Control 1, Control 2, Control 3, Sync Code and Transmit Data. The serial interface consists of serial input and output lines with independent clocks, and four peripheral/modem control lines.

Data to be transmitted is transferred directly into the 3-byte Transmit Data First-In First-Out (FIFO) Register from the data bus. Availability of the input to the FIFO is indicated by a bit in the Status Register; once data is entered, it moves through the FIFO to the last empty location. Data at the output of the FIFO is automatically transferred from the FIFO to the Transmitter Shift Register as the shift register becomes available to transmit the next character. If data is not available from the FIFO (underflow condition), the Transmitter Shift Register is automatically loaded with either a sync code or an all "1"s character. The transmit section may be programmed to append even, odd, or no parity to the transmitted word. An external control line ( $\overline{CTS}$ ) is provided to inhibit the transmitter without clearing the FIFO.

Serial data is accumulated in the receiver based on the synchronization mode selected. In the external sync mode used for parallel-serial operation, the receiver is synchronized by the

Data Carrier Detect ( $\overline{DCD}$ ) input and transfers successive bytes of data to the input of the Receiver FIFO. The single-sync-character mode requires that a match occur between the Sync Code Register and one incoming character before data transfer to the FIFO begins. The two-sync-character mode requires that two sync codes be received in sequence to establish synchronization. Subsequent to synchronization in any mode, data is accumulated in the shift register, and parity is optionally checked. An indication of parity error is carried through the Receiver FIFO with each character to the last empty location. Availability of a word at the FIFO output is indicated by a bit in the Status Register, as is a parity error.

The SSDA and its internal registers are selected by the address bus, Read/Write (R/W) and Enable control lines. To configure the SSDA, Control Registers are selected and the appropriate bits set. The Status Register is addressable for reading status.

Other I/O lines, in addition to Clear-to-Send ( $\overline{CTS}$ ) and Data Carrier Detect ( $\overline{DCD}$ ), include Sync Match/Data Terminal Ready (SM/ $\overline{DTR}$ ) and Transmitter Underflow (TUF). The transmitter and receiver each have individual clock inputs allowing simultaneous operation under separate clock control. Signals to the microprocessor are the Data bus and Interrupt Request ( $\overline{IRQ}$ ).

### ● Initialization

During a power-on sequence, the SSDA is reset via the  $\overline{\text{RES}}$  input and internally latched in a reset condition to prevent erroneous output transmissions. The Sync Code Register, Control Register 2, and Control Register 3 should be programmed prior to the programmed release of the Transmitter and/or Receiver Reset bits; these bits in Control Register 1 should be cleared after the  $\overline{\text{RES}}$  line has gone "High".

### ● Transmitter Operation

Data is transferred to the transmitter section in parallel form by means of the data bus and Transmit Data FIFO. The Transmit Data FIFO is a 3-byte register whose status is indicated by the Transmitter Data Register Available status bit (TDRA) and its associated interrupt enable bit. Data is transferred through the FIFO on negative edges of Enable (E) pulses. Two data transfer modes are provided in the SSDA. The 1-byte transfer mode provides for writing data to the transmitter section (and reading from the receiver section) one byte at a time. The 2-byte transfer mode provides for writing two data characters in succession.

Data will automatically transfer from the last register location in the Transmit Data FIFO (when it contains data) to the Transmitter Shift Register during the last half of the last bit of the previous character. A character is transferred into the Shift Register by the Transmitter Clock. Data is transmitted LSB first, and odd or even parity can be optionally appended. The unused bit positions in short word length characters from the data bus are "don't cares". (Note: The data bus inputs may be reversed for applications requiring the MSB to be transferred taken, e.g., IBM format for floppy disks; however, care must be taken to properly program the control registers – Table 1 will have its bit positions reversed.)

When the Shift Register becomes empty, and data is not available for transfer from the Transmit Data FIFO, an "underflow" occurs, and a character is inserted into the transmitter data stream to maintain character synchronization. The character transmitted on underflow will be either a "Mark" (all "1"s) or the contents of the Sync Code Register, depending upon the state of the Transmit Sync Code on Underflow control bit. The underflow condition is indicated by a pulse ( $\approx \text{Tx CLK}$  "High" period) on the Underflow output (when in Tx Sync on underflow mode). The Underflow output occurs coincident with the transfer of the last half of the last bit preceding the underflow character. The Underflow status bit is set until cleared by means of the Clear Underflow control bit. This output may be used in floppy disk systems to synchronize write operations and for appending CRCC.

Transmission is initiated by clearing the Transmitter Reset bit in Control Register 1. When the Transmitter Reset bit is cleared, the first full positive half-cycle of the Transmit Clock will initiate the transmit cycle, with the transmission of data or underflow characters beginning on the negative edge of the Transmit Clock pulse which started the cycle. If the Transmit Data FIFO was not loaded, an underflow character will be transmitted.

The Clear-to-Send ( $\overline{\text{CTS}}$ ) input provides for automatic control of the transmitter by means of external system hardware; e.g., the modem  $\overline{\text{CTS}}$  output provides the control in a data communications system. The  $\overline{\text{CTS}}$  input resets and inhibits the transmitter section when "High", but does not reset the Transmit Data FIFO. The TDRA status bit is inhibited by  $\overline{\text{CTS}}$  being "High" in either the one-sync character or two-sync-character mode of operation.

In the external sync mode, TDRA is unaffected by  $\overline{\text{CTS}}$  in order to provide Transmit Data FIFO status for preloading and operating the transmitter under the control of the  $\overline{\text{CTS}}$  input. When the Transmitter Reset bit (Tx Rs) is set, the Transmit Data FIFO is cleared and the TDRA status bit is cleared. After one E clock has occurred, the Transmit Data FIFO becomes available for new data with TDRA inhibited.

### ● Receiver Operation

Data and a presynchronized clock are provided to the SSDA receiver section by means of the Receive Data (Rx Data) and Receive Clock (Rx CLK) inputs. The data is a continuous stream of binary data bits without means for identifying character boundaries within the stream. It is, therefore, necessary to achieve character synchronization for the data at the beginning of the data block. Once synchronization is achieved, it is assumed to be retained for all successive characters within the block.

Data communications systems utilize the detection of sync codes during the initial portion of the preamble to establish character synchronization. This requires the detection of a single code or two successive sync codes. Floppy disk and cartridge tape units require sixteen bits of defined preamble and cassettes require eight bits of preamble to establish the reference for the start of record. All three are functionally equivalent to the detection of sync codes. Systems which do not utilize code detection techniques require custom logic external to the SSDA for character synchronization and use of the parallel-to-serial (external sync) mode.

(Note: The Receiver Shift Register is set to ones when reset)

### ● Synchronization

The SSDA provides three operating modes with respect to character synchronization: one-sync-character mode, two-sync-character mode, and external sync mode. The external sync mode requires synchronization and control of the receiving section through the Data Carrier Detect (DCD) input. This external synchronization could consist of direct line control from the transmitting end of the serial data link or from external logic designed to detect the start of the message block. The one-sync-character mode searches on a bit-by-bit basis until a match is achieved between the data in the Shift Register and the Sync Code Register. The match indicates character synchronization is complete and will be retained for the message block. In the two-sync-character mode, the receiver searches for the first sync code match on a bit-by-bit basis and then looks for a second successive sync code character prior to establishing character synchronization. If the second sync code character is not received, the bit-by-bit search for the first sync code is resumed.

Sync codes received prior to the completion of synchronization (one or two character) are not transferred to the Receive Data FIFO. Redundant sync codes during the preamble or sync codes which occur as "fill characters" can automatically be stripped from the data, when the Strip Sync control bit is set, to minimize system loading. The character synchronization will be retained until cleared by means of the Clar Sync bit, which also inhibits synchronization search when set.

### ● Receiving Data

Once synchronization has been achieved, subsequent characters are automatically transferred into the Receive Data FIFO and clocked through the FIFO to the last empty location by E pulses (MPU System  $\phi_2$ ). The Receiver Data Available status bit

(RDA) indicates when data is available to be read from the last FIFO location (#3) when in the 1-byte transfer mode. The 2-byte transfer mode causes the RDA status bit to indicate data is available when the last two FIFO register locations are full. Data being available in the Receive Data FIFO causes an interrupt request if the Receiver Interrupt Enable (RIE) bit is set. The MPU will then read the SSDA Status Register, which will indicate that data is available for the MPU read from the Receiver Data FIFO register. The  $\overline{IRQ}$  and RDA status bits are reset by a read from the FIFO. If more than one character has been received and is resident in the Receive Data FIFO, subsequent E clocks will cause the FIFO to update and the RDA and  $\overline{IRQ}$  status bits will again be set. The read data operation for the 2-byte transfer mode requires an intervening E clock between reads to allow the FIFO data to shift. Optional parity is automatically checked as data is received, and the parity status condition is maintained with each character until the data is read from the Receive Data FIFO. Parity errors will cause an interrupt request if the Error Interrupt Enable (EIE) has been set. The parity bit is not transferred to the data bus but must be checked in the Status Register. NOTE: In the 2-byte transfer mode, parity should be checked prior to reading the second byte, since a FIFO read clears the error bit.

Other status bits which pertain to the receiver section are Receiver Overrun and Data Carrier Detect (DCD). The Overrun status bit is automatically set when a transfer of a character to the Receive Data FIFO occurs and the first register of the Receive Data FIFO is full. Overrun causes an interrupt if Error Interrupt Enable (EIE) has been set. The transfer of the overrunning character into the FIFO causes the previous character in the FIFO input register location to be lost. The Overrun status bit is cleared by reading the Status Register (when the overrun condition is present), followed by a Receive Data FIFO Register read. Overrun cannot occur and be cleared without providing an opportunity to detect its occurrence via the Status Register.

A positive transition on the  $\overline{DCD}$  input causes an interrupt if the EIE control bit has been set. The interrupt caused by  $\overline{DCD}$  is cleared by reading the Status Register when the  $\overline{DCD}$  status bit is "1", followed by a Receive Data FIFO read. The  $\overline{DCD}$  status bit will subsequently follow the state of the  $\overline{DCD}$  input when it goes "Low".

## ■ SSDA REGISTERS

Seven registers in the SSDA can be accessed by means of the bus. The registers are defined as read-only or write-only according to the direction of information flow. The Register Select (RS) input selects two registers in each state, one being read-only and the other write-only. The Read/Write (R/W) input defined which of the two selected registers will actually be accessed. Four registers (two read-only and two write-only) can be addressed via the bus at any particular time. These registers and the required addressing are defined in Table 1.

### ● Control Register 1 (C1)

Control Register 1 is an 8-bit write-only register that can be directly addressed from the data bus. Control Register 1 is addressed when RS = "Low" and R/W = "Low".

### Receiver Reset (Rx Rs), C1 Bit 0

The Receiver Reset control bit provides both a reset and inhibit function to the receiver section. When Rx Rs is set, it clears the receiver control logic, error logic, Rx Data FIFO

Control, Parity Error status bit, and  $\overline{DCD}$  interrupt. The Receiver Shift Register is set ones. The Rx Rs bit must be cleared after the occurrence of a "Low" level on RES in order to enable the receiver section of the SSDA.

### Transmitter Reset (Tx Rs), C1 Bit 1

The Transmitter Reset control bit provides both a reset and inhibit to the transmitter section. When Tx Rs is set, it clears the transmitter control section, Transmitter Shift Register, Tx Data FIFO Control (the Tx Data FIFO can be reloaded after one E clock pulse), the Transmitter Underflow status bit, and the  $\overline{CTS}$  interrupt, and inhibits the TDRA status bit (in the one-sync-character and two-sync-character modes). The Tx Rs bit must be cleared after the occurrence of a "Low" level on RES in order to enable the transmitter section of the SSDA. If the Tx FIFO is not preloaded, it must be loaded immediately after the Tx Rs release to prevent a transmitter underflow condition.

### Strip Synchronization Characters (Strip Sync), C1 Bit 2

If the Strip Sync bit is set, the SSDA will automatically strip all received characters which match the contents of the Sync Code Register. The characters used for synchronization (one or two characters of sync) are always stripped from the received data stream.

### Clear Synchronization (Clear Sync), C1 Bit 3

The Clear Sync control bit provides the capability of dropping receiver character synchronization and inhibiting resynchronization. The Clear Sync bit is set to clear and inhibit receiver synchronization in all modes and is reset to zero to enable resynchronization.

### Transmitter Interrupt Enable (TIE), C1 Bit 4

TIE enable both the Interrupt Request ( $\overline{IRQ}$ ) output and Interrupt Request status bit to indicate a transmitter service request. When TIE is set and the TDRA status bit is "1", the  $\overline{IRQ}$  output will go "Low" (the active state) and the  $\overline{IRQ}$  status bit will go "1".

### Receiver Interrupt Enable (RIE), C1 Bit 5

RIE enable both the Interrupt Request output ( $\overline{IRQ}$ ) and the Interrupt Request status bit to indicate a receiver service request. When RIE is set and the RDA status bit is "1", the  $\overline{IRQ}$  output will go "Low" (the active state) and the  $\overline{IRQ}$  status bit will go "1".

### Address Control 1 (AC1) and Address Control 2 (AC2), C1 Bits 6 and 7

AC1 and AC2 select one of the write-only registers – Control 2, Control 3, Sync Code, or Tx Data FIFO – as shown in Table 1, when RS = "High" and R/W = "Low".

### ● Control Register 2 (C2)

Control Register 2 is an 8-bit write-only register which can be programmed from the bus when the Address Control bits in Control Register 1 (AC1 and AC2) are reset, RS = "High" and R/W = "Low".

### Peripheral Control 1 (PC1) and Peripheral Control 2 (PC2), C2 Bits 0 and 1

Two control bits, PC1 and PC2, determine the operating characteristics of the Sync Match/DTR output. PC1, when "High", selects the Sync Match mode. PC2 provides the inhibit/

enable control for the SM/DTR output in the Sync Match mode. A one-bit-wide pulse is generated at the output when PC2 is "0", and a match occurs between the contents of the Sync Code Register and the incoming data even if sync is inhibited (Clear Sync bit = "1"). The Sync Match pulse is referenced to the negative edge of Rx CLK pulse causing the match.

The Data Terminal Ready (DTR) mode is selected when PC1 is "0". When PC2 = "1" the SM/DTR output = "Low" and vice versa. The operation of PC2 and PC1 is summarized in Table 4.

#### 1-Byte/2-Byte Transfer (1-Byte/2-Byte), C2 Bit 2

When 1-Byte/2-Byte is set, the TDRA and RDA status bits will indicate the availability if their respective data FIFO registers for a single byte data transfer. Alternately, if 1-Byte/2-Byte is reset, the TDRA and RDA status bits indicate when two bytes of data can be moved without a second status read. An intervening Enable pulse must occur between data transfers.

#### Word Length Selects (WS1, WS2, WS3), C2 Bits 3, 4, 5

Word length Select bits WS1, WS2, and WS3 select word length of 7, 8, or 9 bits including parity as shown in Table 3.

#### Transmit Sync Code on Underflow (Tx Sync), C2 Bit 6

When Tx Sync is set, the transmitter will automatically send a sync character when data is not available for transmission. If Tx Sync is reset, the transmitter will transmit a Mark character (including the parity bit position) on underflow. When the underflow is detected, a pulse approximately a Tx CLK "High" period wide will occur on the underflow output if the Tx Sync bit is "1". Internal parity generation is inhibited during underflow except for sync code fill character transmission in 8 bit plus parity word lengths.

#### Error Interrupt Enable (EIE), C2 Bit 7

When EIE is set, the IRQ status bit will go "1" and the IRQ output will go "Low" if:

- 1) A receiver overrun occurs. The interrupt is cleared by reading the Status Register and reading the Rx Data FIFO.
- 2) DCD input has gone to a "High". The interrupt is cleared by reading the Status Register and reading the Rx Data FIFO.
- 3) A parity error exists for the character in the last location (#3) of the Rx Data FIFO. The interrupt is cleared by reading the Rx Data FIFO. The interrupt is cleared by reading the Rx Data FIFO.
- 4) The CTS input has gone to a "High". The interrupt is cleared by writing a "1" in the Clear CTS bit, C3 bit 2, or by a Tx Reset.
- 5) The transmitter has underflowed (in the Tx Sync on Underflow mode). The interrupt is cleared by writing a "1" into the Clear Underflow, C3 bit 3, or Tx Reset.

When EIE is a "0", the IRQ status bit and the IRQ output are disabled for the above error conditions. A "Low" level on the RES input resets EIE to "0".

#### • Control Register 3 (C3)

Control Register 3 is a 4-bit write-only register which can be programmed from the bus when RS = "High" and R/W = "Low" and Address Control bit AC1 = "1" and AC2 = "0".

#### External/Internal Sync Mode Control (E/I Sync), C3 Bit 0

When the E/I Sync Mode bit is "1", the SSDA is in the external sync mode and the receiver synchronization logic is disabled. Synchronization can be achieved by means of the DCD input or by starting Rx CLK at the midpoint of data bit "0" of

a character with DCD "Low". Both the transmitter and receiver sections operate as parallel — serial converters in the External Sync mode. The Clear Sync bit in Control Register 1 acts as a receiver sync inhibit when "High" to provide a bus controllable inhibit. The Sync Code Register can serve as a transmitter fill character register and a receiver match register in this mode. A "Low" on the RES input resets the E/I Sync Mode bit placing the SSDA in the internal sync mode.

#### One-Sync-Character/Two-Sync-Character Mode, Control (1 Sync/2 Sync), C3 Bit 1

When the 1 Sync/2 Sync bit is set, the SSDA will synchronize on a single match between the received data and the contents of the Sync Code Register. When the 1 Sync/2 Sync bit is reset, two successive sync characters must be received prior to receiver synchronization. If the second sync character is not detected, the bit by bit search resumes from the first bit in the second character. See the description of the Sync Code Register for more details.

#### Clear CTS Status (Clear CTS), C3 Bit 2

When a "1" is written into the Clear CTS bit, the stored status and interrupt are cleared. Subsequently, the CTS status bit reflects the state of the CTS input. The Clear CTS control bit does not affect the CTS input nor its inhibit of the transmitter section. The Clear CTS command bit is self-clearing, and writing a "0" into this bit is a nonfunctional operation.

#### Clear Transmit Underflow Status (CTUF), C3 Bit 3

When a "1" is written into the CTUF status bit, the CTUF bit and its associated interrupt are reset. The CTUF command bit is self-clearing and writing a "0" into this bit is a nonfunctional operation.

#### • Sync Code Register

The Sync Code Register is an 8-bit register for storing the programmable sync code required for received data character synchronization in the one-sync-character and two-sync-character modes. The Sync Code Register also provides for stripping the sync/fill characters from the received data (a programmable option) as well as automatic insertion of fill characters in the transmitted data stream. The Sync Code Register is not utilized for receiver character synchronization in the external sync mode; however, it provides storage of receiver match and transmit fill characters.

The Sync Code Register can be loaded when AC2 and AC1 are a "1" and "0" respectively, and R/W = "Low" and RS = "High".

The Sync Code Register may be changed after the detection of a match with the received data (the first sync code having been detected) to synchronize with a double-word sync pattern. (This sync code change must occur prior to the completion of the second character.) The sync match (SM) output can be used to interrupt the MPU system to indicate that the first eight bits have matched. The service routine would then change the sync match register to the second half of the pattern. Alternately, the one-sync-character mode can be used for sync codes for 16 or more bits by using software to check the second and subsequent bytes after reading them from the FIFO.

The detection of the sync code can be programmed to appear on the Sync Match/DTR output by writing a "1" in PC1 (C2 bit 0) and a "0" in PC2 (C2 bit 1). The Sync Match output will go "High" for one bit time beginning at the character interface between the sync code and the next character.

### ● Parity for Sync Character Transmitter

Transmitter does not generate parity for the sync character except 9-bit mode.

- 9-bit (8-bit + parity) – 8-bit sync character + parity
- 8-bit (7-bit + parity) – 8-bit sync character (no parity)
- 7-bit (6-bit + parity) – 7-bit sync character (no parity)

### Receiver

#### At Synchronization

Receiver automatically strips the sync character(s) (two sync characters if '2 sync' mode is selected) which is used to establish synchronization. And parity is not checked for these sync characters.

#### After Synchronization is Established

When 'strip sync' bit is selected, the sync characters (fill characters) are stripped and parity is not checked for the stripped sync (fill) characters. When 'strip sync' bit is not selected (0), the sync character is assumed to be normal data and it is transferred into FIFO after parity checking. (When non-parity format is selected, parity is not checked.)

Strip Sync (C1 Bit 2)	Data Format (C2 Bit 3-5)	Operation
1	x	No transfer of sync code. No parity check of sync code.
0	With Parity	*Transfer data and sync codes. Parity check.
0	Without Parity	*Transfer data and sync codes. No parity check.

\* Subsequent to synchronization

x .... don't care

It is necessary to pay attention to the selected sync character in the following cases.

- 1) Data format is (6 + parity), (7 + parity),
- 2) Strip sync is not selected ("0").
- 3) After synchronization when sync code is used as a fill character.

Transmitter sends sync character without parity, but receiver checks the parity as if it is normal data. Therefore, the sync character should be chosen to match the parity check selected for the receiver in this special case.

### ● Receive Data First-In First-Out Register

#### (Rx Data FIFO)

The Receive Data FIFO Register consists of three 8-bit registers which are used for buffer storage of received data. Each 8-bit register has an internal status bit which monitors its full or empty condition. Data is always transferred from a full register to an adjacent empty register. The transfer from register to register occurs on E pulses. The RDA status bit will be "1" when data is available in the last location of the Rx Data FIFO.

In an Overrun condition, the overrunning character will be transferred into the full first stage of the FIFO register and will cause the loss of that data character. Successive overruns continue to overwrite the first register of the FIFO. This destruction of data is indicated by means of the Overrun status

bit. The Overrun bit will be set when the overrun occurs and remains set until the Status Register is read, followed by a read of the Rx Data FIFO.

Unused data bits for short word lengths (including the parity bit) will appear as "0's" on the data bus when the Rx Data FIFO is read.

### ● Transmit Data First-In First-Out Register (Tx Data FIFO)

The Transmit Data FIFO Register consists of three 8-bit registers which are used for buffer storage of data to be transmitted. Each 8-bit register has an internal status bit which monitors its full or empty condition. Data is always transferred from a full register to an adjacent empty register. The transfer is clocked by E pulses.

The TDRA status bit will be "High" if the Tx Data FIFO is available for data.

Unused data bits for short word lengths will be handled as "don't cares". The parity bit is not transferred over the data bus since the SSDA generates parity at transmission.

When an Underflow occurs, the Underflow character will be either the contents of the Sync Code Register or an all "1's" character. The underflow will be stored in the Status Register until cleared and will appear on the Underflow output as a pulse approximately a Tx CLK "High" period wide.

### ● Status Register

The Status Register is an 8-bit read-only register which provides the real-time status of the SSDA and the associated serial data channel. Reading the Status Register is a non-destructive process. The method of clearing status bits depends upon the function each bit represents and is discussed for each bit in the register.

#### Receiver Data Available (RDA), S Bit 0

The Receiver Data Available status bit indicates when receiver data can be read from the Rx Data FIFO. The receiver data being present in the last register (#3) of the FIFO causes RDA to be "1" for the 1-byte transfer mode. The RDA bit being "1" indicates that the last two registers (#2 and #3) are full when in the 2-byte transfer mode. The second character can be read without a second status read (to determine that the character is available). And E pulse must occur between reads of the Rx Data FIFO to allow the FIFO to shift. Status must be read on a word-by-word basis if receiver data error checking is important. The RDA status bit is reset automatically when data is not available.

#### Transmitter Data Register Available (TDRA), S Bit 1

The TDRA status bit indicates that data can be loaded into the Tx Data FIFO Register. The first register (#1) of the Tx Data FIFO being empty will be indicated by a "1" in the TDRA status bit in the 1-byte transfer mode. The first two registers (#1 and #2) must be empty for TDRA to be "1" when in the 2-byte transfer mode. The Tx Data FIFO can be loaded with two bytes without an intervening status read; however, one E pulse must occur between loads. TDRA is inhibited by the Tx Reset or RES. When Tx Reset is set, the Tx Data FIFO is cleared and then released on the next E clock pulse. The Tx Data FIFO can then be loaded with up to three characters of data, even though TDRA is inhibited. This feature allows preloading data prior to the release of Tx Reset. A "High" level on the CTS input inhibits the TDRA status bit in either sync mode of operation (one-sync-character or two-sync-character). CTS does not affect TDRA in the external sync mode. This

enables the SSDA to operate under the control of the CTS input with TDRA indicating the status of the Tx Data FIFO. The CTS input does not clear the Tx Data FIFO in any operating mode.

#### Data Carrier Detect (DCD), S Bit 2

A positive transition on the DCD input is stored in the SSDA until cleared by reading both Status and Rx Data FIFO. A "1" written into Rx Rs also clears the stored DCD status. The DCD status bit, when set, indicates that the DCD input has gone "High". The reading of both Status and Receive Data FIFO allows Bit 2 of subsequent Status reads to indicate the state of the DCD input until the next positive transition.

#### Clear-to-Send (CTS), S Bit 3

A positive transition on the CTS input is stored in the SSDA until cleared by writing a "1" into the Clear CTS control bit or the Tx Rs bit. The CTS status bit, when set, indicates that the CTS input has gone "High". The Clear CTS command (a "1" into C3 Bit 2) allows Bit 3 of subsequent Status reads to indicate the state of the CTS input until the next positive transition.

#### Transmitter Underflow (TUF), S Bit 4

When data is not available for the transmitter, an underflow occurs and is so indicated in the Status Register (in the Tx Sync on underflow mode). The underflow status bit is cleared by writing a "1" into the Clear Underflow.(CTUF) control bit or

the Tx Rs bit. TUF indicates that a sync character will be transmitted as the next character. A TUF is indicated on the output only when the contents of the Sync Code Register is to be transferred (transmit sync code on underflow = "1").

#### Receiver Overrun (Rx Ovrn), S Bit 5

Overrun indicates data has been received when the Rx Data FIFO is full, resulting in data loss. The Rx Ovrn status bit is set when Overrun occurs. The Rx Ovrn status bit is cleared by reading Status followed by reading the Rx Data FIFO or by setting the Rx Rs control bit.

#### Receiver Parity Error (PE), S Bit 6

The parity error status bit indicates that parity for the character in the last register of the Rx Data FIFO did not agree with selected parity. The parity error is cleared when the character to which it pertains is read from the Rx Data FIFO or when Rx Rs occurs. The DCD input does not clear the Parity Error or Rx Data FIFO status bits.

#### Interrupt Request (IRQ), S Bit 7

The Interrupt Request status bit indicates when the IRQ output is in the active state (IRQ output = "Low"). The IRQ status bit is subject to the same interrupt enables (RIE, TIE, and EIE) as the IRQ output. The IRQ status bit simplifies status inquiries for polling systems by providing single bit indication of service requests.

Table 1 SSDA Programming Model

Register	Control* Inputs				Address Control		Register Content							
	RS	R/W	AC2	AC1	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0		
Status (S)	0	1	X	X	Interrupt Request (IRQ)	Receiver Parity Error (PE)	Receiver Overrun (Rx Ovrn)	Transmitter Underflow (TUF)	Clear-to-Send (CTS)	Data Carrier Detect (DCD)	Transmitter Data Register Available (TDRA)	Receiver Data Available (RDA)		
Control 1 (C1)	0	0	X	X	Address Control 2 (AC2)	Address Control 1 (AC1)	Receiver Interrupt Enable (RIE)	Transmitter Interrupt Enable (TIE)	Clear Sync	Strip Sync Characters (Strip Sync)	Transmitter Reset (Tx Rs)	Receiver Reset (Rx Rs)		
*** Receive Data FIFO	1	1	X	X	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>		
Control 2 (C2)	1	0	0	0	Error Interrupt Enable (EIE)	Transmit Sync Code on Underflow (Tx Sync)	Word Length Select 3 (WS3)	Word Length Select 2 (WS2)	Word Length Select 1 (WS1)	1-Byte/2-Byte Transfer (1-Byte/2-Byte)	Peripheral Control 2 (PC2)	Peripheral Control 1 (PC1)		
Control 3 (C2)	1	0	0	1	Not Used	Not Used	Not Used	Not Used	Clear Transmitter Underflow Status (CTUF)	Clear CTS Status (Clear CTS)	One-Sync-Character/Two-Sync Character Mode Control (1 Sync/2 Sync)	External/Internal Sync Mode Control (E/I Sync)		
Sync Code **	1	0	1	0	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>		
*** Transmit Data FIFO	1	0	1	1	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>		

\* 0 ; "Low" level, 1 ; "High" level

\*\* "FF" should not be used as Sync Code.

\*\*\* When the SSDA is used in applications requiring the MSB of data to be receive and transmitted first, the data bus inputs to the SSDA may be reversed (D<sub>0</sub> to D<sub>7</sub>, etc.). Caution must be used when this is done since the bit positions in this table will be reversed, and the parity should not be selected.

Table 2 Functions of SSDA Register

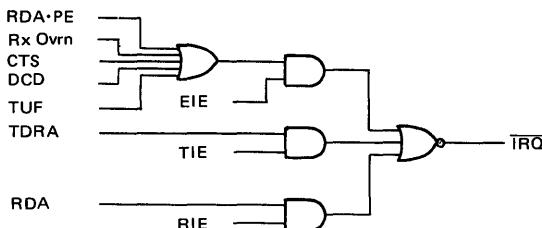
Register	Bit	Symbol	Function		
Status Register (S)	7	IRQ	The IRQ flag is cleared when the source of the IRQ is cleared. The source is determined by the enables in the Control Registers: TIE, RIE, EIE.		
	6	PE	Conditions for Set	When parity error is detected in receive data.	Read Rx Data FIFO, or a "1" into Rx Rs (C1 Bit 0).
	5	Rx Ovrn		When receive data FIFO overruns.	Read Status and then Rx Data FIFO, or a "1" into Rx Rs (C1 Bit 0).
	4	TUF		When under flow is occurred in the transmitter.	A "1" into CTUF (C3 Bit 3) or into Tx Rs (C1 Bit 1).
	3	CTS		When CTS signal rises.	A "1" into Clear CTS (C3 Bit 2) or a "1" into Tx Rs (C1 Bit 1)
	2	DCD		When DCD signal rises.	Read Status and then Rx Data FIFO or a "1" into Rx Rs (C1 Bit 0)
	1	TDRA		1 Byte Transfer Mode; when the transmit data FIFO (#1) is empty. 2 Byte Transfer Mode; when the transmit data FIFO (#1, #2) is empty.	Write into Tx Data FIFO.
	0	RDA		1 Byte Transfer Mode; when the data is received in the receive data FIFO (#3). 2 Byte Transfer Mode; when the data is received in the receive data FIFO (#2, #3).	Read Rx Data FIFO.
	7	AC2	Used to access other registers, as shown Table 1.		
	6	AC1			
Control Register 1 (C1)	5	RIE	When "1", enables interrupt on RDA (S Bit 0).		
	4	TIE	When "1", enables interrupt on TDRA (S Bit 1).		
	3	Clear Sync	When "1", clears receiver character synchronization.		
	2	Strip Sync	When "1", strips all sync codes from the received data stream.		
	1	Tx Rs	When "1", resets and inhibits the transmitter section.		
	0	Rx Rs	When "1", resets and inhibits the receiver section.		
	7	EIE	When "1", enables the PE, Rx Ovrn, TUF, CTS, and DCD interrupt flags (S Bits 6 through 2).		
	6	Tx Sync	When "1", allows sync code contents to be transferred on underflow, and enables the TUF Status bit and output. When "0", an all mark character is transmitted on underflow.		
Control Register 2 (C2)	5	WS3			
	4	WS2			
	3	WS1	Word Length Select		
	2	1-Byte/2-Byte	When "1", enables the TDRA and RDA bits to indicate when a 1-byte transfer can occur; when "0", the TDRA and RDA bits indicate when a 2-byte transfer can occur.		
Control Register 3 (C3)	1	PC2			
	0	PC2	SM/DTR Output Control		
	3	CTUF	When "1", clears TUF (S Bit 4), and IRQ if enabled.		
	2	Clear CTS	When "1", clears CTS (S Bit 3), and IRQ if enabled.		
Control Register 3 (C3)	1	1-Sync/2-Sync	When "1", selects the one-sync-character mode; when "0", selects the two-sync-character mode.		
	0	E/I Sync	When "1", selects the external sync mode; when "0", selects the internal sync mode.		

Table 3 Word Length

Bit 5 WS3	Bit 4 WS2	Bit 3 WS1	Word Length
0	0	0	6 Bits + Even Parity
0	0	1	6 Bits + Odd Parity
0	1	0	7 Bits
0	1	1	8 Bits
1	0	0	7 Bits + Even Parity
1	0	1	7 Bits + Odd Parity
1	1	0	8 Bits + Even Parity
1	1	1	8 Bits + Odd Parity

Table 4 SM/DTR Output Control

Bit 1 PC2	Bit 0 PC1	SM/DTR Output at Pin 5
0	0	"High" Level
0	1	Pulse  1-Bit Wide, on SM
1	0	"Low" Level
1	1	SM Inhibited, "Low"



#### ■ INTERFACE SIGNALS FOR MPU

The SSDA interfaces to the HD6800 MPU with an 8-bit bi-directional data bus, a chip select line, a register select line, an interrupt request line, read/write line, an enable line, and a reset line. These signals, in conjunction with the HD6800 VMA output, permit the MPU to have complete control over the SSDA.

##### ● Bi-Directional Data Bus ( $D_0 \sim D_7$ )

The bi-directional data bus ( $D_0 \sim D_7$ ) allow for data transfer between the SSDA and the MPU. The data bus output drivers are three-state devices that remain in the high impedance (off) state except when the MPU performs an SSDA read operation.

##### ● Enable (E)

The Enable signal, E, is a high impedance TTL compatible input that enables the bus input/output data buffers, clocks data to and from the SSDA, and moves data through the FIFO Registers. This signal is normally the continuous HMCS6800 System  $\phi 2$  clock, so that incoming data characters are shifted through the FIFO.

##### ● Read/Write (R/W)

The Read/Write line is a high impedance input that is TTL compatible and is used to control the direction of data flow through the SSDA's input/output data bus interface. When Read/Write is "High" (MPU read cycle), SSDA output drivers are turned on if the chip is selected and a selected register is read. When it is "Low", the SSDA output drivers are turned off and the MPU writes into a selected register. The Read/Write signal is also used to select read-only or write-only registeres within the SSDA.

##### ● Chip Select ( $\overline{CS}$ )

This high impedance TTL compatible input line is used to address the SSDA. The SSDA is selected when  $\overline{CS}$  is "Low". VMA should be used in generating the  $\overline{CS}$  input to insure that false selects will not occur. Transfers of data to and from the SSDA are then performed under the control of the Enable signal, Read/Write, and Register Select.

#### ● Register Select (RS)

The Register Select line is a high impedance input that is TTL compatible. A "High" level is used to select Control Registers C2 and C3, the Sync Code Register, and the Transmit/Receive Data Registers. A "Low" level selects the Control 1 and Status Registers (see Table 1).

#### ● Interrupt Request (IRQ)

IRQ is a TTL compatible, open-drain (no internal pullup), active "Low" output that is used to interrupt the MPU. The IRQ remains "Low" until cleared by the MPU.

#### ● Reset (RES)

The RES input provides a means of resetting the SSDA from an external source. In the "Low" state, the RES input causes the following:

- 1) Receiver Reset (Rx Rs) and Transmitter Reset (Tx Rs) bits are set causing both the receiver and transmitter sections to be held in a reset condition.
- 2) Peripheral Control bits PC1 and PC2 are reset to zero, causing the SM/DTR output to be "High".
- 3) The Error Interrupt Enable (EIE) bit is reset.
- 4) An internal synchronization mode is selected.
- 5) The Transmitter Data Register Available (TDRA) status bit is cleared and inhibited.

When RES returns "High" (the inactive state), the transmitter and receiver sections will remain in the reset state until the Receiver Reset and Transmitter Reset bits are cleared via the bus under software control. The control Register bits affected by RES (Rx Rs, Tx Rs, PC1, PC2, EIE, and E/I Sync) cannot be changed when RES is "Low".

#### ■ CLOCK INPUTS

Separate high impedance TTL compatible inputs are provided for clocking of transmitted and received data.

##### ● Transmit Clock (Tx CLK)

The Transmit Clock input is used for the clocking of transmitted data. The transmitter shifts data on the negative transition of the clock.

##### ● Receive Clock (Rx CLK)

The Receive Clock input is used for clocking in received data. The clock and data must be synchronized externally. The receiver samples the data on the positive transition of the clock.

#### ■ SERIAL INPUT/OUTPUT LINES

##### ● Receive Data (Rx Data)

The Receive Data line is a high impedance TTL compatible input through which data is received in a serial format. Data rates are from 0 to 600 kbps.

##### ● Transmit Data (Tx Data)

The Transmit Data output line transfers serial data to a modem or other peripheral. Data rates are from 0 to 600 kbps.

#### ■ PERIPHERAL/MODEM CONTROL

The SSDA includes several functions that permit limited control of a peripheral or modem. The functions included are CTS, SM/DTR, DCD, and TUF.

##### ● Clear-to-Send (CTS)

The CTS input provides a real-time inhibit to the transmitter

section (the Tx Data FIFO is not disturbed). A positive CTS transition resets the Tx Shift Register and inhibits the TDRA status bit and its associated interrupt in both the one-sync-character and two-sync-character modes of operation. TDRA is not affected by the CTS input in the external sync mode.

The positive transition of CTS is stored within the SSDA to insure that its occurrence will be acknowledged by the system. The stored CTS information and its associated IRQ (if enabled) are cleared by writing a "1" in the Clear CTS bit. The CTS status bit subsequently follows the CTS input when it goes "Low".

The CTS input provides character timing for transmitter data when in the external sync mode. Transmission is initiated on the negative transition of the first full positive clock pulse of the transmitter clock (Tx CLK) after the release of CTS (see Figure 6).

- **Data Carrier Detect (DCD)**

The DCD input provides a real-time inhibit to the receiver section (the Rx FIFO is not disturbed). A positive DCD transition resets and inhibits the receiver section except for the Receive FIFO and the RDRA status bit and its associated IRQ.

The positive transition of DCD is stored within the SSDA to insure that its occurrence will be acknowledged by the system. The stored DCD information and its associated IRQ (if enabled) are cleared by reading the Status Register and then the Receiver FIFO, or by writing a "1" into the Receiver Reset bit. The DCD

status bit subsequently follows the DCD input when it goes "Low". The DCD input provides character synchronization timing for the receiver during the external sync mode of operation. The receiver will be initialized and data will be sampled on the positive transition of the first full Receive Clock cycle after release of DCD (see Figure 7).

- **Sync Match/Data Terminal Ready (SM/DTR)**

The SM/DTR output provides four functions (see Table 4) depending on the state of the PC1 and PC2 control bits. When the Sync Match mode is selected (PC1 = "1", PC2 = "0"), the output provides a one-bit-wide pulse when a sync code is detected. This pulse occurs for each sync code match even if the receiver has already attained synchronization. The SM output is inhibited when PC2 = "1". The DTR mode (PC1 = "0") provides an output level corresponding to the complement of PC2 (DTR = "0" when PC2 = "1".) (see Table 4.)

- **Transmitter Underflow (TUF)**

The Underflow output indicates the occurrence of a transfer of a "fill character" to the Transmitter Shift Register when the last location (#3) in the Transmit Data FIFO is empty. The Underflow output pulse is approximately a Tx CLK "High" period wide and occurs during the last half of the last bit of the character preceding the "Underflow" (see Figure 4). The Underflow output pulse does not occur when the Tx Sync bit is in the reset state.

# HD46508, HD46508-1

## ADU (Analog Data Acquisition Unit) PRELIMINARY

The HD46508 is a monolithic NMOS device with a 10-bit analog-to-digital converter, a programmable voltage comparator, a 16-channel analog multiplexer and HMCS6800 microprocessor family compatible interface.

Each of 16 analog inputs is either converted to a digital data by the analog-to-digital converter or compared with the specified value by the programmable comparator. The analog-to-digital converter uses successive approximation method as the conversion technique. It's intrinsic resolution is 10 bits but it can be 8 bits if the programmer so desires. The programmable voltage comparator compares the input voltage with the value specified by the programmer. The result (greater than, or smaller than) is reflected to the flag in the status register.

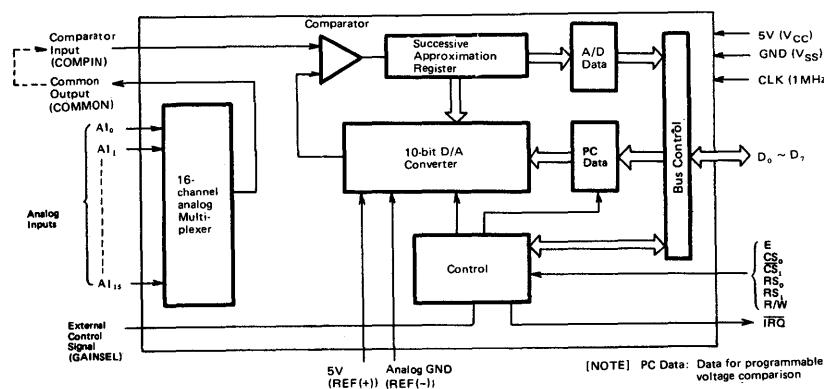
The device can expand its capability by controlling the external circuits such as sample holder, pre-amplifier and external multiplexer.

With these features, this device is ideally suited to applications such as process control, machine control and vehicle control.

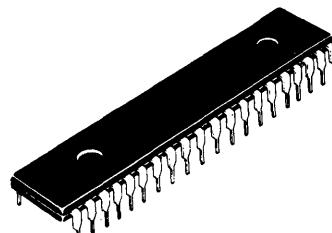
### ■ FEATURES

- 16-channel Analog multiplexer
- Programmable A/D Converter resolution (10-bit or 8-bit)
- Programmable Voltage comparison (PC)
- Conversion Time 100 $\mu$ s (A/D), 13 $\mu$ s(PC)
- External Sample and Hold Circuit Control
- Auto Range-switching Control of External Amplifier
- Waiting Function for the Settling Time of External Amplifier
- Interrupt Control (Only for A/D conversion)
- Single +5V Power Supply
- Compatible with HMCS6800 Bus (The connection with other Asynchronous Buses possible)

### ■ BLOCK DIAGRAM

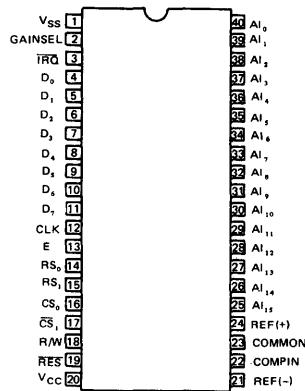


HD46508P, HD46508P-1, HD46508PA, HD46508PA-1



(DP-40)

### ■ PIN ARRANGEMENT

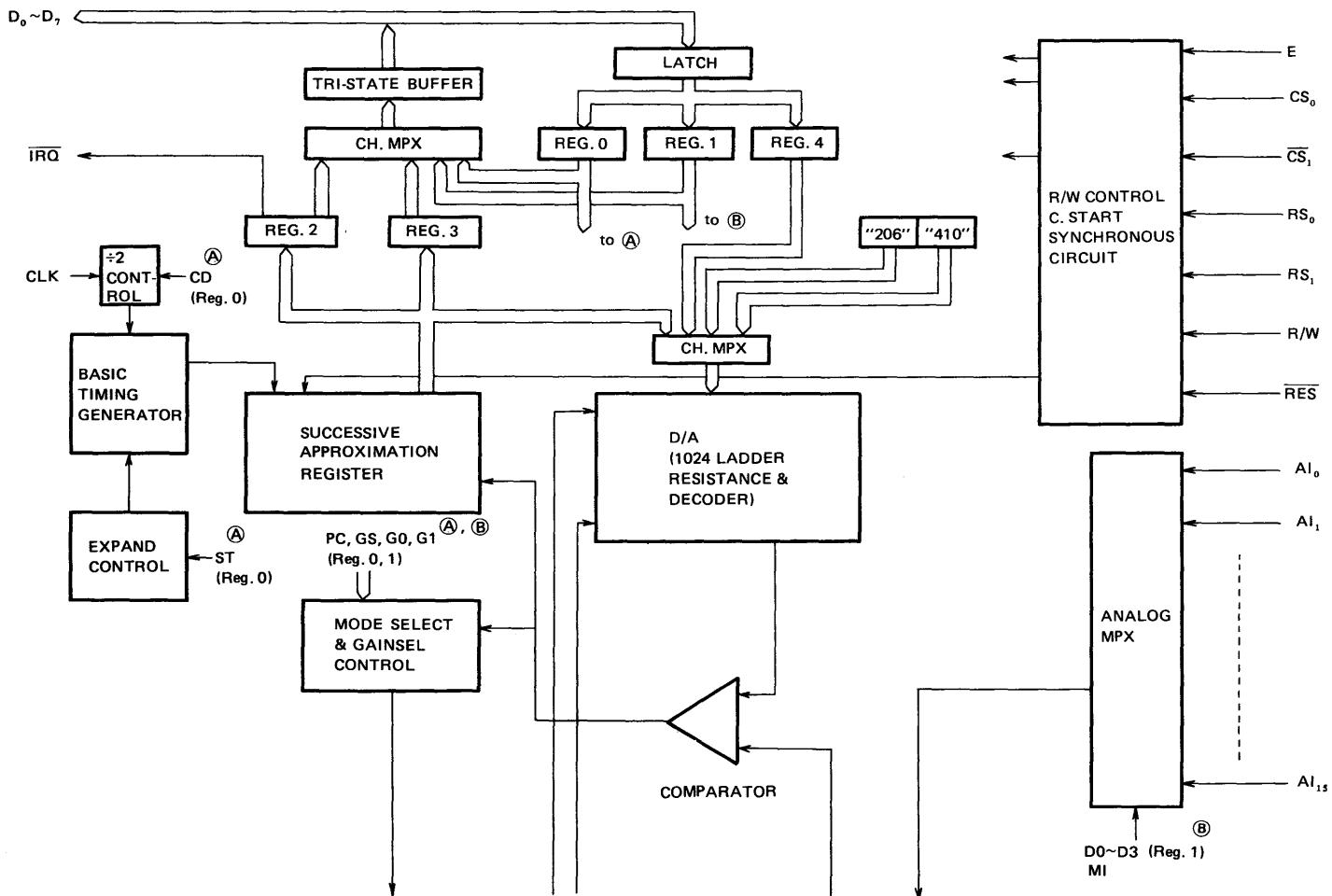


(Top View)

### ■ ORDERING INFORMATION

ADU	Bus Timing	Non Linearity*
HD46508PA	1 MHz	$\pm 1$ LSB
HD46508PA-1	1.5 MHz	
HD46508P	1 MHz	$\pm 3$ LSB
HD46508P-1	1.5 MHz	

\* Specification for 10 bit A/D conversion



"206" : Fixed Data for Auto Range-Switching x 4  
 "410" : Fixed Data for Auto Range-Switching x 2

Figure 1 Internal Block Diagram

## ■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	$V_{CC}^*$	-0.3 ~ +7.0	V
Input Voltage	$V_{in}^*$	-0.3 ~ +7.0	V
Analog Input Voltage	$V_{Ain}^*$	-0.3 ~ +7.0	V
Operating Temperature	$T_{opr}$	-20 ~ +75	°C
Storage Temperature	$T_{stg}$	-55 ~ +150	°C

\* With respect to  $V_{SS}$  (SYSTEM GND)

[NOTE] 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.

## ■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol	min	typ	max	Unit
Supply Voltage	$V_{CC}^*$	4.75	5	5.25	V
Input "High" Voltage	$V_{IH}^*$	2.0	—	$V_{CC}$	V
Input "Low" Voltage	$V_{IL}^*$	-0.3	—	0.8	V
Analog Input Voltage	$V_{Ain}^*$	0	—	5.0	V
Reference Voltage	$V_{REF(+)}^*$	—	5.0	$V_{CC}+0.25$	V
	$V_{REF(-)}^*$	-0.1	0	—	
Voltage Center of Ladder	$\frac{V_{REF(+)} + V_{REF(-)}}{2}$	—	$\frac{V_{CC}}{2}$	$\frac{V_{CC}+0.25}{2}$	V
Operating Temperature	$T_{opr}$	-20	25	75	°C

\*With respect to  $V_{SS}$  (SYSTEM GND)

## ■ ELECTRICAL CHARACTERISTICS

### ● DC CHARACTERISTICS <1> ( $V_{CC} = 5V \pm 5\%$ , $T_a = -20 \sim +75^\circ C$ , unless otherwise noted.)

Item	Symbol	Test Condition	min	typ	max	Unit	
Input "High" Voltage	$V_{IH}$		2.0	—	$V_{CC}$	V	
Input "Low" Voltage	$V_{IL}$		-0.3	—	0.8	V	
Output "High" Voltage	$D_0 \sim D_7$	$V_{OH}$	$I_{OH} = -205\mu A$	2.4	—	—	
	GAINSEL		$I_{OH} = -200\mu A$	2.4	—	—	
			$I_{OH} = -10\mu A$	$V_{CC}-1.0$	—	—	
Output "Low" Voltage	$D_0 \sim D_7$ , GAINSEL	$V_{OL}$	$I_{OL} = 1.6\text{ mA}$	—	—	0.4	
	IRQ		$I_{OL} = 3.2\text{ mA}$	—	—	0.4	
Input Leakage Current	E, CLK, R/W RES, RS <sub>0</sub> , RS <sub>1</sub> CS <sub>0</sub> , CS <sub>1</sub>	$I_{in}$	$V_{in} = 0 \sim 5.25V$	—	—	2.5	$\mu A$
Three-State (off state) Input Current	$D_0 \sim D_7$	$I_{TSI}$	$V_{in} = 0.4 \sim 2.4V$	—	—	10	$\mu A$
Output Leakage Current	IRQ	$I_{LOH}$	$V_{OH} = 2.4V$	—	—	10	$\mu A$
Power Dissipation		$P_D$		—	—	500	mW
Input Capacitance	$D_0 \sim D_7$	$C_{in}$		—	—	12.5	pF
	E, CLK, R/W RES, RS <sub>0</sub> , RS <sub>1</sub> CS <sub>0</sub> , CS <sub>1</sub>		$V_{in} = 0V, T_a = 25^\circ C$ $f = 1\text{ MHz}$	—	—	10.0	pF
Output Capacitance	IRQ, GAINSEL	$C_{out}$	$V_{in} = 0V, T_a = 25^\circ C$ $f = 1\text{ MHz}$	—	—	10.0	pF

● DC CHARACTERISTICS <2> ( $V_{CC} = 5V \pm 5\%$ ,  $T_a = -20 \sim +75^\circ C$ , unless otherwise noted.)

Item	Test Condition	min	typ	max	Unit
Analog Multiplexer ON Resistance	$V_{Ain} = 5.0V$ , $V_{CC} = 4.75V$ , $T_a = 25^\circ C$	—	—	3	kΩ
△ ON Resistance Between any 2 Channels	$V_{Ain} = 5.0V$ , $V_{CC} = 4.75V$ , $T_a = 25^\circ C$	—	—	300	Ω
OFF Channel Leakage Current	$V_{Ain} = 5.0V$ $V_{CC} = 4.75V$ , $T_a = 25^\circ C$ COMMON = 0V	—	10	100	nA
	$V_{Ain} = 0V$ , $T_a = 25^\circ C$ $V_{CC} = 4.75V$ , COMMON = 5V	-100	-10	—	nA
Analog Multiplexer Input Capacitance		—	—	7.5	pF
Ladder Resistance (from REF(+) to REF(-))	$V_{REF(+)} = 5.0V$ $V_{REF(-)} = 0V$ , $T_a = 25^\circ C$	10	—	60	kΩ

● CONVERTER SECTION ( $T_a = 25^\circ C$ ,  $V_{CC} = V_{REF(+)} = 5.0V$ , unless otherwise noted.)

1. 10-BIT A/D CONVERSION

Item	HD46508PA, HD46508PA-1			HD46508P, HD46508P-1			Unit
	min	typ	max	min	typ	max	
Resolution	—	10	—	—	10	—	bits
Non-linearity Error *	—	±1/2	±1	—	±1	±3	LSB
Zero-Error	—	±1/2	±3/4	—	±1/2	±1	LSB
Full-Scall Error	—	±1/4	±1/2	—	±1/2	±1	LSB
Quantization Error	—	—	±1/2	—	—	±1/2	LSB
Absolute Accuracy *	—	±1	±3/2	—	±2	±4	LSB

2. 8-BIT A/D CONVERSION

Item	HD46508PA, HD46508PA-1			HD46508P, HD46508P-1			Unit
	min	typ	max	min	typ	max	
Resolution	—	8	—	—	8	—	bits
Non-linearity Error *	—	±1/8	±1/4	—	±1/4	±3/4	LSB
Zero-Error	—	±1/4	±3/8	—	±3/8	±1/2	LSB
Full-Scall Error	—	±1/4	±3/8	—	±3/8	±1/2	LSB
Quantization Error	—	—	±1/2	—	—	±1/2	LSB
Absolute Accuracy *	—	±5/8	±3/4	—	±3/4	±5/4	LSB

3. PROGRAMMABLE VOLTAGE COMPARISON (PC)

Item	HD46508PA, HD46508PA-1			HD46508P, HD46508P-1			Unit
	min	typ	max	min	typ	max	
Resolution	—	8	—	—	8	—	bits
Non-linearity Error *	—	±1/8	±1/4	—	±1/4	±3/4	LSB
Zero-Error	—	±1/4	±3/8	—	±3/8	±1/2	LSB
Full-Scall Error	—	±1/4	±3/8	—	±3/8	±1/2	LSB
Absolute Accuracy *	—	±3/8	±5/8	—	±1/2	±1	LSB

\*Temperature Coefficient; 25 ppm of FSR/°C (max)

● AC CHARACTERISTICS ( $T_a = -20 \sim +75^\circ C$ )

1. CLOCK WAVEFORM

Item	Symbol	Test Conditions	CD* = 0			CD* = 1			Unit
			min	typ	max	min	typ	max	
CLK Cycle Time	$t_{cycc}$	Fig. 2	1.0	—	10	0.5	—	5	$\mu s$
CLK "High" Pulse Width	$PW_{CH}$		0.45	—	4.5	0.22	—	2.2	$\mu s$
CLK "Low" Pulse Width	$PW_{CL}$		0.40	—	4.0	0.21	—	2.1	$\mu s$
Rise and Fall Time of CLK	$t_{Cr}, t_{Cf}$		—	—	25	—	—	25	ns

\* CD : CLK Divider bit

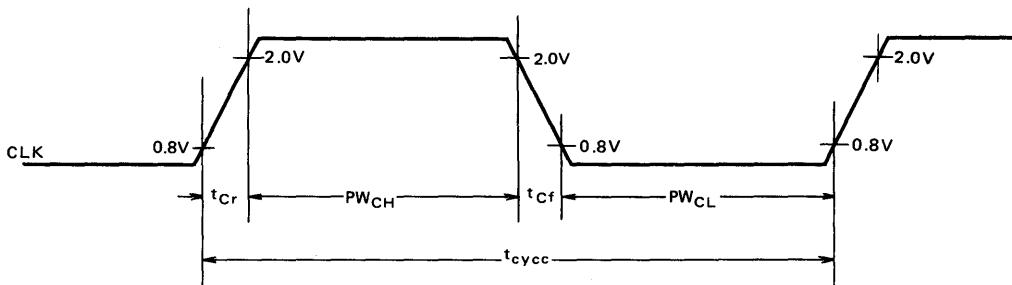


Figure 2 CLK Waveform

2.  $\overline{IRQ}$ , GAINSEL OUTPUT

Item	Symbol	Test condition	min	typ	max	Unit
$\overline{IRQ}$ Release Time	$t_{IR}$	Fig. 3	—	—	650	ns
GAINSEL Delay Time	$t_{GSD1}$	Fig. 4	—	—	750	ns
	$t_{GSD2}$		—	—	750	ns

$t_{GSD1}$  : TTL Load

$t_{GSD2}$  : CMOS Load

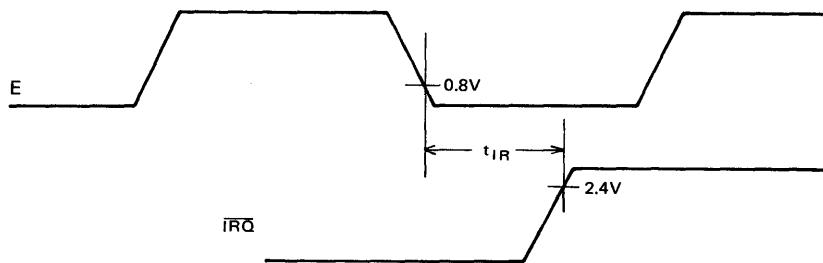
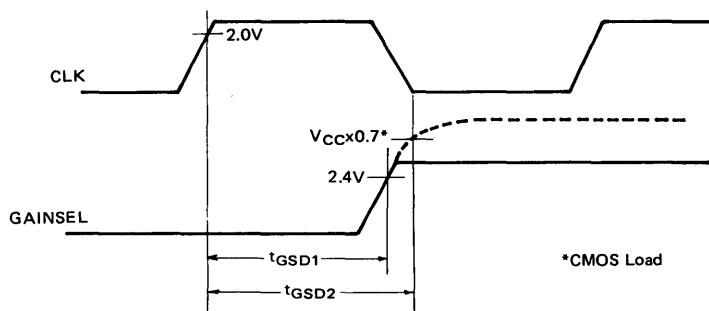


Figure 3  $\overline{IRQ}$  Release Time

## (1) Sample &amp; Hold



## (2) x2, x4 Auto Range-Switching, Programmable Gain

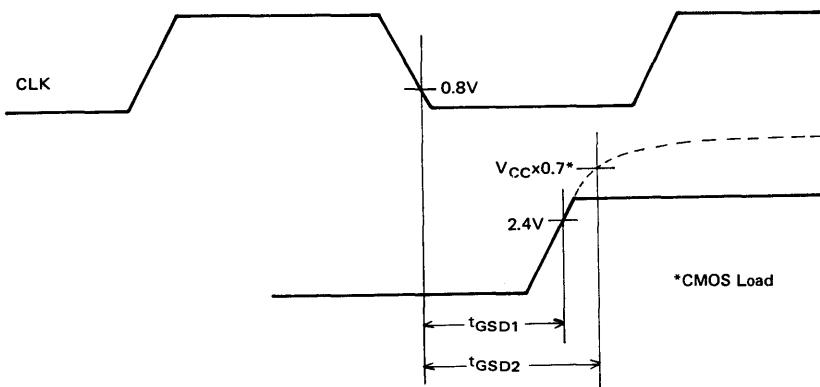


Figure 4 GAINSEL Delay Time

## 2. BUS TIMING CHARACTERISTICS

## READ OPERATION SEQUENCE

Item	Symbol	Test Condition	HD46508P HD46508A			HD46508P-1 HD46508PA-1			Unit
			min	typ	max	min	typ	max	
Enable Cycle Time	t <sub>cycE</sub>	Fig. 5	1.0	—	—	0.666	—	—	μs
Enable "High" Pulse Width	PW <sub>EH</sub>		0.45	—	—	0.28	—	—	μs
Enable "Low" Pulse Width	PW <sub>EL</sub>		0.40	—	—	0.28	—	—	μs
Rise and Fall Time of Enable	t <sub>E<sub>r</sub>,t<sub>E<sub>f</sub></sub></sub>		—	—	25	—	—	25	ns
Address Set Up Time	t <sub>AS</sub>		140	—	—	140	—	—	ns
Data Delay Time	t <sub>DDR</sub>		—	—	320	—	—	220	ns
Data Access Time	t <sub>ACC</sub>		—	—	460	—	—	360	ns
Data Hold Time	t <sub>H</sub>		10	—	—	10	—	—	ns
Address Hold Time	t <sub>AH</sub>		10	—	—	10	—	—	ns

## WRITE OPERATION SEQUENCE

Item	Symbol	Test Condition	HD46508P HD46508PA			HD46508P-1 HD46508PA-1			Unit
			min	typ	max	min	typ	max	
Enable Cycle Time	$t_{cycE}$	Fig. 6	1.0	—	—	0.666	—	—	$\mu s$
Enable "High" Pulse Width	$PW_{EH}$		0.45	—	—	0.280	—	—	$\mu s$
Enable "Low" Pulse Width	$PW_{EL}$		0.40	—	—	0.280	—	—	$\mu s$
Rise and Fall Time of Enable	$t_{Er}, t_{Ef}$		—	—	25	—	—	25	ns
Address Set Up Time	$t_{AS}$		140	—	—	140	—	—	ns
Data Set Up Time	$t_{DSW}$		195	—	—	80	—	—	ns
Data Hold Time	$t_H$		10	—	—	10	—	—	ns
Address Hold Time	$t_{AH}$		10	—	—	10	—	—	ns

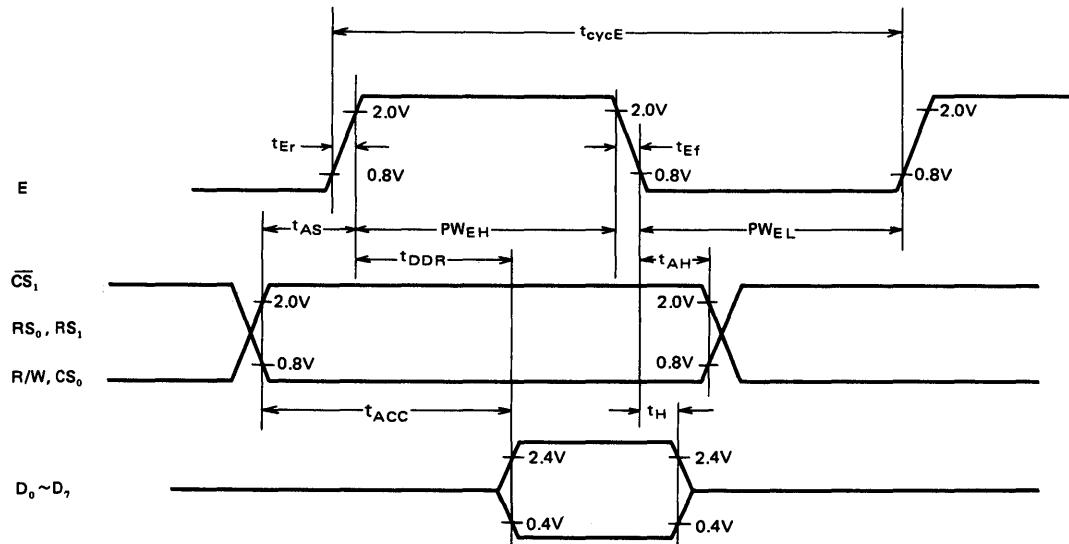


Figure 5 Read Timing

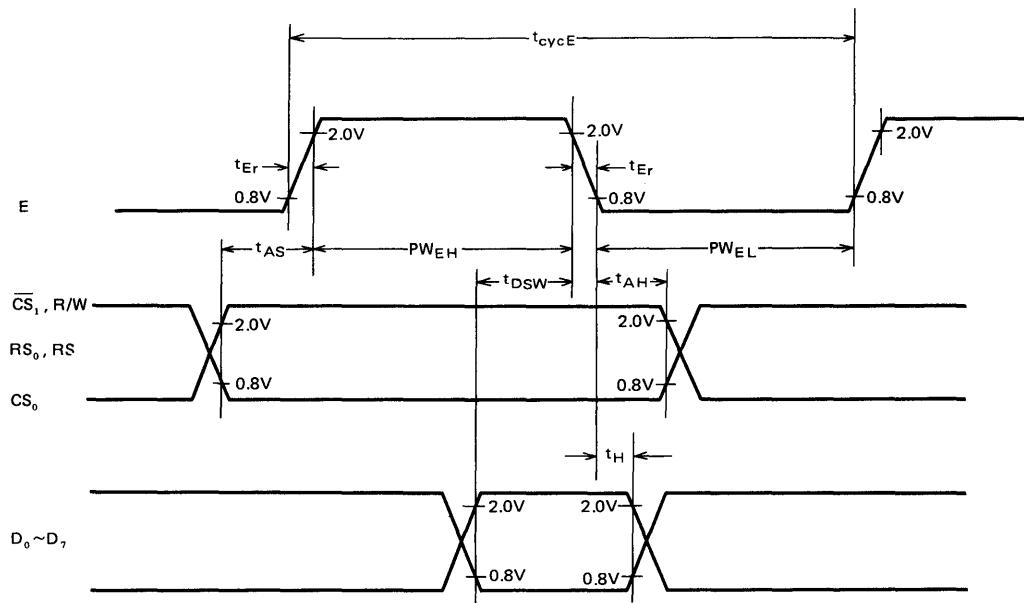


Figure 6 Write Timing

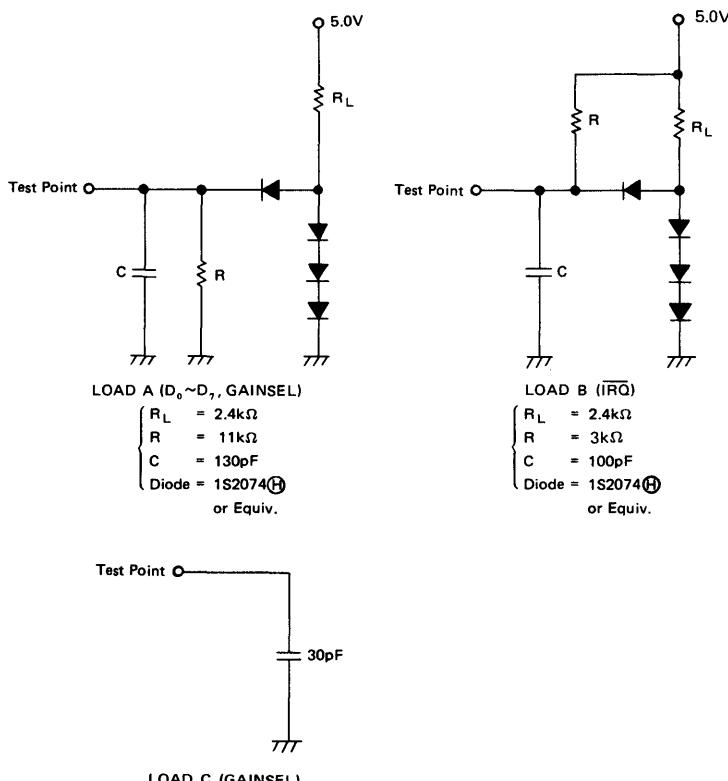


Figure 7 Test Load

## ■ SIGNAL DESCRIPTION

### ● Processor Interface

#### Data Bus ( $D_0 \sim D_7$ )

The Bi-directional data lines ( $D_0 \sim D_7$ ) allow data transfer between the ADU and MPU. Data bus output drivers are three state buffers that remain in the high-impedance state except when MPU performs a ADU read operation.

#### Enable (E)

The Enable signal (E) is used as strobe signal in MPU R/W operation with the ADU internal registers. This signal is normally derived from the HMCS6800 system clock ( $\phi_2$ ).

#### Chip Select ( $CS_0, CS_1$ )

The Chip Select lines ( $CS_0, \overline{CS}_1$ ) are used to address the ADU. The ADU is selected when  $CS_0$  is at "High" and  $\overline{CS}_1$  is at "Low" level.

#### Read/Write (R/W)

The R/W line controls the direction of data transfer between the ADU and MPU. When R/W is at "High" level, data of ADU is transferred to MPU. When R/W is at "Low" level, data of MPU is transferred to ADU.

#### Register Select ( $RS_0, RS_1$ )

The Register Select line ( $RS_0, RS_1$ ) are used to select one of the 4 ADU internal registers. Table 1 shows the relation between ( $RS_0, RS_1$ ) address and the selected register. The lowest 2 address lines of MPU are usually used for these signals.

#### Reset ( $\overline{RES}$ )

This input is used to reset the ADU. An input "Low" level on  $\overline{RES}$  line forces the ADU into following status.

- 1) All the shift-registers in ADU are cleared and the conversion operation is stopped.
- 2) All the outputs go down to "Low" level.

#### Interrupt Request (IRQ) (Open Drain Output)

This output line is used to inform the A/D conversion end signal to the MPU. This signal becomes active "Low" level when IE bit in the control register 1 is "1" and IRQ bit in the control register 2 goes "1" at the end of conversion. Programmable voltage comparison does not affect this signal.

### ● Analog Data Interface

#### Analog Input ( $AI_0 \sim AI_{15}$ )

The Input Analog Data to be measured is applied to the Analog Input ( $AI_0 \sim AI_{15}$ ). These are multiplexed by internal 16 channel multiplexer and output to COMMON pin. Particular input channel is selected when the multiplexer channel address is programmed into the control Register (R1).

#### Multiplexer Common Output (COMMON)

This signal is the output of the 16 channel analog multiplexer, and may be connected to the input of pre-amplifier or sample/hold circuit according to user's purposes. When no external circuit needed, this output should be connected to the COMPIN input.

#### Comparator Input (COMPIN)

This is a high impedance input line that is used to transmit selected analog data to comparator. The COMMON line is usually connected to this input. When external Pre-amplifier or Sample/hold circuit is used, output of these circuits may be connected to this input.

#### Reference Voltage (+) (REF (+))

This line is used to apply the standard voltage to the internal ladder resistors.

#### Reference Voltage (-) (REF (-))

This line is connected to the analog ground.

### ● ADU Control

#### Conversion Clock (CLK)

The CLK is a standard clock input signals which define internal timing for A/D conversion and PC operation.

#### Gain Select (GAINSEL) (CMOS Compatible Output)

This output is used to control the pre-amplifier gain according to the range of analog input signal. By using this output, high accuracy precision A/D conversion is possible.

[NOTE] This LSI is different from other HMCS6800 family LSIs in flowing function

- RES doesn't affect IE bit of R0

## ■ FUNCTION OF INTERNAL REGISTERS

### ● Structure

Table 1 Internal Registers of the ADU

$\overline{CS}_1$	$CS_0$	$RS_1$	$RS_0$	Reg. #	Register Name	Read	Write	Data Bit							
								7	6	5	4	3	2	1	0
0	1	0	0	R0	Control Register 0	<input type="radio"/>	<input type="radio"/>	IE	CD	ST				G1	G0
0	1	0	1	R1	Control Register 1	<input type="radio"/>	<input type="radio"/>	SC	GS	PC	MI	D3	D2	D1	D0
0	1	1	0	R2	Status & A/D Data Register (H)	<input type="radio"/>	<input checked="" type="radio"/>	IRQ	BSY	PCO		OV	DW	C9	C8
0	1	1	1	R3	A/D Data Register (L)	<input type="radio"/>	<input checked="" type="radio"/>	C7	C6	C5	C4	C3	C2	C1	C0
0	1	1	1	R4	PC Data Register	<input checked="" type="radio"/>	<input type="radio"/>	B7	B6	B5	B4	B3	B2	B1	B0

**Control Register 0 (R0)**

7	6	5	4	3	2	1	0			
IE	CD	ST					G1	G0		
									"1"	"0"
									Mode Select	See Table 2
									Not Used	
									Not Used	
									Not Used	
									Settling Time	Available
									CLK Divider	CLK/2
									Interrupt Enable*	Enable IRQ
										Mask IRQ

\*RES doesn't affect IE bit.

**Control Register 1 (R1)**

7	6	5	4	3	2	1	0			
SC	GS	PC	MI	D3	D2	D1	D0			
									"1"	"0"
									MPX Channel Address	See Table 3
									MPX Inhibit	Inhibited
									Prog. Comparator Select	Prog. Comparator mode
									GAINSEL Enable	GAINSEL Enable
									Short-cycle Conversion	8-bit Length
										10-bit Length

Figure 9 Control Register 1

**Status & A/D Data Register (H)**

7	6	5	4	3	2	1	0			
IRQ	BSY	PCO		OV	DW	C9	C8			
									"1"	"0"
									Upper bit (10 bit data)	
									Data Weight	See Table 4.
									Data Over Scale flag	Data is over scale
									Not Used	Within the scale
									Programmable Comparator Output	$V_{Ain} > V_p$
									Busy flag	Under Conversion
									IRQ flag	Requested
										Conversion Completed
										Not Requested

 $V_{Ain}$  : Unknown Input Voltage $V_p$  : Programmed Voltage by R4

C9, C8 bits are cleared when 8 bit A/D conversion is performed.

Figure 10 Status &amp; A/D Data Register (H)

**A/D Data Register (L)**

7	6	5	4	3	2	1	0
C7	C6	C5	C4	C3	C2	C1	C0

Lower order 8 bit Data (Normal 10 bit Conversion)

8 bit Data (8 bit Short-cycle Conversion)

Figure 11 A/D Data Register (L)

PC Data Register

7	6	5	4	3	2	1	0
B7	B6	B5	B4	B3	B2	B1	B0

8 bit Data for Programmable Voltage Comparison

Figure 12 PC Data Register

#### ● Description for the Internal Registers

##### Control Register 0 (R0)

This Register is a 5-bit read/write register that is used to specify Interrupt Enable (IE), CLK Divider (CD), Settling Time (ST) and Mode Select (G0, G1). This Register should be written before writing R1.

##### IE bit: (Interrupt Enable)

IE = "1", Interrupt is requested through the  $\overline{IRQ}$  output.  
IE = "0", Interrupt is masked.

##### CD bit: (Clock Divider)

CD = "1", CLK  $\div 2$  is used as internal clock.  
CD = "0", CLK is used directly.

##### ST bit: (Settling Time)

ST = "1", First comparison is executed after 1 expanded cycle in order to compensate external amplifiers settling delay.  
ST = "0", Cycle is not delayed.

##### G0, G1 bit; (Mode select)

These bits are used to specify the function of GAINSEL signal when GS bit is "1".

SC bit (Short-cycle)	$\begin{cases} \text{SC} = "1", & \text{Short-cycle conversion} \\ & (8 \text{ bit length}) \\ \text{SC} = "0", & \text{Normal conversion} \\ & (10 \text{ bit length}) \end{cases}$
GS bit (GAINSEL Enable)	$\begin{cases} \text{GS} = "1", & \text{GAINSEL signal is enabled. The function of GAINSEL is specified by (G0, G1) bits.} \\ \text{GS} = "0", & \text{GAINSEL signal is disabled. ("Low" level)} \end{cases}$
PC bit (Program comparator)	$\begin{cases} \text{PC} = "1", & \text{Programmable voltage comparator mode} \\ \text{PC} = "0", & \text{A/D conversion mode} \end{cases}$
MI bit (MPX Inhibit)	$\begin{cases} \text{MI} = "1", & \text{Internal MPX channel is inhibited in order to use external MPX channel.} \\ \text{MI} = "0", & \text{Internal MPX channel is used.} \end{cases}$

##### D0~D3 (MPX channel)

These bits are used to select the particular MPX channel.

Table 2 Function of G0, G1

G1	G0	Mode Select
0	0	Sample & Hold
0	1	Auto Range-Switching $\times 2$
1	0	Auto Range-Switching $\times 4$
1	1	Programmable Gain Control

##### Control Register 1 (R1)

This register is an 8-bit read/write register that is used to store the command for A/D conversion mode and programmable comparison mode. This register includes MPX channel address ( $D_0 \sim D_3$ ), MPX inhibit (MI), programmable comparator select (PC), GAINSEL enable (GS) and short-cycle conversion (SC) bits. When this register (R1) is programmed, each conversion mode starts.

Table 3 MPX Channel Addressing

Channel #1	D3	D2	D1	D0	Analog Input
0	0	0	0	0	A <sub>l0</sub>
1	0	0	0	1	A <sub>l1</sub>
2	0	0	1	0	A <sub>l2</sub>
3	0	0	1	1	A <sub>l3</sub>
4	0	1	0	0	A <sub>l4</sub>
5	0	1	0	1	A <sub>l5</sub>
6	0	1	1	0	A <sub>l6</sub>
7	0	1	1	1	A <sub>l7</sub>
8	1	0	0	0	A <sub>l8</sub>
9	1	0	0	1	A <sub>l9</sub>
10	1	0	1	0	A <sub>l10</sub>
11	1	0	1	1	A <sub>l11</sub>
12	1	1	0	0	A <sub>l12</sub>
13	1	1	0	1	A <sub>l13</sub>
14	1	1	1	0	A <sub>l14</sub>
15	1	1	1	1	A <sub>l15</sub>

Table 4 Function Select

PC	SC	Function	GS	(G0, G1)
0	0	10 bit AD CONV.	0	DISABLE
			1	ENABLE*
	1	8 bit AD CONV.	0	DISABLE
			1	ENABLE*
1	x	PROG. COMP (8 bit)	x	DISABLE

x = Do not care

\* = See Table 6

[NOTE] CD bit and ST bit are effective in every case.

**Status & A/D Data Register (H) (R2)**

This register is a 7-bit read only register that is used to store the upper 2-bit data (C8, C9), data weight (DW), data overscale (OV), programmable comparator output (PCO), busy (BSY) and interrupt request(IRQ).

(C8, C9) : These bits store upper 2-bit data measured by 10 bit length conversion.  
(Upper bit data)

DW bit : This bit indicates data weight when Auto range-switching mode is selected. This bit is set or reset when the conversion has completed. The conditions are shown in following Table. In this mode GAINSEL output also goes "High" or "Low" on the same condition shown in Table 5. Other status of DW bit is shown in Table 6.

OV bit  
(Over scale) : This bit is set when analog data is greater than or equal to reference Voltage ( $V_{REF(+)}$ ).

PCO bit  
(Programmable  
comparator  
Output) : This bit indicates the result of programmable voltage comparison.  
"1" → PCO  $V_{Ain} > V_p$   
"0" → PCO  $V_{Ain} < V_p$   
 $V_{Ain}$  : Analog Input Voltage to be compared  
 $V_p$  : Programmed Voltage (R4)

BSY bit  
(Busy) : This bit indicates that the ADU is now under conversion.

IRQ bit  
(Interrupt  
Request) : This bit is set when the A/D conversion has completed and cleared by reading the R3.

**A/D Data Register (L) (R3)**

This register is an 8-bit read-only register that is used to store the lower 8 bits data of 10-bit conversion or full 8 bits data of the 8-bit conversion.

**PC Data Register (R4)**

This register is an 8-bit write-only register prepared for Programmable Voltage comparison. Stored data is converted to digital voltage, and compared with analog input to be measured. The result of comparison is set into PCO bit.

Table 5 Data Weight (DW) Set or Reset Condition

Mode \ Condition	Set ("1")	Reset ("0")
Auto Range-Switching (x2)	$V_{Ain} < \frac{410}{1024} \cdot V_{REF(+)}$	$V_{Ain} > \frac{410}{1024} \cdot V_{REF(+)}$
Auto Range-Switching (x4)	$V_{Ain} < \frac{206}{1024} \cdot V_{REF(+)}$	$V_{Ain} > \frac{206}{1024} \cdot V_{REF(+)}$

 $V_{Ain}$  : Analog Input Voltage to be measured $V_{REF(+)}$  : Voltage Applied to REF(+)

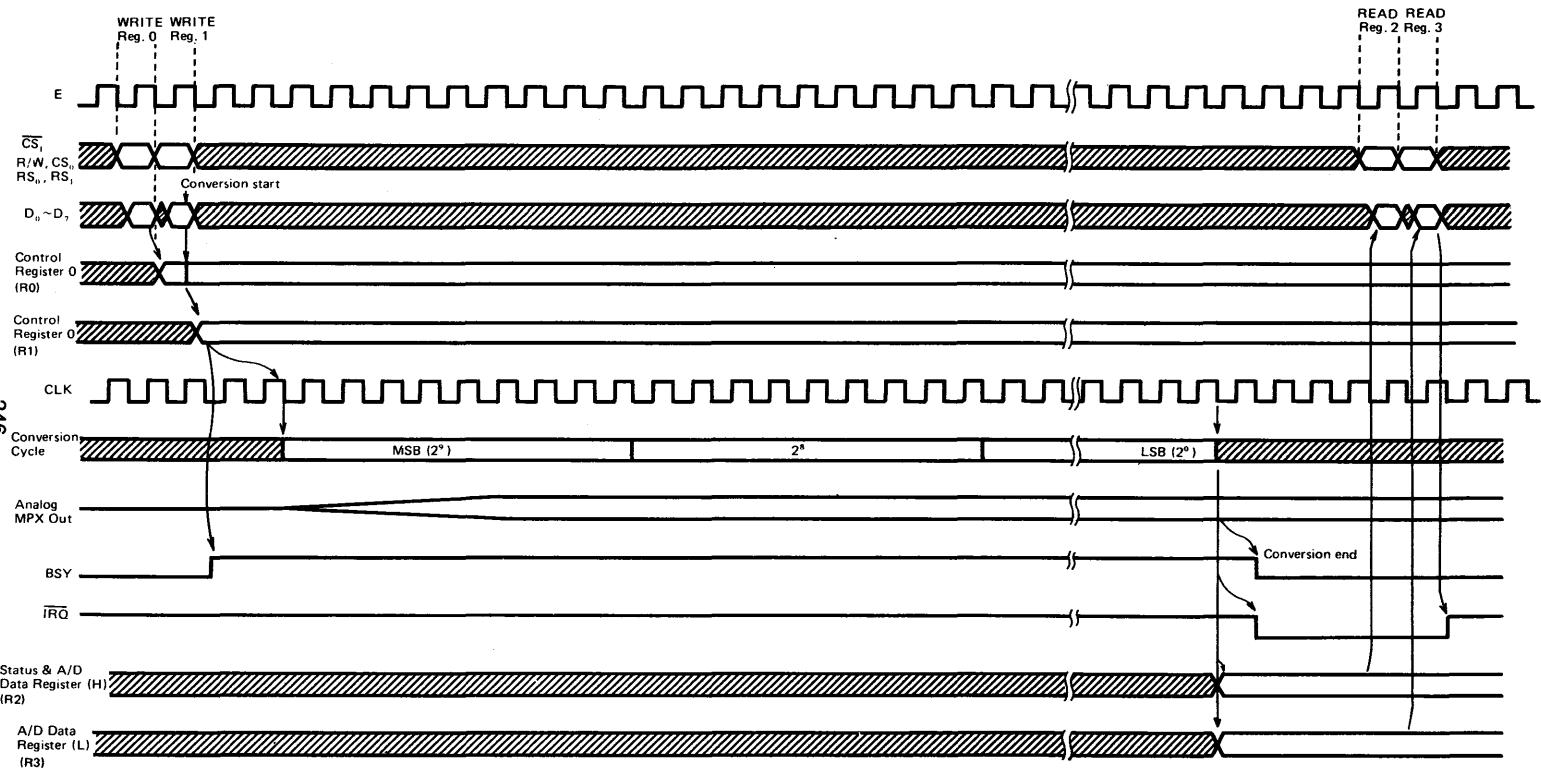


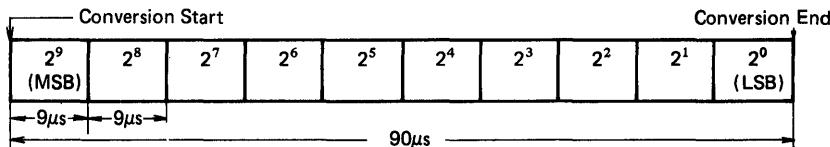
Figure 13 A/D Conversion Timing Chart (Basic Sequence)

• A/D Conversion and PC sequence ( $t_{cyc}=1\mu s$ )

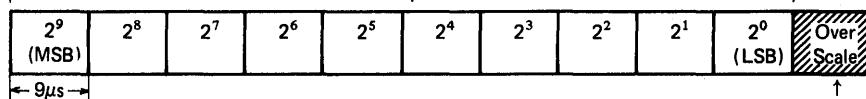
10 bits A/D Conversion

1) Basic Sequence

{  
SC = "0"  
ST = "0"  
GS = "0"

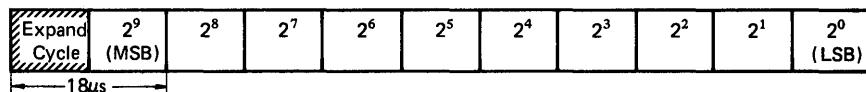


2) Basic Sequence  
(When overscale is detected)



3) Expanded Sequence

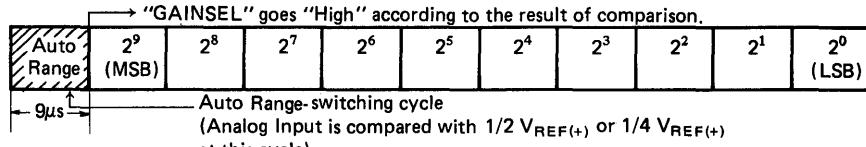
{  
SC = "0"  
ST = "1"  
GS = "0"



MSB cycle is expanded to compensate external amplifier's settling delay.

4) Auto Range-Switching Control Sequence

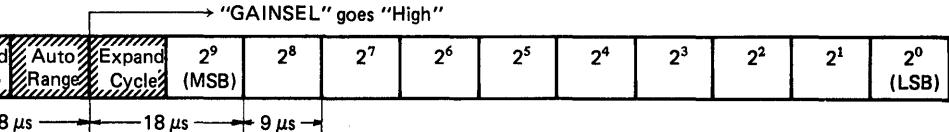
{  
SC = "0"  
ST = "0"  
GS = "1"  
(G0 = "0")  
(G1 = "1")  
or  
(G0 = "1")  
(G1 = "0")



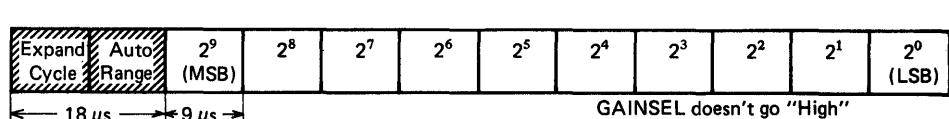
Auto Range-switching cycle  
(Analog Input is compared with 1/2 VREF(+) or 1/4 VREF(+) at this cycle)

5) Auto Range-Switching & Expansion Control Sequence

{  
SC = "0"  
ST = "1"  
GS = "1"  
(G0 = "0")  
(G1 = "1")  
or  
(G0 = "1")  
(G1 = "0")



a) Analog Input < 1/2 VREF(+) or 1/4 VREF(+)

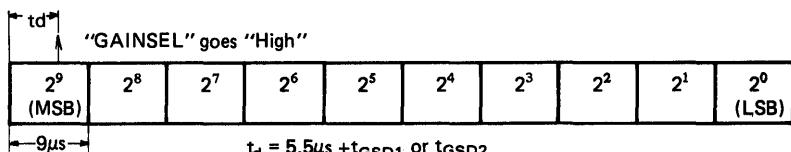


b) Analog Input > 1/2VREF(+) or 1/4VREF(+)

GAINSEL doesn't go "High"

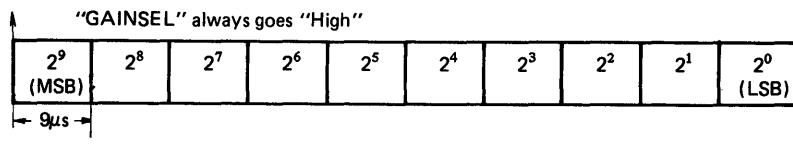
6) Sample & Hold Control Sequence

{  
SC = "0"  
ST = "0"  
GS = "1"  
(G0 = "0")  
(G1 = "0")

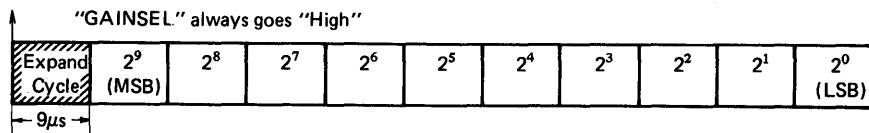


$t_d = 5.5\mu s + t_{GSD1} \text{ or } t_{GSD2}$

- 7) Programmable Gain Control Sequence  
 $(SC = "0")$   
 $(ST = "0")$   
 $(GS = "1")$   
 $(G0 = "1")$   
 $(G1 = "1")$

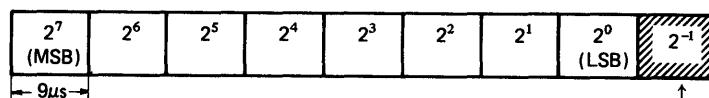


- 8) Programmable Gain & Expansion Control Sequence  
 $(SC = "0")$   
 $(ST = "1")$   
 $(GS = "1")$   
 $(G0 = "1")$   
 $(G1 = "1")$



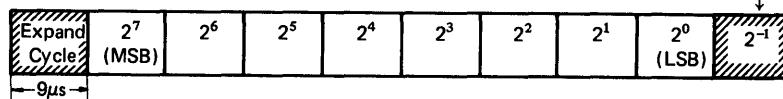
#### 8 Bit A/D Conversion

- 1) Basic Sequence  
 $(SC = "1")$   
 $(ST = "0")$   
 $(GS = "0")$



Additional conversion cycle  
for rounding the LSB - 1 Bit.

- 2) Expanded Sequence  
 $(SC = "1")$   
 $(ST = "1")$   
 $(GS = "0")$

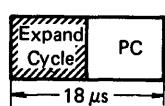


#### Programmable Voltage Comparison

- 1) Basic Sequence  
 $(PC = "1")$   
 $(ST = "0")$



- 2) Expanded Sequence  
 $(PC = "1")$   
 $(ST = "1")$



#### ■ HOW TO USE THE ADU

##### ● Functions of GAINSEL

The ADU is equipped with programmable GAINSEL output signal. By using GAINSEL output and external circuit, the ADU is able to implement following control.

1) Auto Range-Switching (Auto Gain) Control

2) Programmable Gain control

3) Sample & Hold control

GAINSEL output is controlled by Mode Select bit (G0, G1) when GAINSEL enable bit (GS) is "1".

Table 6 GAINSEL Control

GS	G1	G0	GAINSEL	Control Mode	DW
0	x	x	"Low"	Normal Use (GAINSEL is not used)	0
1	0	0	"High"	Sample & Hold control	0
1	0	1	*	Auto Range Switching x 2 control	**
1	1	0	*	Auto Range Switching x 4 control	**
1	1	1	"High"	Programmable Gain control	1

\* GAINSEL goes "High" or "Low" according to the condition shown in Table 5.

\*\* See, Table 5.

- Additional Circuits for GAINSEL Use  
 $x1, x4$  Auto/Programmable Gain

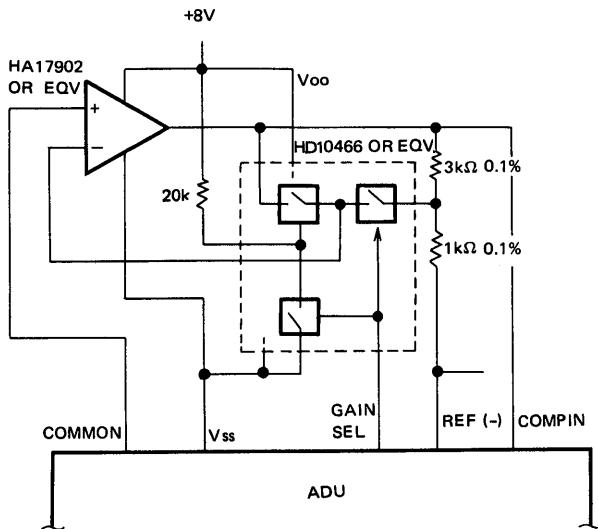


Figure 14 Pre-amplifier Circuit  
( $x1, x4$  Auto-Range Switching)

#### $x1$ Sample & Hold

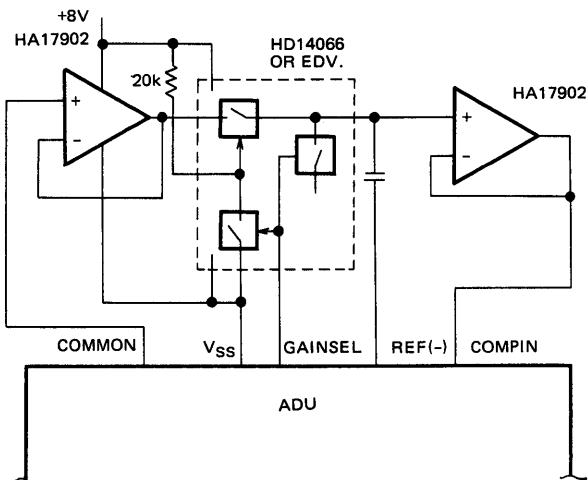
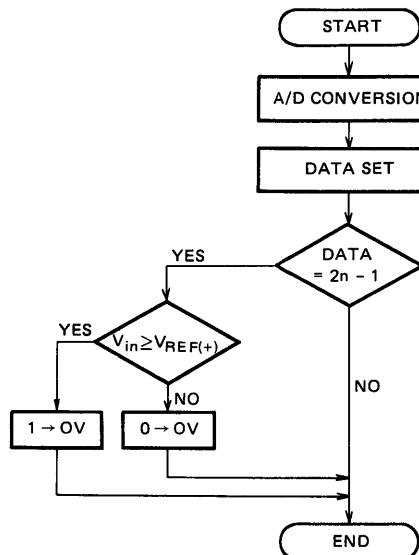


Figure 15 Sample & Hold Circuit

#### • Overscale Check

ADU is equipped with hardware overscale detection function. The overscale detection is performed automatically when the result of A/D conversion is  $2^n - 1$  (all bits = 1). When analog input  $V_{Ain}$  is higher than  $V_{REF(+)}$ , overscale bit (OV) is set to "1". The definition of the overscale is illustrated in Fig. 17. And the flow of overscale check is shown in Fig. 16.



OV	DATA	NOTE
0	11 ..... 1	NOT OVERSCALE
1	11 ..... 1	OVERSCALE

Figure 16 Overscale Check Flow

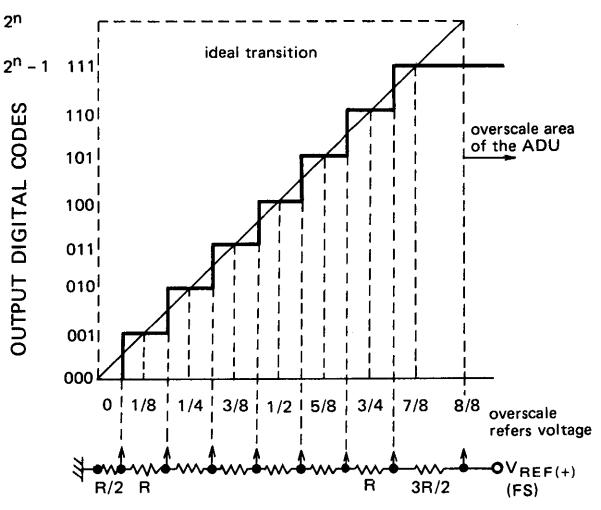


Figure 17 Definition ADU's Overscale

- Usage of the PC

The ADU has a programmable threshold voltage comparator (PC) function. The threshold voltage is pre-setable from 0V to 5V range with 8 bit resolution. The comparator's

output is stored into PC0 bit at the end of comparison.

The programmable voltage comparison time is so short that the interrupt is not requested at this mode. The end of comparison needs to be confirmed by reading the 1 $\rightarrow$ 0 transition of the BSY bit in R2.

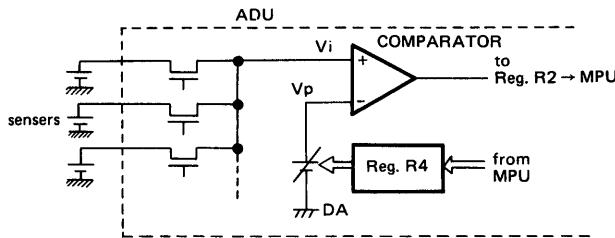


Figure 18 Function Diagram of the PC

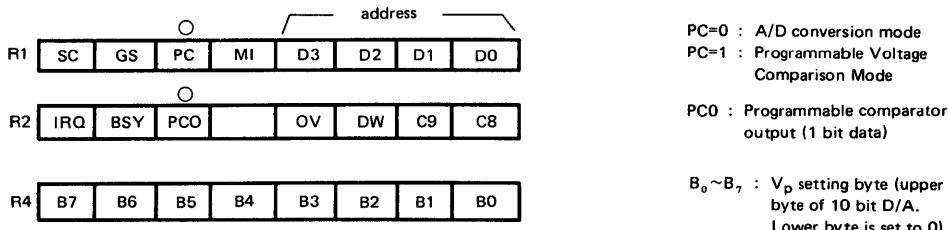


Figure 19 Registers of the PC Mode

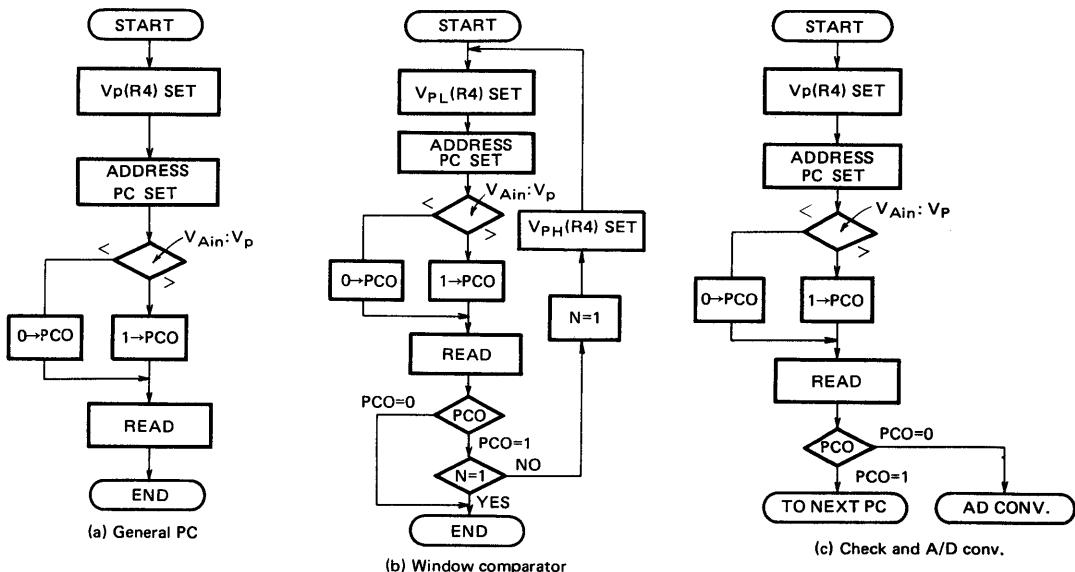
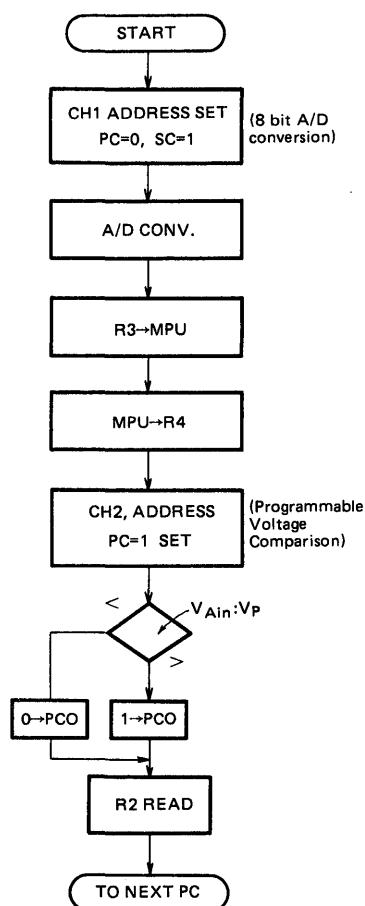


Figure 20 PC Application Flow Chart Examples



(d) Voltage Comparison between two channels.

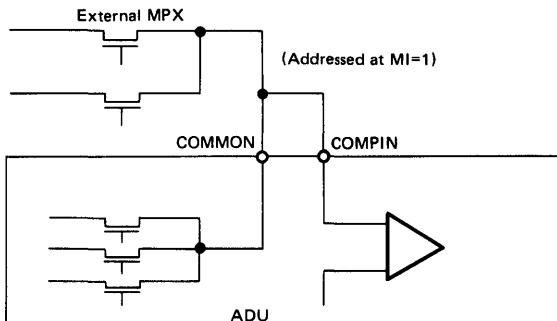
Figure 20 PC Application Flow Chart Examples (continued)

#### • How to use MI bit

MI bit (R1) functions as follows.

- { MI = 1: Internal MPX channel is inhibited in order to use attached external MPX channel.
- MI = 0: Internal MPX channel is enabled.

MI bit used to select either of External MPX and Internal MPX. External MPX is connected as follows.



[NOTE] When external MPX is used as the way figure 20,  
1 dummy AD conversion or PC at MI=1 should be  
performed.

Figure 21 How to use External MPX

#### ■ EXAMPLE OF APPLIED CIRCUIT OF THE ADU

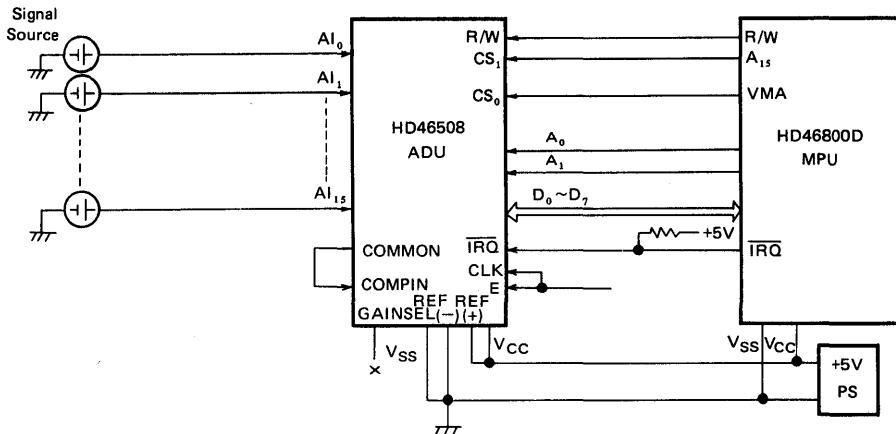


Figure 22 Single ADU System

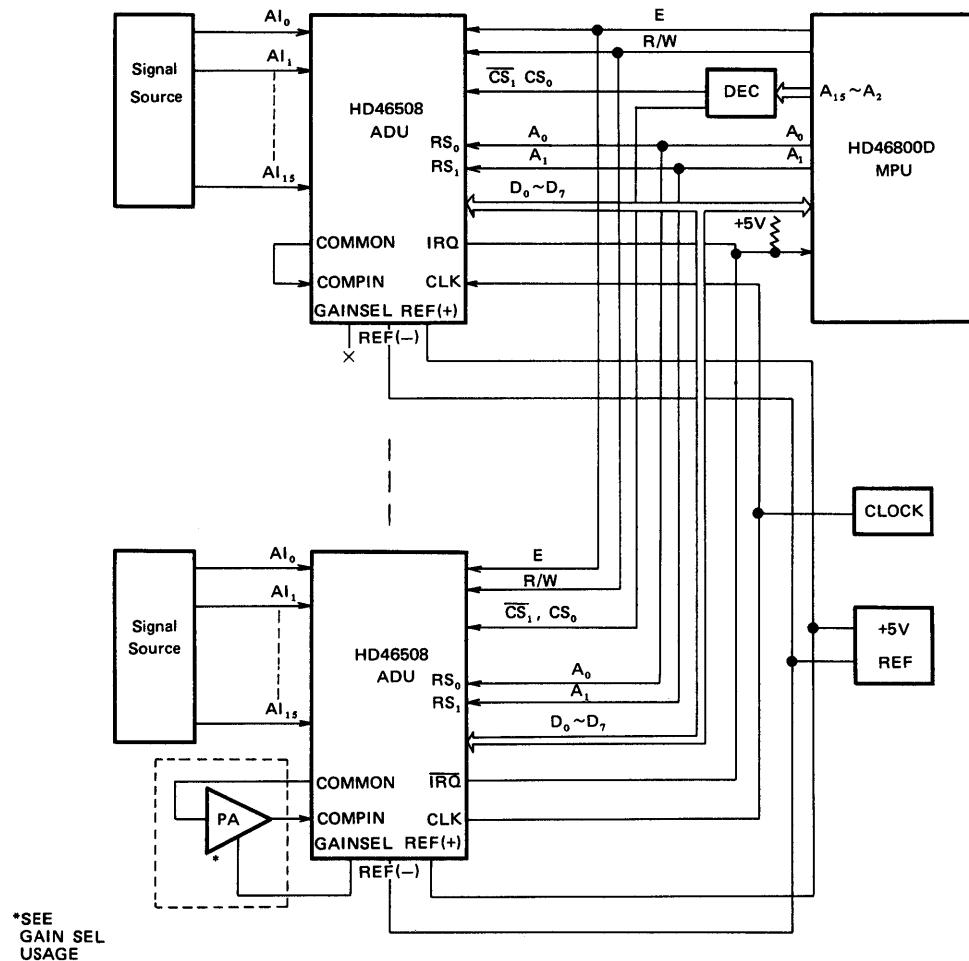


Figure 23 Multi ADU System

HITACHI reserves the right to make changes to any products herein to improve functioning or design. Although the information in this document has been carefully reviewed and is believed to be reliable, HITACHI does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others.

# **HD68000**

# **MPU (Micro Processing Unit) PRELIMINARY**

Advances in semiconductor technology have provided the capability to place on a single silicon chip a microprocessor at least an order of magnitude higher in performance and circuit complexity than has been previously available. The HD68000 is one of such VLSI microprocessors. It combines state-of-the-art technology and advanced circuit design techniques with computer sciences to achieve an architecturally advanced 16-bit microprocessor.

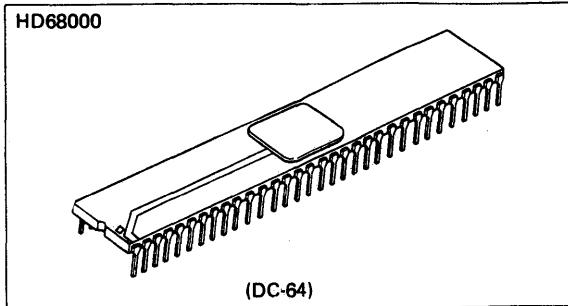
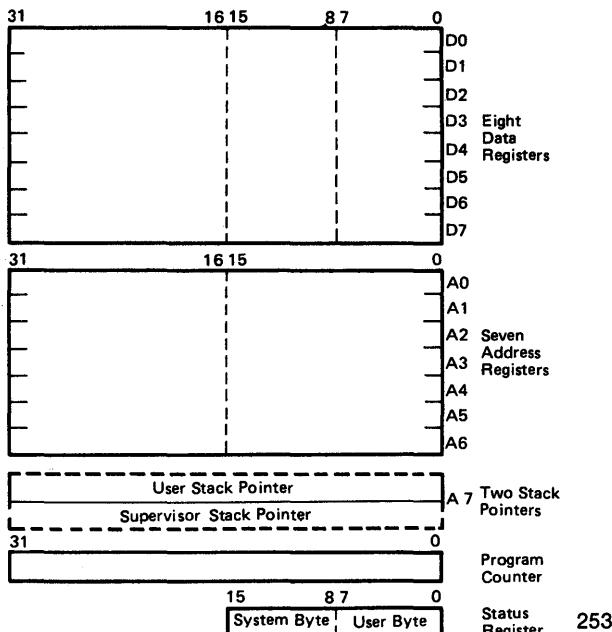
The resources available to the HD68000 user consist of the following:

As shown in the programming model, the HD68000 offers seventeen 32-bit registers in addition to the 32-bit program counter and a 16-bit status register. The first eight registers (D0-D7) are used as data registers for byte (8-bit), word (16-bit), and long word (32-bit) data operations. The second set of seven registers (A0-A6) and the system stack pointer may be used as software stack pointers and base address registers. In addition, these registers may be used for word and long word address operations. All 17 registers may be used as index registers.

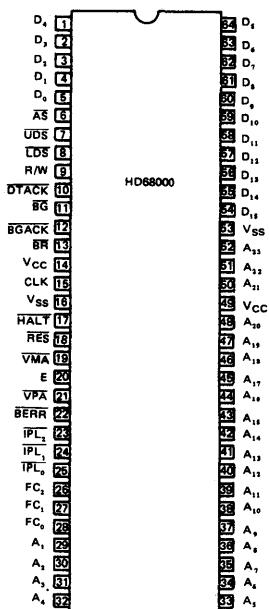
## ■ FEATURES

- 32-Bit Data and Address Registers
  - 16 Megabyte Direct Addressing Range
  - 56 Powerful Instruction Types
  - Memory Mapped I/O
  - 14 Addressing Modes
  - Compatible with MC68000L

## ■ PROGRAMMING MODEL



#### ■ PIN ARRANGEMENT



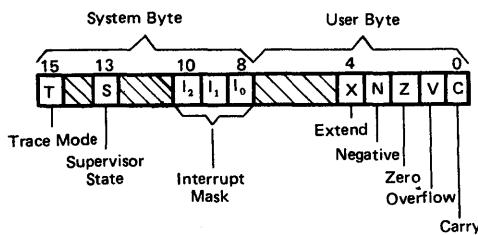
(Top View)

**These information and specification  
are subject to change without notice.**

A 23-bit address bus provides a memory addressing range of greater than 16 megabytes. This large range of addressing capability, coupled with a memory management unit, allows large, modular programs to be developed and operated without resorting to cumbersome and time consuming software bookkeeping and paging techniques.

The status register contains the interrupt mask (eight levels available) as well as the condition codes; extend(X), negative (N), zero (Z), overflow (V), and carry (C). Additional status bits indicate that the processor is in a trace(T) mode and/or in a supervisor (S) state.

## ■ STATUS REGISTER



Five basic data types are supported. These data types are:

- Bits
- BCD Digits (4-bit)
- Bytes (8-bit)
- Word (16-bit)
- Long Words (32-bit)

In addition, operations on other data types such as memory addresses, status word data, etc., are provided for in the instruction set.

The 14 addressing modes, shown in Table 1, include six basic types:

- Register Direct
- Register Indirect
- Absolute
- Immediate
- Program Counter Relative
- Implied

Included in the register indirect addressing modes is the capability to do postincrementing, predecrementing, offsetting and indexing. Program counter relative mode can also be modified via indexing and offsetting.

Table 1 Data Addressing Modes

Mode	Generation
<b>Register Direct Addressing</b>	
Data Register Direct	EA = D <sub>n</sub>
Address Register Direct	EA = A <sub>n</sub>
<b>Absolute Data Addressing</b>	
Absolute Short	EA = (Next Word)
Absolute Long	EA = (Next Two Words)
<b>Program Counter Relative Addressing</b>	
Relative with Offset	EA = (PC) + d <sub>16</sub>
Relative with Index and Offset	EA = (PC) + (X <sub>n</sub> ) + d <sub>8</sub>
<b>Register Indirect Addressing</b>	
Register Indirect	EA = (A <sub>n</sub> )
Postincrement Register-Indirect	EA = (A <sub>n</sub> ), An ← An + N
Predecrement Register Indirect	An ← An - N, EA = (A <sub>n</sub> )
Register Indirect With Offset	EA = (A <sub>n</sub> ) + d <sub>16</sub>
Indexed Register Indirect With Offset	EA = (A <sub>n</sub> ) + (X <sub>n</sub> ) + d <sub>8</sub>
<b>Immediate Data Addressing</b>	
Immediate	DATA = Next Word(s)
Quick Immediate	Inherent Data
<b>Implied Addressing</b>	
Implied Register	EA = SR, USP, SP, PC

(NOTES)

EA = Effective Address  
 An = Address Register  
 D<sub>n</sub> = Data Register  
 X<sub>n</sub> = Address or Data Register used as Index Register  
 SR = Status Register  
 PC = Program Counter  
 ( ) = Contents of  
 d<sub>8</sub> = Eight-bit Offset (displacement)  
 d<sub>16</sub> = Sixteen-bit Offset (displacement)  
 N<sub>16</sub> = 1 for Byte, 2 for Words and 4 for Long Words  
 ← = Replaces

The HD68000 instruction set is shown in Table 2. Some additional instructions are variations, or subsets, of these and they appear in Table 3. Special emphasis has been given to the instruction set's support of structured high-level languages to facilitate ease of programming. Each instruction, with few exceptions, operates on bytes, words, and long words and most instructions can use any of the 14 addressing modes. Combining instruction types, data types, and addressing modes, over 1000 useful instructions are provided. These instructions include signed and unsigned multiply and divide, "quick" arithmetic operations, BCD arithmetic and expanded operations (through traps).

Table 2 Instruction Set

Mnemonic	Description	Mnemonic	Description
ABCD	Add Decimal with Extend	MOVEM	Move Multiple Registers
ADD	Add	MOVEP	Move Peripheral Data
AND	Logical And	MULS	Signed Multiply
ASL	Arithmetic Shift Left	MULU	Unsigned Multiply
ASR	Arithmetic Shift Right	NBCD	Negate Decimal with Extend
Bcc	Branch Conditionally	NEG	Negate
BCHG	Bit Test and Change	NOP	No Operation
BCLR	Bit Test and Clear	NOT	One's Complement
BRA	Branch Always	OR	Logical Or
BSET	Bit Test and Set	PEA	Push Effective Address
BSR	Branch to Subroutine	RESET	Reset External Devices
BTST	Bit Test	ROL	Rotate Left without Extend
CHK	Check Register Against Bounds	ROR	Rotate Right without Extend
CLR	Clear Operand	ROXL	Rotate Left with Extend
CMP	Compare	ROXR	Rotate Right with Extend
DBcc	Test Condition, Decrement and Branch	RTE	Return from Exception
DIVS	Signed Divide	RTR	Return and Restore
DIVU	Unsigned Divide	RTS	Return from Subroutine
EOR	Exclusive Or	SBCD	Subtract Decimal with Extend
EXG	Exchange Registers	Scc	Set Conditional
EXT	Sign Extend	STOP	Stop
JMP	Jump	SUB	Subtract
JSR	Jump to Subroutine	SWAP	Swap Data Register Halves
LEA	Load Effective Address	TAS	Test and Set Operand
LINK	Link Stack	TRAP	Trap
LSL	Logical Shift Left	TRAPV	Trap on Overflow
LSR	Logical Shift Right	TST	Test
MOVE	Move	UNLK	Unlink

Table 3 Variations of Instruction Types

Instruction Type	Variation	Description	Instruction Type	Variation	Description
ADD	ADD	Add	MOVE	MOVE	Move
	ADDA	Add Address		MOVEA	Move Address
	ADDQ	Add Quick		MOVEQ	Move Quick
	ADDI	Add Immediate		MOVE from SR	Move from Status Register
	ADDX	Add with Extend		MOVE to SR	Move to Status Register
AND	AND	Logical And	MOVE to CCR	MOVE to CCR	Move to Condition Codes
	ANDI	And Immediate		MOVE USP	Move User Stack Pointer
CMP	CMP	Compare	NEG	NEG	Negate
	CMPA	Compare Address		NEGX	Negate with Extend
	CMPM	Compare Memory	OR	OR	Logical Or
	CMPI	Compare Immediate		ORI	Or Immediate
EOR	EOR	Exclusive Or	SUB	SUB	Subtract
	EORI	Exclusive Or Immediate		SUBA	Subtract Address
				SUBI	Subtract Immediate
				SUBQ	Subtract Quick
				SUBX	Subtract with Extend

### ■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	$V_{CC}^*$	-0.3 to +7.0	V
Input Voltage	$V_{IN}^*$	-0.3 to +7.0	V
Operating Temperature Range	$T_{opr}$	0 ~ 70	°C
Storage Temperature	$T_{stg}$	-55 ~ 150	°C

\* With respect to  $V_{SS}$  (SYSTEM GND)

[NOTE] 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.

### ■ ELECTRICAL CHARACTERISTICS

• DC CHARACTERISTICS ( $V_{CC} = 5V \pm 5\%$ ,  $V_{SS} = 0$ ,  $T_a = 0 \sim +70^\circ C$ , Fig. 1, 2, 3, unless otherwise noted.)

Item	Symbol	Test Condition	min	typ	max	Unit
Input "High" Voltage	$V_{IH}$		2.0	—	$V_{CC}$	V
Input "Low" Voltage	$V_{IL}$		$V_{SS} - 0.3$	—	0.8	V
Input Leakage Current	$I_{in}$		—	1.0	—	$\mu A$
			—	2.0	—	
Three-State (Off State) Input Current	$I_{TSI}$		—	7.0	—	$\mu A$
Output "High" Voltage	$V_{OH}$	$I_{OH} = -400\mu A$	2.4	—	—	V
Output "Low" Voltage	$I_{OL}$	$I_{OL} = 1.6mA$	—	—	0.5	V
	$I_{OL}$	$I_{OL} = 3.2mA$	—	—	0.5	
	$I_{OL}$	$I_{OL} = 5.0mA$	—	—	0.5	
	$I_{OL}$	$I_{OL} = 5.3mA$	—	—	0.5	
Power Dissipation	$P_D$	$f = 8MHz$	—	1.0	—	W
Capacitance (Package Type Dependent)	$C_{in}$	$V_{in} = 0V, T_a = 25^\circ C$ $f = 1MHz$	—	10.0	—	pF

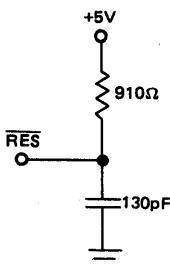


Figure 1 RES Test Load

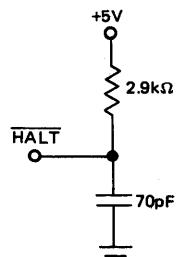
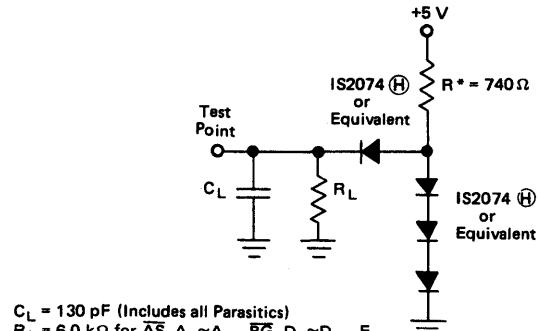


Figure 2 HALT Test Load



$C_L = 130 pF$  (Includes all Parasitics)  
 $R_L = 6.0 k\Omega$  for  $\overline{AS}, A_1 \sim A_{23}, \overline{BG}, D_0 \sim D_{15}, E, FC_0 \sim FC_2, LDS, R/W, UDS, VMA$   
 $*R = 1.22 k\Omega$  for  $A_1 \sim A_{23}, \overline{BG}, E, FC_0 \sim FC_2$

Figure 3 Test Loads

● AC CHARACTERISTICS ( $V_{CC} = 5V \pm 5\%$ ,  $V_{SS} = 0V$ ,  $T_a = 0\sim+70^\circ C$ , unless otherwise noted.)

Number	Item	Symbol	Test Condition	HD68000-4		HD68000-6		HD68000-8		Unit
				min	max	min	max	min	max	
1	Frequency of Operation	f		2	4	2	6	2	8	MHz
2	Clock Period	t <sub>cyc</sub>		250	500	167	500	125	500	ns
3	Clock Width "Low"	t <sub>CL</sub>		115	250	75	250	55	250	ns
4	Clock Width "High"	t <sub>CH</sub>		115	250	75	250	55	250	ns
5	Clock Fall Time	t <sub>Cf</sub>		—	10	—	10	—	10	ns
6	Clock Rise Time	t <sub>Cr</sub>		—	10	—	10	—	10	ns
7	Clock "Low" to Address/FC Valid	t <sub>CLAV</sub>		—	90	—	90	—	90	ns
8	Clock "High" to Address/FC/Data High Impedance (maximum)	t <sub>CHAZx</sub>		—	120	—	100	—	100	ns
9 <sup>1</sup>	Clock "High" to AS, DS "Low" (maximum)	t <sub>CHAZn</sub>		20	—	20	—	20	—	ns
10	Clock "High" to AS, DS "Low" (minimum)	t <sub>CHSLx</sub>		—	80	—	70	—	70	ns
11 <sup>2</sup>	Address/FC Valid to AS, DS (read) "Low"	t <sub>AVSL</sub>		20	—	20	—	20	—	ns
12 <sup>1</sup>	Clock "Low" to AS, DS "High"	t <sub>CLSH</sub>		55	—	35	—	30	—	ns
13 <sup>2</sup>	AS, DS "High" to Address/FC Invalid	t <sub>SHAZ</sub>		—	90	—	80	—	70	ns
14 <sup>2</sup>	AS, DS Width "Low"	t <sub>SL</sub>		60	—	40	—	30	—	ns
15 <sup>2</sup>	AS, DS Width "High"	t <sub>SH</sub>		285	—	170	—	115	—	ns
16	Clock "High" to AS, DS "High" Impedance	t <sub>CHSZ</sub>		285	—	180	—	150	—	ns
17 <sup>2</sup>	DS "High" to R/W "High"	t <sub>SHRH</sub>		—	120	—	100	—	80	ns
18 <sup>1</sup>	Clock "High" to R/W "High" (maximum)	t <sub>CHRhx</sub>		60	—	50	—	30	—	ns
19	Clock "High" to R/W "High" (minimum)	t <sub>CHRhn</sub>		—	90	—	80	—	70	ns
20 <sup>1</sup>	Clock "High" to R/W "Low"	t <sub>CHRL</sub>		10	—	10	—	10	—	ns
21 <sup>2</sup>	Address/FC Valid to R/W "Low"	t <sub>AVR</sub>		—	90	—	80	—	70	ns
22 <sup>2</sup>	R/W "Low" to DS "Low" (write)	t <sub>RLSL</sub>		—	90	—	80	—	70	ns
23	Clock "Low" to Data Out Valid	t <sub>CLDO</sub>		—	120	—	100	—	80	ns
24	Clock "High" to R/W, VMA "High" Impedance	t <sub>CHRZ</sub>		60	—	40	—	30	—	ns
25 <sup>2</sup>	DS "High" to Data Out Invalid	t <sub>SHDO</sub>		55	—	35	—	30	—	ns
26 <sup>2</sup>	Data Out Valid to DS "Low" (write)	t <sub>DOSL</sub>		55	—	35	—	30	—	ns
27	Data In to Clock "Low" (setup time)	t <sub>DICL</sub>		30	—	25	—	15	—	ns
28 <sup>2</sup>	DS "High" to DTACK "High"	t <sub>SHDAH</sub>		0	240	0	160	0	120	ns
29	DS "High" to Data Invalid (hold time)	t <sub>SHDI</sub>		0	—	0	—	0	—	ns
30	AS, DS "High" to BERR "High"	t <sub>SHBEH</sub>		0	—	0	—	0	—	ns
31 <sup>2</sup>	DTACK "Low" to Data In (setup time)	t <sub>DALDI</sub>		—	180	—	120	—	90	ns
32	HALT and RES Input Transition Time	t <sub>RHrf</sub>		0	200	0	200	0	200	ns
33	Clock "High" to BG "Low"	t <sub>CHGL</sub>		—	90	—	80	—	70	ns
34	Clock "High" to BG "High"	t <sub>CHGH</sub>		—	90	—	80	—	70	ns
35	BG "Low" to BG "Low"	t <sub>BRGL</sub>		1.5	3.0	1.5	3.0	1.5	3.0	clk. per.
36	BG "High" to BG "High"	t <sub>BRHG</sub>		1.5	3.0	1.5	3.0	1.5	3.0	clk. per.
37	BGACK "Low" to BG "High"	t <sub>GALGH</sub>		1.5	3.0	1.5	3.0	1.5	3.0	clk. per.
38	BG "Low" to Bus "High" Impedance (with AS high)	t <sub>GLZ</sub>		0	1.5	0	1.5	0	1.5	clk. per.
39	BG Width "High"	t <sub>GH</sub>		1.5	—	1.5	—	1.5	—	clk. per.
40	Clock "Low" to VMA "Low"	t <sub>CLVML</sub>		—	90	—	80	—	70	ns
41	Clock "Low" to E Transition	t <sub>CLE</sub>		—	65	—	60	—	55	ns
42	E Output Rise and Fall Time	t <sub>Er, tEf</sub>		—	25	—	25	—	25	ns
43 <sup>2</sup>	VMA "Low" to E "High"	t <sub>VMLEH</sub>		325	—	240	—	200	—	ns
44	AS, DS "High" to VPA "High"	t <sub>SHVPH</sub>		0	240	0	160	0	120	ns
45	E "Low" to Address/VMA/FC Invalid	t <sub>ELAI</sub>		55	—	35	—	30	—	ns
46	BGACK Width	t <sub>BGL</sub>		1.5	—	1.5	—	1.5	—	clk. per.
47	Asynchronous Input Setup Time	t <sub>ASI</sub>		30	—	25	—	20	—	ns
48	BERR "Low" to DTACK "Low"	t <sub>BELDAL</sub>		50	—	50	—	50	—	ns
49	E "Low" to AS, DS Invalid	t <sub>ELSI</sub>		—80	—	—80	—	—80	—	ns
50	E Width "High"	t <sub>EH</sub>		900	—	600	—	450	—	ns
51	E Width "Low"	t <sub>EL</sub>		1400	—	900	—	700	—	ns
52	E "Low" to Data Out Invalid	t <sub>ELDO</sub>		20	—	20	—	20	—	ns

Fig. 4,5,6

[NOTE] 1. For a loading capacitance of less than or equal to 50 picofarads, subtract 5 nanoseconds from the values given in these columns.  
 2. Actual value depends on actual clock period.

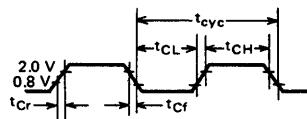
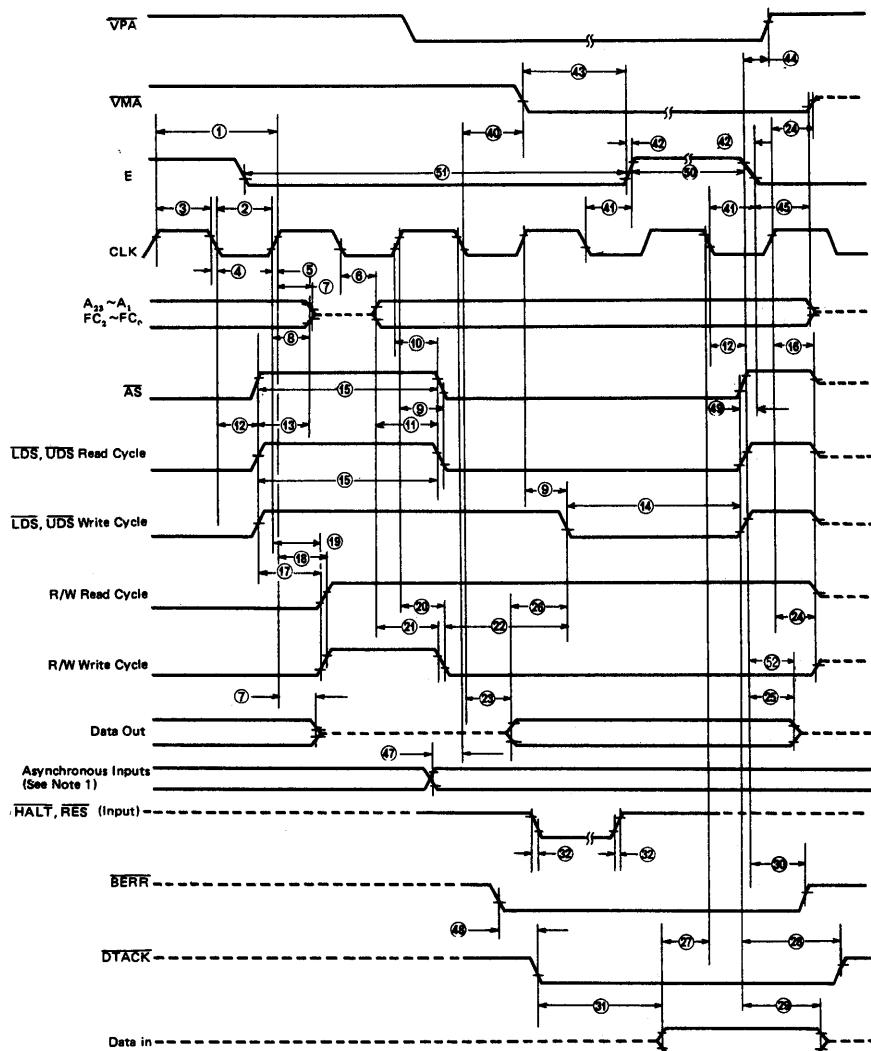


Figure 4 Input Clock Waveform

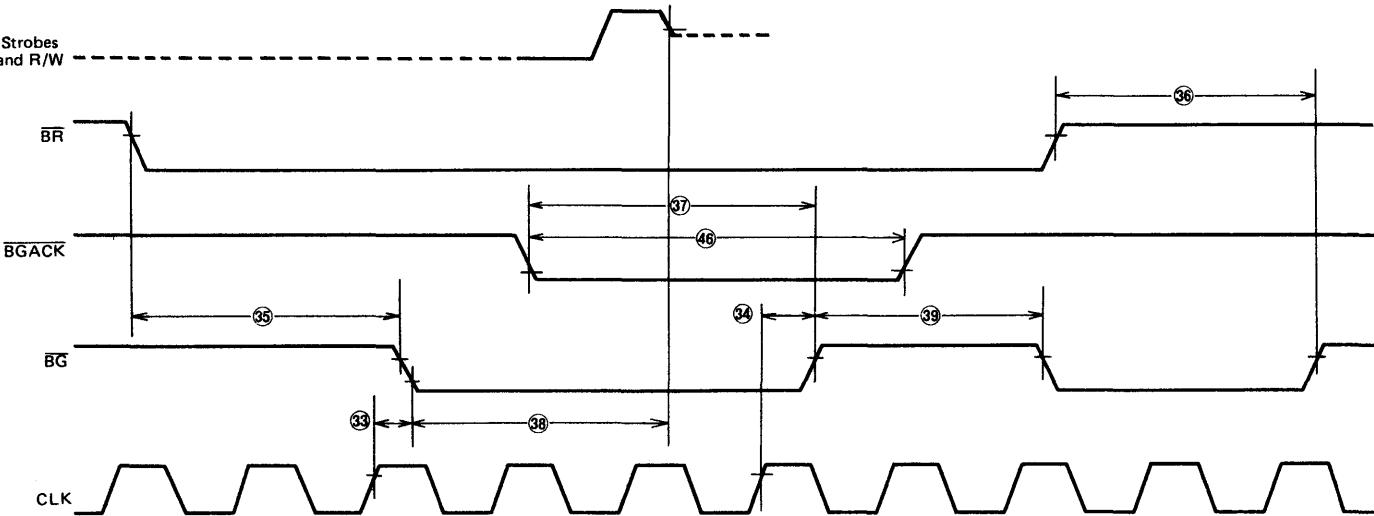
These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.



- [NOTE]
1. Setup time for the asynchronous inputs BERR, BGACK, BR, DTACK, IPL<sub>0</sub>~IPL<sub>2</sub>, and VPA guarantees their recognition at the next falling edge of the clock.
  2. Waveform measurements for all inputs and outputs are specified at: logic high = 2.0 volts, logic low = 0.8 volts

Figure 5 AC Electrical Waveforms  
258

These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.



- [NOTE]
1. Setup time for the asynchronous inputs **BERR**, **BGACK**, **BR**, **DTACK**, **IPL<sub>0</sub>** ~ **IPL<sub>2</sub>**, and **VPA** guarantees their recognition at the next falling edge of the clock.
  2. Waveform measurements for all inputs and outputs are specified at: logic high = 2.0 volts, logic low = 0.8 volts

Figure 6 AC Electrical Waveforms – Bus Arbitration

## ■ DATA ORGANIZATION AND ADDRESSING CAPABILITIES

The following paragraphs describe the data organization and addressing capabilities of the HD68000.

### ● OPERAND SIZE

Operand sizes are defined as follows: a byte equals 8-bits, a word equals 16-bit, and a long word equals 32-bit. The operand size for each instruction is either explicitly encoded in the instruction or implicitly defined by the instruction operation. All explicit instructions support byte, word or long word operands. Implicit instructions support some subset of all three sizes.

### ● DATA ORGANIZATION IN REGISTERS

The eight data registers support data operands of 1, 8, 16, or 32-bit. The seven address registers together with the active stack pointer support address operands of 32-bit.

### ● DATA REGISTERS

Each data register is 32-bit wide. Byte operands occupy the low order 8-bit, word operands the low order 16-bit, and long word operands the entire 32-bit. The least significant bit is addressed as bit zero; the most significant bit is addressed as bit 31.

When a data register is used as either a source or destination operand, only the appropriate low-order portion is changed; the remaining high-order portion is neither used nor changed.

### ● ADDRESS REGISTERS

Each address register and the stack pointer is 32-bit wide and holds a full 32-bit address. Address registers do not support byte sized operands. Therefore, when an address register is used as a source operand, either the low order word or the entire long word operand is used depending upon the operation size. When an address register is used as the destination operand, the entire register is affected regardless of the operation size. If the operation size is word, any other operands are sign extended to 32-bit before the operation is performed.

### ● DATA ORGANIZATION IN MEMORY

Bytes are individually addressable with the high order byte having an even address the same as the word, as shown in Figure 7. The low order byte has an odd address that is one count higher than the word address. Instructions and multibyte data are accessed only on word (even byte) boundaries. If a long word datum is located at address n (n even), then the second word of that datum is located at address n + 2.

The data types supported by the HD68000 are: bit data, integer data of 8, 16, or 32-bit, 32-bit addresses and binary coded decimal data. Each of these data types is put in memory, as shown in Figure 8.

### ● ADDRESSING

Instructions for the HD68000 contain two kinds of information: the type of function to be performed, and the location of the operand(s) on which to perform that function. The methods used to locate (address) the operand(s) are explained in the following paragraphs.

Instructions specify an operand location in one of three ways:

**Register Specification** — the number of the register is given in the register field of the instruction.

**Effective Address** — use of the different effective address modes.

**Implicit Reference** — the definition of certain instructions implies the use of specific registers.

### ● INSTRUCTION FORMAT

Instructions are from one to five words in length, as shown in Figure 9. The length of the instruction and the operation to be performed is specified by the first word of the instruction which is called the operation word. The remaining words further specify the operands. These words are either immediate operands or extensions to the effective address mode specified in the operation word.

### ● PROGRAM/DATA REFERENCES

The HD68000 separates memory references into two classes: program references, and data references. Program references, as the name implies, are references to that section of memory that contains the program being executed. Data references refer to that section of memory that contains data. Generally, operand reads are from the data space. All operand writes are to the data space.

### ● REGISTER SPECIFICATION

The register field within an instruction specifies the register to be used. Other fields within the instruction specify whether the register selected is an address or data register and how the register is to be used.

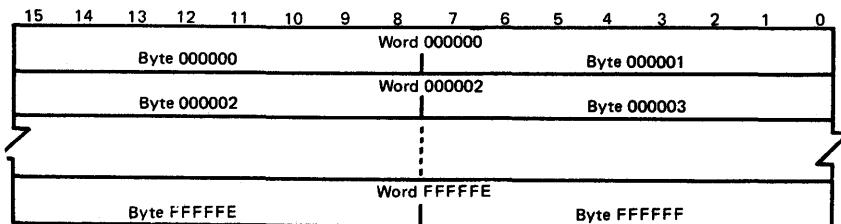
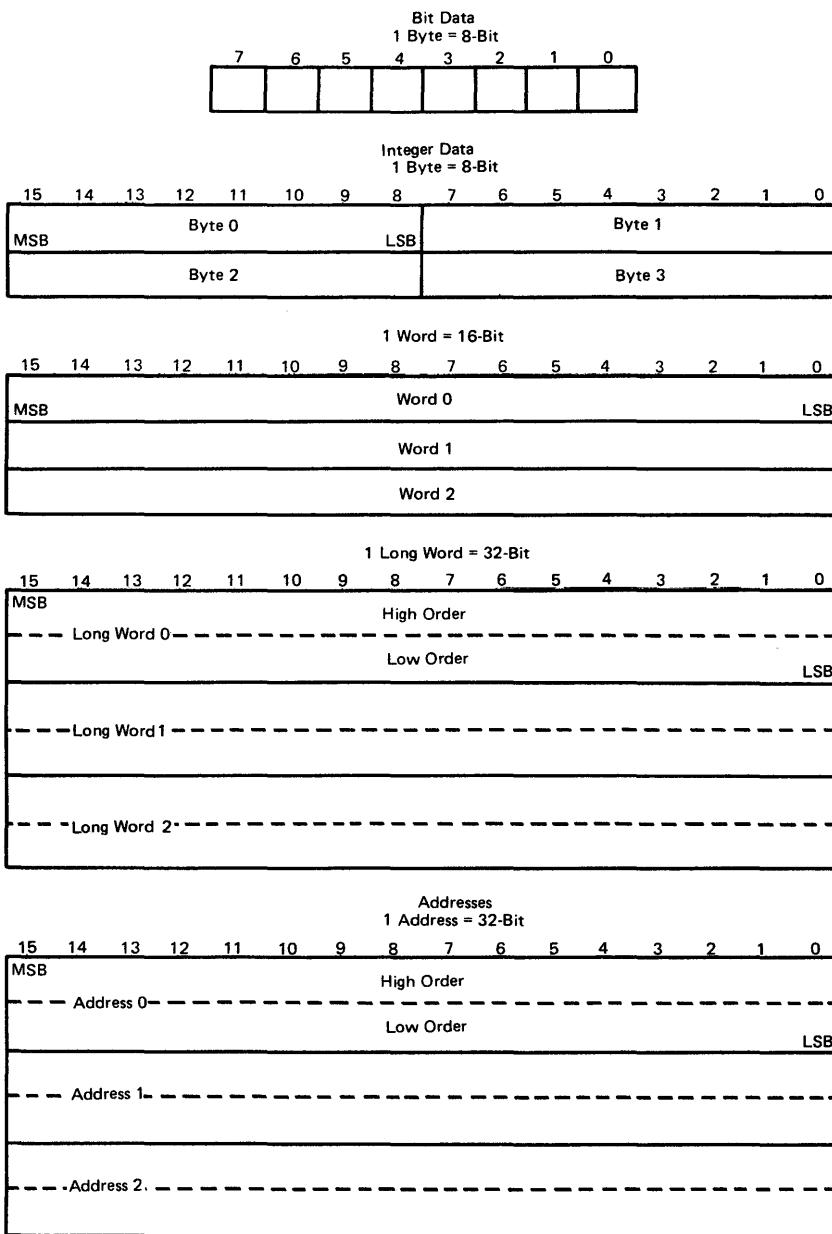
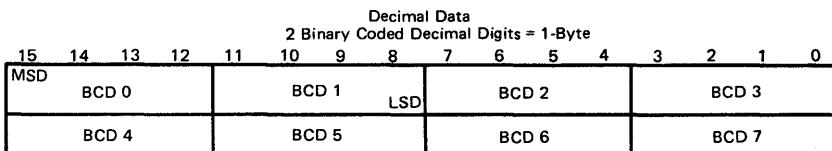


Figure 7 Word Organization In Memory

HD68000



**MSB = Most Significant Bit**  
**LSB = Least Significant Bit**



**MSD = Most Significant Digit**  
**LSD = Least Significant Digit**

**Figure 8 Data Organization in Memory**  
**261**

### • EFFECTIVE ADDRESS

Most instructions specify the location of an operand by using the effective address field in the operation word. For example, Figure 10 shows the general format of the single effective address instruction operation word. The effective address is composed of two 3-bit fields: the mode field, and the register field. The value in the mode field selects the different address modes. The register field contains the number of a register.

The effective address field may require additional information to fully specify the operand. This additional information, called the effective address extension, is contained in the following word or words and is considered part of the instruction, as shown in Figure 9. The effective address modes are grouped into three categories: register direct, memory addressing, and special.

### REGISTER DIRECT MODES

These effective addressing modes specify that the operand is in one of the 16 multifunction registers.

#### Data Register Direct

The operand is in the data register specified by the effective address register field.

#### Address Register Direct

The operand is in the address register specified by the effective address register field.

### MEMORY ADDRESS MODES

These effective addressing modes specify that the operand is in memory and provide the specific address of the operand.

#### Address Register Indirect

The address of the operand is in the address register specified by the register field. The reference is classified as a data reference with the exception of the jump and jump to subroutine instructions.

### Address Register Indirect With Postincement

The address of the operand is in the address register specified by the register field. After the operand address is used, it is incremented by one, two, or four depending upon whether the size of the operand is byte, word, or long word. If the address register is the stack pointer and the operand size is byte, the address is incremented by two rather than one to keep the stack pointer on a word boundary. The reference is classified as a data reference.

### Address Register Indirect With Predecrement

The address of the operand is in the address register specified by the register field. Before the operand address is used, it is decremented by one, two, or four depending upon whether the operand size is byte, word, or long word. If the address register is the stack pointer and the operand size is byte, the address is decremented by two rather than one to keep the stack pointer on a word boundary. The reference is classified as a data reference.

### Address Register Indirect With Displacement

This address mode requires one word of extension. The address of the operand is the sum of the address in the address register and the sign-extended 16-bit displacement integer in the extension word. The reference is classified as a data reference with the exception of the jump and jump to subroutine instructions.

### Address Register Indirect With Index

This address mode requires one word of extension. The address of the operand is the sum of the address in the address register, the signextended displacement integer in the low order eight bits of the extension word, and the contents of the index register. The reference is classified as a data reference with the exception of the jump and jump to subroutine instructions.

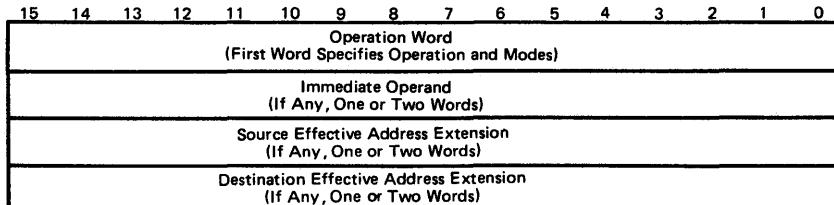


Figure 9 Instruction Format

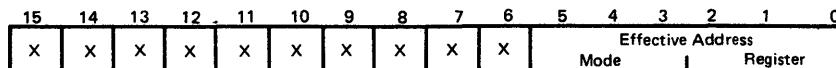


Fig 10 Single-Effective-Address Instruction Operation Word General Format

## SPECIAL ADDRESS MODES

The special address modes use the effective address register field to specify the special addressing mode instead of a register number.

### Absolute Short Address

This address mode requires one word of extension. The address of the operand is the extension word. The 16-bit address is sign extended before it is used. The reference is classified as a data reference with the exception of the jump and jump to subroutine instructions.

### Absolute Long Address

This address mode requires two words of extension. The address of the operand is developed by the concatenation of the extension words. The high-order part of the address is the first extension word; the low-order part of the address is the second extension word. The reference is classified as a data reference with the exception of the jump and jump to subroutine instructions.

### Program Counter With Displacement

This address mode requires one word of extension. The address of the operand is the sum of the address in the program counter and the sign-extended 16-bit displacement integer in the extension word. The value in the program counter is the address of the extension word. The reference is classified as a program reference.

### Program Counter With Index

This address mode requires one word of extension. The address is the sum of the address in the program counter, the sign-extended displacement integer in the lower eight bits of the extension word, and the contents of the index register. The value in the program counter is the address of the extension word. This reference is classified as a program reference.

### Immediate Data

This address mode requires either one or two words of extension depending on the size of the operation.

Byte operation – operand is low order byte of extension word

Word operation – operand is extension word

Long word operation – operand is in the two extension words, high-order 16 bits are in the first extension word, low-order 16 bits are in the second extension word.

### Condition Codes or Status Register

A selected set of instructions may reference the status register by means of the effective address field. These are:

ANDI to CCR

ANDI to SR

EORI to CCR

EORI to SR

ORI to CCR

ORI to SR

## EFFECTIVE ADDRESS ENCODING SUMMARY

Table 4 is a summary of the effective addressing modes discussed in the previous paragraphs.

Table 4 Effective Address Encoding Summary

Addressing Mode	Mode	Register
Data Register Direct	000	register number
Address Register Direct	001	register number
Address Register Indirect	010	register number
Address Register Indirect with Postincrement	011	register number
Address Register Indirect with Predecrement	100	register number
Address Register Indirect with Displacement	101	register number
Address Register Indirect with Index	110	register number
Absolute Short	111	000
Absolute Long	111	001
Program Counter with Displacement	111	010
Program Counter with Index	111	011
Immediate or Status Register	111	100

Table 5 Implicit Instruction Reference Summary

Instruction	Implied Register(s)
Branch Conditional (B <sub>CC</sub> ), Branch Always (BRA)	PC
Branch to Subroutine (BSR)	PC, SP
Check Register against Bounds (CHK)	SSP, SR
Test Condition, Decrement and Branch (DB <sub>CC</sub> )	PC
Signed Divide (DIVS)	SSP, SR
Unsigned Divide (DIVU)	SSP, SR
Jump (JMP)	PC
Jump to Subroutine (JSR)	PC, SP
Link and Allocate (LINK)	SP
Move Condition Codes (MOVE CCR)	SR
Move Status Register (MOVE SR)	SR
Move User Stack Pointer (MOVE USP)	USP
Push Effective Address (PEA)	SP
Return from Exception ( RTE )	PC, SP, SR
Return and Restore Condition Codes (RTR)	PC, SP, SR
Return from Subroutine (RTS)	PC, SP
Trap (TRAP)	SSP, SR
Trap on Overflow (TRAPV)	SSP, SR
Unlink (UNLK)	SP

### • IMPLICIT REFERENCE

Some instructions make implicit reference to the program counter (PC), the system stack pointer (SP), the supervisor stack pointer (SSP), the user stack pointer (USP), or the status register (SR). Table 5 provides a list of these instructions and the registers implied.

### • SYSTEM STACK

The system stack is used implicitly by many instructions; user stacks and queues may be created and maintained through the addressing modes. Address register seven (A7) is the system stack pointer (SP). The system stack pointer is either the supervisor stack pointer (SSP) or the user stack pointer (USP), depending on the state of the S-bit in the status register. If the S-bit indicates supervisor state, SSP is the active system stack pointer, and the USP cannot be referenced as an address

register. If the S-bit indicates user state, the USP is the active system stack pointer, and the SSP cannot be referenced. Each system stack fills from high memory to low memory.

#### ■ INSTRUCTION SET SUMMARY

The following paragraphs contain an overview of the form and structure of the HD68000 instruction set. The instructions form a set of tools that include all the machine functions to perform the following operations:

- Data Movement
- Integer Arithmetic
- Logical
- Shift and Rotate
- Bit Manipulation
- Binary Coded Decimal
- Program Control
- System Control

The complete range of instruction capabilities combined with the flexible addressing modes described previously provide a very flexible base for program development.

#### ● DATA MOVEMENT OPERATIONS

The basic method of data acquisition (transfer and storage) is provided by the move (MOVE) instruction. The move instruction and the effective addressing modes allow both address and data manipulation. Data move instructions allow byte, word, and long word operands to be transferred from memory to memory, memory to register, register to memory, and register to register. Address move instructions allow word and long word operand transfer and ensure that only legal address manipulations are executed. In addition to the general move instruction there are several special data movement instructions: move multiple registers (MOVEM), move peripheral data (MOVEP), exchange registers (EXG), load effective address (LEA), push effective address (PEA), link stack (LINK), unlink stack (UNLK), and move quick (MOVEQ). Table 6 is a summary of the data movement operations.

Table 6 Data Movement Operations

Instruction	Operand Size	Operation
EXG	32	Rx $\leftrightarrow$ Ry
LEA	32	EA $\rightarrow$ An
LINK	—	An $\rightarrow$ SP@ — SP $\rightarrow$ An SP + d $\rightarrow$ SP
MOVE	8, 16, 32	(EA)s $\rightarrow$ EAd
MOVEM	16, 32	(EA) $\rightarrow$ An, Dn An, Dn $\rightarrow$ EA
MOVEP	16, 32	(EA) $\rightarrow$ Dn Dn $\rightarrow$ EA
MOVEQ	8	#xxx $\rightarrow$ Dn
PEA	32	EA $\rightarrow$ SP@ —
SWAP	32	Dn[31:16] $\leftrightarrow$ Dn[15:0]
UNLK	—	An $\rightarrow$ SP SP@ + $\rightarrow$ An

[NOTES] s = source  
d = destination  
[ ] = bit numbers  
@ — = indirect with predecrement  
@ + = indirect with postincrement

#### ● INTEGER ARITHMETIC OPERATIONS

The arithmetic operations include the four basic operations of add (ADD), subtract (SUB), multiply (MUL), and divide (DIV) as well as arithmetic compare (CMP), clear (CLR), and negate (NEG). The add and subtract instructions are available for both address and data operations, with data operations accepting all operand sizes. Address operations are limited to legal address size operands (16 or 32 bits). Data, address, and memory compare operations are also available. The clear and negate instructions may be used on all sizes of data operands.

The multiply and divide operations are available for signed and unsigned operands using word multiply to produce a long word product, and a long word dividend with word divisor to produce a word quotient with a word remainder.

Multiprecision and mixed size arithmetic can be accomplished using a set of extended instructions. These instructions are: add extended (ADDX), subtract extended (SUBX), sign extend (EXT), and negate binary with extend (NEGX).

A test operand (TST) instruction that will set the condition codes as a result of a compare of the operand with zero is also available. Test and set (TAS) is a synchronization instruction useful in multiprocessor systems. Table 7 is a summary of the integer arithmetic operations.

Table 7 Integer Arithmetic Operations

Instruction	Operand Size	Operation
ADD	8, 16, 32	Dn + (EA) $\rightarrow$ Dn (EA) + Dn $\rightarrow$ EA (EA) + #xxx $\rightarrow$ EA
	16, 32	An + (EA) $\rightarrow$ An
	16, 32	Dx + Dy + X $\rightarrow$ Dx Ax@ — + Ay@ — + X $\rightarrow$ Ax@
CLR	8, 16, 32	0 $\rightarrow$ EA
	8, 16, 32	Dn - (EA) (EA) - #xxx Ax@ + - Ay@ + An - (EA)
CMP	16, 32	Dn/(EA) $\rightarrow$ Dn
DIVS	32 $\div$ 16	Dn/(EA) $\rightarrow$ Dn
DIVU	32 $\div$ 16	Dn/(EA) $\rightarrow$ Dn
EXT	8 $\rightarrow$ 16	(Dn) <sub>8</sub> $\rightarrow$ Dn <sub>16</sub>
	16 $\rightarrow$ 32	(Dn) <sub>16</sub> $\rightarrow$ Dn <sub>32</sub>
MULS	16 $\times$ 16 $\rightarrow$ 32	Dn $\times$ (EA) $\rightarrow$ Dn
MULU	16 $\times$ 16 $\rightarrow$ 32	Dn $\times$ (EA) $\rightarrow$ Dn
NEG	8, 16, 32	0 - (EA) $\rightarrow$ EA
NEGX	8, 16, 32	0 - (EA) - X - EA
SUB	8, 16, 32	Dn - (EA) $\rightarrow$ Dn (EA) - Dn $\rightarrow$ EA (EA) - #xxx $\rightarrow$ EA
	16, 32	An - (EA) $\rightarrow$ An
	16, 32	Dx - Dy - X $\rightarrow$ Dx Ax@ - - Ay@ - - X $\rightarrow$ Ax@
SUBX	8, 16, 32	(EA) - 0, 1 $\rightarrow$ EA [7]
TAS	8	(EA) - 0
TST	8, 16, 32	(EA) - 0
[NOTE]	[ ]	= bit number

### • LOGICAL OPERATIONS

Logical operation instructions AND, OR, EOR, and NOT are available for all sizes of integer data operands. A similar set of immediate instructions (ANDI, ORI, and EORI) provide these logical operations with all sizes of immediate data. Table 8 is a summary of the logical operations.

Table 8 Logical Operations

Instruction	Operand Size	Operation
AND	8, 16, 32	Dn $\wedge$ (EA) $\rightarrow$ Dn (EA) $\wedge$ Dn $\rightarrow$ EA (EA) $\wedge$ #xxx $\rightarrow$ EA
OR	8, 16, 32	Dn $\vee$ (EA) $\rightarrow$ Dn (EA) $\vee$ Dn $\rightarrow$ EA (EA) $\vee$ #xxx $\rightarrow$ EA
EOR	8, 16, 32	(EA) $\oplus$ Dy $\rightarrow$ EA (EA) $\oplus$ #xxx $\rightarrow$ EA
NOT	8, 16, 32	$\sim$ (EA) $\rightarrow$ EA

[NOTE]  $\sim$  = invert

### • BIT MANIPULATION OPERATIONS

Bit manipulation operations are accomplished using the following instructions: bit test (BTST), bit test and set (BSET), bit test and clear (BCLR), and bit test and change (BCHG). Table 10 is a summary of the bit manipulation operations. (Bit 2 of the status register is Z.)

Table 10 Bit Manipulation Operations

Instruction	Operand Size	Operation
BTST	8, 32	$\sim$ bit of (EA) $\rightarrow$ Z
BSET	8, 32	$\sim$ bit of (EA) $\rightarrow$ Z 1 $\rightarrow$ bit of EA
BCLR	8, 32	$\sim$ bit of (EA) $\rightarrow$ Z 0 $\rightarrow$ bit of EA
BCHG	8, 32	$\sim$ bit of (EA) $\rightarrow$ Z $\sim$ bit of (EA) $\rightarrow$ bit of EA

### • BINARY CODED DECIMAL OPERATIONS

Multiprecision arithmetic operations on binary coded decimal numbers are accomplished using the following instructions: add decimal with extend (ABCD), subtract decimal with extend (SBCD), and negate decimal with extend (NBCD). Table 11 is a summary of the binary coded decimal operations.

Table 11 Binary Coded Decimal Operations

Instruction	Operand Size	Operation
ABCD	8	Dx <sub>10</sub> + Dy <sub>10</sub> + X $\rightarrow$ Dx Ax@ <sub>-10</sub> + Ay@ <sub>-10</sub> + X $\rightarrow$ Ax@ <sub>-10</sub>
SBCD	8	Dx <sub>10</sub> - Dy <sub>10</sub> - X $\rightarrow$ Dx Ax@ <sub>-10</sub> - Ay@ <sub>-10</sub> - X $\rightarrow$ Ax@ <sub>-10</sub>
NBCD	8	0 - (EA) <sub>10</sub> - X $\rightarrow$ EA

Table 9 Shift and Rotate Operations

Instruction	Operand Size	Operation
ASL	8, 16, 32	
ASR	8, 16, 32	
LSL	8, 16, 32	
LSR	8, 16, 32	
ROL	8, 16, 32	
ROR	8, 16, 32	
ROXL	8, 16, 32	
ROXR	8, 16, 32	

### • PROGRAM CONTROL OPERATIONS

Program control operations are accomplished using a series of conditional and unconditional branch instructions and return instructions. These instructions are summarized in Table 12.

The conditional instructions provide setting and branching for the following conditions:

CC	- carry clear	LS	- low or same
CS	- carry set	LT	- less than
EQ	- equal	MI	- minus
F	- never true	NE	- not equal
GE	- greater or equal	PL	- plus
GT	- greater than	T	- always true
HI	- high	VC	- no overflow
LE	- less or equal	VS	- overflow

### • SYSTEM CONTROL OPERATIONS

System control operations are accomplished by using privileged instructions, trap generating instructions, and instructions that use or modify the status register. These instructions are summarized in Table 13.

Table 12 Program Control Operations

Instruction	Operation
<b>Conditional</b>	
BCC	Branch conditionally (14 conditions) 8- and 16-bit displacement
DBCC	Test condition, decrement, and branch 16-bit displacement
SCC	Set byte conditionally (16 conditions)
<b>Unconditional</b>	
BRA	Branch always 8- and 16-bit displacement
BSR	Branch to subroutine 8- and 16-bit displacement
JMP	Jump
JSR	Jump to subroutine
<b>Returns</b>	
RTR	Return and restore condition codes
RTS	Return from subroutine

Table 13 System Control Operations

Instruction	Operation
<b>Privileged</b>	
RESET	Reset external devices
RTE	Return from exception
STOP	Stop program execution
ORI to SR	Logical OR to status register
MOVE USP	Move user stack pointer
ANDI to SR	Logical AND to status register
EORI to SR	Logical EOR to status register
MOVE EA to SR	Load new status register
<b>Trap Generating</b>	
TRAP	Trap
TRAPV	Trap on overflow
CHK	Check register against bounds
<b>Status Register</b>	
ANDI to CCR	Logical AND to condition codes
EORI to CCR	Logical EOR to condition codes
MOVE EA to CCR	Load new condition codes
ORI to CCR	Logical OR to condition codes
MOVE SR to EA	Store status register

## SIGNAL AND BUS OPERATION DESCRIPTION

The following paragraphs contain a brief description of the input and output signals. A discussion of bus operation during the various machine cycles and operations is also given.

## SIGNAL DESCRIPTION

The input and output signals can be functionally organized into the groups shown in Figure 11. The following paragraphs provide a brief description of the signals and also a reference (if applicable) to other paragraphs that contain more detail about the function being performed.

## ADDRESS BUS ( $A_1$ THROUGH $A_{23}$ )

This 23-bit, unidirectional, three-state bus is capable of addressing 8 megawords of data. It provides the address for bus operation during all cycles except interrupt cycles. During interrupt cycles, address lines  $A_1$ ,  $A_2$ , and  $A_3$  provide information about what level interrupt is being serviced while address lines  $A_4$  through  $A_{23}$  are all set to a logic high.

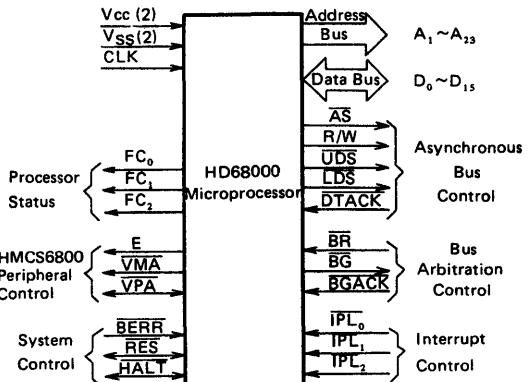


Figure 11 Input and Output Signals

## DATA BUS ( $D_0$ THROUGH $D_{15}$ )

This 16-bit, bidirectional, three-state bus is the general purpose data path. It can transfer and accept data in either word or byte length. During an interrupt acknowledge cycle, the external device supplies the vector number on data lines  $D_0$ ~ $D_7$ .

## ASYNCHRONOUS BUS CONTROL

Asynchronous data transfers are handled using the following control signals: address strobe, read/write, upper and lower data strobes, and data transfer acknowledge. These signals are explained in the following paragraphs.

## Address Strobe (AS)

This signal indicates that there is a valid address on the address bus.

## Read/Write (R/W)

This signal defines the data bus transfer as a read or write cycle. The R/W signal also works in conjunction with the upper and lower data strobes as explained in the following paragraph.

## Upper And Lower Data Strobes (UDS, LDS)

These signals control the data on the data bus, as shown in Table 14. When the R/W line is "High", the processor will read from the data bus as indicated. When the R/W line is "Low", the processor will write to the data bus as shown.

## Data Transfer Acknowledge (DTACK)

This input indicates that the data transfer is completed. When the processor recognizes DTACK during a read cycle, data is latched and the bus cycle terminated. When DTACK is recognized during a write cycle, the bus cycle is terminated.

## BUS ARBITRATION CONTROL

These three signals form a bus arbitration circuit to determine which device will be the bus master device.

## Bus Request (BR)

This input is wire ORed with all other devices that could be bus masters. This input indicates to the processor that some other device desires to become the bus master.

Table 14 Data Strobe Control of Data Bus

UDS	LDS	R/W	D <sub>8</sub> ~D <sub>15</sub>	D <sub>0</sub> ~D <sub>7</sub>
High	High	—	No valid data	No valid data
Low	Low	High	Valid data bits 8~15	Valid data bits 0~7
High	Low	High	No valid data	Valid data bits 0~7
Low	High	High	Valid data bits 8~15	No valid data
Low	Low	Low	Valid data bits 8~15	Valid data bits 0~7
High	Low	Low	Valid data bits 0~7*	Valid data bits 0~7
Low	High	Low	Valid data bits 8~15	Valid data bits 8~15*

\* These conditions are a result of current implementation and may not appear on future devices.

### Bus Grant (BG)

This output indicates to all other potential bus master devices that the processor will release bus control at the end of the current bus cycle.

### Bus Grant Acknowledge (BGACK)

This input indicates that some other device has become the bus master. This signal cannot be asserted until the following four conditions are met:

1. a bus grant has been received
2. address strobe is inactive which indicates that the microprocessor is not using the bus
3. data transfer acknowledge is inactive which indicates that either memory or the peripherals are not using the bus.
4. bus grant acknowledge is inactive which indicates that no other device is still claiming bus mastership.

### INTERRUPT CONTROL (IPL<sub>0</sub>, IPL<sub>1</sub>, IPL<sub>2</sub>)

These input pins indicate the encoded priority level of the device requesting an interrupt. Level seven is the highest priority while level zero indicates that no interrupts are requested. The least significant bit is given in IPL<sub>0</sub> and the most significant bit is contained in IPL<sub>2</sub>.

### SYSTEM CONTROL

The system control inputs are used to either reset or halt the processor and to indicate to the processor that bus errors have occurred. The three system control inputs are explained in the following paragraphs.

### Bus Error (BERR)

This input informs the processor that there is a problem with the cycle currently being executed. Problems may be a result of:

1. nonresponding devices
2. interrupt vector number acquisition failure
3. illegal access request as determined by a memory management unit
4. other application dependent errors.

The bus error signal interacts with the halt signal to determine if exception processing should be performed or the current bus cycle should be retried.

Refer to BUS ERROR AND HALT OPERATION paragraph for additional information about the interaction of the bus error and halt signals.

### Reset (RES)

This bidirectional signal line acts to reset (initiate a system initialization sequence) the processor in response to an external reset signal. An internally generated reset (result of a RESET instruction) causes all external devices to be reset and the internal state of the processor is not affected. A total system reset (processor and external devices) is the result of external halt and reset signals applied at the same time. Refer to RESET OPERATION paragraph for additional information about reset operation.

### Halt (HALT)

When this bidirectional line is driven by an external device, it will cause the processor to stop at the completion of the current bus cycle. When the processor has been halted using this input, all control signals are inactive and all three-state lines are put in their high-impedance state. Refer to BUS ERROR AND HALT OPERATION paragraph for additional information about the interaction between the halt and bus error signals.

When the processor has stopped executing instructions, such as in a double bus fault condition, the halt line is driven by the processor to indicate to external devices that the processor has stopped.

### HMCS6800 PERIPHERAL CONTROL

These control signals are used to allow the interfacing of synchronous HMCS6800 peripheral devices with the asynchronous HD68000. These signals are explained in the following paragraphs.

### Enable (E)

This signal is the standard enable signal common to all HMCS6800 type peripheral devices. The period for this output is ten HD68000 clock periods (six clocks "Low", four clocks "High").

### Valid Peripheral Address (VPA)

This input indicates that the device or region addressed is a HMCS6800 family device and that data transfer should be synchronized with the enable (E) signal. This input also indicates that the processor should use automatic vectoring for an interrupt. Refer to INTERFACE WITH HMCS6800 PERIPHERALS.

### Valid Memory Address (VMA)

This output is used to indicate to HMCS6800 peripheral devices that there is a valid address on the address bus and the processor is synchronized to enable. This signal only responds to a valid peripheral address (VPA) input which indicates that the peripheral is a HMCS6800 family device.

### PROCESSOR STATUS (FC<sub>0</sub>, FC<sub>1</sub>, FC<sub>2</sub>)

These function code outputs indicate the state (user or supervisor) and the cycle type currently being executed, as shown in Table 15. The information indicated by the function code outputs is valid whenever address strobe (AS) is active.

### CLOCK (CLK)

The clock input is a TTL compatible signal that is internally buffered for development of the internal clocks needed by the processor. The clock input shall be a constant frequency.

Table 15 Function Code Outputs

$FC_2$	$FC_1$	$FC_0$	Cycle Type
Low	Low	Low	(Undefined, Reserved)
Low	Low	High	User Data
Low	High	Low	User Program
Low	High	High	(Undefined, Reserved)
High	Low	Low	(Undefined, Reserved)
High	Low	High	Supervisor Data
High	High	Low	Supervisor Program
High	High	High	Interrupt Acknowledge

**SIGNAL SUMMARY**

Table 16 is a summary of all the signals discussed in the previous paragraphs.

**• BUS OPERATION**

The following paragraphs explain control signal and bus operation during data transfer operations, but arbitration, bus error and halt conditions, and reset operation.

**DATA TRANSFER OPERATIONS**

Transfer of data between devices involves the following leads:

Address Bus  $A_1$  through  $A_{23}$

Data Bus  $D_0$  through  $D_{15}$

Control Signals

The address and data buses are separate parallel buses used to transfer data using an asynchronous bus structure. In all cycles, the bus master assumes responsibility for deskewing all signals it issues at both the start and end of a cycle. In addition, the bus master is responsible for deskewing the acknowledge and data signals from the slave device.

The following paragraphs explain the read, write, and read-modify-write cycles. The indivisible read-modify-write cycle is the method used by the HD68000 for interlocked multiprocessor communications.

[NOTE] The terms assertion and negation will be used extensively. This is done to avoid confusion when dealing with a mixture of "active-low" and "active-high" signals. The term assert or assertion is used to indicate that a signal is active or true independent of whether that voltage is "Low" or "High". The term negate or negation is used to indicate that a signal is inactive or false.

**Read Cycle**

During a read cycle, the processor receives data from memory or a peripheral device. The processor reads bytes of data in all cases. If the instruction specifies a word (or double word) operation, the processor reads both bytes. When the instruction specifies byte operation, the processor uses an internal A0 bit to determine which byte to read and then issues the data strobe required for that byte. For byte operations, when the A0 bit equals zero, the upper data strobe is issued. When the A0 bit equals one, the lower data strobe is issued. When the data is received, the processor correctly positions it

Table 16 Signal Summary

Signal Name	Mnemonic	Input/Output	Active State	Three State
Address Bus	$A_1 \sim A_{23}$	Output	High	Yes
Data Bus	$D_0 \sim D_{15}$	Input/Output	High	Yes
Address Strobe	$\overline{AS}$	Output	Low	Yes
Read/Write	R/W	Output	Read-High Write-Low	Yes
Upper and Lower Data Strobes	UDS, LDS	Output	Low	Yes
Data Transfer Acknowledge	DTACK	Input	Low	No
Bus Request	$\overline{BR}$	Input	Low	No
Bus Grant	$\overline{BG}$	Output	Low	No
Bus Grant Acknowledge	BGACK	Input	Low	No
Interrupt Priority Level	$IPL_0, IPL_1, IPL_2$	Input	Low	No
Bus Error	BERR	Input	Low	No
Reset	RES	Input/Output	Low	No*
Halt	HALT	Input/Output	Low	No*
Enable	E	Output	High	No
Valid Memory Address	VMA	Output	Low	Yes
Valid Peripheral Address	VPA	Input	Low	No
Function Code Output	$FC_0, FC_1, FC_2$	Output	High	Yes
Clock	CLK	Input	High	No
Power Input	$V_{cc}$	Input	—	—
Ground	$V_{ss}$	Input	—	—

\* Open drain

internally.

A word read cycle flow chart is given in Figure 12. A byte read cycle flow chart is given in Figure 13. Read cycle timing is given in Figure 14 and Figure 15 details word and byte read cycle operation.

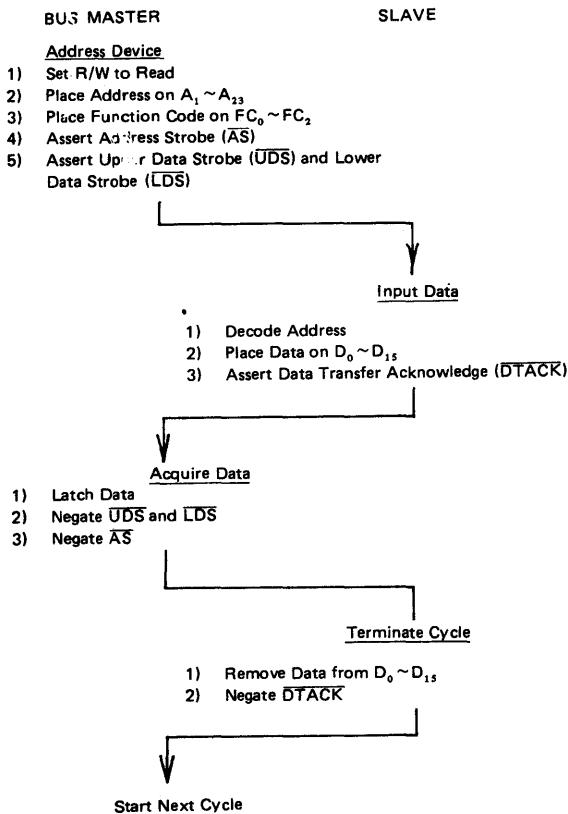


Figure 12 Word Read Cycle Flow Chart

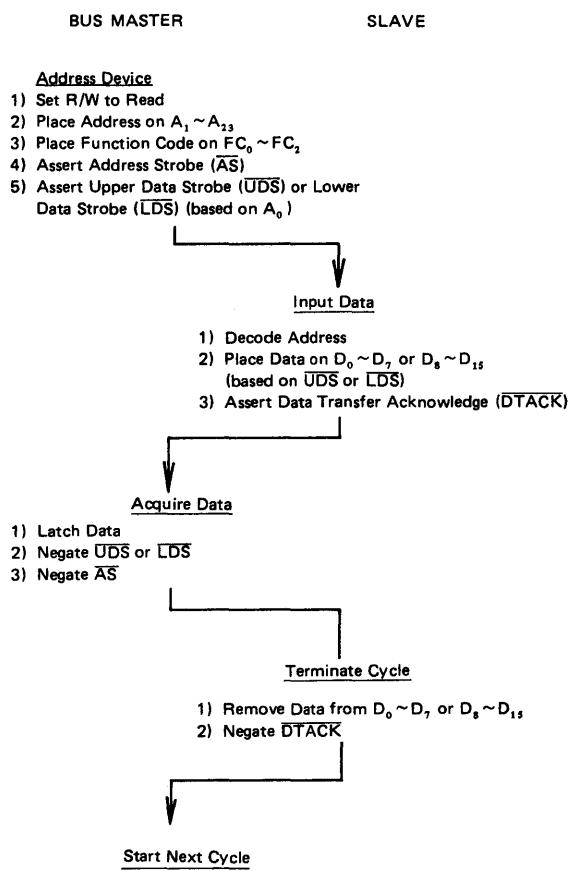


Figure 13 Byte Read Cycle Flow Chart

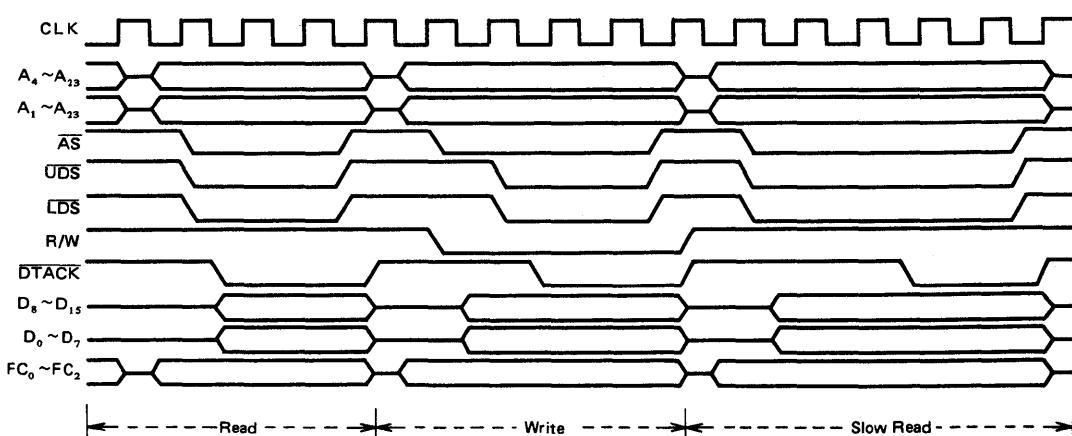


Figure 14 Read and Write Cycle Timing Diagram

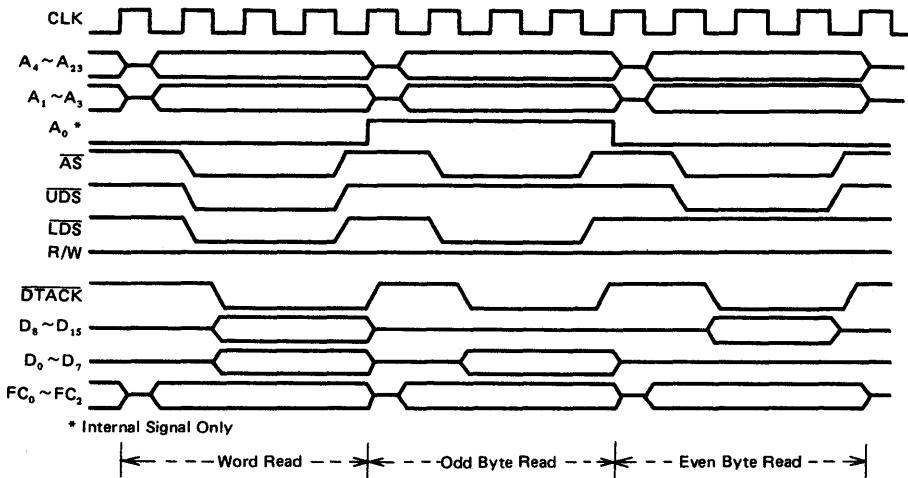


Figure 15 Word and Byte Read Cycle Timing Diagram

### Write Cycle

During a write cycle, the processor sends data to memory or a peripheral device. The processor writes bytes of data in all cases. If the instruction specifies a word operation, the processor writes both bytes. When the instruction specifies a byte operation, the processor uses an internal A0 bit to determine which byte to write and then issues the data strobe required for that byte. For byte operations, when the A0 bit equals zero, the upper data strobe is issued. When the A0 bit equals one, the lower data strobe is issued. A word write cycle flow chart is given in Figure 16. A byte write cycle flow chart is given in Figure 17. Write cycle timing is given in Figure 14 and Figure 18 details word and byte write cycle operation.

### Read-Modify-Write Cycle

The read-modify-write cycle performs a read, modifies the data in the arithmetic-logic unit, and writes the data back to the same address. In the HD68000 this cycle is indivisible in that the address strobe is asserted throughout the entire cycle. The test and set (TAS) instruction uses this cycle to provide meaningful communication between processors in a multiple processor environment. This instruction is the only instruction that uses the read-modify-write cycles and since the test and set instruction only operates on bytes, all read-modify-write cycles are byte operations. A read-modify-write cycle flow chart is given in Figure 19 and a timing diagram is given in Figure 20.

### BUS ARBITRATION

Bus arbitration is a technique used by master-type devices to request, be granted, and acknowledge bus mastership. In its simplest form, it consists of:

1. Asserting a bus mastership request.
2. Receiving a grant that the bus is available at the end of the current cycle.
3. Acknowledging that mastership has been assumed.

Figure 21 is a flow chart showing the detail involved in a request from a single device. Figure 22 is a timing diagram for the same operations. This technique allows processing of bus requests during data transfer cycles.

### BUS MASTER SLAVE

#### Address Device

- 1) Place Address on  $A_1 \sim A_{23}$
- 2) Place Function Code on  $FC_0 \sim FC_2$
- 3) Assert Address Strobe (AS)
- 4) Set R/W to Write
- 5) Place Data on  $D_0 \sim D_{15}$
- 6) Assert Upper Data Strobe (UDS) and Lower Data Strobe (LDS)

- Input Data
- 1) Decode Address
  - 2) Store Data on  $D_0 \sim D_{15}$
  - 3) Assert Data Transfer Acknowledge (DTACK)

#### Terminate Output Transfer

- 1) Negate UDS and LDS
- 2) Negate AS
- 3) Remove Data from  $D_0 \sim D_{15}$
- 4) Set R/W to Read

#### Terminate Cycle

- 1) Negate DTACK

#### Start Next Cycle

Figure 16 Word Write Cycle Flow Chart

BUS MASTERSLAVEAddress Device

- 1) Place Address on  $A_1 \sim A_{23}$
- 2) Place Function Code on  $FC_0 \sim FC_2$
- 3) Assert Address Strobe ( $\bar{AS}$ )
- 4) Set R/W to Write
- 5) Place Data on  $D_0 \sim D_7$ , or  $D_8 \sim D_{15}$  (according to  $A_0$ )
- 6) Assert Upper Data Strobe ( $\bar{UDS}$ ) or Lower Data Strobe ( $\bar{LDS}$ ) (based on  $A_0$ )

Input Data

- 1) Decode Address
- 2) Store Data on  $D_0 \sim D_7$ , if  $\bar{LDS}$  is asserted.  
Store Data on  $D_8 \sim D_{15}$  if  $\bar{UDS}$  is asserted.
- 3) Assert Data Transfer Acknowledge (DTACK)

Terminate Output Transfer

- 1) Negate  $\bar{UDS}$  and  $\bar{LDS}$
- 2) Negate  $\bar{AS}$
- 3) Remove Data from  $D_0 \sim D_7$ , or  $D_8 \sim D_{15}$
- 4) Set R/W to Read

Terminate Cycle

- 1) Negate DTACK

Start Next Cycle

Figure 17 Byte Write Cycle Flow Chart

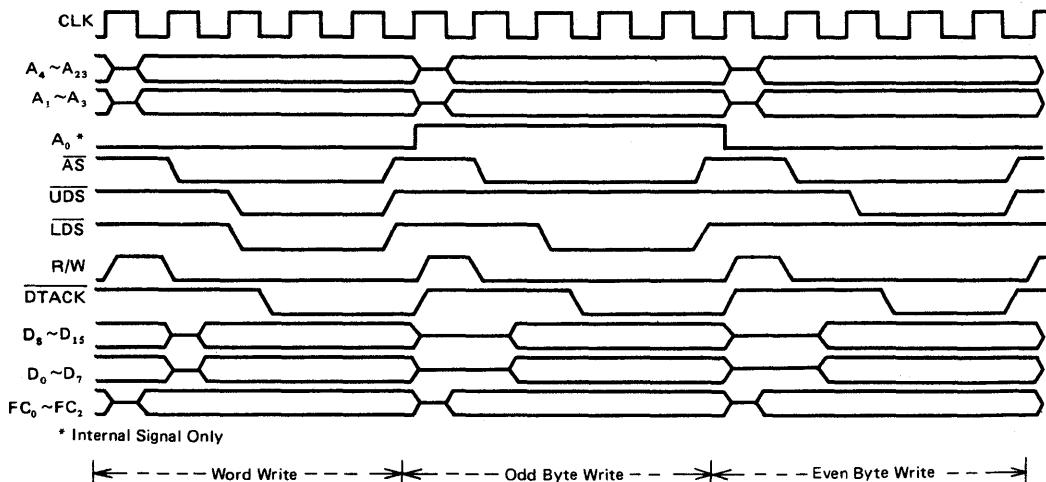


Figure 18 Word and Byte Write Cycle Timing Diagram

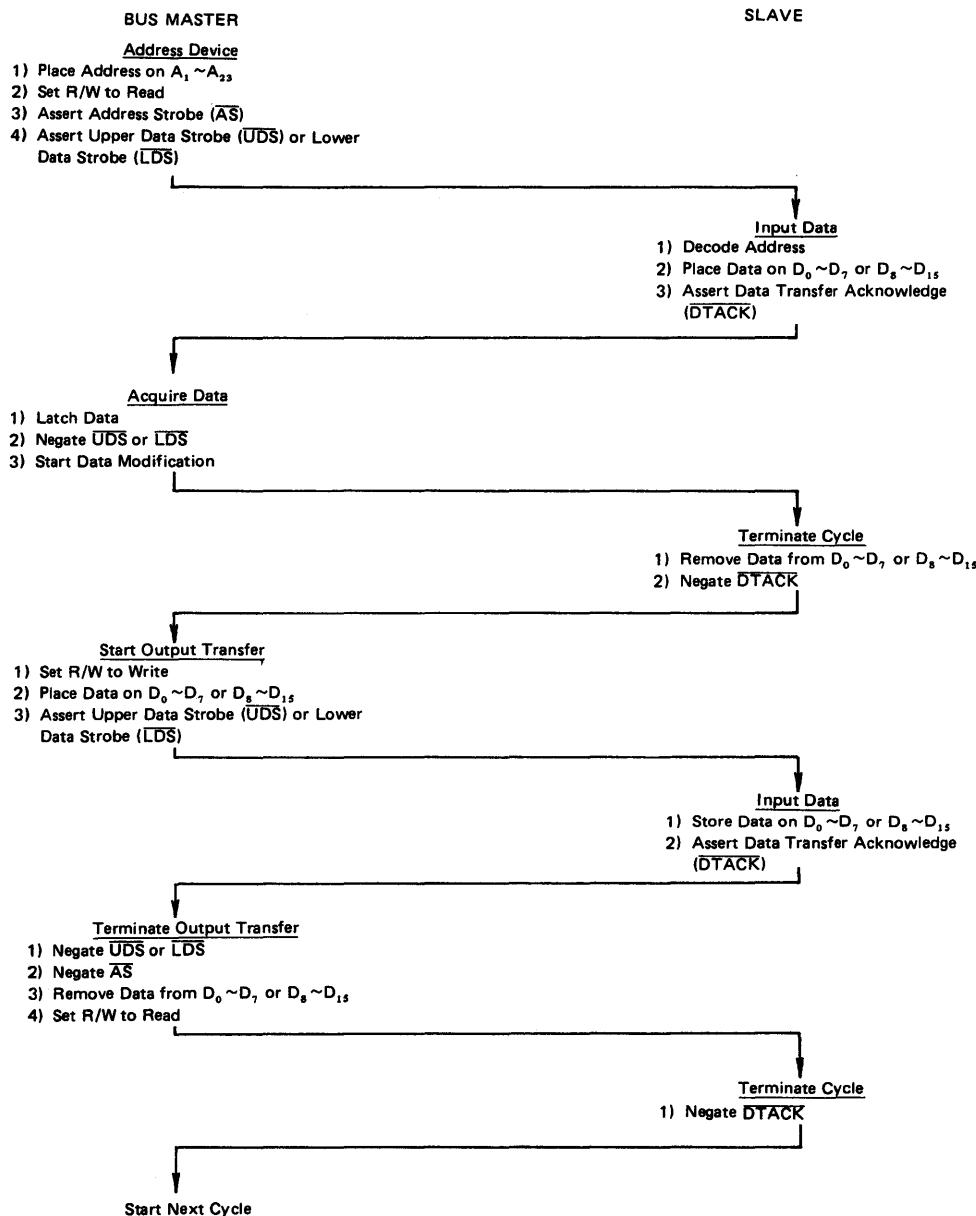


Figure 19 Read-Modify-Write Cycle Flow Chart

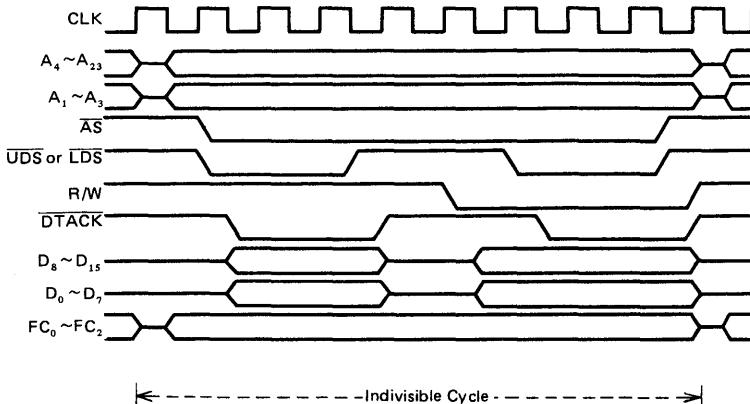
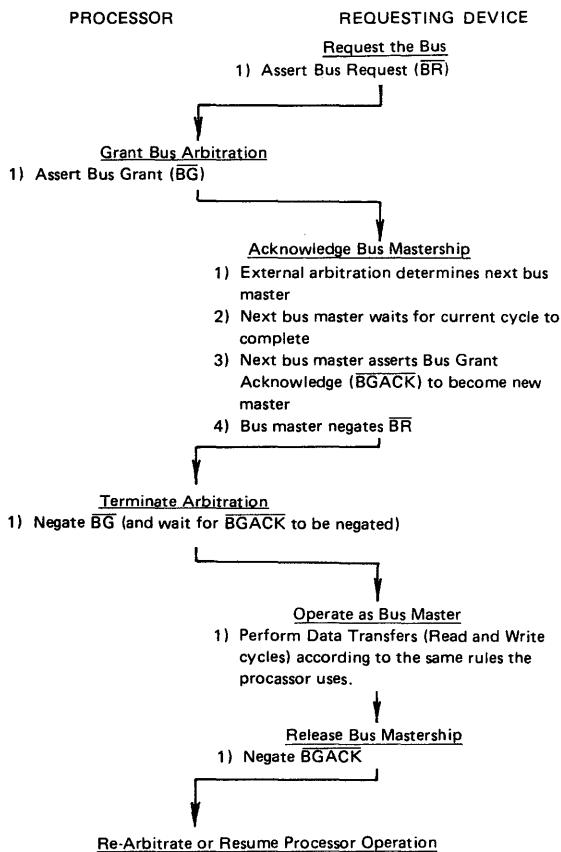


Figure 20 Read-Modify-Write Cycle Timing Diagram



The timing diagram shows that the bus request is negated at the time that an acknowledge is asserted. This type of operation would be true for a system consisting of the processor and one device capable of bus mastership. In systems having a number of devices capable of bus mastership, the bus request line from each device is wire ORed to the processor. In this system, it is easy to see that there could be more than one bus request being made. The timing diagram shows that the bus grant signal is negated a few clock cycles after the transition of the acknowledge (BGACK) signal.

However, if the bus requests are still pending, the processor will assert another bus grant within a few clock cycles after it was negated. This additional assertion of bus grant allows external arbitration circuitry to select the next bus master before the current bus master has completed its requirements. The following paragraphs provide additional information about the three steps in the arbitration process.

## Requesting the Bus

External devices capable of becoming bus masters request the bus by asserting the bus request (**BR**) signal. This is a wire ORed signal (although it need not be constructed from open collector devices) that indicates to the processor that some external device requires control of the external bus. The processor is effectively at a lower bus priority level than the external device and will relinquish the bus after it has completed the last bus cycle it has started.

When no acknowledge is received before the bus request signal goes inactive, the processor will continue processing when it detects that the bus request is inactive. This allows ordinary processing to continue if the arbitration circuitry responded to noise inadvertently.

### **Receiving the Bus Grant**

The processor asserts bus grant (**BG**) as soon as possible. Normally this is immediately after internal synchronization. The only exception to this occurs when the processor has made an internal decision to execute the next bus cycle but has not progressed far enough into the cycle to have asserted the address strobe (**AS**) signal. In this case, bus grant will not be asserted until one clock after address strobe is asserted to indicate to

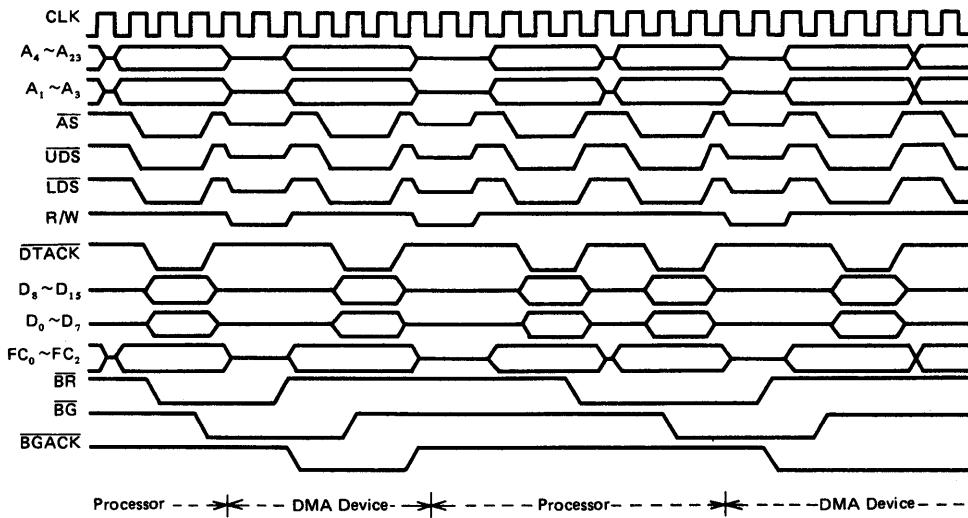


Figure 22 Bus Arbitration Cycle Timing Diagram

external devices that a bus cycle is being executed.

The bus grant signal may be routed through a daisy-chained network or through a specific priority-encoded network. The processor is not affected by the external method of arbitration as long as the protocol is obeyed.

#### Acknowledgement of Mastership

Upon receiving a bus grant, the requesting device waits until address strobe, data transfer acknowledge, and bus grant acknowledge are negated before issuing its own BGACK. The negation of the address strobe indicates that the previous master has completed its cycle, the negation of bus grant acknowledgement indicates that the previous master has released the bus. (While address strobe is asserted no device is allowed to "break into" a cycle.) The negation of data transfer acknowledge indicates the previous slave has terminated its connection to the previous master. Note that in some applications data transfer acknowledgement might not enter into this function. General purpose devices would then be connected such that they were only dependent on address strobe. When bus grant acknowledgement is issued the device is bus master until it negates bus grant acknowledgement. Bus grant acknowledgement should not be negated until after the bus cycle(s) is (are) completed. Bus mastership is terminated at the negation of bus grant acknowledgement.

The bus request from the granted device should be dropped when bus grant acknowledgement is asserted. If bus request is still asserted after bus grant acknowledgement is negated, the processor performs another arbitration sequence and issues another bus grant. Note that the processor does not perform any external bus cycles before it re-asserts bus grant.

#### BUS ERROR AND HALT OPERATION

In a bus architecture that requires a handshake from an external device, the possibility exists that the handshake might not occur. Since different systems will require a different maximum response time, a bus error input is provided. External circuitry must be used to determine the duration between address strobe and data transfer acknowledge before issuing a

bus error signal. When a bus error signal is received, the processor has two options: initiate a bus error exception sequence or try running the bus cycle again.

#### Exception Sequence

The bus error exception sequence is entered when the processor receives a bus error signal and the halt pin is inactive. Figure 23 is a timing diagram for the exception sequence. The sequence is composed of the following elements:

1. Stacking the program counter and status register
2. Stacking the error information
3. Reading the bus error vector table entry
4. Executing the bus error handler routine

The stacking of the program counter and the status register is the same as if an interrupt had occurred. Several additional items are stacked when a bus error occurs. These items are used to determine the nature of the error and correct it, if possible. The bus error vector is vector number two located at address \$000008. The processor loads the new program counter from this location. A software bus error handler routine is then executed by the processor. Refer to EXCEPTION PROCESSING for additional information.

#### Re-Running the Bus Cycle

When the processor receives a bus error signal and the halt pin is being driven by an external device, the processor enters the re-run sequence. Figure 24 is a timing diagram for re-running the bus cycle.

The processor completes the bus cycle, then puts the address, data and function code output lines in the high-impedance state. The processor remains "halted," and will not run another bus cycle until the halt signal is removed by external logic. Then the processor will re-run the previous bus cycle using the same address, the same function codes, the same data (for a write operation), and the same controls. The bus error signal should be removed before the halt signal is removed.

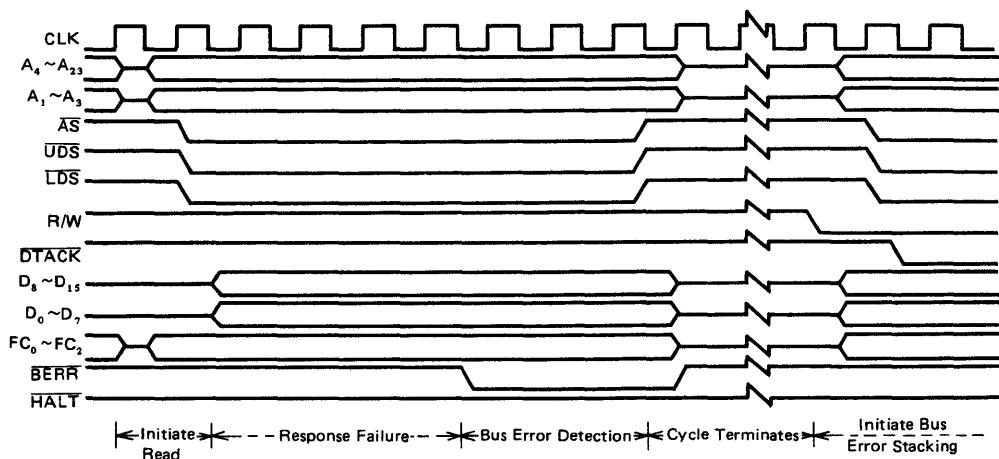


Figure 23 Bus Error Timing Diagram

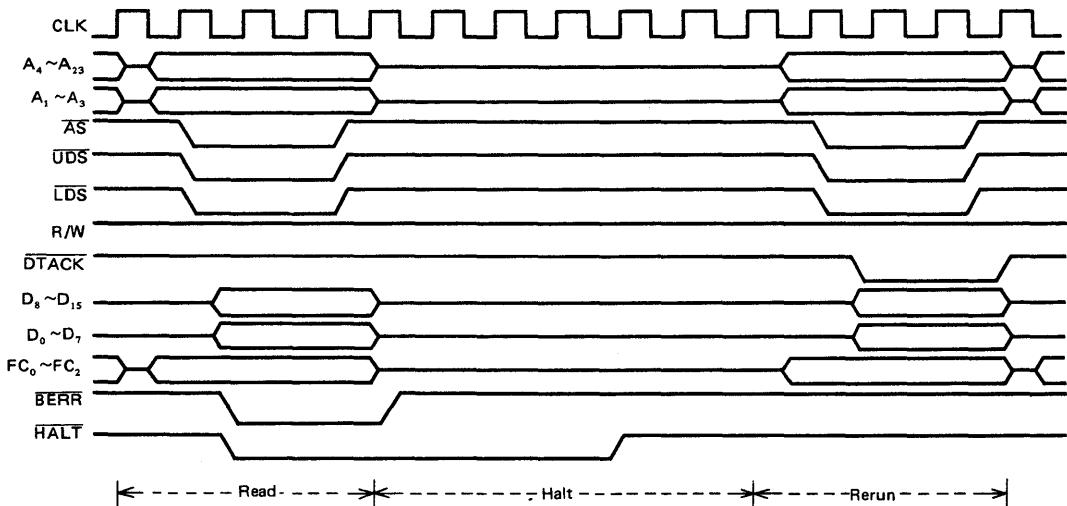


Figure 24 Re-Run Bus Cycle Timing Information

[NOTE] The processor will not re-run a read-modify-write cycle. This restriction is made to guarantee that the entire cycle runs correctly and that the write operation of a Test-and-Set operation is performed without ever releasing AS.

#### Halt Operation with No Bus Error

The halt input signal to the HD68000 performs a Halt/Run/Single-Step function in a similar fashion to the HMCS6800 halt function. The halt and run modes are somewhat self explanatory in that when the halt signal is constantly active the processor "halts" (does nothing) and when the halt signal is constantly inactive the processor "runs" (does something).

The single-step mode is derived from correctly timed transitions on the halt signal input. It forces the processor to execute a single bus cycle by entering the "run" mode until the processor starts a bus cycle then changing to the "halt" mode. Thus, the single-step mode allows the user to proceed through (and therefore debug) processor operations one bus cycle at a time.

Figure 25 details the timing required for correct single-step operations. Some care must be exercised to avoid harmful interactions between the bus error signal and the halt pin when using the single cycle mode as a debugging tool. This is also true of interactions between the halt and reset lines since these can reset the machine.

When the processor completes a bus cycle after recognizing that the halt signal is active, most three-state signals are put in the high-impedance state. These include:

1. address lines
2. data lines
3. function code lines

This is required for correct performance of the re-run bus cycle operation.

Note that when the processor honors a request to halt, the function codes are put in the high-impedance state (their buffer characteristics are the same as the address buffers). While the

processor is honoring the halt request, bus arbitration performs as usual. That is, halting has no effect on bus arbitration. It is the bus arbitration function that removes the control signals from the bus.

The halt function and the hardware trace capability allow the hardware debugger to trace single bus cycles or single instructions at a time. These processor capabilities, along with a software debugging package, give total debugging flexibility.

#### Double Bus Faults

When a bus error exception occurs, the processor will attempt to stack several words containing information about the state of the machine. If a bus error exception occurs during the stacking operation, there have been two bus errors in a row. This is commonly referred to as a double bus fault. When a double bus fault occurs, the processor will halt. Once a bus error exception has occurred, any bus error exception occurring before the execution of the next instruction constitutes a double bus fault.

Note that a bus cycle which is re-run does not constitute a bus error exception, and does not contribute to a double bus fault. Note also that this means that as long as the external hardware requests it, the processor will continue to re-run the same bus cycle.

The bus error pin also has an effect on processor operation after the processor receives an external reset input. The processor reads the vector table after a reset to determine the address to start program execution. If a bus error occurs while reading the vector table (or at any time before the first instruction is executed), the processor reacts as if a double bus fault has occurred and it halts. Only an external reset will start a halted processor.

#### RESET OPERATION

The reset signal is a bidirectional signal that allows either the processor or an external signal to reset the system. Figure 26 is a timing diagram for reset operations. Both the halt and the reset

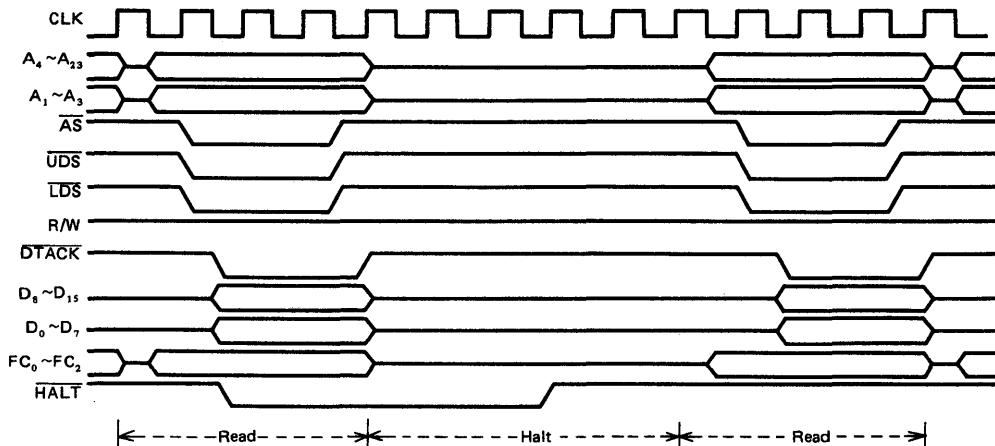


Figure 25 Halt Signal Timing Characteristics

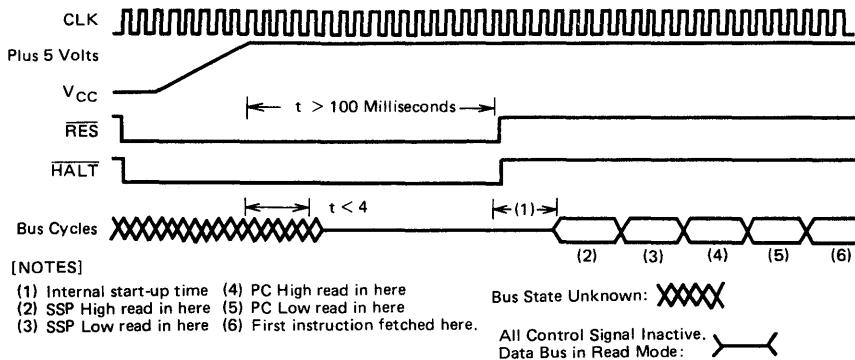


Figure 26 Reset Operation Timing Diagram

lines must be applied to ensure total reset of the processor.

When the reset and halt lines are driven by an external device, it is recognized as an entire system reset, including the processor. The processor responds by reading the reset vector table entry (vector number zero, address \$000000) and loads it into the supervisor stack pointer (SSP). Vector table entry number one at address \$000004 is read next and loaded into the program counter. The processor initializes the status register to an interrupt level of seven. No other registers are affected by the reset sequence.

When a RESET sequence is executed, the processor drives the reset pin for 124 clock pulses. In this case, the processor is trying to reset the rest of the system. Therefore, there is no effect on the internal state of the processor. All of the processor's internal registers and the status register are unaffected by the execution of a RESET instruction. All external devices connected to the reset line should be reset at the completion of the RESET instruction.

When V<sub>CC</sub> is initially applied to the processor, an external reset must be applied to the reset pin for 100 milliseconds.

#### ■ EXCEPTION PROCESSING

The following paragraphs describe the actions of the HD68000 which are outside the normal processing associated with the execution of instructions. The functions of the bits in the supervisor portion of the status register are covered: the supervisor/user bit, the trace enable bit, and the processor interrupt priority mask. Finally, the sequence of memory references and actions taken by the processor on exception conditions is detailed.

#### ■ PROCESSING STATES

The HD68000 is always one of three processing states: normal, exception, or halted. The normal processing states is that associated with instruction execution; the memory references are to fetch instructions and operands, and to store results. A special case of the normal state is the stopped state which the processor enters when a STOP instruction is executed. In this state, no further memory references are made.

The exception processing state is associated with interrupts, trap instructions, tracing and other exceptional conditions. The exception may be internally generated by an instruction or by an usual condition arising during the execution of an instruction. Externally, exception processing can be forced by an

interrupt, by a bus error, or by a reset. Exception processing is designed to provide an efficient context switch so that the processor may handle unusual conditions.

The halted processing state is an indication of catastrophic hardware failure. For example, if during the exception processing of a bus error another bus error occurs, the processor assumes that the system is unusable and halts. Only an external reset can restart a halted processor. Note that a processor in the stopped state is not in the halted state, nor vice versa.

#### ● PRIVILEGE STATES

The processor operates in one of two states of privilege: the "user" state or the "supervisor" state. The privilege state determines which operations are legal, is used by the external memory management device to control and translate accesses, and is used to choose between the supervisor stack pointer and the user stack pointer in instruction references.

The privilege state is a mechanism for providing security in a computer system. Programs should access only their own code and data areas, and ought to be restricted from accessing information which they do not need and must not modify.

The privilege mechanism provides security by allowing most programs to execute in user state. In this state, the accesses are controlled, and the effects on other parts of the system are limited. The operating system executes in the supervisor state, has access to all resources, and performs the overhead tasks for the user state programs.

#### SUPERVISOR STATE

The supervisor state is the higher state of privilege. For instruction execution, the supervisor state is determined by the S-bit of the status register; if the S-bit is asserted (high), the processor is in the supervisor state. All instructions can be executed in the supervisor state. The bus cycles generated by instructions executed in the supervisor state are classified as supervisor references. While the processor is in the supervisor privilege state, those instructions which use either the system stack pointer implicitly or address register seven explicitly access the supervisor stack pointer.

All exception processing is done in the supervisor state, regardless of the setting of the S-bit. The bus cycles generated during exception processing are classified as supervisor references. All stacking operations during exception processing use the supervisor stack pointer.

## USER STATE

The user state is the lower state of privilege. For instruction execution, the user state is determined by the S-bit of the status register; if the S-bit is negated (low), the processor is executing instructions in the user state.

Most instructions execute the same in user state as in the supervisor state. However, some instructions which have important system effects are made privileged. User programs are not permitted to execute the STOP instruction, or the RESET instruction. To ensure that a user program cannot enter the supervisor state except in a controlled manner, the instructions which modify the whole status register are privileged. To aid in debugging programs which are to be used as operating systems, the move to user stack pointer (MOVE USP) and move from user stack pointer (MOVE from USP) instructions are also privileged.

The bus cycles generated by an instruction executed in user state are classified as user state references. This allows an external memory management device to translate the address and to control access to protected portions of the address space. While the processor is in the user privilege state, those instructions which use either the system stack pointer implicitly, or address register seven explicitly, access the user stack pointer.

## PRIVILEGE STATE CHANGES

Once the processor is in the user state and executing instructions, only exception processing can change the privilege state. During exception processing, the current setting of the S-bit of the status register is saved and the S-bit is asserted, putting the processing in the supervisor state. Therefore, when instruction execution resumes at the address specified to process the exception, the processor is in the supervisor privilege state.

## REFERENCE CLASSIFICATION

When the processor makes a reference, it classifies the kind of reference being made, using the encoding on the three function code output lines. This allows external translation of addresses, control of access, and differentiation of special processor states, such as interrupt acknowledge. Table 17 lists the classification of references.

### EXCEPTION PROCESSING

Before discussing the details of interrupts, traps, and tracing, a general description of exception processing is in order. The

Table 17 Reference Classification

Function Code Output			Reference Class
FC <sub>2</sub>	FC <sub>1</sub>	FC <sub>0</sub>	
0	0	0	(Unassigned)
0	0	1	User Data
0	1	0	User Program
0	1	1	(Unassigned)
1	0	0	(Unassigned)
1	0	1	Supervisor Data
1	1	0	Supervisor Program
1	1	1	Interrupt Acknowledge

processing of an exception occurs in four steps, with variations for different exception causes. During the first step, a temporary copy of the status register is made, and the status register is set for exception processing. In the second step the exception vector is determined, and the third step is the saving of the current processor context. In the fourth step a new context is obtained, and the processor switches to instruction processing.

## EXCEPTION VECTORS

Exception vectors are memory locations from which the processor fetches the address of a routine which will handle that exception. All exception vectors are two words in length (Figure 27), except for the reset vector, which is four words. All exception vectors lie in the supervisor data space, except for the reset vector which is in the supervisor program space. A vector number is an eight-bit number which, when multiplied by four, gives the address of an exception vector. Vector numbers are generated internally or externally, depending on the cause of the exception. In the case of interrupts, during the interrupt acknowledge bus cycle, a peripheral provides an 8-bit vector number (Figure 28) to the processor on data bus lines D<sub>0</sub> through D<sub>7</sub>. The processor translates the vector number into a full 24-bit address, as shown in Figure 29. The memory layout for exception vectors is given in Table 18.

As shown in Table 18, the memory layout is 512 words long (1024 bytes). It starts at address 0 and proceeds through address 1023. This provides 255 unique vectors; some of these are

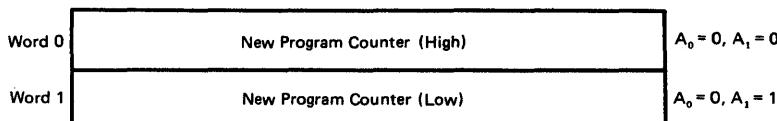
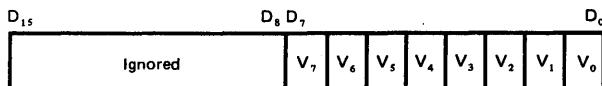


Figure 27 Exception Vector Format



Where: V<sub>7</sub> is the MSB of the Vector Number  
V<sub>0</sub> is the LSB of the Vector Number

Figure 28 Peripheral Vector Number Format

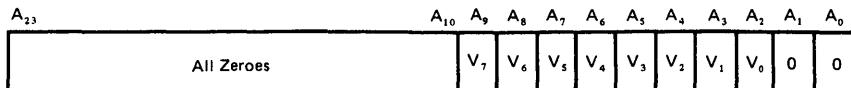


Figure 29 Address Translated from 8-Bit Vector Number

Table 18 Exception Vector Assignment

Vector Number(s)	Address			Assignment
	Dec	Hex	Space	
0	0	000	SP	Reset: Initial SSP
—	4	004	SP	Reset: Initial PC
2	8	008	SD	Bus Error
3	12	00C	SD	Address Error
4	16	010	SD	Illegal Instruction
5	20	014	SD	Zero Divide
6	24	018	SD	CHK Instruction
7	28	01C	SD	TRAPV Instruction
8	32	020	SD	Privilege Violation
9	36	024	SD	Trace
10	40	028	SD	Line 1010 Emulator
11	44	02C	SD	Line 1111 Emulator
12*	48	030	SD	(Unassigned, reserved)
13*	52	034	SD	(Unassigned, reserved)
14*	56	038	SD	(Unassigned, reserved)
15	60	03C	SD	Uninitialized Interrupt Vector
16~23*	64	04C	SD	(Unassigned, reserved)
	95	05F		—
24	96	060	SD	Spurious Interrupt
25	100	064	SD	Level 1 Interrupt Autovector
26	104	068	SD	Level 2 Interrupt Autovector
27	108	06C	SD	Level 3 Interrupt Autovector
28	112	070	SD	Level 4 Interrupt Autovector
29	116	074	SD	Level 5 Interrupt Autovector
30	120	078	SD	Level 6 Interrupt Autovector
31	124	07C	SD	Level 7 Interrupt Autovector
32~47	128	080	SD	TRAP Instruction Vectors
	191	08F		—
48~63*	192	0C0	SD	(Unassigned, reserved)
	255	OFF		—
64~255	256	100	SD	User Interrupt Vectors
	1023	3FF		—

\* Vector numbers 12, 13, 14, 16 through 23 and 48 through 63 are reserved for future enhancements by Hitachi. No user peripheral devices should be assigned these numbers.

reserved for TRAPS and other system functions. Of the 255, there are 192 reserved for user interrupt vectors. However, there is no protection on the first 64 entries, so user interrupt vectors may overlap at the discretion of the systems designer.

### KINDS OF EXCEPTIONS

Exceptions can be generated by either internal or external causes. The externally generated exceptions are the interrupts and the bus error and reset requests. The interrupts are requests from peripheral devices for processor action while the bus error and reset inputs are used for access control and processor restart. The internally generated exceptions come from instructions, or from address errors or tracing. The trap (TRAP), trap on overflow (TRAPV), check register against bounds (CHK) and divide (DIV) instructions all can generate exceptions as part of their instruction execution. In addition, illegal instructions, word fetches from odd addresses and privilege violations cause exceptions. Tracing behaves like a very high priority, internally generated interrupt after each instruction execution.

### EXCEPTION PROCESSING SEQUENCE

Exception processing occurs in four identifiable steps. In the first step, an internal copy is made of the status register. After the copy is made, the S-bit is asserted, putting the processor into the supervisor privilege state. Also, the T-bit is negated which will allow the exception handler to execute unhindered by tracing. For the reset and interrupt exceptions, the interrupt priority mask is also updated.

In the second step, the vector number of the exception is determined. For interrupts, the vector number is obtained by a processor fetch, classified as an interrupt acknowledge. For all other exceptions, internal logic provides the vector number. This vector number is then used to generate the address of the exception vector.

The third step is to save the current processor status, except for the reset exception. The current program counter value and the saved copy of the status register are stacked using the supervisor stack pointer. The program counter value stacked usually points to the next unexecuted instruction, however for bus error and address error, the value stacked for the program counter is unpredictable, and may be incremented from the address of the instruction which caused the error. Additional information defining the current context is stacked for the bus error and address error exceptions.

The last step is the same for all exceptions. The new program counter value is fetched from the exception vector. The processor then resumes instruction execution. The instruction at the address given in the exception vector is fetched, and normal instruction decoding and execution is started.

### MULTIPLE EXCEPTIONS

These paragraphs describe the processing which occurs when multiple exceptions arise simultaneously. Exceptions can be grouped according to their occurrence and priority. The Group 0 exceptions are reset, bus error, and address error. These exceptions cause the instruction currently being executed to be aborted, and the exception processing to commence at the next minor cycle of the processor. The Group 1 exceptions are trace and interrupt, as well as the privilege violations and illegal instructions. These exceptions allow the current instruction to execute to completion, but preempt the execution of the next instruction by forcing exception processing to occur (privilege violations and illegal instructions are detected when they are the next instruction to be executed). The Group 2 exceptions occur

as part of the normal processing of instructions. The TRAP, TRAPV, CHK, and zero divide exceptions are in this group. For these exceptions, the normal execution of an instruction may lead to exception processing.

Group 0 exceptions have highest priority, while Group 2 exceptions have lowest priority. Within Group 0, reset has highest priority, followed by bus error and then address error. Within Group 1, trace has priority over external interrupts, which in turn takes priority over illegal instruction and privilege violation. Since only one instruction can be executed at a time, there is no priority relation within Group 2.

The priority relation between two exceptions determines which is taken, or taken first, if the conditions for both arise simultaneously. Therefore, if a bus error occurs during a TRAP instruction, the bus error takes precedence, and the TRAP instruction processing is aborted. In another example, if an interrupt request occurs during the execution of an instruction while the T-bit is asserted, the trace exception has priority, and is processed first. Before instruction processing resumes, however, the interrupt exception is also processed, and instruction processing commences finally in the interrupt handler routine. A summary of exception grouping and priority is given in Table 19.

Table 19 Exception Grouping and Priority

Group	Exception	Processing
0	Reset	Exception processing begins at the next minor cycle
	Bus Error	
	Address Error	
1	Trace	Exception processing begins before the next instruction
	Interrupt	
	Illegal	
	Privilege	
2	TRAP, TRAPV,	Exception processing is started by normal instruction execution
	CHK,	
	Zero Divide	

### ● EXCEPTION PROCESSING DETAILED DISCUSSION

Exceptions have a number of sources, and each exception has processing which is peculiar to it. The following paragraphs detail the sources of exceptions, how each arises, and how each is processed.

### RESET

The reset input provides the highest exception level. The processing of the reset signal is designed for system initiation, and recovery from catastrophic failure. Any processing in progress at the time of the reset is aborted and cannot be recovered. The processor is forced into the supervisor state, and the trace state is forced off. The processor interrupt priority mask is set at level seven. The vector number is internally generated to reference the reset exception vector at location 0 in the supervisor program space. Because no assumptions can be made about the validity of register contents, in particular the supervisor stack pointer, neither the program counter nor the status register is saved. The address contained in the first two words of the reset exception vector is fetched as the initial supervisor stack pointer, and the address in the last two words of the reset exception vector is fetched as the initial program

counter. Finally, instruction execution is started at the address in the program counter. The power-up/restart code should be pointed to by the initial program counter.

The RESET instruction does not cause loading of the reset vector, but does assert the reset line to reset external devices. This allows the software to reset the system to a known state and then continue processing with the next instruction.

## INTERRUPTS

Seven levels of interrupt priorities are provided. Devices may be chained externally within interrupt priority levels, allowing an unlimited number of peripheral devices to interrupt the processor. Interrupt priority levels are numbered from one to seven, level seven being the highest priority. The status register contains a three bit mask which indicates the current processor priority, and interrupts are inhibited for all priority levels less than or equal to the current processor priority.

An interrupt request is made to the processor by encoding the interrupt request level on the interrupt request lines; a zero indicates no interrupt request. Interrupt requests arriving at the processor do not force immediate exception processing, but are made pending. Pending interrupts are detected between instruction executions. If the priority of the pending interrupt is lower than or equal to the current processor priority, execution continues with the next instruction and the interrupt exception processing is postponed. (The recognition of level seven is slightly different, as explained in a following paragraph.)

If the priority of the pending interrupt is greater than the current processor priority, the exception processing sequence is started. First a copy of the status register is saved, and the privilege state is set to supervisor, tracing is suppressed, and the processor priority level is set to the level of the interrupt being acknowledged. The processor fetches the vector number from the interrupting device, classifying the reference as an interrupt acknowledge and displaying the level number of the interrupt being acknowledged on the address bus. If external logic requests an automatic vectoring, the processor internally generates a vector number which is determined by the interrupt level number. If external logic indicates a bus error, the interrupt is taken to be spurious, and the generated vector number references the spurious interrupt vector. The processor then

proceeds with the usual exception processing, saving the program counter and status register on the supervisor stack. The saved value of the program counter is the address of the instruction which would have been executed had the interrupt not been present. The content of the interrupt vector whose vector number was previously obtained is fetched and loaded into the program counter, and normal instruction execution commences in the interrupt handling routine. A flow chart for the interrupt acknowledge sequence is given in Figure 30; a timing diagram is given in Figure 31.

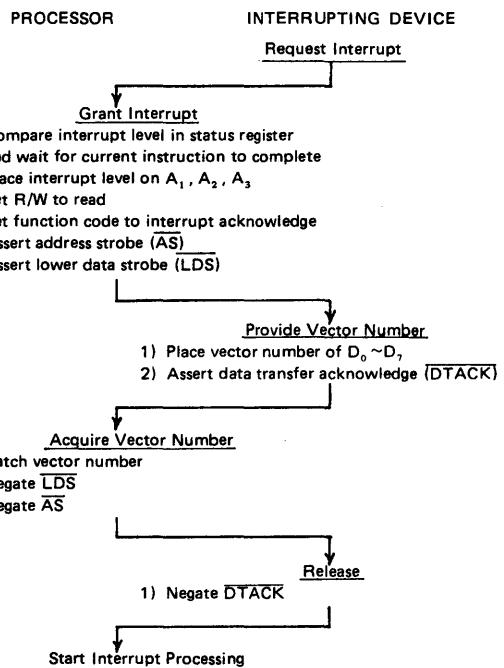


Figure 30 Interrupt Acknowledge Sequence Flow Chart

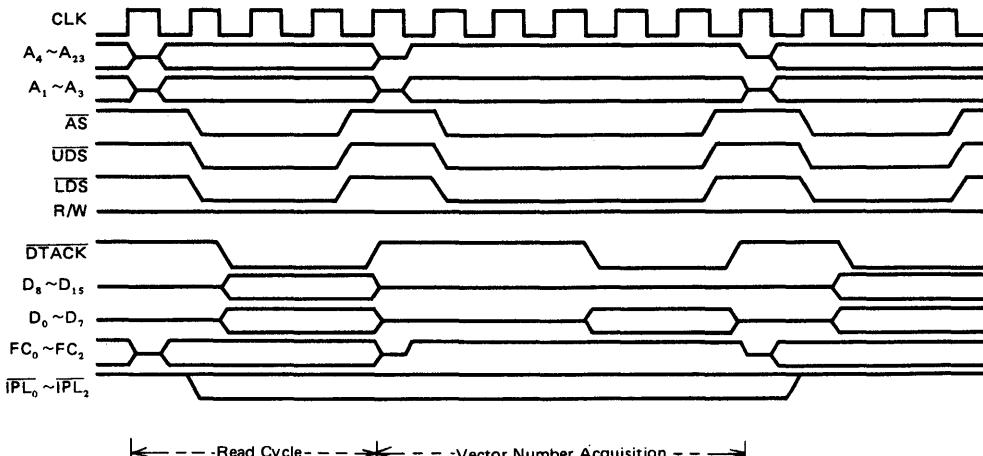


Figure 31 Interrupt Acknowledge Sequence Timing Diagram

Priority level seven is a special case. Level seven interrupts cannot be inhibited by the interrupt priority mask, thus providing a "non-maskable interrupt" capability. An interrupt is generated each time the interrupt request level changes from some lower level to level seven. Note that a level seven interrupt may still be caused by the level comparison if the request level is a seven and the processor priority is set to a lower level by an instruction.

### INSTRUCTION TRAPS

Traps are exceptions caused by instructions. They arise either from processor recognition of abnormal conditions during instruction execution, or from use of instructions whose normal behavior is trapping.

Some instructions are used specifically to generate traps. The TRAP instruction always forces an exception, and is useful for implementing system calls for user programs. The TRAPV and CHK instructions force an exception if the user program detects a runtime error, which may be an arithmetic overflow or a subscript out of bounds.

The signed divide (DIVS) and unsigned divide (DIVU) instructions will force an exception if a division operation is attempted with a divisor of zero.

### ILLEGAL AND UNIMPLEMENTED INSTRUCTIONS

Illegal instruction is the term used to refer to any of the word bit patterns which are not the bit pattern of the first word of a legal instruction. During instruction execution, if such an instruction is fetched, an illegal instruction exception occurs.

Word patterns with bits 15 through 12 equaling 1010 or 1111 are distinguished as unimplemented instructions and separate exception vectors are given to these patterns to permit efficient emulation. This facility allows the operating system to detect program errors, or to emulate unimplemented instructions in software.

### PRIVILEGE VIOLATIONS

In order to provide system security, various instructions are privileged. An attempt to execute one of the privileged instructions while in the user state will cause an exception. The privileged instructions are:

STOP	AND (word) Immediate to SR
RESET	EOR (word) Immediate to SR
RTE	OR (word) Immediate to SR
MOVE to SR	MOVE USP

### TRACING

To aid in program development, the HD68000 includes a facility to allow instruction by instruction tracing. In the trace state, after each instruction is executed an exception is forced, allowing a debugging program to monitor the execution of the program under test.

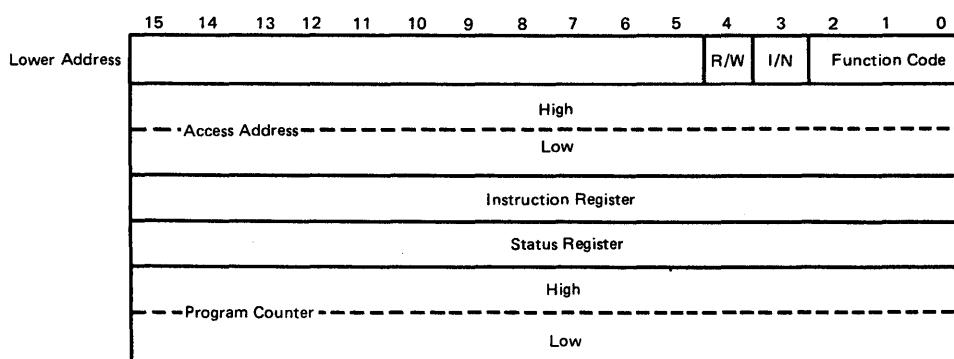
The trace facility uses the T-bit in the supervisor portion of the status register. If the T-bit is negated (off), tracing is disabled, and instruction execution proceeds from instruction to instruction as normal. If the T-bit is asserted (on) at the beginning of the execution of an instruction, a trace exception will be generated after the execution of that instruction is completed. If the instruction is not executed, either because an interrupt is taken, or the instruction is illegal or privileged, the trace exception does not occur. The trace exception also does not occur if the instruction is aborted by a reset, bus error, or address error exception. If the instruction is indeed executed and an interrupt is pending on completion, the trace exception is processed before the interrupt exception. If, during the execution of the instruction, an exception is forced by that instruction, the forced exception is processed before the trace exception.

As an extreme illustration of the above rules, consider the arrival of an interrupt during the execution of a TRAP instruction while tracing is enabled. First the trap exception is processed, then the trace exception, and finally the interrupt exception. Instruction execution resumes in the interrupt handler routine.

### BUS ERROR

Bus error exceptions occur when the external logic requests that a bus error be processed by an exception. The current bus cycle which the processor is making is then aborted. Whether the processor was doing instruction or exception processing, that processing is terminated, and the processor immediately begins exception processing.

Exception processing for bus error follows the usual sequence of steps. The status register is copied, the supervisor state is entered, and the trace state is turned off. The vector number is generated to refer to the bus error vector. Since the processor was not between instructions when the bus error exception request was made, the context, additional information is saved on the supervisor stack. The program counter and the copy of the status register are of course saved. The value



R/W (read/write): write = 0, read = 1, I/N (instruction/not); instruction = 0, not = 1

Figure 32 Supervisor Stack Order

saved for the program counter is advanced by some amount, two to ten bytes beyond the address of the first word of the instruction which made the reference causing the bus error. If the bus error occurred during the fetch of the next instruction, the saved program counter has a value in the vicinity of the current instruction, even if the current instruction is a branch, a jump, or a return instruction. Besides the usual information, the processor saves its internal copy of the first word of the instruction being processed, and the address which was being accessed by the aborted bus cycle. Specific information about the access is also saved: whether it was a read or a write, whether the processor was processing an instruction or not, and the classification displayed on the function code outputs when the bus error occurred. The processor is processing an instruction if it is in the normal state or processing a Group 2 exception; the processor is not processing an instruction if it is processing a Group 0 or a Group 1 exception. Figure 32 illustrates how this information is organized on the supervisor stack. Although this information is not sufficient in general to effect full recovery from the bus error, it does allow software diagnosis. Finally, the processor commences instruction processing at the address contained in the vector. It is the responsibility of the error handler routine to clean up the stack and determine where to continue execution.

If a bus error occurs during the exception processing for a bus error, address error, or reset, the processor is halted, and all processing ceases. This simplifies the detection of catastrophic system failure, since the processor removes itself from the system rather than destroy all memory contents. Only the RESET pin can restart a halted processor.

#### ADDRESS ERROR

Address error exceptions occur when the processor attempts to access a word or a long word operand or an instruction at an odd address. The effect is much like an internally generated bus error, so that the bus cycle is aborted, and the processor ceases whatever processing it is currently doing and begins exception processing. After exception processing commences, the sequence is the same as the for bus error including the information that is stacked, except that the vector number refers to the address error vector instead. Likewise, if an address error occurs during the exception processing for a bus error, address error, or reset, the processor is halted.

#### ■ INTERFACE WITH HMCS6800 PERIPHERALS

Hitachi's extensive line of HMCS6800 peripherals are directly compatible with the HD68000. Some of these devices that are particularly useful are:

- HD6821 Peripheral Interface Adapter
- HD6840 Programmable Timer Module
- HD6843S Floppy Disk Controller
- HD6845S CRT Controller
- HD46508 Data Acquisition Unit
- HD6850 Asynchronous Communication Interface Adapter
- HD6852 Synchronous Serial Data Adapter

To interface the synchronous HMCS6800 peripherals with the asynchronous HD68000, the processor modifies its bus cycle to meet the HMCS6800 cycle requirements whenever an HMCS6800 device address is detected. This is possible since both processors use memory mapped I/O. Figure 33 is a flow chart of the interface operation between the processor and HMCS6800 devices.

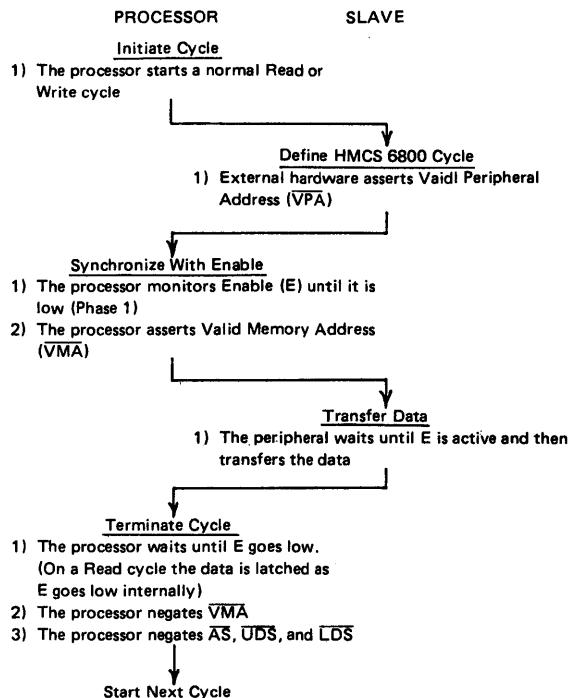


Figure 33 HMCS6800 Interfacing Flow Chart

#### ● DATA TRANSFER OPERATION

Three signals on the processor provide the HMCS6800 interface. They are: enable (E), valid memory address (VMA), and valid peripheral address (VPA). Enable corresponds to the E or  $\phi_2$  signal in existing HMCS6800 systems. It is the bus clock used by the frequency clock that is one tenth of the incoming HD68000 clock frequency. The timing of E allows 1 MHz peripherals to be used with an 8 MHz HD68000. Enable has a 60/40 duty cycle; that is, it is low for six input clocks and high for four input clocks. This duty cycle allows the processor to do successive VPA accesses on successive E pulses.

HMCS6800 cycle timing is given in Figure 34. At state zero (S0) in the cycle, the address bus and function codes are in the high-impedance state. One half clock later, in state 1, the address bus and function code outputs are released from the high-impedance state.

During state 2, the address strobe (AS) is asserted to indicate that there is a valid address on the address bus. If the bus cycle is a read cycle, the upper and/or lower data strobes are also asserted in state 2. If the bus cycle is a write cycle, the read/write (R/W) signal is switched to low (write) during state 2. One half clock later, in state 3, the write data is placed on the data bus, and in state 4 the data strobes are issued to indicate valid data on the data bus.

The processor now inserts wait states until it recognizes the assertion of  $\overline{VPA}$ . The  $\overline{VPA}$  input signals the processor that the address on the bus is the address of an HMCS6800 device (or an area reserved for HMCS6800 devices) and that the bus should conform to the  $\phi_2$  transfer characteristics of the HMCS6800 bus. Valid peripheral address is derived by decoding the address bus, conditioned by address strobe.

After the recognition of  $\overline{VPA}$ , the processor assures that the Enable (E) is "Low", by waiting if necessary, and subsequently asserts VMA. Valid memory address is then used as part of the chip select equation of the peripheral. This ensures that the HMCS6800 peripherals are selected and deselected at the correct time. The peripheral now runs its cycle during the high portion of the E signal.

During a read cycle, the processor latches the peripheral data in state 6. For all cycles, the processor negates the address and data strobes one half clock cycle later in state 7, and the Enable signal goes "Low" at this time. Another half clock later, the address bus is put in the high-impedance state. During a write cycle, the data bus is put in the high-impedance state and the read/write signal is switched high at this time. The peripheral logic must remove  $\overline{VPA}$  within one clock after address strobe is negated.

Figure 35 shows the timing required by HMCS6800 peripherals, the timing specified for the HMCS6800, and the corresponding timing for the HD68000. For further details on peripheral timing, consult the current data sheet for the peripheral of interest. Notice that the HD68000 VMA is active low, contrasted with the active high HMCS6800 VMA. This allows the processor to put its buses in the high-impedance state on DMA requests without inadvertently selecting peripherals.

#### • INTERRUPT INTERFACE OPERATION

During an interrupt acknowledge cycle while the processor is fetching the vector; if  $\overline{VPA}$  is asserted, the HD68000 will assert VMA and complete a normal HMCS6800 read cycle as shown in Figure 36. The processor will then use an internally generated vector that is a function of the interrupt being serviced. This process is known as autovectoring. The seven autovectors are vector number 25 through 31 (decimal).

This operates in the same fashion (but is not restricted to) the HMCS6800 interrupt sequence. The basic difference is that there are six normal interrupt vectors and one NMI type vector. As with both the HMCS6800 and the HD68000's normal vectored interrupt, the interrupt service routine can be located anywhere in the address space. This is due to the fact that while the vector numbers are fixed, the contents of the vector table entries are assigned by the user.

Since VMA is asserted during autovectoring, the HMCS6800 peripheral address decoding should prevent unintended accesses.

#### ■ INSTRUCTION SET

The following paragraphs provide information about the addressing categories and instruction set of the HD68000.

#### • ADDRESSING CATEGORIES

Effective address modes may be categorized by the ways in which they may be used. The following classifications will be used in the instruction definitions.

**Data** If an effective address mode may be used to refer to data operands, it is considered a data addressing effective address mode.

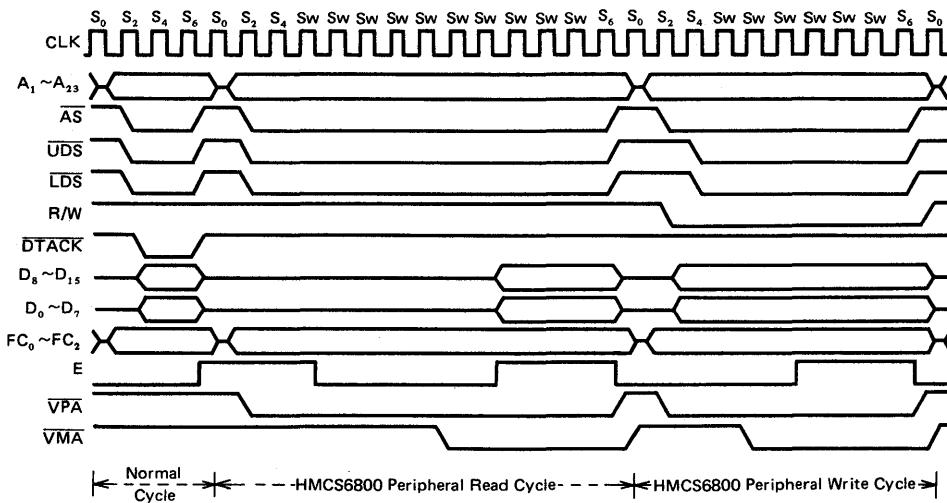


Figure 34 HMCS6800 Cycle Operation

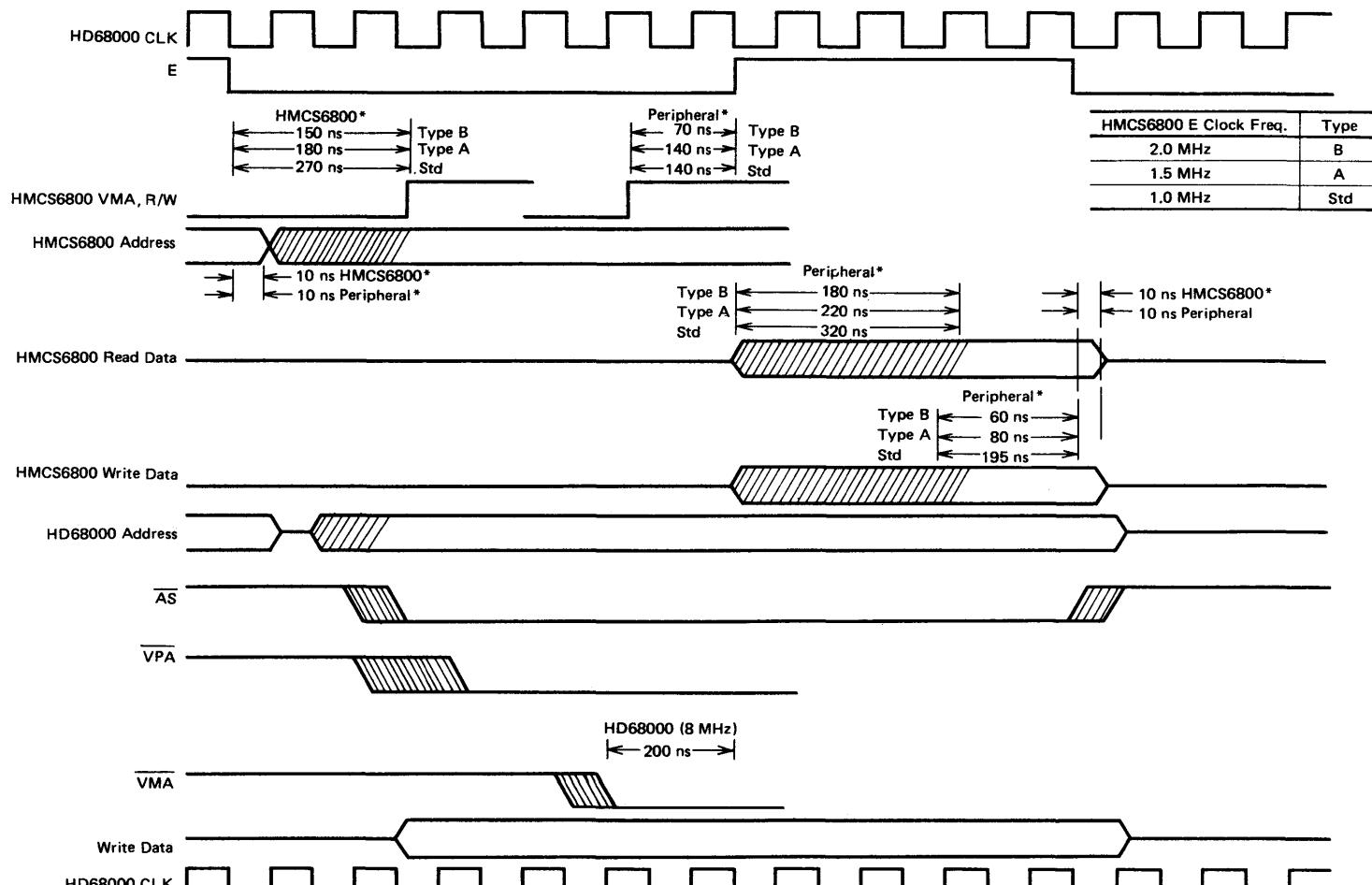


Figure 35 HD68000 to HMCS6800 Peripheral Timing Diagram

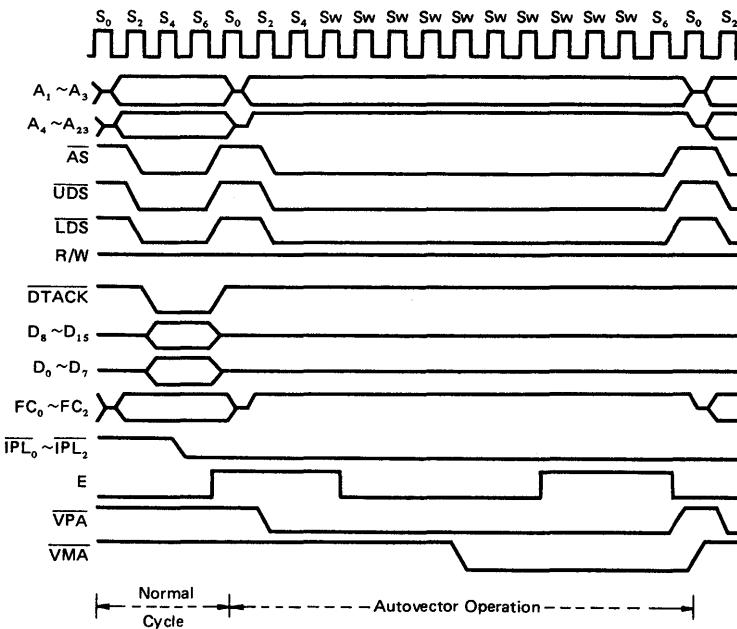


Figure 36 Autovector Operation Timing Diagram

**Memory** If an effective address mode may be used to refer to memory operands, it is considered a memory addressing effective address mode.

**Alterable** If an effective address mode may be used to refer to alterable (writeable) operands, it is considered an alterable addressing effective address mode.

**Control** If an effective address mode may be used to refer to memory operands without an associated size, it is considered a control addressing effective address mode.

Table 20 shows the various categories to which each of the effective address modes belong. Table 21 is the instruction set summary.

The status register addressing mode is not permitted unless it is explicitly mentioned as a legal addressing mode.

These categories may be combined, so that additional, more restrictive, classifications may be defined. For example, the instruction descriptions use such classifications as alterable memory or data alterable. The former refers to those addressing modes which are both alterable and memory addresses, and the

Table 20 Effective Addressing Mode Categories

Effective Address Modes	Mode	Register	Data	Addressing Categories		
				Memory	Control	Alterable
Dn	000	Register number	X	-	-	X
An	001	Register number	-	-	-	X
An@	010	Register number	X	X	X	X
An@ +	011	Register number	X	X	-	X
An@ -	100	Register number	X	X	-	X
An@ (d)	101	Register number	X	X	X	X
An@ (d, ix)	110	Register number				
xxx. W	111	000	X	X	X	X
xxx. L	111	001	X	X	X	X
PC@ (d)	111	010	X	X	X	-
PC@ (d, ix)	111	011	X	X	X	-
#xxx	111	100	X	X	-	-

Table 21 Instruction Set

Mnemonic	Description	Operation	Condition Codes				
			X	N	Z	V	C
ABCD	Add Decimal with Extend	(Destination) <sub>10</sub> + (Source) <sub>10</sub> → Destination	*	U	*	U	*
ADD	Add Binary	(Destination) + (Source) → Destination	*	*	*	*	*
ADDA	Add Address	(Destination) + (Source) → Destination	—	—	—	—	—
ADDI	Add Immediate	(Destination) + Immediate Data → Destination	*	*	*	*	*
ADDQ	Add Quick	(Destination) + Immediate Data → Destination	*	*	*	*	*
ADDX	Add Extended	(Destination) + (Source) + X → Destination	*	*	*	*	*
AND	AND Logical	(Destination) Λ (Source) → Destination	—	*	*	0	0
ANDI	AND Immediate	(Destination) Λ Immediate Data → Destination	—	*	*	0	0
ASL, ASR	Arithmetic Shift	(Destination) Shifted by <count> → Destination	*	*	*	*	*
Bcc	Branch Conditionally	If CC then PC + d → PC	—	—	—	—	—
BCHG	Test a Bit and Change	~(<bit number>) OF Destination → Z ~(<bit number>) OF Destination → <bit number> OF Destination	—	—	*	—	—
BCLR	Test a Bit and Clear	~(<bit number>) OF Destination → Z 0 → <bit number> OF Destination	—	—	*	—	—
BRA	Branch Always	PC + d → PC	—	—	—	—	—
BSET	Test Bit and Set	~(<bit number>) OF Destination → Z 1 → <bit number> OF Destination	—	—	*	—	—
BSR	Branch to Subroutine	PC → SP@ - ; PC + d → PC	—	—	—	—	—
BTST	Test a Bit	~(<bit number>) OF Destination → Z	—	—	*	—	—
CHK	Check Register against Bound	If Dn < 0 or Dn > <ea> then TRAP	—	*	U	U	U
CLR	Clear an Operand	0 → Destination	—	0	1	0	0
CMP	Compare	(Destination) - (Source)	—	*	*	*	*
CMPA	Compare Address	(Destination) - (Source)	—	*	*	*	*
CMPI	Compare Immediate	(Destination) - Immediate Data	—	*	*	*	*
CMPM	Compare Memory	(Destination) - (Source)	—	*	*	*	*
DBcc	Test Condition, Decrement and Branch	If ~ CC then Dn - 1 → Dn; if Dn ≠ -1 then PC + d → PC	—	—	—	—	—
DIVS	Signed Divide	(Destination)/(Source) → Destination	—	*	*	*	0
DIVU	Unsigned Divide	(Destination)/(Source) → Destination	—	*	*	*	0
EOR	Exclusive OR Logical	(Destination) ⊕ (Source) → Destination	—	*	*	0	0
EORI	Exclusive OR Immediate	(Destination) ⊕ Immediate Data → Destination	—	*	*	0	0
EXG	Exchange Register	Rx ↔ Ry	—	—	—	—	—
EXT	Sign Extend	(Destination) Sign-extended → Destination	—	*	*	0	0
JMP	Jump	Destination → PC	—	—	—	—	—
JSR	Jump to Subroutine	PC → SP@ - ; Destination → PC	—	—	—	—	—
LEA	Load Effective Address	Destination → An	—	—	—	—	—
LINK	Link and Allocate	An → SP@ - ; SP → An; SP + d → SP	—	—	—	—	—
LSL, LSR	Logical Shift	(Destination) Shifted by <count> → Destination	*	*	*	0	*
MOVE	Move Data from Source to Destination	(Source) → Destination	—	*	*	0	0
MOVE to CCR	Move to Condition Code	(Source) → CCR	*	*	*	*	*
MOVE to SR	Move to the Status Register	(Source) → SR	*	*	*	*	*

\* = Affected    0 = Cleared    U = Undefined    — = Unaffected    1 = Set

Table 21 Instruction Set (Cont.)

Mnemonic	Description	Operation	Condition Codes				
			X	N	Z	V	C
MOVE from SR	Move from the Status Register	SR → Destination	—	—	—	—	—
MOVE USP	Move User Stack Pointer	USP → An; An → USP	—	—	—	—	—
MOVEA	Move Address	(Source) → Destination	—	—	—	—	—
MOVEM	Move Multiple Registers	Registers → Destination (Source) → Registers	—	—	—	—	—
MOVEP	Move Peripheral Data	(Source) → Destination	—	—	—	—	—
MOVEQ	Move Quick	Immediate Data → Destination	—	*	*	0	0
MULS	Signed Multiply	(Destination)* (Source) → Destination	—	*	*	0	0
MULU	Unsigned Multiply	(Destination)* (Source) → Destination	—	*	*	0	0
NBCD	Negate Decimal with Extend	0 – (Destination) <sub>10</sub> – X → Destination	*	U	*	U	*
NEG	Negate	0 – (Destination) → Destination	*	*	*	*	*
NEGX	Negate with Extend	0 – (Destination) – X → Destination	*	*	*	*	*
NOP	No Operation	—	—	—	—	—	—
NOT	Logical Complement	~(Destination) → Destination	—	*	*	0	0
OR	Inclusive OR Logical	(Destination) v (Source) → Destination	—	*	*	0	0
ORI	Inclusive OR Immediate	(Destination) v Immediate Data → Destination	—	*	*	0	0
PEA	Push Effective Address	Destination → SP@ –	—	—	—	—	—
RESET	Reset External Devices	—	—	—	—	—	—
ROL, ROR	Rotate (Without Extend)	(Destination) Rotated by <count> → Destination	—	*	*	0	*
ROXL, ROXR	Rotate with Extend	(Destination) Rotated by <count> → Destination	*	*	*	0	*
RTE	Return from Exception	SP@ + → SR; SP@ + → PC	*	*	*	*	*
RTR	Return and Restore Condition Codes	SP@ + → CC; SP@ + → PC	*	*	*	*	*
RTS	Return from Subroutine	SP@ + → PC					
SBCD	Subtract Decimal with Extend	(Destination) <sub>10</sub> – (Source) <sub>10</sub> – X → Destination	*	U	*	U	*
Scc	Set According to Condition	If CC then 1's → Destination else 0's → Destination	—	—	—	—	—
STOP	Load Status Register and Stop	Immediate Data → SR; STOP	*	*	*	*	*
SUB	Subtract Binary	(Destination) – (Source) → Destination	*	*	*	*	*
SUBA	Subtract Address	(Destination) – (Source) → Destination	—	—	—	—	—
SUBI	Subtract Immediate	(Destination) – Immediate Data → Destination	*	*	*	*	*
SUBQ	Subtract Quick	(Destination) – Immediate Data → Destination	*	*	*	*	*
SUBX	Subtract with Extend	(Destination) – (Source) – X → Destination	*	*	*	*	*
SWAP	Swap Register Halves	Register [31:16] ↔ Register [15:0]	—	*	*	0	0
TAS	Test and Set and Operand	(Destination) Tested → CC; 1 → [7] OF Destination	—	*	*	0	0
TRAP	Trap	PC → SSP@- ; SR → SSP@- ; (Vector) → PC	—	—	—	—	—
TRAPV	Trap on Overflow	If V then TRAP	—	—	—	—	—
TST	Test an Operand	(Destination) Tested → CC	—	*	*	0	0
UNLK	Unlink	An → SP; SP@ + → An	—	—	—	—	—

[ ] = Bit number

latter refers to addressing modes which are both data and alterable.

#### ■ INSTRUCTION EXECUTION TIMES

The following paragraphs contain listings of the instruction execution times in terms of external clock (CLK) periods. In this timing data, it is assumed that the memory cycle time is no greater than four periods of the external processor clock input, which prevents the insertion of wait states in the bus cycle. The number of bus read and write cycle for each instruction is also included with the timing data. This data is enclosed in parenthesis following the execution periods and is shown as: (R/W) where R is the number of read cycles and W is the number of write cycles.

[NOTE] The number of periods includes instruction fetch and all applicable operand fetches and stores.

#### ● EFFECTIVE ADDRESS OPERAND CALCULATION TIMING

Table 22 lists the number of clock periods required to compute an instruction's effective address. It includes fetching of any extension words, the address computation, and fetching of the memory operand. The number of bus read and write cycles is shown in parenthesis as (R/W). Note there are no write cycles involved in processing the effective address.

#### ● MOVE INSTRUCTION CLOCK PERIODS

Table 23 and 24 indicate the number of clock periods for the move instruction. This data includes instruction fetch, operand reads, and operand writes. The number of bus read and write cycles is shown in parenthesis as: (R/W).

#### ● STANDARD INSTRUCTION CLOCK PERIODS

The number of clock periods shown in Table 25 indicates the time required to perform the operations, store the results, and read the next instruction. The number of bus read and write cycle is shown in parenthesis as: (R/W). The number of clock periods plus the number of read and write cycles must be added to those of the effective address calculation where indicated.

In Table 25, the headings have the following meanings: An = address register operand, Dn = data register operand, ea = an operand specified by an effective address, and M = memory effective address operand.

#### ● IMMEDIATE INSTRUCTION CLOCK PERIODS

The number of clock periods shown in Table 26 includes the time to fetch immediate operands, perform the operations, store the results, and read the next operation. The number of bus read and write cycles is shown in parenthesis as: (R/W). The number of clock periods plus the number of read and write cycles must be added to those of the effective address calculation where indicated.

In Table 26, the headings have the following meanings: # = immediate operand, Dn = data register operand, M = memory operand, and SR = status register.

#### ● SINGLE OPERAND INSTRUCTION CLOCK PERIODS

Table 27 indicates the number of clock periods for the single operand instructions. The number of bus read and write cycles is shown in parenthesis as: (R/W). The number of clock periods plus the number of read and write cycles must be added to those of the effective address calculation where indicated.

#### ● SHIFT/ROTATE INSTRUCTION CLOCK PERIODS

Table 28 indicates the number of clock periods for the shift and rotate instructions. The number of bus read and write cycles is shown in parenthesis as: (R/W). The number of clock periods plus the number of read and write cycles must be added to those of the effective address calculation where indicated.

#### ● BIT MANIPULATION INSTRUCTION CLOCK PERIODS

Table 29 indicates the number of clock periods required for the bit manipulation instructions. The number of bus read and write cycles is shown in parenthesis as: (R/W). The number of clock periods plus the number of read and write cycles must be added to those of the effective address calculation where indicated.

Table 22 Effective Address Calculation Timing

Addressing Mode		Byte, Word	Long
Dn	Register		
	Data Register Direct	0 (0/0)	0 (0/0)
An	Address Register Direct	0 (0/0)	0 (0/0)
	Memory		
An@	Address Register Indirect	4 (1/0)	8 (2/0)
	Address Register Indirect with Postincrement	4 (1/0)	8 (2/0)
An@ -	Address Register Indirect with Predecrement	6 (1/0)	10 (2/0)
	Address Register Indirect with Displacement	8 (2/0)	12 (3/0)
An@ (d, ix)*	Address Register Indirect with Index	10 (2/0)	14 (3/0)
	Absolute Short	8 (2/0)	12 (3/0)
xxx. W	Absolute Long	12 (3/0)	16 (4/0)
PC@ (d)	Program Counter with Displacement	8 (2/0)	12 (3/0)
PC@ (d, ix)*	Program Counter with Index	10 (2/0)	14 (3/0)
	Immediate	4 (1/0)	8 (2/0)

\* The size of the index register (ix) does not affect execution time.

Table 23 Move Byte and Word Instruction Clock Periods

Source	Destination									
	Dn	An	An@	An@+	An@-	An@(d)	An@(d, ix)*	xxx.W	xxx.L	
Dn	4 (1/0)	4(1/0)	9 (1/1)	9 (1/1)	9 (1/1)	13 (2/1)	15 (2/1)	13 (2/1)	17 (3/1)	
An	4 (1/0)	4 (1/0)	9 (1/1)	9 (1/1)	9 (1/1)	13 (2/1)	15 (2/1)	13 (2/1)	17 (3/1)	
An@	8 (2/0)	8 (2/0)	13 (2/1)	13 (2/1)	13 (2/1)	17 (3/1)	19 (3/1)	17 (3/1)	21 (4/1)	
An@ +	8 (2/0)	8 (2/0)	13 (2/1)	13 (2/1)	13 (2/1)	17 (3/1)	19 (3/1)	17 (3/1)	21 (4/1)	
An@ -	10 (2/0)	10 (2/0)	15 (2/1)	15 (2/1)	15 (3/1)	19 (3/1)	21 (3/1)	19 (3/1)	23 (4/1)	
An@ (d)	12 (3/0)	12 (3/0)	17 (3/1)	17 (3/1)	17 (3/1)	21 (4/1)	23 (4/1)	21 (4/1)	25 (5/1)	
An@ (d, ix)*	14 (3/0)	14 (3/0)	19 (3/1)	19 (3/1)	19 (3/1)	23 (4/1)	25 (4/1)	23 (4/1)	27 (5/1)	
xxx.W	12 (3/0)	12 (3/0)	17 (3/1)	17 (3/1)	17 (3/1)	21 (4/1)	23 (4/1)	21 (4/1)	25 (5/1)	
xxx.L	16 (4/0)	16 (4/0)	21 (4/1)	21 (4/1)	21 (4/1)	25 (5/1)	27 (5/1)	25 (5/1)	29 (6/1)	
PC@ (d)	12 (3/0)	12 (3/0)	17 (3/1)	17 (3/1)	17 (3/1)	21 (4/1)	23 (4/1)	21 (4/1)	25 (5/1)	
PC@ (d, ix)*	14 (3/0)	14 (3/0)	19 (3/1)	19 (3/1)	19 (3/1)	23 (4/1)	25 (4/1)	23 (4/1)	27 (5/1)	
#xxx	8 (2/0)	8 (2/0)	13 (2/1)	13 (2/1)	13 (2/1)	17 (3/1)	19 (3/1)	17 (3/1)	21 (4/1)	

\* The size of the index register (ix) does not affect execution time.

Table 24 Move Long Instruction Clock Periods

Source	Destination									
	Dn	An	An@	An@+	An@-	An@(d)	An@(d, ix)*	xxx.W	xxx.L	
Dn	4 (1/0)	4 (1/0)	14 (1/2)	14 (1/2)	16 (1/2)	18 (2/2)	20 (2/2)	18 (2/2)	22 (3/2)	
An	4 (1/0)	4 (1/0)	14 (1/2)	14 (1/2)	16 (1/2)	18 (2/2)	20 (2/2)	18 (2/2)	22 (3/2)	
An@	12 (3/0)	12 (3/0)	22 (3/2)	22 (3/2)	22 (3/2)	26 (4/2)	28 (4/2)	26 (4/2)	30 (5/2)	
An@ +	12 (3/0)	12 (3/0)	22 (3/2)	22 (3/2)	22 (3/2)	26 (4/2)	28 (4/2)	26 (4/2)	30 (5/2)	
An@ -	14 (3/0)	14 (3/0)	24 (3/2)	24 (3/2)	24 (3/2)	28 (4/2)	30 (4/2)	28 (4/2)	32 (5/2)	
An@ (d)	16 (4/0)	16 (4/0)	26 (4/2)	26 (4/2)	26 (4/2)	30 (5/2)	32 (5/2)	30 (5/2)	34 (6/2)	
An@ (d, ix)*	18 (4/0)	18 (4/0)	28 (4/2)	28 (4/2)	28 (4/2)	32 (5/2)	34 (5/2)	32 (5/2)	36 (6/2)	
xxx.W	16 (4/0)	16 (4/0)	26 (4/2)	26 (4/2)	26 (4/2)	30 (5/2)	32 (5/2)	30 (5/2)	34 (6/2)	
xxx.L	20 (5/0)	20 (5/0)	30 (5/2)	30 (5/2)	30 (5/2)	34 (6/2)	36 (6/2)	34 (6/2)	38 (7/2)	
PC@ (d)	16 (4/0)	16 (4/0)	26 (4/2)	26 (4/2)	26 (4/2)	30 (5/2)	32 (5/2)	30 (5/2)	34 (6/2)	
PC@ (d, ix)*	18 (4/0)	18 (4/0)	28 (4/2)	28 (4/2)	28 (4/2)	32 (5/2)	34 (5/2)	32 (5/2)	36 (6/2)	
#xxx	12 (3/0)	12 (3/0)	22 (3/2)	22 (3/2)	22 (3/2)	26 (4/2)	28 (4/2)	26 (4/2)	30 (5/2)	

\* The size of the index register (ix) does not affect execution time.

Table 25 Standard Instruction Clock Periods

Instruction	Size	op <ea>, An	op <ea>, Dn	op Dn, <M>
ADD	Byte, Word	8 (1/0)+	4 (1/0)+	9 (1/1)+
	Long	6 (1/0)+**	6 (1/0)+**	14 (1/2)+
AND	Byte, Word	—	4 (1/0)+	9 (1/1)+
	Long	—	6 (1/0)+**	14 (1/2)+
CMP	Byte, Word	6 (1/0)+	4 (1/0)+	—
	Long	6 (1/0)+	6 (1/0)+	—
DIVS	—	—	158 (1/0)+*	—
DIVU	—	—	140 (1/0)+*	—
EOR	Byte, Word	—	4 (1/0)***	9 (1/1)+
	Long	—	8 (1/0)***	14 (1/2)+
MULS	—	—	70 (1/0)+*	—
MULU	—	—	70 (1/0)+*	—
OR	Byte, Word	—	4 (1/0)+	9 (1/1)+
	Long	—	6 (1/0)+**	14 (1/2)+
SUB	Byte, Word	8 (1/0)+	4 (1/0)+	9 (1/1)+
	Long	6 (1/0)+**	6 (1/0)+**	14 (1/2)+

+ Add effective address calculation time

\* Indicates maximum value

\*\* Total of 8 clock periods for instruction if the effective address is register direct

\*\*\* Only available effective address mode is data register direct.

Table 26 Immediate Instruction Clock Periods

Instruction	Size	op #, Dn	op #, M	op #, SR
ADDI	Byte, Word	8 (2/0)	13 (2/1)+	—
	Long	16 (3/0)	22 (3/2)+	—
ADDQ	Byte, Word	4 (1/0)	9 (1/1)+	—
	Long	8 (1/0)	14 (1/2)+	—
ANDI	Byte, Word	8 (2/0)	13 (2/1)+	20 (3/0)
	Long	16 (3/0)	22 (3/2)+	—
CMPI	Byte, Word	8 (2/0)	8 (2/0)+	—
	Long	14 (3/0)	12 (3/0)+	—
EORI	Byte, Word	8 (2/0)	13 (2/1)+	20 (3/0)
	Long	16 (3/0)	22 (3/2)+	—
MOVEQ	Long	4 (1/0)	—	—
ORI	Byte, Word	8 (2/0)	13 (2/1)+	20 (3/0)
	Long	16 (3/0)	22 (3/2)+	—
SUBI	Byte, Word	8 (2/0)	13 (2/1)+	—
	Long	16 (3/0)	22 (3/2)+	—
SUBQ	Byte, Word	4 (1/0)	9 (1/1)+	—
	Long	8 (1/0)	14 (1/2)+	—

+ Add effective address calculation time

Table 27 Single Operand Instruction Clock Periods

Instruction	Size	Register	Memory
CLR	Byte, Word	4 (1/0)	9 (1/1)+
	Long	6 (1/0)	14 (1/2)+
NBCD	Byte	6 (1/0)	9 (1/1)+
	Byte, Word	4 (1/0)	9 (1/1)+
NEG	Long	6 (1/0)	14 (1/2)+
	Byte, Word	4 (1/0)	9 (1/1)+
NEGX	Long	6 (1/0)	14 (1/2)+
	Byte, Word	4 (1/0)	9 (1/1)+
NOT	Long	6 (1/0)	14 (1/2)+
	Byte, Word	4 (1/0)	9 (1/1)+
Scc	Byte, False	4 (1/0)	9 (1/1)+
	Byte, True	6 (1/0)	9 (1/1)+
TAS	Byte	4 (1/0)	11 (1/1)+
TST	Byte, Word	4 (1/0)	4 (1/0)+
	Long	4 (1/0)	4 (1/0)+

+ Add effective address calculation time.

Table 28 Shift/Rotate Instruction Clock Periods

Instruction	Size	Register	Memory
ASR, ASL	Byte, Word	6 + 2n (1/0)	9 (1/1)+
	Long	8 + 2n (1/0)	—
LSR, LSL	Byte, Word	6 + 2n (1/0)	9 (1/1)+
	Long	8 + 2n (1/0)	—
ROR, ROL	Byte, Word	6 + 2n (1/0)	9 (1/1)+
	Long	8 + 2n (1/0)	—
ROXR, ROXL	Byte, Word	6 + 2n (1/0)	9 (1/1)+
	Long	8 + 2n (1/0)	—

Table 29 Bit Manipulation Instruction Clock Periods

Instruction	Size	Dynamic		Static	
		Register	Memory	Register	Memory
BCHG	Byte	—	9 (1/1)+	—	13 (2/1)+
	Long	8 (1/0)*	—	12 (2/0)*	—
BCLR	Byte	—	9 (1/1)+	—	13 (2/1)+
	Long	10 (1/0)*	—	14 (2/0)*	—
BSET	Byte	—	9 (1/1)+	—	13 (2/1)+
	Long	8 (1/0)*	—	12 (2/0)*	—
BTST	Byte	—	4 (1/0)+	—	8 (2/0)+
	Long	6 (1/0)	—	10 (2/0)	—

+ Add effective address calculation time

\* Indicates maximum value

- CONDITIONAL INSTRUCTION CLOCK PERIODS**

Table 30 indicates the number of clock periods required for the conditional instructions. The number of bus read and write cycles is indicated in parenthesis as: (R/W). The number of clock periods plus the number of read and write cycles must be added to those of the effective address calculation where indicated.

- JMP, JSR, LEA, PEA, MOVEM INSTRUCTION CLOCK PERIODS**

Table 31 indicates the number of clock periods required for the jump, jump to subroutine, load effective address, push effective address, and move multiple registers instructions. The number of bus read and write cycles is shown in parenthesis as: (R/W).

- MULTI-PRECISION INSTRUCTION CLOCK PERIODS**

Table 32 indicates the number of clock periods for the multi-precision instructions. The number of clock periods includes the time to fetch both operands, perform the operations, store the results, and read the next instructions. The number of read and write cycles is shown in parenthesis as: (R/W).

In Table 32, the headings have the following meanings: Dn = data register operand and M = memory operand.

- MISCELLANEOUS INSTRUCTION CLOCK PERIODS**

Table 33 indicates the number of clock periods for the following miscellaneous instructions. The number of bus read and write cycles is shown in parenthesis as: (R/W). The number of clock periods plus the number of read and write cycles must be added to those of the effective address calculation where indicated.

Table 30 Conditional Instruction Clock Periods

Instruction	Displacement	Trap or Branch Taken	Trap or Branch Not Taken
B <sub>CC</sub>	Byte	10 (1/0)	8 (1/0)
	Word	10 (1/0)	12 (2/0)
BRA	Byte	10 (1/0)	—
	Word	10 (1/0)	—
BSR	Byte	20 (2/2)	—
	Word	20 (2/2)	—
DB <sub>CC</sub>	CC true	—	12 (2/0)
	CC false	10 (2/0)	14 (3/0)
CHK	—	43 (5/3)+*	8 (1/0)+
TRAP	—	37 (4/3)	—
TRAPV	—	37 (5/3)	4 (1/0)

+ Add effective address calculation time

\* Indicates maximum value

Table 31 JMP, JSR, LEA, PEA, MOVEM Instruction Clock Periods

Instruction	Size	An@	An@+	An@-	An@(d)	An@(d,ix)*	xxx. W	xxx. L	PC@(d)	PC@(d,ix)*
JMP	—	8 (2/0)	—	—	10 (2/0)	14 (3/0)	10 (2/0)	12 (3/0)	10 (2/0)	14 (3/0)
JSR	—	18 (2/2)	—	—	20 (2/2)	24 (2/2)	20 (2/2)	22 (3/2)	20 (2/2)	24 (2/2)
LEA	—	4 (1/0)	—	—	8 (2/0)	12 (2/0)	8 (2/0)	12 (3/0)	8 (2/0)	12 (2/0)
PEA	—	14 (1/2)	—	—	18 (2/2)	22 (2/2)	18 (2/2)	22 (3/2)	18 (2/2)	22 (2/2)
MOVEM	Word	12 + 4n (3+n/0)	12 + 4n (3+n/0)	—	16 + 4n (4+n/0)	18 + 4n (4+n/0)	16 + 4n (4+n/0)	20 + 4n (5+n/0)	16 + 4n (4+n/0)	18 + 4n (4+n/0)
	Long	12 + 8n (3+2n/0)	12 + 8n (3+2n/0)	—	16 + 8n (4+2n/0)	18 + 8n (4+2n/0)	16 + 8n (4+2n/0)	20 + 8n (6+2n/0)	16 + 8n (4+2n/0)	18 + 8n (4+2n/0)
MOVEM	Word	8 + 5n (2/n)	—	8 + 5n (2/n)	12 + 5n (3/n)	14 + 5n (3/n)	12 + 5n (3/n)	16 + 5n (4/n)	—	—
	Long	8 + 10n (2/2n)	—	8 + 10n (2/2n)	12+10n (3/2n)	14 + 10n (3/2n)	12+10n (3/2n)	16+10n (4/2n)	—	—

n is the number of registers to move

\* is the size of the index register (ix) does not affect the instruction's execution time.

Table 32 Multi-Precision Instruction Clock Periods

Instruction	Size	op Dn, Dn	op M, M
ADDX	Byte, Word	4 (1/0)	19 (3/1)
	Long	8 (1/0)	32 (5/2)
CMPM	Byte, Word	—	12 (3/0)
	long	—	20 (5/0)
SUBX	Byte, Word	4 (1/0)	19 (3/1)
	Long	8 (1/0)	32 (5/2)
ABCD	Byte	6 (1/0)	19 (3/1)
SBCD	Byte	6 (1/0)	19 (3/1)

Table 33 Miscellaneous Instruction Clock Periods

Instruction	Size	Register	Memory	Register → Memory	Memory → Register
MOVE from SR	—	6 (1/0)	9 (1/1)+	—	—
MOVE to CCR	—	12 (2/0)	12 (2/0) +	—	—
MOVE to SR	—	12 (2/0)	12 (2/0) +	—	—
MOVEP	Word	—	—	18 (2/2)	16 (4/0)
	Long	—	—	28 (2/4)	24 (6/0)
EXG	—	6 (1/0)	—	—	—
EXT	Word	4 (1/0)	—	—	—
	Long	4 (1/0)	—	—	—
LINK	—	18 (2/2)	—	—	—
MOVE from USP	—	4 (1/0)	—	—	—
MOVE to USP	—	4 (1/0)	—	—	—
NOP	—	4 (1/0)	—	—	—
RESET	—	132 (1/0)	—	—	—
RTE	—	20 (5/0)	—	—	—
RTR	—	20 (5/0)	—	—	—
RTS	—	16 (4/0)	—	—	—
STOP	—	4 (0/0)	—	—	—
SWAP	—	4 (1/0)	—	—	—
UNLK	—	12 (3/0)	—	—	—

+ Add effective address calculation time

#### • EXCEPTION PROCESSING CLOCK PERIODS

Table 34 indicates the number of clock periods for exception processing. The number of clock periods includes the time for all stacking, the vector fetch, and the fetch of the first instruction of the handler routine. The number of bus read and write cycles is shown in parenthesis as: (R/W).

Table 34 Exception Processing Clock Periods

Exception	Periods
Address Error	57 (4/7)
Bus Error	57 (4/7)
Interrupt	47 (5/3)*
Illegal Instruction	37 (4/3)
Privileged Instruction	37 (4/3)
Trace	37 (4/3)

\* The interrupt acknowledge bus cycle is assumed to take four external clock periods.

HITACHI reserves the right to make changes to any products herein to improve functioning or design. Although the information in this document has been carefully reviewed and is believed to be reliable, HITACHI does not assume any liability arising out of the application or use of any product or circuit described herein, neither does it convey any license under its patent rights nor the rights of others.

# **APPENDIX**

# PACKAGE INFORMATION

Packages are classified into 3 types; dual-in-line plastic, dual-in-line ceramic (glass-sealed) and dual-in-line ceramic (with lid), according to the quality of material used for packaging.

Type	Function	Package*		
		Pin No.	C**	G
HD6800	Micro Processing Unit	40	O	O
HD68A00			O	O
HD68B00			O	O
HD6802S	Microprocessor with Clock and RAM	40	O	O
HD6809	8/16 Bit Micro Processing Unit	40	O	O
HD68A09			O	O
HD68B09			O	O
HD6821	Peripheral Interface Adapter	40	O	O
HD68A21			O	O
HD68B21			O	O
HD6840	Programmable Timer Module	40		
HD68A40			O	O
HD68B40				
HD6850	Asynchronous Communications Interface Adapter	24	O	O
HD68A50			O	O
HD6852	Synchronous Serial Data Adapter	24	O	O
HD68A52			O	O
HD6846	Combination ROM I/O Timer	40	O	O
HD6843S	Floppy Disk Controller	40	O	O
HD68A43S			O	O
HD6844	Direct Memory Access Controller	40	O	O
HD68A44			O	O
HD6845	CRT Controller	40	O	O
HD68A45			O	O
HD68B45			O	O
HD46508	Analog Data Acquisition Unit	40		O
HD46508-1				O
HD68000-4	16/32 Bit Microprocessor	64		
HD68000-6			O-std.	
HD68000-8				
HD68450-4	16 Bit Direct Memory Access Controller	64		
HD68450-6			O-std.	
HD68450-8				

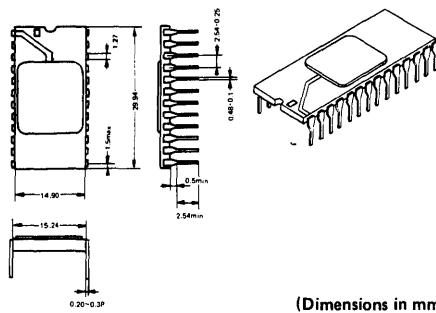
\* The package codes of C, G and P are applied to the package materials as follows.

C; Ceramic with Lid  
 G; Glass — Sealed Ceramic  
 P; Plastic

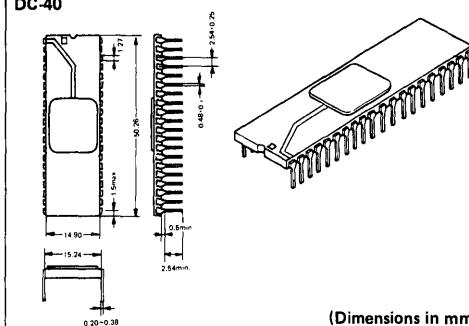
\*\* Special Order Only

● DUAL-IN-LINE CERAMIC (with Lid)

**DC-24**

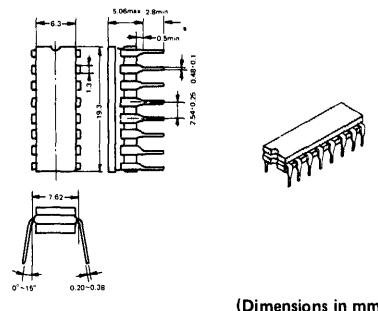


**DC-40**

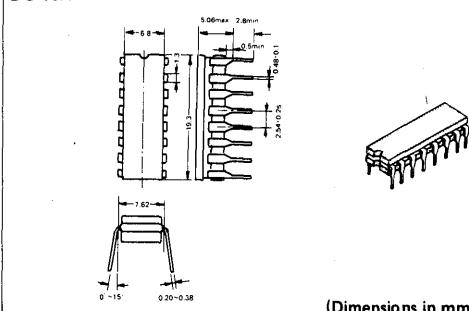


● DUAL-IN-LINE CERAMIC (Glass-sealed)

**DG-16**

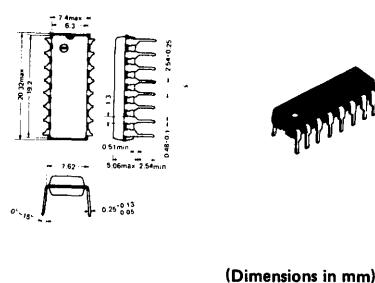


**DG-16A**

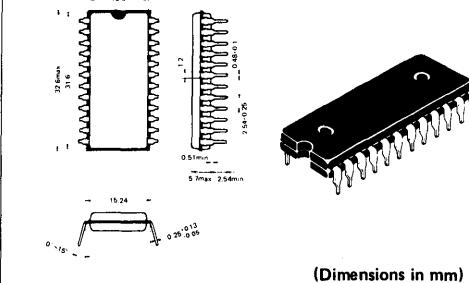


● DUAL-IN-LINE PLASTIC

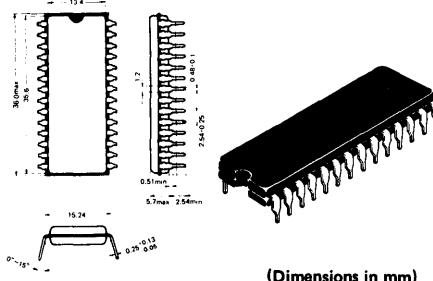
**DP-16**



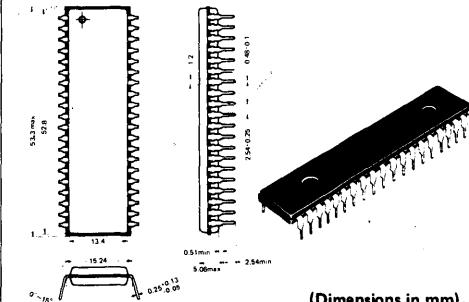
**DP-24**



**DP-28**



**DP-40**



# HMCS6800 FAMILY INSTRUCTIONS

## ■ ACCUMULATOR AND MEMORY OPERATIONS

Operations	Mnemonic	Boolean/Arithmetic Operation	HD6800 HD6802				HD6801 HD6803				HD6805				HD6809			
			IMMED	DIRECT	INDEX	EXTND	IMPL	RELATIVE	IMMED	DIRECT	INDEX	EXTND	IMPL	RELATIVE	IMMED	DIRECT	INDEX	EXTND
Add	ADDA	A + M → A	○	○	○	○			○	○	○	○		○	○	○	○	○
	ADDB	B + M → B	○	○	○	○			○	○	○	○		○	○	○	○	○
Add Double	ADDD	A:B + M:M+1 → A:B							○	○	○	○			○	○	○	○
Add Accumulators	ABA	A + B → A					○					○						
Add with Carry	ADCA	A + M + C → A	○	○	○	○			○	○	○	○		○	○	○	○	○
	ADCB	B + M + C → B	○	○	○	○			○	○	○	○			○	○	○	○
Subtract	SUBA	A - M → A	○	○	○	○			○	○	○	○		○	○	○	○	○
	SUBB	B - M → B	○	○	○	○			○	○	○	○			○	○	○	○
Subtract Double	SUBD	A:B - M:M+1 → A:B							○	○	○	○			○	○	○	○
Subtract Accumulators	SBA	A - B → A					○					○						
Subtract with Carry	SBCA	A - M - C → M	○	○	○	○			○	○	○	○		○	○	○	○	○
	SBCB	B - M - C → M	○	○	○	○			○	○	○	○			○	○	○	○
Multiply	MUL	A × B → A:B										○						○
Decimal Adjust A	DAA	Decimal Adjust Accumulator					○					○						
Increment	INC	M + 1 → M	○	○					○	○				○	○	○	○	○
	INCA	A + 1 → A			○				○		○			○				○
	INCB	B + 1 → B			○				○		○							○
Decrement	DEC	M - 1 → M	○	○					○	○				○	○	○	○	○
	DECA	A - 1 → A			○				○		○			○				○
	DEC B	B - 1 → B			○				○		○							○
Clear	CLR	0 → M	○	○					○	○				○	○	○	○	○
	CLRA	0 → A			○					○				○				○
	CLRB	0 → B			○					○								○
Compare	CMPA	A - M	○	○	○	○			○	○	○	○		○	○	○	○	○
	CMPB	B - M	○	○	○	○			○	○	○	○			○	○	○	○
Compare Double	CMPD	A:B - M:M+1																○
Compare Accumulators	CBA	A - B			○						○							
Test Zero or Minus	TST	M - 00	○	○					○	○				○	○	○	○	○
	TSTA	A - 00			○					○				○				○
	TSTB	B - 00			○					○								○
And	ANDA	A • M → A	○	○	○	○			○	○	○	○		○	○	○	○	○
	ANDB	B • M → B	○	○	○	○			○	○	○	○			○	○	○	○
Or	ORAA	A + M → A	○	○	○	○			○	○	○	○		○	○	○	○	○
	ORA B	B + M → B	○	○	○	○			○	○	○	○			○	○	○	○
Exclusive Or	EORA	A ⊕ M → A	○	○	○	○			○	○	○	○		○	○	○	○	○
	EORB	B ⊕ M → B	○	○	○	○			○	○	○	○		○	○	○	○	○
Complement 1's	COM	M → M		○	○				○	○				○	○	○	○	○
	COMA	Ā → A				○				○				○				○
	COMB	Ā → B				○				○								○
Complement 2's	NEG	00 - M → M		○	○				○	○				○	○	○	○	○
(Negate)	NEGA	00 - A → A				○				○				○				○
	NEGB	00 - B → B				○				○								○
Bit Test	BITA	A • M	○	○	○	○			○	○	○	○		○	○	○	○	○
	BITB	B • M	○	○	○	○			○	○	○	○			○	○	○	○
Bit Clear	BCLR n	0 → M • Bit (n)												○				
Bit Set	BSET n	1 → M • Bit (n)												○				
	SEX	Sign Extend B into A																○

Operations	Mnemonic	Boolean/ Arithmetic Operation	HD6800 HD6802				HD6801 HD6803				HD6805				HD6809						
			IMMED	DIRECT	INDEX	EXTND	IMPL	RELATIVE	IMMED	DIRECT	INDEX	EXTND	IMPL	RELATIVE	IMMED	DIRECT	INDEX	EXTND	INDIRECT	IMPL	RELATIVE
Load	LDA	M → A	○	○	○	○	○		○	○	○	○		○	○	○	○	○	○	○	
	LDAA																				
	LDAB	M → B	○	○	○	○			○	○	○	○					○	○	○	○	
Store	LDB																				
	LD	M:M+1 → A:B							○	○	○	○					○	○	○	○	
	LDAA																				
Transfer	STA	A → M	○	○	○				○	○	○			○	○	○	○	○	○	○	
	STAB																				
	STB	B → M	○	○	○				○	○	○						○	○	○	○	
Exchange	STD	A:B → M:M+1							○	○	○						○	○	○	○	
	TFR	Register 1 → Register 2																			○
	TAB	A → B							○					○							
Push Data	TBA	B → A							○					○							
	EXG	Register 1 → Register 2																			○
	PSHA	A → Ms S - 1 → S							○					○							
Push Data	PSHB	B → Ms S - 1 → S							○					○							
	PSHS	Registers → Ms S - n → S																			○
	PSHU	Registers → Mu U - n → U																			○
Pull Data	PULA	Ms → A S + 1 → S							○					○							
	PULB	Ms → B S + 1 → S							○					○							
	PULS	Ms → Registers S + n → S																			○
Shift Left Arithmetic	PULU	Mu → Registers U + n → U																			○
	ASL	M							○	○				○	○		○	○	○	○	○
	ASLA	A							○					○			○				○
Shift Right Arithmetic	ASLB	B							○					○			○				○
	ASLD	C							○					○			○				○
	A:B	MSB							○					○			○				○
Shift Right Arithmetic	ASR	M							○	○				○	○		○	○	○	○	○
	ASRA	A							○					○			○				○
	ASRB	B							○					○			○				○
Shift Right Logical	LSR	MSB							○	○				○	○		○	○	○	○	○
	LSRA	LSB							○					○			○				○
	LSRB	C							○					○			○				○
Rotate Left	LSRD	MSB							○					○			○				○
	ROL	LSB							○					○			○				○
	ROLA	C							○					○			○				○
Rotate Right	ROLB	MSB							○					○			○				○
	ROR	LSB							○					○			○				○
	RORA	C							○					○			○				○
Rotate Right	RORB	MSB							○					○			○				○

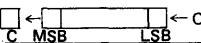
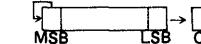
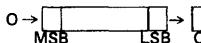
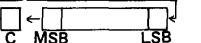
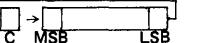
## JUMP AND BRANCH INSTRUCTIONS

Operations	Mnemonic	Branch Test	HD6800 HD6802				HD6801 HD6803				HD6805				HD6809						
			IMMED	DIRECT	INDEX	EXTND	IMPL	RELATIVE	IMMED	DIRECT	INDEX	EXTND	IMPL	RELATIVE	IMMED	DIRECT	INDEX	EXTND	INDIRECT	IMPL	RELATIVE
Branch if Carry Clear	BCC	C = 0				O						O			O						
	LBCC																				
Branch if Carry Set	BCS	C = 1				O						O			O						
	LBCS																				
Branch if = Zero	BEQ	Z = 0				O						O			O						
	LBEQ																				
Branch if Not Equal Zero	BNE	Z ≠ 0				O						O			O						
	LBNE																				
Branch if ≥ Zero	BGE	N ⊕ V = 0				O						O			O						
	LBGE																				
Branch if > Zero	BGT	Z + (N ⊕ V) = 0				O						O			O						
	LBGT																				
Branch if < Zero	BLT	N ⊕ V = 1				O						O			O						
	LBLT																				
Branch if ≤ Zero	BLE	Z + (N ⊕ V) = 1				O						O			O						
	LBLE																				
Branch if Higher	BHI	C + Z = 0				O						O			O						
	LBHI																				
Branch if Lower or Same	BLS	C + Z = 1				O						O			O						
	LBLS																				
Branch if Plus	BPL	N = 0				O						O			O						
	LBPL																				
Branch if Minus	BMI	N = 1				O						O			O						
	LBMI																				
Branch if overflow Clear	BVC	V = 0				O						O			O						
	LBVC																				
Branch if Overflow Set	BVS	V = 1				O						O			O						
	LBVS																				
Branch if Half Carry Clear	BHCC	H = 0														O					
Branch if Half Carry Set	BHCS	H = 1														O					
Branch if Interrup Mask Clear	BMC	I = 0														O					
Branch if Interrupt Mask Set	BMS	I = 1														O					
Branch if Interrupt Line High	BIH	INT = 1														O					
Branch if Interrupt Line Low	BIL	INT = 0														O					
Branch if Bit n of M Clear	BRCLR n	M (n) = 0														O					
Branch if Bit n of M Set	BRSET n	M (n) = 1														O					
Branch Always	BRA											O			O				O		O
	LBRA																				O
Branch Never	BRN															O			O		O
	LBRN																				O
Branch to Subroutine	BSR											O			O				O		O
	LBSR																				O
Jump	JMP								O O			O O			O O O			O O O O			O O O O

(to be continued)

Operations	Mnemonic	Branch Test	HD6800 HD6802				HD6801 HD6803				HD6805				HD6809								
			IMMED	DIRECT	INDEX	EXTND	IMPL	IMMED	DIRECT	INDEX	EXTND	IMPL	IMMED	DIRECT	INDEX	EXTND	IMPL	IMMED	DIRECT	INDEX	EXTND	IMPL	RELATIVE
Jump to Subroutine	JSR				O	O			O	O	O			O	O	O		O	O	O	O	O	
Return from Subroutine	RTS				O				O	O				O	O	O		O	O	O	O	O	
No Operation	NOP				O				O		O			O	O	O		O	O	O	O	O	
Software Interrupt	SWI				O				O		O			O	O	O		O	O	O	O	O	
	SWI2																						
	SWI3																						
Return from Interrupt	RTI					O				O		O			O	O		O					
Wait	WAI				O				O		O			O	O	O		O	O	O	O	O	
	CWAI																						
Synchronize to Interrupt	SYNC																O	O	O	O	O	O	

#### ■ INDEX REGISTER AND STACK POINTER INSTRUCTIONS

Operations	Mnemonic	Boolean/ Arithmetic Operation	HD6800 HD6802				HD6801 HD6803				HD6805				HD6809							
			IMMED	DIRECT	INDEX	EXTND	IMPL	IMMED	DIRECT	INDEX	EXTND	IMPL	IMMED	DIRECT	INDEX	EXTND	IMPL	IMMED	DIRECT	INDEX	EXTND	IMPL
Increment	INX	X + 1 → X					O	O			O	O										
Decrement	DEX	X - 1 → X			O						O	O										
ADD with B	ABX	B + X → X									O											O
Clear	CLRX	0 → X									O	O										
Negate	NEGX	00 - X → X									O	O										
Complement 1's	COMX	FF - X → X									O	O										
Shift Left Arithmetic	ASLX																					O
Shift Right Arithmetic	ASRX																					O
Shift Right Logical	LSRX																					O
Rotate Left	ROLX																					O
Rotate Right	RORX																					O
Test	TSTX	X - 00																				O
Compare	CPX	X - M:M+1	O	O	O	O		O	O	O	O		O	O	O	O		O	O	O	O	O
	CMPX	X - M:M+1																				
Load	LDX	M:M+1 → X	O	O	O	O		O	O	O	O		O	O	O	O		O	O	O	O	O
	LDY	M:M+1 → Y																				
Store	STX	X → M:M+1	O	O	O			O	O	O			O	O	O			O	O	O	O	O
	STY	Y → M:M+1																				
Load effective Address	LEAX	Effective address → X																				O
	LEAY	Effective address → Y																				O
Push	PSHX	X → Ms									O											
Pull	PULX	Ms → X									O											

(to be continued)

Transfer X, A	TAX	A → X						O	
	TXA	X → A						O	
Transfer X, S	TSX	S → X		O			O		
	TXS	X → S		O			O		
Increment	INS	S + 1 → S		O			O		
Decrement	DES	S - 1 → S		O			O		
Reset	RSP	\$7F → S						O	
Compare	CMPS	S - M : M+1						O O O O O	
	CMPU	U - M : M+1						O O O O O	
Load	LDS	M : M+1 → S	O O O O	O O O O				O O O O O	
	LDU	M : M+1 → U						O O O O O	
Store	STS	S → M : M+1	O O O	O O O				O O O O	
	STU	U → M : M+1						O O O O	
Load effective Address	LEAS	Effective Address → S						O	
	LEAU	Effective Address → U						O	

## ■ CONDITION CODE REGISTER INSTRUCTIONS

Operations	Mnemonic	Boolean Operation	HD6800 HD6802				HD6801 HD6803				HD6805				HD6809				
			IMMED	DIRECT	INDEX	EXTND	IMPL	RELATIVE	IMMED	DIRECT	INDEX	EXTND	IMPL	RELATIVE	IMMED	DIRECT	INDEX	EXTND	IMPL
Clear Carry	CLC	0 → C			O					O						O			
Clear Interrupt Mask	CLI	0 → I			O					O						O			
Clear InterruptMask	CLV	0 → V			O					O						O			
Set Carry	SEC	1 → C			O					O						O			
Set InterruptMask	SEI	1 → I			O					O						O			
Set Overflow	SEV	1 → V			O					O						O			
Transfer A, CC	TAP	A → CC			O					O						O			
	TPA	CC → A			O					O						O			
And CC	ANDCC	CC + imm → CC														O			
Or CC	ORCC	CC + imm → CC														O			

### LEGEND

A	Accumulator A	H	Half Carry from 3 bit
B	Accumulator B	I	Interrupt Mask
X	Index Register	N	Negative
Y	Index Register (6809 only)	Z	Zero
S	Stack Pointer	V	Overflow
U	User Stack Pointer (6809 only)	C	Carry Bit from 7 bit
→	Transfer into	MSB	Most Signification bit
+	Arithmetic Plus	LSB	Least Signification bit
-	Arithmetic Minus	IMMED	Immediate
·	Boolean AND	DIRECT	Direct
+	Boolean Inclusive OR	INDEX	Indexed
⊕	Boolean Exclusive OR	EXTND	Extended
:	Connection Area	EXT INDIRECT	Extended indirect
M	Complement of M	IMPL	Implied (Inherent, Accumulator)
Ms	Stack area pointed by S.	RELATIVE	Relative
Mu	Stack area pointed by U.		
CC	Condition Code Register		

# MEMORIES

## ■ MOS MEMORIES

### ● SELECTION GUIDE TO MOS RAMs

Mode	Total Bit	Type No.	Process	Organiza-tion (word × bit)	Access Time (ns) max.	Cycle Time (ns) min.	Supply Voltage (V)	Power Dissipa-tion (mW)	Package**				Cross-refer-ence
									Pin No.	C	G	P	
Static	4k-bit	HM472114A-1	NMOS	1024 × 4	150	150	+ 5	200	18	●	●		
		HM472114A-2			200	200				●	●	2114L-2	
		HM472114-3			300	300				●	●	2114L-3	
		HM472114-4			450	450				●	●	2114L-4	
		HM4334-3	CMOS	1024 × 4	300	460	+ 5	20	18			HM-6514	
		HM4334-4			450	640						●	
		HM6148	CMOS	1024 × 4	70	70	+ 5	200	18			2148	
		HM6148-6			85	85						2148-6	
		HM6148L			70	70						●	
		HM6148L-6			85	85						●	
		HM4315	CMOS	4096 × 1	450	640	+ 5	20	18			●	
		HM6147	CMOS	4096 × 1	70	70	+ 5	75	18	●	●	2147	
		HM6147-3			55	55				●	●	2147-3	
		HM6147L			70	70						●	
		HM6147L-3			55	55						●	
		HM6116-2	CMOS	2048 × 8	120	120	+ 5	180	24			●	
		HM6116-3			150	150						●	
		HM6116-4			200	200						●	
		HM6116L-2			120	120	+ 5	160	16			●	
		HM6116L-3			150	150						●	
		HM6116L-4			200	200						●	
Dynamic	16k-bit	HM4716A-1	NMOS	16384 × 1	120	320	+12, + 5, - 5	350	16	●	●		
		HM4716A-2			150	320				●	●	MK4116-2	
		HM4716A-3			200	375				●	●	MK4116-3	
		HM4716A-4			250	410				●	●	MK4116-4	
		HM4816	NMOS	16384 × 1	100	200	+ 5	250	16	●			
	64k-bit	HM4864-2*	NMOS	65536 × 1	150	270	+ 5	170	16	●			
		HM4864-3*			200	335				●			

### ● SELECTION GUIDE TO MOS ROMs

Program	Total Bit	Type No.	Process	Organiza-tion (word × bit)	Access Time (ns) max.	Supply Voltage (V)	Power Dissipa-tion (mW)	Package**				Cross-refer-ence
								Pin No.	C	G	P	
Mask	16k-bit	HN462316E	NMOS	2048 × 8	350	+ 5	350	24			●	2316E
		HN46332		4096 × 8	350			250	24		●	
		HN48364		8192 × 8	350			225	24		●	
	64k-bit	HN462716		2048 × 8	450			310	24	●	●	2716
UV Erasable & Electrically	32k-bit	HN462532		4096 × 8	450		450	24			●	TMS2532
		HN462732		2048 × 8	350			160	24		●	2732
	Electrically Erasable & Programmable	16k-bit	HN48016								●	

\* Preliminary

\*\* The package codes of C, G, and P are applied to the package materials as follows.

C : Ceramic with Lid, G : Glass-sealed Ceramic, P : Plastic

## ■ BIPOLAR MEMORIES

### ● SELECTION GUIDE TO BIPOLAR RAMs

Level	Total Bit	Type No.	Organiza-tion (word × bit)	Output	Access Time (ns) max.	Supply Voltage (V)	Power Dissipa-tion (mW/bit)	Package **				Cross-refer-ence
								Pin No.	C	G	P	
ECL	256-bit	HM2105	256 × 1	Open Emitter	35	-5.2	1.8	16	•	•	F10410	
		HM2106			15		1.8		•	•		
		HM10414*			10		2.8		•	•	F10414	
		HM10414-1*			8		2.8		•	•		
	1k-bit	HM10422*	256 × 4		10	-4.5	0.8	24	•	•	F10422	
		HM100422*			10		0.8	16	•	•	F100422	
		HM2110			35	-5.2	0.5		•	•	F10415	
		HM2110-1			25		0.5		•	•	F10415A	
		HM2110-2			20		0.5	16	•	•		
		HM2112	1024 × 1		10		0.8		•	•		
		HM2112-1			8		0.8		•	•		
	4k-bit	HM10470*			25		0.2	18	•	•	F10470	
		HM10470-1*			15		0.2	•	•			
TTL	256-bit	HM2504	256 × 1	Open Collector	55	+ 5	1.8	16	•	•	93411	
		HM2504-1			45		0.5	16	•	•	93411A	
	1k-bit	HM2510	1024 × 1		70		0.5	16	•	•	93415	
		HM2510-1			45		0.5	16	•	•	93415A	
		HM2510-2			35		0.5	16	•	•		
		HM2511			70		0.5	16	•	•		
		HM2511-1			45		0.5	16	•	•	93425	

### ● SELECTION GUIDE TO BIPOLAR PROM

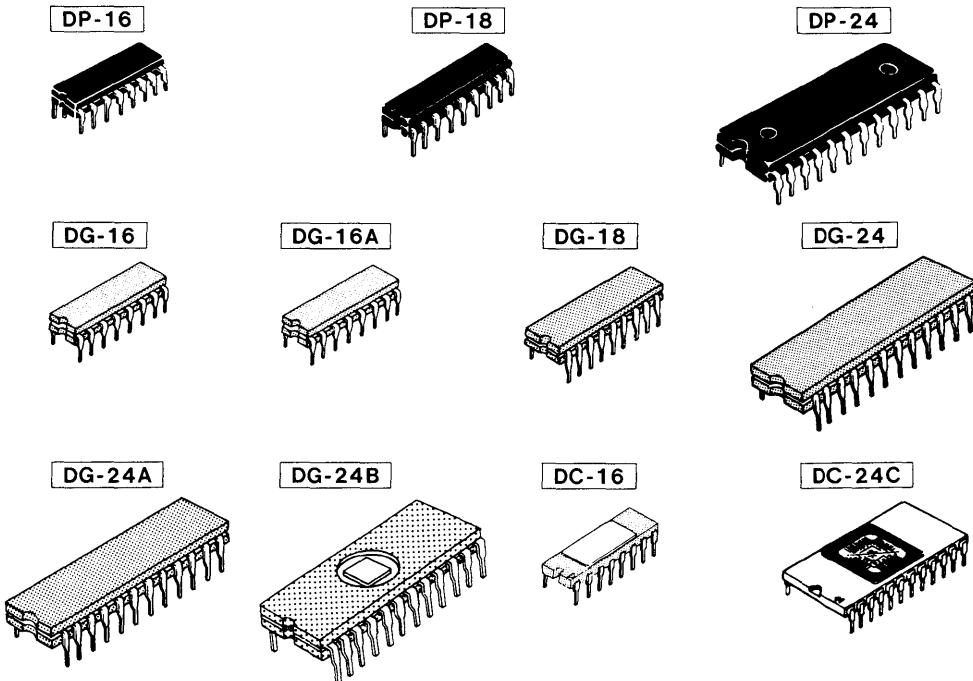
Level	Total Bit	Type No.	Organiza-tion (word × bit)	Output	Access Time (ns) max.	Supply Voltage (V)	Power Dissipa-tion (mW/bit)	Package **				Cross-refer-ence	
								Pin No.	C	G	P		
TTL	4k-bit	HN25044	1024 × 4	O/C	50	+ 5	500	18	•	•	82S136		
		HN25045			3-S				•	•	82S137		
	8k-bit	HN25084*	2048 × 4		60		600	18	•	•	82S184		
		HN25085*			3-S				•	•	82S185		
		HN25088*	1024 × 8	O/C	60		600	24	•	•	82S180		
		HN25089*							•	•	82S181		
		•							•				

\* Preliminary

\*\* The package codes of C, G, and P are applied to the package materials as follows.

C : Ceramic with Lid, G : Glass-sealed Ceramic, P : Plastic

■ OUTLINE



● APPLICABLE ICs

DP-16	HM4716AP-1, HM4716AP-2, HM4716AP-3, HM4716AP-4
DP-18	HM472114AP-1, HM472114AP-2, HM472114P-3, HM472114P-4, HM4334P-3, HM4334P-4, HM6148P, HM6148P-6, HM6148LP, HM6148LP-6, HM4315P, HM6147P, HM6147P-3, HM6147LP, HM6147LP-3
DP-24	HM6116P-2, HM6116P-3, HM6116P-4, HM6116LP-2, HM6116LP-3, HM6116LP-4, HN462316EP, HN46332P, HN48364P, HN48016P
DG-16	HM2105, HM2106, HM10414, HM10414-1, HM2504, HM2504-1
DG-16A	HM4716A-1, HM4716A-2, HM4716A-3, HM4716A-4, HM2110, HM2110-1, HM2110-2, HM2112, HM2112-1, HM2510, HM2510-1, HM2510-2, HM2511, HM2511-1
DG-18	HM472114A-1, HM472114A-2, HM472114-3, HM472114-4, HM6147, HM6147-3, HM10470, HM10470-1, HN25044, HN25045, HN25084, HN25085
DG-24	HN25088, HN25089
DG-24A	HM10422, HM100422
DG-24B	HN462716G
DC-16	HM4816, HM4864-2, HM4864-3
DC-24C	HN462716, HN462532, HN462732

# LINEAR IC & INTERFACE CIRCUITS

## ■ LINEAR ICs

	Functions	Type No.	Package Code					Cross-reference
			M	P	PS	G	GS	
Operational Amplifiers	General Purpose	HA17741			DP-8		DG-8	Fairchild μA741C
	High Speed	HA17715	T-100					Fairchild μA715C
	Dual	HA17458			DP-8		DG-8	NS LM1458
		HA17747		DP-14		DG-14		Fairchild μA747C
		HA17904			DP-8		DG-8	NS LM2904
	Quad.	HA17301		DP-14		DG-14		Motorola MC3301
		HA17902		DP-14		DG-14		NS LM2902
Voltage Comparators	Single	HA1813			DP-8			
	Universal	HA1812			DP-8		DG-8	
	Dual	HA17903			DP-8		DG-8	NS LM2903
		HA1807				DG-14		
	Quad.	HA17901		DP-14		DG-14		NS LM2901
Voltage Regulators	Variable	2~37V, 150mA	HA17723				DG-14	Fairchild μA723C
	Fixed	5V, 1A	HA17805	T-220AB				Fairchild μA7805C
		6V, 1A	HA17806	T-220AB				Fairchild μA7806C
		7V, 1A	HA17807	T-220AB				
		8V, 1A	HA17808	T-220AB				Fairchild μA7808C
		12V, 1A	HA17812	T-220AB				Fairchild μA7812C
		15V, 1A	HA17815	T-220AB				Fairchild μA7815C
		18V, 1A	HA17818	T-220AB				Fairchild μA7818C
		24V, 1A	HA17824	T-220AB				Fairchild μA7824C
		5V, 0.5A	HA178M05	T-220AB				Fairchild μA78M05C
		6V, 0.5A	HA178M06	T-220AB				Fairchild μA78M06C
		7V, 0.5A	HA178M07	T-220AB				
		8V, 0.5A	HA178M08	T-220AB				Fairchild μA78M08C
		12V, 0.5A	HA178M12	T-220AB				Fairchild μA78M12C
		15V, 0.5A	HA178M15	T-220AB				Fairchild μA78M15C
		18V, 0.5A	HA178M18	T-220AB				Fairchild μA78M18C
		20V, 0.5A	HA178M20	T-220AB				Fairchild μA78M20C
		24V, 0.5A	HA178M24	T-220AB				Fairchild μA78M24C
A/D, D/A Converters	Switching Regulator Controller	HA17524		DP-16		DG-16		Silicon General SG3524
	8-bit Double Integral Type A/D	HA16613		DP-28				
	8-bit D/A	HA17408		DP-16		DG-16		AMD AM1408
Other Function	Differential Video Amp.	HA17733	T-100					Fairchild μA733C
	5 Transistor Arrays	HA1127				DG-14		RCA CA3045
	Precision Timers	HA17555			DP-8		DG-8	Signetics NE555
	Monostable Multivibrators	HA1607			DP-8			
	Micromotor Speed Controller	HA16503		DP-14				
	Light-measurement Amp. for Camera	HA16506		DP-14				
		HA16564		DP-14				
	Coin Sensor	HA16603		DP-16				
	Electric Leakage Breaker	HA16604		SP-8				
	Burner Controller	HA16605W		DP-20				

## ■ INTERFACE CIRCUITS

Functions			Type	Package Code		Cross-reference
				P	G	
Line Driver/ Receiver	Driver	Dual	HD75109	DP-14	DG-14	Texas SN75109
			HD75110	DP-14	DG-14	Texas SN75110
		Triple	HD2904		DG-16	
		Quad.	HD75188	DP-14	DG-14	Texas SN75188
	Receiver	Dual	HD75107A	DP-14	DG-14	Texas SN75107A
			HD75108A	DP-14	DG-14	Texas SN75108A
			HD2905		DG-16	
			HD2915		DG-16	
			HD75154	DP-16	DG-16	Texas SN75154
		Quad.	HD75189	DP-14	DG-14	Texas SN75189
Peripheral Driver	Dual NAND + NPN Transistor		HD75450A	DP-14	DG-14	Texas SN75450A
	Dual AND		HD75451A	DP-8	DG-8	Texas SN75451A
	Dual NAND		HD75452	DP-8	DG-8	Texas SN75452
	Dual OR		HD75453	DP-8	DG-8	Texas SN75453
	Dual NOR		HD75454	DP-8	DG-8	Texas SN75454
Memory Support	Core Memory	Dual Sense Amplifier	HD1902		DG-16	Texas SN7524
	IC Memory	Quad. TTL-MOS Clock Driver	HD2912		DG-16	
		Quad. TTL-MOS Clock Driver	HD2916		DG-16A	
		Quad. ECL-TTL Driver	HD2923		DG-16A	
Other	Printer Driver		HD2919	DP-16		

## ■ OUTLINE

DP-8



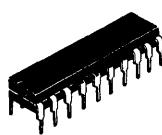
DP-14



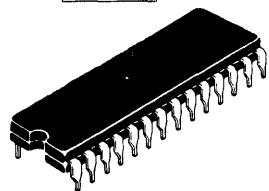
DP-16



DP-20



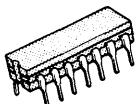
DP-28



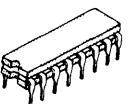
DG-8



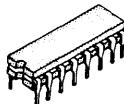
DG-14



DG-16



DG-16A



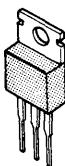
T-100



SP-8



T-220AB



# TTL HD74/HD74S/HD74LS SERIES

## ■ PERFORMANCE (per gate)

Performance	HD74 Series	HD74S Series	HD74LS Series
Propagation Delay Time	10ns	3ns	10ns
Power Dissipation	10mW	20mW	2mW
Speed-Power Product	100pJ	60pJ	20pJ

## ■ MAIN CHARACTERISTICS ( $T_a = -20 \sim +75^\circ C$ )

Parameter	Series		HD74 Series		HD74S Series		HD74LS Series	
	min.	max.	min.	max.	min.	max.	min.	max.
$V_{OL}$ ( $I_{OL}$ max.)	—	0.4V	—	0.5V	—	0.5V	—	0.5V
$V_{OH}$ ( $I_{OH} = -400\mu A$ )	2.4V	—	2.7V	—	2.7V	—	—	—
$V_{IL}$	—	0.8V	—	0.8V	—	0.8V	—	0.8V
$V_{IH}$	2V	—	2V	—	2V	—	—	—
$I_{IL}$	—	-1.6mA	—	-2mA	—	-0.4mA	—	—
$I_{IH}$ ( $V_{IH}$ min.)	—	40 $\mu A$	—	50 $\mu A$	—	20 $\mu A$	—	—

## ■ SELECTION GUIDE

### • NAND/NOR/AND/OR GATES

Function	HD74 Series	HD74S Series	HD74LS Series
Quad. 2-input Positive NAND Gates	00	00	00
Quad. 2-input Positive NAND Gates (with Open Collector Output)	01	—	01
Quad. 2-input Positive NOR Gates	02	02	02
Quad. 2-input Positive NAND Gates (with Open Collector Output)	03	03	03
Hex Inverters	04	04	04
Hex Inverters (with Open Collector Output)	05	05	05
Hex Inverter Buffers/Drivers (with Open Collector High-voltage Output)	06	—	—
Hex Buffers/Drivers (with Open Collector High-voltage Output)	07	—	—
Quad. 2-input Positive AND Gates	08	—	08
Quad. 2-input Positive AND Gates (with Open Collector Output)	09	—	09
Triple 3-input Positive NAND Gates	10	10	10
Triple 3-input Positive AND Gates	—	11	11
Triple 3-input Positive NAND Gates (with Open Collector Output)	12	12	12
Dual 4-input Schmitt NAND Gates	13	—	13
Hex Schmitt-trigger Inverters	14	—	14
Triple 3-input Positive AND Gates (with Open Collector Output)	—	15	15
Hex Inverter Buffers/Drivers (with Open Collector High-voltage Output)	16	—	—
Hex Buffers/Drivers (with Open Collector High-voltage Output)	17	—	—
Dual 4-input Positive NAND Gates	20	20	20
Dual 4-input Positive AND Gates	—	—	21
Dual 4-input Positive NAND Gates (with Open Collector Output)	22	22	22
Expandable Dual 4-input Positive NOR Gates (with Strobe)	23	—	—
Dual 4-input Positive NOR Gates	25	—	—
Quad. 2-input High-voltage Interface NAND Gates	26	—	26
Triple 3-input Positive NOR Gates	27	—	27
8-input Positive NAND Gate	30	—	30
Quad. 2-input Positive OR Gates	32	—	32
Quad. 2-input Positive NAND Buffers	37	—	37
Quad. 2-input Positive NAND Buffers (with Open Collector Output)	38	—	38
Dual 4-input Positive NAND Buffers	40	40	40
Quad. Bus Buffer Gates with 3-state Output (Inverting)	125	—	125A
Quad. Bus Buffer Gates with 3-state Output (Noninverting)	126	—	126A
Quad. 2-input Positive NAND Schmitt Triggers	132	—	132
13-input Positive NAND Gate	—	133	—
12-input Positive NAND Gate (with 3-state Out.)	—	134	—

(to be continued)

Function	HD74 Series	HD74S Series	HD74LS Series
Dual 4-input Positive NAND Line Drivers	—	140	—
Octal Bus Transceivers (with Noninverted 3-state Output)	—	—	245
Hex Bus Buffers/Drivers (with 3-state Output)	—	—	365A
Hex Bus Buffers/Drivers (with 3-state Output)	—	—	366A
Hex Bus Buffers/Drivers (with 3-state Output)	—	—	367A
Hex Bus Buffers/Drivers (with 3-state Output)	—	—	368A

#### • AND-OR-INVERT GATES

Function	HD74 Series	HD74S Series	HD74LS Series
Expandable Dual 2-wide 2-input AND-OR-INVERT Gates	50	—	—
Dual 2-wide 2-input AND-OR-INVERT Gates	51	—	51
Expandable 4-wide 2-input AND-OR-INVERT Gate	53	—	—
4-wide 2-input AND-OR-INVERT Gate	54	—	54
2-wide 4-input AND-OR-INVERT Gate	—	—	55
4-2-3-2-input AND-OR-INVERT Gate	—	64	—
4-2-3-2-input AND-OR-INVERT Gate (with Open Collector Output)	—	65	—

#### • EXPANDER

Function	HD74 Series	HD74S Series	HD74LS Series
Dual 4-input Expanders	60	—	—

#### • FLIP FLOPS

Function	HD74 Series	HD74S Series	HD74LS Series
J-K Master-Slave Flip Flop (AND Inputs)	72	—	—
Dual J-K Flip Flops	73	—	73
Dual D-type Edge-triggered Flip Flops	74	74	74A
Dual J-K Flip Flops (with PR and CLR)	76	—	76
Dual J-K Flip Flops (with PR, Common CLR, and Common CK)	—	—	78
Dual J-K Flip Flops	107	—	107
Dual J- $\bar{K}$ Positive Edge-triggered Flip Flops (With PR and CLR)	—	—	109A
Dual J-K Negative-edge-triggered Flip Flops (with PR and CLR)	—	112	112
Dual J-K Negative-edge-triggered Flip Flops (with PR)	—	113	113
Dual J-K Negative-edge-triggered Flip Flops (with PR, Common CLR, and Common CK)	—	114	114
Monostable Multivibrator	121	—	—
Retriggerable Monostable Multivibrator	—	—	122
Dual Retriggerable Monostable Multivibrators	123	—	123
Hex D-type Flip Flops (with CLR)	174	174	174
Quad. D-type Flip Flops (with CLR)	175	175	175
Dual Monostable Multivibrators (with Schmitt Trigger)	221	—	221

#### • COUNTERS

Function	HD74 Series	HD74S Series	HD74LS Series
Decade Counter	90A	—	90
Divide-by-Twelve Counter	92A	—	92
4-bit Binary Counter	93A	—	93
Presettable Decade Counter/Latch	176	—	—
4-bit Binary Counter/Latch	177	—	—
Synchronous Decade Counter	160	—	160
Synchronous 4-bit Binary Counter	161	—	161
Fully Synchronous Decade Counter	162	—	162

Function	HD74 Series	HD74S Series	HD74LS Series
Fully Synchronous 4-bit Binary Counter	163	—	163
Synchronous Decade Decimal Rate Multiplier	167	—	—
Synchronous Decade Up/Down Counter	190	—	190
Synchronous 4-bit Binary Up/Down Counter	191	—	191
Synchronous Decade Up/Down Counter	192	—	192
Synchronous 4-bit Binary Up/Down Counter	193	—	193
Decade Counter	290	—	290
4-bit Binary Counter	293	—	293
Dual 4-bit Decade Counters	—	—	390
Dual 4-bit Binary Counters	—	—	393
Dual 4-bit Decade Counters	—	—	490
Synchronous Decade Up/Down Counter	—	—	668
Synchronous 4-bit Binary Up/Down Counter	—	—	669

• 4-BIT, 5-BIT SHIFT/STORAGE REGISTERS

Function	HD74 Series	HD74S Series	HD74LS Series
4-bit Right-Shift, Left-Shift Register	95A	—	95B
5-bit Shift Register (Dual Parallel-in, Parallel-out)	96	—	96
4-bit D-type Register (with 3-state Output)	173	—	173
4-bit Parallel-in, Parallel-out Bidirectional Shift Register	194	—	194A
4-bit Parallel-in, Parallel-out Shift Register (J-K Inputs for First Stage)	195	—	195A
4-bit Right-shift, Left-shift Register	—	—	295B

• 8-BIT SHIFT REGISTERS

Function	HD74 Series	HD74S Series	HD74LS Series
8-bit Shift Register	91A	—	91
8-bit Parallel-out Shift Register	164	—	164
Parallel-load 8-bit Shift Register	166	—	166
8-bit Parallel-in, Parallel-out Bidirectional Shift Register	198	—	—
8-bit Parallel-in, Parallel-out Shift Register (J-K Inputs for First Stage)	199	—	—
8-bit Universal Shift/Storage Register	—	—	299

• ENCODERS

Function	HD74 Series	HD74S Series	HD74LS Series
10-line-to-4-line Priority Encoder	147	—	—
8-line-to-3-line Priority Encoder	148	—	148

• DECODERS/DEMULTIPLEXERS

Function	HD74 Series	HD74S Series	HD74LS Series
BCD-to-Decimal Decoder	42A	—	42
Excess 3-to-Decimal Decoder	43A	—	—
Excess 3-Gray-to-Decimal Decoder	44A	—	—
3-to-8-line Decoder	—	—	138
Dual 2-to-4-line Decoders/Demultiplexers	—	—	139
4-line-to-16-line Decoder/Demultiplexer	154	—	154
Dual 2-line-to-4-line Decoders/Demultiplexers	155	—	155
Dual 2-line-to-4-line Decoders/Demultiplexers (With Open Collector Output)	156	—	156
4-line-to-16-line Decoder/Demultiplexer (with Open Collector Output)	159	—	—

(to be continued)

● DECODERS/LAMP DRIVERS/BUFFERS

Function	HD74 Series	HD74S Series	HD74LS Series
BCD-to-Decimal Decoder/Driver (with 30V Out.)	45	—	—
BCD-to-Decimal Decoder/Driver (with 15V Out.)	145	—	145
BCD-to-Seven Segment Decoder/Driver (with 30V Output)	46A	—	—
BCD-to-Seven Segment Decoder/Driver (with 15V Output)	47A	—	47
BCD-to-Seven Segment Decoder	—	—	48
BCD-to-Seven Segment Decoder	—	—	49
BCD-to-Decimal Decoder/Driver (with 60V Out.)	141	—	—
BCD-to-Seven Segment Decoder/Driver (with 15V Output)	—	—	247
BCD-to-Seven Segment Decoder/Driver	—	—	248
BCD-to-Seven Segment Decoder/Driver	—	—	249

● LATCHES

Function	HD74 Series	HD74S Series	HD74LS Series
Quad. Bistable Latches	75	—	75
4-bit Bistable Latch	—	—	77
Quad. S-R Latches	279	—	279
8-bit Addressable Latch	—	—	259
4-bit Bistable Latch	—	—	375

● RANDOM ACCESS MEMORIES (less than 256-bit)

Function	HD74 Series	HD74S Series	HD74LS Series
64-bit Random Access Memory (16w by 4b)	89	—	—

● ARITHMETIC ELEMENTS

Function	HD74 Series	HD74S Series	HD74LS Series
4-bit Binary Full Adder	83A	—	83A
4-bit Magnitude Comparator	85	—	85
Quad. 2-input Exclusive-OR Gates	86	86	86
Quad. Exclusive-OR/NOR Gates	—	135	—
Quad. 2-input Exclusive-OR Gates (with Open Collector Output)	136	—	136
8-bit Odd/Even Parity Generator/Checker	180	—	—
4-bit Arithmetic Logic Unit/Function Generator	—	181	181
Look-Ahead Carry Generator (for ALU)	182	182	—
Dual Carry Save Full Adders	H183	—	—
Quad. 2-input Exclusive-NOR Gates (with Open Collector Output)	—	—	266
9-bit Odd/Even Parity Generator/Checker	—	280	280
4-bit Binary Full Adder (with Fast Carry)	283	—	283
Quad. 2-input Exclusive-OR Gates	—	—	386

(to be continued)

● DATA SELECTORS/MULTIPLEXERS

Function	HD74 Series	HD74S Series	HD74LS Series
16-bit Data Selector/Multiplexer	150	—	—
8-bit Data Selector/Multiplexer (with Strobe)	151A	151	151
8-bit Data Selector/Multiplexer	—	—	152
Dual 4-line-to-1-line Data Selectors/Multiplexers	153	—	153
Quad. 2-line-to-1-line Data Selectors/Multiplexers	157	157	157
Quad. 2-line-to-1-line Data Selectors/Multiplexers	—	158	158
8-bit Data Selector/Multiplexer (with Strobe and 3-state Output)	251	251	251
Dual 4-line-to-1-line Data Selectors/Multiplexers (with 3-state Output)	—	—	253
Quad. 2-line-to-1-line Data Selectors/Multiplexers (with 3-state Output)	—	257	257
Quad. 2-line-to-1-line Data Selectors/Multiplexers (with 3-state Output)	—	258	258
Quad. 2-input Multiplexers (with Storage)	—	—	298

● MICROPROCESSOR SUPPORT FUNCTIONS

Function	HD74 Series	HD74S Series	HD74LS Series
Octal Buffers/Line Drivers/Line Receivers (Inverted 3-state Output)	—	—	240
Octal Buffers/Line Drivers/Line Receivers (Noninverted 3-state Output)	—	—	241
Octal Buffers/Line Drivers/Line Receivers (Inverted 3-state Output)	—	—	244

■ OUTLINE

DP-14



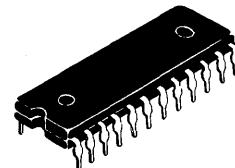
DP-16



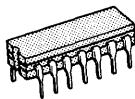
DP-20



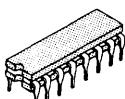
DP-24



DG-14



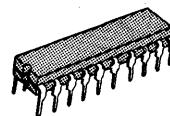
DG-16



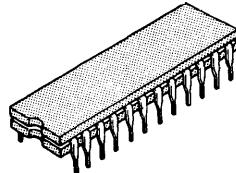
DG-16A



DG-20



DG-24











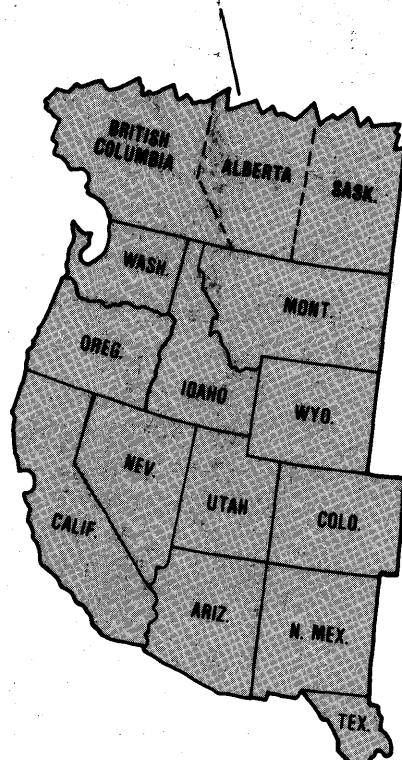


# REGIONAL OFFICES

For further information, contact your Regional Sales Office:

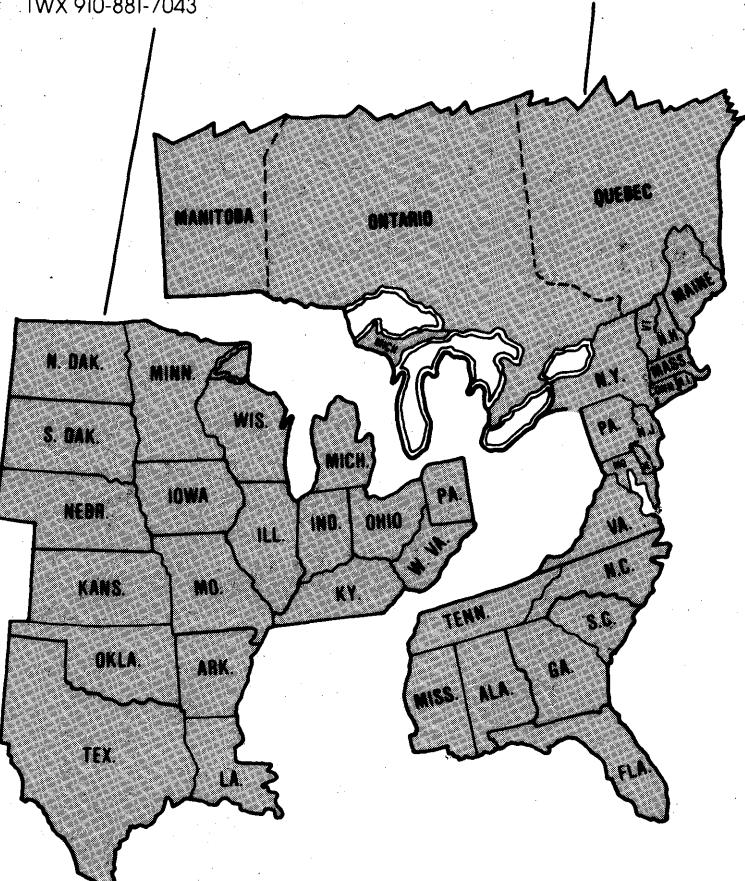
## Western

9700 Reseda Blvd.  
Suite 208  
Northridge, CA 91324  
(213) 701-6606



## Central

6200 Savoy Drive, Suite 704  
Houston, TX 77036  
(713) 974-0534  
TWX 910-881-7043



## Eastern

594 Marrett Road, Suite 22  
Lexington, MA 02173  
(617) 861-1642  
TWX 710-326-1413



# HITACHI

Hitachi America, Ltd., Semiconductor and IC Sales and Service Division  
1800 Bering Drive, San Jose, CA 95112 (408).292-6404

Symbol of Semiconductor Quality, Worldwide