HARRIS
SEMICONDUCTOR
PRODUCTS DIVISION
A DIVISION OF HARRIS CORPORATION

# HM-6100
## CMOS 12 BIT MICROPROCESSOR (CPU)

### Features

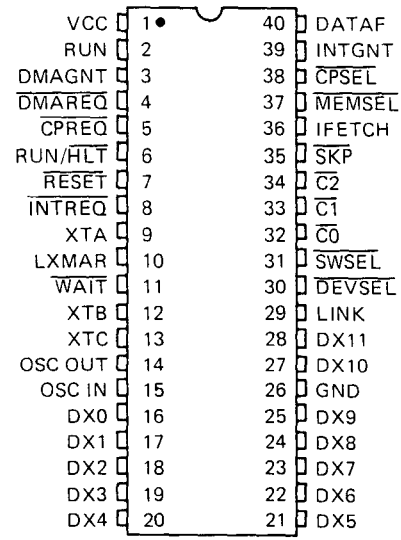
- LOW POWER - TYP. < 5.0μW STANDBY
- SINGLE SUPPLY 4-11 VOLT
- FULL TEMPERATURE RANGE -55°C TO +125°C
- STATIC OPERATION
- SINGLE PHASE CLOCK, ON CHIP CRYSTAL OSC.
- SOFTWARE COMPATIBLE WITH PDP-8/E
- 12-BIT DATA WORD
- OVER 90 SINGLE WORD INSTRUCTIONS
- RELOCATABLE MEMORY ORGANIZATION
- BASIC ADDRESSING TO 4K 12 BIT WORDS
- PROVISION FOR DEDICATED CONTROL PANEL
- 128 GENERAL PURPOSE REGISTERS
- 8 AUTOINDEXING REGISTERS
- FLEXIBLE PROGRAMMED I/O TRANSFERS
- VECTORED INTERRUPT CAPABILITY

### Description

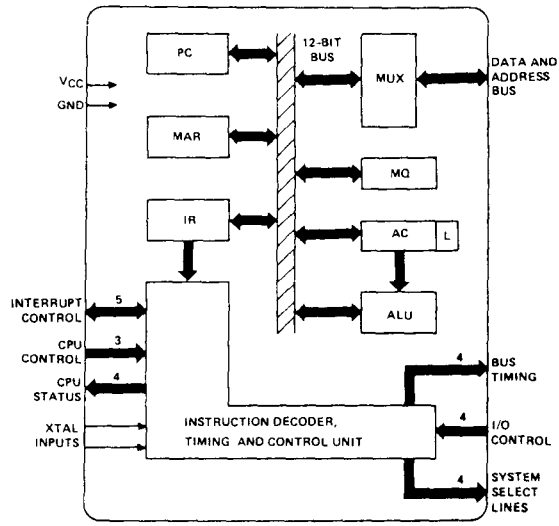
The HM-6100 is a single address, fixed word length, parallel transfer microprocessor using 12-bit two's complement arithmetic. It is a general purpose processor which recognizes the instruction set of Digital Equipment Corporation's PDP-8/E Minicomputer.

Standard features include indirect addressing and facilities for instruction skipping, program interrupts as a function of input/output device conditions, and auto-restart. Five 12-bit registers are used to control microprocessor operations, address memory, perform arithmetic or logical operations, and store data. The device design is optimized to minimize the number of external components required for interfacing with standard memory and peripheral devices.

### Pinout




### Functional Diagram



## ABSOLUTE MAXIMUM RATINGS

| | |
|---|---|
| Supply Voltage | +12.0V |
| Input or Output Voltage Applied | Gnd -0.3V to VCC +0.3V |
| Storage Temperature Range | -65°C to 150°C |
| Operating Temperature Range | |
| Industrial HM-6100A-9 | -40°C to +85°C |
| Military HM-6100A-2 | -55°C to +125°C |

## ELECTRICAL CHARACTERISTICS

VCC = 10.0 ±0.5 Volts, $T_A$ = Industrial or Military

D.C.

| SYMBOL | PARAMETER | MIN | TYP | MAX | UNITS | TEST CONDITIONS |
|---|---|---|---|---|---|---|
| VIH | Logical "1" Input Voltage | 70% VCC | | | V | |
| VIHC | Logical "1" Osc. Input Voltage | VCC-.5 | | | V | |
| VIL | Logical "0" Input Voltage | | | 20% VCC | V | |
| VILC | Logical "0" Osc. Input Voltage | | | GND +.5 | V | |
| IIL | Input Leakage (1) | -1.0 | | +1.0 | μA | $0V \leq VIN \leq VCC$ |
| VOH | Logical "1" Output Volt. (2) | VCC-2.0 | | | V | IOH = -0.2mA |
| VOL | Logical "0" Output Volt. (2) | | | 0.45 | V | IOL = 2.0 |
| IO | Output Leakage | -1.0 | | 1.0 | μA | $0V \leq VO \leq VCC$ |
| ICC1 | Supply Current (Static) | | | 800 | μA | VIN =VCC, Freq. = 0 |
| ICC2 | Supply Current (Operating) | | 1.0 | 10.0 | mA | VCC=10.5V, Freq=2.0MHz |
| CI | Input Capacitance (3) | | 5 | 7 | pF | |
| CO | Output Capacitance (3) | | 8 | 10 | pF | |
| CIO | Input/Output Capacitance (3) | | 8 | 10 | pF | |
| COSC | Oscillator IN/OUT CAP. (3) | | 30 | | pF | |

Notes: (1) Except pin 14 and 15
(2) Except pin 14
(3) Guaranteed and sampled, but not 100% tested.

A.C.

| | | TA = 25°C VCC = 10.0V (1) | | TA = Indust. VCC = 10 ±0.5V | | TA = Military VCC = 10 ±0.5V | | | |
|---|---|---|---|---|---|---|---|---|---|
| SYMBOL | PARAMETER | MIN | MAX | MIN | MAX | MIN | MAX | UNITS | TEST CONDITIONS |
| fMAX | Max Operating Frequency | | 8.0 | | 5.71 | | 5.0 | MHz | CL = 50pF |
| TS | Major State Time | 250 | | 350 | | 400 | | ns | See Timing Diagram |
| TLX | LXMAR Pulse Width | 110 | | 150 | | 170 | | ns | |
| TAS | Address Setup Time | 30 | | 55 | | 70 | | ns | |
| TAH | Address Hold Time | 75 | | 60 | | 70 | | ns | |
| TAL | Access Time from LXMAR | 225 | | 295 | | 340 | | ns | |
| TEN | Output Enable (Memory) | 125 | | 185 | | 220 | | ns | |
| TEND | Output Enable (I/O) | 150 | | 250 | | 290 | | ns | |
| TWP | Write Pulse Width | 110 | | 140 | | 160 | | ns | |
| TDS | Data Setup (Memory) | 80 | | 115 | | 140 | | ns | |
| TDSD | Data Setup (I/O) | 85 | | 110 | | 140 | | ns | |
| TDH | Data Hold Time | 50 | | 60 | | 70 | | ns | |
| TST | Status Signals Valid | | 150 | | 200 | | 250 | ns | |
| TRS | Request Inputs Setup | 0 | | 0 | | 0 | | ns | |
| TRH | Request Inputs Hold | 100 | | 150 | | 200 | | ns | |
| TWS | Wait Setup Time | 0 | | 25 | | 25 | | ns | |
| TWH | Wait Hold Time | 50 | | 50 | | 80 | | ns | |
| TRHS | Run Halt Setup Time | 0 | | 30 | | 30 | | ns | |
| TRHP | Run Halt Pulse Width | 60 | | 60 | | 80 | | ns | |

NOTE 1: All devices guaranteed at worst case limits. Room temperature, 10V data provided for information - not guaranteed.

## ABSOLUTE MAXIMUM RATINGS

| | |
|---|---|
| Supply Voltage | +8.0V |
| Input or Output Voltage Applied | Gnd -0.3V to VCC +0.3V |
| Storage Temperature Range | -65°C to 150°C |
| Operating Temperature Range | |
| Industrial HM-6100-9 | -40°C to +85°C |
| Military HM-6100-2 | -55°C to +125°C |

## ELECTRICAL CHARACTERISTICS

$V_{CC}$ = 5.0 ± 10% Volts, $T_A$ = Industrial or Military

D.C.

| SYMBOL | PARAMETER | MIN | TYP | MAX | UNITS | TEST CONDITIONS |
|---|---|---|---|---|---|---|
| VIH | Logical "1" Input Voltage | 70% VCC | | | V | |
| VIHC | Logical "1" Osc. Input Voltage | VCC-.5 | | | V | |
| VIL | Logical "0" Input Voltage | | | 20% VCC | V | |
| VILC | Logical "0" Osc. Input Voltage | | | GND +.5 | V | |
| IIL | Input Leakage (1) | -1.0 | | +1.0 | µA | $0V \leq V_{IN} \leq V_{CC}$ |
| VOH | Logical "1" Output Volt. (2) | 2.4 | | | V | IOH = -0.2mA |
| VOL | Logical "0" Output Volt. (2) | | | 0.45 | V | IOL = 2.0mA |
| IO | Output Leakage | -1.0 | | +1.0 | µA | $0V \leq V_O \leq V_{CC}$ |
| ICC1 | Supply Current (Static) | | | 400 | µA | VIN = VCC, Freq. = 0 |
| ICC2 | Supply Current (Operating) | | | 2.5 | mA | VCC=5.5V, Freq=2.0MHz |
| CI | Input Capacitance (3) | | 5 | 7 | pF | |
| CO | Output Capacitance (3) | | 8 | 10 | pF | |
| CIO | Input/Output Capacitance (3) | | 8 | 10 | pF | |
| COSC | Oscillator IN/OUT CAP. (3) | | 30 | | pF | |

Notes: (1) Except pin 14 and 15
(2) Except pin 14
(3) Guaranteed and sampled, but not 100% tested.

A.C.

| | | TA = 25°C VCC = 5.0V (1) | | TA = Indust. VCC = 5.0 ± 10%V | | TA = Military VCC = 5.0 ± 10%V | | | |
|---|---|---|---|---|---|---|---|---|---|
| SYMBOL | PARAMETER | MIN | MAX | MIN | MAX | MIN | MAX | UNITS | TEST CONDITIONS |
| fMAX | Max Operating Frequency | | 4.0 | | 3.33 | | 2.5 | MHz | CL = 50pF |
| TS | Major State Time | 500 | | 600 | | 800 | | ns | See Timing Diagram |
| TLX | LXMAR Pulse Width | 220 | | 230 | | 355 | | ns | |
| TAS | Address Setup Time | 80 | | 85 | | 200 | | ns | |
| TAH | Address Hold Time | 150 | | 125 | | 175 | | ns | |
| TAL | Access Time from LXMAR | 450 | | 520 | | 745 | | ns | |
| TEN | Output Enable (Memory) | 250 | | 300 | | 470 | | ns | |
| TEND | Output Enable (I/O) | 300 | | 470 | | 655 | | ns | |
| TWP | Write Pulse Width | 200 | | 235 | | 330 | | ns | |
| TDS | Data Setup (Memory) | 160 | | 135 | | 250 | | ns | |
| TDSD | Data Setup (I/O) | 185 | | 225 | | 350 | | ns | |
| TDH | Data Hold Time | 125 | | 125 | | 170 | | ns | |
| TST | Status Signals Valid | | 250 | | 300 | | 325 | ns | |
| TRS | Request Inputs Setup | 0 | | 0 | | 0 | | ns | |
| TRH | Request Inputs Hold | 200 | | 250 | | 300 | | ns | |
| TWS | Wait Setup Time | 0 | | 50 | | 50 | | ns | |
| TWH | Wait Hold Time | 100 | | 100 | | 150 | | ns | |
| TRHS | Run Halt Setup Time | 0 | | 50 | | 50 | | ns | |
| TRHP | Run Halt Pulse Width | 100 | | 100 | | 150 | | ns | |

NOTE 1: All devices guaranteed at worst case limits. Room temperature, 5V data provided for information - not guaranteed.

## ABSOLUTE MAXIMUM RATINGS

| | |
|---|---|
| Supply Voltage | 8.0V |
| Input or Output Voltage Applied | Gnd -0.3V to VCC +0.3V |
| Storage Temperature Range | -65°C to 150°C |
| Operating Temperature Range | |
| Industrial HM-6100C-9 | -40°C to +85°C |

## ELECTRICAL CHARACTERISTICS

VCC = 5.0 ± 5% Volts, $T_A$ = Industrial

D.C.

| SYMBOL | PARAMETER | MIN | TYP | MAX | UNITS | TEST CONDITIONS |
|---|---|---|---|---|---|---|
| VIH | Logical "1" Input Voltage | 70% VCC | | | V | |
| VIHC | Logical "1" Osc. Input Voltage | VCC-.5 | | | V | |
| VIL | Logical "0" Input Voltage | | | .8 | V | |
| VILC | Logical "0" Osc. Input Voltage | | | GND +.5 | V | |
| IIL | Input Leakage (1) | -10 | | +10 | μA | $0V \leq VIN \leq VCC$ |
| VOH | Logical "1" Output Volt. (2) | 2.4 | | | V | IOH = -0.2mA |
| VOL | Logical "0" Output Volt. (2) | | | 0.45 | V | IOL = 1.6mA |
| IO | Output Leakage | -10 | | +10 | μA | $0V \leq VO \leq VCC$ |
| ICC1 | Supply Current (Static) | | | 600 | μA | VIN = VCC, Freq. = 0 |
| ICC2 | Supply Current (Operating) | | | 5.0 | mA | VCC=5.5V, Freq=2.0MHz |
| CI | Input Capacitance (3) | | 5 | 7 | pF | |
| CO | Output Capacitance (3) | | 8 | 10 | pF | |
| CIO | Input/Output Capacitance (3) | | 8 | 10 | pF | |
| COSC | Oscillator IN/OUT CAP. (3) | | 30 | | pF | |

Notes: (1) Except pin 14 and 15
(2) Except pin 14
(3) Guaranteed and sampled, but not 100% tested.

A.C.

| | | TA = 25°C VCC=5.0V(1) | | TA = Indust. VCC = 5.0 ± 5% | | | |
|---|---|---|---|---|---|---|---|
| SYMBOL | PARAMETER | MIN | MAX | MIN | MAX | UNIT | TEST CONDITION |
| fMAX | Max operating Freq. | | 3.33 | | 2.5 | MHz | CL = 50pF |
| TS | Major State Time | 600 | | 800 | | ns | See Timing Diagram |
| TLX | LXMAR Pulse Width | 270 | | 335 | | ns | |
| TAS | Address Setup Time | 100 | | 120 | | ns | |
| TAH | Address Hold Time | 150 | | 175 | | ns | |
| TAL | Access Time from LXMAR | 500 | | 650 | | ns | |
| TEN | Output Enable (Memory) | 300 | | 400 | | ns | |
| TEND | Output Enable (I/O) | 350 | | 575 | | ns | |
| TWP | Write Pulse Width | 250 | | 320 | | ns | |
| TDS | Data Setup (Memory) | 180 | | 240 | | ns | |
| TDSD | Data Setup (I/O) | 200 | | 275 | | ns | |
| TDH | Data Hold Time | 130 | | 175 | | ns | |
| TST | Status Signals Valid | | 300 | | 350 | ns | |
| TRS | Request Inputs Setup | 0 | | 0 | | ns | |
| TRH | Request Inputs Hold | 100 | | 130 | | ns | |
| TWS | Wait Setup Time | 0 | | 0 | | ns | |
| TWH | Wait Hold Time | 100 | | 130 | | ns | |
| TRHS | Run Halt Setup Time | 0 | | 70 | | ns | |
| TRHP | Run Halt Pulse Width | 100 | | 130 | | ns | |

Note 1: All devices guaranteed at worst case limits. Room temperature, 5V data provided for information - not guaranteed.

## *Timing and State Control*

The HM-6100 generates all the timing and state signals internally. A crystal is used to control the CPU operating frequency. The CPU divides the crystal frequency by two. With a 4MHz crystal, the internal states will be of 500ns duration. The major timing states are described in Figure 1.

T1 For memory reference instructions, a 12-bit address is sent on the DataX, DX, lines. The Load External Address Register, LXMAR, is used to clock an external register to store the address information externally, if required. When executing an Input-Output I/O instruction, the instruction being executed is sent on the DX lines to be stored externally. The external address register then contains the device address and control information.

Various CPU request lines are priority sampled if the next cycle is an Instruction Fetch cycle. Current state of the CPU is available externally.

T2 Memory/Peripheral data is read for an input transfer (READ). $\overline{WAIT}$ controls the transfer duration. If $\overline{WAIT}$ is active during input transfers, the CPU waits in the T2 state. The wait duration is an integral multiple of the crystal frequency - 250ns for 4MHz.

For Memory reference instructions, the Memory Select, $\overline{MEMSEL}$, lines are active. For I/O instruction the $\overline{DEVSEL}$, line is active. Control lines, therefore, distinguish the contents of the external register as memory or device address.

External device sense lines $\overline{C0}$, $\overline{C1}$, $\overline{C2}$, and SKP are sampled if the instruction being executed is an I/O instruction.

Control Panel Memory Select, $\overline{CPSEL}$, and Switch Register Select, $\overline{SWSEL}$, become active low for data transfers between the HM-6100 and Control Panel Memory and the Switch Register, respectively.

T3, T4, T5
ALU operation and internal register transfers.

T6 This state is entered for an output transfer (WRITE). The address is defined during T1. $\overline{WAIT}$ controls the time for which the WRITE data must be maintained.

The following illustrates the timing of the CPU when its operating frequency is low enough that propagation delays can be ignored. It effectively shows the timing of the CPU when it is single clocked.

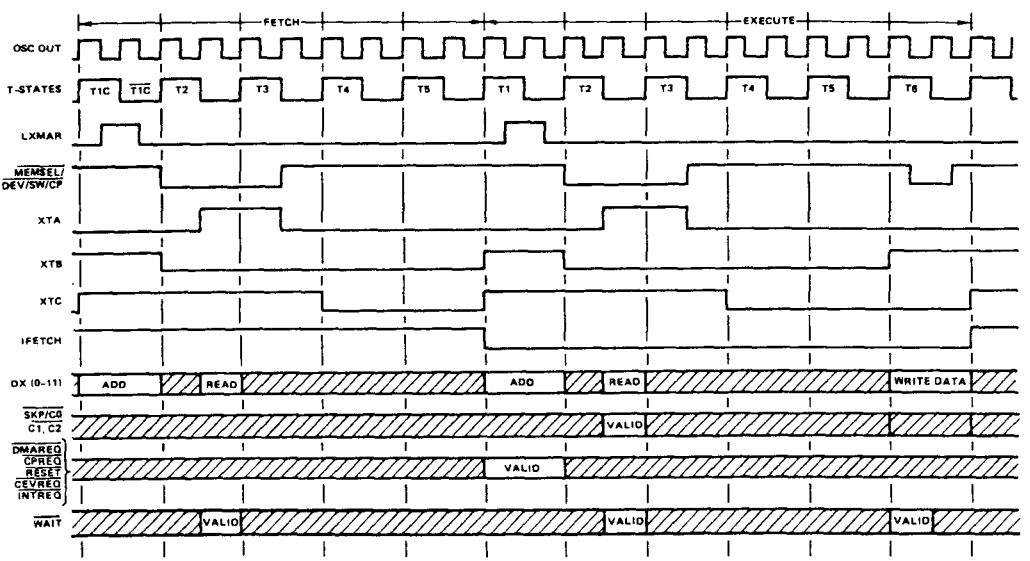



FIGURE 1 – Static Timing

The dynamic or high frequency timing illustrates the propagation delays at specified operating frequencies. (Refer to specifications) It defines the interface requirements for memory and I/O devices on the bus.

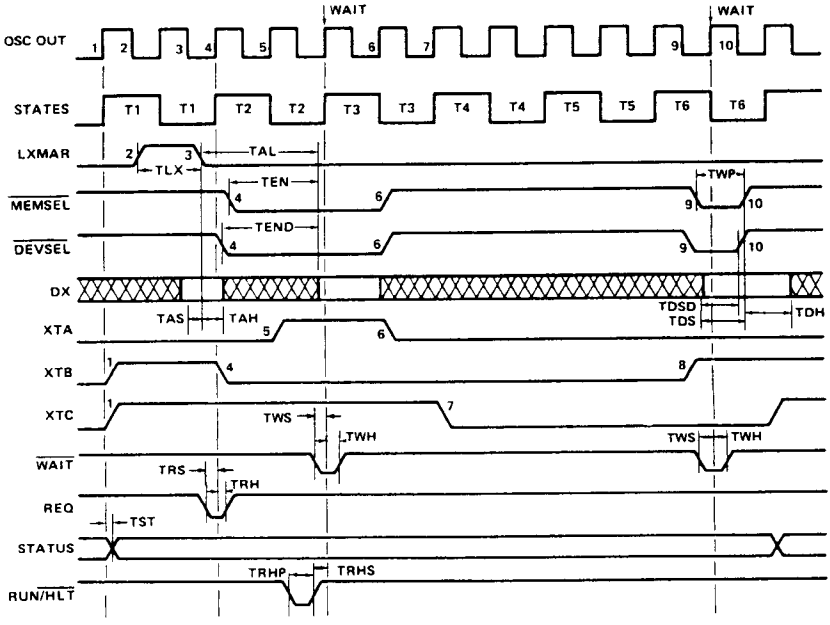



FIGURE 2 – Dynamic Timing

## *Microprocessor Architecture*

The block diagram of the CPU architecture, shown on the front page, consists of the following major functional segments:

- CPU Registers
- Arithmetic and Logic Unit
- Dx-Bus Multiplexer
- Timing and Control Unit

Each one is briefly described below.

### CPU REGISTERS

The CPU consists of five, 12-bit registers, of which three are user programmable; 1) Accumulator (AC), 2) Program Counter (PC), and 3) Multiply Quotient (MQ). The remaining two registers are the Instruction Register (IR) and the Memory Address Register (MAR) which are used exclusively for internal operations. The CPU registers are defined as follows.

### ACCUMULATOR AND LINK (AC/L)

All arithmetic and logical operations are performed in the AC. For any arithmetic operation, the AC data and memory data are combined in the ALU and the result is temporarily stored in the AC. Under software control, the AC can be cleared, set, complemented, incremented, tested or rotated. Using the Operate Microinstructions, a variety of register operate instructions can be derived.

The link is a one-bit extension of the AC. It can be complemented with a carry out of the ALU or cleared, set, complemented, tested and rotated along with the rest of the AC. It also serves as the carry output for two's complement arithmetic.

### MULTIPLY QUOTIENT (MQ)

The MQ register can be used as a temporary storage for the AC. The MQ may be OR'ed with the AC and the result stored in the AC or the contents of the AC and MQ may be swapped. The MQ is used in conjunction with the AC to perform multiplication, division, and double-precision operations.

### PROGRAM COUNTER (PC)

The PC supports both memory and input-output device operations. For memory operations, the PC is controlled exclusively by internal logic and instructions fetched from memory. During an instruction fetch cycle the contents of the PC are transferred to the memory address register (MAR) while the current instruction is being decoded. The PC is then loaded with a new address or simply incremented for the next instruction depending upon the type of instruction. The next instruction obtained from memory is then loaded into the Instruction Register. For example, if the instruction is a JMP X, then the branch address X is loaded into the PC for program controlled branching.

Branching can also be controlled by an external device during input-output operations. This feature allows I/O controlled vectored interrupts.

### MEMORY ADDRESS REGISTER (MAR)

The MAR contains the address of the memory location that is currently selected for memory or I/O read-write operations. It is also used for microprogram control during data transfers to and from memory and peripherals.

### INSTRUCTION REGISTER (IR)

The instruction fetched from memory is held in the IR while being interpreted by the Instruction Decoder. The IR specifies the initial step of the microprogram sequence for each instruction and is also used to store temporary data for microprogram control.

### ARITHMETIC AND LOGIC UNIT (ALU)

The ALU performs 12-bit arithmetic, logical and rotate operations. Its input is derived from the AC and any one of the other CPU registers. The type of operations performed by the ALU include:

| | |
|---|---|
| ADD | Left-right shifts and rotates |
| Logical AND | Increment |
| Logical OR | Complement |
| Test AC | Set/Clear |

### DX-BUS MULTIPLEXER

To keep the CPU pin count to a reasonable 40 and still maintain a 12-bit word structure, the address and data paths are multiplexed by the DX-Bus Multiplexer. It handles data, address and instruction transfers between the CPU and memory or peripheral devices on a time-multiplexed basis.

### TIMING AND CONTROL UNIT

The Timing and Control Unit generates the state and cycle timing signals from a single-phase clock and maintains the proper sequences of events required for any processing task. It also decodes the instruction obtained from the IR and combines the result with various timing signals and external control inputs to provide control and gating signals required by other functional units (both internal and external to the CPU).

---

## *Memory Organization*

The HM-6100 has a basic addressing capacity of 4096 12-bit words. The addressing capacity may be extended to 32K words by Extended Memory Control hardware. Every location has a unique 4 digit octal (12 bit binary) address, $0000_8$ to $7777_8$ ($0000_{10}$ to $4095_{10}$). The Memory is subdivided into 32 PAGES of 128 words each. Memory Pages are numbered sequentially from Page $00_8$, containing addresses $0000$-$0177_8$, to Page $37_8$, containing addresses $7600_8$-$7777_8$. The first 5 bits of a 12-bit MEMORY ADDRESS denote the PAGE NUMBER and the low order 7 bits specify the PAGE ADDRESS of the memory location within the given Page.

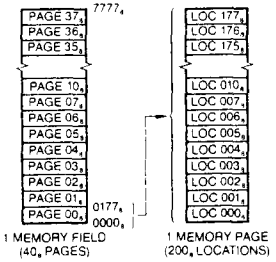



**FIGURE 3 – Memory Organization**

## *Memory and Processor Instructions*

The HM-6100 instructions are 12-bit words stored in memory. The HM-6100 makes no distinction between instruction and data; it can manipulate instructions as stored variables or execute data as instructions. There are three general classes of HM-6100 instructions. They are Memory Reference Instructions (MRI), Operate Instructions (OPR), and Input/Output Transfer Instructions (IOT).

During an instruction fetch cycle, the HM-6100 fetches the instruction pointed to by the PC. The contents of the PC are transferred to the MAR. The PC is incremented by 1. The PC now contains the address of the "current" instruction which must be fetched from memory. Bits 0-4 of the MAR identify the CURRENT PAGE, that is, the Page from which instructions are currently being fetched and bits 5-11 of the MAR identify the location within the Current Page. (PAGE ZERO (0), $0000_8$-$0177_8$, by definition, denotes the first 128 words of memory and is called the Register Page.)

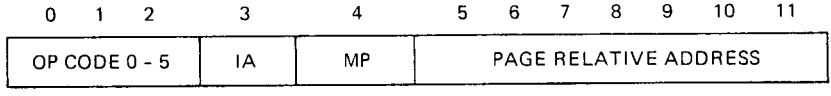
Since the HM-6100 is a static design it can operate at any crystal frequency from 0 to 8MHz. State times required for execution are given for each instruction. Execution time can be calculated from the equation:

$$T = N*(2*(1/F))$$

where N is the number of state times and F is the crystal or input clock frequency.

### MEMORY REFERENCE INSTRUCTIONS (MRI)

The Memory Reference Instructions operate on the contents of a memory location or use the contents of a memory location to operate on the AC or the PC. The first 3 bits of a Memory Reference Instruction specify the operation code, or OPCODE, and the low order 9 bits, the OPERAND address, as shown in Figure 4.




IA: Indirect Addressing 0 = Direct; 1 = Indirect
MP: Memory Page 0 = Register Page; 1 = Current Page

**FIGURE 4 – Memory Reference Instruction Format**

Bits 5 through 11, the PAGE ADDRESS, identify the location of the OPERAND on a given page, but they do not identify the page itself. The page is specified by bit 4, called the CURRENT PAGE OR REGISTER PAGE BIT. If bit 4 is a 0, the page address is interpreted as a location on the Register Page. If bit 4 is a 1, the page address specified is interpreted to be on the Current Page.

By this Method, 256 locations may be directly addressed, 128 on the REGISTER PAGE and 128 on the CURRENT PAGE. Other locations are addressed by using bit 3. When bit 3 is a 0, the operand address is a DIRECT ADDRESS. An INDIRECT ADDRESS (pointer address) identifies the location that contains the desired address (effective address). To address a location that is not directly addressable, not in the REGISTER PAGE or in the CURRENT PAGE, the absolute address of the desired location is stored in one of the 256 directly addressable locations (pointer address). Upon execution, the MRI will operate on the contents of the location identified by the address contained in the pointer location. Note that locations $0010_8$-$0017_8$ in the Register Page are AUTOINDEXED. When these locations are used for index registers their contents are incremented by 1 and restored before they are used as the operand address. These locations are therefore convenient for indexing applications.

Combinations of mode and page bits yield four (4) addressing modes:

- Current Page, Direct
- Current Page, Indirect
- Register Page, Direct
- Register Page, Indirect

A fifth addressing mode results from use of the AUTOINDEX registers:

- Register Page, Autoindexed

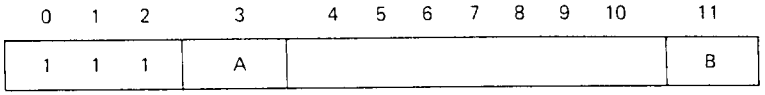
TABLE 1

| MNE-MONIC | OP CODE | NUMBER OF STATES | | | OPERATION |
|---|---|---|---|---|---|
| | | DIRECT | INDIRECT | AUTO-INDEXED | |
| AND | 0XXX | 10 | 15 | 16 | LOGICAL AND: Causes a bit-by-bit boolean AND between the contents of the Accumulator and the contents of the effective address (XXX) specified by the instruction. The result is left in the AC and the data word in the referenced location is not altered. |
| TAD | 1XXX | 10 | 15 | 16 | TWO'S COMPLEMENT ADD: Performs a binary two's complement addition between the specified data word and the contents of the AC; the result is left in the AC. If a carry out occurs, the state of the Link is complemented. If the AC is initially cleared, this instruction acts as a LOAD from memory. |
| ISZ | 2XXX | 16 | 21 | 22 | INCREMENT AND SKIP IF ZERO: The contents of the effective address are incremented by 1 and restored. If the result is zero, the next sequential instruction is skipped. |
| DCA | 3XXX | 11 | 16 | 17 | DEPOSIT AND CLEAR THE ACCUMULATOR: The contents of the AC are stored in the effective address and the AC is cleared. |
| JMS | 4XXX | 11 | 16 | 17 | JUMP TO SUBROUTINE: The contents of the PC are stored in the effective address and the effective address + 1 is stored in the PC. The link, AC, and MQ are unchanged. |
| JMP | 5XXX | 10 | 15 | 16 | JUMP: The effective address is loaded into the PC thus causing program execution to branch to a new location. |
| IOT | 6XXX | 17 | | | INPUT/OUTPUT TRANSFER: Used to initiate the operation of peripheral devices and to transfer data between the peripherals and the CPU. |
| OPI | 7XXX | | | | OPERATE Instructions: Used to perform logical operations on the contents of the major registers. |
| | | 10 | | | 2 - Cycle OPERATE |
| | | 15 | | | 3 - Cycle OPERATE |

## *Operate Instructions*

The Operate Instructions, which have an OPCODE of 7$_8$(111), consist of 3 groups of microinstructions. Group 1 microinstructions, which are identified by the presence of a 0 in bit 3, are used to perform logical operations on the contents of the accumulator and link. Group 2 micro instructions, which are identified by the presence of a 1 in bit 3 and a 0 in bit 11, are used primarily to test the contents of the accumulator and then conditionally skip the next sequential instruction. Group 3 microinstructions have a 1 in bit 3 and a 1 in bit 11 and are used to perform logical operations on the contents of the AC and MQ.

The basic OPR instruction format is shown in Figure 5. Operate microinstructions from any group may be microprogrammed with other operate microinstructions of the same group. The actual code for a microprogrammed combination of two, or more, microinstructions is the bitwise logical OR of the octal codes for the individual microinstructions. When more than one operation is microprogrammed into a single instruction, the operations are performed in a prescribed sequence, with logical sequence number 1 microinstructions performed first, logical sequence number 2 microinstructions performed second, logical sequence number 3 microinstructions performed third, and so on. Two operations with the same logical sequence number, within a given group of microinstructions, are performed simultaneously.

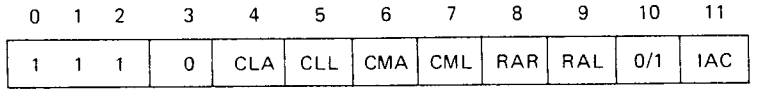



| MICROINSTRUCTION | A | B |
|---|---|---|
| Group 1 | 0 | 0/1 |
| Group 2 | 1 | 0 |
| Group 3 | 1 | 1 |

FIGURE 5 – Basic OPR Instruction Format

## GROUP 1 MICROINSTRUCTIONS

Figure 6 shows the instruction format of a group 1 microinstruction. Any one of bits 4 to 11 may be set, loaded with a binary 1, to indicate a specific group 1 microinstruction. If more than one of these bits is set, the instruction is a microprogrammed combination of group 1 microinstructions, which will be executed according to the logical sequence shown in Figure 6.




Logical Sequences:
1- CLA CLL
2 - CMA CML
3 - IAC
4 - RAR RAL RTR RTL BSW

| BIT 8 | BIT 9 | BIT 10 | FUNCTION |
|---|---|---|---|
| 0 | 0 | 1 | BSW |
| 0 | 1 | 0 | RAL |
| 0 | 1 | 1 | RTL |
| 1 | 0 | 0 | RAR |
| 1 | 0 | 1 | RTR |

FIGURE 6 – Group 1 Microinstruction Format

Table 2-1 lists commonly used group 1 microinstructions, their assigned mnemonics, octal number, instruction format, logical sequence, the operation they perform, and the number of states. The same format is followed in Table 3 and 4 which corresponds to group 2 and 3 microinstructions, respectively.

There are several commonly used microprogrammed combinations of group 1 microinstructions. These are listed in Table 2-2. When writing programs it is necessary to load various constants into the AC for such purposes as initiallizing counters and to provide comparisons. Table 2-3 lists those constants which can be loaded directly via microprogrammed combinations of group 1 instructions.

TABLE 2 - 1

| MNE-MONIC | OCTAL CODE | LOGICAL SEQUENCE | NUMBER OF STATES | OPERATION |
|---|---|---|---|---|
| NOP | 7000 | 1 | 10 | NO OPERATION - This instruction causes a 10 state delay in program execution, without affecting the state of the HM-6100. It may be used for timing synchronization or as a convenient means of deleting an instruction from a program. |
| CLA | 7200 | 1 | 10 | CLEAR ACCUMULATOR - The accumulator is loaded with binary 0's. |

FIGURE 2 - 1 Continued

| MNE-MONIC | OCTAL CODE | LOGICAL SEQUENCE | NUMBER OF STATES | OPERATION |
|---|---|---|---|---|
| CLL | 7100 | 1 | 10 | CLEAR LINK - The link is loaded with a binary 0. |
| CMA | 7040 | 2 | 10 | COMPLEMENT ACCUMULATOR - The content of each bit of the AC is complemented. This has the effect of replacing the contents of the AC with its one's complement. |
| CML | 7020 | 2 | 10 | COMPLEMENT LINK - The content of the link is complemented. |
| IAC | 7001 | 3 | 10 | INCREMENT ACCUMULATOR - The content of the AC is incremented by one (1) and the carry out componments the Link (L). |
| BSW | 7002 | 4 | 15 | BYTE SWAP - The right six (6) bits of the AC are exchanged or SWAPPED with the left six bits. AC(0) is swapped with AC(6), AC(1) with AC(7), etc. The link is not affected. |
| RAL | 7004 | 4 | 15 | ROTATE ACCUMULATOR LEFT - The content of the AC and L are rotated one binary position to the left. AC(0) is shifted to L and L is shifted to AC(11). The ROTATE instructions use what is commonly called a circular shift, meaning that any bit rotated off one end of the accumulator will reappear at the other end. |
| RTL | 7006 | 4 | 15 | ROTATE TWO LEFT - The contents of the AC and L are rotated two binary positions to the left. AC(1) is shifted to L and L is shifted to AC(10). |
| RAR | 7010 | 4 | 15 | ROTATE ACCUMULATOR RIGHT - The contents of the AC and L are rotated one binary position to the right. AC(11) is shifted to L and L is shifted to AC(0). |
| RTR | 7012 | 4 | 15 | ROTATE TWO RIGHT - The contents of the AC and L are rotated two binary positions to the right. AC(10) is shifted to L and L is shifted to AC(1). |

TABLE 2 - 2

| MNE-MONIC | OCTAL CODE | LOGICAL SEQUENCE | NUMBER OF STATES | OPERATION |
|---|---|---|---|---|
| CLA CLL | 7300 | 1 | 10 | CLEAR ACCUMULATOR - CLEAR LINK |
| CIA | 7041 | 2, 3 | 10 | COMPLEMENT AND INCREMENT ACCUMULATOR - The content of the AC is replaced with its two's complement. The carry out complements the link. This is a microprogrammed combination of CMA and IAC. |
| STL | 7120 | 1, 2 | 10 | SET THE LINK - The LINK is loaded with a binary 1 corresponding with a microprogrammed combination of CLL and CML. |
| STA | 7240 | 1, 2 | 10 | SET THE ACCUMULATOR - Each bit of the AC is set to 1 corresponding to a microprogrammed combination of CLA and CMA. |
| CLA IAC | 7201 | 1, 3 | 10 | Sets the accumulator to a 1. |

TABLE 2 - 2 Continued

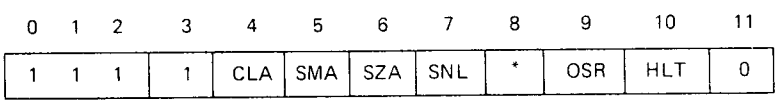
| MNE-MONIC | OCTAL CODE | LOGICAL SEQUENCE | NUMBER OF STATES | OPERATION |
|---|---|---|---|---|
| GLK | 7204 | 1, 4 | 15 | GET LINK - The AC is cleared and the content of the link is shifted into AC(11) while a 0 is shifted into the link. This is a microprogrammed combination of CLA and RAL. |
| CLL RAL | 7104 | 1, 4 | 15 | CLEAR LINK - ROTATE ACCUMULATOR LEFT |
| CLL RTL | 7106 | 1, 4 | 15 | CLEAR LINK - ROTATE TWO LEFT |
| CLL RAR | 7110 | 1, 4 | 15 | CLEAR LINK - ROTATE ACCUMULATOR RIGHT |
| CLL RTR | 7112 | 1, 4 | 15 | CLEAR LINK - ROTATE TWO RIGHT |

TABLE 2 - 3

| MNEMONIC | OCTAL CODE | LOGICAL SEQUENCE | NUMBER OF STATES | DECIMAL CONSTANT | INSTRUCTIONS COMBINED |
|---|---|---|---|---|---|
| NL0000 | 7300 | 1 | 10 | 0 | CLA CLL |
| NL0001 | 7301 | 1, 3 | 10 | 1 | CLA CLL IAC |
| NL0002 | 7305 | 1, 3, 4 | 15 | 2 | CLA CLL IAC RAL |
| NL0003 | 7325 | 1, 2, 3, 4 | 15 | 3 | CLA CLL CML IAC RAL |
| NL0004 | 7307 | 1, 3, 4 | 15 | 4 | CLA CLL IAC RTL |
| NL0006 | 7327 | 1, 2, 3, 4 | 15 | 6 | CLA CLL CML IAC RTL |
| NL0100 | 7303 | 1, 3, 4 | 15 | 64 | CLA CLL BSW |
| NL2000 | 7332 | 1, 2, 4 | 15 | 1024 | CLA CLL CML RTR |
| NL3777 | 7350 | 1, 2, 4 | 15 | 2047 | CLA CLL CMA RAR |
| NL4000 | 7330 | 1, 2, 4 | 15 | -0 | CLA CLL CML RAR |
| NL5777 | 7352 | 1, 2, 4 | 15 | -1025 | CLA CLL CMA RTL |
| NL6000 | 7333 | 1, 2, 3, 4 | 15 | -1024 | CLA CLL CML IAC RTL |
| NL7775 | 7346 | 1, 2, 4 | 15 | -3 | CLA CLL CMA RTL |
| NL7776 | 7344 | 1, 2, 4 | 15 | -2 | CLA CLL CMA RAL |
| NL7777 | 7340 | 1, 2 | 10 | -1 | CLA CLL CMA |

## GROUP 2 MICROINSTRUCTIONS

Figure 7 shows the instruction format of group 2 microinstructions, Bits 4 - 10 may be set to indicate a specific group 2 microinstruction. If more than one of bits 4 - 7 or 9 - 10 is set, the instruction is a microprogrammed combination group 2 microinstructions, which will be executed according to the logical sequence shown in Figure 7.




Logical Sequences:
1 (BIT 8 = 0) -SMA or SZA or SNL
(BIT 8 = 1) -SPA or SNA or SZL
2 -CLA
3 -OSR, HLT

* Reverse sensing BIT:
Unconditional SKIP when
BITS 5, 6, & 7 are 0's

**FIGURE 7 -- Group 2 Microinstruction Format**

Skip microinstructions may be microprogrammed with CLA, OSR, or HLT microinstructions. Skip microinstructions which have a 0 in bit 8, however, may not be microprogrammed with skip microinstructions which have a 1 in bit 8. When two or more skip microinstructions are microprogrammed into a single instruction, the resulting condition on which the decision will be based is the logical OR of the individual conditions when bit 8 is 0, or when bit 8 is 1, the decision will be based on the logical AND.

TABLE 3 -1

| MNE-MONIC | OCTAL CODE | LOGICAL SEQUENCE | NUMBER OF STATES | OPERATION |
|---|---|---|---|---|
| NOP | 7400 | 1 | 10 | NO OPERATION - See Group 1 microinstructions. |
| CLA | 7600 | 2 | 10 | CLEAR ACCUMULATOR - The accumulator is loaded with binary O's. |
| HLT | 7402 | 3 | 10 | HALT - Program stops at the conclusion of the current machine cycle. If HLT is combined with others in OPR 2, the other operations are completed before the end of the cycle. |
| SKP | 7410 | 1 | 10 | SKIP - The content of the PC is incremented by 1, to skip the next instruction. |
| SNL | 7420 | 1 | 10 | SKIP ON NON-ZERO LINK - The content of L is sampled; the next sequential instruction is skipped if L contains a 1. If L contains a 0, the next instruction is executed. |
| SZL | 7430 | 1 | 10 | SKIP ON ZERO LINK - The instruction is skipped if the link contains a 0. |
| SZA | 7440 | 1 | 10 | SKIP ON ZERO ACCUMULATOR - The content of the AC is sampled; the next sequential instruction is skipped if all AC bits are 0. If any bit in the AC is a 1, the next instruction is executed. |
| SNA | 7450 | 1 | 10 | SKIP ON NON-ZERO ACCUMULATOR - The next instruction is skipped if any one bit of the AC contains a 1. If every bit in the AC is 0, the next instruction is executed. |
| SMA | 7500 | 1 | 10 | SKIP ON MINUS ACCUMULATOR - If the content of AC(0) contains a negative two's complement number, the next sequential instruction is skipped. If AC(0) contains a 0, the next instruction is executed. |
| SPA | 7510 | 1 | 10 | SKIP ON POSITIVE ACCUMULATOR - If the content of AC(0) contains a 0, indicating a positive two's complement number, the next sequential instruction is skipped. |
| OSR | 7404 | 3 | 15 | OR WITH SWITCH REGISTER - The content of the Switch Register is inclusively OR'ed with the content of the AC and the result stored in the AC. The HM-6100 sequences the OSR instruction through a 2-cycle execute phase referred to as OPR 2A and OPR 2B. This instruction provides the simplest way to input data to the HM-6100 from peripherals. |
| LAS | 7604 | 1, 3 | 15 | LOAD ACCUMULATOR WITH SWITCH REGISTER - The content of the AC is loaded with the content of the SR, bit for bit. This is equivalent to a microprogrammed combination of CLA and OSR. |

Table 3 - 2 lists every legal combination of skip microinstructions, along with the resulting condition upon which the decision to skip or execute the next sequential instruction is based. When these combinations include a CLA, the accumulator is cleared after the decision is made. This is a useful trick to save code when a new value will be TAD'ed into the AC.

TABLE 3 - 2

| MNEMONIC | OCTAL CODE | LOGICAL SEQUENCE | NUMBER OF STATES | OPERATION |
|---|---|---|---|---|
| SZA SNL | 7460 | 1 | 10 | Skip if AC = 0 or L = 1 or both. |
| SNA SZL | 7470 | 1 | 10 | Skip if AC ≠ 0 and L = 0. |
| SMA SNL | 7520 | 1 | 10 | Skip if AC < 0 or L = 1 or both. |
| SPA SZL | 7530 | 1 | 10 | Skip if AC ≥ 0 and L = 0 |
| SMA SZA | 7540 | 1 | 10 | Skip if AC ≤ 0. |
| SPA SNA | 7550 | 1 | 10 | Skip if AC > 0. |
| SMA SZA SNL | 7560 | 1 | 10 | Skip if AC ≤ 0 or L = 1 or both. |
| SPA SNA SZL | 7570 | 1 | 10 | Skip if AC > 0 and L = 0. |

When writing an actual program, it is useful to think in terms of the FORTRAN relational operators - .LT., .EQ., etc.- when trying to compare numbers. The following method along with Table 3 - 3 will provide this.

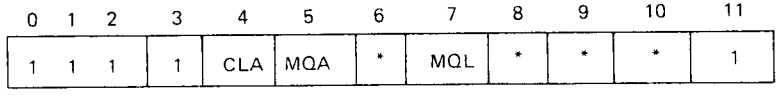
```
CLA CLL                 /Initialize AC and Link
TAD B                   /Fetch 2nd number
CML CMA IAC             /Create "-B" (AC & L act like a 13 bit accumulator)
TAD A                   /Fetch 1st number
Test <CLA>              /Use instructions from Table 3 - 3 to provide test
                        /The CLA is optional to provide a clear AC after test
JMP FAIL                /Branch to FAIL routine if test failed
. . .                   /Test passed, continue with program

. . .
```

TABLE 3 - 3

| SKIP IF | UNSIGNED COMPARE | SIGNED COMPARE |
|---|---|---|
| A .NE. B | SNA | SNA |
| A .LT. B | SNL | SMA |
| A .LE. B | SNL SZA | SMA SZA |
| A .EQ. B | SZA | SZA |
| A .GE. B | SZL | SPA |
| A .GT. B | SZL SNA | SPA SNA |

## GROUP 3 MICROINSTRUCTIONS

Figure 8 shows the instruction format of group 3 microinstructions which requires bits 3 and 11 to contain a 1. Bits 4, 5 or 7 may be set to indicate a specific group 3 microinstruction. If more than one of the bits is set, the instruction is a microprogrammed combination of group 3 microinstructions following the logical sequence listed in Figure 8.




Logical Sequences:
1 - CLA
2 - MQA, MQL
3 - NOP

* Don't care

FIGURE 8 – Group 3 Microinstruction Format

TABLE 4

| MNE-MONIC | OCTAL CODE | LOGICAL SEQUENCE | NUMBER OF STATES | OPERATION |
|---|---|---|---|---|
| NOP | 7401 | 3 | 10 | NO OPERATION - See group 1 microinstructions. |
| CLA | 7600 | 1 | 10 | CLEAR ACCUMULATOR |
| MQA | 7501 | 2 | 10 | MQ REGISTER INTO ACCUMULATOR - The content of the MQ is logical OR'ed with the content of the AC and the result is loaded into the AC. The original content of the AC is lost but the original content of the MQ is retained. This instruction provides the programmer with an inclusive OR operation. |
| MQL | 7421 | 2 | 10 | MQ REGISTER LOAD - The content of the aAC is loaded into the MQ, the AC is cleared and the original content of the MQ is lost. This is similar to a DCA instruction. |
| ACL | 7701 | 1, 2 | 10 | CLEAR ACCUMULATOR AND LOAD MQ REGISTER INTO ACCUMULATOR - This is equivalent to a microprogrammed combination of CLA and MQA. It is similar to the two instruction combination of CLA and TAD. |
| CAM | 7621 | 1, 2 | 10 | CLEAR ACCUMULATOR AND MQ REGISTER - The content of the AC and MQ are loaded with binary 0's. This is equivalent to a microprogram combination of CLA and MQL. |
| SWP | 7521 | 2 | 10 | SWAP ACCUMULATOR AND MQ REGISTER - The content of the AC and MQ are interchanged by accomplishing a microprogrammed combination of MQA and MQL. |
| CLA SWP | 7721 | 1, 2 | 10 | CLEAR ACCUMULATOR AND SWAP ACCUMULATOR AND MQ REGISTER - The content of the AC is cleared. The content of the MQ is loaded into the AC and the MQ is cleared. |

## *Input Output Transfer Instructions (IOT)*

The input/output transfer instructions, which have an OPCODE of $6_8$ are used to initiate the operation of peripheral devices and to transfer data between peripherals and the HM-6100. Three types of data transfer may be used to receive or transmit information between the HM-6100 and one or more peripheral I/O devices. PROGRAMMED DATA TRANSFER provides a straightforward means of communicating with relatively slow I/O devices, such as Teletypes, cassettes, card readers and CRT displays. INTERRUPT TRANSFERS use the interrupt system to service several peripheral devices simultaneously, on an intermittent basis, permitting computational operations to be performed concurrently with the data I/O operations. Both Programmed Data Transfers and Program Interrupt Transfers use the accumulator as a buffer, or storage area, for all data transfers. Since data may be transferred only between the accumulator and the peripheral, only one 12 bit word at a time may be transferred. DIRECT MEMORY ACCESS, DMA, Transfers variable-size blocks of data between high-speed peripherals and the memory with minimum of program control required by the HM-6100.

### IOT INSTRUCTION FORMAT

The Input/Output Transfer instruction format is represented in Figure 9.

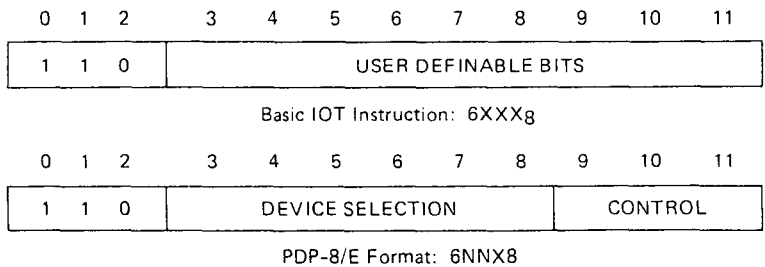



**FIGURE 9 – IOT Instruction Format**

The first three bits, 0 - 2, are always set to 6$_8$ (110) to specify an IOT instruction. The next 9 bits, 3 - 11, are user definable and can provide a minimal implementation when each bit controls one operation. When following PDP-8/E format, the next six bits, 3 - 8, contain the device selection code that determines the specific I/O device for which the IOT instruction is intended and, therefore, permit interface with up to 64 I/O devices. The last three bits, 9 - 11, contain the operation specification code that determines the specific operation to be performed. The nature of this operation for any given IOT instruction depends entirely upon the circuitry designed into the I/O device interface.

## PROGRAMMED DATA TRANSFER

Programmed Data Transfer is the easiest, simplest, most convenient and most common means of performing data I/O. For microprocessor applications, it may also be the most cost effective approach. The data transfer begins when the HM-6100 fetches an instruction from the memory and recognizes that the current instruction is an IOT (2). This is referred to an IFETCH and consists of five (5) internal states. The HM-6100 sequences the IOT instruction through a 2-cycle execute phase referred to as IOTA and IOTB. Bits 0 - 11 of the IOT instruction are available on DX0 - 11 at IOTA ∧ LXMAR (3). These bits must be latched in an external address register. $\overline{\text{DEVSEL}}$ is active low to enalbe data transfers between the HM-6100 and the peripheral device (4) & (5). Input-Output Instruction Timing is shown in Figure 10. The selected peripheral device communicates with the HM-6100 through 4 control lines - $\overline{\text{C0}}$, $\overline{\text{C1}}$, $\overline{\text{C2}}$ and $\overline{\text{SKP}}$. In the HM-6100 the type of data transfer, during an IOT instruction, is specified by the peripheral device(s) by asserting the control lines as shown in Tables 5-1 and 5-2.

The control line $\overline{\text{SKP}}$, when low during an IOT, causes the HM-6100 to skip the next sequential instruction. This feature is used to sense the status of various signals in the device interface. The C0, C1, and C2 lines are treated independently of the $\overline{\text{SKP}}$ line. In the case of a RELATIVE or ABSOLUTE JUMP, the skip operation is performed after the jump. The input signals to the HM-6100, DX0 - 11, $\overline{\text{C0}}$, $\overline{\text{C1}}$, $\overline{\text{C2}}$ and $\overline{\text{SKP}}$, are sampled during IOTA on the rising edge of time state 3 (4). The data from the HM-6100 is available to the device during $\overline{\text{DEVSEL}}$ ∧ $\overline{\text{XTC}}$ (5). The IOTB cycle is internal to the HM-6100 to perform the operations requested during IOTA. Both IOTA and IOTB consists of six (6) internal states.

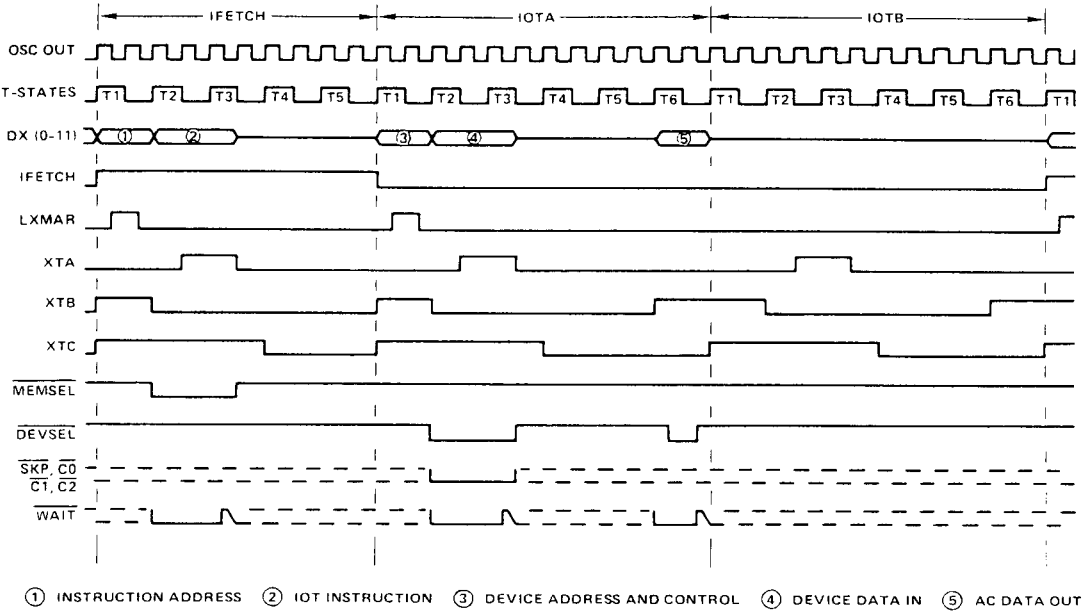



FIGURE 10 – Input-output instruction timing

TABLE 5 - 1
AC DATA TRANSFERS

| CONTROL LINES | | | | OPERATION | DESCRIPTION |
|---|---|---|---|---|---|
| $\overline{SKP}$ | $\overline{C0}$ | $\overline{C1}$ | $\overline{C2}$ | | |
| H | H | H | H | DEV ← AC | The content of the AC is sent to the device. |
| H | L | H | H | DEV ← AC; CLA | The content of the AC is sent to a device and then the AC is cleared. |
| H | H | L | H | AC ← AC V DEV;<br>DEV ← AC | Data is received from a device OR'ed with the data in the AC and the result is stored in the AC. The new AC content is sent to the device. |
| H | L | L | H | AC ← DEV;<br>DEV ← AC | Data is received from a device and loaded into the AC. The new AC content is sent to the device. |
| L | H | H | H | DEV ← AC;<br>PC ← PC + 1 | The content of the AC is sent to the device and the microprocessor skips the next sequential instruction. |
| L | L | H | H | DEV ← AC; CLA;<br>PC ← PC + 1 | The content of the AC is sent to a device, the AC is cleared, and the microprocessor skips the next sequential instruction. |
| L | H | L | H | AC ← AC V DEV;<br>DEV ← AC;<br>PC ← PC + 1 | Data is OR'ed into the AC, the new AC sent to the device, and the microprocessor skips the next sequential instruction. |
| L | L | L | H | AC ← DEV;<br>DEV ← AC<br>PC ← PC + 1 | Data is loaded into the AC, the new AC contents sent to the device, and the next sequential instruction skipped. |

TABLE 5 - 2
PC VECTOR TRANSFERS

| CONTROL LINES | | | | OPERATION | DESCRIPTION |
|---|---|---|---|---|---|
| $\overline{SKP}$ | $\overline{C0}$ | $\overline{C1}$ | $\overline{C2}$ | | |
| H | * | H | L | PC ← PC + DEV | Data from the device is added to the contents of the PC. This is referred to as a RELATIVE JUMP. |
| H | * | L | L | PC ← DEV | Data is received from a device and loaded into the PC. This is referred to as an ABSOLUTE JUMP. |
| L | * | H | L | PC ← PC + DEV;<br>PC ← PC + 1 | The RELATIVE JUMP is performed and then the microprocessor skips the next sequential instruction. |
| L | * | L | L | PC ← DEV;<br>PC ← PC + 1 | The ABSOLUTE JUMP is executed and then the next sequential instruction is skipped. |

* Don't Care

## PROGRAM INTERRUPT TRANSFERS

The program interrupt system may be used to initiate programmed data transfers in such a way that the time spent waiting for device status is greatly reduced or eliminated altogether. It also provides a means of performing concurrent programmed data transfers between the HM-6100 and the peripheral devices. This is accomplished by isolating the I/O handling routines from the mainline program and using the interrupt system to ensure that these routines are entered only when an I/O device status is set, indicating that the device is actually ready to perform the next data transfer, or that is requires some sort of intervention from the running program.

TABLE 6
PROCESSOR IOT INSTRUCTIONS

| MNE-MONIC | OCTAL CODE | OPERATION |
|---|---|---|
| SKON | 6000 | SKIP IF INTERRUPT ON - If Interrupt system is enabled, the next sequential instruction is skipped. The Interrupt system is disabled. |
| ION | 6001 | INTERRUPT TURN ON - The internal interrupt acknowledge system is enabled. The interrupt system is enabled after the CPU executes the next sequential instruction. |
| IOF | 6002 | INTERRUPT TURN OFF - The interrupt system is disabled. Note that the interrupt system is automatically disabled when the CPU acknowledges an INT request. |
| SRQ | 6003 | SKIP IF INT REQUEST - The next sequential instruction is skipped if the INT request bus is low. |
| GTF | 6004 | GET FLAGS - The following machines states are read into the indicated bits of AC.<br>bit 0 - Link<br>bit 1 - Greater than flag*<br>bit 2 - INT request bus<br>bit 3 - Interrupt Inhibit FF*<br>bit 4 - Interrupt Enable FF*<br>bit 5 - User flag*<br>bit 6 - 11 - Save Field Register*<br>* These bits are modified by external devices driving the DX bus and the $\overline{C}$-lines ($\overline{C0}$ = L, $\overline{C1}$ = L). For example, bits 1 and 6 - 11 are part of the Extended Memory Control. |
| RTF | 6005 | RETURN FLAGS - Link is restored from AC (0). Interrupt system is enabled after the next sequential instruction is executed. All AC bits are available externally to restore external states. (ex. Extended memory control). ($\overline{C0}$ = H, $\overline{C1}$ = H) |
| SGT | 6006 | SKIP ON GREATER THAN FLAG - Operation is determined by external devices, if any. This flag is external and must control the skip line. |
| CAF | 6007 | CLEAR ALL FLAGS - AC and link are cleared. Interrupt system is disabled. |

The interrupt system allows certain external conditions to interrupt the computer program by driving the $\overline{\text{INTREQ}}$ input to the HM-6100 low. If no higher priority requests are outstanding and the interrupt system is enabled, the HM-6100 grants the device interrupt at the end of the current instruction. After an interrupt has been granted, the Interrupt Enable Flip-Flop in the HM-6100 is reset so that no more interrupts are acknowledged until the interrupt system is re-enabled under program control.

The current content of the Program Counter, PC, is deposited in location $0000_8$ of the memory and the program fetches the instruction from location $0001_8$. The return address is available in location $0000_8$. This address must be saved, possibly in a software stack, if nested interrupts are permitted. The INTGNT signal is activated by the HM-6100 when a device interrupt is acknowledged. This signal is reset by executing any IOT instruction. The INTGNT is also useful in implementing an External Vectored Priority Interrupt network.

The user program controls the interrupt mechanism of the HM-6100 by executing the processor IOT instructions listed in Table 6. Several of these interrupt IOT instructions are also used if the memory is extended beyond 4K words.

## DIRECT MEMORY ACCESS (DMA)

Direct Memory Access, sometimes called data break, is the perferred form of data transfer for use with high-speed storage devices such as magnetic disk or tape units. The DMA mechanism transfers data directly between memory and peripheral devices. The HM-6100 is involved only is setting up the transfer; the transfers take place with no processor intervention on a "cycle stealing" basis. The DMA transfer rate is limited only by the bandwidth of the memory and the data transfer characteristics of the device.

The device generates a DMA Request when it is ready to transfer data. The HM-6100 grants the $\overline{\text{DMAREQ}}$ by activating the DMAGNT signal at the end of the current instruction. The HM-6100 suspends any further instruction fetches until the $\overline{\text{DMAREQ}}$ line is released. The DX lines are tri-stated, all SEL lines are high, and the external timing signals XTA, XTB, and XTC are active. The device which generated the $\overline{\text{DMAREQ}}$ must provide the address and necessary control signals to the memory for data transfers. The $\overline{\text{DMAREQ}}$ line can also be used as a level sensitive "pause" line.

## *Control Panel Interrupt Transfer*

The HM-6100 CPU provides a unique Control Panel (CP) feature through its $\overline{\text{CPREQ}}$ input and $\overline{\text{CPSEL}}$ output lines. After acknowledging the control panel request, the CPU generates the necessary timing to execute program code in CP memory while also providing the capability to transfer data between CP memory and the user memory using the AC as a buffer. This allows the user memory to be examined and/or modified by the CP software. The CPU will output the $\overline{\text{MEMSEL}}$ signal for all user memory references while the $\overline{\text{CPSEL}}$ signal is generated for CP memory references as shown in Figure 11.

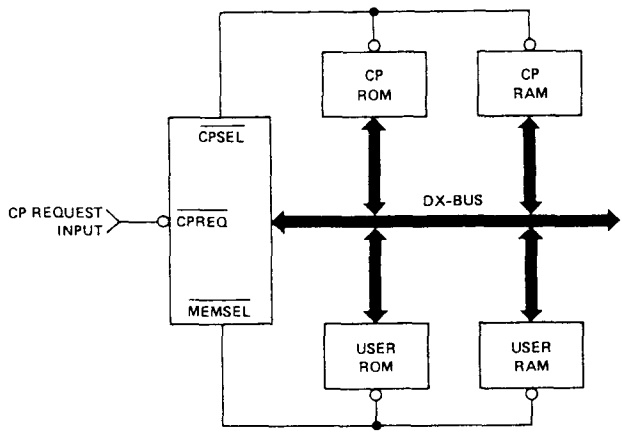



**FIGURE 11 – Control Panel Block Diagram**

The designer can make use of the control panel features to implement various functions that will be "transparent" to the user's (main) memory. Some of the more common functions include:

- Binary Loader and Punch
- Register Examination and Modification
- Single Cycle
- Octal Debug with Breakpoints
- Octal listing
- Auto Bootstrap

When a $\overline{\text{CPREQ}}$ is granted the PC is stored in location 0000 of Panel Memory and the HM-6100 resumes operation at location 7777 of the Panel Memory. The $\overline{\text{CPREQ}}$ bypasses the interrupt enable system and the processor IOT instruction, ION and IOF, are ignored while the HM-6100 is in the Control Panel Mode. Once a $\overline{\text{CPREQ}}$ is granted, the HM-6100 will not recognize any $\overline{\text{DMAREQ}}$ or $\overline{\text{INTREQ}}$ until the CPREQ has been fully serviced.

During Control Panel program execution access to the user memory is gained through use of indirect TAD, AND, DCA and ISZ instructions. The CPU will transfer control from $\overline{\text{CPSEL}}$ to $\overline{\text{MEMSEL}}$ during the execute phase of these instructions. The instructions are always fetched from control panel memory.

Exiting from the control panel routine is achieved by executing the following sequence:

- ION
- JMP I 0000 /Exit via location 0000 in Panel Memory

Location 0000 contains either the original return address deposited by the HM-6100 when the CP routine was entered, or it may be a new starting address defined by the CP routine.

---

## *Internal Priority Structure*

After an instruction is completely sequenced, the major state generator scans the internal priority network as shown in in Figure 12. The state of the priority network decides the next sequence of the HM-6100.

The CPU samples the $\overline{\text{RESET}}$ line, the request lines $\overline{\text{CPREQ}}$, $\overline{\text{DMAREQ}}$, and $\overline{\text{INTREQ}}$, and the state of its internal RUN flip-flop during the last execute cycle of each instruction. The worst case response time of the HM-6100 to an external request is, therefore the time required to execute the longest instruction preceded by any 6-state execution cycle. For the HM-6100, this is an autoindexed ISZ, 22 states, preceded by any 6-state execution cycle instruction. The worst case response time is, therefore, 28 states, 14 $\mu$s at 4MHz clock frequency.

When the HM-6100 is initially powered up, the state of the timing generator is undefined. The generator is automatically initialized with a maximum of 34 clock pulses. The request inputs, as the HM-6100 is powered on, must span at least 58 clock pulses to be recognized, 34 clocks for the counter to initialize and a maximum of two HM-6100 cycles (20 to 24 clocks) for the state generator to sample the request lines. A positive transition of RUN/$\overline{\text{HLT}}$ should occur at least 10 clock pulses after $\overline{\text{RESET}}$ to be recognized.

The priority hierarchy is:

- $\overline{\text{RESET}}$ - If the $\overline{\text{RESET}}$ line is asserted at the sample time, the processor immediately sets its program counter to 7777, clears the Accumulator and Link, and puts the processor in the $\overline{\text{HALT}}$ state. While halted, the processor continues to cycle and generate the timing signals XTA, XTB, and XTC. During reset the DX line is tri-stated and the SEL lines are high.
- $\overline{\text{CPREQ}}$ - If the $\overline{\text{RESET}}$ line is not found to be asserted, but the $\overline{\text{CPREQ}}$ line is, the processor grants the control panel interrupt request at the end of the current cycle.
- RUN/$\overline{\text{HLT}}$ - If neither of the foregoing lines are asserted, but the processor finds its internal RUN FF in the halt state, it enters the HALT cycle at the end of the last execute cycle. Pulsing the RUN/$\overline{\text{HLT}}$ line low causes the HM-6100 to alternately run and halt. The internal RUN FF changes state on the rising edge of the RUN/$\overline{\text{HLT}}$ line. While halted the processor continues to generate the timing signals XTA, XTB, and XTC.
- $\overline{\text{DMAREQ}}$ - DMA requests are granted at the end of the current cycle only if none of the above actions are pending.
- $\overline{\text{INTREQ}}$ - An interrupt request is granted at the end of the current cycle only if none of the higher priority lines preempts it.
- IFETCH - If none of the above actions are indicated, the processor will fetch the next sequential instruction in the next cycle.

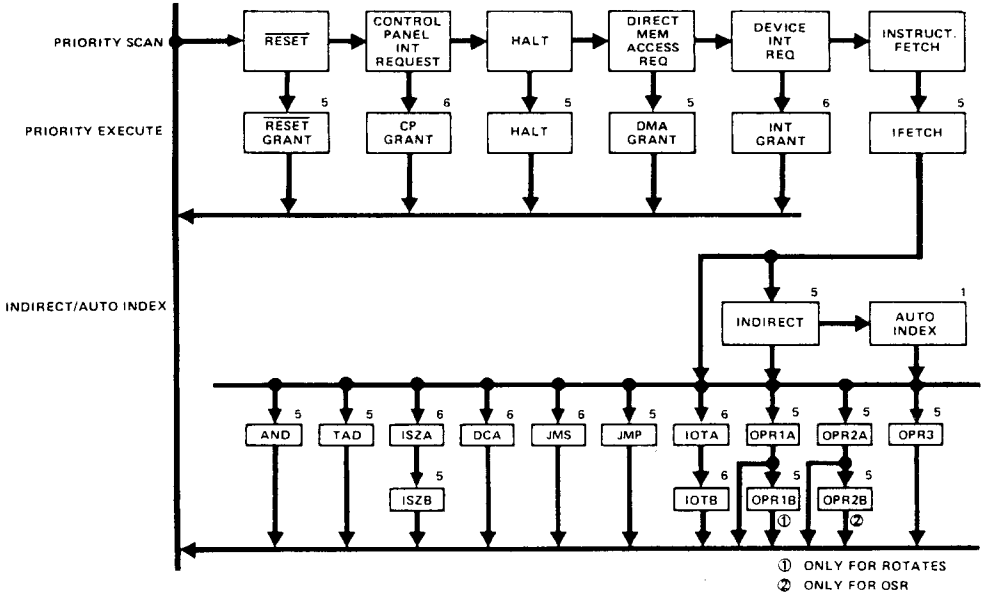



**FIGURE 12 – Major processor states and number of clock cycles in each state.**

## *Use of Wait Input*

The HM-6100 samples the $\overline{\text{WAIT}}$ line during input-output data transfers. The $\overline{\text{WAIT}}$ line, if active low, controls the transfer duration. If $\overline{\text{WAIT}}$ is active during input transfers (READ), the CPU waits in the T2 state. For an output transfer (WRITE), $\overline{\text{WAIT}}$ controls the time for which the write data is maintained on the DX lines by extending the T6 state. When operating at the max frequency, the internal delay of the HM-6100 causes the falling edge select lines to be past the $\overline{\text{WAIT}}$ setup time for WRITE. The rising edge of the select line for READ can be used to activate $\overline{\text{WAIT}}$ for a WRITE. The wait duration is an integral multiple of the oscillator time period (Figure 13).

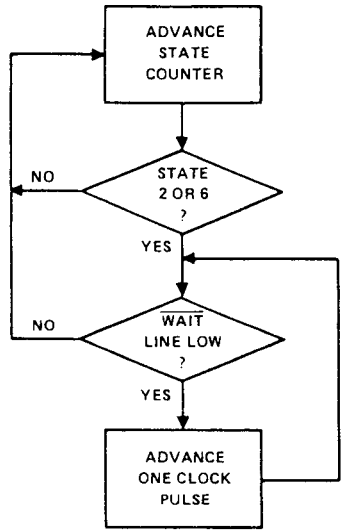



FIGURE 13 – WAIT sequencing steps.

## *HM-6100 Oscillator Requirements*

### USING AN EXTERNAL CRYSTAL

An inexpensive crystal can be used thereby eliminating the need for a clock generator. The crystal operates at parallel resonance, and thus is looks inductive in the circuit. An "AT" cut crystal should be used because it has a low temperature coefficient and can be used over a wide temperature range. The Feedback resistor and shunt capacitance are included internally. The crystal parameters needed are:

- Frequency
- Mod of Resonance - Parallel (anti-resonant)
- Maximum Power level - 1 milliwatt
- Load Capacitance - 32pF
- Series Resistance (max) - 250Ω

For precise frequency determination the effect of the stray circuit capacitance and internal 30pF capacitance must be taken into account.

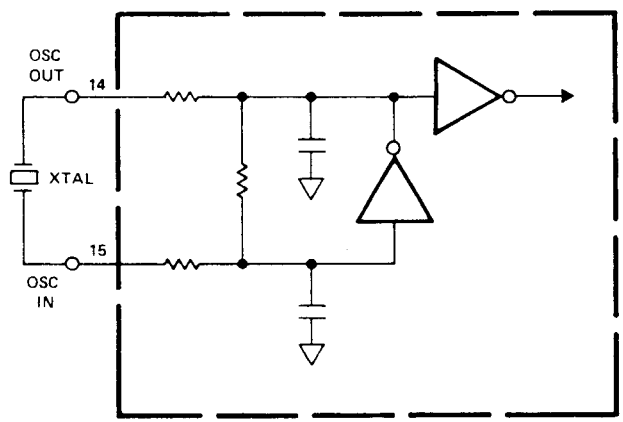



FIGURE 14 – Oscillator input schematic

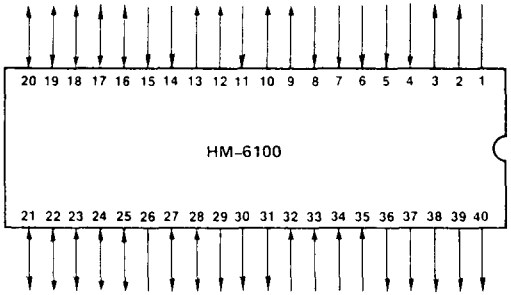
### USING AN EXTERNAL CLOCK GENERATOR

When a system clock is needed, eg. for a baud rate generator for UARTs, the HM-6100 can be externally clocked, thus eliminating the need for separate crystals. The external clock can be connected to the oscillator output pin while grounding oscillator input. This has the effect of over driving the small internal oscillator inverter causing an increase in supply current.

Duty cycle - 50/50
$T_{rise}$, $T_{fall}$ - 20ns

## PIN DEFINITIONS

| PIN | SYMBOL | ACTIVE LEVEL | DESCRIPTION |
|---|---|---|---|
| 1 | VCC | | Supply voltage. |
| 2 | RUN | H | The signal indicates the run state of the CPU and may be used to power down the external circuitry. |
| 3 | DMAGNT | H | Direct Memory Access Grant—DX lines are three-state. |
| 4 | $\overline{\text{DMAREQ}}$ | L | Direct Memory Access Request—DMA is granted at the end of the current instruction. Upon DMA grant, the CPU suspends program execution until the $\overline{\text{DMAREQ}}$ line is released. |
| 5 | $\overline{\text{CPREQ}}$ | L | Control Panel Request—a dedicated interrupt which bypasses the normal device interrupt request structure. |
| 6 | RUN/$\overline{\text{HLT}}$ | L | Pulsing the Run/Halt line causes the CPU to alternately run and halt by changing the state of the internal RUN/$\overline{\text{HLT}}$ flip flop. |
| 7 | $\overline{\text{RESET}}$ | L | Clears the AC and loads $7777_8$ into the PC. CPU is halted. |
| 8 | $\overline{\text{INTREQ}}$ | L | Peripheral device interrupt request. |
| 9 | XTA | H | External coded minor cycle timing—signifies input transfers to the HM-6100. |
| 10 | LXMAR | H | The Load External Address Register is used to store memory and peripheral address externally. |
| 11 | $\overline{\text{WAIT}}$ | L | Indicates that peripherals or external memory is not ready to transfer data. The CPU state gets extended as long as $\overline{\text{WAIT}}$ is active. The CPU is in the lowest power state with clocks running. |
| 12 | XTB | H | External coded minor cycle timing—signifies output transfers from the HM-6100. |
| 13 | XTC | H | External coded minor cycle timing—used in conjunction with the Select Lines to specify read or write operations. |
| 14 | OSC OUT | | Crystal input to generate the internal timing (also external clock input). |
| 15 | OSC IN | | See Pin 14—OSC OUT (also external clock ground). |
| 16 | DX0 | | DataX—multiplexed data in, data out and address lines. |
| 17 | DX1 | | See Pin 16—DX0. |
| 18 | DX2 | | See Pin 16—DX0. |
| 19 | DX3 | | See Pin 16—DX0. |
| 20 | DX4 | | See Pin 16—DX0. |




| PIN | SYMBOL | ACTIVE LEVEL | DESCRIPTION |
|---|---|---|---|
| 21 | DX5 | | See Pin 16—DX0. |
| 22 | DX6 | | See Pin 16—DX0. |
| 23 | DX7 | | See Pin 16—DX0. |
| 24 | DX8 | | See Pin 16—DX0. |
| 25 | DX9 | | See Pin 16—DX0. |
| 26 | GND | | Ground |
| 27 | DX10 | | See Pin 16—DX0. |
| 28 | DX11 | | See Pin 16—DX0. |
| 29 | LINK | H | Link flip flop. |
| 30 | $\overline{\text{DEVSEL}}$ | L | Device Select for I/O transfers. |
| 31 | $\overline{\text{SWSEL}}$ | L | Switch Register Select for the OR THE SWITCH REGISTER INSTRUCTION (OSR). OSR is a Group 2 Operate Instruction which reads a 12 bit external switch register and OR's it with the contents of the AC. |
| 32 | $\overline{\text{C0}}$ | L | Control line inputs from the peripheral device during an I/O transfer (Table 5). |
| 33 | $\overline{\text{C1}}$ | L | See Pin 32—C0. |
| 34 | $\overline{\text{C2}}$ | L | See Pin 32—C0. |
| 35 | $\overline{\text{SKP}}$ | L | Skips the next sequential instruction if active during an I/O instruction. (Table 5) |
| 36 | IFETCH | H | Instruction Fetch Cycle |
| 37 | $\overline{\text{MEMSEL}}$ | L | Memory Select for memory transfers. |
| 38 | $\overline{\text{CPSEL}}$ | L | The Control Panel Memory Select becomes active, instead of the $\overline{\text{MEMSEL}}$, for control panel routines. Signal may be used to distinguish between control panel and main memories. |
| 39 | INTGNT | H | Peripheral device Interrupt Grant |
| 40 | DATAF | H | Data Field pin indicates the execute phase of indirectly addressed AND, TAD, ISZ and DCA instructions so that the data transfers are controlled by the Data Field, DF, and not the Instruction Field, IF, if Extended Memory Control hardware is used to extend the addressing space from 4K to 32K words. |