CALIFORNIA COMPUTER SYSTEMS

APPLE II™ PROGRAMMABLE TIMER MODULE

MODEL 7440A

OWNERS'S MANUAL

TABLE OF CONTENTS

07440-9001A

California Computer Systems
95050
Santa Clara, Calif
USA

APPLE II ™ is a trade mark of APPLE COMPUTER, INC.

INTRODUCTION

Every digital computer has a clock buried way down deep into its inner workings. Your APPLE II is no exception. The only trouble with this is that it is buried so deep in the hardware that the software can°t get to it. But how often have you wished that you could use it to time intervals? Or some other time based function?

This is the gap that your CCS 7440 Programmable Timer Module fits in. It is a programmable subsystem designed to provide you a variety of variable system time functions. In reality, it has three completely independant software controllable timers on one chip. Being software controllable, it provides you with a 256 byte software storage space in which either RAMs or custom tailored ROMs may be placed. To allow you to tailor the module to your application, a bread board area is also provided. Of course, it provides the needed support circuitry to couple the interface to your computer.

What are some of the possible uses that you can put this module to? Some which come to mind include a 40 megahertz frequency counter, capacitance meter, an inductance meter, a pulse generator, an interval timer, and many more. In fact, the only limit is in the fact the function you wish to control or monitor must be a time function or convertable into a time based function.

In this manual, we will try to provide you with sufficient information to develop and implement your application. We do, however, recommend that you get a data sheet on the MC 6840 LSI chip which forms the heart of this unit. It will provide you with much supplimental detailed information which we may have overlooked. Also, Motorola Semiconductor Products Inc publishes a "Programmable Timer Fundamentals and Applications" manual which will prove very valuable to you.

Happy timing!

I. THEORY OF OPERATION

Your 7440A Programmable Timer Module was designed around the 6840 IC device. It performs all the basic timing functions required, but it needs additional help around it to make it operate properly in your computer. For your convenience, we will break the subsystem into four convenient functional blocks and discuss each one separately. The blocks include the computer interface, the timer chip itself, the external interface area, and the on board program storage area.

The 6840's three timers may be independently programmed to operate in modes which fit into a wide variety of applications. The module is fully bus compatable with the Peripheral I/O connectors inside your computer and is accessed by loading and storing operations from the computer in much the same manner as a memory device. In a typical application, the Timer will be loaded by first storing two bytes of data into an associated Counter Latch. This data is then put in the counter via a Counter Initialization cycle. The counter now decrements on each following clock period until the preprogrammed condition (out of several possible) causes it to halt or recycle. The timers are thus programmable, cyclic in nature, controllable by the external inputs or your computer program, and accessible by the computer at any time.

Since the 6840 is so central to the operation of the module, much of the following discussion will use the 6840 as a reference point.

A. Computer Interface.

The 6840 interfaces to the peripheral I/O bus with an eight bit bidirectional data bus, the Device Select line, the Read/-Write line, the Phase 0 line (Enable line), the Interrupt Request line, the Reset and three address lines (Register Select lines). These signals permit computer control of the module.

1. Bidirectional Data: The bidirectional data lines allow transfer of data between the computer and the 6840. Since the 6840 has but limited bus driving capability, a DP8304 octal line driver has been included to help out. It has sufficient power to reliably drive the bus. To do this though, takes a lot of power from the +5 volt line. Since actual traffic over thse data path is small in terms of use versus inactive time, a power down circuit has been added to conserve the power drawn from the computer. Transistor Q2 monitors the Device Select line. Any time Device Select goes active (negative logic), Q2 applies power to the DP8304. When the data transfer is completed and Device Select goes false, power is removed from the 8304.

2. Device Select: This signal is generated by the computer. The 6840 uses this signal to allow the desired data transfer to occur.

3. Read/-Write: This signal originates in the computer to control the data transfer direction on the Data Bus. With Device Select true, a low state on this line allows the transfer of data from the computer through the 8304 to the 6840 input buffers. Actual data transfer will occur on the trailing edge of Phase 0 (6840 Enable). Alternately, if the Read/-Write is high and Device Select low, Phase 0 going high allows data from the 6840 to be transferred to the computer through the 8304.

4. Enable (System Phase 0): The Enable signal synchronizes data transfer between the computer and the 6840. It also does a similar synchronization function on the external clock, reset or gate inputs to the 6840. If the internal clock is selected, the 6840 uses this signal as the internal clock.

5. -Interrupt Request (-IRQ): The -IRQ signal along with the interrupt arbitration logic, is used to notify your computer that a predetermined event has occurred. The -IRQ line from the 6840 is set if, and only if, the Composite Interrupt Flag (Bit 7 of the Internal Status Register) is asserted. The conditions under which the -IRQ line goes active are discussed in conjunction with the Status Register in the Software section below.

The interrupt arbitration logic is but one link in the interrupt daisy chain. The entire daisy chain prioritizes peripheral generated interrupts insuring that only one device interrupts the computer at one time. If this unit wants service from the processor, it can present an -IRQ signal to the arbitration logic. If no higher priority request is in progress (connector pin 28 active high), then an interrupt request is sent to the computer (connector pin 30). Concurrently, pin 23 is driven low, maintaining the interrupt daisy chain and signalling lower priority devices that an interrupt is pending, forcing them to wait their turn. Upon servicing the 6840, its interrupt request is removed, and the arbitration logic sets the daisy chain signal high again. Now, the lower priority devices can request interrupts as they need. Note that the highest priority device should be installed in the leftmost Peripheral I/O connector in the groug J4 thru J12 in your computer. J2 does not support the daisy chain. Also, no empty slots are allowed inbetween the cards. If there is, the daisy chain will be broken, and will not work.

6. External Reset: A low level at this input is clocked into the 6840 by the Enable input. Two Enable pulses are needed to synchronize and process the signal. The 6840 then recognizes the active low or inactive high on the third Enable pulse. If the Reset signal is asynchronous, an additional Enable period is required if setup times are not met. The Reset input must be stable High/Low for the minimum time stated in the 6840 specification sheets. Unless you have modified your computer's Reset circuitry, the minimum setup times will be met each time you push the Reset key.

Recognition of a low level at this input by the 6840 causes the following action to occur:

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

In short, all setup you may have done with the chip is wiped out each time you press the Reset key.

7. Register Select Lines (RS0, RS1, RS2): Your computer enables a block of 16 addresses for each of the Peripheral I/O connectors each time the Device Enable signal is true at the connector. This block of addresses is for use by the board in the specific slot, with address lines A0 to A3 determining which specific address is desired. In our case, the 6840's RS0, RS1, and RS2 lines are connected to A0, A1, and A2 respectively. These lines are used in conjunction with the R/W line to select the internal registers, counters, and latches as shown in the Software area.

We already stated that the 6840 may be accessed by load and store operations in much the same manner as a memory device. The instructions available in your computer which perform operations directly on memory shouldn't be used when the 6840 is addressed. These instructions really fetch a byte from memory, perform an operation, then restore the result to the same address location. Since the 6840 uses the R/W line as an additional register select, the modified data may not be stored in the same register if these instructions are used.

Note also that address line A3 is not used by the 6840 in its register selection process. This line is a "don't care" line when addressing the PTM. As a result, the same registers may be addressed in the upper or lower half of the 16 address block.

B. The 6840

We have already seen the computer interface to the 6840. Now let's look inside the 6840 and find out what is in there. In turn, we will look at the control registers, the status register, the counters and, finally, the timer operating modes.

1. Control Registers: Three write only registers in the 6840 are used to modify timer operation to suit a variety of applications. Control Register #2 has unique address space (RS0 = 1, RS1 = RS2 = 0) and therefore may be written into at any time. The remaining Control Registers (#1 and #3) share the same address; a logic zero on all Register Select lines. The least significant bit of Control Register #2 (CR20) is used as an additional addressing bit to differentiate between Control Register #1 or #3. So, with all Register Selects and R/W inputs at logic 0, Control Register #1 will be written into if CR20 is a logic one. Under the same conditions, Control Register #3 will be written into if CR20 is a logic zero. Control Register #3 can also be written into after a Reset condition has occurred, since control register bits (except CR10) are cleared. Therefore, one may write in the sequence CR3, CR2, CR1.

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

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

Control Register Bits CR10, CR20, and CR30 are unique in that each selects a different function.

The remaining Control Register bits (1 through 7) select common functions, with a particular Control Register affecting only its corresponding timer. For example, Bit 1 of Control Register #1 (CR11) selects whether an internal or external clock is to be used with Timer #1. Similarly, CR21 selects the clock source for Timer #2, and CR31 performs this function for Timer #3.

Control Register Bit 2 determines if the binary information contained in the Counter Latches (and subsequently loaded in the counter) is to be treated as a 16 bit word or two 8 bit bytes. In the single 16 bit Counter Mode (CRx2=0), the counter decrements to zero after N + 1 enabled (-G = 0) clock periods, where N is the 16 bit number in the Counter Latches. With CRx2 = 1, a similar Time Out will occur after (L + 1) * (M + 1) enabled clock periods, where L and M, respectively, refer to the LSB and MSB bytes in the Counter Latches.

Control Register Bits 3, 4, and 5 are explained in detail in the Tilmer Operating Mode section. Bit 6 is an interrupt mask bit which will be explained more fully in conjunction with the Status Register; and bit 7 is used to enable the corresponding Timer Output.

A summary of the control register programming modes will be shown in the Software section.

2. Status Register: The 6840 has an internal read only Status Register which holds four Interrupt Flags. (The remaining four bits of the register are not used, and default to zeros when being read.) Bits 0, 1, and 2 are assigned to Timers 1, 2, and 3, respectively, as individual flag bits. Bit 7 is a Composite Interrupt Flag. This flag bit will be asserted, if any of the individual flag bits is set while Bit 6 of the corresponding Control Register is at a logic one. The conditions for asserting the Composite Interrupt Flag bit can then be stated as:

INT = I1 * CR16 + I2 * CR26 + I3 * CR36

where INT = Composite Interrupt Flag (Bit 7)
I1 = Timer #1 Interrupt Flag (Bit 0)
I2 = Timer #2 Interrupt Flag (Bit 1)
I3 = Timer #3 Interrupt Flag (Bit 2)
* = Boolean AND Operator
+ = Boolean OR Operator

An interrupt flag is cleared by a Timer Reset condition, i.e. External Reset = 0 or Internal Reset Bit (CR10) = 1. It will also be cleared by a Read Timer Counter Command provided the Status Register

has previously been read while the interrupt flag is set. This condition for Read Status Register - Read Timer Counter (RS - RT) sequence is meant to prevent missing interrupts which may occur after reading the status, but before reading the Timer Counter.

An individual interrupt flag is also cleared by a Write Timer Latches (W) or Counter Initialization (CI) sequence, providing W or CI affects the Timer corresponding to the individual interrupt flag.

3. Counter Operation: Now, we're to the heart of the 6840. Without the counters, we wouldn't need any of the other areas. Here, we will look at the Counter Latch Initialization, Counter Initialization and the Asynchronous External Input/Output lines.

a. Counter Latch Initialization: Each of the 3 independent timers consist of a 16 bit addressable counter and 16 bits of addressable latches. The counters are preset to the binary numbers stored in the latches. Counter initialization results in the transfer of the latch contents to the counter.

Since the PTM data bus is 8 bits wide and the counters are 16 bits wide, a temporary register (MSB Buffer Register) is provided. This holding register is write only and for the Most Significant Byte of the desired latch data. Three addresses are provided for the MSB Buffer Register (as seen in the Software section), but they all lead to the same Buffer. Data from the MSB Buffer is automatically transferred to the Most Significant Byte of Timer #x when a Write Timer #x Latches Command is performed. So it can be seen that the 6840 has been designed for transfer of two bytes of data into the counter latches provided that the MSB is transferred first.

A logic zero at the Reset input (pushing the Reset key) also initializes the counter latches. In this case, all latches will assume a count of 65,536 ($FFFF). It is important to note that an Internal Reset (Bit 0 of Control Register 1 =1) has no effect on the counter latches.

b. Counter Initialization: This is defined as the transfer of the data from the latches to the counter with subsequent clearing of the Individual Interrupt Flag associated with the counter. Counter Initialization always occurs when a reset condition (Reset key depressed or CR10 = 1) is recognized. It also occurs (depending on Timer Mode) with a Write Timer Latches command or recognition of a negative transition of the -GATE input.

Counter recycling or reinitialization occurs if negative transition of the clock input is recognized after the counter has reached an all zero state. In this case, the data is transferred from the Latches to the Counter on the transition.

c. Asynchronous Input/Output Lines: Each of the three timers within the 6840 has external clock and gate inputs as well as a counter output line. The inputs are high impedance, TTL compatible lines and outputs are capable of driving 2 standard TTL loads.

CLOCK Inputs (C1, C2, and C3): Input pins C1, C2, and C3 will accept asynchronous TTL voltage level signals to decrement Timers 1, 2, and 3 respectively. The high and low levels of the external clocks must each be stable for at least one system clock period plus the sum of setup and hold times for the inputs. The asynchronous clock rate can vary from DC to the limit imposed by Enable Setup and Hold time. In our specific case, since Enable is driven by the computer Phase 0 at 1 megahertz, C1 and C2 have an upper limit of 500 kilohertz. C3 is the same if the Divide by 1 option is selected, or it can go up to 4 megahertz if the Divide by 8 option is programmed by setting CR30 equal to 1.

The external clock inputs are clocked by Enable pulses. Three Enable periods are used to synchronize and process the external clock. The fourth Enable pulse decrements the internal counter. This does not affect the input frequency, it merely creates a delay from a clock input transition to internal recognition of that transition by the 6840. All references to -C inputs in this manual relate to internal recognition of that transition. Note that a clock hign or low level which does not meet minimum setup and hold time specifications may require an additional Enable pulse for recognition. When observing recurring events, a lack of synchronization will result in "jitter" being seen on the output of the 6840 if using asynchronous clocks and gate input signals. There are two types of jitter. "System jitter" is the result of the input signals being out of synchronization with the Enable, permitting signals with marginal setup and hold time to be recognized by either the bit time nearest the input transition or the subsequent bit time. "Input jitter" can be as great as the time between input signal negative going transitions plus the system jitter, if the first transition is reccognized during one system cycle, and not recognized the next cycle, or vice versa.

External clock input C3 represent a special case when Timer #3 is programmed to utilize its optional Divide by 8 prescaler. The maximum input frequency

and allowable duty cycles for this case are specified at 4 megahertz for your unit, as stated earlier. At this frequency, a 50% duty cycle is needed. For lower frequencies, the pulse high and the pulse low times must each be at least 125 nanoseconds. The output of the prescaler (the divided by 8 signal) is treated in the same manner as the other clock inputs. That is, it is clocked into the counter by Enable pulses, is recognized on the fourth Enable pulse (provided setup and hold time requirements are met), and must make an output pulse at least as wide as the sum of an Enable period, setup, and hold times. This equates to (1000 + 200 + 50) nanoseconds in our case.

GATE Inputs (G1, G2 and G3): Input pins G1, G2, and G3 all accept asynchronous TTL compatable signals which are minus logic and used as triggers or clock gating functions to Timers 1, 2, and 3, respectively. The gating inputs are clocked into the 6840 by the Enable signal in the same manner as the previously discussed clock inputs. That is, a GATE transition is recognized by the 6840 on the fourth Enable pulse (provided setup and hold time requirements are met), and the high or low levels of the GATE input must be stable for at least one system clock period plus the sum of setup and hold times. All references to G transition in this manual are to internal recognition of the imput transition.

The Gx inputs of all the timers directly affect the internal 16 bit counter. The operation of G3 is therefore independent of the Divide by 8 prescaler.

TIMER Outputs (O1, O2, and O3): Timer outputs O1, O2, and O3 are capable of driving up to two TTL loads and produce defined output waveforms for either Continuous or Single Shot Timer modes. Ox waveform definition is accomplished by selection either Single 16 bit or Dual 8 bit operating modes. The single 16 bit mode will produce square wave outputs when in the continuous timer mode and will produce a single pulse in the Single Shot Timer mode. The Dual 8 bit mode will produce a variable duty cycle pulse in both the continuous and single shot timer modes. One bit of each Control Register (CRx7) is used to enable the corresponding output. If this bit is cleared, the output will stay low regardless of the operating mode.

The Continuous and Single Shot Timer Modes are the only ones for which output response is defined in this manual. Refer to the Motorola Programmable Timer Fundamentals and Applications manual for a discussion of the output signals in other modes. Signals appear at the outputs (unless CRx7 = 0) during Frequency and Pulse Width comparison modes, but the actual waveform is not predictable in typical applications.

4. Timer Operating Modes

The 6840 was designed to operate effectively in wide ranging applications. This is accomplished by using three bits of each control register (CRx3, CRx4, and CRx5) to defined different operating modes of the Timers. These modes are outlined in Table 1.

TABLE 1 OPERATING MODES

Control Register Bit CRx3	CRx4	CRx5	Timer Operating Mode
0	*	0	Continuous
0	*	1	Single Shot
1	0	*	Frequency Comparison
1	1	*	Pulse Width Comparison

* Defines Additional Timer Functions

In addition to the four timer modes in Table 1, the remaining control register bit is used to modify counter loading and enabling or interrupt conditions.

a. Continuous Operating Mode (See Table 2)

Any of the timers in the 6840 may be programmed to operate in a continuous mode by writing zeros into bits 3 and 5 of the corresponding control register. Assuming that the timer output is enabled (CRx7 = 1), either a square wave or variable duty cycle waveform will be generated at the Timer Output, Ox. The type of output is selected via Control Register Bit 2.

Either a Timer Reset (CR10 = 1 or External Reset = 0) condition or internal recognition of a negative transition of the Gate results in Counter Loading. A Write Timer Latches command can be selected as a Counter Initialization signal by clearing CRx4.

The counter is enabled by an absence of a Timer Reset condition and a logic zero at the Gate input. The counter will then decrement on the first clock signal recognized during or after the counter loading cycle. It continues to decrement on each clock signal so long as G stays low and no reset condition exists. A Counter Time Out (the first clock after all counter bits = 0) results in the individual interrupt flag being set and reinitialization of the counter.

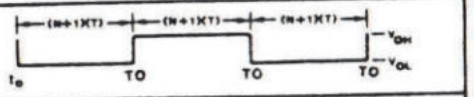
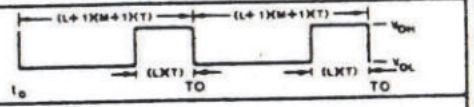
A special condition exists for the dual 8 bit mode (CRx2 = 1) if L = 0. In this case, the counter will revert to a mode similar to the single 16 bit mode except Time Out occurs after M + 1 clock pulses. The output, if enabled, goes low during the counter initialization cycle and reverses state at each Time

Out. The counter remains cyclical (is reinitialized at each Time Out) and the individual interrupt flag set when Time Out occurs. If M = L = 0, the internal counters do not change, but the output toggles at a rate of 1/2 the clock frequency.

In dual 8 bit mode (CRx2 = 1) the MSB decrements once for every full countdown of the LSB + 1. When the LSB = 0, the MSB is unchanged; on the next clock pulse, the LSB reset to the count in the LSB Latches and the MSB is decremented by one. The output, if enabled, remains low during and after initialization and will stay low until the counter MSB is all zeros. The output will go high at the beginning of the next clock pulse. The output remains high until both the LSB and MSB of the counter are zero. At the beginning of the next clock pulse the defined TimeOut (TO) will occur and the output will go low.

The discussion of the Continuous Mode assumes that the application requires an output signal. It should be noted that the Timer operates in the same manner with the output disabled (CRx7 = 0). A Read Timer Counter command is valid regardless of CRx7.

Table 2. CONTINUOUS OPERATING MODES

CONTINUOS MODE (CRX3=0,CRX5=0)			
Control Register		Initialization/Output Waveforms	
CRX2	CRX4	Counter Initialization	ϕTimer Output (OX)(CRX7=1)
0	0	$\overline{G}$↓+W+R	
0	1	$\overline{G}$↓+R	
1	0	$\overline{G}$↓+W+R	
1	1	$\overline{G}$↓+R	

$\overline{G}$↓ = Negative transition of $\overline{Gate}$ input
W = Write Timer Latches Command
R = Timer Reset (CR10 = 1 or External $\overline{Reset}$ = 0)
N = 16-Bit Number in Counter Latch
L = 8-Bit Number in LSB Counter Latch
M = 8-Bit Number in MSB Counter Latch
T = Clock Input Negative Transitions to Counter
t_0 = Counter Initialization Cycle
TO = Counter Time Out (All Zero Condition)

ϕAll time intervals shown above assume the the $\overline{Gate}$ ($\overline{G}$) and $\overline{Clock}$ ($\overline{C}$) signals are synchronized to Enable (System ϕ2) with the specified setup and hold time requirements.

b. Single Shot Timer Mode (See Table 3)

This mode is identical to the Continuous mode with three exceptions. The first of these is obvious by the name: the output returns to a low level after the initial Time Out and remains low until another Counter Initialization cycle happens. The waveforms available are shown in Table 3.

As indicated in Table 3, the internal counting mechanism remains cyclical in the Single Shot Mode. Each Time Out of the counter results in the setting of an individual interrupt flag and reinitialization of the counter.

The second major difference between the Single Shot vs Continuous modes is that the internal counter enable is not dependent on the Gate input remaining in the low state for the Single Shot mode.

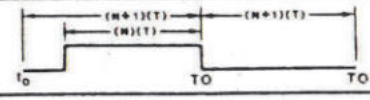
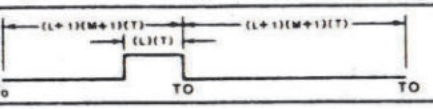
Another special condition is introduced in the Single Shot mode. If L = M = 0 (Dual 8 bit) or N = 0 (Single 16 bit), the Out goes low on the first clock received during or after Counter Initialization. The output stays low until the Operating Mode is changed or nonzero data is written into the Counter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

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

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

Aside from these differences, the modes are identical.

Table 3. SINGLE SHOT MODES

SINGLE-SHOT MODE (CRX3=0, CRX7=1, CRX5=1)			
Control Register		Initialization/Output Waveforms	
CRX2	CRX4	Counter Initialization	Timer Output (OX)
0	0	$\overline{G}$↓+W+R	
0	1	$\overline{G}$↓+R	
1	0	$\overline{G}$↓+W+R	
1	1	$\overline{G}$↓+R	

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

Refer to the Programmable Timer Fundamentals and Applications manual for a discussion of the output signals in these modes. The counter does operate in either Single 16 or Dual 8 bit modes as programmed by CRx2. Other features of the Time Interval Modes are outlined in Table 4.

Table 4. TIME INTERVAL MODES

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

c. Frequency Comparison or Period Measurement Mode (CRx3 = 1, CRx4 = 0) The Frequency Comparison Mode with CRx5 = 1 is straightforward. If Time Out occurs prior to the first negative transition of the Gate input after a Counter Initialization cycle, an individual interrupt flag is set. The counter is disabled, and a counter initialization cycle cannot go until the interrupt flag is cleared and a negative transition on G is detected.

If CRx5 = 0, as shown in Table 4 and Table 5, an interrupt is generated if the Gate input returns low prior to Time Out. If Counter Time Out occurs first, the counter is recycled and continues to decrement. A bit is set within the timer on the initial Time Out that prevents further individual interrupt generation until another Counter Initialization cycle has been completed. When this internal bit is set, a negative transition of the Gate input starts a new Counter Initialization cycle.

Any of the timers in the 6840 may be programmed to compare the period of pulses (giving the frequency after calculations) at the Gate input with the time period required for Counter Time Out. A negative transition of the Gate input enables the counter and starts a Counter Initialization cycle - provided that other conditions as noted in Table 5 are satisfied.

The counter decrements on each clock pulse recognized at or after Counter Initialization until an Interrupt is generated, a Write Timer Latchs command is issued, or a Timer Reset condition occurs. It can be seen from Table 5 that an interrupt condition is generated if CRx5 = 0 and the period of the pulse (single pulse or measured separately repetitive pulses) at the Gate input is less than the Counter Time Out period. If CRx5 = 1, an interrupt is generated if the reverse is true.

Assume now with CRx5 = 1 that a counter loading has occurred and that the Gate input has returned low before Counter Time Out. Since there is no individual interrupt flag generated, this automatically starts a new counter initialization cycle. The process will continue with frequency comparison being performed on each Gate input cycle until the mode is changed, or a cycle is found to be above the predetermined limit.

Table 5. FREQUENCY COMPARISON MODE

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

d. Pulse Width Comparison Mode (CRx3 = 1, CRx4 = 1) This mode is similar to the Frequency Comparison Mode except for a positive, rather than a negative, transition of the Gate input terminates the count. With CRx5 = 0, and individual interrupt flag will be generated if the zero level pulse applied to the Gate input is less than the time period needed for Counter Time Out. With CRx5 = 1, the interrupt is generated when the reverse condition is true.

As can be seen in Table 6, a positive transition of the Gate input disables the counter. With CRx5=0, it is therefore possible to directly obtain the width of any pulse causing an interrupt. Similar data for other Time Interval Modes and conditions can be obtained, if two sections of the 6840 are dedicated to the purpose.

Table 6. PULSE WIDTH COMPARISON MODE

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

C. External Interface (Patch Area)

In the upper right hand corner of your Timer Module, you will find a matrix of uncommitted pads. Along the left side of this matrix are two pads for 5 volts and pads which connect to the external inputs of the 6840. These pads are labelled O for Output, G for Gate, and C for Clock lines. Each is followed by a number which indicates the specific timer (1, 2, or 3) to which they connect. Along the bottom of the matrix are 12 pads; 4 labelled O and 8 labelled I. These connect to uncommitted gates on the board. The leftmost 3 of these is an open collector exclusive OR the remaining 3 groups of 3 are open collector NAND gates. Finally, 4 more pads in the lower right of the board are labelled +5 (for +5 volts). All of these pads are plated through. This is the area provided for you to construct any special "front end" which you may desire.

Note that all the pads in the right most column are connected together on the back side and that they are connected to ground. Also, every 3 pads in the horizontal rows are connected together on the back side. These are all provided to ease your task of connecting point to point wiring, should you choose this way of wiring the interconnections. Of course, wire wrap will also work just fine.

Sufficient room is provided for up to two 24 pin MSI/LSI chip, three 16 (or 14) pin chips, up to eight 8 pin chips, or some limited combinations of these possibilities. Some room is also left over to mount any discrete components you may desire.

Use of low power technology for your front ends is highly desirable. Your computer only provides a maximum rated 500 milliamps of +5 volt current to all installed peripheral cards. This card, before adding any front end components idles at 80 (135 worst case) milliamperes. When accessing memories, the current goes up to 280 (405 worst case) milliamps if 74S287 or equivelent ROMs are installed. When accessing the 6840, current is about 145 (235 worst case) milliamps. If 2112A RAMs are installed, they add 70 (110 worst case) milliamperes to the idle current. There is not a lot of room for power hogs in the front end.

D. Control Program Memory

Your computer dedicates 256 bytes of memory space for each peripheral connector. This space allows sufficient space for most interface unique soft/firmware. The CCS card has provision to place two 256x4 bit Read Only Memories (ROMs) on board to store your controller

firmware. Since the ROMs consume quite a bit of power, they are also equipped with a power down feature, like the one described for the 8304. Should you want to develop your own controller firmware, this unit allows you to install two Random Access Memories. CCS offers a RAM-Pak to support this feature. Since the program RAMs are volatile, the memory power down feature must be disabled and the R/W control line enabled to these RAMs. This is done by installation of one jumper wire on the interface card. With the RAMs, you can develop and test out your controller program thoroughly before committing it to ROMs.

II. INSTALLATION AND CHECKOUT

Now that we have an understanding on how the card works, let's try it out. The ultimate test is using it in an actual operating environment. But before we plug in the card and expect it to work right off, we have some set up to do. So, dig out your paper and pencil. You need to design and build your front end for your specific application.

Front End Hints

Only you know to what use you will put the timers to. You might get away with building nothing if your use needs only one timer with no external connections. Or, for a 40 megahertz frequency counter, you may want to add a divide by 10 chip to the Timer #3 clock and a precision time base crystal oscillator from which an accurate gate control can be developed. Or, you may just want to add a jumper from O1 to C2, which allows a .1 second increment timer to operate.

Whatever your choice, we suggest that you socket all logic chips. Point to point wiring is fairly easy to do; we suggest using 22 or 24 solid insulated wire for this. Be sure to use a good quality, rosin core solder. .032 diameter 60/40 solder is our preference. Use a low wattage soldering iron. A 25 watt unit is adequate. Too much heat can cause destruction of the pads. After the front end is built, clean off any rosin residue with an appropriate solvent.

Visually inspect your work. Make sure you made the right connections, no shorts are apparent, and all solder connections are shiny and properly filleted. Insure you did connect the +5 volt and ground leads.

After you are satisfied with the custom front end, finish checking out the rest of the board before performing any functional tests on the front end. Knowing the rest of the board works properly will help minimize troubleshooting the front end.

Memory Chip and Card Installation

Now, let us install the ROMs (or RAMs) onto the card. The ROMs come in matched pairs: one holds the lower 4 bits of each control program byte, while the other holds the upper 4 bits of each byte. Insure that the upper half byte ROM is the leftmost one on the card and the other one goes into the remaining socket. See the instructions in the ROM-Pak to determine which ROM is which. If you are using RAMs instead of ROMs, be sure to install the RAM jumper. Its location may be found just above the gold plated board connector fingers. Either RAM can go into either slot. After the ROMs (RAMs) are seated in the sockets, check to insure that the pin 1s are properly aligned towards

the upper left hand corner of the board. Also, check to insure that all pins entered the sockets and aren't rolled under the chips. Push the chips down firmly to seat them into the sockets.

Your card is now ready to be put in the computer.

****

WARNING: Do not remove the computer top cover if the line power cord is plugged in. Damage may result to you and your computer.

****

Now, place the computer directly in front of you. Remove the top cover by laying the palms of your hands onto the back edge of the computer, with your fingers hanging over the rear. Curl your fingers around the rear edge until you feel the ridge at your fingertips. Gently but firmly pry up until two distinct "pops" are heard. Don't lift the cover any further. Slide it to the rear to remove it from the computer. Towards the inside rear of the computer, you will see eight 50 pin connecters. From left to right, they are numbered #0, #1,...,#7. Place the CCS Programmable Timer card into any of these connectors except #0 (the left-most). #0 is reserved for other use. Insert the card by holding it so that the parts side of the card is towards the right. Align the card edge into the chosen connector (we suggest slot #2 if it isn't in use yet) and gently push the card down until it is firmly seated. Replace the top cover onto the computer. Finally, plug in the line power cord. We're ready to test the unit.

Check-out

The best test for any computer hardware is in its actual use. There are a few tests we should perform first, though, to gain confidence that the timer unit will really work. For the remainder of this section, we will assume the timer unit card is in slot #2. If you put it in a different slot, you'll have to modify the tests accordingly.

Test 1. ROM Test (if installed)

This test displays the contents of the ROMs on the TV screen. Then, you can compare the display with the ROM program listing. This verifies that the ROMs are properly installed and can be read by the computer.

Note: Your computer's disassembler can't recreate any assembler pseudo operation codes, such as ORG or EQU.

Occasionally, use of the ORG instruction may hide an instruction from the disassembler. For instance:

```
BCS     *
ORG     *-1
SEC
```

will disassemble as:

```
BCS     *+$38
```

Watch out for this kind of programming trick when you are comparing the listings. It is valid code, but may make you think you have bad ROMs. Programming tricks like this are used to save memory in tight situations.

Procedure:

a. Turn on and Reset your computer.
b. Type in C200L (CR)
c. Compare the listing to the TV display.
d. When you run out of screen display, type in L (CR).
e. Repeat c. and d. until all 256 bytes of ROM are read.
f. If problems result, compare the hexadecimal values of the memory locations. The ROMs may be reversed. If not, see your dealer.

Test 2. RAM Test (if installed).

This test verifies that we can read and write all locations of the controller RAMs. It copies a 256 byte segment of your system's firmware into the RAMs, then compares this copy to the original. Errors, if any, are displayed on the TV screen.

Procedure:

a. Turn on and Reset the computer.
b. Type in C200<F000.F0FFM (CR)
c. Type in C200<F000.F0FFV (CR)
d. A * should appear almost immediately on the screen if all is OK. If not, insure the RAM jumper is installed. Otherwise, see your CCS dealer.

If you installed ROMs, you are ready to use the CCS Programmable Timer Module. If you installed RAMs or left the memories off, you are now ready to start developing your custom controller software.

NOTES:

IV. SOFTWARE

Your 7440A Programmable Timer Module is quite a versatile device. It gets this versatility by striking a balance between the hardware and its controlling software. The hardware takes care of most of the tasks which are the same, regardless of use. The software is left to do what it does best, the application unique tasks. With this personality contained in the software it is impossible to make one program to be all things to everybody. CCS offers some standard ROM Paks which may meet your needs. Review their function definitions carefully. If a ROM Pak fits your needs, then use it. If not, then read on. In this section, we will provide you with the information needed to control the timer, a look at how to use interrupts in your computer, some of the firmware hooks in your computer, and an example program which allows you to control the duration of events (such as game turns) in 0.1 second increments, up to 6553.5 seconds, and shows you how to interface the duration limit to Integer BASIC programs.

A. The 6840 Addresses, Commands, and Status

1. Your computer dedicates 16 addresses to each Peripheral I/O connector (except connector #0). These are in addition to the 256 addresses assigned to the RAMs/ROMs, and are for use of the LSI chip functions. These addresses are at $C0xy (or -16128 + 16*n + y) where x = 8 + n, n = the Peripheral I/O connector slot number, and y is = specific 1 of 16 addresses desired. For this unit, only the first 8 addresses are used. The specific 6840 registers that are addressed at each location are summarized in the following table.

Table 7. Register Selection

Register Address	Operations Write	Read
C0x0 (CR20=0)	Write CR 1	No operation
C0x0 (CR20=1)	Write CR 3	No operation
C0x1	Write CR 2	Read Status Register
C0x2	Write MSB Buffer Reg	Read Timer 1 Counter
C0x3	Write Timer 1 Latches	Read LSB Buffer Reg
C0x4	Write MSB Buffer Reg	Read Timer 2 Counter
C0x5	Write Timer 2 Latches	Read LSB Buffer Reg
C0x6	Write MSB Buffer Reg	Read Timer 3 Counter
C0x7	Write Timer 3 Latches	Read LSB Buffer Reg

Note: x = 8 + Connector Slot Number

2. 6840 Commands. The timers' commands were described in Section I above. The following table summarizes them in one place.

Table 8. Control Register Programming

CRx3	CRx4	CRx5	Timer Operating Mode
0	*	0	Continuous
0	*	1	Single-Shot
1	0	*	Frequency Comparison
1	1	*	Pulse Width Comparison

* Defines Additional Timer Functions

3. 6840 Status. The timers' status was also discussed in Section I. Here is the summary:

Table 9. Status Register Contents

Bit 7 6 5 4 3 2 1 0	Meaning
x 0 0 0 0 x x 1	Timer 1 event has occurred
x 0 0 0 0 x 1 x	Timer 2 event has occurred
x 0 0 0 0 1 x x	Timer 3 event has occurred
1 0 0 0 0 y y y	6840 generated an interrupt (Interrupts enabled)

Note: The "event" is determined by the current command for the timer in question.
x = Immaterial for the described meaning.
y = 1 in at least one of these positions; the related timer event caused the interrupt.

B. Interrupts

The Programmable Timer Module becomes most useful when it is allowed to generate interrupts. In fact, most applications demand use of a interrupt structure. But, what is that structure in your computer?

Whenever your computer recognizes an interrupt request (Peripheral I/O connector pin 30 goes low and interrupts enabled) it first saves the current program counter contents on the stack, then the program status register on the stack, and then jumps to the location pointed to by locations $FFFE. This address is in the computer's ROMs. So at this point, we have no control over it. Since ($FFFE) = $FA86, the computer starts executing the code at $FA86. First, it saves the A register in location $0045, then retrieves the status byte off the stack. It now tests to see if a BREAK instruction caused the interrupt. If not, it pushs the status back onto the stack and jumps indirect to the address contained in locations $03FE and $03FF. Here is where we can finally gain control of the interrupt processing. We can plug the address of our interrupt handler routine into $03FE and $03FF.

Usually, interrupt routines are asynchronous to the main program being executed. This means that the main program should never know that interrupts happen. We, then, must insure that we do not lose any register contents while we process the interrupt. So, the first thing we must do is save all the registers. After we have processed the interrupt, we must restore them to their original contents and return control to the main program.

That, in a nutshell, defines what we must do to process interrupts. We implied, but did not mention, another step. Before we can have any interrupts to process, we must start them going. This step involves plugging in the interrupt handler address vector into locations $03FE and $03FF and initializing the timers to their proper commands and data to start the process going.

Programming the initialization and interrupt handling routines is really quite easy, but there is absolutely no room for errors. Troubleshooting can be quite difficult, because error results can be most unpredictable. Think your routines through carefully and then hand step through them, playing like you are the computer. Keep track of all register contents and pointers. Make sure they really do what you want them to do. Finally, load them into the computer and try them out. Hope for the best, but expect the worst. Persevere, and the results will be most satisfying.

We mentioned that most interrupt schemes work asynchronously from the main program. An example of this would be displaying and continuously updating a readout of the timer into the corner of the display. How do we control the main program by a timer event? One way would be to have the main program continuously poll the readout, looking for the desired event. This way is easy to program, but wastes a lot of execution time looking for that event to happen. With Integer BASIC, we are in luck. There is an entry point which allows us to start RUNning the main program without destroying any of the variables in the BASIC program. Our assembly language routine, after it has done its task, can transfer control to this entry point. Then the BASIC program can do its processing without ever losing or resetting its own variables. The software example shown below uses this technique.

The above guide assumes that only one interrupt generating device is active in your computer at any given instant. If more than one device will be active at the same time, then we must provide some way to determine which one caused the interrupt. The hardware interrupt arbitration logic does not help us here at all. So, we must build a polling routine which, after

receiving an interrupt, finds which unit caused the interrupt, and then transfer control to the unit's unique handler routine. Since the polling routine will be executed first, the interrupt handler jump vector should point to it and should save all the registers before doing any polling. Each unique handler can do the register restoring, or a separate segment of the common code can do it. No example is given for such a polling routine, because it is highly dependent on the units installed in the system.

C. Software "Hooks"

Your computer already has a significant amount of soft(firm)ware built into it. When we add this timer into the system, we want to take advantage of what is there already and not interfere with it, at the same time. For these reasons, we need to identify the spots in the memory allocation we can use, and the addresses of any needed interface points.

We have already identified the interface points that we need. These are the interrupt handler jump vector location, and the Integer BASIC RUN (Saving Variable Values) entry point. Let us discuss these in a little more detail.

The Interrupt Handler Jump Vector location is at $03FE and $03FF (1022 and 1023 decimal). During our initialization of the timer, we need to store the address of our interrupt handler into these locations. Note that the low order byte of the address goes into $03FE and the high order byte goes into $03FF. This is just the reverse of what one would normally expect, but such is the way of the 6502 microprocessor.

We also need to know what the conditions of the computer are when we gain control in our handler. Specifically, the computer's firmware has saved the A register in location $0045, and used the A register to test for a BREAK instruction. So, the value of the A register is unknown and not the same as it was before the interrupt. We need to go to location $0045 to find the value of A. X and Y are unaltered; we can save them on the stack for later restoral. The interrupt causes the Program Counter and P register to be saved on the stack, so we don't need to worry about them.

The entry point into BASIC is at address $E836, and causes the BASIC program in memory to execute from the beginning, but doesn't zero the program variables, as the normal RUN command does. To use this ability, we must test a variable for a zero value in the first program step. If it's zero, we have a normal RUN, and must do the initial program setup. Part of that setup must be to make the tested variable non-zero.

When an interrupt occurs, the > BASIC program is again RUN from the start. This time, though, we find that the variable is non-zero, and know that we need to process a time out from the timer.

What about constant storage areas in RAM? For instance, we cannot read back commands or timer data from the 6840 once they have been written. In many instances we need to know the current command is a specific timer. Since the video display only use 120 bytes out of each 128, this leaves 8 groups of 8 bytes per group to use as we wish. Noting that we have 8 Peripheral I/O connectors, we can allocate one byte per chunk per connector. This way, we can use the connector's page address as the index to access our temporary storage. Noting that Connector #0 will not need its 8 bytes because of its special nature, we now have 8 dedicated and 8 shared temporary locations at our disposal. Table 10 shows the addresses and our suggested usage for the timer module.

Table 10. Temporary Storage Locations

Dec	Hex	Hex-Page	Usage
1144+n	$0478+n	$3B8+Cn	Timer #1 Command
1272+n	$04F8+n	$438+Cn	Timer #2 Command
1400+n	$0578+n	$4B8+Cn	Timer #3 Command
1528+n	$05F8+n	$538+Cn	User Defined (Count up/down mask in the example below.)
1656	$0678		Timer #1 LSB Data
1656+n	$0678+n	$5B8+Cn	Timer #2 LSB Data
1784	$06F8		Timer #1 MSB Data
1784+n	$06F8+n	$638+Cn	Timer #2 MSB Data
1912+n	$0778+n	$6B8+Cn	Timer #3 LSB Data
2040+n	$07F8+n	$738+Cn	Timer #3 MSB Data

Note: n is the connector slot number.
Cn is the connector page address.
n = 0 defines the shared addresses. Two were used for the Timer #1 data. Location $07F8 is dedicated for other uses: DO NOT USE.

D. An Example

We now have sufficient information to put the timer to use. For this example, let's assume you want the computer to referee a game of your choice which allows for any number of players. Each player's turn will be time limited. If the player does not move within the allotted time, the computer forces him to pass and goes on to the next player. The total number of players can be specified during the game setup, as can the turn duration. The turn duration can be any value between .1 second and 1 hour, 49 minutes, 13.5 seconds. The game, itself, is not important to this example. We'll leave that detail to you.

Now, let's decide how to set up the hardware. The 1 hour, 49 minute, and 13.5 second maximum time just happens to be 65535 one-tenth second counts. That also just happens to be the capacity of one of our 16 bit counters. If we can provide a .1 second clock, command it to function as a Single Shot timer which generates an interrupt when it times out, we will have our turn timer. We will use the RUN, Saving Variables entry point that we mentioned earlier to convey the time out to the Integer BASIC program. Let's assign Timer #2 to this function and Timer #1 to generate the .1 second clock we need.

Having made this allocation of hardware, we need to interconnect the two timers. So, install a jumper wire from the Timer #1 Output (O1) to Timer #2's Clock input (C2). That's all we need to do to the hardware!

Now, what about the commands and data for the timers? Timer #2 is fairly easy. Looking at Table 7 above, we want to determine the command which:

a. Enables Interrupts from Timer #2;

b. Enables the Single Shot mode where the Reset causes initialization (but not a Write to Latches; we want a delayed start capability;)
c. Uses the normal 16 bit count mode;
d. Uses an External Clock (from Timer #1);
e. Allows access to Timer #1 command register.

These conditions translate into a command of $71 or decimal value of 113. The data we need to provide to the Timer #2 latches is the number of .1 second ticks each turn will last.

Timer #1 is a bit more difficult to set up. Here we want a square wave output of .1 second duration. We do not want it to generate interrupts, and we will use the 16 bit mode to gain sufficient capacity. We'll use the computer's internal clock and divide it down to give us the .1 second clock. We will want to hold all registers in a preset mode until we are ready to start a turn, then we'll allow all timers to operate. These conditions translate into a command of $93 (147 decimal) to preset the counters, followed by a $92 (146 decimal) to let the counters run.

Timer #1's data is the problem area. This is the divisor of the system clock to give us the .1 second output. From Table 2 above, we find that this divisor may be expressed by the equation:

2*(N + 1)*T = .1 second where
N is the divisor we're seeking, and
T is the period of Timer #1's internal clock.

Since we're using the internal clock, it is connected to the 6840's Enable line. This line is connected to the Phase 0 line of your computer, which is listed in the "Red Book" as 1 megahertz. After investigating this, we find that the system's master clock is 14.318 megahertz and Phase 0 derived by dividing the 14.318 megahertz signal by 14. So, in reality, Phase 0 is a 1.022714 megahertz signal. The T that we need is the inverse of this value, or 14/14318000 for ease of expression. Putting this into our equation and solving for N, we find N equal to 51134.714 (aren't computers nice?) Since we need to have an integer in the range of 1 to 65536, we can round off to 51135 and be close enough for our purposes. The round off error is less than .001 per cent. This is probably more accurate than the crystal master oscillator frequency. You'll never perceive the error in this application.

Now, let's actually get into the software. First let us look at the Assembly Language routines, then we will go into the >BASIC routines which work with them.

The Assembly Language listing really has two completely independent interrupt handlers in it. One provides the interface to your > BASIC program, while the other provides an asynchronous display of Timer 3s counter. Since our example needs only the former, we will describe it in some detail, but provide only an overview of the other.

The > BASIC Time Out handler, itself, is really two separate routines. The first one initializes the timer, while the other one actually handles the interrupt. The initializing routine expect that all the timer command and data register contents have already been stored into the locations identified in Table 10. Its entry point is at $C000 + $100*n (-16384 + 256*n) where n is the Peripheral I/O connector slot number of the timer card. When control passes to this location, the flags register is saved, then two of the flags are set to positively identify which routine (out of four) is currently executing. Then it goes to the common code, which saves the rest of the registers and sets up the indices. These indices are based on the slot address: (x) is the offset from $C080 (-16256), which is $n0 (16*n), to address the 6840's registers; and (y) is the Timer board's memory page address, which is $Cn (192+n). In both cases, n is the slot number. At this point, the common code ends and control passed to the unique code streams. For the initialization path, the handler entry point address is put into locations $3FE and $3FF. From this point on, the code is the same for either initialization process. All the timers receive a software reset by taking Timer 1's command, setting bit 0 to 1, and then writing it into Timer 2 and then 1, in that order. Writing into Timer 2 first

insures access to Timer 1. Bit 0 is now set to 0 and written into Timer 2 to gain access to the Timer 3 command register. Timer 3's command is now fetched and stored into the Timer. Finally, Timer 2's command is fetched, bit 0 set to 1 to allow easy access to the software reset function of Timer 1, and then stored into the #2 command register.

At this time, the data latches are given starting values, the registers are all restored to their entry values, interrupts are enabled, and a subroutine exit taken to return control to the calling program.

Note that in this example, the timers are not yet running. This last step is left for your main program to do. To do it, first get Timer 1's command from the save location ($0478+n or 1144+n), set bit 0 to 0, and then store or POKE it into location $C080 + $10 * n (-16256 + 16 * n). This removes the software reset and starts the timers running.

When the designated timer times out, it causes an interrupt to be generated. When your computer sees the interrupt request, it passes control to location $C10B in our example. After it saves the registers and set the indices, it issues a software reset to the timers. It then restores the registers insuring the stack pointer is the same as it was just before the interrupt, and transfers control to the > BASIC's RUN saving variables entry point. Your BASIC program now has control. You can do whatever you want.

The asynchronous interrupt handler uses the same initialization routine. The only differences here are the entry points. The initializer entry point is at $Cn04 (-16380 + 256 * n) and it sets up the interrupt vector to point to $Cn0D (-16371 + 256 * n). Again, you must turn the timers loose to run, but once they start, they do their thing independent of any other program in the computer. Its thing is to read the 16 bit value from Timer 3's counter, change it to decimal with leading zeroes suppressed and display the results in columns 33 to 37 of line n + 1 of the TV display. The displayed number is in the range of 0 to 65535. It will either display the number of counts since last Timer 3 initialization (assuming the data for Timer 3 is initialized to $FFFF) or counts remaining to time out. To select the option, store an $FF (255) or a 0 into location $05F8 + n (1528 + n) before the timers start to run. The $FF mask correlates to counts since start; the 0 mask to counts remaining.

Your program can gain the information by querying the video display memory. The locations involved are $0420 + $80 * n to $0424 + $80 * n (1056 + 128 * n to 1060 + 128 * n).

Returning to the BASIC Turn Timer program, here's the > BASIC part of the interface:

```
 10 REM  TURN TIMER INTERFACE PROGRAM
 15 REM  FIRST, SEE IF INTERRUPT ENTRY
 20 IF PLAYER  <> 0 THEN GOTO 300
 25 REM  HERE IS THE INITIALIZATION CODE
 30 INPUT "WHAT IS THE TIMER'S SLOT", N
 35 IF N < 1 OR N > 7 THEN GOTO 30
 40 REM  LINES 50 TO 70 LOAD THE TIMER'S RAM FROM DISK
 45 REM  OMIT IF YOU USE ROM OR NO DISK
 50 PRINT "(Ctrl D)BLOAD TIMER, A$A00"
 55 BASE = -16384 + 256 * N
 60 FOR I = 0 TO 255
 65 POKE BASE + I, PEEK (2560 + I)
 70 NEXT I
 75 REM  SET THE TEST VARIABLE TO NON-ZERO
 80 PLAYER = 1
 85 REM  INITIALIZE THE TIMER REGISTER HOLD LOCATIONS
 90 REM  THE COMMANDS
100 POKE 1144+N,147
105 POKE 1272+N,113
110 POKE 1400+N,0
115 REM  SET TIMER 1 DATA FOR .1 SECOND
120 POKE 1656,192 : POKE 1784, 199
125 REM  DEVELOP THE TURN TIME FOR TIMER 2 DATA
130 INPUT "TURN TIME:  MINUTES", TIME
135 INPUT "SECONDS", SEC
140 TIME = 60 * TIME + SEC
145 OFFSET = 0
150 IF TIME > 3200 THEN OFFSET = 125
155 TIME = TIME - 256 * OFFSET/10
160 INPUT "TENTHS OF SECONDS",SEC
165 TIME = TIME * 10 + SEC
170 POKE 1656 + N, TIME MOD 256
175 POKE 1784 +N, TIME/256 + OFFSET
180 REM  TIMER 3 DATA
185 POKE 1912 + N, 255 * N
200 PRINT "TIMER READY"
205 REM  GET THE NUMBER OF PLAYERS
210 INPUT "NUMBER OF PLAYERS", MAX
215 IF MAX < 1 THEN GOTO 210
220 REM  HERE IS WHERE YOUR SET UP LOGIC GOES
225 GOTO 320
290 REM  HERE IS WHERE WE COME FOR AN INTERRUPT
300 PRINT : PRINT "TIME'S UP!"
305 PLAYER = PLAYER + 1
310 IF PLAYER > MAX THEN PLAYER = 1
315 REM  START THE NEXT TURN AND TURN ON TIMER
320 CMD = 2 * (PEEK (1144 + N)/2)
325 POKE -16256 + 16 * N, CMD
330 REM  INSURE INTERRUPTS ARE ENABLED
335 CALL -16261 + 256 * N
340 PRINT "PLAYER ", PLAYER, "'S TURN"
345 REM  HERE IS WHERE YOUR GAME LOGIC GOES
350 PRINT "WAITING"
```

```
 355 GOTO 355
9999 END
```



```
0000                    ;
0000                    ; Interrupt Handler Firmware for the
0000                    ; CCS 7710A Programmable Timer Module
0000                    ;
0000                    ; System Linkage Equates
0000                    ;
0000                    ACC     EQU     $0045
0000                    IRQLOC  EQU     $03FE
0000                    RUNSAV  EQU     $E836
0000                    RETURN  EQU     $FFCB
0000                    ;
0000                    ; Timer Constants and Work Areas
0000                    ;
0000                    SCRNL   EQU     $0006
0000                    SCRNH   EQU     $0007
0000                    CONL    EQU     $0008
0000                    CONH    EQU     $0009
0000                    BINL    EQU     $0018
0000                    BINH    EQU     $0019
0000                    ZSUPR   EQU     $001A
0000                    T1CMD   EQU     $0478-$C0
0000                    T2CMD   EQU     $04F8-$C0
0000                    T3CMD   EQU     $0578-$C0
0000                    UDMASK  EQU     $05F8-$C0
0000                    T1DATL  EQU     $0678
0000                    T1DATH  EQU     $06F8
0000                    T2DATL  EQU     $0678-$C0
0000                    T2DATH  EQU     $06F8-$C0
0000                    T3DATL  EQU     $0778-$C0
0000                    T3DATH  EQU     $07F8-$C0
0000                    ;
0000                    ; Timer Register Addresses
0000                    ;
0000                    T1CR    EQU     $C080
0000                    T2CR    EQU     $C081
0000                    T3CR    EQU     T1CR
0000                    TSTAT   EQU     T2CR
0000                    T1DRL   EQU     $C082
0000                    T1DRH   EQU     $C083
0000                    T2DRL   EQU     $C084
0000                    T2DRH   EQU     $C085
0000                    T3DRL   EQU     $C086
0000                    T3DRH   EQU     $C087
0000                    ;
0000                    ; Initialization Entry Points
0000                    ;
0000 08                 BINIT   PHP             ;Save flags
0001 38                         SEC             ;Set EP indicator
0002 B0 02                      BCS     INIA
0004 08                 INIT    PHP             ;Normal init
0005 18                         CLC             ;Set EP flag
0006 2C CB FF           INIA    BIT     RETURN  ;Set V=1
0009 70 06                      BVS     COM     ;Skip Int EPs
000B 38                 BINT    SEC             ;BASIC Int EP
000C 90 00                      BCC     *       ;Skip next inst
000E                            ORG     * - 1
```

```
000D 18          INT     CLC             ;Normal EP
000E B8                  CLV             ;Set EP flag
000F A5 45               LDA     ACC     ;Restore A reg
0011             ;
0011             ; The Common Code to save registers
0011             ;   and set the indices
0011             ;
0011 48          COM     PHA             ;Save A reg
0012 8A                  TXA             ;Save X reg
0013 48                  PHA
0014 98                  TYA             ;Save Y reg
0015 48                  PHA
0016 20 CB FF            JSR     RETURN  ;Find slot page
0019 BA                  TSX             ;from stack
001A BC 00 01            LDY     $0100,X
001D 98                  TYA             ;Get IO index
001E 08                  PHP
001F 0A                  ASL
0020 0A                  ASL
0021 0A                  ASL
0022 0A                  ASL
0023 AA                  TAX             ;  into X
0024 28                  PLP
0025 50 55               BVC     INTA
0027             ;
0027             ; The Initialization Code
0027             ;
0027 A9 0B               LDA     BINT    ;Set Int vector
0029 F0 02               BCS     INITA
002B A9 0D               LDA     INT
002D 8D FE 03    INITA   STA     IRQLOC
0030 8C FF 03            STY     IRQLOC + 1
0033 B9 B8 03            LDA     T1CMD,Y ;Set commands
0036 09 01               ORA     #1
0038 9D 81 C0            STA     T2CR,X
003B 9D 80 C0            STA     T1CR,X
003E 0A                  ASL             ;Set bit 0=0
003F 9D 81 C0            STA     T2CR,X
0042 B9 B8 04            LDA     T3CMD,Y
0045 9D 80 C0            STA     T3CR,X
0048 B9 38 04            LDA     T2CMD,Y
004B 09 01               ORA     #1      ;Insure T1 access
004D 9D 81 C0            STA     T2CR,X
0050             ; Now, set the data registers
0050 AD F8 06            LDA     T1DATH
0053 9D 82 C0            STA     T1DRH,X
0056 AD 78 06            LDA     T2DATL
0059 9D 83 C0            STA     T1DRL,X
005C B9 38 06            LDA     T2DATH,Y
005F 9D 84 C0            STA     T2DRH,X
0062 B9 B8 05            LDA     T2DATL,Y
0065 9D 85 C0            STA     T2DRL,X
0068 B9 38 07            LDA     T3DATH,Y
006B 9D 86 C0            STA     T3DRH,X
006E B9 B8 06            LDA     T3DATL,Y
0071 9D 87 C0            STA     T3DRL,X
```



```
0074 68                  PLA             ;Restore regs
0075 A8                  TAY
0076 68                  PLA
0077 AA                  TAX
0078 68                  PLA
0079 28                  PLP
007A 58                  CLI             ;Enable ints
007B 60                  RTS             ;Return
007C             ;
007C             ; The BASIC Interrupt Handler
007C             ;
007C 90 13       INTA    BCC     INTB
007E B9 B8 03            LDA     T1CMD,Y ;Get cmd to
0081 09 01               ORA     #1      ; turn off
0083 9D 80 C0            STA     T1CR,X  ; timers
0086 68                  PLA             ;Restore regs
0087 A8                  TAY
0088 68                  PLA
0089 AA                  TAX
008A 68                  PLA
008B 28                  PLP             ;Set stack
008C 68                  PLA             ; straight
008D 68                  PLA
008E 4C 36 E8            JMP     RUNSAV  ;Go to BASIC
0091             ;
0091             ; The Normal Interrupt Handler
0091             ;
0091 98          INTB    TYA             ;Set up screen
0092 29 07               AND     #7      ;address vector
0094 6A                  ROR
0095 09 04               ORA     #4
0097 85 07               STA     SCRNH   ;Page byte
0099 6A                  ROR
009A 29 F0               AND     #$F0
009C 09 20               ORA     #$20
009E 85 06               STA     SCRNL   ;Low byte
00A0 BD 81 C0            LDA     TSTAT,X ;Allow int reset
00A3 BD 86 C0            LDA     T3DRH,X ;Get data
00A6 59 38 05            EOR     UDMASK,Y  ;Mask it
00A9 85 19               STA     BINH    ;Save for conv
00AB BD 87 C0            LDA     T3DRL,X ;Get rest data
00AE 59 38 05            EOR     UDMASK,Y  ;Mask it
00B1 85 18               STA     BINL    ;Save it, too
00B3 A9 A0               LDA     #$A0    ;Zero supr key
00B5 85 1A               STA     ZSUPR   ;Save it
00B7 84 09               STY     CONH    ;Set conv const
00B9 A9 F7               LDA     CONST-1 ; address
00BB 85 08               STA     CONL
00BD A0 08               LDY     #8      ;Loop control
```

```
00BF                    ;
00BF                    ; Hexadecimal to Decimal Conversion
00BF                    ;  This code uses a repetitive subtraction
00BF                    ; process for conversion.  It subtracts
00BF                    ; powers of ten (in binary form)
00BF                    ; to develop a 5 digit decimal number
00BF                    ; from a 16 bit unsigned binary number.
00BF                    ;
00BF A2 FF              INTC    LDX     #$FF    ;Dec digit init
00C1 E8                 INTD    INX
00C2 38                         SEC             ;No borrow
00C3 A5 18                      LDA     BINL
00C5 F1 08                      SBC     (CONL),Y ;Sub constant
00C7 48                         PHA             ;Temp store
00C8 88                         DEY             ;Rest of Const
00C9 A5 19                      LDA     BINH    ;Rest of number
00CB F1 08                      SBC     (CONH),Y
00CD 90 08                      BCC     INTE    ;This power done
00CF C8                         INY             ;Reset index
00D0 85 19                      STA     BINH    ;Save result
00D2 68                         PLA
00D3 85 18                      STA     BINL
00D5 B0 EA                      BCS     INTD    ;Try again
00D7 68                 INTE    PLA             ;Reset stack
00D8 8A                         TXA             ;Get digit
00D9 D0 06                      BNE     INTF    ;Skip zero supr
00DB A9 A0                      LDA     #$A0    ;Get a blank
00DD C5 1A                      CMP     ZSUPR   ;Prev digit?
00DF F0 04                      BEQ     INTG    ;Yes, skip
00E1 09 B0              INTF    ORA     #$B0    ;ASCII num zone
00E3 85 1A                      STA     ZSUPR   ;Kill zero sup
00E5 A2 00              INTG    LDX     #0      ;Zero index
00E7 81 06                      STA     (SCRNL,X) ;Onto screen
00E9 E6 06                      INC     SCRNL   ;Set pointer
00EB A5 18                      LDA     BINL    ;In case done
00ED 88                         DEY             ;Next 10 power
00EE F0 F1                      BEQ     INTF    ;Last digit?
00F0 10 CD                      BPL     INTC    ;Done?
00F2 68                         PLA             ;Yes, restore
00F3 A8                         TAY             ;  registers
00F4 68                         PLA
00F5 AA                         TAX
00F6 68                         PLA
00F7 40                         RTI             ;Return
00F8                    ;
00F8                    ; Conversion Constants
00F8                    ;
00F8 00 0A              CONST   DB      #0,#10  ; 10
00FA 00 64                      DB      #0,#100 ; 100
00FC 03 E8                      DB      #3,#232 ; 1000
00FE 27 10                      DB      #39,#16 ; 10000
0100                            END
```


Section IV Table of Contents

IV. TECHNICAL INFORMATION (continued)

SPECIFICATIONS:

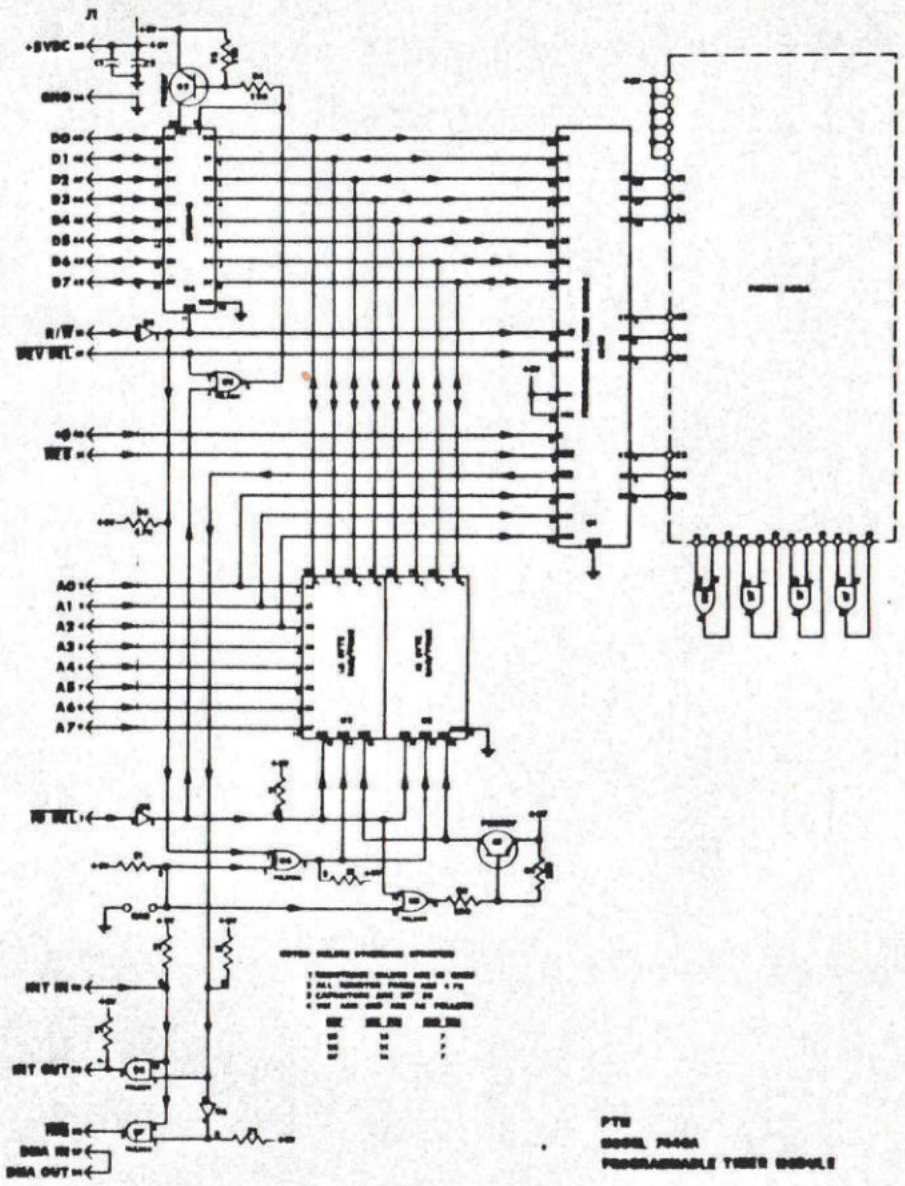
SIZE:	5"l x 2.75"h x 0.75"w (max)
WEIGHT:	< 5 oz.

SYSTEM INTERFACE

Internal:	APPLE II Peripherial slots 1 thru 7
External:	User defined via Patch Area
OPERATION MODES:	Continuous count Single shot Period measurement Pulse width measurement
PROGRAM MEMORY:	ROM (Mask) or PROM (Fuse link) or RAM (Static: 2-2112's)
Size:	256 (x 8) Bytes
Note:	ROM/PROM Auto Powered Down
REQUIRED POWER:	+5v dc
SALIENT FEATURES:	For other features, see the 6840 Data Sheet for details For applications, see the Motorola publication #MC6840UM (AD): "PROGRAMMABLE TIMER Fundamentals and Applications" Patch Area on .1" grid, adequate for most applications Supports Daisy Chain Interrupts with on board arbitration logic Allows DMA Daisy Chain (Pass thru) Glass Epoxy (FR-4) PC Board Gold Plated Connector Fingers Solder Mask both sides of board Component Silkscreen

V. TECHNICAL SP[illegible]FICATIONS (continued)

Schematic/Logic Diagram



IV. TECHNICAL INFORMATION (continued)

SPECIFICATIONS:

SIZE:	5"l x 2.75"h x 0.75"w (max)
WEIGHT:	< 5 oz.

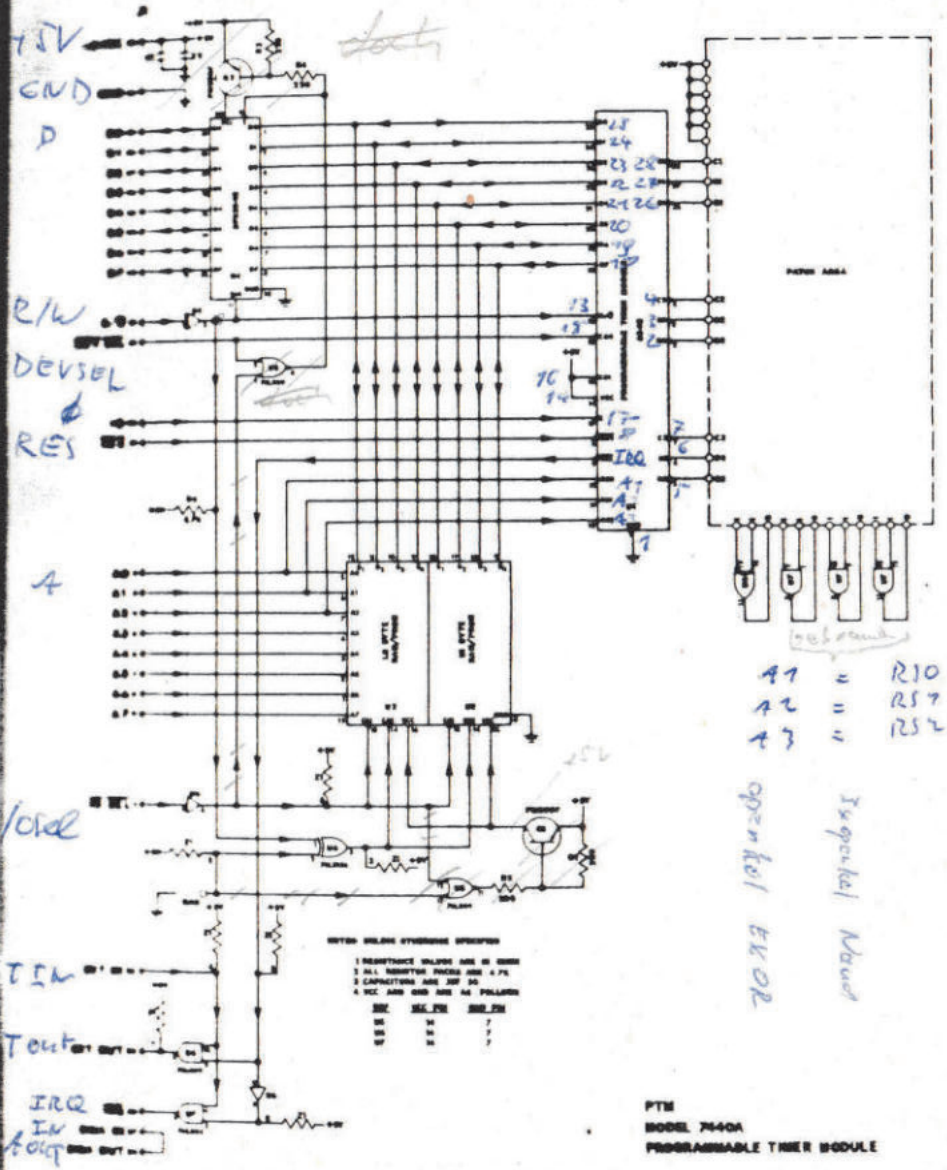
SYSTEM INTERFACE

Internal:	APPLE II Peripherial slots 1 thru 7
External:	User defined via Patch Area

OPERATION MODES:	Continuous count Single shot Period measurement Pulse width measurement
PROGRAM MEMORY:	ROM (Mask) or PROM (Fuse link) or RAM (Static: 2-2112's)
Size:	256 (x 8) Bytes
Note:	ROM/PROM Auto Powered Down
REQUIRED POWER:	+5v dc
SALIENT FEATURES:	For other features, see the 6840 Data Sheet for details For applications, see the Motorola publication #MC6840UM (AD): "PROGRAMMABLE TIMER Fundamentals and Applications" Patch Area on .1" grid, adequate for most applications Supports Daisy Chain Interrupts with on board arbitration logic Allows DMA Daisy Chain (Pass thru) Glass Epoxy (FR-4) PC Board Gold Plated Connector Fingers Solder Mask both sides of board Component Silkscreen

V. TECHNICAL SPECIFICATIONS (continued)

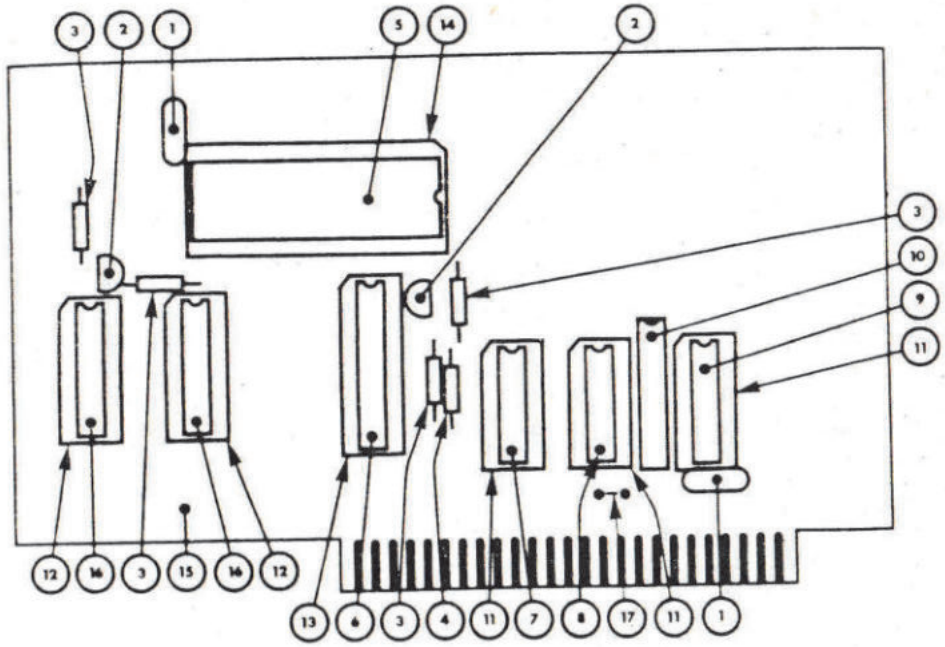
Schematic/Logic Diagram



PARTS LIST - PTM (Assy 07440-0001A)
CCS 7440A PROGRAMMABLE TIMER MODULE

#	QTY	REF	CCS P/N	DESCRIPTION
1	2	C1,2	42034-21048	CAPACITOR, MONOLYTHIC .1uf, 50vdc
2	2	Q1,2	36100-02907	TRANSISTOR, SI; PNP GENERAL PURPOSE, PN2907
3	4	R1-4	40002-02215	RESISTOR, FIXED, COMP 220ohm, 1/4w, 10%
4	1	R5	40002-04725	RESISTOR, FIXED, COMP 4.7k, 1/4w, 10%
5	1	U1	31107-00040	IC, DIGITAL, MOS; 6840 PROGRAMMABLE TIMER
6	1	U4	30505-8304B	IC, DIGITAL, TTL; 8304B OCTAL BUS DRVR/RCVR
7	1	U5	30002-00009	IC, DIGITAL, TTL; 74LS09 QUAD 2IN AND (OC)
8	1	U6	30002-00136	IC, DIGITAL, TTL; 74LS136 QUAD 2IN EX-OR (OC)
9	1	U7	30002-00003	IC, DIGITAL, TTL; 74LS03 QUAD 2IN NAND (OC)
10	1	Z1	40930-74726	RESISTOR NETWORK, SIP 4.7K x 7
11	3	XU5-7	58102-00140	SOCKET, IC; LOW PROFILE 14 PIN DIP
12	2	XU2,3	58102-00160	SOCKET, IC; LOW PROFILE 16 PIN DIP
13	1	XU4	58102-00200	SOCKET, IC; LOW PROFILE 20 PIN DIP
14	1	XU1	58102-00280	SOCKET, IC; LOW PROFILE 28 PIN DIP
15	1	-	07440-0002A	BOARD, PC PTM, REV A
16	2	U2,3	XXXXX-XXXXX	IC, DIGITAL; 256X4 PROM UNBURNED
17	-	-	XXXXX-XXXXX	WIRE, BARE; BUS #22 AWG, TINNED
			XXXXX-XXXXX	IC, DIGITAL, 256X4 RAM 2112
			XXXXX-XXXXX	ROM-PAK, 7440A STD
-	1	-	07440-9001A	MANUAL PTM, REV A
-	1	-	xxxxx-xxxxx	CARTON, STOWAGE

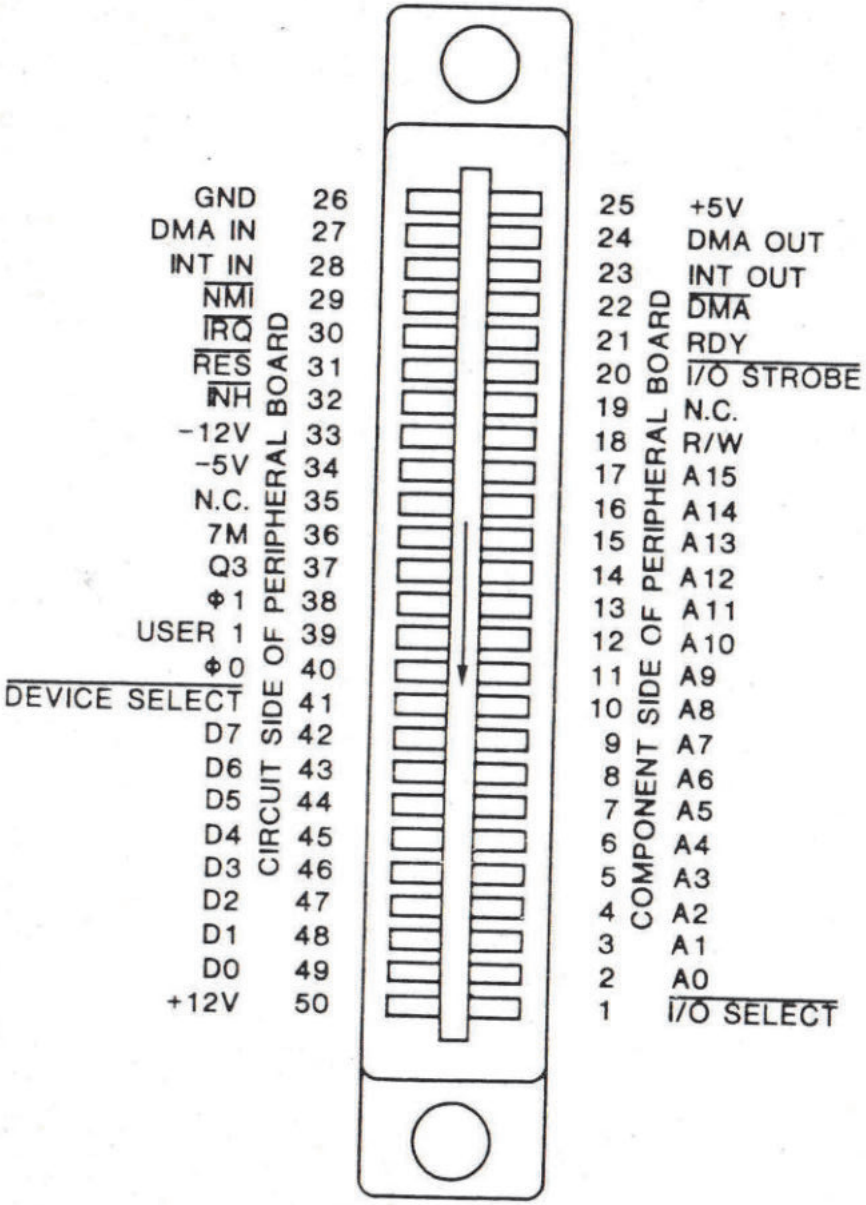
PARTS BREAKDOWN - PROGRAMMABLE TIMER MODULE



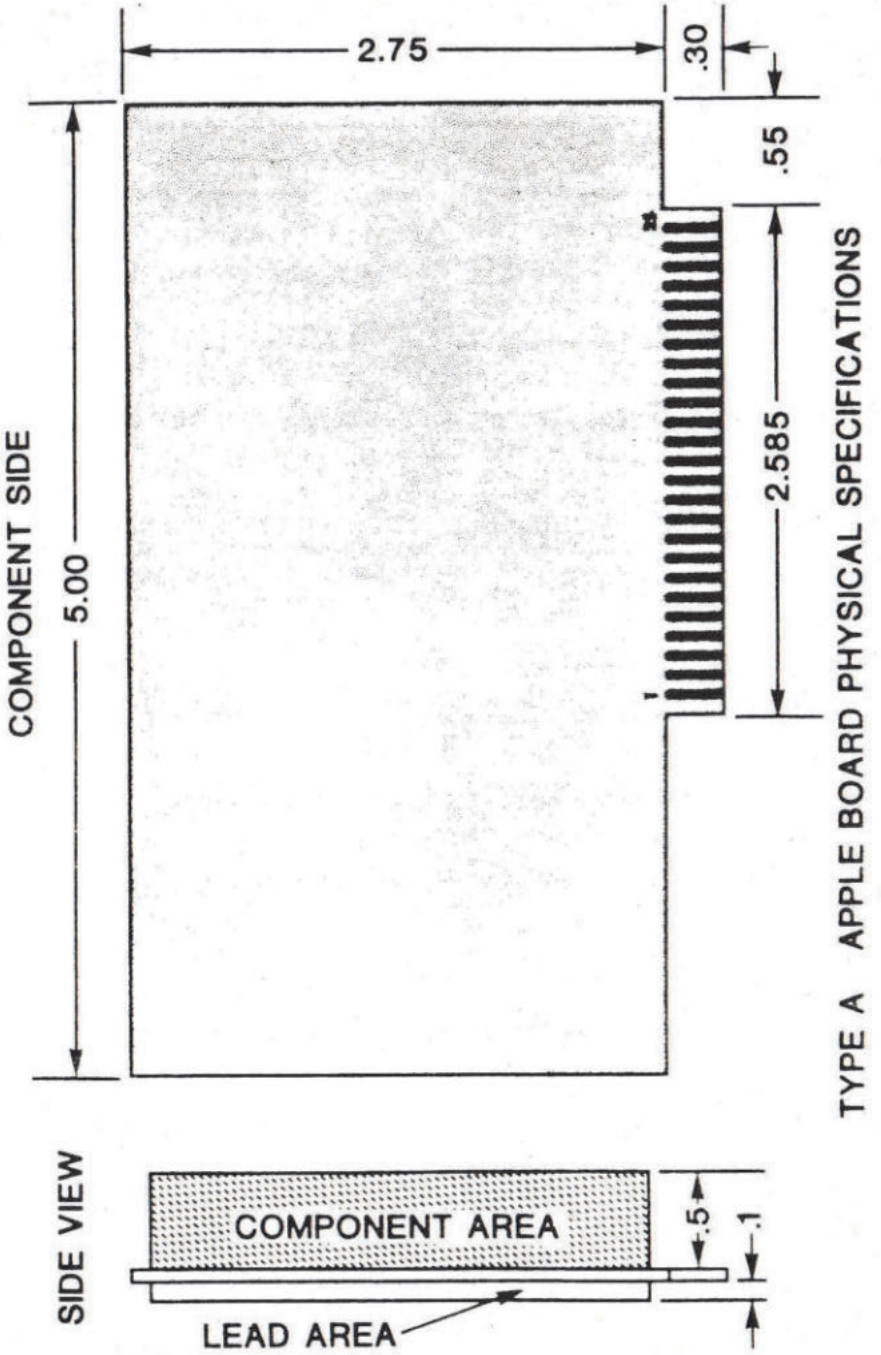
All numbers reference parts list.

Parts breakdown Programmable Timer Module

PERIPHERAL CONNECTOR PINOUT
TOP VIEW
BACK OF APPLE MAIN BOARD



FRONT OF APPLE MAIN BOARD



TYPE A APPLE BOARD PHYSICAL SPECIFICATIONS

V. LIMITED WARRANTY

LIMITED WARRANTY

California Computer Systems (CCS) warrants to the original purchaser of its products that

1.) its CCS assembled and tested products will be free from materials defects for a period of one (1) year, and be free from defects of workmanship for a period of ninty (90) days; and
2.) its kit products will be free from materials defects for a period of ninety (90) days.

The responsibility of CCS hereunder, and the sole and exclusive remedy of the original purchaser for a breach of any warranty hereunder, is limited to the correction or replacement by CCS at CCS's option, at CCS's service facility, of any product or part which has been returned to CCS and in which there is a defect covered by this warranty; provided, however, that in the case of CCS assembled and tested products, CCS will correct any defect in materials and workmanship free of charge if the product is returned to CCS within (90) days of original purchase from CCS; and CCS will correct defects in materials in its products and restore the product to an operational status for a labor charge of $25.00, provided that the product is returned to CCS within ninety (90) days in the case of kit products, or one (1) year in the case of CCS assembled and tested products. All such returned products shall be shipped prepaid and insured by original purchaser to:

Warranty Service Department
California Computer Systems
309 Laurelwood Road
Santa Clara, California
95050

CCS shall have the right of final determination as to the existence and cause of a defect, and CCS shall have the sole right to decide whether the product should be repaired or replaced.

This warranty shall not apply to any product or any part thereof which has been subject to

(1) accident, neglect, negligence, abuse or misuse;
(2) any maintenance, overhaul, installation, storage, operation, or use, which is improper; or
(3) any alteration, modification, or repair by anyone other than CCS or its authorized representative.

THIS WARRANTY IS EXPRESSLY IN LIEU OF ALL OTHER WARRANTIES EXPRESSED OR IMPLIED OR STATUTORY INCLUDING THE WARRANTIES OF DESIGN, MERCHANTABILITY, OR FITNESS OR SUITABILITY FOR USE OR INTENDED PURPOSE AND OF ALL

V. LIMITED WARRANTY (continued)

OTHER OBLIGATIONS OR LIABILITIES OF CCS. To any extent that this warranty cannot exclude or disclaim implied warranties, such warranties are limited to the durration of this express warranty or to any shorter time permitted by law.

CCS expressly disclaims any and all liability arising from the use and/or orreration of its products sold in any and all applications not specifically recommended, tested, or certified by CCS, in writing. With respect to applications not speciffically recommended, tested, or certified by CCS, the original purchaser acknowledges that he has examined the products to which this warranty attaches, and their specifications and descriptions, and is familiar with the operational characteristics thereof. The original purchaser has not relied upon the judgement or any representations of CCS as to the suitability of any CCS product and acknowledges that CCS has no knowledge of the intended use of its products. CCS EXPRESSLY DISCLAIMS ANY LIABILITY ARISING FROM THE USE AND/OR OPERATION OF ITS PRODUCTS, AND SHALL NOT BE LIABLE FOR ANY CONSEQUENTIAL OR INCIDENTAL OR COLLATERAL DAMAGES OR INJURY TO PERSONS OR PROPERTY.

CCS's obligations under this warranty are conditioned on the original purchaser's maintence of explicit records which will accurately reflect operating conditions and maintenance preformed on CCS's products and establish the nature of any unsatisfactory condition of CCS's products. CCS, at its request, shall be given access to such records for substantiating warranty claims. No action may be brought for breach of any express or implied warranty after one (1) year from the expiration of this express warranty's applicable warranty period. CCS assumes no liability for any events which may arise from the use of technical information on the application of its products supplied by CCS. CCS makes no warranty whatsoever in respect to accessories or parts not supplied by CCS, or to the extent that any defect is attributable to any part not supplied by CCS.

CCS niether assumes nor authorizes any person other than a duly authorized officer or representative to assume for CCS any other liability or extension or alteration of this warranty in connection with the sale or any shipment of CCS's products. Any such assumption of liability or modification of warranty must be in writing and signed by such duly authorized officer or representative to be enforceable. These warranties apply to the orginal purchaser only, and do not run to successors, assigns, or subsequent purchasers or owners; AS TO ALL PERSONS OR ENTITIES OTHER THAN THE ORIGINAL PURCHASER, CCS MAKES NO WARRANTIES WHATSOEVER, EXPRESS OR IMPLIED OR

V. LIMITED WARRANTY (continued)

STATUTORY. The term "original purchaser", as used in this warranty shall be deemed to mean only that person to whom its product is originally sold by CCS.

Unless otherwise agreed, in writing, and except as may be necessary to comply with this warranty, CCS reserves the right to make changes in its products without any obligation to incorporate such changes in any product manufactured theretofore.

This warranty is limited to the terms stated herein. CCS disclaims all liability for incidental or consequential damages. Some states do not allow limitations on how long an implied warranty lasts and some do not allow the exclusion or limitation of incidental or consequential damages so the above limitations and exclusions may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

MC6840 PROGRAMMABLE TIMER

- **THREE 16 BIT COUNTER/TIMERS**
- **PROGRAM CONTROLLED GENERATION OF SQUARE WAVES, PULSE TRAINS AND SINGLE PULSES**
- **PROGRAM CONTROL OF INTERRUPTS, OUTPUTS, COUNTER CONFIGURATION, AND CLOCK SOURCE**
- **28 PIN PACKAGE**

E1997

The MC6840 Programmable Timer Module (PTM) is a programmable subsystem component of the M6800 family designed to provide variable system time intervals.

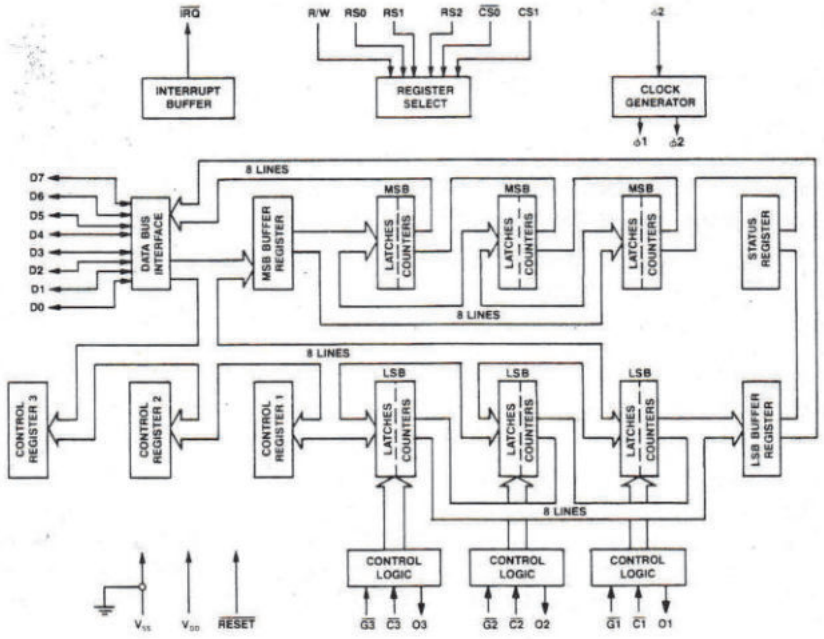
The MC6840 has three 16-bit binary counters, three corresponding control registers, and a status register. These counters are under software control and may be used to cause system interrupts and/or generate output signals. The MC6840 may be utilized for such tasks as frequency measurements, event counting, interval measuring, the generation of square waves, gated delay signals, single pulses of controlled duration, and pulse width modulation.

Typical uses for the PTM are process control, communications control, testing, point of sale terminals, game systems, data processing, and floppy disk systems.

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

The various internal parts of the PTM are illustrated in the Block Diagram below.

BLOCK DIAGRAM



E1996

FIGURE 1.

A. **Data Lines (D0 through D7)**

The eight bidirectional data lines permit transfer of data to/from the PTM and the MPU. The data bus output drivers are three-state devices that remain in the high impedance (off) state except when the MPU performs the PTM read operation. (Read/Write and Enable Lines high and PTM Chip Selects activated.)

B. **Read/Write Line (R/W)**

This signal is generated by the MPU to control the direction of the data transfers on the data bus. A low state on the PTM Read/Write line enables the input buffers and data is transferred from the MPU to the PTM (MPU write) on the falling edge of the Enable ($\phi 2$) signal if the device has been selected. Alternately (under the same conditions), a high on the Read/Write line sets up the PTM for a transfer of data to the data bus (MPU read).

C. **Chip Select Lines ($\overline{CS0}$, CS1)**

These are the lines which are tied to the address lines of the MPU. It is through these lines that the PTM is selected. With ($\overline{CS0}$ = 0 and CS1 = 1), the device is selected and data transfer will occur.

D. **Register Select Lines (RS0, RS1, RS2)**

These inputs are used in conjunction with the R/W line to select the internal registers, counters, and latches. Since the PTM uses the R/W line as an additional register select input, DO NOT USE M6800 INSTRUCTIONS WHICH PERFORM OPERATIONS DIRECTLY ON MEMORY. These instructions actually fetch a byte of memory, perform an operation, then restore it to the same address location. The modified data may not be restored to the same register if these instructions are used.

E. **Enable Line ($\phi 2$)**

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

F. **Interrupt Request Line ($\overline{IRQ}$)**

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

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

G. **Reset Line**

A low level at this input is clocked into the PTM by the Enable (System $\phi 2$) input. Two Enable pulses are required to synchronize and process the signal. The PTM then recognizes the active "low" or inactive "high" on the third Enable pulse. If the Reset signal is asynchronous, an additional Enable period is required if setup times are not met. The Reset input must be stable High/Low for the minimum time stated in the AC Operating Characteristics.

When the $\overline{\text{RESET}}$ input is low the following action occurs:

A. All counter latches are preset to their maximal count values.

B. All control register bits are cleared with the exception of CR10 (internal reset bit) which is set.

C. All counters are preset to the contents of the latches.

D. All counters outputs are reset and all counter clocks are disabled.

E. All status register bits (interrupt flags) are cleared.

FIGURE 2. E4502-1

H. **Clock Input Lines ($\overline{C1}$, $\overline{C2}$, $\overline{C3}$)**

Input pins $\overline{C1}$, $\overline{C2}$, and $\overline{C3}$ will accept asynchronous TTL voltage level signals to decrement Timers 1, 2, and 3, respectively. The high and low levels of the external clocks must each be stable for at least one system clock period plus the sum of the setup and hold times of the inputs. The asynchronous clock rate can vary from dc to the limit imposed by Enable (System $\phi 2$), Setup, and Hold time.

The external clock inputs are clocked in by Enable (System $\phi 2$) pulses. Three Enable periods are used to synchronize and process the external clock. The fourth Enable pulse decrements the internal counter. This does not affect the input frequency, it merely creates a delay between a clock input transition and internal recognition of that transition by the PTM. All references to C inputs relate to internal recognition of the input transition. Note that a clock high or low level which does not meet setup and hold time specifications may require an additional Enable pulse for recognition. When observing recurring events, a lack of synchronization will result in "jitter" being observed on the output of the PTM when using asynchronous clocks and gate input signals. There are two types of jitter. "System jitter" is the result of the input signals being out of synchronization with the Enable (System $\phi 2$), permitting signals with marginal setup and hold time to be recognized by either the bit time nearest the input transition or the subsequent bit time.

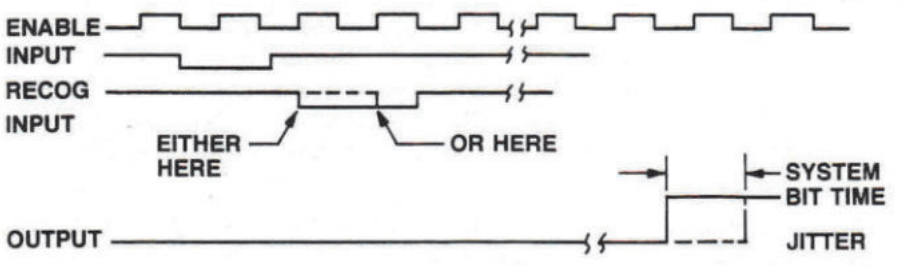


FIGURE 3. E4502A

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

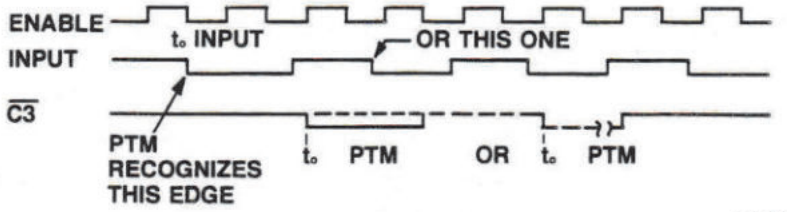


FIGURE 4.

External clock input $\overline{C3}$ represents a special case when Timer #3 is programmed to utilize its optional ÷8 prescaler mode. The maximum input frequency and allowable duty cycles for this case are specified in the data sheet under the AC Operating Characteristics. The output of the ÷8 prescaler is treated in the same manner as the previously discussed clock inputs. That is, it is clocked into the counter by Enable pulses, is recognized on the fourth Enable pulse (provided setup and hold time requirements are met), and must produce an output pulse at least as wide as the sum of an Enable period, setup, and hold times.

I. **Gate Input Lines ($\overline{G1}$, $\overline{G2}$, $\overline{G3}$)**

Input pins $\overline{G1}$, $\overline{G2}$, and $\overline{G3}$ accept asynchronous TTL compatible signals which are used as triggers or clock gating functions to Timers 1, 2, and 3, respectively. The gating inputs are clocked into the PTM by the Enable (System $\phi 2$) signal in the same manner as the previously discussed clock inputs. That is, a Gate transition is recognized by the PTM on the fourth Enable pulse (provided setup and hold time requirements are met), and the high or low levels of the Gate input must be stable for at least one sytem clock period plus the sum of setup and hold times. All references to G transition in this document relate to internal recognition of the input transition.

The Gate inputs of all timers directly affect the internal 16-bit counter. The operation of $\overline{G3}$ is therefore independent of the ÷8 prescaler selection.

J. **Timer Output Lines (O1, O2, O3)**

Timer outputs O1, O2, and O3 are capable of driving up to two TTL loads and produce a defined output waveform for either Continuous or Single-Shot Timer modes. Output waveform definition is accomplished by selecting either Single 16-bit or Dual 8-bit operating modes. The single 16-bit mode will produce a square-wave output in the continuous timer mode and will produce a single pulse in the Single-Shot Timer mode. The Dual 8-bit mode will produce a variable duty cycle pulse in both the continuous and single shot Timer modes. One bit of each Control Register (CRX7) is used to enable the corresponding output. If this bit is cleared, the output will remain low (V_{OL}) regardless of the operating mode.

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

Since the PTM data bus is 8-bits wide and the counters are 16-bits wide, a temporary register (MSB Buffer Register) is provided. This "write only" register is for the Most Significant Byte for the desired latch data. Three addresses are provided for the MSB Buffer Register but they all lead to the same Buffer.

Data from the MSB Buffer will automatically be transferred into the MSB Latches of Timer #X when a Write Timer #X LSB Latches command is performed, i.e., the first instruction - write to the MSB Buffer Register with data AA - the result is; data AA is stored into the MSB Buffer Register. The next instruction - write to the LSB Latches #1 with data BB. The result is; data AA which was stored in the MSB Buffer Register is transferred to the MSB Latches #1 and data BB is stored into LSB Latches #1.

PROGRAMMABLE TIMER PROGRAMMING MODEL

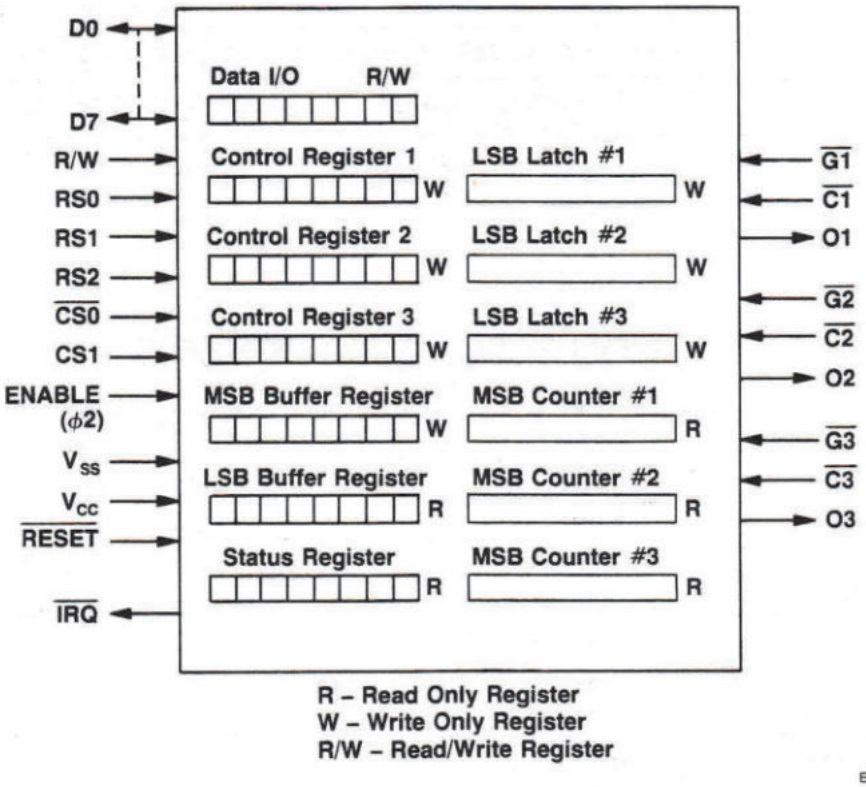


FIGURE 5.

When reading one of the timers, the MS Byte contents can be read directly with a Read MSB Counter command. This places the LS Byte contents of the selected counter into the LS Byte Buffer register. A Read LS Byte Buffer register command can then be used to obtain the data from the LS Byte Buffer register.

REGISTER SELECTION

REGISTER SELECT INPUTS			OPERATIONS	
RS2	RS1	RS0	R/W = 0	R/W = 1
0	0	0	CR20=0 WRITE CONTROL REGISTER #3 CR20=1 WRITE CONTROL REGISTER #1	NO OPERATION
0	0	1	WRITE CONTROL REGISTER #2	READ STATUS REGISTER
0	1	0	WRITE MSB BUFFER REGISTER	READ TIMER #1 COUNTER
0	1	1	WRITE TIMER #1 LATCHES	READ LSB BUFFER REGISTER
1	0	0	WRITE MSB BUFFER REGISTER	READ TIMER #2 COUNTER
1	0	1	WRITE TIMER #2 LATCHES	READ LSB BUFFER REGISTER
1	1	0	WRITE MSB BUFFER REGISTER	READ TIMER #3 COUNTER
1	1	1	WRITE TIMER #3 LATCHES	READ LSB BUFFER REGISTER

FIGURE 6.

E4503

CONTROL REGISTER BITS

CR10 Internal Reset Bit	CR20 Control Register Address Bit	CR30 Timer #3 Clock Control
0 All timers allowed to operate	0 CR#3 may be written	0 T3 Clock is not prescaled
1 All timers held in preset state	1 CR#1 may be written	1 T3 clock is prescaled by ÷8

CRX1	Timer #X Clock Source
0	TX uses external clock source on $\overline{CX}$ input
1	TX uses system clock
CRX2	Timer #X Counting Mode Control
0	TX configured for normal (16-bit) counting mode
1	TX configured for dual 8 bit counting mode
CRX3 CRX4 CRX5	Timer #X Counter Mode and Interrupt Control
CRX6	Timer #X Interrupt Enable
0	Interrupt flag masked on $\overline{IRQ}$
1	Interrupt flag enabled to $\overline{IRQ}$
CRX7	Timer #X Counter Output Enable
0	TX output masked on output OX
1	TX output enabled on output OX

X = Registers 1, 2 or 3

FIGURE 7.

E4504

Control Registers

Three Write-Only registers in the MC6840 are used to control the timer mode of operation; enable interrupts, enable outputs, select external clock, and control setting of interrupts. Each control register controls its respective counter/timer operation with the exception of bit 0 in Control Register #1 (external reset for all counters) and bit 0 in Control Register #2 (control register address bit).

Control Register #1 — Bit 0

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

Control Register #2 — Bit 0

The least significant bit of Control Register #2 is used as an additional addressing bit for Control Registers #1 and #3. This bit in conjunction with the register select lines (RS0, RS1, and RS2) is used to select either Control Register #1 or Control Register #3. To address Control Register #1; RS0, RS1, and RS2 are all set to a 0 and Control Register #2 Bit 0 is set to a 1. To address Control Register #3; RS0, RS1, and RS2 are all set to a 0 and Control Register #2 Bit 0 is set to a 0. Control Register #3 can also be written into after a Reset low condition has occurred, since all control register bits (except Control Register #1 Bit 0) are cleared. Therefore, the sequence in writing to the timer control registers might be Control Register #3, Control Register #2, and Control Register #1.

Control Register #3 — Bit 0

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

Control Register #1, #2, and #3 — Bit 1

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

Control Register #1, #2, and #3 — Bit 2

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

Contr. –

Counter Register #1, #2, and #3 — Bits 3, 4, and 5

Control Register Bits 3, 4, and 5 control four basic modes of operation available with the PTM and each of these has two variations making a total of eight selections from which to choose. These modes of operation are better discussed in a separate section (**Timer Operating Modes**).

Control Register #1, #2, #3 — Bit 6

Control Register Bit 6 is the Interrupt Request (IRQ) masking bit. When Control Register #1 Bit 6 is set to an "0" the interupt line (IRQ) to the MPU is masked, i.e., a Time Out (T.O.) has occurred on Timer #1 but the IRQ line to the MPU will remain high. Timer #2 and Timer #3 may still cause an interrupt (if unmasked Bit 6 = "1") to the MPU when a T.O. occurs in either Timer #2 or Timer #3. The masking of the IRQ in one timer will not affect the interrupt requests of the other timers.

Control Register #1, #2, and #3 — Bit 7

Bit 7 of each control register determines if a timer output is enabled or disabled and does not affect the operation of the timer in any way. If the timer is operating with this bit cleared. (0 = output masked), the control register can be written to with the same data, except for setting bit 7 (enable output), and the operation will not be affected. The output will appear when bit 7 is set (1).

MC6840 CONTROL REGISTER PROGRAMMING

7 6 5 4 3 2 1 0		REGISTER 1	REGISTER 2	REGISTER 3
X X X X X X X ↕	0	ALL TIMERS OPERATE	REG #3 MAY BE WRITTEN	T3 $\overline{CLK}$ ÷ 1
	1	ALL TIMERS PRESET	REG #1 MAY BE WRITTEN	T3 $\overline{CLK}$ ÷ 8

7 6 5 4 3 2 1 0		
X X X X X X ↕ X	0	EXTERNAL CLOCK ($\overline{CX}$ INPUT)
	1	INTERNAL CLOCK (ENABLE)
X X X X X ↕ X X	0	NORMAL (16 BIT) COUNT MODE
	1	DUAL 8 BIT COUNT MODE
X X X ↕ 0 X X X	0	WRITE TO TIMERS WILL INITIALIZE
	1	WRITE TO TIMERS WILL NOT INITIALIZE
X X X ↕ 1 X X X	0	FREQUENCY COMPARISON
	1	PULSE WIDTH COMPARISON
X X ↕ X 0 X X X	0	CONTINUOUS OPERATING MODE
	1	SINGLE-SHOT MODE
X X ↕ X 1 X X X	0	INTERRUPT IF LESS THAN COUNTERTIME OUT
	1	INTERRUPT IF GREATER THAN COUNTER TIME OUT
X ↕ X X X X X X	0	INTERRUPT FLAG MASKED ($\overline{IRQ}$)
	1	INTERRUPT FLAG ENABLED ($\overline{IRQ}$)
↕ X X X X X X X	0	TIMER OUTPUT MASKED
	1	TIMER OUTPUT ENABLED

E5947

FIGURE 8.

STATUS REGISTER

Status Register Bit	Definition
SR0	Timer #1 Interrupt Flag
0	T1 Interrupt will not assert SR7 and $\overline{IRQ}$.
1	T1 Interrupt will assert SR7 and $\overline{IRQ}$ if CR16=1
SR1	Timer #2 Interrupt Flag
0	T2 Interrupt will not assert SR7 and $\overline{IRQ}$
1	T2 Interrupt will assert SR7 and $\overline{IRQ}$ if CR26=1
SR2	Timer #3 Interrupt Flag
0	T3 Interrupt will not assert SR7 and $\overline{IRQ}$
1	T3 Interrupt will assert SR7 and $\overline{IRQ}$ if CR36=1
SR3 thru SR6	Not used and will default to "0" when read
SR7	Composite Status Bit
0	No Timer is asserting $\overline{IRQ}$
1	CR16·SRO+CR26·SR1+CR36·SR2 is asserting $\overline{IRQ}$

FIGURE 9. E4506

Status Register

The MC6840 has an internal Read-Only Status Register which contains four Interrupt Flags. (The remaining four bits of the register are not used, and default to zero when being read.) Bits 0, 1, and 2 are assigned to Timers 1, 2, and 3, respectively, as individual flag bits, while Bit 7 is a Composite Interrupt Flag. This flag bit will be asserted if any of the individual flag bits is set while Bit 6 of the corresponding Control Register is at a logic one. The conditions for asserting the Composite Interrupt Flag bit can therefore be expressed as:

$$INT = I1 \cdot CR16 + I2 \cdot CR26 + I3 \cdot CR36$$

where INT = Composite Interrupt Flag (Bit 7)
I1 = Timer #1 Interrupt Flag (Bit 0)
I2 = Timer #2 Interrupt Flag (Bit 1)
I3 = Timer #3 Interrupt Flag (Bit 2)

An Interrupt flag is cleared by a Timer Reset condition, i.e., External Reset = 0 or Internal Reset Bit (CR10) = 1. It will also be cleared by a Read Timer Counter Command provided that the Status Register has previously been read while the interrupt flag was set. This condition on the Read Status Register-Read Timer Counter (RS-RT) sequence is designed to prevent missing interrupts which might occur after the status register is read, but prior to reading the Timer Counter.

Counter Latch Initialization

Each of the three independent timers consists of a 16-bit addressable counter (read only) and 16-bits of addressable latches (write only). The counters are preset to the binary numbers stored in the latches at the time a counter initialization takes place. Latch initialization is performed by writing to the MSB Buffer and to the LSB Latches (16

Bits of binary data). At the time of the write to the LSB Latches, the data in the MSB Buffer is transferred into the MSB Latches. It should be noted that when using the PTM with the M6800 family microprocessors, the 16 Bit Store instruction (STS and STX) may be used to transfer data in the order required by the PTM. A Store Index Register Instruction (STX), for example, results in the MSB of the Index register being transferred to the address of the MSB Buffer, then the LSB of the Index register being transferred to the next higher address which will be the LSB Latches.

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

Counter Initialization

Counter Initialization is defined as the transfer of data from the latches to the counter with subsequent clearing of the Individual Interrupt Flag associated with the counter. Counter Initialization always occurs when a reset condition (Reset = 0 or CR10 = 1) is recognized. It can also occur — depending on the Timer Mode — with a Write Timer Latches command or recognition of a negative transition of the Gate input.

Timer Operating Modes

Each of the individual timers in the PTM can be programmed for four basic operating modes. This is accomplished by using the three bits of each control register (CRX3, CRX4, and CRX5) to define the different operating modes. These modes are outlined in the following table.

OPERATING MODES

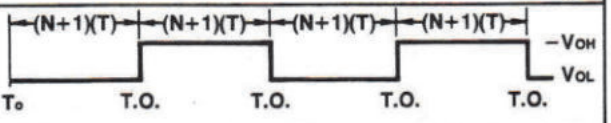
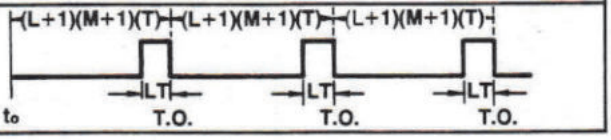
Control Register			Timer Operating Mode
CRX3	CRX4	CRX5	
0	*	0	Continuous
0	*	1	Single-Shot
1	0	*	Frequency Comparison
1	1	*	Pulse Width Comparison

*Defines Additional Timer Functions

E4506A

FIGURE 10.

CONTINUOUS OPERATING MODES

CONTINUOUS MODE (CRX_3=0, CRX_7=1, CRX_5=0			
REGISTER		INITIALIZATION/OUTPUT WAVEFORMS	
CRX_2	CRX_4	COUNTER INITIALIZATION	*TIMER OUTPUT (OX)
0	0	$\overline{G}\downarrow$ + W + R	
0	1	$\overline{G}\downarrow$ + R	
1	0	$\overline{G}\downarrow$ +W+R	
1	1	$\overline{G}\downarrow$ +R	

G↓ = Negative transition of $\overline{GATE}$ input.
W = Write Timer Latches Command.
R = Timer Reset (CR10=1 or External $\overline{RESET}$=0)
N = 16 Bit Number in Counter Latch.
L = 8 Bit Number in LSB Counter Latch.
M = 8 Bit Number in MSB Counter Latch.
T = Period of Clock Input to Counter.
to = Counter Initialization Cycle.
T.O. = Counter Time Out (All Zero Condition).

***All time intervals shown above assume that the Gate ($\overline{G}$) and Clock ($\overline{C}$) signals are synchronized to system ø2 with the specified setup and hold time requirements.**

E4507-1

FIGURE 11.

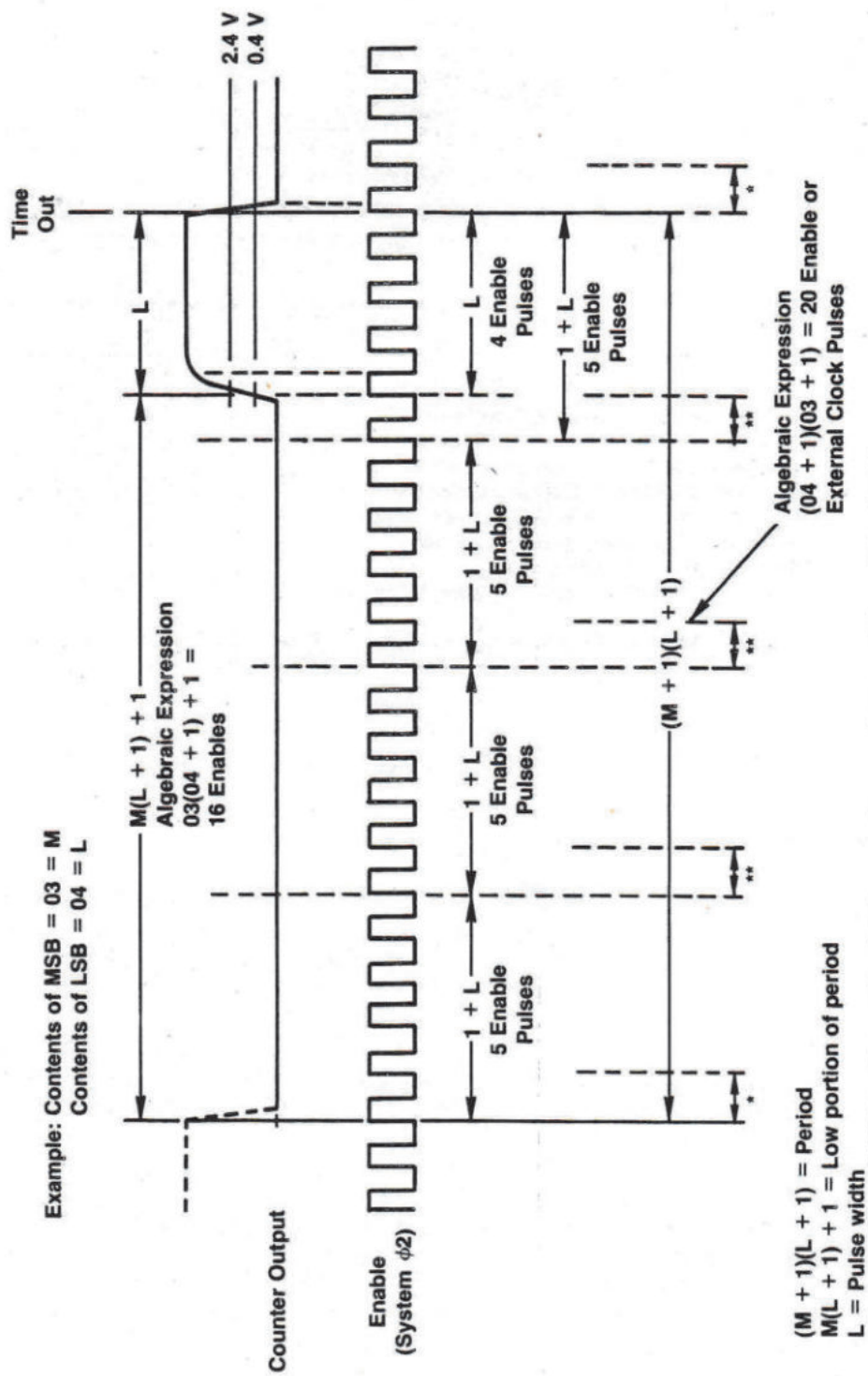
Continuous Operating Mode

Any of the timers in the PTM may be programmed to operate in a continuous mode by writing zeroes into bits 3 and 5 of the corresponding control register. Assuming that the timer output is enabled (CRX7 = 1), either a square wave or a variable duty cycle waveform will be generated at the Timer Output, OX. The type of output is selected via Control Register Bit 2.

Either a Timer Reset (CR10 = 1 or External Reset = 0) condition or internal recognition of a negative transition of the Gate input results in Counter Initialization. A Write Time Latches command can be selected as a Counter Initialization signal by clearing CRX4.

In the dual 8-bit mode (CRX2 = 1) (refer to the example in Figure 12) the MSB decrements once for every full countdown of the LSB + 1. When the LSB = 0, the MSB is unchanged; on the next clock pulse the LSB is reset to the count in the LSB Latches and the MSB is decremented by 1 (one). The output, if enabled, remains low during and after initialization and will remain low until the counter MSB is all zeroes. The output will go high at the beginning of the next clock pulse. The output remains high until both the LSB and the MSB of the counter are all zeroes. At the beginning of the next clock pulse the defined Time Out (TO) will occur and the output will go low. In the normal 16-bit mode, the period of the output of the example in Figure 12 would span 1546 clock pulses as opposed to the 20 clock pulses using the Dual 8-bit mode.

TIMER OUTPUT WAVEFORM EXAMPLE
(Continuous Dual 8-Bit Mode using Internal Enable)



E5946

FIGURE 12.

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

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

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

CASCADING IN THE CONTINUOUS MODE

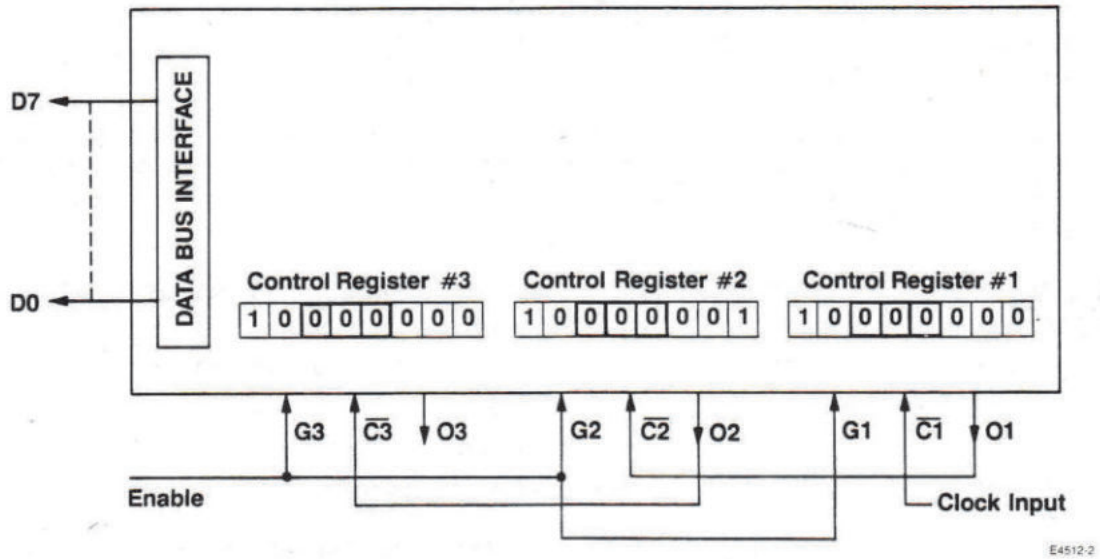


FIGURE 13.

Single Shot Operating Mode

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

SINGLE – SHOT OPERATING MODES

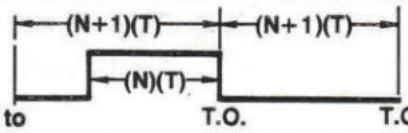
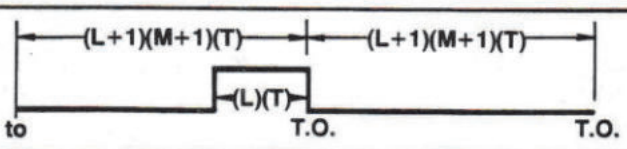
SINGLE – SHOT MODE (CRX_3=0, CRX_7=1, CRX_5=1)			
CONTROL REGISTER		INITIALIZATION/OUTPUT WAVEFORMS	
CRX_2	CRX_4	COUNTER INITIALIZATION	(TIMER OUTPUT (OX)
0	0	$\overline{G}\downarrow + W + R$	
0	1	$\overline{G}\downarrow + R$	
1	0	$\overline{G}\downarrow + W + R$	
1	1	$\overline{G}\downarrow + R$	

FIGURE 14.

E4508

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

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

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

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

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

Aside from these differences, the two modes are identical.

FREQUENCY COMPARISON MODE

CRX3=1, CRX4=0				
CONTROL REG BIT 5 (CRX5)	COUNTER INITIALIZATION	COUNTER ENABLE FLIP-FLOP SET (CE)	COUNTER ENABLE FLIP-FLOP RESET ($\overline{CE}$)	INTERRUPT FLAG SET (I)
0	$\overline{G}\downarrow\cdot\overline{I}\cdot(\overline{CE} + T.O.\cdot CE) + R$	$\overline{G}\downarrow\cdot\overline{W}\cdot\overline{R}\cdot\overline{I}$	W+R+I	$\overline{G}\downarrow$ BEFORE T.O.
1	$\overline{G}\downarrow\cdot\overline{I}+R$			T.O. BEFORE $\overline{G}\downarrow$

FIGURE 15.

E4510-2

Frequency Comparison Mode

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

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

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

Any of the timers within the PTM may be programmed to compare the period of a pulse (giving the frequency after calculations) at the Gate input with the time period required for Counter Time Out. A negative transition of the Gate input enables the counter and starts a Counter Initialization cycle - provided that other conditions as noted in the Table above are satisfied. The counter decrements on each clock signal recognized during or after Counter Initialization until an Interrupt is generated, a Write Timer Latches command is issued, or a Timer Reset condition occurs. It can be seen from the above Table that an interrupt condition will be generated if CRX5 = 0 and the period of the pulse (single pulse or measured separately repetitive pulses) at the Gate input is less than the Counter Time Out period. If CRX5 = 1, an interrupt is generated if the reverse is true.

Assume now with CRX5 = 1 that a Counter Initialization has occurred and that the Gate input has returned low prior to Counter Time Out. Since there is no individual Interrupt Flag generated, this automatically starts a new Counter Initialization Cycle. The process will continue with frequency comparison being performed on each Gate input cycle until the mode is changed, or a cycle is determined to be above the predetermined limit.

PULSE WIDTH COMPARISON MODE

CRX3=1, CRX4=1				
CONTROL REG BIT 5 (CRX5)	COUNTER INITIALIZATION	COUNTER ENABLE FLIP-FLOP SET (CE)	COUNTER ENABLE FLIP-FLOP RESET (CE)	INTERRUPT FLAG SET (I)
0	$\overline{G}\downarrow\cdot\overline{I}+R$	$\overline{G}\downarrow\cdot\overline{W}\cdot\overline{R}\cdot\overline{I}$	W+R+I+G	$\overline{G}\uparrow$ BEFORE T.O.
1				T.O. BEFORE $\overline{G}\uparrow$

FIGURE 16.

E4511

Pulse Width Comparison Mode

This mode is similar to the Frequency Comparison Mode except for a positive, rather than negative, transition of the Gate input terminates the count. With CRX5 = 0, an Indi-

vidual Interrupt Flag will be generated if the zero level pulse applied to the Gate input is less than the time period required for Counter Time Out. With CRX5 = 1, the interrupt is generated when the reverse condition is true.

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

TIME INTERVAL MODES

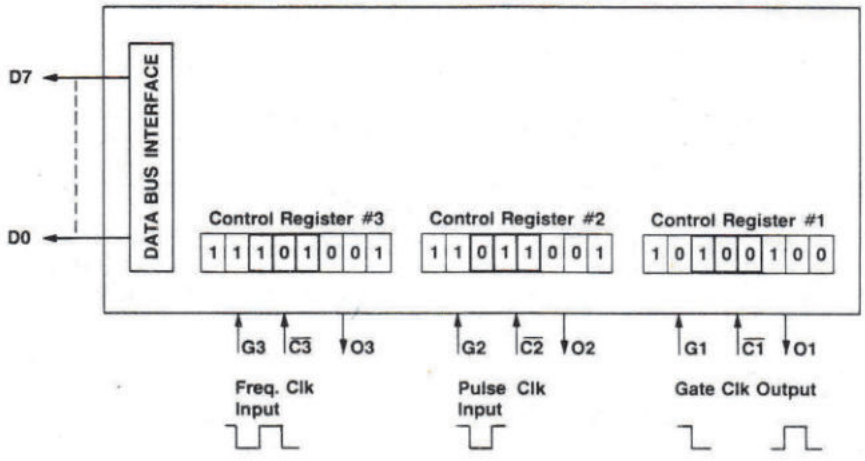
CRX3 = 1			
CRX4	CRX5	APPLICATION	CONDITION FOR SETTING INDIVIDUAL INTERRUPT FLAG
0	0	FREQUENCY COMPARISON	INTERRUPT GENERATED IF $\overline{\text{GATE}}$ INPUT PERIOD (1/F) IS LESS THAN COUNTER TIME OUT (T.O.)
0	1	FREQUENCY COMPARISON	INTERRUPT GENERATED IF $\overline{\text{GATE}}$ INPUT PERIOD (1/F) IS GREATER THAN COUNTER TIME OUT (T.O.)
1	0	PULSE WIDTH COMPARISON	INTERRUPT GENERATED IF $\overline{\text{GATE}}$ INPUT "DOWN TIME" IS LESS THAN COUNTER TIME OUT (T.O.)
1	1	PULSE WIDTH COMPARISON	INTERRUPT GENERATED IF $\overline{\text{GATE}}$ INPUT "DOWN TIME" IS GREATER THAN COUNTER TIME OUT (T.O.)

E4509

FIGURE 17.

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

The output signal is not defined in any of these modes, but the counter does operate in either Single 16-bit or Dual 8-bit modes as programmed by CRX2. Other features of the Time Interval Modes are outlined in above.



TIMER #1 — SINGLE SHOT MODE
TIMER #2 — PULSE WIDTH COMPARISON MODE
TIMER #3 — FREQUENCY COMPARISON MODE
Apply input to be compared to the $\overline{\text{GATE}}$ input of Timer #3.

E5948

FIGURE 18.

COUNTER MODE AND INTERRUPT CONTROL

MODE CRX5	CRX4	CRX3	COUNTER INITIALIZATION "CI"	COUNTER ENABLE "CE"	INTERRUPT FLAG Setting	INTERRUPT FLAG Clearing
0	0	0 (continuous mode)	$\overline{G\downarrow} + W + R$	$\overline{G} \cdot \overline{R}$	T.O.	RS-RT or CI
1	0	0 (single shot mode)	$\overline{G\downarrow} + W + R$	$\overline{R}$	T.O.	RS-RT or CI
0	1	0 (continuous mode)	$\overline{G\downarrow} + R$	$\overline{G} \cdot \overline{R}$	T.O.	RS-RT or CI or W
1	1	0 (single shot mode)	$\overline{G\downarrow} + R$	$\overline{R}$	T.O.	RS-RT or CI or W
0	0	1 (frequency (comparison mode)	$\overline{G\downarrow} \cdot \overline{I} \cdot (\overline{CE} + T.O.) + R$	$CB_{set} = \overline{G\downarrow} \cdot \overline{W} \cdot \overline{R} \cdot \overline{I}$ $CB_{reset} = W+R+I$	$\overline{G\downarrow}$ before T.O.	RS-RT or CI or W
1	0	1 (frequency (comparison mode)	$\overline{G\downarrow} \cdot \overline{I} + R$	$CB_{set} = \overline{G\downarrow} \cdot \overline{W} \cdot \overline{R} \cdot \overline{I}$ $CB_{reset} = W+R+I$	T.O. before $\overline{G\downarrow}$	RS-RT or CI or W
0	1	1 (pulse width (comparison mode)	$\overline{G\downarrow} \cdot \overline{I} + R$	$CB_{set} = \overline{G\downarrow} \cdot \overline{W} \cdot \overline{R} \cdot \overline{I}$ $CB_{reset} = W+R+I+G$	$\overline{G\uparrow}$ before T.O.	RS-RT or CI or W
1	1	1 (pulse width (comparison mode)	$\overline{G\downarrow} \cdot \overline{I} + R$	$CB_{set} = \overline{G\downarrow} \cdot \overline{W} \cdot \overline{R} \cdot \overline{I}$ $CB_{reset} = W+R+I+G$	T.O. before $\overline{G\uparrow}$	RS-RT or CI or W

R = Ext. $\overline{RESET}$ or internal reset
W = Write Timer Latches command
I = Interrupt Flag
$\overline{G}$ = Internal recognition of Gate signal
C = Internal recognition of selected clock signal
$\overline{G\downarrow}$ = Internal recognition of $\overline{G}$ going low
$\overline{G\uparrow}$ = Internal recognition of $\overline{G}$ going high
RS-RT = Read Status Reg.-Read Timer Counter commands (normal mode of clearing interrupts) if the interrupt flag was set when the Status Register was read.
T.O. = time out point at which an interrupt may occur, if enabled (all zero condition)

E4505

FIGURE 19.

Programming Problem

An electronics manufacturer is using an M6800 microprocessor, with a 1 MHz clock, to control the testing of modules. A new product has been developed and a small modification may be required to do the additional testing. It was determined that by adding the M6840 timer to the M6800 system, the additional test conditions can be met. The functions the timer must perform are:

a) The timer must output to the tester a positive pulse 200 microseconds wide. This pulse must fall within 25 to 27 milliseconds after the initialization of each module test. At the initialization of each module test the tester will output a negative 5 microsecond pulse.

b) The timer is to measure a negative going pulse to that the pulse width is within 1 millisecond ±5%. If this pulse width is less than 1 ms −5% or greater than 1 ms +5%, an interrupt is required to the microprocessor.

Show the interfacing lines between the tester and the PTM. Also the information required in each of the timer latches and Control Registers.

PTM Programming Problem Solution

1. Determine the binary number required in the latches to produce a delayed 200 μs pulse. Timer #3 was selected to output the 200 μs pulse delayed by approximately 25,900 μs (center of window minus one-half the output pulse width - this will center the output pulse within the window). Time Out will occur at the end of the pulse or at 26,100 μs. Using the Single-Shot Mode Table, the formula (L + 1) (M + 1) (T) = T.O. will determine the pulse end and Time Out (T.O.). Since T.O. = 26,100 μs,

$$M = \frac{26{,}100}{(L+1)(T)} - 1 = \frac{26{,}100}{201} - 1 = 128.85 \text{ or } 129$$

(L+1) (M+1) = T.O. = 26,130 μs (end of pulse) and 26,130 −200 = 25,930 μs (start of pulse). Timer #3 latches are programmed with $M = 129_{10} = 81_{16}$

$L = 200_{10} = C8_{16}$

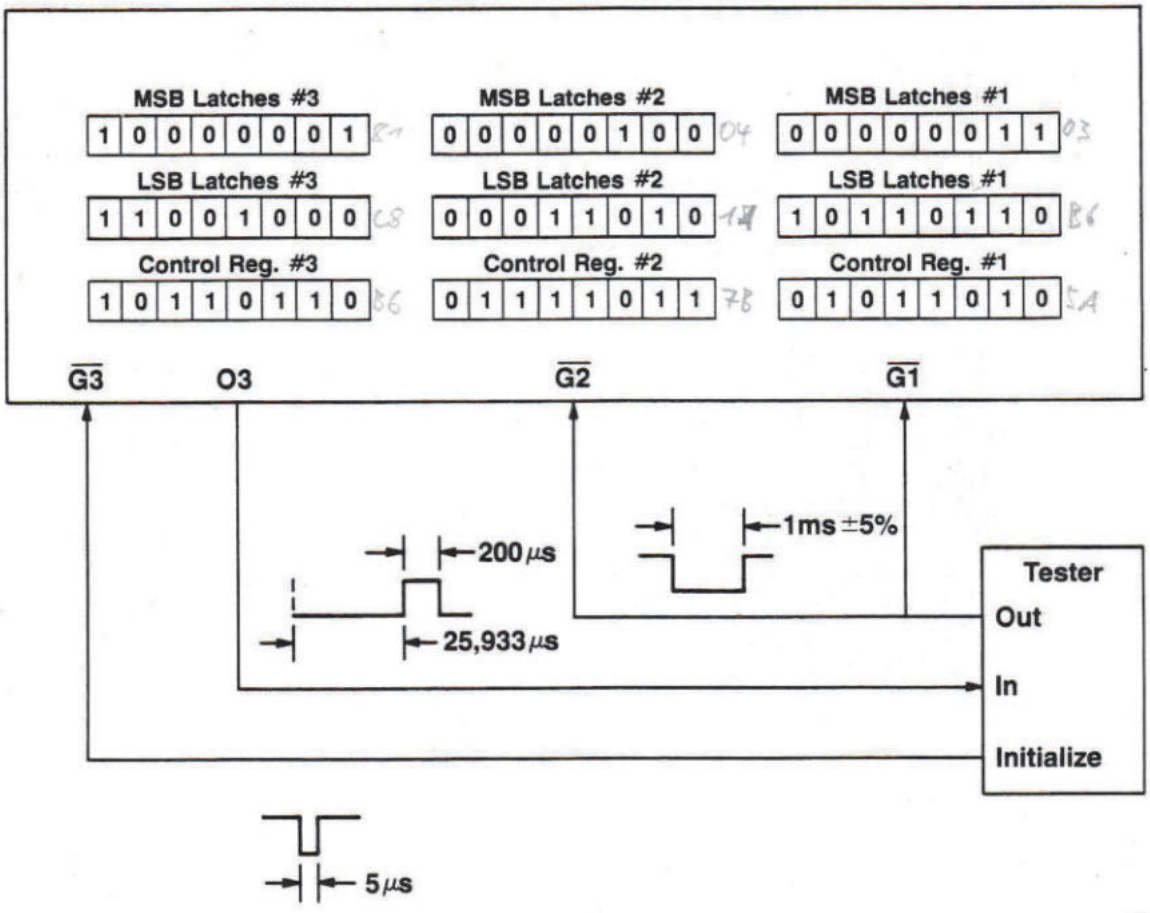
$(1000\ 0001\ 1100\ 1000_2)$.

2. Determine the binary number required in the latches to produce an interrupt if the measured pulse width is greater than 1 millisecond +5%. Using the Pulse Width Comparison Mode table, it is noted that a timer can be programmed to cause an interrupt if a T.O. occurs prior to an input transition on the Gate input (G↑). By setting the latches to the value 1050 μs (1 millisecond +5%) and generating the interrupt as described above, the interrupt can be used to identify a module which has a pulse width greater than 1 ms +5%. Timer #2 was selected for this task and the latches are programmed with 1050_{10} or $041A_{16}$ (0000 0100 001 1010_2).

3. Determine the binary number required in the latches to produce an interrupt if the measured pulse width is less than 1 millisecond −5%. Using the Pulse Width Comparison Mode table, it is noted that a timer can be programmed to cause an interrupt if a Gate input (G↑) occurs prior to the T.O. By setting the latches to a value of 950 μs (1 millisecond −5%) and generating an interrupt as described above, the interrupt can be used to identify a module which has a pulse width less than 1 ms −5%. Timer #1 was selected for this task and the latches are programmed with 950_{10} or $03B6_{16}$ (0000 0011 1011 0110_2).

4. Recognition of an external reset (RESET) will clear all Control Register bits except CR10 (internal reset bit). With CR20 cleared it is possible to write to Control Register #3. Determine the bit pattern required in Control Register #3 (CR3) since it will be the first Control Register that we should write to. Set CR30 = 0. Prescaling the clock (8) is not required. Set CR31 = 1. The 1 MHz Enable Clock will be used to time the output pulse. Set CR32 = 1, CR33 = 0, CR34 = 1, and CR35 = 1. This will set the delayed Single-Shot Mode of operation. Set CR36 = 0. Interrupts are not required at Time Out (T.O.). Set CR37 = 1. Unmask the output at 03.
5. Determine the bit pattern for Control Register #2 (CR2). Set CR20 = 1. After writing a 1 to CR20, Control Register #3 can be addressed. Set CR21 = 1. The 1 MHz Enable Clock will be used to measure the pulse width Set CR22 = 0, CR23 = 1, CR24 = 1, and CR25 = 1. This will set the Pulse Width Comparison Mode of operation with the interrupt flag set if a T.O. occurs prior to the Gate input (G2↑) transition. Set CR26 = 1. Allow interrupts to the microprocessor via the $\overline{IRQ}$ line. Set CR27 = 0. Output in Pulse Width Mode is undefined therefore Mask output at 02.
6. Determine bit pattern for Control Register #1. CR10 = 0, allow all timers to operate. CR11 = 1, use Enable Clock. CR12 = 0, CR13 = 1, CR14 = 1, CR15 = 0. This will set the Pulse Width Comparison Mode of operation with the interrupt flag set if a Gate input (G1↑) transition occurs prior to a T.O. CR16 = 1, allow interrupts to the microprocessor via the $\overline{IRQ}$ line. CR17 = 0, output not required.

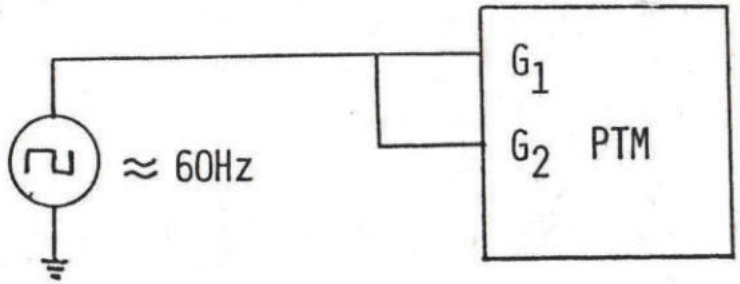
PTM PROGRAMMING SOLUTION

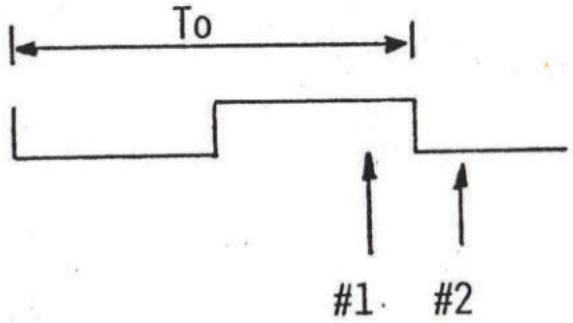


E5945

```
PAGE 001  TME2     .SA:0

 NAM TME2
*
*
*    SET UP TIMER #3 FOR A 200 MICROSECOND PULSE
*    OUTPUT.  DELAY BY 25,930 MICROSECONDS FROM THE
*    NEGATIVE TRANSITION AT THE GATE 3 INPUT
*    (3 MICROSECONDS USED TO SYNCHRONIZE AND PROCESS
*    THE EXTERNAL GATE INPUT TO THE TIMER.  TOTAL
*    DELAY = 25,933 MICROSECONDS).
*****************************************************
*
*
*    SET UP TIMER #2 FOR MEASUREMENT OF A 1050
*    MICROSECOND PULSE WIDTH --- IF GREATER CAUSE
*    AN INTERRUPT --- STATUS REGISTER BIT 1 ---
*****************************************************
*
*
*    SET UP TIMER #1 FOR MEASUREMENT OF A 950
*    MICROSECOND PULSE WIDTH --- IF LESS CAUSE
*    AN INTERRUPT --- STATUS REGISTER BIT 0 ---
*
*****************************************************
*
*
 OPT NOP
 ORG $4000
CR31 RMB 1 ADDRESS FOR CR1 & CR3
CRST RMB 1 ADDRESS FOR CR2 $ STATUS REG.
TIME1 RMB 2 ADDRESS FOR TIMER #1 LATCHES
TIME2 RMB 2 ADRESS FOR TIMER #2 LATCHES
TIME3 RMB 2 ADDRESS FOR TIMER #3 LATCHES
 ORG $2000
 LDX #$81C8
 STX TIME3 WRITE TO LATCHES #3.
 LDX #1050
 STX TIME2 WRITE LATCHES #2
 LDX #950
 STX TIME1 WRITE LATCHES #1
 LDAA #%10110110
 STAA CR31 WRITE CR#3
 LDAA #%01111011
 STAA CRST WRITE CR#2
 LDAA #%01011010
 STAA CR31 WRITE CR#1
 END
```

```
PAGE  001   TME2

00001                        NAM    TME2
00002                  *
00003                  *
00004                  *   SET UP TIMER #3 FOR A 200 MICROSECOND PULSE
00005                  *   OUTPUT.  DELAY BY 25,930 MICROSECONDS FROM THE
00006                  *   NEGATIVE TRANSITION AT THE GATE 3 INPUT
00007                  *   (3 MICROSECONDS USED TO SYNCHRONIZE AND PROCESS
00008                  *   THE EXTERNAL GATE INPUT TO THE TIMER.  TOTAL
00009                  *   DELAY = 25,933 MICROSECONDS).
00010                  ************************************************
00011                  *
00012                  *
00013                  *   SET UP TIMER #2 FOR MEASUREMENT OF A 1050
00014                  *   MICROSECOND PULSE WIDTH --- IF GREATER CAUSE
00015                  *   AN INTERRUPT --- STATUS REGISTER BIT 1 ---
00016                  ************************************************
00017                  *
00018                  *
00019                  *   SET UP TIMER #1 FOR MEASUREMENT OF A 950
00020                  *   MICROSECOND PULSE WIDTH --- IF LESS CAUSE
00021                  *   AN INTERRUPT --- STATUS REGISTER BIT 0 ---
00022                  *
00023                  ************************************************
00024                  *
00025                  *
00026                        OPT    NOP
00027 4000                     ORG    $4000
00028 4000 0001    CR31    RMB    1          ADDRESS FOR CR1 & CR3
00029 4001 0001    CRST    RMB    1          ADDRESS FOR CR2 $ STATUS REG.
00030 4002 0002    TIME1   RMB    2          ADDRESS FOR TIMER #1 LATCHES
00031 4004 0002    TIME2   RMB    2          ADRESS FOR TIMER #2 LATCHES
00032 4006 0002    TIME3   RMB    2          ADDRESS FOR TIMER #3 LATCHES
00033 2000                     ORG    $2000
00034 2000 CE 81C8             LDX    #$81C8
00035 2003 FF 4006             STX    TIME3      WRITE TO LATCHES #3.
00036 2006 CE 041A             LDX    #1050
00037 2009 FF 4004             STX    TIME2      WRITE LATCHES #2
00038 200C CE 03B6             LDX    #950
00039 200F FF 4002             STX    TIME1      WRITE LATCHES #1
00040 2012 86 B6               LDA A  #%10110110
00041 2014 B7 4000             STA A  CR31       WRITE CR#3
00042 2017 86 7B               LDA A  #%01111011
00043 2019 B7 4001             STA A  CRST       WRITE CR#2
00044 201C 86 5A               LDA A  #%01011010
00045 201E B7 4000             STA A  CR31       WRITE CR#1
00046       0000               END

TOTAL ERRORS 00000
```





$T_O = \frac{1}{60Hz} = 16.66 msec$

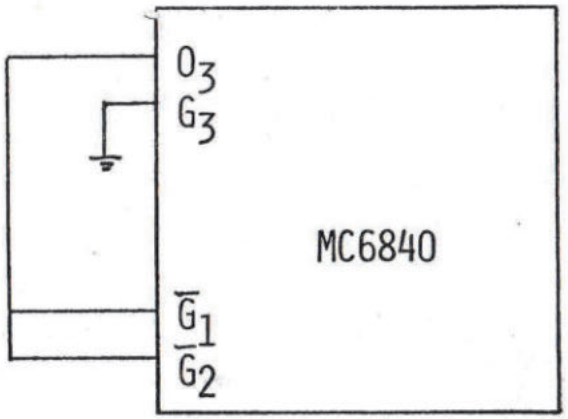
$F_H = 60Hz + 10\%(60Hz) = 66Hz \longrightarrow T = 15151\,\mu sec = \$3B2F$

$F_L = 60Hz - 10\%(60Hz) = 54Hz \longrightarrow T = 18518\,\mu sec = \$4856$

TIMER #1 BITS 543 = 101

TIMER #2 BITS 543 = 001

FREQUENCY COMPARISON MODE EXAMPLE

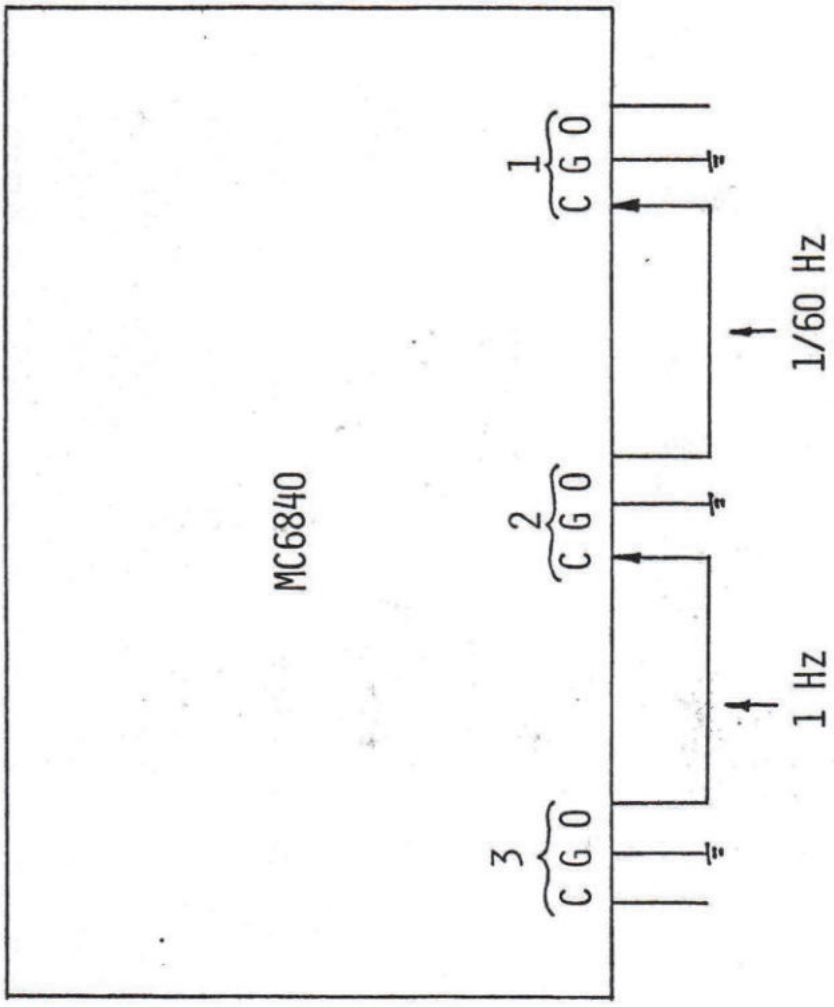


$4000	#3 CONTROL	$82
$4001	#2 CONTROL	$2B
$4000	#1 CONTROL	$0A
$4002	#1 LATCHES	$3B
$4003		$2F
$4004	#2 LATCHES	$48
$4005		$56
$4006	#3 LATCHES	$20
$4007		$8D

FREQUENCY COMPARISON MODE EXAMPLE

PTM

EXAMPLE FOR CONTINUOUS MODE



CONTINUOUS MODE EXAMPLE

```
PAGE  001   PTMCME

00001                          NAM     PTMCME
00002 0000 86 83               LDA A   #$83
00003 0002 B7 4000             STA A   $4000     *$83-->#3 TIMER
00004 0005 86 81               LDA A   #$81
00005 0007 B7 4001             STA A   $4001     *-->$81-->#2 TIMER
00006 000A 86 00               LDA A   #0
00007 000C B7 4000             STA A   $4000     *00-->#1 TIMER
00008 000F CE F423             LDX     #$F423
00009 0012 FF 4006             STX     $4006     *$F423-->#3 TIMER LATCHES
00010 0015 CE 001D             LDX     #$1D
00011 0018 FF 4004             STX     $4004     *$001D-->#2 TIMER LATCHES
00012       0000               END

TOTAL ERRORS 00000
```