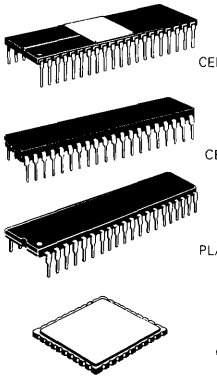
MOTOROLA

# MC146805H2

## Advance Information

**CMOS**
(HIGH-PERFORMANCE SILICON-GATE)

8-BIT
MICROCOMPUTER



L SUFFIX
CERAMIC PACKAGE
CASE 715

S SUFFIX
CERDIP PACKAGE
CASE 734

P SUFFIX
PLASTIC PACKAGE
CASE 711

Z SUFFIX
CHIP CARRIER
CASE 761

3

### 8-BIT MICROCOMPUTER UNIT

The MC146805H2 Microcomputer Unit (MCU) belongs to the M146805 CMOS Family of low-cost, single-chip microcomputers. This 8-bit MCU contains an on-chip oscillator, CPU, RAM, I/O, and a timer. The fully static design allows operation at two software selectable frequencies, further reducing its already low power consumption. It is a low-power processor designed for low-end to mid-range applications in the consumer, automotive, industrial, and communications market where very low power consumption constitutes an important factor. The following are the major features of the MC146805H2 MCU.

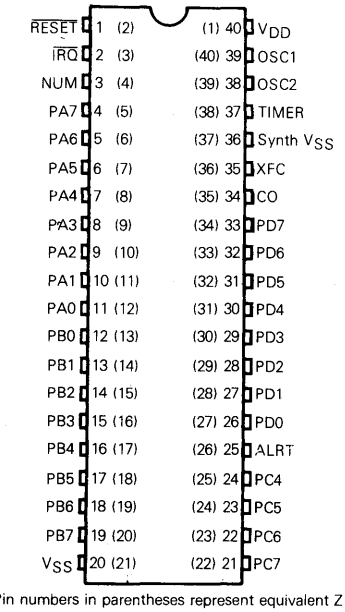
#### HARDWARE FEATURES

- Typical Full Speed Operating Power of 20 mW at 5 V
- Typical WAIT Mode Power of 4 mW
- Typical STOP Mode Power of 5 $\mu$W
- 8-Bit Architecture
- Fully Static Operation
- 112 Bytes of On-Chip RAM
- 2048 Bytes on On-Chip ROM
- 24 Bidirectional I/O Lines Plus Four Input-Only Lines
- Internal 8-Bit Timer with Software Programmable 7-Bit Prescaler
- Watchdog Timer
- External Timer Input
- External and Timer Interrupts
- Self-Check Mode
- Master Reset and Power-On Reset
- Single 3- to 6-Volt Supply
- On-Chip Oscillator
- 40-Pin Dual-in-Line Package
- Chip Carrier Also Available
- Alert Tone Generator
- Frequency Synthesizer

#### SOFTWARE FEATURES

- Similar to the MC6800
- Efficient Use of Program Space
- Versatile Interrupt Handling
- True Bit Manipulation
- Addressing Modes with Indexed Addressing for Tables
- Efficient Instruction Set
- Memory Mapped I/O
- Most Self-Check Routines User Callable

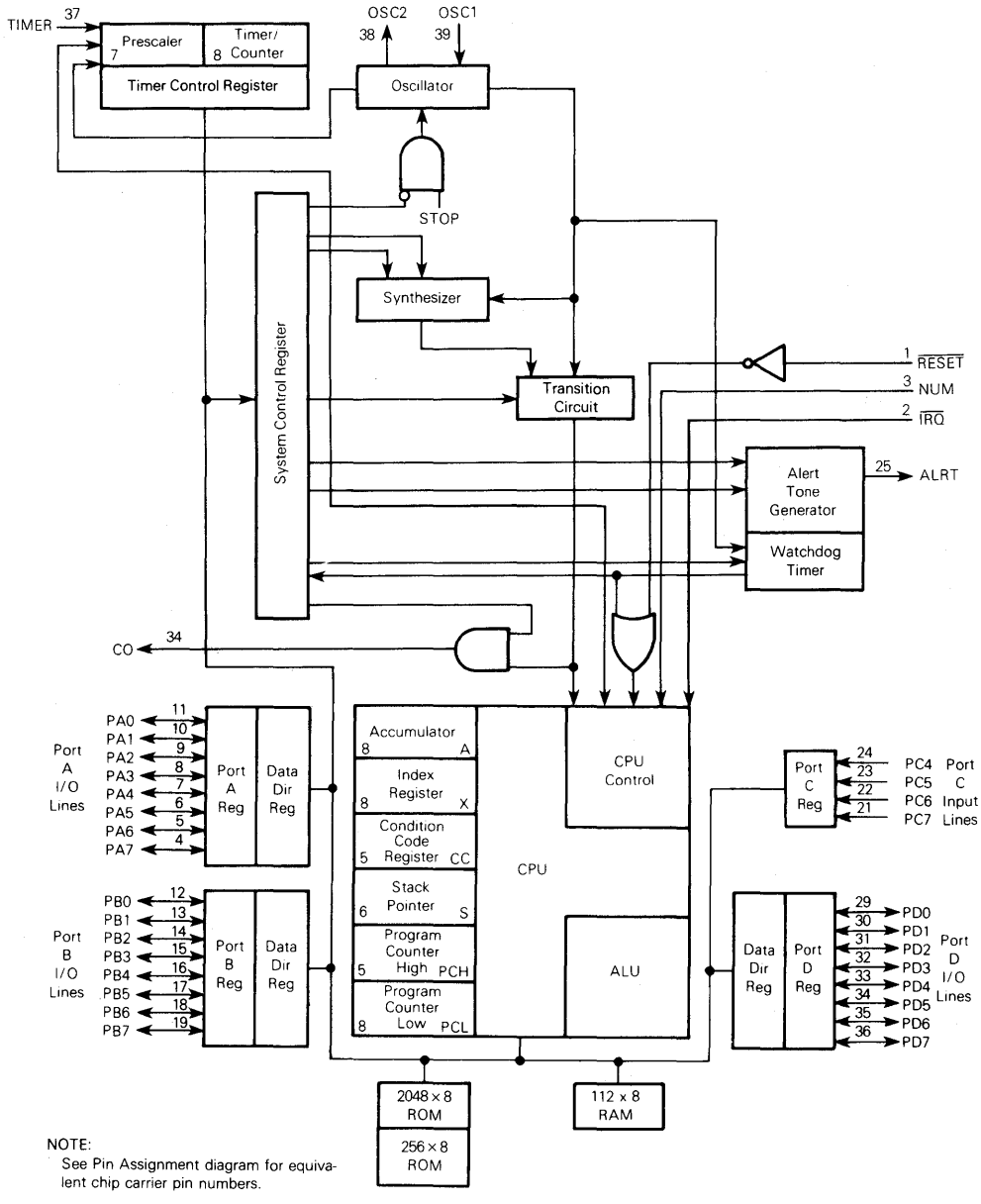
PIN ASSIGNMENT




Pin numbers in parentheses represent equivalent Z suffix chip carrier pins.

This document contains information on a new product. Specifications and information herein are subject to change without notice.

FIGURE 1 — BLOCK DIAGRAM




NOTE:
See Pin Assignment diagram for equivalent chip carrier pin numbers.

**MAXIMUM RATINGS** (Voltages Referenced to $V_{SS}$)

| Ratings | Symbol | Value | Unit |
|---|---|---|---|
| Supply Voltage | $V_{DD}$ | −0.3 to +8.0 | V |
| All Input Voltages Except OSC1 | $V_{in}$ | $V_{SS}$ −0.5 to $V_{DD}$ +0.5 | V |
| Current Drain Per Pin Excluding $V_{DD}$ and $V_{SS}$ | I | 10 | mA |
| Operating Temperature Range | $T_A$ | 0 to 70 | °C |
| Storage Temperature Range | $T_{stg}$ | −55 to +150 | °C |

This device contains circuitry to protect the inputs against damage due to high static voltages of electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high impedance circuit. For proper operation it is recommended that $V_{in}$ and $V_{out}$ be constrained to the range $V_{SS} \le (V_{in}$ or $V_{out}) \le V_{DD}$. Reliability of operation is enhanced if unused inputs are connected to an appropriate logic voltage level (e.g., either $V_{SS}$ or $V_{DD}$).

**THERMAL CHARACTERISTICS**

| Characteristics | Symbol | Value | Unit |
|---|---|---|---|
| Thermal Resistance | | | |
| Plastic | | 100 | |
| Cerdip | $\theta_{JA}$ | 60 | °C/W |
| Ceramic | | 50 | |
| Chip Carrier | | TBD | |

3

**DC ELECTRICAL CHARACTERISTICS** ($V_{DD}$=5.0 Vdc ±10%, $V_{SS}$=0 Vdc, $T_A$=0°C to 70°C unless otherwise noted)

| Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| Output Voltage, $I_{Load} \le 10.0\ \mu A$ | $V_{OL}$ | – | 0.1 | V |
| | $V_{OH}$ | $V_{DD}$−0.1 | – | V |
| Output High Voltage | | | | |
| Alert Tone Generator ($I_{Load}$= −2 mA) | | 2.4 | – | |
| CO ($I_{Load}$= −4 mA) | $V_{OH}$ | 2.4 | – | V |
| PA0-PA7, PD0-PD7 ($I_{Load}$= −2 mA) | | 2.4 | – | |
| PB0-PB7 ($I_{Load}$= −100 μA) | | 2.4 | – | |
| Output Low Voltage | | | | |
| Alert Tone Generator ($I_{Load}$=900 μA) | | – | 0.4 | |
| CO ($I_{Load}$=800 μA) | $V_{OL}$ | – | 0.4 | V |
| PA0-PA7, PB0-PB7, PD0-PD7 ($I_{Load}$=800 μA) | | – | 0.4 | |
| Input High Voltage | | | | |
| PA0-PA7, PB0-PB7, PC4-PC7, PD0-PD7 | | $V_{DD}$−2.0 | – | |
| TIMER, $\overline{IRQ}$, $\overline{RESET}$ | $V_{IH}$ | $V_{DD}$−0.8 | – | V |
| OSC1 | | $V_{DD}$−0.8 | – | |
| Input Low Voltage All Inputs | $V_{IL}$ | – | 0.8 | V |
| Total Supply Current ($C_L$=50 pF on Ports, no dc Loads, $t_{cyc}$=1μs) | | | | |
| RUN ($V_{IL}$=0.2 V, $V_{IH}$=$V_{DD}$−0.2 V) | | – | TBD | mA |
| WAIT | $I_{DD}$ | – | TBD | mA |
| STOP | | – | TBD | μA |
| I/O Port Input Leakage | $I_{IL}$ | – | ±10 | μA |
| Input Curent | | | | |
| $\overline{RESET}$, $\overline{IRQ}$, TIMER, OSC1 | $I_{in}$ | – | ±1 | μA |
| Capacitance | | | | |
| Ports | $C_{out}$ | – | 12 | pF |
| $\overline{RESET}$, $\overline{IRQ}$, TIMER, OSC1 | $C_{in}$ | – | 8 | pF |

TBD = To be determined.

NOTE:
Test conditions for $I_{DD}$ are as follows:
All ports programmed as inputs
$V_{IL}$=0.2 V (PA0-PA7, PB0-PB7, PC4-PC7, PD0-PD7)
$V_{IH}$=$V_{DD}$−0.2 V for $\overline{RESET}$, $\overline{IRQ}$, and TIMER
OSC1 input as a squarewave from 0.2 V to $V_{DD}$−0.2 V
OSC2 output load=20 pF (WAIT $I_{DD}$ is affected linearly by the OSC2 capacitance. STOP $I_{DD}$ is also affected linearly by this capacitance if the oscillator is not killed.)

**DC ELECTRICAL CHARACTERISTICS** ($V_{DD}$=3.0 Vdc, $V_{SS}$=0 Vdc, $T_A$=0°C to 70°C unless otherwise noted)

| Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| Output Voltage, $I_{Load} \leq 10.0\ \mu A$ | $V_{OL}$<br>$V_{OH}$ | –<br>$V_{DD}-0.1$ | 0.1<br>– | V<br>V |
| Output High Voltage<br>Alert Tone Generator ($I_{Load}=-0.5$ mA)<br>CO ($I_{Load}=-1.0$ mA)<br>PA0-PA7, PD0-PD7 ($I_{Load}=-0.5$ mA)<br>PB0-PB7 ($I_{Load}=-50\ \mu A$) | $V_{OH}$ | <br>1.4<br>1.4<br>1.4<br>1.4 | <br>–<br>–<br>–<br>– | <br>V |
| Output Low Voltage<br>Alert Tone Generator ($I_{Load}=900\ \mu A$)<br>CO ($I_{Load}=800\ \mu A$)<br>PA0-PA7, PB0-PB7, PD0-PD7 ($I_{Load}=800\ \mu A$) | $V_{OL}$ | <br>–<br>–<br>– | <br>0.3<br>0.3<br>0.3 | <br>V |
| Input High Voltage<br>PA0-PA7, PB0-PB7, PC4-PC7, PD0-PD7<br>TIMER, $\overline{IRQ}$, $\overline{RESET}$<br>OSC1 | $V_{IH}$ | <br>$V_{DD}-0.3$<br>$V_{DD}-0.3$<br>$V_{DD}-0.3$ | <br>–<br>–<br>– | <br>V |
| Input Low Voltage All Inputs | $V_{IL}$ | – | 0.3 | V |
| Total Supply Current ($C_L$=50 pF on Ports no dc Loads, $t_{cyc}=5\ \mu s$)<br>RUN ($V_{IL}$=0.2 V, $V_{IH}=V_{DD}-0.2$ V)<br>WAIT<br>STOP | $I_{DD}$ | <br>–<br>–<br>– | <br>TBD<br>TBD<br>TBD | <br>mA<br>mA<br>$\mu A$ |
| I/O Ports Input Leakage Current | $I_{IL}$ | – | ±10 | $\mu A$ |
| Input Current<br>$\overline{RESET}$, $\overline{IRQ}$, TIMER, OSC1 | $I_{in}$ | – | ±1 | $\mu A$ |
| Capacitance<br>Ports<br>$\overline{RESET}$, $\overline{IRQ}$, TIMER, OSC1 | $C_{out}$<br>$C_{in}$ | <br>– | <br>12<br>8 | <br>pF<br>pF |

TBD=To be determined.

NOTE:

Test conditions for $I_{DD}$ are as follows:

All ports programmed as inputs

$V_{IL}$=0.2 V (PA0-PA7, PB0-PB7, PC4-PC7, PD0-PD7)

$V_{IH}=V_{DD}-0.2$ V for $\overline{RESET}$, $\overline{IRQ}$, and TIMER

OSC1 input is a squarewave from 0.2 V to $V_{DD}-0.2$ V

OSC2 output load=20 pF (WAIT and STOP $I_{DD}$ are affected linearly by the OSC2 capacitance.)

**TABLE 1 — CONTROL TIMING**

($V_{DD}$=5.0 Vdc ±10%, $V_{SS}$=0 Vdc, $T_A$=0°C to 70°C)

| Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| Crystal Oscillator Startup Time (See Figure 5) | $t_{OXOV}$ | – | TBD | ms |
| Stop Recovery Startup Time (Crystal Oscillator) (See Figure 6) | $t_{ILCH}$ | – | TBD | ms |
| Timer Pulse Width (See Figure 4) | $t_{TH}$, $t_{TL}$ | 0.5 | – | $t_{cyc}$ |
| $\overline{RESET}$ Pulse Width (See Figure 5) | $t_{RL}$ | 1.5 | – | $t_{cyc}$ |
| Timer Period (See Figure 4) | $t_{TLTL}$ | 1.0 | – | $t_{cyc}$ |
| Interrupt Pulse Width Low (See Figure 15) | $t_{ILIH}$ | 1.0 | – | $t_{cyc}$ |
| Interrupt Pulse Period (See Figure 15) | $t_{ILIL}$ | * | – | $t_{cyc}$ |
| OSC1 Pulse Width | $t_{OH}$, $t_{OL}$ | TBD | – | ns |
| Cycle Time | $t_{cyc}$ | 1000 | – | ns |
| Frequency of Operation<br>Crystal<br>Synthesizer | <br>$f_{osc}$<br>$f_{synth}$ | <br>30<br>0.5 | <br>50<br>2.0 | <br>kHz<br>MHz |

*The minimum period $t_{ILIL}$ should not be less than the number of $t_{cyc}$ cycles it takes to execute the interrupt service routine plus 20 $t_{cyc}$ cycles.

TABLE 2 — CONTROL TIMING
($V_{DD}$ = 3.0 Vdc, $V_{SS}$ = 0 Vdc, $T_A$ = 0°C to 70°C)

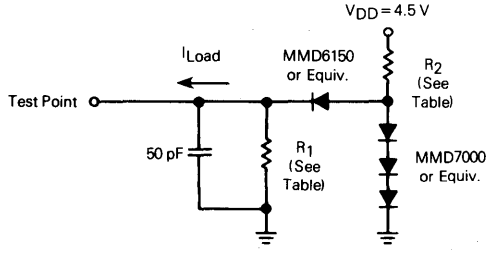
| Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| Crystal Oscillator Startup Time (See Figure 5) | $t_{OXOV}$ | – | TBD | ms |
| Stop Recovery Startup Time (Crystal Oscillator) (See Figure 6) | $t_{ILCH}$ | – | TBD | ms |
| Timer Pulse Width (See Figure 4) | $t_{TH}$, $t_{TL}$ | 0.5 | – | $t_{cyc}$ |
| $\overline{\text{RESET}}$ Pulse Width (See Figure 5) | $t_{RL}$ | 1.5 | – | $t_{cyc}$ |
| Timer Period (See Figure 4) | $t_{TLTL}$ | 1.0 | – | $t_{cyc}$ |
| Interrupt Pulse Width Low (See Figure 14) | $t_{ILIH}$ | 1.0 | – | $t_{cyc}$ |
| Interrupt Pulse Period (See Figure 14) | $t_{ILIL}$ | * | – | $t_{cyc}$ |
| OSC1 Pulse Width | $t_{OH}$, $t_{OL}$ | TBD | – | ns |
| Cycle Time | $t_{cyc}$ | 5000 | – | ns |
| Frequency of Operation<br>Crystal<br>Synthesizer | <br>$f_{osc}$<br>$f_{synth}$ | <br>30<br>120 | <br>50<br>600 | <br>kHz<br>kHz |

*The minimum period $t_{ILIL}$ should not be less than the number of $t_{cyc}$ cycles it takes to execute the interrupt service routines plus 20 $t_{cyc}$ cycles.

3

FIGURE 2 — EQUIVALENT TEST LOAD

| Port | R1 | R2 |
|---|---|---|
| B | 24.3 kΩ | 4.32 kΩ |
| A and D | 1.21 kΩ | 3.1 kΩ |

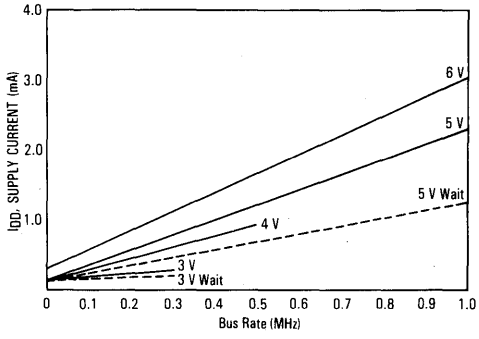



## TYPICAL CURRENT CALCULATIONS

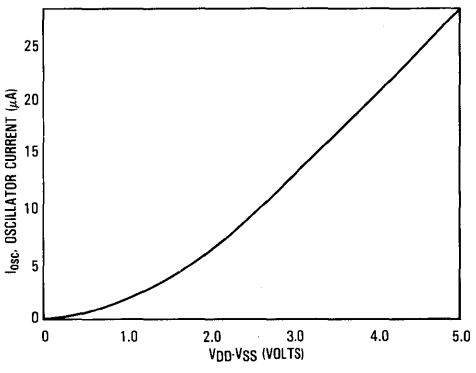
The operating current of the MCU ($I_{DD}$) is a function of supply voltage, bus rate, capacitive loading on any active pins (i.e., OSC1, OSC2, etc.), the processor state (RUN, WAIT, or STOP), the synthesizer state (ON or OFF), and the resistive loading on all outputs. Inputs, such as input ports can also cause significant increases in currents if they are placed in the active region of the input device. Because of this, inputs should never be allowed to simply "float". It is impossible to determine a "typical" $I_{DD}$ for a particular application without first knowing all of the above conditions and their corresponding currents. Thus, some "typical" current curves are provided in Figure 3 (a, b, and c). It should be emphasized that these are only approximations and no minimums or maximums are implied.

FIGURE 3 — TYPICAL OPERATING CURRENT vs FREQUENCY/VOLTAGE

(a) — Processor Current vs Bus Frequency




(b) — Typical Crystal Oscillator Current vs Voltage




(c) — Typical Synthesizer Current vs Frequency

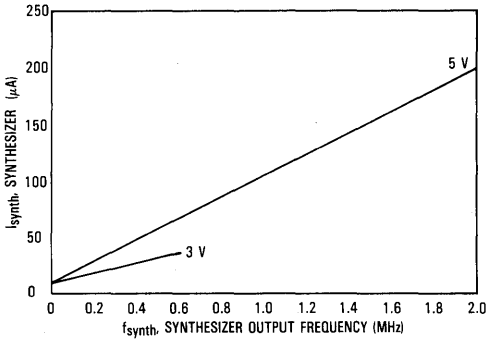



FIGURE 4 — TIMER RELATIONSHIPS

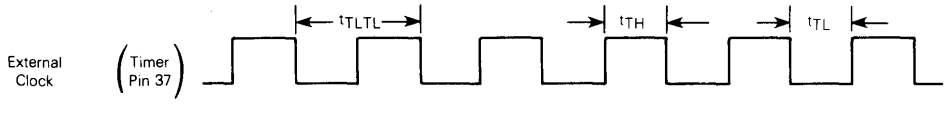
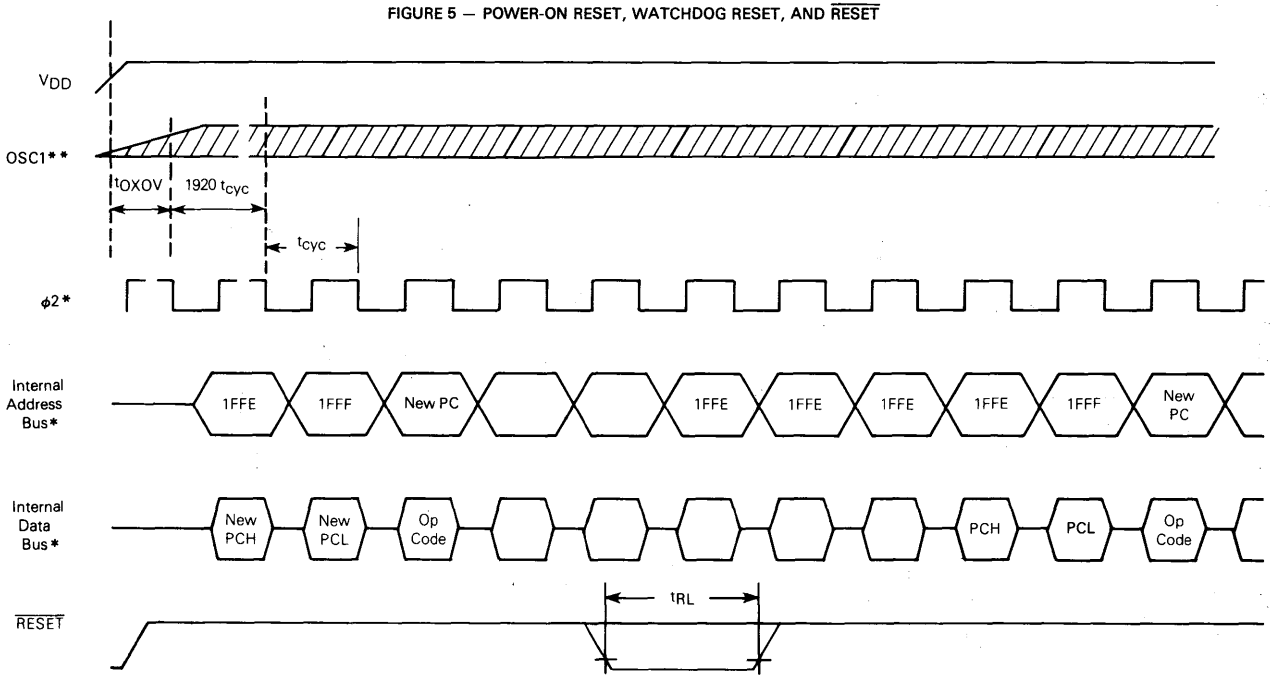
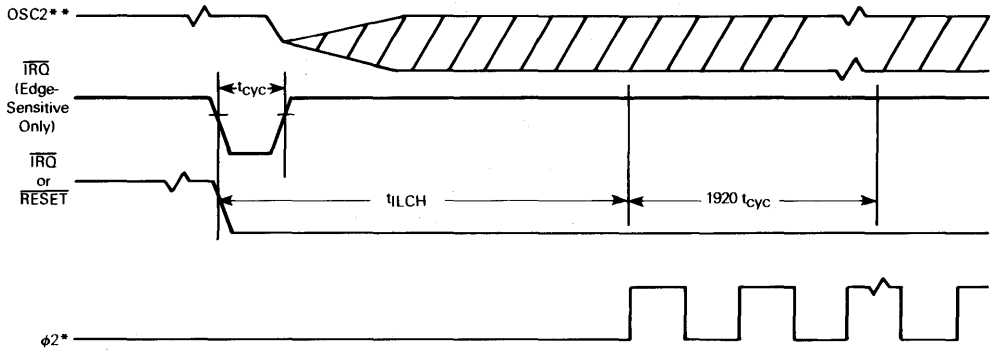


FIGURE 5 — POWER-ON RESET, WATCHDOG RESET, AND $\overline{\text{RESET}}$




*Internal timing signal and bus information not available externally.
**OSC1 line is not meant to represent frequency. It is only used to represent time.

FIGURE 6 — STOP RECOVERY AND POWER-ON RESET




*Internal timing signals not available externally.
**Represents the internal control of crystal oscillator.



## FUNCTIONAL PIN DESCRIPTION

### $V_{DD}$, $V_{SS}$, AND SYNTH $V_{SS}$

Power is supplied to the MCU using these pins. $V_{DD}$ is power and $V_{SS}$ is ground. A separate ground is provided for the synthesizer which must be at the same potential as $V_{SS}$. These grounds (synthesizer $V_{SS}$ and $V_{SS}$) may be bypassed independently to minimize noise if necessary.

### IRQ (MASKABLE INTERRUPT REQUEST)

IRQ is a mask programmable option which provides two different choices of interrupt triggering sensitivity. These options are: 1) negative edge-sensitive triggering only, or 2) both negative edge-sensitive and level-sensitive triggering. In the latter case, either type of input to the IRQ pin will produce the interrupt. The MCU completes the current instruction before it responds to the interrupt request. When the IRQ pin goes low for at least one $t_{cyc}$, a logic one is latched internally to signify an interrupt has been requested. When the MCU completes its current instruction, the interrupt latch is tested. If the interrupt latch contains a logic one and the interrupt mask bit (I bit) in the condition code register is clear, the MCU then begins the interrupt sequence.

If the option is selected to include level-sensitive triggering, then the IRQ input requires an external resistor to $V_{DD}$ for "wire-OR" operation. See **INTERRUPTS** for more detail.

### RESET

The RESET input is not required for startup but can be used to reset the MCU internal state and provide an orderly software startup procedure. Refer to **RESETS** for a detailed description.

### TIMER

The TIMER input may be used as an external clock for the on-chip timer. Refer to **TIMER** for additional information about the timer circuitry.

### OSC1, OSC2

The MC146805H2 is configured to accept a crystal to control the internal oscillator. An external clock may also be used. These are discussed below.

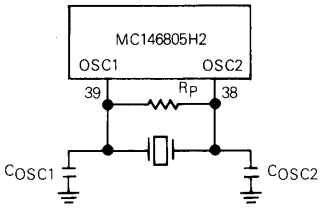
**CRYSTAL** — The circuit shown in Figure 7(b) is recommended when using a crystal. The internal oscillator is designed to interface with a parallel resonant quartz crystal resonator in the frequency range specified for $f_{osc}$ in Table 1 and Table 2 control timing. Using an external CMOS oscillator is recommended when crystals outside the specified ranges are to be used. The crystal and components should be mounted as close as possible to the input pins to minimize output distortion and startup stabilization time.

**EXTERNAL CLOCK** — An external clock should be applied to the OSC1 input with OSC2 not connected, as shown in Figure 7(d). The $t_{OXOV}$ or $t_{ILCH}$ specifications do not apply when using an external clock input.

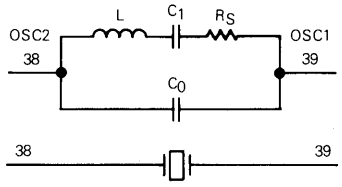
FIGURE 7 — OSCILLATOR CONNECTIONS

| | Vendor A | Vendor B | Units |
|---|---|---|---|
| $R_{S(Max)}$ | 50 | 23 | kΩ |
| $C_0$ | 0.8 | 1.5 | pF |
| $C_1$ | — | 2.35 | pF |
| $C_{OSC1}$ | 10 | 35 | pF |
| $C_{OSC2}$ | 2.5-10 | 5-30 | pF |
| Q | 60 | 90 | K |
| $R_P$ | 2-10 | 2-10 | MΩ |

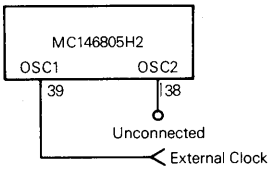
(a) Typical Crystal Parameters @ 32.768 kHz




(b) Crystal Oscillator Connections




(c) Equivalent Crystal Circuit




(d) External Clock Source Connections

**CO**

This pin provides a clock output which represents the clock input to the CPU; however, it is twice the frequency of the bus rate since the CPU divides the clock by two to obtain the bus rate. It can be used to provide an external synchronizing clock or as a test point for checking the on-chip clock input to the CPU.

**ALRT**

This output provides one of three tone signals to drive an external amplifier whenever the tone generator is activated by the microcomputer program. Whenever the tone generator is turned off, this pin represents a high impedance. Refer to **AUDIO ALERT TONE GENERATOR** for more information.

**XFC**

This pin provides a means for connecting an external capacitor to the synthesizer phase lock loop filter. Refer to **PHASE LOCK LOOP** for additional information concerning this capacitor.

**PA0-PA7**

These eight I/O lines comprise port A. The state of any pin is software programmable. Refer to **INPUT/OUTPUT PROGRAMMING** for a description of I/O programming.

**PB0-PB7**

These eight lines comprise port B. The state of any pin is software programmable. Refer to **INPUT/OUTPUT PROGRAMMING** for a description of I/O programming.

**PC4-PC7**

These four lines comprise port C, a fixed input port. When port C is read, the four least significant bits on the data bus are zeros. There is no data direction register associated with port C.

**PD0-PD7**

These eight lines comprise port D. The state of any pin is software programmable. Refer to **INPUT/OUTPUT PROGRAMMING** for a description of I/O programming.

## INPUT/OUTPUT PROGRAMMING

Any port A, B, or D pin may be software programmed as an input or output by the state of the corresponding bit in the port data direction register (DDR). A particular port A, B, or D pin is configured as an output if its corresponding DDR bit is set to a logic one. A pin is configured as an input if its corresponding DDR bit is cleared to a logic zero. At reset, all DDRs are cleared, which configures all port A, B, and D pins as inputs. Port C is input only. A particular port A, B, or D pin configured as an output will output the data in the corresponding bit of its port data latch. Refer to Figure 8 and Table 3.



FIGURE 8 — TYPICAL PORT A, B, OR D I/O CIRCUITRY

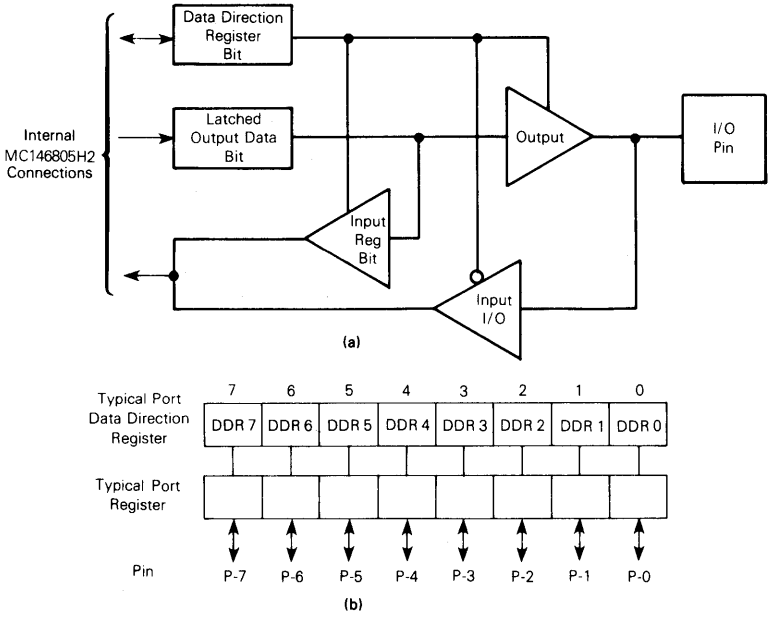


TABLE 3 — PORT A, B, OR D I/O PIN FUNCTIONS

| R/$\overline{W}$* | DDR | I/O Pin Function |
|---|---|---|
| 0 | 0 | The I/O pin is in input mode. Data is written into the output data latch. |
| 0 | 1 | Data is written into the output data latch and output to the I/O pin. |
| 1 | 0 | The state of the I/O pin is read. |
| 1 | 1 | The I/O pin is in an output mode. The output data latch is read. |

*R/$\overline{W}$ is an internal signal.

## SELF-CHECK

The MC146805H2 self-check is performed using the circuit in Figure 9. Self-check is initiated by connecting the NUM and TIMER pins to a logic one and then executing a reset. After reset, five subroutines are called that execute the following tests:

I/O — Functionally exercises ports A, B, C, D
RAM — Walking bit test
ROM — Exclusive OR with odd ones parity result
Timer — Functionally exercise timer
Interrupts — Functionally exercise external and timer interrupts

Self-check results are shown in Table 4. The following subroutines are available to user programs and do not require any external hardware.

FIGURE 9 — SELF-CHECK CIRCUIT

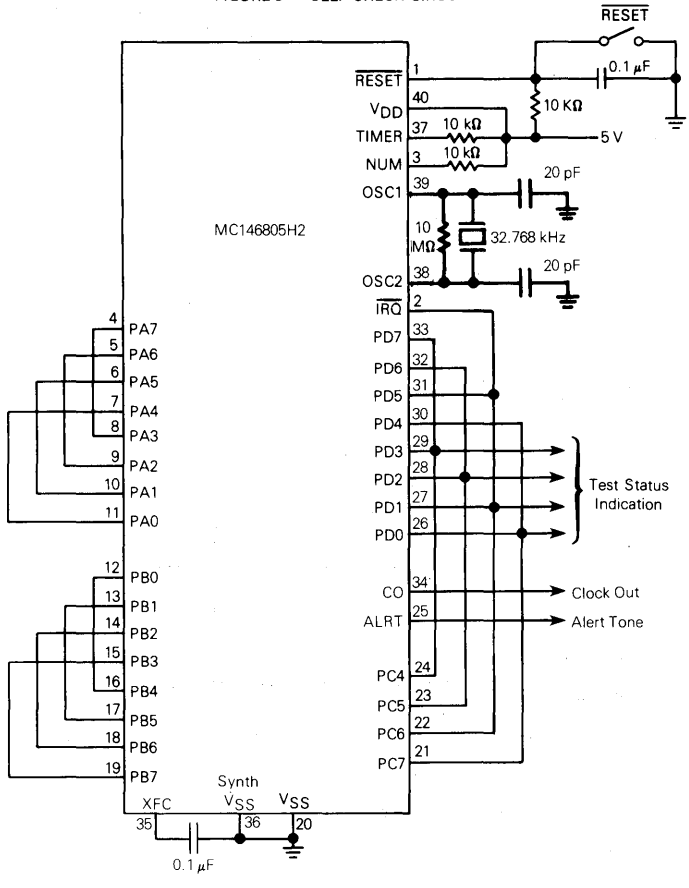


TABLE 4 — SELF-CHECK RESULTS

| PD3 | PD2 | PD1 | PD0 | Remarks |
|---|---|---|---|---|
| 1 | 0 | 1 | 0 | Bad I/O |
| 1 | 0 | 1 | 1 | Bad Timer |
| 1 | 1 | 0 | 0 | Bad RAM |
| 1 | 1 | 0 | 1 | Bad ROM |
| 1 | 1 | 1 | 0 | Bad Interrupt or Request Flag |
| Cycling | | | | Good Part |
| All Others | | | | Bad Part |

### RAM SELF-CHECK SUBROUTINES

Returns with the Z bit clear if any error is detected; otherwise the Z bit is set.

The RAM test must be called with the stack pointer at $007F. When run, the test checks every RAM cell except for $007F and $007E which are assumed to contain the return address.

A and X are modified. All RAM locations except the top two are modified.

### ROM CHECKSUM SUBROUTINE

Returns with Z bit cleared if any error was found; otherwise Z=1. X=0 on return, and A is zero if the test passed. RAM locations $0040-$0043 are overwritten.

### TIMER TEST SUBROUTINE

Returns with Z bit cleared if any error was found; otherwise Z=1.

This routine runs a simple test on the timer. In order to work correctly as a user subroutine, the internal clock must be the clocking source and interrupts must be disabled. Also, on exit, the clock will be running and the interrupt mask not set so the caller must protect himself from interrupts if necessary.

The A and X register contents are lost since this routine uses them in determining how many times the clock counts in 128 cycles. The number of counts should be a power of two since the prescaler is a power of two. If not, the timer probably is not counting correctly. The routine also detects if the timer is running at all.

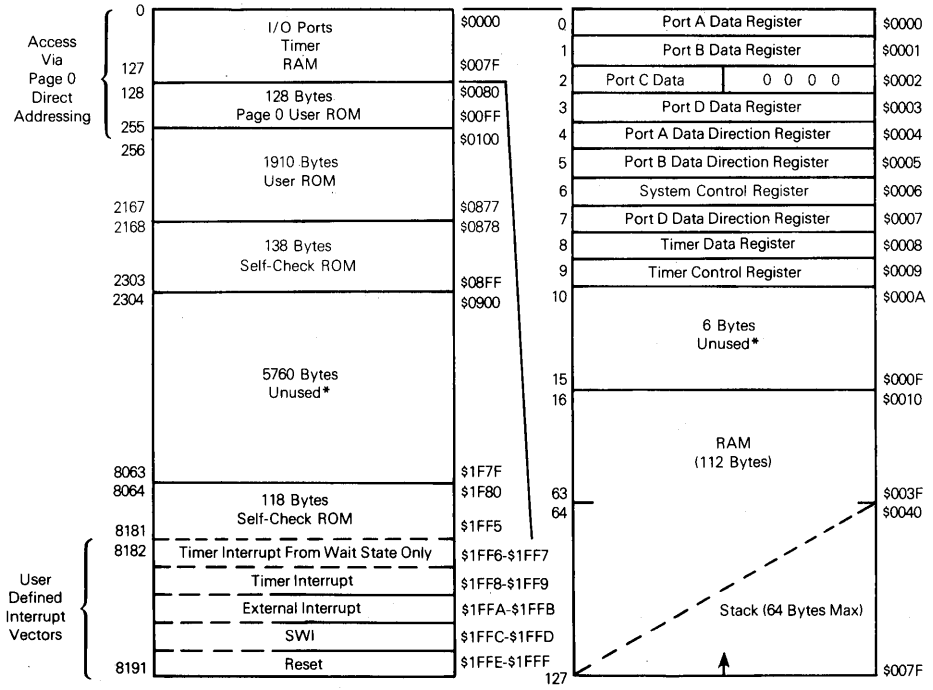
## MEMORY

The MC146805H2 has a total address space of 8192 bytes of memory and I/O registers. The address space is shown in Figure 10.

The first 128 bytes of memory (first half of page zero) are comprised of the I/O port locations, timer locations, and 112 bytes of RAM. The next 2038 bytes (including the 128 bytes of the second half of page zero) comprise the user ROM. The 10 highest address bytes contain the reset and the interrupt vectors.

The stack pointer is used to address data stored on the stack. Data is stored on the stack during interrupts and subroutine calls. At power up, the stack pointer is set to $007F and it is decremented as data is pushed onto the stack. When data is removed from the stack, the stack pointer is incremented. A maximum of 64 bytes of RAM is available for stack usage. Since most programs use only a small part of the allocated stack locations for interrupts and/or subroutine stacking purposes, the unused bytes are usable for program data storage.

FIGURE 10 — ADDRESS MAP




*Reads of unused locations undefined.

## REGISTERS

The MC146805H2 contains five registers, as shown in the programming model of Figure 11. The interrupt stacking order is shown in Figure 12.

### ACCUMULATOR (A)

The accumulator is an 8-bit general purpose register used to hold operands, results of the arithmetic calculations, and data manipulations.

### INDEX REGISTER (X)

The X register is an 8-bit register which is used during the indexed modes of addressing. It provides an 8-bit value which is used to create an effective address. The index register is also used for data manipulations with the read-modify-write type of instructions and as a temporary storage register when not performing addressing operations.

### PROGRAM COUNTER (PC)

The program counter is a 13-bit register that contains the address of the next instruction to be executed by the processor.

### STACK POINTER (SP)

The stack pointer is a 13-bit register containing the address of the next free location on the stack. When accessing memory, the seven most significant bits are permanently configured to 0000001. These seven bits are appended to the six least significant register bits to produce an address within the range of $007F to $0040. The stack area of RAM is used to store the return address on subroutine calls and the machine state during interrupts. During external or power-on reset, and during a reset stack pointer (RSP) instruction, the stack pointer is set to its upper limit ($007F). Nested interrupt and/or subroutines may use up to 64 (decimal) locations, beyond which the stack pointer wraps around and points to its upper limit thereby losing the previously stored information. A subroutine call occupies two RAM bytes on the stack, while an interrupt uses five RAM bytes.

### CONDITION CODE REGISTER (CC)

The condition code register is a 5-bit register which indicates the results of the instruction just executed. These bits can be individually tested by a program and specific action taken as a result of their state. Each bit is explained in the following paragraphs.

FIGURE 11 — PROGRAMMING MODEL

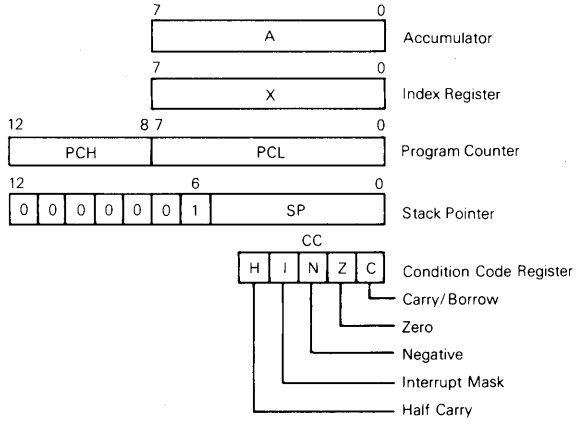
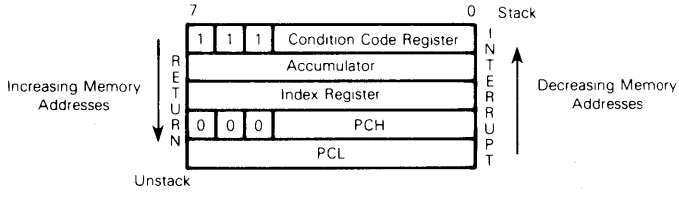



FIGURE 12 — STACKING ORDER




NOTE: Since the Stack Pointer decrements during pushes, the PCL is stacked first, followed by PCH, etc. Pulling from the stack is in the reverse order.

**HALF CARRY BIT (H)** — The H bit is set to a one when a carry occurs between bits 3 and 4 of the ALU and during an ADD or ADC instruction. The H bit is useful in binary coded decimal subroutines.

**INTERRUPT MASK BIT (I)** — When the I bit is set, both the external interrupt and the timer interrupt are disabled. Clearing this bit enables the above interrupts. If an interrupt occurs while the I bit is set, the interrupt is latched and is processed after the I bit is next cleared.

**NEGATIVE (N)** — When set, this bit indicates that the results of the last arithmetic, logical, or data manipulation is negative (bit 7 in the result is a logic one).

**ZERO (Z)** — When set, this bit indicates that the result of the last arithmetic, logical, or data manipulation is zero.

**CARRY/BORROW (C)** — When set, this bit indicates that a carry or borrow out of the arithmetic logic unit (ALU) occurred during the last arithmetic operation. This bit is also affected during bit test and branch instructions, shifts, and rotates.

## RESETS

The MC146805H2 has three reset modes: an active low external reset pin ($\overline{\text{RESET}}$), a power-on reset function, and a dead-man timer reset function; refer to Figure 5.

### $\overline{\text{RESET}}$

The $\overline{\text{RESET}}$ input pin is used to reset the MCU to provide an orderly software startup procedure. When using the external reset mode, the $\overline{\text{RESET}}$ pin must stay low for a minimum of one $t_{CYC}$. The $\overline{\text{RESET}}$ pin contains an internal Schmitt trigger as part of its input (internally) to improve its noise immunity.

### POWER-ON RESET

The power-on reset occurs when a positive transition is detected on $V_{DD}$. The power-on reset is used strictly for

power turn-on conditions and should not be used to detect any drops in the power supply voltage. There is no provision for a power-down reset. The power-on circuitry provides for a 1920 $t_{CYC}$ delay from the time that the oscillator becomes active. If the external $\overline{RESET}$ pin is low at the end of the 1920 $t_{CYC}$ time out, the processor remains in the reset condition until $\overline{RESET}$ goes high.

### WATCHDOG TIMER

The watchdog timer contains an 18-stage divider which is clocked by the crystal oscillator output. A program controlled input (SCR4) from the system control register clears the watchdog timer as described in **SYSTEM CONTROL REGISTER.** Unless the watchdog timer is periodically cleared by the system program it will time out and reset the MCU.

Since the watchdog timer is connected directly to the crystal oscillator it is not affected by the oscillator gating as discussed in **SYSTEM CONTROL REGISTER.** If the oscillator is left running, the watchdog timer will time out and reset the MCU. This is a safety feature to preclude the MCU from becoming "lost", and inadvertently stopping the oscillator and thereby "killing" itself. As will be discussed in **SYSTEM CONTROL REGISTER,** the SCR3 bit is heavily protected to protect the system during a loss of MCU control.

### RESET CONDITIONS

Either of the three types of reset conditions causes the following to occur:

- Timer control register interrupt request bit TCR7 is cleared to a logic zero to preclude premature timer interrupts.
- Timer control register interrupt mask bit TCR6 is set to a logic one to preclude timer interrupt processing.
- All data direction register bits are cleared to logic zeros to define all ports as input.
- Stack pointer is preset to its upper limit, $007F.
- The internal address bus is forced to the reset vector ($1FFE, $1FFF).
- Condition code register interrupt mask bit (I) is set to a logic one to mask any external interrupts.
- STOP and WAIT latches are cleared to place MCU in normal operation.
- External interrupt latch is cleared to ensure no external interrupt is processed.
- System control register bits SCR6, SCR7, and SCR2 are cleared; however, bits SCR5, SCR3, SCR1, and SCR0 are set. Bit SCR4 could be either set or cleared as discussed in the **SYSTEM CONTROL REGISTER.**

All other functions, such as other registers, the timer, etc. are not cleared by the reset conditions.

## INTERRUPTS

Systems often require that normal processing be interrupted so that some external event may be serviced. The MC146805H2 may be interrupted by one of three different methods: either one of two maskable hardware interrupts (external input or timer) or a nonmaskable software interrupt (SWI).

Interrupts cause the processor registers to be saved on the stack and the interrupt mask (I bit) set to prevent additional interrupts. The RTI instruction causes the register contents to be recovered from the stack followed by a return to normal processing. The stack order is shown in Figure 12.

Unlike $\overline{RESET}$, hardware interrupts do not cause the current instruction execution to be halted, but are considered pending until the current instruction execution is completed.

**NOTE**

The current instruction is considered to be the one already fetched and being operated on.

When the current instruction is complete, the processor checks all pending hardware interrupts and, if an interrupt is pending and is unmasked, proceeds with interrupt processing; otherwise, the next instruction is fetched and executed. Note that masked interrupts are latched for later interrupt servicing.

If both an external interrupt and a timer interrupt are pending at the end of an instruction execution, the external interrupt is serviced first. The SWI is executed the same as any other instruction. Refer to Figure 13 for the interrupt and instruction processing sequence.

Table 5 shows the execution priority of the $\overline{RESET}$, $\overline{IRQ}$, timer interrupts, and the software interrupt, SWI. Two conditions are shown, one with the I bit set and the other with the I bit clear; however, in either case $\overline{RESET}$ has the highest priority of execution. If the I bit is set as per Table 5(a), the second highest priority is assigned to SWI. This is illustrated in Figure 13 which shows that the $\overline{IRQ}$ or timer interrupts are not executed when the I bit is set. If the I bit is clear as per Table 5(b), the priorities change in that the next instruction (including SWI) is not fetched until after the $\overline{IRQ}$ and timer interrupts have been recognized (and serviced). Also, when the I bit is clear, if both $\overline{IRQ}$ and timer interrupts are pending, the $\overline{IRQ}$ interrupt is always serviced before the timer interrupt.

**NOTE**

The conditions for Table 5 assume that, except for $\overline{RESET}$, the current instruction is completed; thus the MCU is at an instruction boundary. Processing is such that at the end of the current instruction, the I bit is tested and if set the next instruction (including SWI) is fetched. If the I bit is cleared, the hardware interrupt latches are tested, and if no hardware interrupt is pending, the program falls through and the next instruction is fetched.

TABLE 5 — INTERRUPT INSTRUCTION EXECUTION PRIORITY AND VECTOR ADDRESS

(a) I Bit Set

| Interrupt/Instruction | Priority | Vector Address |
|---|---|---|
| $\overline{\text{RESET}}$ | 1 | $1FFE-$1FFF |
| SWI | 2 | $1FFC-$1FFD |

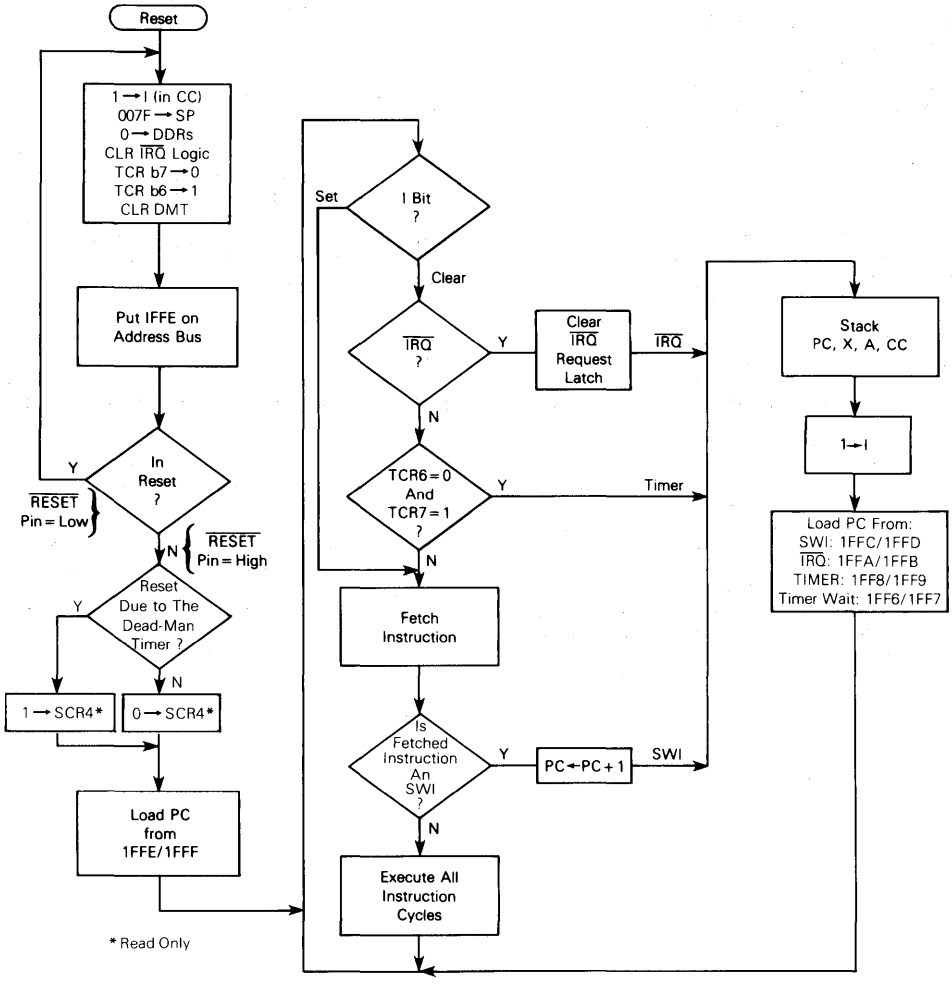
NOTE: $\overline{\text{IRQ}}$ and Timer Interrupts are not executed when the I bit is set; therefore, they are not shown.

(b) I Bit Clear

| Interrupt/Instruction | Priority | Vector Address |
|---|---|---|
| $\overline{\text{RESET}}$ | 1 | $1FFE-$1FFF |
| $\overline{\text{IRQ}}$ | 2 | $1FFA-$1FFB |
| Timer | 3 | $1FF8-$1FF9<br>$1FF6-$1FF7* |
| SWI | 4 | $1FFC-$1FFD |

*The timer vector address from the WAIT mode is $1FF6-$1FF7.

FIGURE 13 — RESET AND INTERRUPT PROCESSING FLOWCHART



### TIMER INTERRUPT

If the timer interrupt mask bit (TCR6) is cleared, then each time the timer decrements to zero (transitions from $01 to $00 to set TCR7) an interrupt request is generated. The actual processor interrupt is generated only if the interrupt mask bit (in the condition code register) is cleared. When the interrupt is recognized, the current state of the machine is pushed onto the stack and the interrupt mask bit in the condition code register is set. This masks further interrupts until the present one is serviced. The processor now vectors to the timer interrupt service routine. The address for this service routine is specified by the contents of $1FF8 and $1FF9 unless the processor is in a WAIT mode, in which case the contents of $1FF6 and $1FF7 specify the timer service routine address. Software must be used to clear the timer interrupt request bit (TCR7). At the end of the timer interrupt service routine, the software normally executes an RTI instruction which restores the machine state and starts executing the interrupted program.

The actual timer interrupt request can be delayed by controlling TCR6 (interrupt mask bit). If TCR6 is programmed to a logic one, no interrupt is generated even if TCR7 (interrupt request bit) is set. Then TCR6 can be programmed (after a specific time) to a logic zero to generate the actual timer interrupt request.

### EXTERNAL INTERRUPT

If the interrupt mask bit of the condition code register has been cleared and the external interrupt pin ($\overline{IRQ}$) has gone low, then the external interrupt is recognized. The action of the external interrupt is identical to the timer interrupt with the exception that the interrupt request input at $\overline{IRQ}$ is latched internally and the service routine address is specified by the contents of $1FFA and $1FFB. Either a level-sensitive and edge-sensitive trigger, or an edge-sensitive only trigger are available as a mask option. Figure 14 shows both a functional and mode timing diagram for the interrupt line. The timing diagram shows two different treatments of the interrupt line ($\overline{IRQ}$) to the MCU. The first method shows single pulses on the interrupt line space far enough apart to be serviced. The minimum time between pulses is a function of the length of the interrupt service routine. Once a pulse occurs, the next pulse should not occur until the MCU software has exited the routine (an RTI occurs). This time ($t_{ILIL}$) is obtained by adding 20 instruction cycles ($t_{cyc}$) to the total number of cycles it takes to complete the service routine including the RTI instruction; refer to Figure 15. The second configuration shows many interrupt lines "wire-ORed" to form the interrupts at the processor. Thus, if after servicing one interrupt the interrupt line remains low, then the next interrupt is recognized.

**NOTE**

The internal interrupt latch is cleared in the first part of the service routine; therefore, one (and only one) external interrupt pulse could be latched during $t_{ILIL}$ and serviced as soon as the I bit is cleared.

### SOFTWARE INTERRUPT

The software interrupt (SWI) is an executable instruction. The action of the software interrupt instruction is similar to the hardware interrupts. The software interrupt is executed regardless of the state of the interrupt mask bit in the condition code register. The service routine address is specified by the contents of memory locations $1FFC and $1FFD. See Figure 13 for interrupt and instruction processing flowchart.

## LOW-POWER MODES

### STOP

The STOP instruction places the MC146805H2 in its lowest power consumption mode. In the STOP mode, all internal processing and the timer operation are halted; refer to Figure 15.

During the STOP mode, timer control register (TCR) bits 6 and 7 are altered to remove any pending timer interrupt requests and to disable any further timer interrupts. The timer prescaler is cleared. The I bit in the condition code register is cleared to enable external interrupts. All other registers and memory remain unaltered. All input/output lines remain unchanged. The processor can only be brought out of the STOP mode by an external interrupt, reset, or dead-man timer timeout.

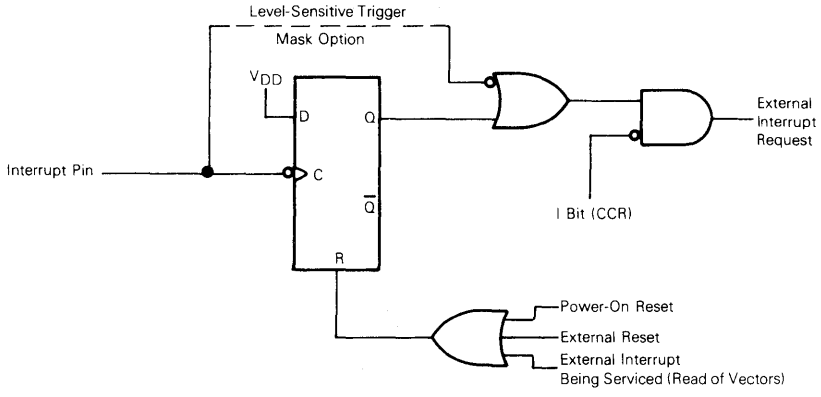
### WAIT

The WAIT instruction places the MC146805H2 in a low power consumption mode, but the WAIT mode consumes somewhat more power than the STOP mode. In the WAIT mode, the internal clock is disabled from all internal circuitry except for the timer; refer to Figure 16. Thus, all internal processing is halted; however, the timer continues to count normally.

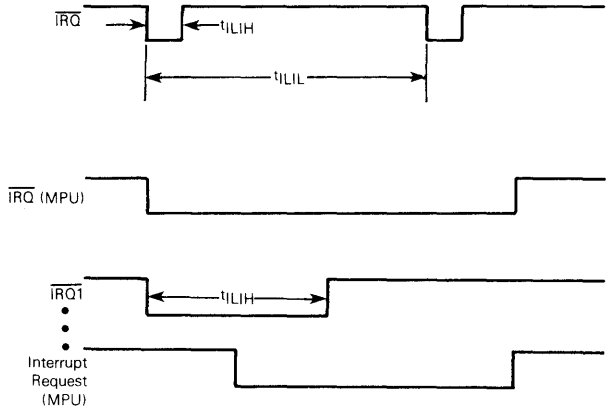
During the WAIT mode, the I bit in the condition code register is cleared to enable interrupts. All other registers, memory, and input/output lines remain in their previous state. The timer may be enabled to allow a periodic exit from the WAIT mode. If an external and a timer interrupt occur at the same time, the external interrupt is serviced first; then, if the timer interrupt request is not cleared in the external interrupt routine, the normal timer interrupt (not the timer wait interrupt) is serviced since the MCU is no longer in the WAIT mode.

FIGURE 14 — EXTERNAL INTERRUPT

**(a) Interrupt Functional Diagram**




**(b) Interrupt Mode Diagram**




**Edge-Sensitive Trigger Condition**

The minimum pulse width ($t_{ILIH}$) is one $t_{cyc}$. The period $t_{ILIL}$ should not be less than the number of $t_{cyc}$ cycles it takes to execute the interrupt service routine plus 20 $t_{cyc}$ cycles.

**Level-Sensitive Trigger Condition**

If after servicing an interrupt the $\overline{IRQ}$ remains low, then the next interrupt is recognized.

Normally used with Wire-ORed Connection

FIGURE 15 — STOP FUNCTION FLOWCHART

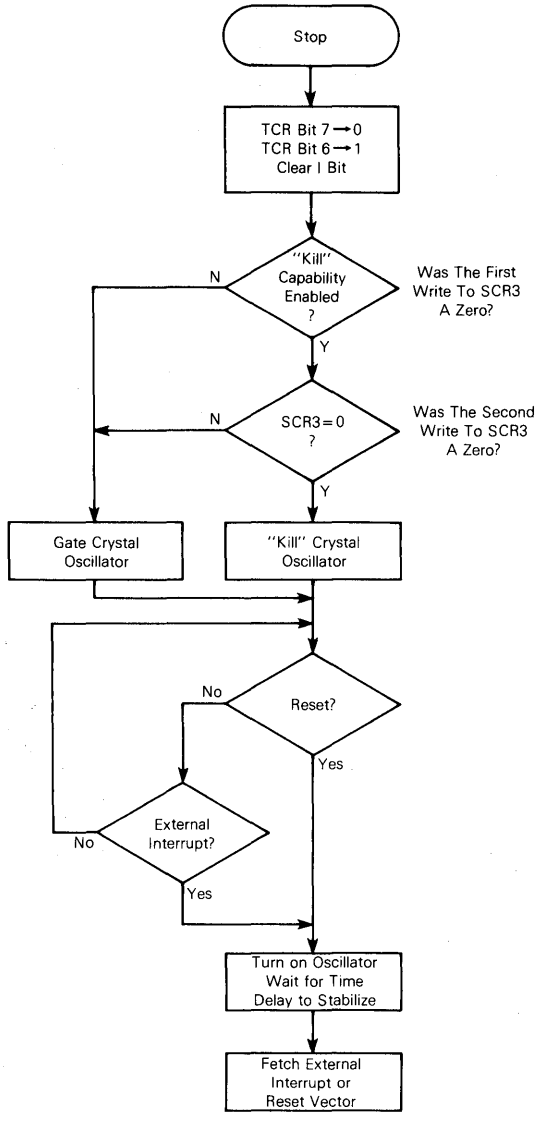
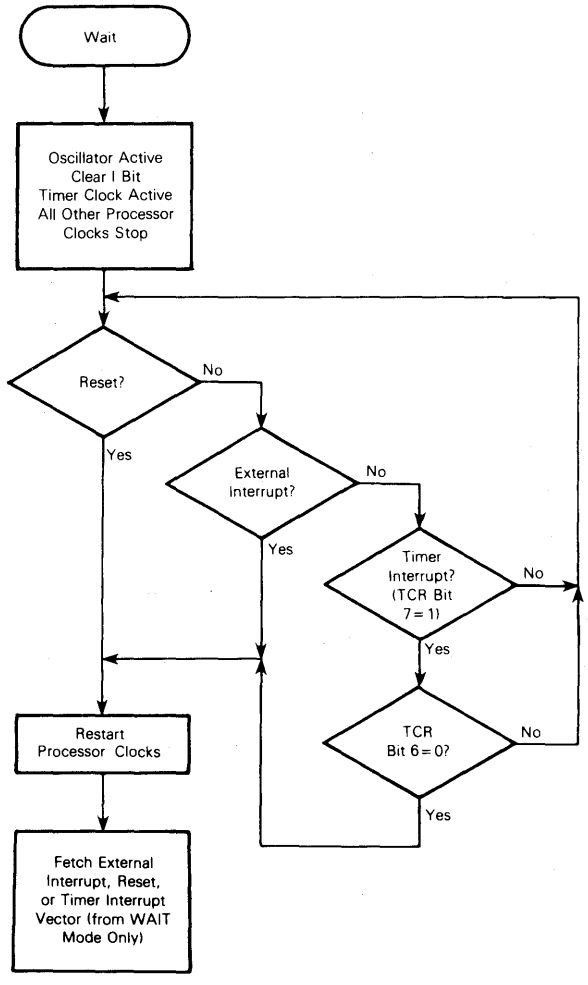


FIGURE 16 — WAIT FUNCTION FLOWCHART




## TIMER

The MCU timer contains an 8-bit software programmable counter (timer data register) with a 7-bit software selectable prescaler. Figure 17 contains a block diagram of the timer. The counter may be loaded under program control and is decremented towards zero by the clock input (prescaler output). When the counter decrements to zero, the timer interrupt request bit (i.e., bit 7 of the timer control register, TCR) is set. Then, if the timer interrupt is not masked (i.e., bit 6 of the TCR and the I bit in the condition code register are both cleared) the processor receives an interrupt. After completion of the current instruction, the processor proceeds to store the appropriate registers on the stack, and then fetches the timer vector address from locations $1FF8 and $1FF9 (or $1FF6 and $1FF7 if in the WAIT mode) in order to begin servicing; refer to **INTERRUPTS.**

The counter continues to count after it reaches zero, allowing the software to determine the number of internal or external input clocks since the timer interrupt request bit was set. The counter may be read at any time by the processor without disturbing the count. The contents of the counter

become stable prior to the read portion of a cycle, and do not change during the read. The timer interrupt request bit (TCR7) remains set until cleared by the software. If the timer interrupt request bit (TCR7) is cleared before the timer interrupt is serviced, the interrupt is lost. TCR7 may also be used as a scanned status bit in a non-interrupt mode of operation (TCR6 = 1).

The prescaler is a 7-bit divider which is used to extend the maximum length of the timer. Bit 0, bit 1, and bit 2 of the TCR are programmed to choose the appropriate prescaler output which is used as the counter input. The processor cannot write into or read from the prescaler; however, its contents are cleared to all zeros by the write operation into TCR when bit 3 of the written data equals a logic one. This allows for truncation-free counting.

The timer input can be configured for four different operating modes, depending on the value written to the TCR4 and TCR5 timer control register bits. Refer to **TIMER CONTROL REGISTER.**



### TIMER INPUT MODE 1

If TCR4 and TCR5 are both programmed to a zero, the input to the timer is from the internal processor clock and the TIMER input pin is disabled. The processor clock mode can be used for periodic interrupt generation, as well as a reference in frequency and event measurement. The processor clock is the instruction cycle clock. During a WAIT instruction, the processor clock input to the timer continues to run at its normal rate.

### TIMER INPUT MODE 2

With TCR4 = 1 and TCR5 = 0, the internal processor clock and the TIMER input pin are ANDed to form the timer input signal. This mode can be used to measure external pulse widths. The external timer input pulse simply turns on the internal processor clock for the duration of the pulse. The resolution of the count in this mode is plus or minus one clock cycle; therefore, accuracy improves with longer input pulse widths.

### TIMER INPUT MODE 3

If TCR4 = 0 and TCR5 = 1, then the crystal oscillator is used to clock the timer. This clock source is independent of the internal processor clock and thus it is useful in keeping real time. It is particularly useful with a 32.768 kHz crystal where the maximum modulus of the timer results in a precise one second interrupt rate.

### TIMER INPUT MODE 4

If TCR4 = 1 and TCR5 = 1, the internal processor clock input to the timer is disabled and the TIMER input pin becomes the input to the timer. The timer can, in this mode, be used to count external events as well as external frequencies for generating periodic interrupts. The counter is clocked on the falling edge of the external signal.

Figure 18 shows a block diagram of the timer subsystem. Power-on reset and the STOP instruction cause the counter to be set to $FO.

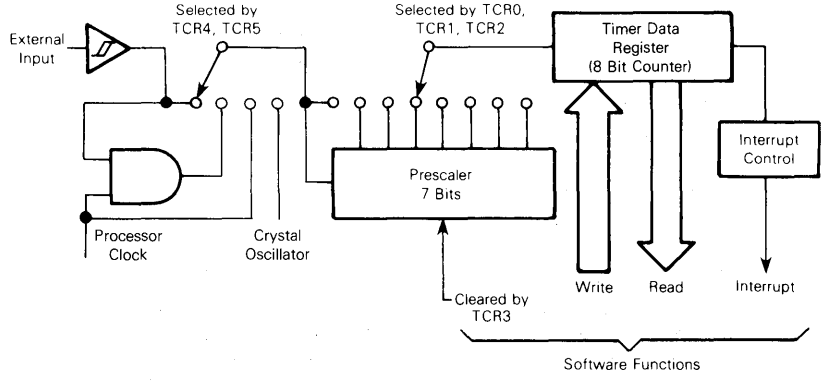
### TIMER CONTROL REGISTER (TCR)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| TCR7 | TCR6 | TCR5 | TCR4 | TCR3 | TCR2 | TCR1 | TCR0 | $0009 |

All bits in this register except bit 3 are read/write bits.

**TCR7** — Timer interrupt request bit: bit used to indicate the timer interrupt when it is logic one.

FIGURE 17 — TIMER BLOCK DIAGRAM




NOTES:
1. Prescaler and timer data register (8-bit counter) are clocked on the falling edge of the internal processor clock, crystal oscillator, or external input.
2. The timer data register counts down continuously.

1 – Set whenever the counter decrements to zero, or under program control.
0 – Cleared on external reset, power-on reset, STOP instruction, or program control.

**TCR6** – Timer interrupt mask bit: when this bit is a logic one it inhibits the timer interrupt to the processor.
1 – Set on external reset, power-on reset, STOP instruction, or program control.
0 – Cleared under program control.

**TCR5, TCR4** – Together, these two bits control the input to the timer. This is illustrated in the table below. (These two bits are unaffected by reset.)

| TCR5 | TCR4 | |
|---|---|---|
| 0 | 0 | Processor clock to timer |
| 0 | 1 | AND of processor clock and TIMER pin |
| 1 | 0 | Crystal oscillator to timer |
| 1 | 1 | TIMER pin to timer |

**TCR3** – Timer prescaler reset bit: writing a one to this bit resets the prescaler to zero. A read of this location always indicates a zero. (Unaffected by reset.)

**TCR2, TCR1, TCR0** – Prescaler select bits: decoded to select one of eight outputs of the prescaler. (Unaffected by reset.)

Prescaler

| TCR2 | TCR1 | TCR0 | Result |
|---|---|---|---|
| 0 | 0 | 0 | ÷ 1 |
| 0 | 0 | 1 | ÷ 2 |
| 0 | 1 | 0 | ÷ 4 |
| 0 | 1 | 1 | ÷ 8 |
| 1 | 0 | 0 | ÷ 16 |
| 1 | 0 | 1 | ÷ 32 |
| 1 | 1 | 0 | ÷ 64 |
| 1 | 1 | 1 | ÷ 128 |

## BUS RATES

The MC146805H2 MCU has the ability to change processor clock rate under program control. This is accomplished by utilizing the MCU program to select either the crystal oscillator or an internal synthesizer as the processor clock source. The entire circuit contains the crystal oscillator, synthesizer, gating and stop circuits, and a transition control circuit (refer to the block diagram of Figure 1). Rate selection is made by writing the system control register bit SCR2. Refer to **SYSTEM CONTROL REGISTER** for a discussion of this and all related controls.

### CRYSTAL OSCILLATOR

The crystal oscillator is designed to operate in the 30 to 50 kHz range for use with low frequency crystals such as the 32.768 kHz watch crystal. The oscillator stop select circuit allows the STOP instruction to completely shut down MCU operation by "killing" the oscillator or by gating off its output. By controlling the gating off of the oscillator, the user is provided with an option of either very low current drain or rapid recovery from the STOP function; i.e., with the oscillator running but gated off during the STOP function, the start up time ($t_{OXOV}$) could be much shorter than if the oscillator were "killed."

### FREQUENCY SYNTHESIZER

The frequency synthesizer uses a conventional phase lock loop which utilizes the crystal oscillator output as its reference frequency. The synthesizer output frequency is 16 $f_{osc}$ for the 3-volt port and 64 $f_{osc}$ for the 5-volt port ($f_{osc}$ represents the crystal oscillator frequency). The synthesizer bandwidth (wide or narrow) is program controlled to provide two different damping factors. This bandwidth control is provided by the system control register SCR0 bit. An external filter capacitor must be connected to the XFC pin (35) as part of the frequency synthesizer loop filter.

### TRANSITION CONTROL CIRCUIT

The transition control circuit provides a means for a smooth transition when switching the processor clock between the crystal oscillator and frequency synthesizer. Switching of the transition circuit is controlled by a bit in the system control register. A buffered output of the transition circuit is available at the CO pin whenever system control register SCR1 bit is set.

## WATCHDOG TIMER

The watchdog timer is designed to periodically time out and reset the MCU unless it is periodically cleared by the system program (the watchdog timer is also cleared during any other MCU reset). The time-out period ($t_{DM}$), which for a 32.768 kHz crystal = 3.00 seconds, is calculated as:

$$t_{DM} = (98{,}296 \pm 16)\left(\frac{1}{f_{osc}}\right)$$

**NOTE**

The variation is the result of never clearing the first four stages of the timer which are used for other internal functions.

A reset from the watchdog timer affects the CPU in the same manner as an external reset ($\overline{RESET}$ pin goes low); however, the watchdog timer reset also sets the SCR4 bit in the system control register. Refer to **SYSTEM CONTROL REGISTER** for additional information.

## AUDIO ALERT TONE GENERATOR

The alert tone generator provides a buffered tone output at the ALRT pin (25). This output provides drive for an external amplifier whenever the tone generator is activated by the microcomputer program; otherwise, the ALRT pin represents a high impedance. Three different tones can be generated by the alert tone generator and these tones are controlled by two bits (SCR6, SCR7) in the system control register. A table illustrating the SCR6-SCR7 bits status versus the alert tone generator output is shown below. Bits SCR6 and SCR7 are cleared by a reset.

| SCR7 | SCR6 | |
|---|---|---|
| 0 | 0 | No output from alert generator (pin 25 high impedance) |
| 0 | 1 | Low frequency output from alert generator (crystal oscillator frequency ÷ 32) |
| 1 | 0 | Medium frequency output from alert generator (crystal oscillator frequency ÷ 16) |
| 1 | 1 | High frequency output from alert generator (crystal oscillator frequency ÷ 8) |

## SYSTEM CONTROL REGISTER

The system control register is an 8-bit register which provides control bits SCR0 through SCR7. These bits are used in determining whether the system clock is furnished by the crystal oscillator or synthesizer, plus controlling the alert generator and resetting the dead-man timer.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| SCR7 | SCR6 | SCR5 | SCR4 | SCR3 | SCR2 | SCR1 | SCR0 | $0006 |
| | | | | SCR3 | | | | |

**SCR6, SCR7** – Together, these two bits control the state and frequency of the alert generator as shown in **AUDIO ALERT TONE GENERATOR.** Both of these bits are cleared by reset.

**SCR5** – This bit determines the on-off state of the synthesizer. This bit is set during reset. If the synthesizer output is already selected by SCR2 (SCR2 = 1), it cannot be turned off by SCR5 (SCR5 will remain at 1 to ensure that the selected synthesizer cannot be turned off).

1 = Synthesizer on
0 = Synthesizer off

**SCR4** – This bit has two different functions. When it is read for the first time following a reset, it is a reset qualifier bit. When it is written to, it is used to clear the dead-man timer.

Read Cycle
1 = Indicates MCU was last reset by dead-man timer. This bit is cleared by a read of system control register.
0 = Indicates MCU was last reset by a power-on or external reset.

Write Cycle
1 = Clears the dead-man timer.
0 = No action taken.

**SCR3** – This bit performs two functions. Together, these functions control the operation of the crystal oscillator in the STOP mode. The first write to this bit accesses a latch which can only be modified by the first write. After the first write, all other writes to SCR3 are written into a second latch. Only if both of these latches are zero will the oscillator be stopped (or "killed") when a STOP instruction is executed. See note and logic diagram below for further discussion.

1 = Gate crystal oscillator on.
0 = "Kill" crystal oscillator.

**SCR2** – This bit is used to determine which clock (synthesizer or crystal oscillator) is passed to the CPU (via the transition control circuit). This bit is cleared during reset. If the synthesizer is off (SCR5 = 0), it cannot be selected (SCR2 will remain at zero to ensure that a turned off synthesizer cannot be selected).

1 = Select synthesizer output.
0 = Select crystal oscillator output.

**SCR1** – This bit is used to either enable or disable the clock output (pin 34). This bit is set during reset.

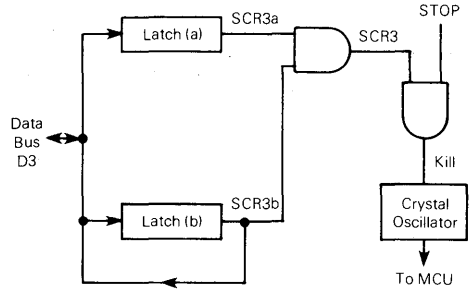
1 = CO output enabled.
0 = CO output disabled (pin 34 held low).

**SCR0** – This bit controls the loop bandwidth of the phase lock loop frequency synthesizer. In the wide bandwidth mode it will lock on frequency quickly. Once it is locked, the narrow bandwidth should be used to maintain better frequency stability. This bit is set during reset.

1 = Wide bandwidth.
0 = Narrow bandwidth.

**NOTE**

The first write after reset (external, power on, or dead-man timer) will determine the crystal oscillator control function. That is, if a one is written to SCR3 as the first write after reset, then the crystal oscillator can never be stopped ("killed") but only gated. However, if a zero is written to SCR3 as the first write after reset, then the oscillator can be "killed" if SCR3 = 0 and a STOP instruction is executed. This method of controlling the oscillator is possible because the SCR3 bit is contained in two different latches (as illustrated logically below). Reads are always from SCR3b. The contents of latch (a) are only affected by the first write to the system control register following an MCU reset. Subsequent writes to this register will alter data in latch (b) only. Thus, all reads before the second write, will always be a one.



## PHASE LOCK LOOP

The phase lock loop (PLL) consists of: a digital phase detector, a variable bandwidth loop filter, a voltage controlled oscillator (VCO), and a feedback frequency divider. A small external capacitor (typically 0.1 microfarads) is used by the loop filter. The synth $V_{SS}$ pin is the ground for the PLL and may be bypassed to minimize noise.

The phase detector compares the frequency and phase of the feedback frequency ($f_{FB}$) and the crystal oscillator reference frequency ($f_{REF}$) and generates the output, $\phi_{COMP}$, as shown in Figure 18. The output waveform is then integrated and amplified. The resultant dc voltage is applied to the voltage controlled oscillator. The output of the VCO is divided by a fixed frequency divider of 64 (in the 5-volt part) or 16 (in the 3-volt part) to provide the feedback frequency for the phase detector.

The startup time and frequency stability of the PLL can be changed via the variable bandwidth control in the loop filter. For the fastest startup, the low stability mode (SCR0 = 1) should be used. The high stability mode (SCR0 = 0) responds slowly and is normally used only after the PLL is at or near the operating frequency (see Figure 19).

The loop filter can source or sink only small currents in the high stability mode (approximately 1 microamp). Therefore, the external filter capacitor (XFC) should be selected for very low leakage. The printed circuit board must be clean and free from conductive material. The capacitor should be located as close to the microcomputer as possible to minimize noise.

FIGURE 18 — PHASE DETECTOR

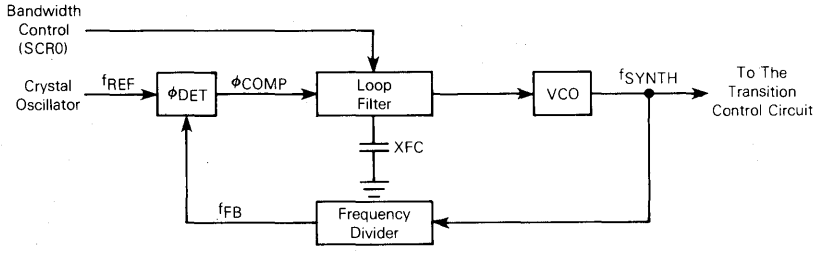
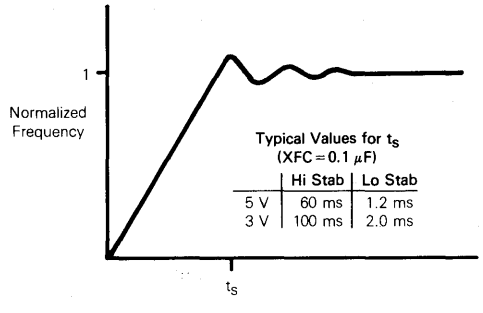



FIGURE 19 — TYPICAL STARTUP OF THE PLL



## INSTRUCTION SET

The MCU has a set of 61 basic instructions. They can be divided into five different types: register/memory, read-modify-write, branch, bit manipulation, and control. The following paragraphs briefly explain each type. All the instructions within a given type are presented in individual tables.

### REGISTER/MEMORY INSTRUCTIONS

Most of these instructions use two operands. The first operand is either the accumulator or the index register. The second operand is obtained from memory using one of the addressing modes. The operand for the jump unconditional (JMP) and jump to subroutine (JSR) instructions is the program counter. Refer to Table 6.



### READ-MODIFY-WRITE INSTRUCTIONS

These instructions read a memory location or a register, modify or test its contents, and write the modified value back to memory or to the register. The test for negative or zero (TST) instruction is an exception to the read-modify-write sequence since it does not modify the value. Refer to Table 7.

### BRANCH INSTRUCTIONS

Most branch instructions test the state of the condition code register and if certain criteria are met, a branch is executed. This adds an offset between −127 and +128 to the current program counter. Refer to Table 8.

### BIT MANIPULATION INSTRUCTIONS

The MCU is capable of setting or clearing any bit which resides in the first 128 bytes of the memory space (where all port registers, port DDRs, timer, timer control, system control and on-chip RAM reside). Bit manipulation in the ROM mapped area will not affect data in the ROM. An additional feature allows the software to test and branch on the state of any bit within the first 256 locations. The bit set, bit clear, and bit test and branch functions are all implemented with a single instruction. For the test and branch instructions, the value of the bit tested is automatically placed in the carry bit of the condition code register. Refer to Table 9.

**NOTE**

The MCU is actually capable of operating on the bit set and bit clear instructions anywhere in the first 256 bytes; however, since only ROM resides in the upper 128 bytes the bit set/clear instructions have no affect on the upper 128 bytes.

### CONTROL INSTRUCTIONS

These instructions are register reference instructions and are used to control processor operation during program execution. Refer to Table 10.

### OPCODE MAP

Table 11 is an opcode map for the instructions used on the MCU.

### ALPHABETICAL LISTING

The complete instruction set is given in alphabetical order in Table 12.

## ADDRESSING MODES

The MCU uses ten different addressing modes to provide the programmer with an opportunity to optimize the code to all situations. The various indexed addressing modes make it possible to locate data tables, code conversion tables, and scaling tables anywhere in the memory space. Short indexed accesses are single byte instructions, while the longest instructions (three bytes) permit accessing tables throughout memory. Short absolute (direct) and long absolute (extended) addressing are also included. One and two byte direct addressing instructions access all data bytes in most applications. Extended addressing permits jump instructions to reach all memory. Table 12 shows the addressing modes for each instruction, with the effects each instruction has on the condition code register. An opcode map is shown in Table 11.

The term "effective address" (EA) is used in describing the various addressing modes, and is defined as the byte address to or from which the argument for an instruction is fetched or stored. The ten addressing modes of the processor are described below. Parentheses are used to indicate "contents of" the location or register referred to; e.g., (PC) indicates the contents of the location pointed to by the PC. An arrow indicates "is replaced by", and a colon indicates concatenation of two bytes. For additional details and graphical illustrations, refer to the *M6805 HMOS/M146805 CMOS Family Microcomputer/Microprocessor User's Manual.*

### INHERENT

In inherent instructions, all the information necessary to execute the instruction is contained in the opcode. Operations specifying only the index register or accumulator, and no other arguments, are included in this mode.

### IMMEDIATE

In immediate addressing, the operand is contained in the byte immediately following the opcode. Immediate addressing is used to access constants which do not change during program execution (e.g., a constant used to initialize a loop counter).

$$EA = PC + 1;\ PC \leftarrow PC + 2$$

### DIRECT

In the direct addressing mode, the effective address of the argument is contained in a single byte following the opcode byte. Direct addressing allows the user to directly address the lowest 256 bytes in memory with a single two byte instruction. This includes all on-chip RAM and I/O registers, and 128 bytes of on-chip ROM. Direct addressing is efficient in both memory and time.

$$EA = (PC + 1);\ PC \leftarrow PC + 2$$
$$\text{Address Bus High} \leftarrow 0;\ \text{Address Bus Low} \leftarrow (PC + 1)$$

### EXTENDED

In the extended addressing mode, the effective address of the argument is contained in the two bytes following the opcode. Instructions with extended addressing modes are capable of referencing arguments anywhere in memory with a single three-byte instruction. When using the Motorola assembler, the user need not specify whether an instruction uses direct or extended addressing. The assembler automatically selects the most efficient addressing mode.

EA = (PC + 1):(PC + 2); PC ← PC + 3
Address Bus High ← (PC + 1); Address Bus Low ← (PC + 2)

### INDEXED, NO OFFSET

In the indexed, no offset addressing mode, the effective address of the argument is contained in the 8-bit index register. Thus, this addressing mode can access the first 256 memory locations. These instructions are only one byte long. This mode is used to move a pointer through a table or to address a frequently referenced RAM or I/O location.

EA = X; PC ← PC + 1
Address Bus High ← 0; Address Bus Low ← X

### INDEXED, 8-BIT OFFSET

Here the EA is obtained by adding the contents of the byte following the opcode to that of the index register; therefore, the operand is located anywhere within the lowest 511 memory locations. For example, this mode of addressing is useful for selecting the mth element in an n element table. All instructions are two bytes. The contents of the index register (X) is not changed. The contents of (PC + 1) is an unsigned 8-bit integer. One byte offset indexing permits look-up tables to be easily accessed in either RAM or ROM.

EA = X + (PC + 1); PC ← PC + 2
Address Bus High ← K; Address Bus Low ← X + (PC + 1)
Where:
K = The carry from the addition of X + (PC + 1)

### INDEXED, 16-BIT OFFSET

In the indexed, 16-bit offset addressing mode, the effective address is the sum of the contents of the unsigned 8-bit index register and the two unsigned bytes following the opcode. This addressing mode can be used in a manner similar to indexed 8-bit offset, except that this three byte instruction allows tables to be anywhere in memory (e.g., jump tables in ROM). As with direct and extended, the M6805 assembler determines the most efficient form of indexed offset; 8- or 16-bit. The contents of the index register is not changed.

EA = X + [(PC + 1):(PC + 2)]; PC ← PC + 3
Address Bus High ← (PC + 1) + K;
Address Bus Low ← X + (PC + 2)
Where:
K = The carry from the addition of X + (PC + 2)

### RELATIVE

Relative addressing is only used in branch instructions. In relative addressing, the contents of the 8-bit signed byte following the opcode (the offset) is added to the PC if and only if the branch condition is true. Otherwise, control proceeds to the next instruction. The span of relative addressing is limited to the range of −126 to +129 bytes from the branch instruction opcode location. The Motorola assembler calculates the proper offset and checks to see if it is within the span of the branch.

EA = PC + 2 + (PC + 1); PC ← EA if branch taken;
otherwise, EA = PC ← PC + 2

### BIT SET/CLEAR

Direct addressing and bit addressing are combined in instructions which set and clear individual memory and I/O bits. In the bit set and clear instructions, the byte is specified as a direct address in the location following the opcode. The first 256 addressable locations are thus accessed. The bit to be modified within that byte is specified within the first three bits of the opcode. The bit set and clear instructions occupy two bytes, one for the opcode (including the bit number) and the other to address the byte which contains the bit of interest.

EA = (PC + 1); PC ← PC + 2
Address Bus High − 0; Address Bus Low ← (PC + 1)

### BIT TEST AND BRANCH

Bit test and branch is a combination of direct addressing, bit set or bit clear addressing, and relative addressing. The actual bit to be tested, within the byte, is specified within the low order nibble of the opcode. The address of the data byte to be tested is located via a direct address in the location following the opcode byte (EA1). The signed relative 8-bit offset is in the third byte (EA2) and is added to the PC if the specified bit is set or cleared in the specified memory location. This single three byte instruction allows the program to branch based on the condition of any bit in the first 256 locations of memory.

EA1 = (PC + 1)
Address Bus High − 0; Address Bus Low ← (PC + 1)
EA2 = PC + 3 + (PC + 2); PC ← EA2 if branch taken;
otherwise, PC ← PC + 3

TABLE 6 — REGISTER/MEMORY INSTRUCTIONS

| | | Addressing Modes | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Immediate | | | Direct | | | Extended | | | Indexed (No Offset) | | | Indexed (8-Bit Offset) | | | Indexed (16-Bit Offset) | | |
| Function | Mnemonic | Op Code | # Bytes | # Cycles | Op Code | # Bytes | # Cycles | Op Code | # Bytes | # Cycles | Op Code | # Bytes | # Cycles | Op Code | # Bytes | # Cycles | Op Code | # Bytes | # Cycles |
| Load A from Memory | LDA | A6 | 2 | 2 | B6 | 2 | 3 | C6 | 3 | 4 | F6 | 1 | 3 | E6 | 2 | 4 | D6 | 3 | 5 |
| Load X from Memory | LDX | AE | 2 | 2 | BE | 2 | 3 | CE | 3 | 4 | FE | 1 | 3 | EE | 2 | 4 | DE | 3 | 5 |
| Store A in Memory | STA | – | – | – | B7 | 2 | 4 | C7 | 3 | 5 | F7 | 1 | 4 | E7 | 2 | 5 | D7 | 3 | 6 |
| Store X in Memory | STX | – | – | – | BF | 2 | 4 | CF | 3 | 5 | FF | 1 | 4 | EF | 2 | 5 | DF | 3 | 6 |
| Add Memory to A | ADD | AB | 2 | 2 | BB | 2 | 3 | CB | 3 | 4 | FB | 1 | 3 | EB | 2 | 4 | DB | 3 | 5 |
| Add Memory and Carry to A | ADC | A9 | 2 | 2 | B9 | 2 | 3 | C9 | 3 | 4 | F9 | 1 | 3 | E9 | 2 | 4 | D9 | 3 | 5 |
| Subtract Memory | SUB | A0 | 2 | 2 | B0 | 2 | 3 | C0 | 3 | 4 | F0 | 1 | 3 | E0 | 2 | 4 | D0 | 3 | 5 |
| Subtract Memory from A with Borrow | SBC | A2 | 2 | 2 | B2 | 2 | 3 | C2 | 3 | 4 | F2 | 1 | 3 | E2 | 2 | 4 | D2 | 3 | 5 |
| AND Memory to A | AND | A4 | 2 | 2 | B4 | 2 | 3 | C4 | 3 | 4 | F4 | 1 | 3 | E4 | 2 | 4 | D4 | 3 | 5 |
| OR Memory with A | ORA | AA | 2 | 2 | BA | 2 | 3 | CA | 3 | 4 | FA | 1 | 3 | EA | 2 | 4 | DA | 3 | 5 |
| Exclusive OR Memory with A | EOR | A8 | 2 | 2 | B8 | 2 | 3 | C8 | 3 | 4 | F8 | 1 | 3 | E8 | 2 | 4 | D8 | 3 | 5 |
| Arithmetic Compare A with Memory | CMP | A1 | 2 | 2 | B1 | 2 | 3 | C1 | 3 | 4 | F1 | 1 | 3 | E1 | 2 | 4 | D1 | 3 | 5 |
| Arithmetic Compare X with Memory | CPX | A3 | 2 | 2 | B3 | 2 | 3 | C3 | 3 | 4 | F3 | 1 | 3 | E3 | 2 | 4 | D3 | 3 | 5 |
| Bit Test Memory with A (Logical Compare) | BIT | A5 | 2 | 2 | B5 | 2 | 3 | C5 | 3 | 4 | F5 | 1 | 3 | E5 | 2 | 4 | D5 | 3 | 5 |
| Jump Unconditional | JMP | – | – | – | BC | 2 | 2 | CC | 3 | 3 | FC | 1 | 2 | EC | 2 | 3 | DC | 3 | 4 |
| Jump to Subroutine | JSR | – | – | – | BD | 2 | 5 | CD | 3 | 6 | FD | 1 | 5 | ED | 2 | 6 | DD | 3 | 7 |

TABLE 7 — READ-MODIFY-WRITE INSTRUCTIONS

| | | Addressing Modes | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Inherent (A) | | | Inherent (X) | | | Direct | | | Indexed (No Offset) | | | Indexed (8-Bit Offset) | | |
| Function | Mnemonic | Op Code | # Bytes | # Cycles | Op Code | # Bytes | # Cycles | Op Code | # Bytes | # Cycles | Op Code | # Bytes | # Cycles | Op Code | # Bytes | # Cycles |
| Increment | INC | 4C | 1 | 3 | 5C | 1 | 3 | 3C | 2 | 5 | 7C | 1 | 5 | 6C | 2 | 6 |
| Decrement | DEC | 4A | 1 | 3 | 5A | 1 | 3 | 3A | 2 | 5 | 7A | 1 | 5 | 6A | 2 | 6 |
| Clear | CLR | 4F | 1 | 3 | 5F | 1 | 3 | 3F | 2 | 5 | 7F | 1 | 5 | 6F | 2 | 6 |
| Complement | COM | 43 | 1 | 3 | 53 | 1 | 3 | 33 | 2 | 5 | 73 | 1 | 5 | 63 | 2 | 6 |
| Negate (2's Complement) | NEG | 40 | 1 | 3 | 50 | 1 | 3 | 30 | 2 | 5 | 70 | 1 | 5 | 60 | 2 | 6 |
| Rotate Left Thru Carry | ROL | 49 | 1 | 3 | 59 | 1 | 3 | 39 | 2 | 5 | 79 | 1 | 5 | 69 | 2 | 6 |
| Rotate Right Thru Carry | ROR | 46 | 1 | 3 | 56 | 1 | 3 | 36 | 2 | 5 | 76 | 1 | 5 | 66 | 2 | 6 |
| Logical Shift Left | LSL | 48 | 1 | 3 | 58 | 1 | 3 | 38 | 2 | 5 | 78 | 1 | 5 | 68 | 2 | 6 |
| Logical Shift Right | LSR | 44 | 1 | 3 | 54 | 1 | 3 | 34 | 2 | 5 | 74 | 1 | 5 | 64 | 2 | 6 |
| Arithmetic Shift Right | ASR | 47 | 1 | 3 | 57 | 1 | 3 | 37 | 2 | 5 | 77 | 1 | 5 | 67 | 2 | 6 |
| Test for Negative or Zero | TST | 4D | 1 | 3 | 5D | 1 | 3 | 3D | 2 | 4 | 7D | 1 | 4 | 6D | 2 | 5 |

TABLE 8 — BRANCH INSTRUCTIONS

| Function | Mnemonic | Relative Addressing Mode | | |
|---|---|---|---|---|
| | | Op Code | # Bytes | # Cycles |
| Branch Always | BRA | 20 | 2 | 3 |
| Branch Never | BRN | 21 | 2 | 3 |
| Branch IFF Higher | BHI | 22 | 2 | 3 |
| Branch IFF Lower or Same | BLS | 23 | 2 | 3 |
| Branch IFF Carry Clear | BCC | 24 | 2 | 3 |
| (Branch IFF Higher or Same) | (BHS) | 24 | 2 | 3 |
| Branch IFF Carry Set | BCS | 25 | 2 | 3 |
| (Branch IFF Lower) | (BLO) | 25 | 2 | 3 |
| Branch IFF Not Equal | BNE | 26 | 2 | 3 |
| Branch IFF Equal | BEQ | 27 | 2 | 3 |
| Branch IFF Half Carry Clear | BHCC | 28 | 2 | 3 |
| Branch IFF Half Carry Set | BHCS | 29 | 2 | 3 |
| Branch IFF Plus | BPL | 2A | 2 | 3 |
| Branch IFF Minus | BMI | 2B | 2 | 3 |
| Branch IFF Interrupt Mask Bit is Clear | BMC | 2C | 2 | 3 |
| Branch IFF Interrupt Mask Bit is Set | BMS | 2D | 2 | 3 |
| Branch IFF Interrupt Line is Low | BIL | 2E | 2 | 3 |
| Branch IFF Interrupt Line is High | BIH | 2F | 2 | 3 |
| Branch to Subroutine | BSR | AD | 2 | 6 |

TABLE 9 — BIT MANIPULATION INSTRUCTIONS

| Function | Mnemonic | Addressing Modes | | | | | |
|---|---|---|---|---|---|---|---|
| | | Bit Set/Clear | | | Bit Test and Branch | | |
| | | Op Code | # Bytes | # Cycles | Op Code | # Bytes | # Cycles |
| Branch IFF Bit n is Set | BRSET n (n=0...7) | – | – | – | 2•n | 3 | 5 |
| Branch IFF Bit n is Clear | BRCLR n (n=0...7) | – | – | – | 01+2•n | 3 | 5 |
| Set Bit n | BSET n (n=0...7) | 10+2•n | 2 | 5 | – | – | – |
| Clear Bit n | BCLR n (n=0...7) | 11+2•n | 2 | 5 | – | – | – |

TABLE 10 — CONTROL INSTRUCTIONS

| Function | Mnemonic | Inherent | | |
|---|---|---|---|---|
| | | Op Code | # Bytes | # Cycles |
| Transfer A to X | TAX | 97 | 1 | 2 |
| Transfer X to A | TXA | 9F | 1 | 2 |
| Set Carry Bit | SEC | 99 | 1 | 2 |
| Clear Carry Bit | CLC | 98 | 1 | 2 |
| Set Interrupt Mask Bit | SEI | 9B | 1 | 2 |
| Clear Interrupt Mask Bit | CLI | 9A | 1 | 2 |
| Software Interrupt | SWI | 83 | 1 | 10 |
| Return from Subroutine | RTS | 81 | 1 | 6 |
| Return from Interrupt | RTI | 80 | 1 | 9 |
| Reset Stack Pointer | RSP | 9C | 1 | 2 |
| No-Operation | NOP | 9D | 1 | 2 |
| Stop | STOP | 8E | 1 | 2 |
| Wait | WAIT | 8F | 1 | 2 |

TABLE 11 — MC146805 CMOS FAMILY INSTRUCTION SET OPCODE MAP

| | Bit Manipulation | | Branch | Read/Modify/Write | | | | | Control | | Register/Memory | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BTB | BSC | REL | DIR | INH | INH | IX1 | IX | INH | INH | IMM | DIR | EXT | IX2 | IX1 | IX | |
| Hi / Low | 0 0000 | 1 0001 | 2 0010 | 3 0011 | 4 0100 | 5 0101 | 6 0110 | 7 0111 | 8 1000 | 9 1001 | A 1010 | B 1011 | C 1100 | D 1101 | E 1110 | F 1111 | Hi / Low |
| 0 0000 | BRSET0 5 3 BTB | BSET0 5 2 BSC | BRA 3 2 REL | NEG 5 2 DIR | NEG 3 1 INH | NEG 3 1 INH | NEG 6 2 IX1 | NEG 5 1 IX | RTI 9 1 INH | | SUB 2 2 IMM | SUB 3 2 DIR | SUB 4 3 EXT | SUB 5 3 IX2 | SUB 4 2 IX1 | SUB 3 1 IX | 0 0000 |
| 1 0001 | BRCLR0 5 3 BTB | BCLR0 5 2 BSC | BRN 3 2 REL | | | | | | RTS 6 1 INH | | CMP 2 2 IMM | CMP 3 2 DIR | CMP 4 3 EXT | CMP 5 3 IX2 | CMP 4 2 IX1 | CMP 3 1 IX | 1 0001 |
| 2 0010 | BRSET1 5 3 BTB | BSET1 5 2 BSC | BHI 3 2 REL | | | | | | | | SBC 2 2 IMM | SBC 3 2 DIR | SBC 4 3 EXT | SBC 5 3 IX2 | SBC 4 2 IX1 | SBC 3 1 IX | 2 0010 |
| 3 0011 | BRCLR1 5 3 BTB | BCLR1 5 2 BSC | BLS 3 2 REL | COM 5 2 DIR | COMA 3 1 INH | COMX 3 1 INH | COM 6 2 IX1 | COM 5 1 IX | SWI 10 1 INH | | CPX 2 2 IMM | CPX 3 2 DIR | CPX 4 3 EXT | CPX 5 3 IX2 | CPX 4 2 IX1 | CPX 3 1 IX | 3 0011 |
| 4 0100 | BRSET2 5 3 BTB | BSET2 5 2 BSC | BCC 3 2 REL | LSR 5 2 DIR | LSRA 3 1 INH | LSRX 3 1 INH | LSR 6 2 IX1 | LSR 5 1 IX | | | AND 2 2 IMM | AND 3 2 DIR | AND 4 3 EXT | AND 5 3 IX2 | AND 4 2 IX1 | AND 3 1 IX | 4 0100 |
| 5 0101 | BRCLR2 5 3 BTB | BCLR2 5 2 BSC | BCS 3 2 REL | | | | | | | | BIT 2 2 IMM | BIT 3 2 DIR | BIT 4 3 EXT | BIT 5 3 IX2 | BIT 4 2 IX1 | BIT 3 1 IX | 5 0101 |
| 6 0110 | BRSET3 5 3 BTB | BSET3 5 2 BSC | BNE 3 2 REL | ROR 5 2 DIR | RORA 3 1 INH | RORX 3 1 INH | ROR 6 2 IX1 | ROR 5 1 IX | | | LDA 2 2 IMM | LDA 3 2 DIR | LDA 4 3 EXT | LDA 5 3 IX2 | LDA 4 2 IX1 | LDA 3 1 IX | 6 0110 |
| 7 0111 | BRCLR3 5 3 BTB | BCLR3 5 2 BSC | BEQ 3 2 REL | ASR 5 2 DIR | ASRA 3 1 INH | ASRX 3 1 INH | ASR 6 2 IX1 | ASR 5 1 IX | | TAX 2 1 INH | | STA 4 2 DIR | STA 5 3 EXT | STA 6 3 IX2 | STA 5 2 IX1 | STA 4 1 IX | 7 0111 |
| 8 1000 | BRSET4 5 3 BTB | BSET4 5 2 BSC | BHCC 3 2 REL | LSL 5 2 DIR | LSLA 3 1 INH | LSLX 3 1 INH | LSL 6 2 IX1 | LSL 5 1 IX | | CLC 2 1 INH | EOR 2 2 IMM | EOR 3 2 DIR | EOR 4 3 EXT | EOR 5 3 IX2 | EOR 4 2 IX1 | EOR 3 1 IX | 8 1000 |
| 9 1001 | BRCLR4 5 3 BTB | BCLR4 5 2 BSC | BHCS 3 2 REL | ROL 5 2 DIR | ROLA 3 1 INH | ROLX 3 1 INH | ROL 6 2 IX1 | ROL 5 1 IX | | SEC 2 1 INH | ADC 2 2 IMM | ADC 3 2 DIR | ADC 4 3 EXT | ADC 5 3 IX2 | ADC 4 2 IX1 | ADC 3 1 IX | 9 1001 |
| A 1010 | BRSET5 5 3 BTB | BSET5 5 2 BSC | BPL 3 2 REL | DEC 5 2 DIR | DECA 3 1 INH | DECX 3 1 INH | DEC 6 2 IX1 | DEC 5 1 IX | | CLI 2 1 INH | ORA 2 2 IMM | ORA 3 2 DIR | ORA 4 3 EXT | ORA 5 3 IX2 | ORA 4 2 IX1 | ORA 3 1 IX | A 1010 |
| B 1011 | BRCLR5 5 3 BTB | BCLR5 5 2 BSC | BMI 3 2 REL | | | | | | | SEI 2 1 INH | ADD 2 2 IMM | ADD 3 2 DIR | ADD 4 3 EXT | ADD 5 3 IX2 | ADD 4 2 IX1 | ADD 3 1 IX | B 1011 |
| C 1100 | BRSET6 5 3 BTB | BSET6 5 2 BSC | BMC 3 2 REL | INC 5 2 DIR | INCA 3 1 INH | INCX 3 1 INH | INC 6 2 IX1 | INC 5 1 IX | | RSP 2 1 INH | | JMP 2 2 DIR | JMP 3 3 EXT | JMP 4 3 IX2 | JMP 3 2 IX1 | JMP 2 1 IX | C 1100 |
| D 1101 | BRCLR6 5 3 BTB | BCLR6 5 2 BSC | BMS 3 2 REL | TST 4 2 DIR | TSTA 3 1 INH | TSTX 3 1 INH | TST 5 2 IX1 | TST 4 1 IX | | NOP 2 1 INH | BSR 6 2 REL | JSR 5 2 DIR | JSR 6 3 EXT | JSR 7 3 IX2 | JSR 6 2 IX1 | JSR 5 1 IX | D 1101 |
| E 1110 | BRSET7 5 3 BTB | BSET7 5 2 BSC | BIL 3 2 REL | | | | | | STOP 2 1 INH | | LDX 2 2 IMM | LDX 3 2 DIR | LDX 4 3 EXT | LDX 5 3 IX2 | LDX 4 2 IX1 | LDX 3 1 IX | E 1110 |
| F 1111 | BRCLR7 5 3 BTB | BCLR7 5 2 BSC | BIH 3 2 REL | CLR 5 2 DIR | CLRA 3 1 INH | CLRX 3 1 INH | CLR 6 2 IX1 | CLR 5 1 IX | WAIT 2 1 INH | TXA 2 1 INH | | STX 5 2 DIR | STX 6 3 EXT | STX 7 3 IX2 | STX 6 2 IX1 | STX 5 1 IX | F 1111 |

**Abbreviations for Address Modes**

| | |
|---|---|
| INH | Inherent |
| A | Accumulator |
| X | Index Register |
| IMM | Immediate |
| DIR | Direct |
| EXT | Extended |
| REL | Relative |
| BSC | Bit Set/Clear |
| BTB | Bit Test and Branch |
| IX | Indexed (No Offset) |
| IX1 | Indexed, 1 Byte (8-Bit) Offset |
| IX2 | Indexed, 2 Byte (16-Bit) Offset |

**LEGEND**

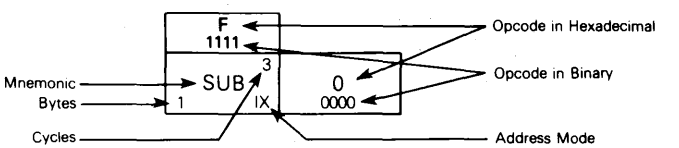


TABLE 12 — INSTRUCTION SET

| Mnemonic | Addressing Modes | | | | | | | | | | Condition Codes | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Inherent | Immediate | Direct | Extended | Relative | Indexed (No Offset) | Indexed (8 Bits) | Indexed (16 Bits) | Bit Set/ Clear | Bit Test & Branch | H | I | N | Z | C |
| ADC | | X | X | X | | X | X | X | | | Λ | ● | Λ | Λ | Λ |
| ADD | | X | X | X | | X | X | X | | | Λ | ● | Λ | Λ | Λ |
| AND | | X | X | X | | X | X | X | | | ● | ● | Λ | Λ | ● |
| ASL | X | | X | | | X | X | | | | ● | ● | Λ | Λ | Λ |
| ASR | X | | X | | | X | X | | | | ● | ● | Λ | Λ | Λ |
| BCC | | | | | X | | | | | | ● | ● | ● | ● | ● |
| BCLR | | | | | | | | | X | | ● | ● | ● | ● | ● |
| BCS | | | | | X | | | | | | ● | ● | ● | ● | ● |
| BEQ | | | | | X | | | | | | ● | ● | ● | ● | ● |
| BHCC | | | | | X | | | | | | ● | ● | ● | ● | ● |
| BHCS | | | | | X | | | | | | ● | ● | ● | ● | ● |
| BHI | | | | | X | | | | | | ● | ● | ● | ● | ● |
| BHS | | | | | X | | | | | | ● | ● | ● | ● | ● |
| BIH | | | | | X | | | | | | ● | ● | ● | ● | ● |
| BIL | | | | | X | | | | | | ● | ● | ● | ● | ● |
| BIT | | X | X | X | | X | X | X | | | ● | ● | Λ | Λ | ● |
| BLO | | | | | X | | | | | | ● | ● | ● | ● | ● |
| BLS | | | | | X | | | | | | ● | ● | ● | ● | ● |
| BMC | | | | | X | | | | | | ● | ● | ● | ● | ● |
| BMI | | | | | X | | | | | | ● | ● | ● | ● | ● |
| BMS | | | | | X | | | | | | ● | ● | ● | ● | ● |
| BNE | | | | | X | | | | | | ● | ● | ● | ● | ● |
| BPL | | | | | X | | | | | | ● | ● | ● | ● | ● |
| BRA | | | | | X | | | | | | ● | ● | ● | ● | ● |
| BRN | | | | | X | | | | | | ● | ● | ● | ● | ● |
| BRCLR | | | | | | | | | | X | ● | ● | ● | ● | Λ |
| BRSET | | | | | | | | | | X | ● | ● | ● | ● | Λ |
| BSET | | | | | | | | | X | | ● | ● | ● | ● | ● |
| BSR | | | | | X | | | | | | ● | ● | ● | ● | ● |
| CLC | X | | | | | | | | | | ● | ● | ● | ● | 0 |
| CLI | X | | | | | | | | | | ● | 0 | ● | ● | ● |
| CLR | X | | X | | | X | X | | | | ● | ● | 0 | 1 | ● |
| CMP | | X | X | X | | X | X | X | | | ● | ● | Λ | Λ | Λ |
| COM | X | | X | | | X | X | | | | ● | ● | Λ | Λ | 1 |
| CPX | | X | X | X | | X | X | X | | | ● | ● | Λ | Λ | Λ |
| DEC | X | | X | | | X | X | | | | ● | ● | Λ | Λ | ● |
| EOR | | X | X | X | | X | X | X | | | ● | ● | Λ | Λ | ● |
| INC | X | | X | | | X | X | | | | ● | ● | Λ | Λ | ● |
| JMP | | | X | X | | X | X | X | | | ● | ● | ● | ● | ● |
| JSR | | | X | X | | X | X | X | | | ● | ● | ● | ● | ● |
| LDA | | X | X | X | | X | X | X | | | ● | ● | Λ | Λ | ● |
| LDX | | X | X | X | | X | X | X | | | ● | ● | Λ | Λ | ● |
| LSL | X | | X | | | X | X | | | | ● | ● | Λ | Λ | Λ |
| LSR | X | | X | | | X | X | | | | ● | ● | 0 | Λ | Λ |
| NEG | X | | X | | | X | X | | | | ● | ● | Λ | Λ | Λ |
| NOP | X | | | | | | | | | | ● | ● | ● | ● | ● |
| ORA | | X | X | X | | X | X | X | | | ● | ● | Λ | Λ | ● |
| ROL | X | | X | | | X | X | | | | ● | ● | Λ | Λ | Λ |
| ROR | X | | X | | | X | X | | | | ● | ● | Λ | Λ | Λ |
| RSP | X | | | | | | | | | | ● | ● | ● | ● | ● |
| RTI | X | | | | | | | | | | ? | ? | ? | ? | ? |
| RTS | X | | | | | | | | | | ● | ● | ● | ● | ● |
| SBC | | X | X | X | | X | X | X | | | ● | ● | Λ | Λ | Λ |
| SEC | X | | | | | | | | | | ● | ● | ● | ● | 1 |
| SEI | X | | | | | | | | | | ● | 1 | ● | ● | ● |
| STA | | | X | X | | X | X | X | | | ● | ● | Λ | Λ | ● |
| STOP | X | | | | | | | | | | ● | 0 | ● | ● | ● |
| STX | | | X | X | | X | X | X | | | ● | ● | Λ | Λ | ● |
| SUB | | X | X | X | | X | X | X | | | ● | ● | Λ | Λ | Λ |
| SWI | X | | | | | | | | | | ● | 1 | ● | ● | ● |
| TAX | X | | | | | | | | | | ● | ● | ● | ● | ● |
| TST | X | | X | | | X | X | | | | ● | ● | Λ | Λ | ● |
| TXA | X | | | | | | | | | | ● | ● | ● | ● | ● |
| WAIT | X | | | | | | | | | | ● | 0 | ● | ● | ● |

Condition Code Symbols

- H Half Carry (From Bit 3)
- I Interrupt Mask
- N Negative (Sign Bit)
- Z Zero
- C Carry/Borrow
- Λ Test and Set if True, Cleared Otherwise
- ● Not Affected
- ? Load CC Register From Stack
- 0 Cleared
- 1 Set

3

## ORDERING INFORMATION

The following information is required when ordering a custom MCU. This information may be transmitted to Motorola in the following media:

EPROM(s) MCM2716s or MCM2532s
MDOS disk file

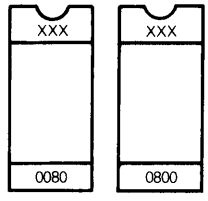
To initiate a ROM pattern for the MCU, it is necessary to first contact your local field service office, local sales person, or your local Motorola representative.

### EPROMs

The MCM2716 or MCM2532 type EPROMs, programmed with the customer program (positive logic sense for address and date), may be submitted for pattern generation. The EPROMs must be clearly marked to indicate which EPROM corresponds to which address space. Figure A-1 illustrates the marking for the two MCM2716 EPROMs required to emulate the MC146805H2.

After the EPROM(s) are marked, they should be placed in conductive IC carriers and securely packed. Do not use styrofoam.

FIGURE A-1 — EPROM MARKING




XXX = Customer ID

### VERIFICATION MEDIA

All original pattern media (EPROMs or floppy disk) are filed for contractual purposes and are not returned. A computer listing of the ROM code will be generated and returned along with a listing verification form. The listing should be thoroughly checked and the verification form completed, signed, and returned to Motorola. The signed verification form constitutes the contractual agreement for creation of the customer mask. If desired, Motorola will program a blank MCM2716 or MCM2532 EPROM (**supplied by the customer**) from the data file used to create the custom mask to aid in the verification process.

### ROM VERIFICATION UNITS

Ten MCUs containing the customer's ROM pattern will be sent for program verification. These units will have been made using the custom mask but are for the purpose of ROM verification only. For expediency they are usually unmarked, packaged in ceramic, and tested only at room temperature and five volts. These RVUs are included in the mask charge and are not production parts. These RVUs are not backed nor guaranteed by Motorola Quality Assurance.

### FLEXIBLE DISKS

The disk media submitted must be single-sided, single density, 8-inch, MDOS compatible floppies. The customer must clearly label the disk with the ROM pattern file name and company name. The floppies are not returned by Motorola as they are used for archival storage. The minimum MDOS system files as well as the absolute binary object file (filename. LO type of file) from the M6805 cross assembler must be on the disk. An object file made from a memory dump using the ROLLOUT command is also admissable. Consider submitting a source listing as well as: filename, .LX(EXORciser loadable format). This file will of course be kept confidential and is used 1) to speed up the process in house if any problems arise, and 2) to speed up our customer to factory interface if a user finds any software errors and needs assistance quickly from the factory representative.

MDOS is Motorola's Disk Operating System available on development systems such as EXORciser, EXORset, etc.

**OPTION LIST**

Select the options for the MCU from the following list. A manufacturing mask will be generated from this information. Select one in each section.

Operating Voltage
- ☐ 3 V (262 kHz bus with 32.768 kHz Crystal)
- ☐ 5 V (1.049 MHz bus with 32.768 kHz Crystal)

Interrupt Trigger
- ☐ Edge-Sensitive
- ☐ Level- and Edge-Sensitive

Customer Name_____

Address_____

City_____ State_____ Zip_____

Phone (____)_____ Extension_____

Contact Ms/Mr_____

Customer Part Number_____

Pattern Media
- ☐ 2532 EPROM
- ☐ 2716 EPROM
- ☐ MDOS Disk File
- ☐ (Note)_____

NOTE: Other media require prior factory approval.

Signature_____

Title_____