Using the TMS370 Timer Modules

Michael S. Stewart
Microcontroller Applications Engineering
Semiconductor Group
Texas Instruments Incorporated
Contributions by Paul Krause

TEXAS INSTRUMENTS

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to or to discontinue any semiconductor product or service identified in this publication without notice. TI advises its customers to obtain the latest version of the relevant information to verify, before placing orders, that the information being relied upon is current.

TI warrants performance of its semiconductor products to current specifications in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Unless mandated by government requirements, specific testing of all parameters of each device is not necessarily performed.

TI assumes no liability for TI applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

Contents

Figures

Tables

Chapter 1

Introduction

The TMS370 family of 8-bit microcontrollers presently provides up to two timer modules designed to meet user demands for timer applications. The TMS370Cx5x and TMS370Cx10 devices contain the Timer 1 module, while the TMS370Cx5x devices contain an additional Timer 2 module.

This application report provides examples of software routines and hardware interface circuits designed to illustrate how the features of the timer modules may be used to solve a variety of system timer requirements. These concepts may be adapted and applied to fit the specific needs of your individual project. Additional information for the Timer 1 and Timer 2 modules may be found in the *TMS370 Family Data Manual* (SPNS014), Sections 7 and 8.

Table 1–1. TMS370 Family Timer Module Capabilities

System Requirements	Timer Resources
Real-Time System Control	Interval Timers with Interrupts
Input Pulse Width Measurement	Pulse Accumulate or Input Capture Functions
External Event Synchronization	Event Count Function
Timer Output Control	Compare Function
Pulse-Width Modulated Output Control	PWM Output Function
System Integrity	Watchdog Function

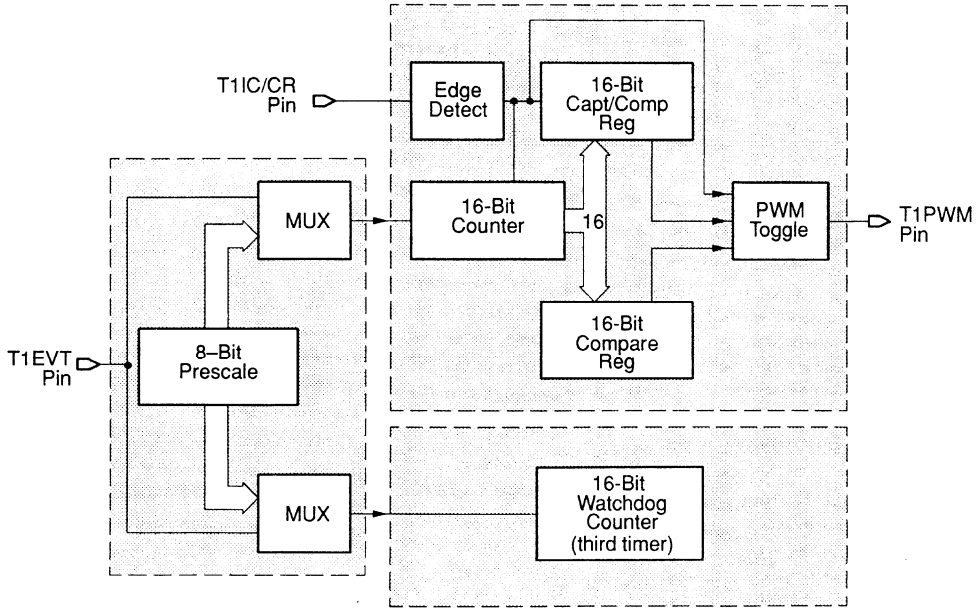
Chapter 2

Module Description

2.1 Timer 1

The Timer 1 module is available on all present TMS370 devices, and contains three major blocks as shown in Figure 2–1: an 8-bit prescaler/clock source block, a 16-bit general purpose timer (T1), and a 16-bit Watchdog timer (WD). Additional functions of the Timer 1 module not illustrated in Figure 2–1 include the Interrupts and I/O pins.

Figure 2–1. Timer Block Diagram



2.1.1 Prescaler/Clock Source

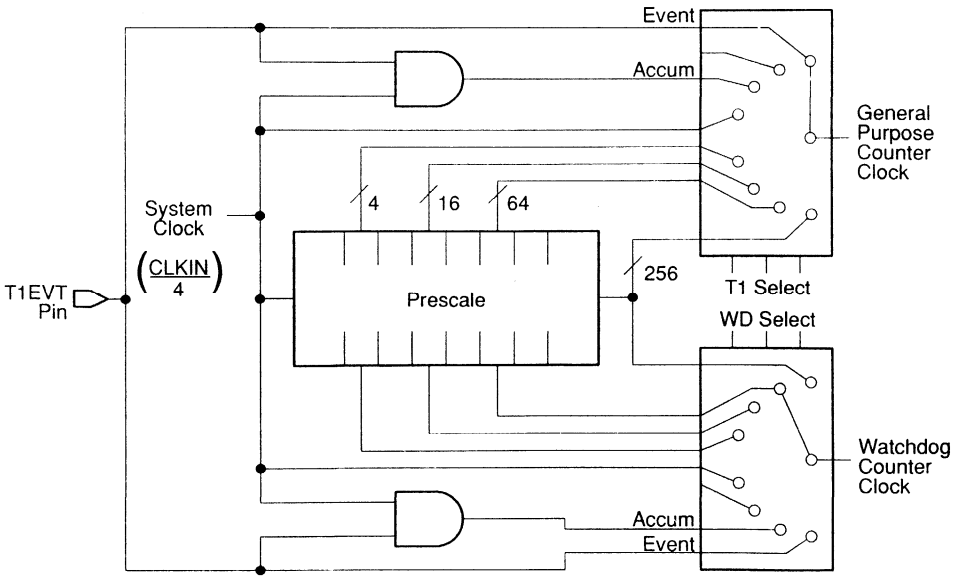
The Prescaler/Clock Source block provides eight available clock sources for the general purpose timer (T1) and the Watchdog timer (WD). (See Figure 2–2.)

These clock sources are:

- System Clock (CLKIN/4)
- Pulse Accumulation
- Event Input
- System Clock with /4 prescale tap
- System Clock with /16 prescale tap
- System Clock with /64 prescale tap
- System Clock with /256 prescale tap
- System Clock Off (Timer not running)

The clock sources may be independently selected for T1 and the WD Counter. For example, you could select the Event Input clock source for T1 while the WD Counter uses the System Clock with /64 prescale tap.

Figure 2–2. Timer 1 System Clock Prescaler



2.1.2 Timer 1 Counter

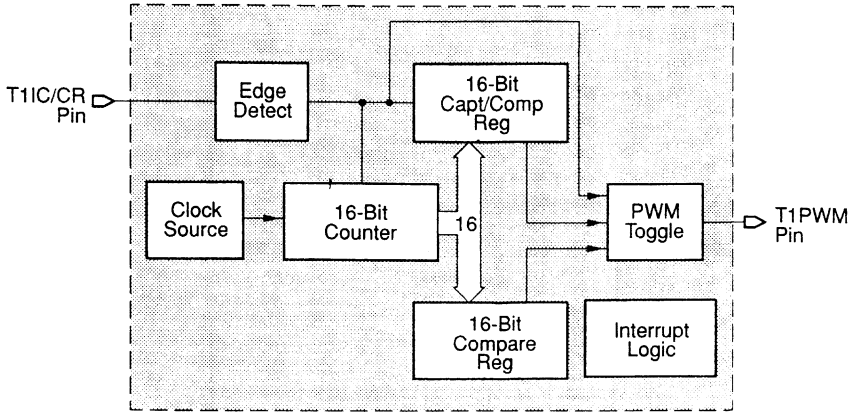
The T1 block (Figure 2–3) contains a 16-bit counter, a 16-bit Compare Register, a 16-bit Capture/Compare Register, and provides input capture, output compare and external event functions. T1 can be operated in either the Dual Compare Mode or the Capture/Compare Mode depending on the needs of your individual application.

The basic functions of the T1 block can be defined as follows:

- The input **Capture** function is used to latch the present value of the 16-bit counter register into the 16-bit Capture/Compare Register on the occurrence of a selected edge on the T1IC/CR pin. This function is available only when operating in the Capture/Compare Mode.
- The output **Compare** function is used to trigger an action (toggle the T1PWM pin for example) when the contents of a Compare Register equal the present value of the Counter Register.
- The external **Edge Detection** function is used to trigger an action (load the Capture Register for example) and occurs when an appropriate external edge is present on the T1IC/CR pin. This function can also toggle the T1PWM pin or reset the counter in the Dual Compare Mode.

For additional information concerning modes of operation or functionality of the T1 block, see Section 7.2 in the *TMS370 Family Data Manual.*

Figure 2–3. 16-Bit Programmable General Purpose Timer 1

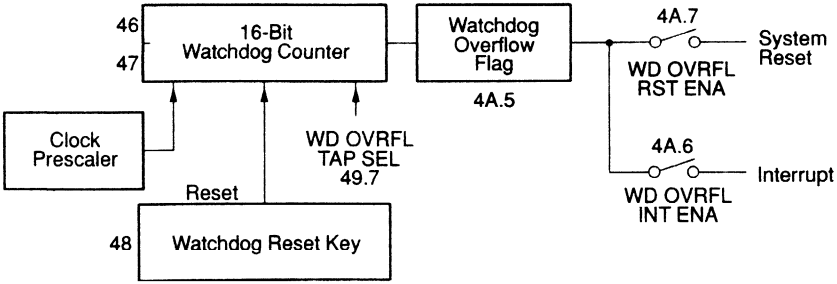


2.1.3 Watchdog Counter (WD)

The Watchdog (WD) Counter (Figure 2–4) is a separate 16-bit counter in the Timer 1 module. The WD Counter can be used to cause a System reset or can be software configured as a simple counter/timer, an event counter, or a pulse accumulator if the Watchdog reset feature is not needed. The time-out duration for the WD Counter depends on the Clock source selected and can be programmed with an overflow resolution ranging from 15 – 24 bits.

Additional information concerning the WD Counter is available in Section 7.3 of the *TMS370 Family Data Manual*.

Figure 2–4. Watchdog Timer



2.1.4 T1 Interrupts

The Timer 1 module provides up to five (5) different interrupt flags depending on the mode of operation. The actions that may trigger an interrupt are as follow:

- **External Edge Detection/Input Capture**: An active transition on the T1IC/CR pin will cause the T1EDGE INT FLAG bit (T1CTL3.7) to be set if the T1EDGE DET ENA bit (T1CTL4.0) is set. In the Dual Compare mode, this action can reset the T1 counter if the T1CR RST ENA bit (T1CTL4.1) is set, and also toggle the T1PWM pin if the T1CR OUT ENA bit (T1CTL4.3) is set.

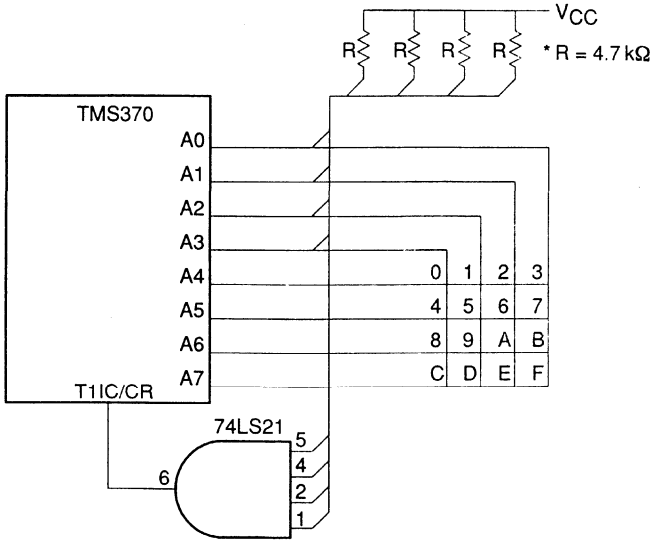
  In the Capture/Compare mode, the T1EDGE INT FLAG BIT (T1CTL3.7) will be set if enabled, and also, the contents of the T1 counter will be loaded into the Capture/Compare register if the T1EDGE DET ENA bit (T1CTL4.0) is enabled.

- **Compare Equal 1**: When the value of the Compare register matches the value of theT1 counter, the T1C1 INT FLAG bit (T1CTL3.5) will be set during both modes of operation. This action can also toggle the T1PWM pin if the T1C1 OUT ENA bit (T1CTL4.6) is enabled. The

T1PWM toggle function is true only for the Dual Compare mode of operation.

- **Compare Equal 2**: In the Dual Compare mode, the Capture/Compare register functions as an additional Compare register, and when the value of the Capture/Compare register matches the value of the T1 counter, the T1C2 INT FLAG bit (T1CTL3.6) will be set. This action can also toggle the T1PWM pin if the T1C2 OUT ENA bit (T1CTL4.5) is enabled. Please note that this function is only available in the Dual Compare mode of operation.
- **Counter Overflow**: The T1 counter has overflowed from 0FFFFh to 0000h. The T2 OVRFL INT FLAG bit (T1CTL2.3) will be set on this occurrence.
- **Watchdog Overflow**: The WD counter has overflowed and the WD OVRFL FLAG bit (T1CTL2.5) will be set. A system reset will occur if the WD OVRFL RST ENA bit (T1CTL2.7) is enabled. Also, an interrupt without system reset can occur in the WD OVRFL RST ENA bit is cleared, and the WD OVRFL INT ENA bit (T1CTL2.6) is set.

Figure 2–5. Keyboard Scan Using T1IC/CR as an External Interrupt



2.1.5 T1 I/O Pins

The Timer 1 module includes three I/O pins which can be dedicated for timer functions or as general purpose I/O pins. The configuration for these pins are controlled through the Timer Port control registers T1PC1 and T1PC2. Their names and Timer 1 functions are as follow:

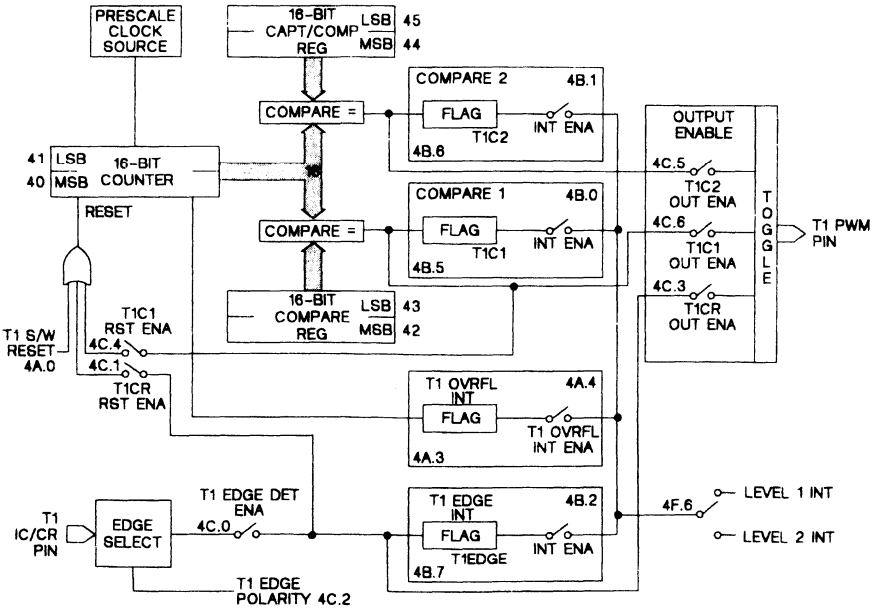
- **T1EVT**: This pin may be used as an external clock input to the Prescale/ Clock Source block. Input frequency may not exceed CLKIN/8.
- **T1IC/CR**: Depending on the mode of operation, this pin may be used to input an external signal to trigger loading of the Capture Register, toggle the T1PWM output pin, reset the counter or generate an interrupt.
- **T1PWM**: The Timer 1 function of this pin is to output a Pulse Width Modulated (PWM) signal from the module.

2.1.6 T1 Operational Modes

The Timer 1 module may be used in either of two modes of operation: Dual Compare mode, or Capture/Compare mode. (See Section 7.2 of the *TMS370 Family Data Manual* for additional information.)

- **Dual Compare Mode**: To operate in the Dual Compare mode, the T1 MODE bit (T1CTL4.7) must be cleared. This mode provides two compare registers (the Capture/Compare register is configured as a Compare register) which can be used to control the period and duty cycle of a PWM signal or for multiple other applications. A block diagram of T1 in the Dual Compare mode is shown in Figure 2–6.

Figure 2–6. Dual Compare Mode for Timer 1

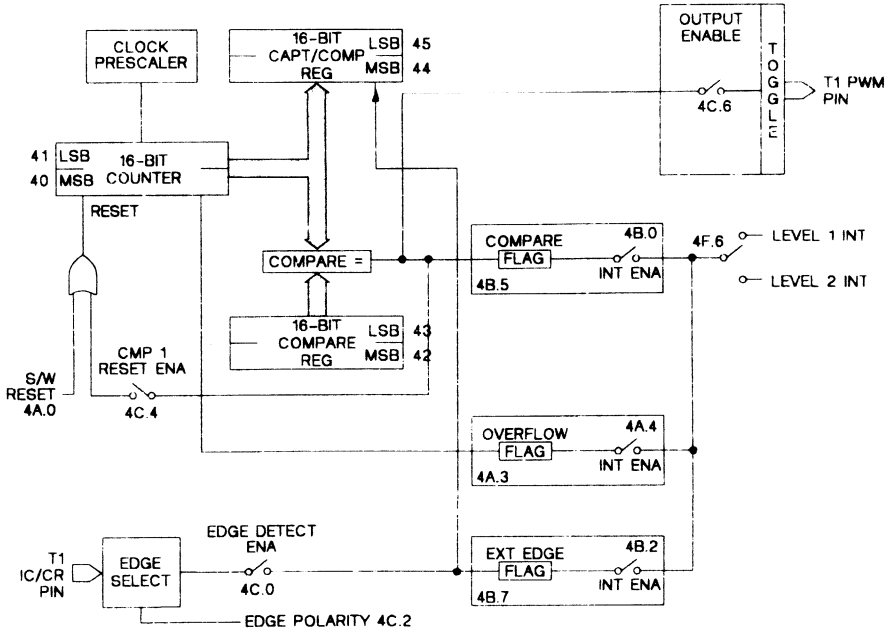


Note:

The numbers on the diagram, such as 4B.5, identify the register and the bit in the peripheral frame. For example, the actual address of 4B.5 is 104Bh, bit 5, in the T1CTL3 register.

- **Capture/Compare Mode**: To operate in the Capture/Compare mode, the T1 Mode bit (T1CTL4.7) must be set. This mode provides one compare register, and the Capture/Compare register is configured as a Capture register. The Compare register can be used to generate periodic interrupts or toggle the T1PWM pin and the Capture register can be used for pulse measurement. A block diagram of T1 in the Capture/Compare mode is shown in Figure 2–7.

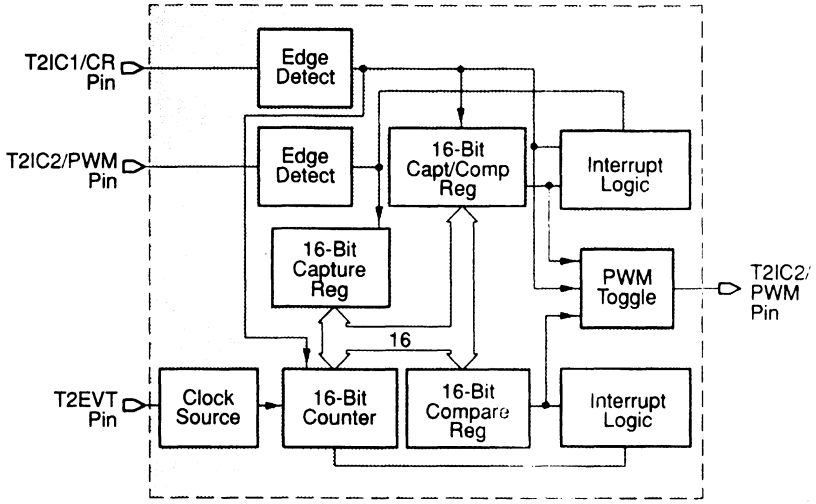
Figure 2–7. Capture/Compare Mode for Timer 1



2.2 Timer 2

The Timer 2 module (T2) is a 16-bit general-purpose timer available on the TMS370Cx5x devices and is illustrated in Figure 2–8. Timer 2 (T2) allows program selection of four input clock sources (System Clock, external event, pulse accumulate, or no clock). Additional blocks of the Timer 2 module not shown in Figure 2–8 include the interrupts and I/O pins.

Figure 2–8. 16-Bit Programmable General Purpose Timer 2



2.2.1 Timer 2 Counter

The T2 block (Figure 2–8) contains a 16-bit counter, a 16-bit Compare Register, and a 16-bit Capture/Compare Register just like T1. T2 also contains an additional Capture Register. T2 provides input capture, output compare, Timer Overflow, and external event functions. You can choose either the Dual Compare Mode or the Dual Capture Mode of operation for T2 depending on the needs of your application.

The basic functions of the T2 block are similar to those described for the T1 block (Section 2.1.2). The addition of an extra Capture Register and the lack of a prescale block are the main differences between T1 and T2. For additional information concerning modes of operation or functionality of the T2 block, see Section 8.2 in the *TMS370 Family Data Manual.*

2.2.2 T2 Interrupts

The Timer 2 module provides four (4) different interrupt flags. Depending on the mode of operation, these interrupt flags can be set by one of five (5) different sources. The actions that may trigger an interrupt are as follow:

- **Input Capture 1/External Edge Detection 1**: An active transition has occurred on the T2IC1/CR pin causing the T2EDGE1 INT FLAG bit (T2CTL2.7) to be set. If the T2EDGE1 DET bit (T2CTL3.0) is enabled, then this action will also load the contents of the T2 counter into the Capture/Compare register. Please note you must be in the Dual Capture mode of operation for the Capture function.
- **Input Capture 2/External Edge Detection 2**: An active transition has occurred on the T2IC2/PWM pin causing the T2EDGE2 INT FLAG bit (T2CTL2.6) to be set. If the T2EDGE2 DET bit (T2CTL3.1) is enabled, then this action will also load the contents of the T2 counter into the Capture register. Please note you must be in the Dual Capture mode of operation for these actions to occur.
- **Compare Equal 1**: When the value of the Compare register matches the value of the T2 counter, the T2C1 INT FLAG bit (T2CTL2.5) will be set. This is true for both modes of operation.
- **Compare Equal 2**: When the value of the Capture/Compare register matches the value of the T2 counter, the T2C2 INT FLAG bit (T2CTL2.6) will be set. This is true for the Dual Compare Mode of operation only.
- **Counter Overflow**: The T2 counter has overflowed from 0FFFFh to 0000h. The T2 OVRFL INT FLAG bit (T2CTL1.3) will be set on this occurrence.

2.2.3 T2 I/O Pins

The Timer 2 module includes three I/O pins which can be dedicated for timer functions or as general purpose I/O pins. Their names and Timer 2 functions are as follow:

- **T2EVT**: This pin may be used as an external clock input or Pulse Accumulation Signal to the T2 Module. Input frequency may not exceed CLKIN/8.
- **T2IC1/CR**: Depending on the mode of operation, this pin may be used to input an external signal to trigger loading of the Capture/Compare register or to toggle the T2PWM output pin. A signal on this pin may also reset the counter.
- **T2IC2/PWM**: In the Dual Compare mode, the function of this pin is to output a Pulse Width Modulated (PWM) signal from the module. In the Dual Capture mode, this pin may be used to input an external signal to trigger loading the Capture Register with the contents of the T2 Counter.

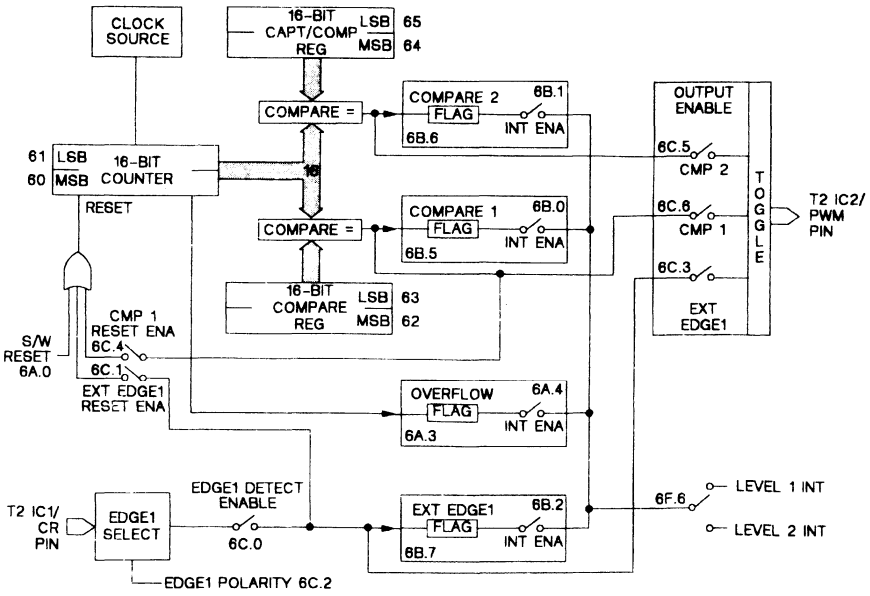
The configuration for these pins are controlled through the Timer Port control registers T2PC1 and T2PC2.

2.2.4 T2 Operational Modes

The Timer 2 module may be used in either of two modes of operation: Dual Compare mode, or the Dual Capture mode. (See Section 8.2 of the *TMS370 Family Data Manual* for additional information.)

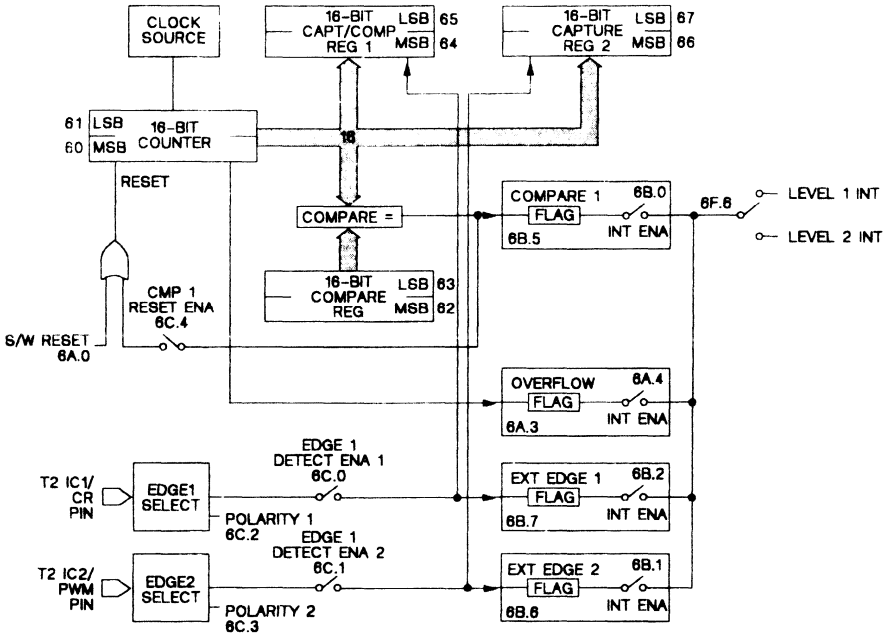
- **Dual Compare Mode**: To operate in the Dual Compare mode, the T2MODE bit (T2CTL3.7) must be cleared. This mode provides two compare registers (the Capture/Compare register is configured as a Compare register) which can be used to control the period and duty cycle of a PWM signal or for multiple other applications. The Dual Compare mode of T2 is identical in function to the Dual Compare mode of T1 with the exception of no optional prescale input for the clock source. A block diagram of T1 in the Dual Compare mode is shown in Figure 2–9.

Figure 2–9. Dual Compare Mode for Timer 2



- **Dual Capture Mode**: To operate in the Dual Capture mode, the T2MODE bit (T2CTL3.7) must be set. This mode provides two capture registers as well as one compare register. In this mode, the Capture/Compare register is configured as a Capture register. The two capture registers may be used for pulse width measurement and timing, and the Compare register can be used to generate periodic interrupts. A block diagram of T1 in the Capture/Compare mode is shown in Figure 2–10.

Figure 2–10. Dual Capture Mode for Timer 2



Note:

The numbers on the diagram, such as 6A.3, identify the register and the bit in the peripheral frame. For example, the actual address of 6A.3 is 106Ah, bit 3, in the T2CTL1 register.

2.3 Timer Formulas

The following formulas are used to calculate the Timer overflow, Watchdog overflow, and Compare Register values for the Timer 1 and Timer 2 modules. The formulas illustrated in this section deal with time periods. Therefore, the variable SYSCLK is used in the formulas. If it is easier to think in terms of frequency, then 4/CLKIN can be substituted for SYSCLK in the following formulas.

2.3.1 Timer 1: T1 and WD Counter Overflow

The maximum counter duration using the internal clock is determined by the internal system clock time (SYSCLK) and the prescale tap (PS). The Counter Overflow formula is shown below:

Maximum Counter Duration (seconds) = $2^{16} \times PS \times SYSCLK$

Counter Resolution = $PS \times SYSCLK$

where: SYSCLK = 4 / CLKIN (external crystal frequency)

PS = 1, 4, 16, 64, or 256 depending on the Prescale Tap selected

Table 2–1 gives the real-time counter overflow rates for various CLKIN and prescaler values. Please note that the value shown must be divided by two for the WD Counter if the WD OVRFL TAP SEL bit (T1CTL1.7) is set (see Section 7.3 in the *TMS370 Family Data Manual*).

Table 2–1. Timer 1 Module Counter Overflow Rates

Select 2	Select 1	Select 0	Divide By	Crystal Oscillator Frequency (MHz)			
				2.0	4.0	10	20
				System Clock Period (ns)			
				2000	1000	400	200
0	0	0	2^{16}	0.131*	0.066	0.026	0.013
0	0	1	(P.A.)	**	**	**	**
0	1	0	(Event)	**	**	**	**
0	1	1	(Stop)	**	**	**	**
1	0	0	2^{18}	0.524	0.262	0.105	0.052
1	0	1	2^{20}	2.10	1.05	0.419	0.210
1	1	0	2^{22}	8.39	4.19	1.68	0.839
1	1	1	2^{24}	33.6	16.8	6.71	3.355

* Time is given seconds.

** Not applicable.

2.3.2 Timer 1: Compare Register Formula

The Compare Register value required for a specific timing application can be calculated using the following formula:

$$\text{Compare Value} = \frac{t}{PS \times SYSCLK} - 1$$

where:

t = desired timer Compare period (seconds)
SYSCLK = 4 / CLKIN (external clock frequency)
PS = 1, 4, 16, 64, or 256 depending on the Prescale Tap selected

Table 2–2 provides some sample Compare Register values to achieve various desired timings using a 20-MHz Crystal.

Table 2–2. Timer 1 Compare Values: (CLKIN = 20 MHz)

Time		Prescale	T1 Compare Register Value (N)		% Error (See Note)
Seconds	mSeconds		Decimal	Hex	
0.0005	0.5	None	2499	009C3h	0.000
0.001	1	None	4999	01387h	0.000
0.002	2	None	9999	0270Fh	0.000
0.005	5	None	24999	061A7h	0.000
0.01	10	None	49999	0C34Fh	0.000
0.02	20	/4	24999	061A7h	0.000
0.05	50	/4	62499	0F423h	0.000
0.1	100	/16	31249	07A11h	0.000
0.2	200	/16	62499	0F423h	0.000
0.5	500	/64	39062	09896h	0.000
1.0	1000	/256	19530	04C4Ah	0.001
2.0	2000	/256	39061	09895h	0.001
3.0	3000	/256	58593	0E4E1h	0.001

Note: % Error induced by the Timer 1 Formula. This error margin will vary depending on the desired Timer Compare period and the minimum Timer resolution (PS × SYSCLK).

2.3.3 Timer 2: T2 Counter Overflow

The maximum counter duration using the internal clock is determined by the internal system clock time (SYSCLK). This relationship is shown below:

Maximum Counter Duration (seconds) = $2^{16} \times$ SYSCLK
Counter Resolution = SYSCLK

where:
SYSCLK = 4 / CLKIN (external crystal frequency)

Table 2–3 gives the real-time counter overflow rates for various CLKIN values.

Table 2–3. Timer 2 Module Counter Overflow Rates

CLKIN Frequency (MHz)	Timer Overflow Rate
20.0	13.11 ms
12.0	21.85 ms
8.0	32.77 ms
5.0	52.43 ms
3.579	73.23 ms
2.0	131.07 ms

2.3.4 Timer 2: Compare Register Formula

The Compare Register value required for a specific timing application can be calculated using the following formula:

$$\text{Compare Value} = \frac{t}{\text{SYSCLK}} - 1$$

where:

t = desired timer Compare period (Seconds)
SYSCLK = 4 / CLKIN (external clock frequency)

Table 2–4 provides some sample Compare Register values to achieve various desired timings.

Table 2–4. Timer 2 Compare Values: (CLKIN = 20 MHz)

Time		T2 Compare Register		% Error
Seconds	**mSeconds**	**Decimal**	**Hex**	**(See Note)**
0.0005	0.5	2499	009C3h	0.000
0.001	1	4999	01387h	0.000
0.002	2	9999	0270Fh	0.000
0.005	5	24999	061A7h	0.000
0.010	10	49999	0C34Fh	0.000
0.013	13	64999	0FDE7h	0.000

Note: % Error induced by the Timer 2 Formula. This error margin will vary depending on the desired Timer Compare period and the minimum Timer resolution (SYSCLK).

Chapter 3

Timer Application Software Routine Examples

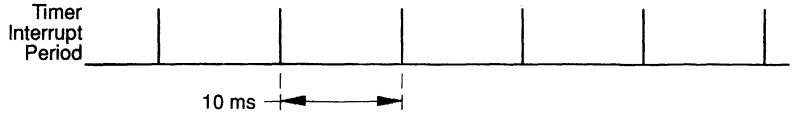
The following examples show various uses of the Timer Modules. Each example includes source code and a timing diagram. The examples shown attempt to illustrate typical timer application requirements. The Register Equate table for all the software examples is shown below. (See the Conclusion of this report to determine how to download copies of the software examples.)

Table 3–1. Common Equate Table

```
T1CNTRM    .EQU   P040    ;Timer 1 Counter MSB
T1CNTRL    .EQU   P04     ;Timer 1 Counter LSB
T1CM       .EQU   P042    ;Timer 1  Compare Register 1 MSB
T1CL       .EQU   P043    ;Timer 1  Compare Register 1 LSB
T1CCM      .EQU   P044    ;Timer 1  Capture 1/Compare 2 Register MSB
T1CCL      .EQU   P045    ;Timer 1  Capture 1/Compare 2 Register LSB
T1CTL1     .EQU   P049    ;Timer 1  control Register 1
T1CTL2     .EQU   P04A    ;Timer 1  control Register 2
T1CTL3     .EQU   P04B    ;Timer 1  control Register 3
T1CTL4     .EQU   P04C    ;Timer 1  control Register 4
1PC1       .EQU   P04D    ;Timer 1 Port Control 1
1PC2       .EQU   P04E    ;Timer 1  Port Control 2
T1PRI      .EQU   P04F    ;Timer 1  Priority control
T2CNTRM    .EQU   P060    ;Timer 2  Counter MSB
T2CNTRL    .EQU   P061    ;Timer 2  Counter LSB
T2CM       .EQU   P062    ;Timer 2  Compare Register 1 MSB
T2CL       .EQU   P063    ;Timer 2  Compare Register 1 LSB
T2CCM      .EQU   P064    ;Timer 2 Capture 1/Compare 2 Register MSB
T2CCL      .EQU   P065    ;Timer 2  Capture 1/Compare 2 Register LSB
T2ICM      .EQU   P066    ;Timer 2  Capture 2 Register MSB
T21C1      .EQU   P06B    ;Timer 2 control Register 2
T2CTL3     .EQU   P06C    ;Timer 2 control Register 3
T2PC1      .EQU   P06D    ;Timer 2 Port Control 1
T2PC2      .EQU   P06E    ;Timer 2 Port Control 2
T2PRI      .EQU   P06F    ;Timer 2 Priority control
```

3.1 Real-Time System Control: Periodic Interrupt of Timer 1

Interrupt the main program every 10ms (100 times a second).



This application routine provides a Timer 1 "Compare Equal" interrupt 100 times a second. This routine compares the present value of the 16-bit T1 counter to the value stored in the 16-bit Timer 1 Compare 1 Register. When these two registers are equal, an interrupt will occur and the T1 counter will be reset. The compare value to give 10 ms is as follows:

$$\begin{aligned} \text{time-needed} &= (4 \times (\text{compare} + 1) \times (\text{prescale})) \,/\, \text{CLKIN} \\ \text{compare} &= (\text{time-needed} \times \text{CLKIN})/\, 4 - 1 \\ \text{compare} &= (\, .010 \times 20 \times 10^6) \,/4 - 1 \\ \text{compare} &= 49999 \text{ or } C34Fh \end{aligned}$$

where:

$$\text{CLKIN} = 20 \text{ MHz}$$

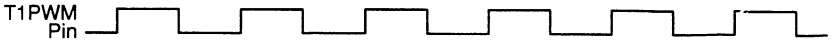
The program loads the value C34Fh into the Timer 1 Compare Register putting the MSB value in first. All output pins associated with Timer 1 are set to general purpose input pins since their Timer 1 pin functions are not needed for this application. The system clock is chosen as the Timer 1 clock source, while the watchdog prescale remains unchanged. The program then resets the counter, clears all interrupt flags and enables the Timer 1 Compare 1 interrupt. The timer is set to run in the Dual Compare mode but the Capture/Compare mode will work just as well in this example. The counter is initialized to reset whenever the Timer 1 Compare 1 Register equals the Counter Register so that the counter will be reset every 10 milliseconds. This routine will continue to interrupt the processor until the Global Interrupt or the Timer 1 Compare 1 Interrupt enable in T1CTL3 is disabled.

The 10 ms Timer Interrupt Routine follows:

```
T1INIT   MOV  #0C3h,T1CM          ;Value to give 10 ms with 20MHz crystal
                                  ; (C34F)
         MOV #04Fh,T1CL           ;Must load MSB first then LSB
         MOV #00000000b,T1PC1     ;T1EVT, T1PWM, AND T1IC/CR pins are set to
         MOV #00000000b,T1PC2     ; general purpose Input pins
         MOV #00000000b,T1PRI     ;Select Interrupt priority level 1.
         MOV #00010000b,T1CTL4    ;Select Dual Compare mode and cause Timer 1
                                  ; to reset on compare equal
         MOV #00000001b,T1CTL3    ;Clear any pending Interrupt flags, and allow
                                  ; the "Compare 1" flag to cause an interrupt
         AND #11110000b,T1CTL1    ;Select the System clock as timer clock
                                  ; source and leave the watchdog unchanged
         MOV #00000001b,T1CTL2    ;Reset the counter, (could enable WD here)
         EINT                     ;Begin interrupting main program
MAIN     ...                      ;Execute main program here
         ...
         --TIMER 1 INTERRUPT SERVICE ROUTINE--
T1INT                             ;Enter T1 int. service routine 100 times/sec
         MOV #00000001b,T1CTL3    ;Clear the T1C1 interrupt flag, reenable
                                  ; T1C1.
         ...                      ;Execute interrupt code
         ...
         RTI
```

3.2 Output Pulse Width Generation: 1-kHz Square Wave

Output a 1-kHz Square Wave (50% Duty Cycle).



This application routine generates a 1-kHz square wave output signal by using the 16-bit Timer 1 Compare Register to toggle the TIPWM output pin. Since the timer needs to toggle the output pin twice to produce one square wave pulse, the timer needs to toggle at a 2-kHz rate or every 0.5 ms. The compare value to give 0.5 ms is:

$$\begin{aligned} \text{time-needed} &= (4 \times (\text{compare} + 1) \times (\text{prescale})) / \text{CLKIN} \\ \text{compare} &= (\text{time-needed} \times \text{CLKIN})/ 4 - 1 \\ \text{compare} &= (0.0005 \times 20 \times 10^6)/ 4 - 1 \\ \text{compare} &= 2499 \text{ or } 09C3h \end{aligned}$$

where:

$$\text{CLKIN} = 20 \text{ MHz}$$

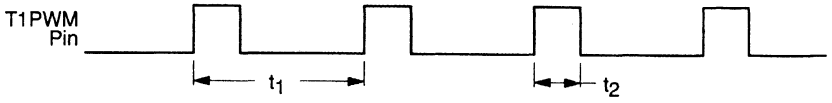
The program loads the value 09C3h into the Timer 1 Compare Register putting the MSB value in first. The T1PWM pin is set to the PWM output function and the other Timer 1 pins are set to general purpose input pins since their Timer 1 pin functions are not needed for this application. The system clock is chosen as the Timer 1 clock source, while the watchdog prescale remains unchanged. The program then resets the counter, clears all interrupt flags and disables all Timer 1 interrupts. The timer is set to run in Dual Compare mode but the Capture/Compare mode will work just as well in this example. The counter will reset whenever the Timer 1 Compare 1 Register equals the Counter Register so that the counter will reset every 0.5 ms. Once the Timer 1 module is initialized a 1-kHz square wave signal will be output continuously on the TIPWM pin without further program intervention.

The 50% Square Wave Signal Routine follows:

```
SQUARE  MOV #009h,T1CM          ;value to give .5 ms with 20MHz
                                ; crystal (9C3h)
        MOV #0C3h,T1CL          ;must load MSB first then LSB
        MOV #00000000b,T1PC1    ;T1EVT pin is set as a general
                                ; purpose input pin
        MOV #00100000b,T1PC2    ;Enable T1PWM pin (Initial output value
                                ; selected by bit 6), T1IC/CR is
                                ; general purpose input pin
        MOV #01010000b,T1CTL4   ;Select Dual Compare mode, enable PWM toggle,
                                ; and cause Timer 1 to reset on compare equal
        AND #11110000b,T1CTL1   ;Select the System clock as timer clock source
                                ;and leave the watchdog unchanged
        MOV #00000000b,T1CTL3   ;Clear and disable all interrupts
        MOV #00000001b,T1CTL2   ;Reset the counter, (could enable
                                ; watchdog here)
MAIN    ...                     ;Execute main program here
```

3.3 Pulse Width Modulation #1

Output a 1-kHz signal with a fixed 20% duty cycle.



In this example of Pulse Width Modulation, the pulse frequency remains 1 kHz while the duty cycle is 20%. The duty cycle is defined as the time the pulse remains high divided by the period of the pulse, so in this case the pulse remains high for 0.2 ms per cycle. The registers get configured like the square wave example in Section 3.2 but now the second compare register gets used to provide the high pulse period, t_2, while the first compare register is used to provide the 1-ms period, t_1. The program will load the value 1387h into the T1 Compare Register to control the 1-ms period (t_1) and 03E7h into the T1 Capture/Compare Register to control the t_2 pulse width. Both compare registers are enabled to toggle the output pin to give the proper pulse signal. Once the program starts the PWM signal, the signal will continue without any further program intervention.

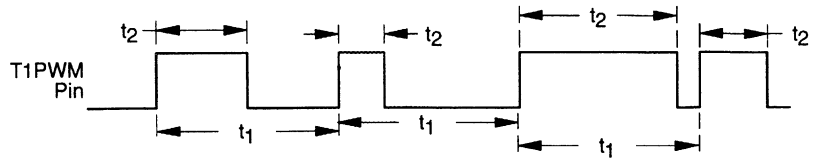
If the duty cycle or frequency needs changing once under way, modify only the Capture/Compare 2 Register or the Compare 1 Register, respectively (see Section 3.4).

The Pulse Width Modulation #1 routine follows:

```
PWM   MOV    #013h,T1CM          ; value to give 1 ms with
                                 ; 20 MHz crystal (1387h)
      MOV    #087h,T1CL          ;must load MSB first then LSB
      MOV    #003h,T1CCM         ;value to give .2 ms with
                                 ;20MHz crystal (3E7h)
      MOV    #0E7h,T1CCL         ;must load MSB first then LSB
      MOV    #00000000b,T1PC1    ;T1EVT pin is set as a general
                                 ; purpose input pin
      MOV    #01110000b,T1CTL4   ;Select Dual Compare mode, enable
                                 ; toggle function of Compare Registers
                                 ; 1 and 2, and cause Timer 1 to reset
                                 ; on Compare 1 equal
      AND    #11110000b,T1CTL1   ;Select the System clock as timer clock source
                                 ; and leave the watchdog unchanged
      MOV    #00000001b,T1CTL2   ;Reset the counter, (could enable
                                 ; watchdog here)
      MOV    #00000000b,T1CTL3   ;Clear and disable all interrupts
      MOV    #01100000b,T1PC2    ;Enable T1PWM pin (Initial output value
                                 ;selected by bit 6), T1IC/CR is general
                                 ; purpose input pin
MAIN  ...                        ;Execute main program here
      ...
```

3.4 Pulse Width Modulation #2

Output a 1-kHz signal with a varying duty cycle.



In this example of Pulse Width Modulation, a fixed frequency signal (1 kHz) will be output with a varying duty cycle. The main difference between this routine and the the previous routine (Pulse Width Modulation #1) is that the duty cycle (t_2) may vary. In this PWM example, the program changes the pulse width by altering the value in the Capture/Compare Register. The Compare Register controls the period of the signal (t_1) and is not changed in this routine, while the Capture/Compare Register controls the varying duty cycle (t_2) .

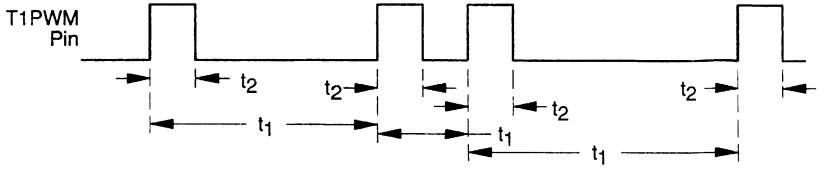
The Timer 1 Service Routine will be entered each time the Compare Register Equal Flag gets set (every 1 ms in this example). The main program is required to load any new values for the PWM duty cycle into the HIDC and LODC working registers. The Timer 1 Service Routine will only be enabled whenever the HIDC:LODC Register pair has been updated and the T1 Compare 1 Interrupt has been enabled (T1CTL3.0). The routine will stop the PWM signal, load the new values, and restart. Stopping the PWM signal will help avoid the possibility of inverting the signal if a larger value was written than previously existed (e.g., changing from a 20% to an 80% duty cycle signal.)

The Pulse Width Modulation #2 routine follows:

```
T1INIT  MOV    #013h,T1CM          ;Value to give 1 ms with 20 MHz.
                                   ; crystal (1387h)
        MOV    #087h,T1CL          ; must load MSB first then LSB
        MOV    HIDC,T1CCM          ;Load value for the Duty Cycle.
        MOV    LODC,T1CCL          ;must load MSB first then LSB
        MOV    #00000000b,T1PC1    ;T1EVT pin is set as a general
                                   ; purpose input.
        MOV    #00000000b,T1PRI    ;Set the T1 interrupt priority to level 1.
        MOV    #01110000b,T1CTL4   ;Select Dual Compare mode, enable toggle
                                   ; function of Compare Registers 1 and 2,
                                   ; and enable Timer 1 to reset on
                                   ; Compare 1 equal
        MOV    #00000000b,T1CTL1   ;Select the system clock as
                                   ; timer clock source.
        MOV    #00000001b,T1CTL2   ;Reset the counter, (could enable WD here)
        MOV    #00000000b,T1CTL3   ;Clear and disable all interrupts.
        MOV    #01100000b,T1PC2    ;Enable T1PWM (Initial output value
                                   ; selected by bit 6), T1IC/CR is
                                   ; general purpose input.
        EINT                       ;Enable interrupts.
MAIN    ...                        ;Execute main program here
        ...                        ;Any updates to the PWM Duty cycle registers
        ...                        ; (HIDC/LODC) will need to be done here.
UPDATE  MOV    #01h,T1CTL3         ;Allow the Compare flag to cause a timer
        ...                        ; interrupt only when the Duty Cycle
        ...                        ; (HIDC/LODC) Registers have been altered.
        ...
        ---Timer 1 Interrupt Routine to follow---
T1INT   MOV    #00000011b,T1CTL1   ;Stop T1 if an update has been made.
        MOV    #00000001b,T1CTL2   ;Reset the counter.
        MOV    #01010000b,T1PC2    ;Reset the T1PWM pin to general purpose
                                   ; output with the present value of the
                                   ; PWM pin.
        MOV    #01010000b,T1PC2    ;T1PWM pin will output a "1"
        MOV    #01100000b,T1PC2    ;Reenable the T1PWM function with an initial
                                   ;value of "1"
        MOV    HIDC,T1CCM          ;Load new value for the PWM duty cycle.
        MOV    LODC,T1CCL           ; Must load MSB first then LSB.
        MOV    #00h,T1CTL1         ;Reselect the System clock as the T1 clock
                                   ; source.
        MOV    #0000000b,T1CTL3    ;Clear the T1 Compare 1 interrupt flag and
                                   ;  disable the T1 Compare 1 flag again.
RETURN  RTI                        ;Return to the MAIN routine.
```

3.5 Pulse Position Modulation (PPM)

Output a fixed 0.2 ms pulse at a variable frequency (1-kHz rate initially).



In this example of Pulse Position Modulation, the high pulse width (t_2) remains constant while the period (t_1) of the pulses vary. The program code for this example is similar to the Pulse Width Modulation #2 example. In the PWM #2 example, the program changes the pulse width by varying the value in the Capture/Compare Register. In this Pulse Position Modulation example, the program varies the frequency of the pulses by changing the value in the compare register (T1CM, T1CL).

For the cleanest transition, clear the Compare 1 Equal Flag and wait until that flag gets set again before putting a new value into the Compare Register. This will help to avoid inverting the signal which could happen if a larger value was written than previously existed.

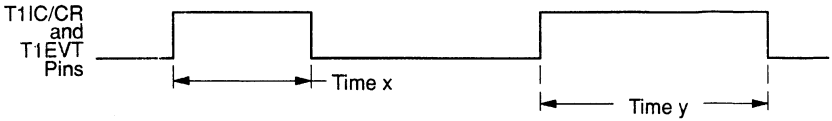
The Pulse Position Modulation Routine follows:

```
T1INIT  MOV   #013h,T1CM          ;Value to give 1 ms with 20 MHz.
                                  ; crystal (1387h)
        MOV   #087h,T1CL          ; must load MSB first then LSB
        MOV   #004h,T1CCM         ;Load value for the .2 ms Duty Cycle
        MOV   #0E1h,T1CCL         ; must load MSB first then LSB
        MOV   #00000000b, T1PC1   ;T1EVT pin is set as a general
                                  ; purpose input.
        MOV   #01100000b,T1PC2    ;Enable T1PWM (Initial output value
                                  ; selected by bit 6), T1IC/CR is
                                  ; general purpose input.
        MOV   #00000000b,T1PRI    ;Set the T1 interrupt priority to level 1.
        MOV   #01110000b,T1CTL4   ;Select Dual Compare mode, enable toggle
                                  ; function of Compare Registers 1 and 2,
                                  ; and enable Timer 1 to reset on
                                  ; Compare 1 equal
        MOV   #00000000b,T1CTL1   ;Select the system clock as
                                  ; timer clock source.
        MOV   #00000000b,T1CTL3   ;Clear and disable all interrupts.
        MOV   #00000001b,T1CTL2   ;Reset the counter, (could enable WD here)
        EINT                      ;Enable interrupts.

MAIN    ...                       ;Execute main program here
        ...                       ;Any updates to the PPM Frequency registers
        ...                       ;HIFREQ/LOFREQ will need to be done here.
UPDATE  MOV   #01h,T1CTL3         ;Allow the Compare flag to cause a timer
        ...                       ; interrupt only when the Duty Cycle
        ...                       ; (HIFREQ/LOFREQ) Registers have been altered.
        ...
T1INT   MOV   #00000000b,T1CTL3   ;Clear the T1 Compare 1 interrupt flag and
                                  ; disable the T1 Compare 1 flag again.
        MOV   #00000011b,T1CTL1   ;Stop T1 if an update has been made.
        MOV   #00000001b,T1CTL2   ;Reset the counter.
        MOV   #01010000b,T1PC2    ;Reset the T1PWM pin to general purpose
                                  ; output with the present value of the
                                  ; PWM pin.
        MOV   #01010000b,T1PC2    ;T1PWM pin will output a "1"
        MOV   #01100000b,T1PC2    ;Reenable the T1PWM function with an initial
                                  ; value of "1"
        MOV   HIFREQ,T1CM         ;Load new value for the PPM Frequency.
        MOV   LOFREQ,T1CL         ;Must load MSB first then LSB.
        MOV   #00h,T1CTL1         ;Reselect the System clock as the T1 clock
                                  ; source.
RETURN  RTI                       ;Return to the MAIN routine
```

3.6 Pulse Width Measurement Using Pulse Accumulation Clock Source

How long is the positive pulse of a signal?

Input connected to both T1IC/CR and T1EVT pins.



This example will measure the time that a single pulse remains high. The signal line connects to both the Input Capture (T1IC/CR) and the Event counter (T1EVT) inputs. Timer 1 will run in the Dual Compare mode and use the Pulse accumulate clock source which will allow the system clock to increment the counter as long as the T1EVT input pin remains high. The signal is also connected to the T1IC/CR counter reset pin to give the program an indication of when the external pulse goes low and ends the pulse accumulation function. The routine configures the Timer 1 pins first and then selects the Dual Compare mode of operation. The Interrupt flags are cleared and a falling edge on the T1IC/CR pin is enabled to cause an interrupt. The Pulse Accumulate clock source is chosen and the counter will then be reset.

The counter will not start until the pulse signal goes high, and will stop counting when the signal goes back low. The interrupt routine checks for the end of pulse or a counter overflow. If the interrupt is caused by an overflow, the counter increments the STOREOF Register, which is equivalent to the most significant byte of the timer register, then returns back to the main program. If the interrupt was caused by the pulse going low, then the routine reads the contents of the Timer 1 Counter Register, stores it into the STOREM:STOREL Register pair and returns to the main program. This example will be able to measure pulses up to approximately 3.36 seconds with the help of the STOREOF Overflow Storage Register. If longer pulses are required to be measured, additional Overflow Storage Registers could be used.

The Pulse Accumulation PWM measurement routine follows:

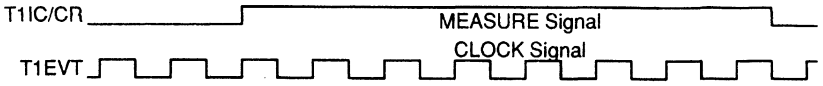
```
        CLR   STOREOF              ;Initialize the registers that will be used
        CLR   STOREM               ;  in this routine.
        CLR   STOREL
        CLR   BITS

TIMEPULSMOV   #00000010b,T1PC1     ;T1EVT and T1IC/CR pins enabled, T1PWM pin
        MOV   #0C000010b,T1PC2       ; is set up as general purpose input pin.
        MOV   #00,T1PRI            ;Select level 1 interrupts for Timer 1.
        MOV   #00000001b,T1CTL4    ;Select Dual Compare mode and watch for
                                   a falling edge on the T1IC/CR pin.
        MOV   #00000001b,T1CTL1    ;Select the Pulse Accumulate clock source.
        MOV   #00000100b,T1CTL3    ;Clear Interrupt flags, enable falling
                                   ;  edge on the T1IC/CR pin to cause an int.
        MOV   #00010001b,T1CTL2    ;Reset the counter, clear and enable the
                                   ;  the Overflow int. (WD could be init
                                   ;  here)
        EINT                       ;Enable interrupts.
        ...
MAIN    ...                        ;MAIN program here.
        ...
        -- TIMER 1 Interrupt routine to follow --
T1INT   MOV   #08h,T1CTL2,OVERFLW ;Was this interrupt caused by overflow?
                                   YES, jump to "OVERFLW"
        MOV   T1CNTRL,STOREL       ;NO, load the value in the T1CNTRL Registers
        MOV   T1CNTRM,STOREM       ;  (lsb first) into the "STORE" Registers.
        MOV   #00000100b,T1CTL3    ;Clear the T1IC/CR flag.
        MOV   #00000001b,T1CTL4    ;Reenable the T1IC/CR falling edge detect.
        OR    #00000001b,BITS      ;Signal to Main routine that pulse was read.
                                   ;  "BITS" reg. may be used by main routine.
        MOV   #00000001b,T1CTL2    ;Reset the counter
        RTI                        ;Return from interrupt.
OVERFLW AND   #11110111b,T1CTL2    ;Clear the overflow flag, then increment #
        INC   STOREOF              ;  of overflows( equivalent timer bits 16-23)
        RTI                        ;Return from interrupt.
```

3.7 Counting External Pulses Relative to an External Signal

How many external CLOCK pulses per MEASURE signal?

MEASURE signal attaches to the T1IC/CR pin.

CLOCK signal attaches to the T1EVT pin.



In this example two signals are input to the processor, a MEASURE signal and a CLOCK signal. The goal is to determine how many CLOCK signals happen during one high pulse of the MEASURE signal. The CLOCK signal connects to the T1EVT pin and the MEASURE signal connects to the T1IC/CR pin. The CLOCK signal will now increment the counter instead of the system clock as in the previous example. Because CLOCK continues to run after MEASURE goes low, the Timer module will run in the Capture/Compare Mode and use the 16-bit Capture/Compare Register to store the value of the counter the instant that MEASURE goes low.

One tricky condition can still occur where a counter overflow happens almost exactly when the MEASURE signal goes low so that both interrupt flags are set. The problem is then whether or not to increment the MSB counter register (STOREOF). If the Capture Register reads FFxxh, then the counter overflowed just after the MEASURE signal went low. If the register reads 00xxh, the counter overflowed just before the MEASURE pulse went low so the MSB counter register (STOREOF) should get incremented.

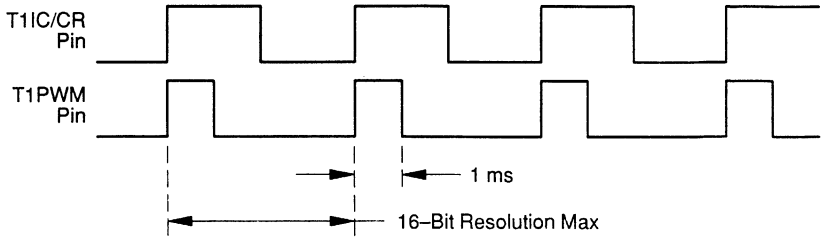
The External Pulse Counting routine follows:

```
        CLR     STOREOF         ;Clear registers used to store the sum of the
        CLR     STOREM          ;  T1EVT pulses.
        CLR     STOREL
T1INIT  MOV     #02h,T1PC1      ;T1EVT and T1IC/CR pins enabled, TIPWM pin set
        MOV     #02h,T1PC2      ;  to general purpose input pin.
        MOV     #40h,T1PRI      ;Select int. priority level 2 for Timer 1.
        MOV     #81h,T1CTL4     ;Select Capture/Compare Mode, enable the T1IC/CR
                                ;  pin to load the Capture Register on a falling
                                ;  pulse.
        MOV     #04h,T1CTL3     ;Clear the Interrupt flags and enable the T1IC/CR
                                ;  pin to cause an interrupt.
        MOV     #02h,T1CTL1     ;Choose Event input as clock source.
        MOV     #11h,T1CTL2     ;Reset the counter, clear and enable the Overflow
                                ;  Int.  (WD could be enabled here)
        EINT
        ...
        Interrupt Routine to follow.
T1INT                           ;INTERRUPT ROUTINE
        BTJZ    #08h,T1CTL2,PULSELO  ;Was interrupt caused by a low pulse?
        AND     #0F7h,T1CTL2    ;NO, clear overflow flag, then increment the
        INC     STOREOF         ; Overflow Register (STOREOF).
        RTI
PULSELO MOV     T1CCL,STOREL    ;YES, load the value in the Capture Register
        MOV     T1CCM,STOREM    ;  lsb first into the "STORE" Registers.
        BTJZ    #08h,T1CTL2,NOOVER   ;Was there an overflow just now?
        AND     #0F7h,T1CTL2    ;YES, clear the overflow flag.
        CMP     #0FFh,STOREM    ;if overflow and pulse low, which came first?
        JEQ     NOOVER          ;if FFxxh, Overflow happened after Pulse low
        INC     STOREOF         ;if 00xxh, Overflow happened first,inc register
NOOVER  MOV     #01h,T1CTL2     ;NO, reset the counter
        MOV     #04h,T1CTL3     ;Clear the interrupt flag, reenable edge int.
        MOV     #81h,T1CTL4     ;Reenable the T1IC/CR edge detect.
        OR      #1,BITS         ;Signal to the Main routine that pulse was read.
        RTI                     ;Return from interrupt.
```

3.8 Output Pulse Drive Referenced to Input Signal: TRIAC Controller or One Shot

Output a 1-ms pulse on every positive edge of an input signal.

(Input signal goes to IC/CR pin.)



In this example, a rising edge on the T1IC/CR input pin causes a 1 ms pulse to be output on the T1PWM pin. To give a simple application, this could be used in 60-Hz lamp dimmer or motor speed controller where the input is the 60-Hz signal and the output connects to the output driver. The timer is set up to clear the counter whenever the input pulse goes high and at the same time toggle the PWM pin. The counter then begins counting and whenever it equals the compare register the PWM pin toggles.The program then enters the interrupt service routine after the rising edge and resets the edge detection circuitry. This routine is the only program intervention needed to do this function. If the pulse length becomes greater than one overflow value plus 1 ms, the PWM will toggle and may corrupt the output. The overflow time for this value of a prescaler is about 54 ms. Change the prescaler to a higher value if a greater range is needed.

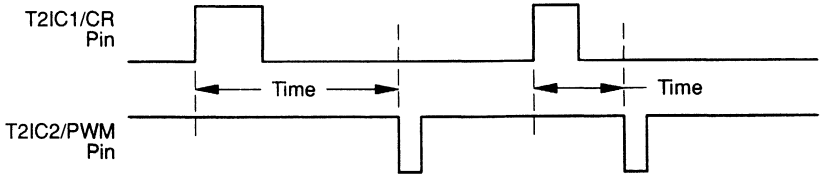
The One Shot routine follows:

```
                                ;single cycle should be under 1 timer overflow.
TRIAC   MOV     #04h,T1CM       ;value to give 1 ms with 20 MHz crystal (04E1h)
        MOV     #E1h,T1CL       ;must load MSB first then LSB
        MOV     #00h,T1PC1      ;T1EVT pin is set as a general purpose input
                                ; pin.
        MOV     #22h,T1PC2      ;Enable the T1PWM and T1IC/CR pins.
        MOV     #4Fh,T1CTL4     ;Select Dual Compare Mode, enable the Compare 1
                                ; Register and a rising edge on the T1IC/CR
                                ; pin to toggle the T1PWM pin, enable the
                                ; T1IC/CR pin to reset the counter.
        MOV     #74h,T1CTL1     ;Select the /4 prescale value and init the WD.
        MOV     #04h,T1CTL3     ;Clear interrupt flags, enable active edge on
                                ;  T1IC/CR to cause an interrupt.
        MOV     #01h,T1CTL2     ;Reset the counter, (could enable watchdog
                                ; here)
        EINT
        ...
T1INTERR
        MOV     #4Fh,T1CTL4     ;Re-enable T1IC/CR active edge interrupt.
        MOV     #04h,T1CTL3     ;clear T1IC/CR active edge interrupt.
        RTI
```

3.9 Pulse Width Measurement: Time Between Edges

What is the time between the rising edge on one signal and the falling edge of another signal?

Use Timer 2 in Dual Capture Mode.



How much time was there between the rising edge of one signal and the falling edge of another signal? This example uses the Timer 2 module with its dual capture registers to accurately give the answer to this problem. In this example the program configures Timer 2 in the Dual Capture Mode with the rising signal input into the T2IC1/CR pin and the falling signal input into the T2IC2/PWM pin. The port pins are configured to the correct value and the interrupts are set up to allow the correct edges to generate interrupts and store the counter value into the appropriate capture register.

The counter continually increments, overflows and generates an interrupt even though it has not detected the first rising edge. This is necessary because the counter may overflow immediately after the circuit detects a rising edge. The software could be too slow to react to this condition which is only a few microseconds wide, so the overflow interrupt remains active. When the circuit detects the rising edge of the first signal, the processor stores the Capture Register value into a register pair. The processor then keeps track of the overflows which happen about every 13.1 ms with a 20-MHz CLKIN signal, and waits for the falling edge of the second signal.

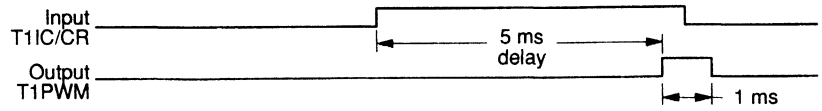
When it detects this falling edge, the first capture latch value is subtracted from this second capture latch value and overflows to give the time from one edge to the other. As in the External Pulse Counting example in Section 3.7, the program must consider the possibility of the falling edge coming at the same time as the counter overflow. By using the two capture latches, this program can handle the instances when the rising and falling edges happen very close together. Since an 8-bit register (TIME2OF) is used to keep track of Timer overflows, this application has a range of 24 bits. This example application can time edges as far apart as about 3.3 seconds, and could easily be increased by adding additional overflow registers.

The Edge Measurement routine follows:

```
EDGES   MOV     #02h,T2PC1      ;Set up T2EVT pin as general purpose input pin
        MOV     #22h,T2PC2      ;Enable T2IC1/CR and T2IC2/PWM pins
        MOV     #8Bh,T2CTL3     ;Select Dual Capture Mode, enable rising edge
                                ;  of T2IC1/CR and falling edge of T21C2/PWM ^
                                ;  to load the capture registers.
        MOV     #11h,T2CTL1     ;Reset counter, enable T2 overflow interrupt
        MOV     #06h,T2CTL2     ;Clear flags, enable T2IC1/CR and T2IC2/PWM
                                ;  edges to cause interrupts.
        EINT
        ...
                                ;T2 Interrupt Routine to follow
T2INTERR
        BTJO    #80h,T2CTL2,EDGE1       ;Jump on T2IC1/CR rising edge?
        BTJO    #40h,T2CTL2,EDGE2       ;Jump on T2IC2/PWM falling edge?
OVERFLOW INC    TIME2OF         ;Neither?  Increment the "TIME2OF" overflow
                                ;  storage register. (timer bits 16-23)
        AND     #F6h,T2CTL1     ;Clear overflow interrupt
        RTI
EDGE2   MOV     T2CAPTL,TIME2L ;Get 2nd capture latch value lsb and store it
        MOV     T2CAPTM,TIME2M ;Get 2nd capture latch value msb and store it
        BTJZ    #08h,T2CTL1,NOOVER      ;Was there an overflow just now?
        CMP     #0FFh,TIME2M    ;If overflow and pulse low, which came first?
        JEQ     NOOVER          ;If FFxxh, Overflow happened after Pulse low
        INC     TIME2OF         ;If 00xxh, Overflow happened first,inc
                                ;register
        AND     #F6h,T2CTL1     ;Clear overflow interrupts
NOOVER  MOV     #06h,T2CTL2     ;Clear edge 2 ints and enable edge 1,2 ints
        SUB     TIME1L,TIME2L   ;Get the difference between the two times
        SBB     TIME1M,TIME2M   ;Store the difference in TIME2
        SBB     #0,TIME2OF      ;Subtract any borrows from the overflows.
        RTI
EDGE1   AND     #62,T2CTL2      ;Disable T2IC1/CR interrupt until T2IC2/PWM
                                ;  edge occurs.
        CLR     TIME2OF         ;Reinitialize the "TIME2OF" Overflow Register.
        MOV     T2CCL,TIME1L    ;Get 1st capture latch value lsb and store it
        MOV     T2CCM,TIME1M    ;Get 1st capture latch value msb
        RTI
```

3.10 Output Pulse Generation (Delayed) Referenced to Input Signal

Output a 1-ms pulse 5 ms after the input signal goes high.



This program outputs a 1-ms pulse 5 ms after the input line goes high. This example uses Timer 1 in the Dual Compare Mode with the output toggle function of the T1IC/PWM pin. The program initializes the counter to look for the rising edge of the input signal on the T1IC/CR pin, and when it finds the edge, the program enters the interrupt service routine. The service routine checks to see if the interrupt was caused by the input rising or the output falling by checking the Compare 1 flag. If the input rising caused the interrupt the program quickly switches the clocking source from pulse accumulation to the system clock. If the input signal goes low before this switch is made then the output pulse will be slightly delayed. After it switches the clock source, the routine enables the PWM to toggle at the 5- and 6-ms points and also generates an interrupt when the Compare 1 Register toggles low at the 6 ms point. When the program enters the interrupt service routine again, it switches the clock back to the pulse accumulation mode and disables the PWM output toggling. The program then resets the timer to trigger only on the rising edge of the T1IC/CR input pin.

The Delayed Output Pulse Generation routine follows:

```
                                ;Put 6 ms into Compare 1 and 5 ms
                                ;into Compare 2
                                ;Input pulse must remain high at least 9 us.
                                ;Input = T1IC/CR   output =T1PWM
DELAY   MOV     #18h,T1CCM      ;value to give 5 ms with 20 MHz crystal (1869h)
        MOV     #69h,T1CCL      ;must load MSB first then LSB
        MOV     #1Dh,T1CM       ;value to give 6 ms with 20 MHz crystal (1D45h)
        MOV     #45h,T1CL       ;must load MSB first then LSB
        MOV     #00h,T1PC1      ;T1EVT pin is set up as general purpose input.
        MOV     #22h,T1PC2      ;Enable the T1IC/CR and T1IC/PWM pins, and
                                ;  initialize the T1IC/PWM pin to "0".
        MOV     #07h,T1CTL4     ;Select Dual Compare Mode, look for rising edge
                                ;  on T1IC/CR pin, and enable edge detection. ^
        MOV     #04h,T1CTL3     ;Clear interrupts, and enable T1IC/CR edge int.
        MOV     #71h,T1CTL1     ;Set-up watchdog, clock source=pulse
                                ; accumulator
        MOV     #01h,T1CTL2     ;Reset the counter, (could enable watchdog
                                ; here)
        EINT
        ...
MAIN    ...                     ;Main Routine can go here.
        ...
T1INTERR                        ;T1 Interrupt Routine to follow.
        BTJO    #20h,T1CTL3,ENDPUL   ;jump if at end of pulse (Compare 1
                                ;    flag=1)
        MOV     #70h,T1CTL1     ;counter now clocked by sys clock
        MOV     #64h,T1CTL4     ;enable PWM outputs, disable edge detect
        MOV     #01h,T1CTL3     ;clear flag, enable compare 1 to trigger at end
        RTI
ENDPUL  MOV     #71h,T1CTL1     ;counter now clocked by pulse accumulations
        MOV     #07h,T1CTL4     ;Re-enable edge interrupt, disable PWM output
        MOV     #04h,T1CTL3     ;clear flag, enable compare 1 to trigger at end
        RTI
```

Chapter 4

Watchdog Operation and Initialization

A Watchdog Timer (WD) operates as a Sentry to guard against improper program flow. Any time the WD timer is enabled to cause a system reset, and then overflows without being reset by a proper value being written to the WDRST register, a system Reset will occur. In other words, the program must write the proper values to the WDRST Key register (Pet the Watchdog) before the WD timer has a chance to time-out, or the WD timer will cause a system reset (Watchdog begins to bite!) This interaction between the program and the WD timer helps ensure program integrity. After the WD is enabled to reset the device, it can only be disabled by removing power from the part.

Watchdog Initialization Example

To initialize the Watchdog Timer to generate a system reset, do the following:

1) Select the appropriate clock source and Watchdog overflow Tap Select bits. (T1CTL1.4,5,6, and 7)

2) Clear the WD OVRFL INT FLAG bit (T1CTL2.5). This bit must be cleared in order to receive Watchdog generated Resets.

3) Set the WD OVRFL RST ENA bit (T1CTL2.7). Once this bit is set, only a Power-Up Reset can clear it. For this condition to occur, V_{CC} must fall to somewhere around 1 V (actual trip point depends on variables such as processing and temperature). The device will stop working before the WD OVRFL RST ENA bit gets cleared. Also, once this bit is set, the following Watchdog bits can not be altered until after a Power-Up Reset:

   a) WD OVRFL INT ENA (T1CTL2.6)

   b) WD OVRFL INT FLAG (T1CTL2.5)

   c) WD OVRFL TAP SEL (T1CTL1.7)

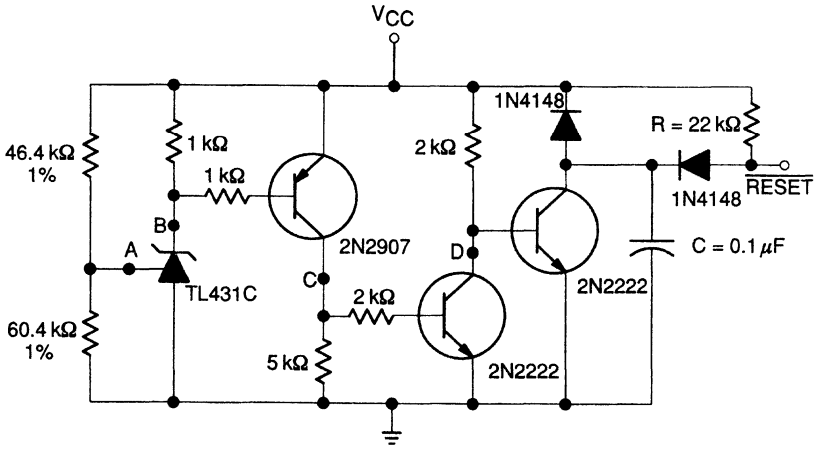
d) WD INPUT SELECT 0–2 (T1CTL1.4–6)

e) Write 55h to the Watchdog RESET key register (WDRST) to enable a proper reset sequence.

There are conditions where the program will fail to work properly due to low Vcc levels and the WD will not catch the failure. Your system should incorporate circuitry to cause a RESET when V_{CC} is out of spec. (See Figure 4–1.)

If a reset occurs, the RESET Subroutine will need to determine if the reset was caused by the WD timer or not by checking the WD OVRFL INT FLAG (T1CLT2.5). *If the reset was caused by the Watchdog, the WD OVRFL INT FLAG bit (T1CTL2.5) must be cleared in order to receive additional Watchdog resets.*

Figure 4–1. Typical Power Up/Down Circuit



The circuit illustrated in Figure 4–1 is a typical reset circuit that could be used to cause a system reset. This circuit will pull the $\overline{\text{RESET}}$ pin low almost immediately when V_{CC} falls below 4.5 V and will not allow $\overline{\text{RESET}}$ to go high until V_{CC} rises above 4.5 V. The basic operation of this circuit is centered around the TL431C precision voltage regulator. This regulator will not conduct if the trip point (Node A) is less than 2.5 V. As long as V_{CC} stays above approximately 4.5 V, the regulator will have greater than 2.5 V applied to Node A through the Ra/Rb divider network. When V_{CC} falls below 4.5 V, Node A will also fall below 2.5 V. The operation of the reset circuit is described below:

1) No reset desired: $V_{CC} > 4.5$ V.
As long as V_{CC} is greater than 4.5 V, Node A will stay above 2.5 V. This

causes the TL431C to be enabled and Node B will be about 2 V. The 2N2907 transistor will then be turned on, causing Node C to be pulled high. This will cause the base of the first 2N2222 transistor to be turned on, and Node D will be grounded. As long as Node D is low, the second 2N2222 transistor will be turned off, and the $\overline{\text{RESET}}$ pin will be held high.

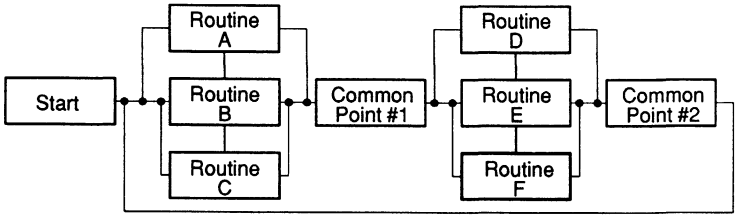
2) Reset desired: $V_{CC} < 4.5$ V.
When V_{CC} is less than 4.5 V, Node A will fall below 2.5 V. This causes the TL431C to be disabled and Node B will be pulled high. The 2N2907 transistor will then be turned off, causing Node C to be grounded. This will cause the base of the first 2N2222 transistor to be grounded, and Node D will be pulled high. As long as Node D is high, the second 2N2222 transistor will be turned on, and provide a ground path for the $\overline{\text{RESET}}$ pin.

The RC network of 22 kΩ and 0.1 μF provides a power-up rise time. Depending on the rise time of the power supply you are using, this may not be necessary. The two diodes (1N4148) are needed since the Reset of the TMS370 device can also be pulled low internally by the Watchdog counter as well as other circuits. The diodes are used to prevent the capacitor from discharging directly to the $\overline{\text{RESET}}$ pin when it is pulled low internally. Also the diode connected from the capacitor to V_{CC} provides an additional path for the capacitor to discharge when V_{CC} goes low suddenly.

4.1 Watchdog Reset Enable Initialization #1

This example can be used for those programs which always pass periodically through two or more points (see Figure 4–2) in the main program routine (not interrupt service routines). In this example, the main program resets the watchdog at those points by writing immediate values directly to the Watchdog reset register.

Figure 4–2. Two-Point Routine Operation



The Watchdog Overflow rate will depend on the worst case time through the Routines A, D, and C as well as D, E, and F. In this example, the Watchdog counter is set to 16 bits in length and the full 8-bit prescale tap is used. If a reset occurs, the RESET Subroutine will need to determine if the reset was caused by the WD timer or not by checking the WD OVRFL INT FLAG (T1CTL2.5).

```
INITWD   MOV  #00h,P048     ;Reset the watchdog while in the GP timer mode
         MOV  #70h,P049     ;Select prescale according to program needs
         MOV  #88h,P04A     ;Lock the watchdog in the watchdog reset mode.

MAIN1    ...
         MOV  #55h,P048     ;must write a 55 first, and on odd writes
                            ; (1,3,5,..)
         ...

MAIN2    ...
         MOV  #0AAh,P048    ;Must write an AA second, and on even writes
                            ;   (2,4,6,.)
         ...

                            ;Was the RESET caused by the Watchdog timer or not?
                            ;The following routine can be used to find out.

RESET    BTJZ #20h,P04A,GPINIT ;Is the WD flag set? If NOT
                            ; go to GPINIT.
WDINIT   AND  #DFh,P04A     ;Clear the WD flag.
         MOV  #55h,P048     ;Reset the WD counter.
         ...
         Do any Initialization here you desire
         ...
         RTS
         ...
GPINIT   Power-up reset routine goes here.
         ...
         RTS
```

4.1.1 Watchdog Reset Enable Initialization #2

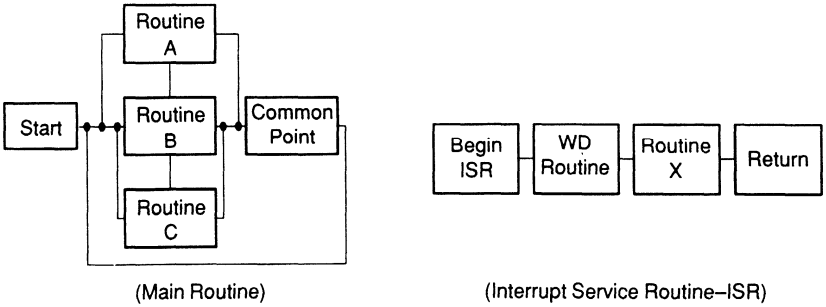
This example can be used for those programs which have many paths through the main program, but also contain a periodic interrupt service routine – ISR (see Figure 4–3). Since a program could get lost in a continuous loop in either the main or interrupt routine, the WD routine should not be entirely contained in either one. For example, a program could get caught in a loop in the main or interrupt routines. Even though the program is not executing properly, if the WDRST key register was being written to correctly in the loop, the WD timer would not cause a reset. Therefore, it is best if you have two separate actions in your code that must operate properly for the WD timer to NOT cause a system reset. If either one fails, a system reset will occur.

In this WD example, two separate actions are required for the WD routine to NOT cause a system reset:

1) The main program must clear a counter register (R4) before an interrupt routine occurs a set number of times (30 in this example). If the counter register is not cleared, the interrupt service will write an invalid data byte to the WDRST key register which will cause a system reset.

2) A periodic interrupt routine must be entered before the WD timer completely times out or a system reset will occur. Also, each time the interrupt routine is entered, the counter register (R4) will be incremented once and compared to a set value (30 in this example). If the counter is ever incremented past 30, the interrupt routine write an invalid data byte to the WDRST key register to cause a system register. Note that the only reason the counter register should every get past 30 is if the main routine does not clear it.

Figure 4–3. 1 Point Main Routine + Interrupt Operation



The Watchdog counter is set to 16 bits in length and no prescale tap is used. If a reset occurs, the RESET Subroutine will need to determine if the reset was caused by the WD timer or not by checking the WD OVRFL INT FLAG (T1CTL2.5).

```
WDCOUNT  EQU  R4
WDSTORE  EQU  R5

         ; The following routine detects if the RESET was caused by WD or not.

RESET    BTJZ #20h,P04A,GPINIT;Is the WD flag set? If NOT go to GPINIT.
WDINIT   AND  #0DFh,P04A      ;Clear the WD flag.
         ...
         Do any Initialization here you desire specific to the WD.
         ...
GPINIT                        Power-up reset routine goes here.

         MOV  #00h,P048       ;Reset the watchdog while in the GP timer mode
         MOV  #00h,P049       ;Select prescale according to program needs
         MOV  #80h,P04A       ;Lock the watchdog in the watchdog reset mode.
                              ;Set up the register values used in the
                              ;following
         CLR  WDCOUNT         ;routine. R4 used as a counter, R5 used as the
         MOV  #0AAh,WDSTORE   ;storage register for the next write to WDRST.

MAIN     ...
         CLR  WDCOUNT         ;Clear the register before Int routine increments
         ...                  ;  it past the value 30.  The reg. can be cleared
                              ;  at several point in a program if necessary.

INTERR                        ;Interrupt routine.
         INC  WDCOUNT         ;Inc. the 'Counter' reg. each Interrupt routine.
         CMP  #30,WDCOUNT     ;Has the 'counter' reg been incremented to 30?
         JL   PETDOG          ;No, jump to PETDOG.  Yes, write an invalid value
         MOV  #00,P048        ;  to WDRST. (This will cause a system reset)

PETDOG   INV  WDSTORE         ;Everything OK, invert old value (AA to 55, or 55
         MOV  WDSTORE,P048    ;  to AA) then "Pet" the watchdog to keep it
                              ;  happy
         ...
         RTI
```

4.2 Watchdog Initialization When System Reset Is Not Desired

If a program does not use the Watchdog reset circuit, any erroneously enabled watchdog can generate a reset. If the program also clears the Watchdog Overflow Int Flag, then the watchdog reset can continue to occur until a powerdown.

If a program does not use the Watchdog circuit, then take the following actions to avoid the continuous reset condition.

1) Assure the $\overline{\text{RESET}}$ pin is low during power up and oscillator start up.

2) Write x011xxxxb to T1CTL1 (P049) to halt clocking to the watchdog circuit.

3) Do not clear or write a zero to the Watchdog Overflow Int Flag (P04A.5). Consider the read-modify-write actions of the AND and XOR instructions and use them with care at this address.

Chapter 5

Specific Applications

5.1 Stepper Motor Control

This application routine uses the Timer 1 Compare Register to generate an interrupt which will be used to drive a stepper motor through the following range of applications:

1) Start stepping the motor at a desired minimum speed of approximately 92 rpm.

2) Accelerate the motor to a desired maximum speed of approximately 1378 rpm.

3) Decelerate the motor back to the minimum speed.

4) Change the motor rotation direction and repeat from step 1.

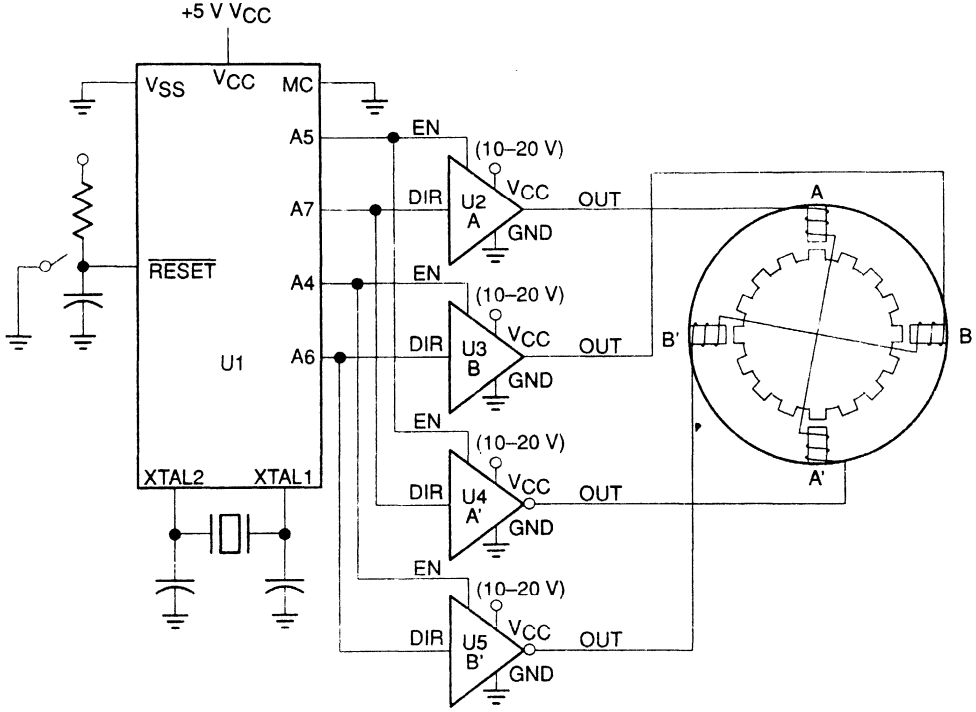
Acceleration, deceleration, and change of direction are controlled by checking bits in the FLAG Register. Bit 7 of FLAG is checked to determine the desired direction of rotation, while bit 0 is tested to see if the speed of rotation should be accelerated or decelerated. If bit 0 is a 1, then the speed needs to be decreased, and conversely if bit 0 is a 0, the speed needs to be increased.

The change of speed is accomplished by altering the value of the MSCOMP and LSCOMP working registers. Since the MSCOMP:LSCOMP Register pair is continually loaded into the Timer 1 Compare Register during the algorithm, any changes to the these registers between writes to the Compare Register will cause the Compare Equal interrupt period to change. If the value of the MSCOMP:LSCOMP Register pair decreases, the Timer 1 interrupt period will decrease, and the motor will step faster. If the value of the MCOMP:LSCOMP Register pair increases, the Timer 1 interrupt period will increase, and the motor will step slower. Change of direction is accomplished at the minimum desired speed, and is completed by altering bit 7 in the FLAG Register.

The hardware circuitry required for this application includes any TMS370 microcontroller with the Timer 1 module, two (2) SN75603 and two (2) SN75604 driver chips, and the stepper motor. The SN75603/4 driver chips are power peripherals with three-state outputs having the capability to sink or source currents up to 2 A. (Other drive chips may be used in this application.) The stepper motor used in this application is configured with 4 stator poles and 25 permanent magnet rotor poles. One hundred steps are required to complete one revolution of the rotor, each step being 3.6 degrees. V_{CC} for the driver chips depends on the stepper motor which is rated at 1 A at 20 V.

The schematic for the application is shown in Figure 5–1.

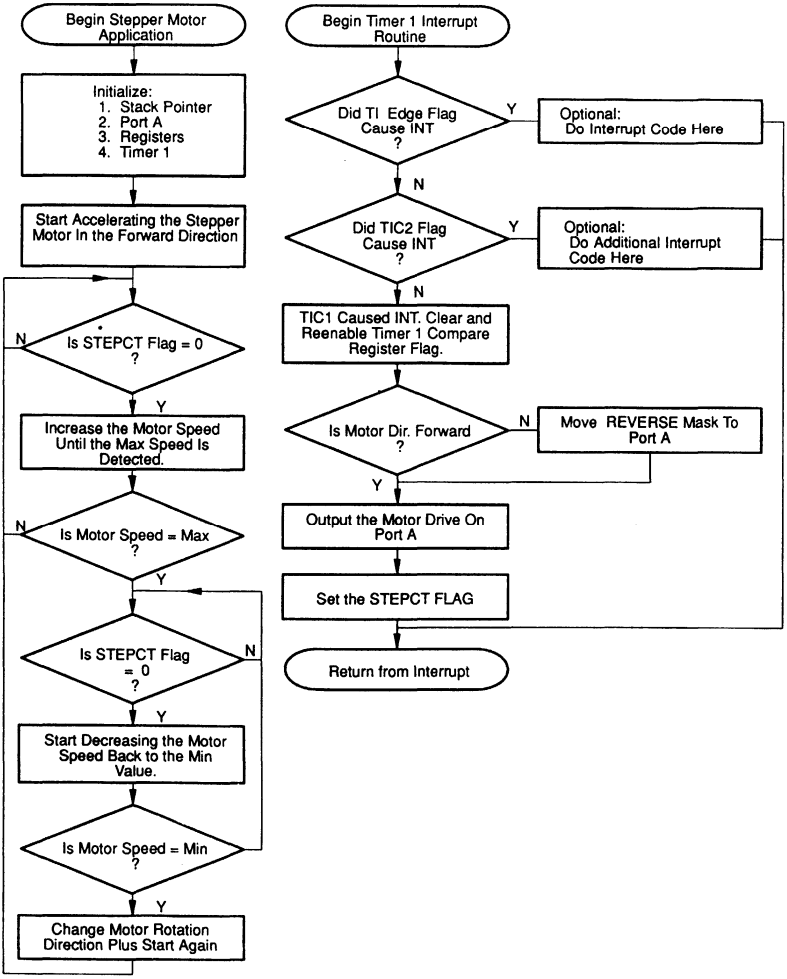
Figure 5–1. Stepper Motor Drive Application Schematic



U1 = TMS370 Family Microcontroller
U2, U3 = SN75603 Peripheral Drivers
U4, U5 = SN75604 Peripheral Drivers

The flowchart for the stepper motor application is shown in Figure 5–2.

Figure 5–2. Stepper Motor Control Application Flowchart



The source code for the Stepper Motor application is as follows:

```
        .title   "STEPPER MOTOR CONTROL"

;       Allocate register space for the four (4) registers used in the routine.

MSCOMP  .equ  R5              ;Used as working registers for new values for
LSCOMP  .equ  R6              ; the Timer 1 Compare Register.
FLAG    .equ  R7              ;Register used to tell if acceleration or
                              ;  deceleration routine is to be used (bit 0),
                              ;  and what direction to operate. (bit 7)
STEPCT  .equ  R8              ;Used to signal a complete write cycle to the
                              ;  4 motor poles. (Write cycle rev counter.)

;       Set up "EQUATE" table for Peripheral File Registers used in routine.

APORT2  .equ  P021            ;Port A Control Register.
ADATA   .equ  P022            ;Port A Data Register.
ADIR    .equ  P023            ;Port A Data Direction Register.
T1CM    .equ  P042            ;Timer 1 Compare Register 1 (MSB)
T1CL    .equ  P043            ;Timer 1 Compare Register 1 (LSB)
T1PC1   .equ  P04D            ;Timer 1 Port Control Register 1
T1PC2   .equ  P04E            ;Timer 1 Port Control Register 2
T1CTL1  .equ  P049            ;Timer 1 Control Register 1
T1CTL2  .equ  P04A            ;Timer 1 Control Register 2
T1CTL3  .equ  P04B            ;Timer 1 Control Register 3
T1CTL4  .equ  P04C            ;Timer 1 Control Register 4
T1PRI   .equ  P04F            ;Timer 1 Priority Control Register.

        .text 7000h

;       Begin Initialization:
;
;             0  Set up Stack Pointer to begin at R10.
;             0  Use MS nibble of Port A as the Stepper Motor drive port.
;             0  Initialize registers to their "Start" values.
;             0  Initialize Timer 1 operation.

START   MOV #10,B             ;Initialize the Stack Pointer to begin at
        LDSP                  ;  Register 10.
        MOV #00h,APORT2       ;Set up Port A MS nibble to be used as the
                              ;  4 pole stepper motor drive port.
        MOV #00h,ADATA        ;Initialize data = "00".
        MOV #0F0h,ADIR        ;Direction: A7=A6=A5=A4=OUT, A3=A2=A1=A0=IN
                              ;REGISTERS:
        MOV #080h,MSCOMP      ;  MSCOMP = "80h" MSCOMP&gml.LSCOMP = 08000h)
        MOV #000h,LSCOMP      ;  LSCOMP = "00h"
        CLR STEPCT            ;  STEPCT = "0"
        CLR FLAG              ;  FLAG   = "0"
        MOV #04h,B            ;  B      = "4", (OPTIONAL) Used only to count
                              ;  complete write cycles to the 4 motor poles.
                              ;  Could add additional code in the T1 int.
                              ;  service routine to count revolutions.
```

```
;       Initialize the Timer 1 module.

INTPGM  MOV #080h,T1CM          ;Value to give minimum speed (rpm) using a
        MOV #00h,T1CL           ;  20 MHz crystal.  Must load the MS byte first
                                ;  then the LS byte.
        MOV #00000000b,T1PC1    ;T1EVT, T1PWM, AND T1IC/CR pins are set to
        MOV #00000000b,T1PC2    ;  general purpose Input pins
        MOV #00010000b,T1CTL4   ;Select Dual Compare Mode and cause Timer 1
                                ;  to reset on compare equal
        MOV #01110000b,T1CTL1   ;Select the system clock as timer clock source
                                ;  and leave the watchdog unchanged
        MOV #00000111b,T1CTL3   ;Clear any pending Interrupt flags, and allow
                                ;  the Compare 1, Compare 2, or T1EDGE int
                                ;  flag to cause the Timer 1 interrupt.
                                ;  (OPTIONAL) Only Compare 1 int is required.
        MOV #00000001b,T1CTL2   ;Reset the counter, (could enable watchdog
                                ;  here)
        MOV #00000000b,T1PRI    ;Set the Timer 1 interrupt priority to level 1.

        EINT                    ;Allow interrupts to the main program

;       Begin main program:  Accelerate and decelerate the stepper motor by
;       changing the value in the Timer 1 Compare Register.  Also, change
;       direction when the minimum speed has occurred.

FASTER  BTJZ #01,STEPCT,FASTER  ;Execute Acceleration program here.
        CLR  STEPCT             ;Clear the STEPCT rev counter register.
        INCW #-80h,LSCOMP       ;Decrease the STORE register pair by 80h
        BTJO #0F7h,MSCOMP,UPDATE ;Has the maximum desired speed been reached?
                                ;  (True when (MSCOMP:LSCOMP) = 0880h)
                                ;  No, Update the Timer 1 Compare Register.
        INC  FLAG               ;  Yes, Set the "ACCEL/DECEL" bit in "FLAG".
UPDATE  MOV  MSCOMP,T1CM        ;Update the Timer 1 Compare Register with
        MOV  LSCOMP,T1CL        ;  the values in the MSCOMP and LSCOMP regs.
        BTJO #01h,FLAG,SLOWER   ;If ACCEL/DECEL bit is set jump to SLOWER,
        JMP  FASTER             ;  if not, jump to SPEEDUP.
SLOWER  BTJZ #01,STEPCT,SLOWER  ;Execute Deceleration program here.
        CLR  STEPCT             ;Clear the STEPCT rev counter register.
        ADD  #80h,LSCOMP        ;Increase the STORE register pair by 80h.
        ADC  #00,MSCOMP
        BTJZ #80h,MSCOMP,UPDATE ;Has the minimum desired speed been reached?
                                ;  (True when (MSCOMP:LSCOMP) = 08000h)
                                ;  No, Update the Timer 1 Compare Register.
CLRFLG  XOR  #81h,FLAG          ;  Yes, Clear the "ACCEL/DECEL" bit and change
                                ;  the "DIRECTION" bit.
        JMP  UPDATE             ;Update the Timer 1 Compare Register.
```

```
;       Timer 1 Interrupt service routine:  Routine will first check to see
;       which of three possible flags caused the interrupt, and jump to the
;       correct routine.  If the T1C1 flag (Compare Register 1) is set, the
;       "STPMTR" routine will be entered.  This routine will load the Motor
;       Pole drivers with a value that will cause the motor to accelerate or
;       decelerate in either the forward or reverse direction, depending on
;       the values of the "ACCEL/DECEL" and "DIRECTION" bits in the "FLAG"
;       Register.

T1INT   BTJO #80h,T1CTL3,EDGE   ;Check to see if the T1 EDGE flag caused int.
        BTJO #40h,T1CTL3,CAPCMP ; No, Check the T1C2 flag.
                                ; No, must have been the T1C1 flag.
STPMTR  MOV #11000111b,T1CTL3   ;Clear the T1C1 int. flag, reenable all ints.
                                ;Execute interrupt code
        BTJZ #80h,FLAG,FORWRD   ;Is "DIRECTION" bit clear?  Yes, then jump
                                ;  to the "FORWRD" routine.  No, continue.
        MOV  REV-1(B),A         ;Move the appropriate "Motor Pole" mask into
        JMP  LOAD               ;  the Port A Data Register (Reverse dir.).
FORWRD  MOV  DRIVE-1(B),A       ;Move the appropriate "Motor Pole" mask into
LOAD    MOV  A,ADATA            ;  the Port A Data Register (Forward dir.).
        DJNZ B,FINIS            ;(OPTIONAL) Decrement the "Cycle" Register
        MOV  #04,B              ;  count and reload with "4" if zero.
SETREV  INC  STEPCT             ;Set the STEPCT rev counter register.
FINIS   RTI                     ;Return to the main program


EDGE
        MOV #01100111b,T1CTL3   ;Clear the T1IC/CR Int.flag, reenable all ints.
                                ;Execute interrupt code
;       An Interrupt routine for a valid signal on the T1IC/CR pin can go here.
        RTI                     ;Return to main program.

CAPCMP
        MOV #10100111b,T1CTL3   ;Clear the T1C2 Int. flag, reenable all ints.
                                ;Execute interrupt code
;       An Interrupt routine for a Capture/Compare Register Equal can go here.
        RTI                     ;Return to main program.

        .data 7E00h
DRIVE   .byte 00010000b         ;A=B=0, A'=B'=1, Only B and B' poles enabled.
        .byte 00100000b         ;A=B=0, A'=B'=1, Only A and A' poles enabled.
        .byte 11010000b         ;A=B=1, A'=B'=0, Only B and B' poles enabled.
        .byte 11100000b         ;A=B=1, A'=B'=0, Only B and B' poles enabled.
REV     .byte 11100000b         ;A=B=1, A'=B'=0, Only B and B' poles enabled.
        .byte 11010000b         ;A=B=1, A'=B'=0, Only B and B' poles enabled.
        .byte 00100000b         ;A=B=0, A'=B'=1, Only A and A' poles enabled.
        .byte 00010000b         ;A=B=0, A'=B'=1, Only B and B' poles enabled.

        .sect "VECTOR",7FF4h
        .word T1INT             ;Location for the Timer 1 interrupt routine.
        .word START             ;  All other interrupt vectors point to
        .word START             ;  the "RESET" vector.
        .word START
        .word START
        .word START
        .end
```

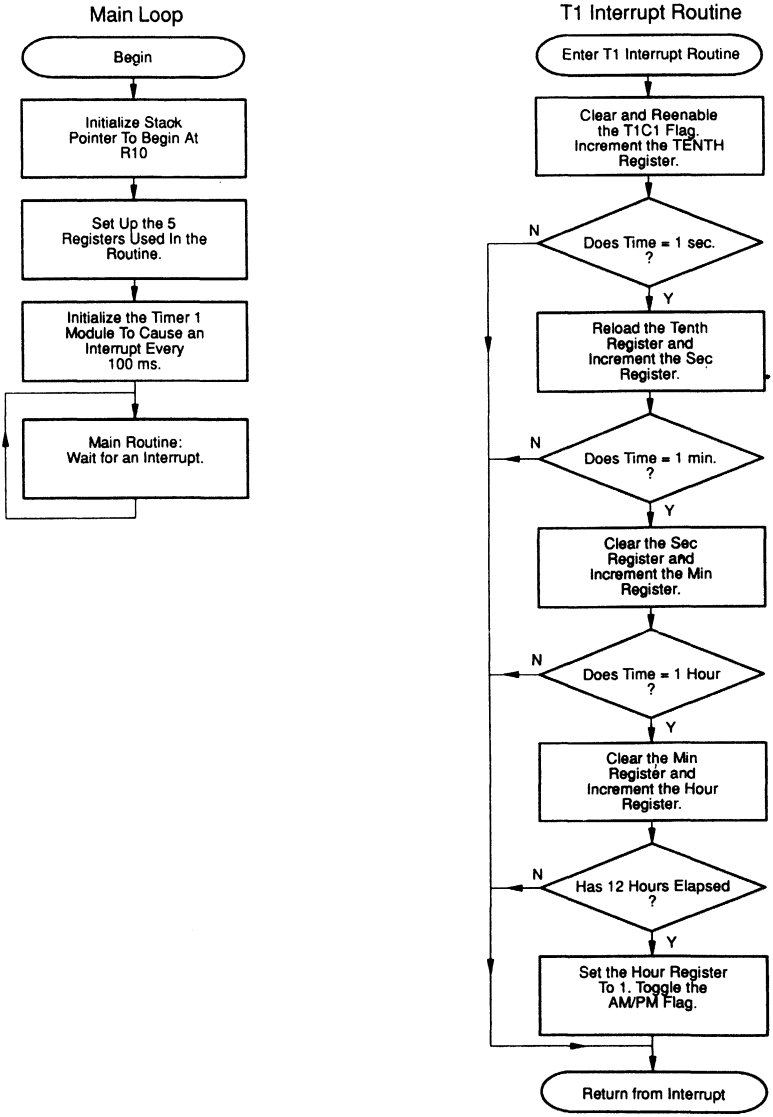
5.2 Time-of-Day Clock Application Routine

This application routine uses the Timer 1 Compare Register to generate an interrupt service routine every 1/10 second (100 ms) which will be used to update a Time-of-Day clock. The value required by the Compare Register to generate a 100-ms interrupt period with a 20-mHz crystal is 07A11h. (See page 2–10 for formula and look-up table.)

The application software will use five (5) registers to keep track of hours, minutes, seconds, tenths of seconds, and an AM/PM mode flag. Additional code and circuitry may be added for external time setting control and calendar application requirements. See Section 5.2.1,

The flowchart for the application is shown in Figure 5–3.

Figure 5–3. Flowchart for Time-of-Day Clock Application



The source code for the Time-of-Day application is as follows:

```
        .title    "Time-Of-Day Clock"

;       This routine will use Timer 1 in the Dual Compare Mode to implement
;       a real-time 12 hour clock (with AM/PM flag) down to 1/10's of seconds.

;       Allocate register space for the five (5) registers used in the
;       application routine.

AMPM    .equ  R5            ;AM/PM "Flag" Register
HOUR    .equ  R6            ;"HOUR" Register
MIN     .equ  R7            ;"MIN" Register
SEC     .equ  R8            ;"SEC" Register
TENTH   .equ  R9            ;Register used to count 10 - 1/10 second
                            ;  T1 Compare 1 interrupts.  (Required only
                            ;  to increase accuracy of clock)

;       Set up "EQUATE" table for Peripheral File Registers which will be used
;       in the routine.

T1CM    .EQU  P042          ;T1 Compare 1 Register (MSB)
T1CL    .EQU  P043          ;T1 Compare 1 Register (LSB)
T1PC1   .EQU  P04D          ;T1 Port Control Register 1
T1PC2   .EQU  P04E          ;T1 Port Control Register 2
T1CTL1  .EQU  P049          ;T1 Control Register 1
T1CTL2  .EQU  P04A          ;T1 Control Register 2
T1CTL3  .EQU  P04B          ;T1 Control Register 3
T1CTL4  .EQU  P04C          ;T1 Control Register 4
T1PRI   .EQU  P04F          ;T1 Interrupt Priority Register


        .text 7000h

;       Begin initialization:
;
;             0  Set up Stack Pointer to begin at R10.
;             0  Initialize Registers to their "START" value (12:00 A.M.).
;             0  Initialize the Timer 1 operation.

BEGIN   MOV   #10,B         ;Initialize the Stack Pointer to begin at
        LDSP                ;  Register 10.
```

```
;       Initialize the Clock registers to 12:00 a.m.

        CLR   SEC             ;Init "SEC" Register to "00".
        CLR   MIN             ;Init "MIN" Register to "00".
        MOV   #12h,HOUR       ;Init "HOUR" Register to "12".
        MOV   #00,AMPM        ;Init "AMPM". 0 = AM, 1 = PM
        MOV   #0Ah,TENTH      ;Init "TENTH" Register with "10"
        MOV   #00,T1PRI       ;Set T1 Priority for Level 1.
        MOV   #7Ah,T1CM       ;Move "07A11h" into the T1 Compare 1
        MOV   #11h,T1CL       ;  Register MSB first.
        MOV   #00h,T1PC1      ;Init all T1EVT, T1PWM, and T1IC/CR
        MOV   #00h,T1PC2      ;  to general purpose inputs.
        MOV   #10h,T1CTL4     ;Select Dual Compare Register Mode and
                              ;  allow Compare 1 reg. to reset timer.
        MOV   #05h,T1CTL1     ;Choose the /16 prescale tap for Timer 1.
        MOV   #01h,T1CTL3     ;Clear flags, enable only the T1C1 flag
                              ;  to cause an interrupt. (Other timer
                              ;  flags may be enabled if desired)
        MOV   #01h,T1CTL2     ;Disable Watchdog, reset Timer 1.
        EINT                  ;Allow interrupts to the main program.

;       Begin your main program here.  (The jump loop shown is for
;       demonstrational purposes only)

MAIN    JMP   MAIN

;       Timer 1 Compare 1 Interrupt Service Routine to follow.

T1INT   MOV   #01h,T1CTL3     ;Clear the Compare 1 flag.
        DJNZ  TENTH,END       ;Check to see if a second has gone by, if
        MOV   #0Ah,TENTH      ;  not, "RTI", if so, continue routine.
        DAC   #01,SEC         ;Add a decimal "1" to "SEC" then see if
        CMP   #060h,SEC       ;  60 seconds have elapsed yet.
        JNE   END             ;  If not, Return to main program.
        CLR   SEC             ;  If so, clear "SEC" then,
        DAC   #01,MIN         ;Add a decimal "1" to "MIN".  See if
        CMP   #060h,MIN         ;  60 Minutes have elapsed yet.
        JNE   END             ;  If not, Return to main program.
        CLR   MIN             ;  If so, clear "MIN" then,
        DAC   #01,HOUR        ;Add a decimal "1" to "HOUR".  See if
        CMP   #013h,HOUR        ;  13 hours have elapsed yet.
        JNE   END             ;  If not, Return to the main program.
        MOV   #01,HOUR        ;  If so, set the "HOUR" Register to "1",
        XOR   #01,AMPM        ;  and toggle the AM/PM flag bit.
END     RTI                   ;Return to the main program.

        .sect "VECTOR",7FF4h
        .word T1INT           ;Location of the Timer 1 Interrupt Vector.
        .word BEGIN           ;All other vectors will jump to "BEGIN".
        .word BEGIN
        .word BEGIN
        .word BEGIN
        .word BEGIN
        .end
```

5.2.1 Optional Calendar Functions for the Time-of-Day Clock

This code could be substituted for the Timer 1 Interrupt Service Routine of the previous example to give a TOD clock which will keep track of days, months, and years including leap years. To implement these functions, you will need to replace the register Equates, the Timer 1 Interrupt Service routine, and the value of the Stack pointer. Also, the new registers will need to be initialized, the previous register references deleted, and three look-up tables added. The Timer 1 initialization and Peripheral File Equates will remain the same since this routine uses the same 1/10th second interrupt time base as the previous routine.

The new code blocks required for the calendar functions are as follow:

1) New Register Equate values:

```
TIME .equ R4
;      R4 = TENTH              0
;      R5 = SECONDS            1
;      R6 = MINUTES            2
;      R7 = HOURS              3
;      R8 = DAYS               4
;      R9 = MONTH              5
;      R10= YEAR               6
MONTH      .equ      R9
YEAR       .equ      R10
YEAR100    .equ      R11      ;Century FLAG register.Incremented on
                              ;  100 yr intervals.
```

2) New Stack pointer Value and Register initialization.

```
START       MOV     #12,B               ;The Stack will need to start at #12
            LDSP                        ;  or greater.

            CLR     TIME                ;Clear TENTHS
            CLR     TIME+1              ;Clear SECONDS
            CLR     TIME+2              ;Clear MINUTES
            CLR     TIME+3              ;Clear HOURS
            MOV     #1,TIME+4           ;Set DAYS to '1'.
            MOV     #1,TIME+5           ;Set MONTHS to '1'.
            MOV     #89,TIME+6          ;Set YEARS to 1989.
            CLR     TIME+7              ;Clear the Century Flag register.
```

3) New Timer 1 Interrupt Service Routine:

```
T1INT       PUSH    A                   ;Save the A and B registers if necessary.
            PUSH    B
            CLR     B                   ;Start index at TENTHS.

LOOP        MOV     TIME(B),A           ;Get the value of the Present Time unit.

            CMP     #4,B                ;Are we checking DAYS?
            JEQ     DOMONTH             ;  If so, special check for months
            CMP     MAX(B),A            ;If not, has the MAX value of this Time
                                        ;  unit been met yet?
            JLO     DONE                ;If not then exit.

NEXT        MOV     MIN(B),A            ;Replace the value the the Time unit with
            MOV     A,TIME(B)           ;  its minimum value.
            CMP     #6,B                ;Are we at the end of the Century Yet?
            JNE     NXTUNIT             ;If not, continue incrementing B.
            INC     YEAR100             ;If so, inc. the Century Flag register.
            JMP     LOOP
NXTUNIT     INC     B                   ;Point to next higher time unit
            JMP     LOOP                ;Jump to Loop.

RESTOREB    POP     B                   ;Restore 'B' with Time Unit count.
DONE        INC     A                   ;Increment the present Time unit.
            MOV     A,TIME(B)
            POP     B                   ;Restore B and A then exit.
            POP     A
            RTI                         ;Return from Interrupt.
```

```
DOMONTH     PUSH    B                   ;M O N T H S
            MOV     MONTH,B             ;Get the value of the MONTH register.
            CMP     #2,B                ;Is it Feb?  If not jump to NORMAL
            JNE     NORMAL
            BTJO    #3,YEAR,NORMAL      ;If it is Feb, check for a leap year.
                                        ;  (leap years will end with 00b)
                                        ;If not leap year jump to NORMAL.

            CMP     #28+1,A             ;If leap year is it Feb. 29th yet?
            JMP     DODAYS

NORMAL      CMP     DAYS-1(B),A         ;If month is not Feb, is it max-ed out yet?
DODAYS      JLO     RESTOREB            ;  If not, restore index and to to DONE.

DONEMON     POP     B                   ;If so, restore index and go to NEXT.
            JMP     NEXT                ;exit to next time unit
```

4) New look-up tables required for routine:

```
MAX         .BYTE   09,59,59,23,31,12,99   ;Max values for TENTH, SECOND
                                           ; MINUTE,HOUR, AY,MONTH, and
                                           ; YEAR.
MIN         .BYTE   00,00,00,00,01,01,00   ;Minimum values for TENTH,
                                           ; SECOND, MINUTE, HOUR, DAY,
                                           ; MONTH,and YEAR.
DAYS        .BYTE   31,28,31,30,31,30      ;Max days in each Month.
            .BYTE   31,31,30,31,30,31
```

5.3 Frequency Counter Application

This routine uses the Timer 1 module in a frequency counter application. The frequency will be calculated by keeping track of a number of pulses for 1 second. The pulses to be counted will be input on the T1IC/CR pin, and the Timer 1 Compare register will be set up to give a 1-second interrupt. The value required by the Compare register to generate a 1-second interrupt period with a 20-mHz crystal is 04C4Ah with a /256 prescale. (See page 2–15 for formula and look-up table.) This counter application is designed to measure an input signal from 1 Hz to approximately 60 kHz.

A series of three registers will keep a decimal count of the number of pulses seen on the T1IC/CR pin until the Compare Equal interrupt is detected. After each Timer 1 Compare equal interrupt, the values in the COUNTX registers will be loaded into the STOREX registers for use by your program. The COUNTX registers will then be cleared and be ready to keep count of any pulses during the next second.

The source code for the Frequency Counter application is as follows:

```
        .title   "FREQUENCY COUNTER"  ;accurate to approx 60KHz

;       Allocate space for the seven (7) registers used in the routine.

COUNTH  .equ R2                 ;The 'COUNTX' registers are used to keep track
COUNTM  .equ R3                 ;  of the external pulses on the T1IC/CR pin.
COUNTL  .equ R4                 ;  They are incremented for each pulse.
STOREH  .equ R5                 ;The program uses the 'STOREX' registers to keep
STOREM  .equ R6                 ;  a record of the last frequency count.  These
STOREL  .equ R7                 ;  registers are updated every second.
ERROR   .equ R8                 ;(Optional, not used by program)  This register
                                ;  is provided to signal the program if an
                                ;  invalid frequency is detected. (Overflow out
                                ;  of the COUNTH register.)

;       Set up 'EQUATE' table for Peripheral File registers used in routine.

T1CM    .equ  P042              ;Timer 1 Compare Register 1 (MSB)
T1CL    .equ  P043              ;Timer 1 Compare Register 1 (LSB)
T1CTL1  .equ  P049              ;Timer 1 Control Register 1
T1CTL2  .equ  P04A              ;Timer 1 Control Register 2
T1CTL3  .equ  P04B              ;Timer 1 Control Register 3
T1CTL4  .equ  P04C              ;Timer 1 Control Register 4
T1PC1   .equ  P04D              ;Timer 1 Port Control Register 1
T1PC2   .equ  P04E              ;Timer 1 Port Control Register 2
T1PRI   .equ  P04F              ;Timer 1 Priority Control register.


;       Begin Initialization:
;
;             0  Set up Stack Pointer to begin at R10.
;             0  Initialize Registers to their 'Start' values.
;             0  Initialize Timer 1 operation.

       .text  7000h             ;Program starting location.

START  MOV    #10,B             ;Initialize the Stack Pointer to begin at
       LDSP                     ;  Register 10.

       CLR    COUNTL            ;Initialize the registers used in this routine
       CLR    COUNTM            ; to zero.
       CLR    COUNTH
       CLR    STOREL
       CLR    STOREM
       CLR    STOREH
       CLR    ERROR

       MOV    #4Ch,T1CM         ;Load the Timer 1 Compare register with #04C4Ah
       MOV    #4Ah,T1CL         ;  (MSB first) to give a 1 S compare.
       MOV    #00h,T1PC1        ;T1EVT and T1PWM are gen. purpose input pins
       MOV    #02h,T1PC2        ;  Enable T1IC/CR.
       MOV    #11h,T1CTL4       ;Select Dual Compare mode, enable falling edge
                                ;  and detect enable of T2IC1/CR.
       MOV    #07h,T1CTL1       ;Select the /256 prescale value.
       MOV    #00h,T1PRI        ;Set up interrupt priority as level 1.
       MOV    #01h,T1CTL2       ;Reset counter.
       MOV    #05h,T1CTL3       ;Clear flags, enable T1IC/CR and the Capture
                                ;  register to cause interrupts.
       EINT                     ;Globally enable Interrupts.
```

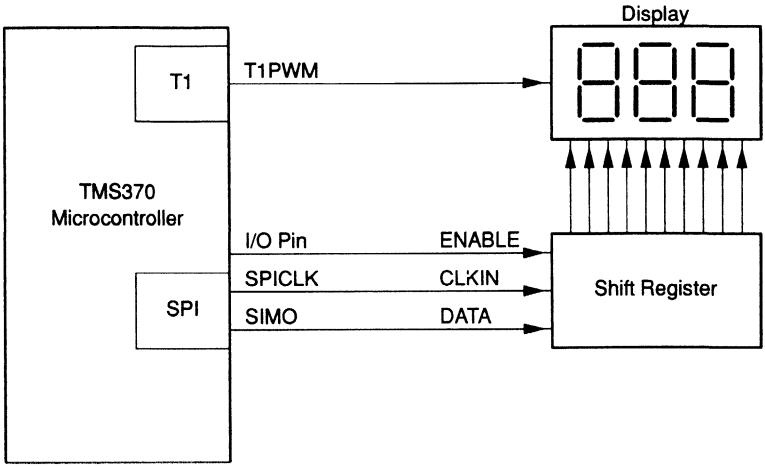
```
;       Begin your main program here.  A simple 'Jump/Loop' routine is used
;       in this application.


MAIN    JMP     MAIN                ;Loop on self while waiting for interrupt.


;       Timer 1 Interrupt service routine:  Routine will first check to see
;       which of the two enabled Timer 1 interrupt sources caused the interrupt.
;       If the T1C1 flag (Compare register 1) is set, the service routine will
;       jump to 'SAVE' and load the contents of the 'COUNTX' registers into the
;       "STOREX' registers, reinitialize the 'COUNTX' registers to zero, then
;       reset the timer.  If the T1EDGE flag (T1IC/CR pin) is set, the service
;       routine will increment the 'COUNTX' registers.

T1INT   BTJO    #20h,T1CTL3,SAVE ;Did T1 Compare reg. cause the T1 interrupt?
                                 ;  Yes, jump to 'SAVE'.
        MOV     #65h,T1CTL3      ;  No, Clear the T1IC/CR pin flag.
        MOV     #11h,T1CTL4      ;Reenable falling edge and detect enable of
                                 ;  T2IC1/CR.
LOW     DAC     #1,COUNTL        ;Increment the pulse count register 'COUNTL'.
        JC      MID              ;If the 'Low Count' register does not roll over,
        RTI                      ;  (Carry = 0) then return to the main program.
MID     DAC     #0,COUNTM        ;If Carry = 1, then COUNTM = <COUNTM> + 1.
        JC      HIGH             ;If the 'Mid Count' register does not roll over,
        RTI                      ;  (C=0), then return to the main program.
HIGH    DAC     #0,COUNTH        ;If Carry = 1, then COUNTH = <COUNTH> + 1.
        JNC     RETURN           ;(OPTIONAL)If the 'High Count' reg rolls over,
        MOV     #0FFh,ERROR      ;  set the 'ERROR' register.
RETURN  RTI                      ;Return to the main program.

SAVE    MOV     COUNTL,STOREL    ;Save the contents of the present Pulse counter
        MOV     COUNTM,STOREM    ; regs. into the STOREH:STOREM:STOREL regs.
        MOV     COUNTH,STOREH
        CLR     COUNTL           ;Clear the contents of the Pulse counter regs.
        CLR     COUNTM
        CLR     COUNTH
        MOV     #0C5h,T1CTL3     ;Clear the T1C1 flag.  Keep interrupts enabled.


;       Code could be added here to use the frequency count data.  For Example,
;       you could use the SPI port to send the data to your display.

        MOV     #01,T1CTL2       ;Reset the timer.
DONE    RTI                      ;Return to the main program.

        .sect "VECTOR",7FF4h
        .word T1INT              ;Set the Timer 1 interrupt vector to 'T1INT'.
        .word START              ;All other vectors point to the RESET vector.
        .word START
        .word START
        .word START
        .word START
        .end
```

5.4 Display Dimming Application Routine

Output a PWM signal with a varying duty cycle to control the brightness of a display. (VF, LED, etc.)

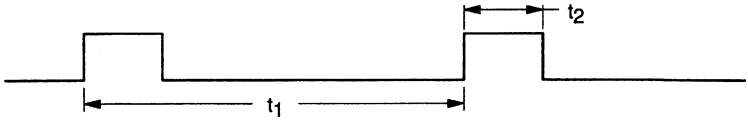
The schematic for this application is as follows:

Figure 5–4. Display Dimming Application



This application requires a PWM signal with a duty cycle which can vary from 0% to 100%. The resolution of the signal is 0.5% (200 steps from 0% to 100%). The Timer 1 module is used in this example, but Timer 2 may be used in a similar manner for those devices which contain Timer 2. Only the Dimming function is covered in this application. The SPI interface is illustrated in *Using the TMS370 SPI and SCI Modules Application Report* (SPNA006).

Figure 5–5. Display Dimming PWM Signal



In this PWM application, the pulse width duty cycle (t_2) may be changed under program control by altering the value in the Capture/Compare register. The Compare register controls the period of the signal (t_1) and is not changed in this routine.

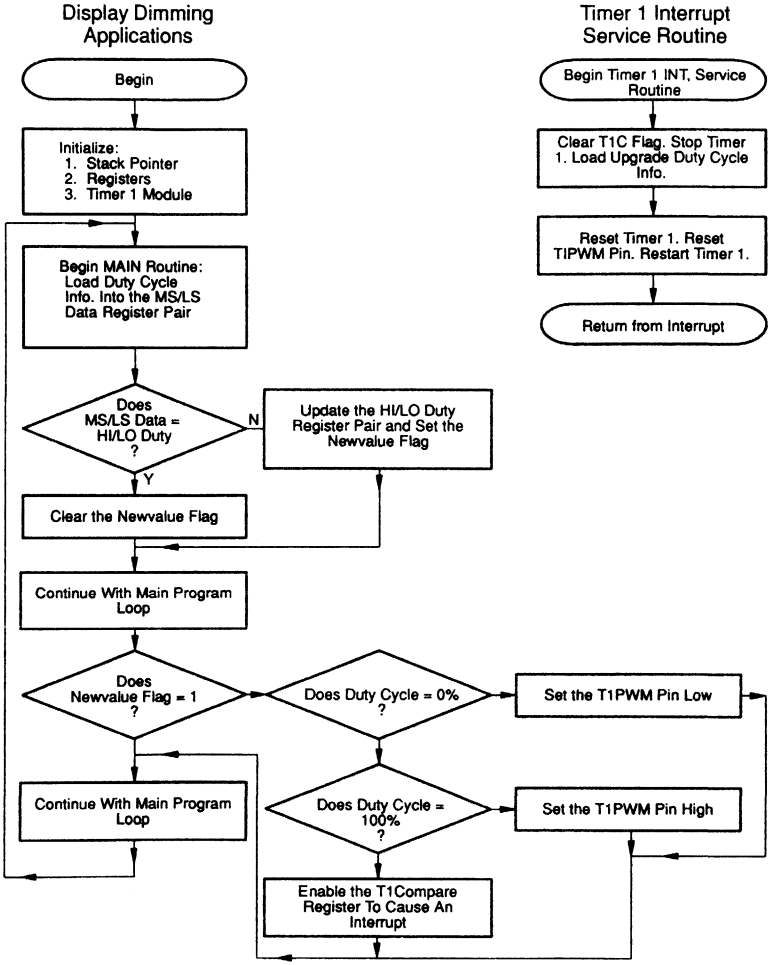
The main program will load any new values for the PWM duty cycle into the MS/LSDATA working registers. These values will be checked against the latest values in the HI/LODUTY registers. If they are different, the HI/

LODUTY registers will be updated, and the MAIN loop will compare to see if the new value is 0% or 100%. If so, the PWM pin will be set either LO or HIGH. If the new value is not 0% or 100%, the Timer 1 interrupt service routine will be enabled, and on the next interrupt, the PWM duty cycle will change.

When the Timer 1 service routine is entered, the routine will stop the PWM signal, load the new values, and restart. Stopping the PWM signal will help avoid the possibility of inverting the signal if a larger value was written than previously existed. (For example, changing from a 20% to an 30% duty cycle signal.)

The program flowchart diagram for this routine is illustrated in Figure 5–6.

Figure 5–6. Display Dimming Flowchart



The source code for the display dimming function is as follows:

```
            .title  "Display Dimming Function"
            .text   7000h

;           Allocate register space for the five (5) registers used in the
;           application routine.

HIDUTY      .equ  R2              ;Register used to store MSB of any new
                                  ;  Duty Cycle value.
LODUTY      .equ  R3              ;Register used to store LSB of any new
                                  ;  Duty Cycle value.
MSDATA      .equ  R4              ;Working registers where duty cycle information
LSDATA      .equ  R5              ;  is stored before the main program loads it
                                  ;  into the HI/LODUTY registers.
FLAGS       .equ  R6              ;Register used to store any software flags.

NEWVALUE    .dbit 0,FLAGS         ;Flag used to trigger a new PWM duty cycle.
                                  ;  (Bit 0 of the FLAGS register is used.)

;         Set up 'EQUATE' table for Peripheral File registers which will be used
;         in the routine.

T1CM        .EQU  P042            ;T1 Compare 1 register (MSB)
T1CL        .EQU  P043            ;T1 Compare 1 register (LSB)
T1CCM       .EQU  P044            ;T1 Capture/Compare 2 register (MSB)
T1CCL       .EQU  P045            ;T1 Capture/Compare 2 register (LSB)
T1PC1       .EQU  P04D            ;T1 Port Control reg 1
T1PC2       .EQU  P04E            ;T1 Port Control reg 2
T1CTL1      .EQU  P049            ;T1 Control register 1
T1CTL2      .EQU  P04A            ;T1 Control register 2
T1CTL3      .EQU  P04B            ;T1 Control register 3
T1CTL4      .EQU  P04C            ;T1 Control register 4
T1PRI       .EQU  P04F            ;T1 Interrupt Priority register


;           Begin initialization:

;               0  Set up Stack Pointer to begin at R050.
;               0  Initialize Registers to their 'START' values.
;               0  Initialize the Timer 1 operation.


START       MOV   #50h,B              ;Initialize the stack pointer to start at
            LDSP                      ;  register R050.

RESET       CLR   HIDUTY              ;Clear all registers.  The Duty cycle of
            CLR   LODUTY              ;  the PWM signal is initialized to 0%.
            CLR   MSDATA
            CLR   LSDATA
            CLR   FLAGS
```

```
;         Initialize the Timer 1 module

T1INIT    MOV    #04Eh,T1CM          ;Set up the T1 Compare register to contain
          MOV    #020h,T1CL          ;  (4E20h).  PWM frequency = 250 Hz.  (The
                                     ;  actual frequency is not very important
                                     ;  But should be > 100 Hz.)
                                     ;Must load MSB first then LSB

          MOV    HIDUTY,T1CCM        ;Load value for the Duty Cycle.
          MOV    LODUTY,T1CCL        ;  must load MSB first then LSB

          MOV    #0,T1PC1            ;T1EVT pin is set as a Gen. purpose input.
          MOV    #0,T1PRI            ;Set the T1 interrupt priority to level 1.
          MOV    #01110000b,T1CTL4   ;Select Dual Comp. mode, enable toggle
                                     ;  function of Compare regs. 1 and 2, and
                                     ;  reset Timer 1 on Compare 1 equal.
          MOV    #00000000b,T1CTL1   ;Select Sys. clock as timer clock source.
          MOV    #00000001b,T1CTL2   ;Reset the counter, (could enable WD here)
          MOV    #00000000b,T1CTL3   ;Clear and disable all interrupts.

          MOV    #00100000b,T1PC2    ;Enable T1PWM (Initial output value (0)
                                     ;  selected by bit 6), T1IC/CR is Gen.
                                     ;  purpose input.
          EINT                       ;Enable interrupts.


MAIN      ...                        ;Begin main program loop here.
          ...

;         In this example, the MAIN program will check the values of the
;         MS/LSDATA register pair against the HI/LODUTY register pair.
;         If the values are different, the PWM duty cycle will need to be
;         changed.  The MAIN loop also checks to see if any new value is
;         between 0% and 100%.  If so, the T1INT service will be entered.
;         If the new value is 0% or 100% exactly, the T1PWM pin will be set
;         to a general purpose output pin, with the data value of '0' (0%)
;         or '1' (100%).

CHKSAME   CMP    MSDATA,HIDUTY       ;Check to see if the new reading in MSDATA
          JNE    UPDATE              ;  ='s HIDUTY.  If not, jump to UPDATE.
          CMP    LSDATA,LODUTY       ;If so, check to see if new reading in
          JEQ    SAMEVALU            ;  LSDATA ='s LSDUTY.  If value is same as
                                     ;  last time.  No need to update HI/LODUTY.
                                     ;If not, goto UPDATE.

;         The values in the MS/LSDATA registers are NOT equal to the HI/LODUTY
;         values, therefore the HI/LODUTY registers need to be updated.

UPDATE    MOVW   LSDATA,LODUTY       ;A new value has been read and stored in
          SBIT1  NEWVALUE            ;  the HIDUTY/LODUTY register pair.
          JMP    ONWARD              ;  Set NEWVALUE then jump to ONWARD.

;         The values in the MS/LSDATA registers ARE equal to the HI/LODUTY
;         values.  No update of the HI/LODUTY registers is required.

SAMEVALU  SBIT0  NEWVALUE            ;The value read from MS/LSDATA ='s
                                     ;   HI/LODUTY. CLear NEWVALUE.
```

```
;         Continue on with the MAIN loop.

ONWARD    (NEXT INSTRUCTION)
          ...
          ...

          JBIT1 NEWVALUE,CHK0        ;Check to see if a new value has been
                                     ;  stored into the HI/LODUTY regs.
                                     ;  If so check for 0% or 100%.
          BR    ONWARD1              ;If not, branch to ONWARD1.


;         Check to see if the 'NEW' Duty Cycle is either 0% or 100%.  If so,
;         set the T1PWM pin accordingly.

CHK0      CMP   #0,LODUTY            ;Is LODUTY = 0? No, Check to see = 100%.
          JNE   CHK100               ;  Yes check the HIDUTY register.
          CMP   #0,HIDUTY            ;Is HIDUTY also = 0?
          JEQ   SETLOW               ;  Yes, set T1PWM line low.
                                     ;  No, check if 100%.
CHK100    CMP   #20h,LODUTY          ;Is LODUTY = 20h?
          JNE   T1ENABLE             ;  No, jump to T1ENABLE.
          CMP   #4Eh,HIDUTY          ;  If so, is HIDUTY = 4Eh?
          JEQ   SETHIGH              ;  Yes, set T1PWM line high.

;         If there has been a new value detected for the PWM duty cycle, and
;         that new value is not 0 (0%) or 4E20h (100%), then clear the NEWVALUE
;         Flag and enable T1INT.

T1ENABLE  SBIT0 NEWVALUE             ;Clear the NEWVALUE flag.
          MOV   #01h,T1CTL3          ;Allow the Compare flag to cause a timer
                                     ;  interrupt only when the PWM Duty Cycle
                                     ;  needs to be altered.

          ...                        ;Continue MAIN routine.
          ...
          BR    MAIN

;         This next section of code will only be executed if the desired duty
;         cycle is either 0% or 100% exactly.

SETLOW    MOV   #00010000b,T1PC2     ;Make the T1PWM pin an output pin with the
                                     ;  present data being output.
          MOV   #00010000b,T1PC2     ;Output a low value on the T1PWM pin.
          JMP   ONWARD1

SETHIGH   CLR MSDATA
          CLR LSDATA
          MOV   #01010000b,T1PC2     ;Make the T1PWM pin an output pin with the
                                     ;  present data being output.
          MOV   #01010000b,T1PC2     ;Output a high value on the T1PWM pin.
ONWARD1   ...                        ;Continue with the MAIN routine.
          ...
          ...
GOBACK    BR    MAIN
```

```
;       The Timer 1 Interrupt Service Routine follows.  This routine will
;       only be entered if a different duty cycle value is detected, and
;       that new duty cycle value is:  0 < value <4E20h.  (Between 0% and
;       100%.)

T1INT   MOV   #00000011b,T1CTL1   ;Stop T1 since an update has been read.
        MOV   HIDUTY,T1CCM        ;Load new value for the PWM duty cycle.
        MOV   LODUTY,T1CCL        ;  Must load MSB first then LSB.
        MOV   #00000001b,T1CTL2   ;Reset the counter.
        MOV   #01010000b,T1PC2    ;Reset the T1PWM pin to gen. purpose output
                                  ;  with the present value of the PWM pin.
        MOV   #01010000b,T1PC2    ;T1PWM pin will output a '1'.
        MOV   #01100000b,T1PC2    ;Reenable the T1PWM function with an
                                  ;  initial value of '1'.
        MOV   #01110000b,T1CTL4   ;Reenable the PWM toggling (T1C and T1CC)
        MOV   #00h,T1CTL1         ;Reselect the System clock as the T1 clock
                                  ;  source.
                                  ;The PWM signal will now run with the new
                                  ;  duty cycle until the next change.
        MOV   #00000000b,T1CTL3   ;Clear the T1 Compare 1 interrupt flag and
                                  ;  disable the T1 Compare 1 flag again.

RETURN  RTI                       ;Return to the MAIN routine.


;       Set up the Interrupt vectors.  Only T1INT is used in this example,
;       but the rest of the vectors have been loaded with the RESET vector
;       in case of an extraneous pulse.

        .sect "VECTORS",7FF4h     ;Location of the VECTOR table.
        .word T1INT               ;Timer 1 interrupt vector.
        .word START               ;Points to RESET vector.
        .word START               ;        "
        .word START               ;        "
        .word START               ;        "
        .word START               ;RESET vector.
        .end
```

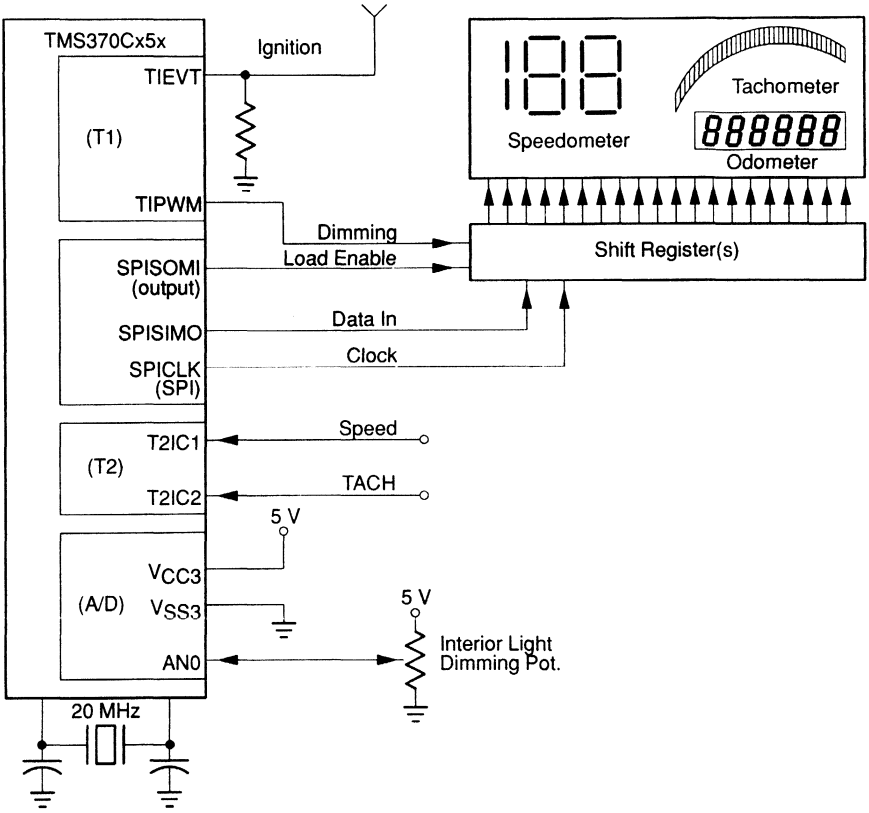
5.5 Speedometer and Tachometer Display Application

The purpose of this application example is to show you how a TMS370 device could be used to control a digital instrumentation cluster. The TMS370 module requirements for this example include Timer 1, Timer 2 , one A/D channel, and the SPI module. Also, the on-chip EEPROM could be used to keep a nonvolatile record of the odometer readings. This routine is written to show how the Timer modules could be used to control the Dimming and Pulse Width Measurement requirements of a digital instrument cluster. Certain calculation algorithms and subroutines are application specific and are left uncoded. Additional information concerning the A/D, EEPROM, and SPI modules may be found the following documents:

- *Using the TMS370 SPI and SCI Modules Application Report* (SPNA006)
- *Using the TMS370 EEPROM Modules Application Report* (SPNA007)
- *Using the TMS370 A/D Module Application Report* (SPNA005)

A block diagram of the digital instrumentation example is shown in Figure 5–7 .

Figure 5–7. Digital Instrumentation Cluster Application

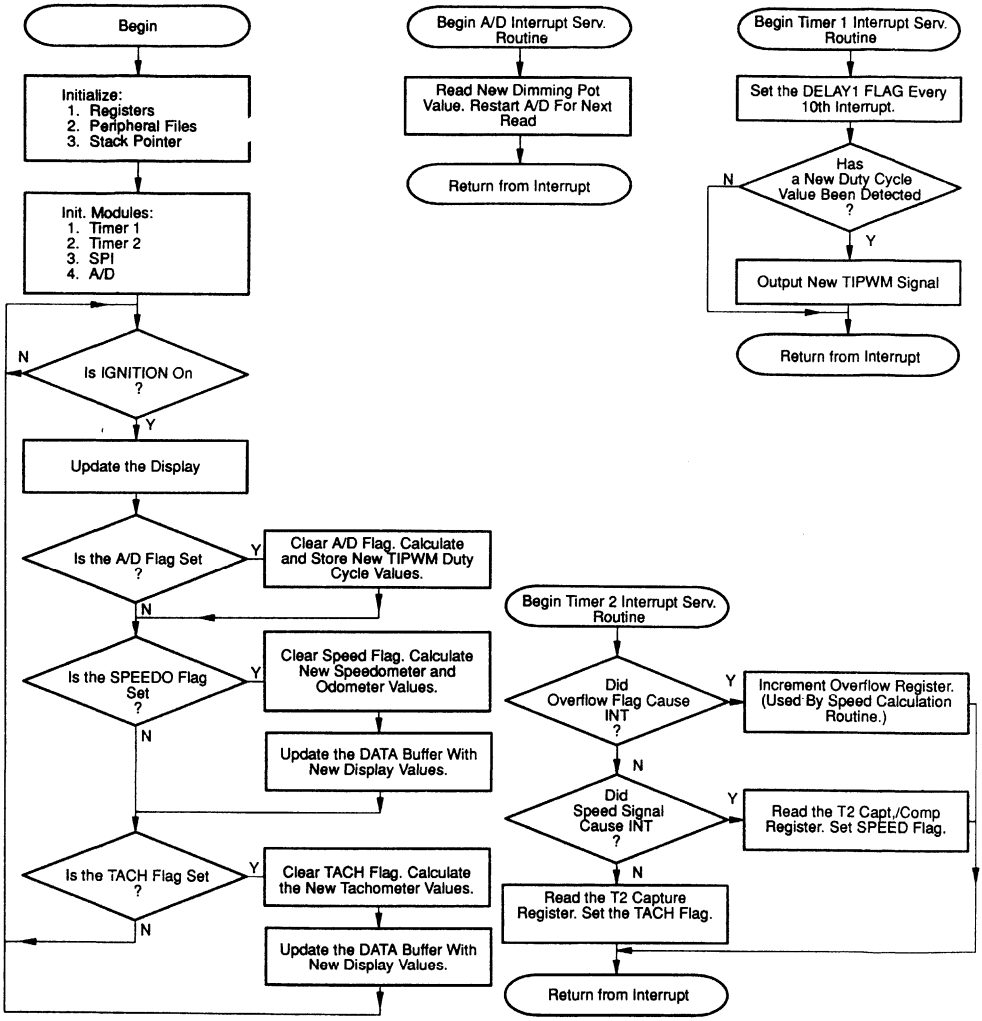


5.5.1 Application Overview and Theory of Operation

The basic functions of this application example include input signal measurement, display dimming, serial communication, and conversion of one A/D channel. The Speed and Tach readings will be measured using the two Input Capture registers of Timer 2. The dimming of the display will be controlled by reading an A/D channel which is connected to a potientiometer. This A/D information will be used to determine the duty cycle of a PWM signal output from Timer 1. The information sent to the display will be controlled using the SPI module. The MAIN routine in this example will check to see that the Ignition switch is on. Once the Ignition switch is on, the display will begin to be updated, and a series of Flags will be checked to determine any needed operation.

The flowchart for this routine is shown in Figure 5–8.

Figure 5–8. Instrumentation Flowchart



5.5.1.1 Timer 1 Module Operation

The Timer 1 module will be used to output a PWM signal to control the brightness of the display. Timer 1 will operate in the Dual Compare mode. The period of the PWM signal will be controlled by the T1 Compare 1 register, and the pulse width will be controlled by the T1 Capture/Compare register. The pulse width duty cycle may be changed under program control by altering the value in the T1 Capture/Compare register.

The MAIN routine will check to see if the newest reading from the A/D has changed since it was last read. If the values are different, the NEWVALUE flag will be set. If the values are the same, the NEWVALUE flag will be cleared. The Timer 1 service routine will check this flag. If the NEWVALUE flag is cleared, the present PWM duty cycle will continue. If the NEWVALUE flag if set, the interrupt routine will stop the PWM signal, load the new duty cycle values (HI/LODUTY) into the T1CC registers, and restart the PWM signal. Stopping the PWM signal will help avoid the possibility of inverting the signal if a larger value was written than previously existed. (For example, changing from a 20% to an 30% duty cycle signal.)

5.5.1.2 Timer 2 Module Operation

Timer module 2 will be used to measure the Speed and Tach input signals. The module will be set up for the Dual Capture mode to enable both 16-bit Capture registers. The T2IC1 pin, the T2 Capture/Compare register, and any T2 counter overflows will be used to determine the SPEED function, while the T2IC2 pin, the T2 Capture register, and any T2 counter overflows will be used for the TACH function.

When a valid signal occurs on either T2 Input Capture pin, the associated Capture register is loaded with the value of the T2 counter. The Timer 2 service routine will then read the contents of the Capture register and any T2 overflows which may have occurred. This information can be used to determine the speed and tach readings by keeping track of how long it has been since the last pulse occurred. The actual conversion routines used to determine the speedometer, odometer, and tachometer display information is application dependent, and is not coded in this example.

5.5.1.3 SPI Module Operation

The SPI module will be used to send the display information to the instrument cluster. The MAIN routine will constantly update the display with any new TACH information every 1/20 second, and update the complete display every 1/2 second. The actual number of bytes to be sent, the data format, and how often the display needs updated are all application specific variables that you may alter for your needs.

5.5.1.4 A/D Module Operation

One channel of the A/D module (AN0) will be read continually to determine the desired brightness of the display. The display brightness is application specific, so you will need to define the algorithm used to determine duty cycle of the Timer 1 PWM signal. Also, the brightness of the display may not be linear with the duty cycle of the PWM signal.

5.5.2 Digital Instrumentation Cluster Software Example

The source code for the Instrumentation Cluster is as follows:

```
            .title  "Digital Instrumentation Cluster Controller"
            .text 7000h

;           Allocate space for the registers used in the application routine.

HIDUTY      .equ  R2            ;Register used to store MSB of any new
                                ;  Duty Cycle value.
LODUTY      .equ  R3            ;Register used to store LSB of any new
                                ;  Duty Cycle value.
MS50        .equ  R4            ;Used for the 50 mS delay in T1 Int routine.
HALFSEC     .equ  R5            ;Used for 1/2 second decrementer value.

0DO100K     .equ  R6            ;Used to store the Odo's 100K digit info.
ODO10K      .equ  R7            ;Used to store the Odo's 10K digit info.
ODO1000     .equ  R8            ;Used to store the Odo's 1K digit info.
ODO100      .equ  R9            ;Used to store the Odo's 100's digit info.
ODO10       .equ  R10           ;Used to store the Odo's 10's digit info.
ODO1        .equ  R11           ;Used to store the Odo's 1's digit info.
ODOTENTH    .equ  R12           ;Used to store the Odo's 1/10's digit info.

FLAGS       .equ  R13           ;Register used to store any software flags.
OVERCNT     .equ  R14           ;Used to keep count of T2 overflows.
OVERSPD     .equ  R15           ;Used for any T2 overflows during SPEED pulse.
OVERTACH    .equ  R16           ;Used for any T2 overflows during TACH pulse.
TEST1       .equ  R17           ;Used for the ignition switch test.
ADREAD      .equ  R18           ;Used to store A/D data in A/D int. routine.
TACH1       .equ  R19           ;Used to store a byte of TACH information.
TACH2       .equ  R20           ;Used to store a byte of TACH information.
TACH3       .equ  R21           ;Used to store a byte of TACH information.
TACH4       .equ  R22           ;Used to store a byte of TACH information.
HISPEED     .equ  R23           ;Used to store the Speedo's 100's digit info.
MIDSPEED    .equ  R24           ;Used to store the Speedo's 10's digit info.
LOSPEED     .equ  R25           ;Used to store the Speedo's 1's digit info.
ADLAST      .equ  R26           ;Storage register for the Last A/D reading.
SPEEDMSB    .equ  R27           ;Used in the SPEED calc. routine for the MSB.
SPEEDLSB    .equ  R28           ;Used in the SPEED calc. routine for the LSB.
TACHMSB     .equ  R29           ;Used in the TACH calc. routine for the MSB.
TACHLSB     .equ  R30           ;Used in the TACH calc. routine for the LSB.
```

```
DATA        .equ  R31           ;Set aside a 20 byte block of RAM that will
                                ;  be used to store the SPI information.
                                ;  In this example the 'DATA' block is set
                                ;  up as follows:
                                ;  DATA     :    Tach information (n)
                                ;  DATA+1   :    Tach information (n+1)
                                ;  DATA+2   :    Tach information (n+2)
                                ;  DATA+3   :    Tach information (n+3)
                                ;  DATA+4   :    Speedometer (100's Digit)
                                ;  DATA+5   :    Speedometer (10's Digit)
                                ;  DATA+6   :    Speedometer (1's Digit)
                                ;  DATA+7   :    Odometer (100K digit)
                                ;  DATA+8   :    Odometer (10K digit)
                                ;  DATA+9   :    Odometer (1K digit)
                                ;  DATA+10  :    Odometer (100's digit)
                                ;  DATA+11  :    Odometer (10's digit)
                                ;  DATA+12  :    Odometer (1's digit)
                                ;  DATA+13  :    Odometer (1/10's digit)
                                ;  DATA+14  :    Unused in this example.
                                ;  DATA+15  :             "
                                ;  DATA+16  :             "
                                ;  DATA+17  :             "
                                ;  DATA+18  :             "
                                ;  DATA+19  :             "
                                ;  DATA+20  :             "

NEWVALUE    .dbit 0,FLAGS       ;Flag used to trigger a new PWM duty cycle.
IGNITION    .dbit 1,FLAGS       ;Flag used to tell the MAIN routine is the
                                ;  Ignition switch is on or off.
DELAY1      .dbit 2,FLAGS       ;Flag used to signal a 1/10th second delay.
SPDREAD     .dbit 3,FLAGS       ;Flag used to show a new SPEED reading has been
                                ;  taken.
TACHREAD    .dbit 4,FLAGS       ;Flag used to show a new TACH reading has been
                                ;  taken.
ADFLAG      .dbit 5,FLAGS       ;Flag used to signal new A/D information has
                                ;  been read.
```

```
;       Set up 'EQUATE' table for Peripheral File registers which will be used
;       by the Timer 1, Timer 2, SPI, and A/D Modules.

T1CNTRM   .EQU   P040          ;Timer 1 Counter MSB
T1CNTRL   .EQU   P041          ;Timer 1 Counter LSB
T1CM      .EQU   P042          ;Timer 1 Compare register MSB
T1CL      .EQU   P043          ;Timer 1 Compare register LSB
T1CCM     .EQU   P044          ;Timer 1 Capture/Compare register MSB
T1CCL     .EQU   P045          ;Timer 1 Capture/Compare register LSB
T1CTL1    .EQU   P049          ;Timer 1 control register 1
T1CTL2    .EQU   P04A          ;Timer 1 control register 2
T1CTL3    .EQU   P04B          ;Timer 1 control register 3
T1CTL4    .EQU   P04C          ;Timer 1 control register 4
T1PC1     .EQU   P04D          ;Timer 1 Port Control 1
T1PC2     .EQU   P04E          ;Timer 1 Port Control 2
T1PRI     .EQU   P04F          ;Timer 1 Interrupt Priority control

T2CNTRM   .EQU   P060          ;Timer 2 Counter MSB
T2CNTRL   .EQU   P061          ;Timer 2 Counter LSB
T2CM      .EQU   P062          ;Timer 2 Compare register MSB
T2CL      .EQU   P063          ;Timer 2 Compare register LSB
T2CCM     .EQU   P064          ;Timer 2 Capture 1 / Compare 2 register MSB
T2CCL     .EQU   P065          ;Timer 2 Capture 1 / Compare 2 register LSB
T2CTL1    .EQU   P06A          ;Timer 2 control register 1
T2CTL2    .EQU   P06B          ;Timer 2 control register 2
T2CTL3    .EQU   P06C          ;Timer 2 control register 3
T2PC1     .EQU   P06D          ;Timer 2 Port Control 1
T2PC2     .EQU   P06E          ;Timer 2 Port Control 2
T2PRI     .EQU   P06F          ;Timer 2 Interrupt Priority control

SPICCR    .EQU   P032          ;SPI Configuration Control register
SPICTL    .EQU   P033          ;SPI Control register
SPIBUF    .EQU   P037          ;Receive Data Buffer register
SPIDAT    .EQU   P039          ;Serial Data register
SPIPC1    .EQU   P03D          ;SPI Port Control reg 1
SPIPC2    .EQU   P03E          ;SPI Port Control reg 2
SPIPRI    .EQU   P03F          ;SPI Interrupt Priority register

ADCTL     .EQU   P070          ;Analog Control register
ADSTAT    .EQU   P071          ;Analog Status and Interrupt register
ADDATA    .EQU   P072          ;Analog Conversion Data register
ADIN      .EQU   P07D          ;Port E Data Input register
ADENA     .EQU   P07E          ;Port E Input Enable register
ADPRI     .EQU   P07F          ;Port E Interrupt Priority register
```

```
;         Begin initialization:

;             0  Set up Stack Pointer to begin at R60.
;             0  Initialize Registers to their 'START' values.
;             0  Initialize the Timer 1 Module.
;             0  Initialize the Timer 2 Module.
;             0  Initialize the SPI Module.
;             0  Initialize the A/D Module.


START     MOV   #60,B               ;Initialize the stack pointer to start at
          LDSP                      ;  register R60.
          DINT                      ;Globally disable all interrupts.

;         INITIALIZE the Registers to their power-up values.

RESET     MOV   #0C3h,HIDUTY        ;The Duty cycle of the PWM signal is
          MOV   #048h,LODUTY        ;  initialized to approximately 100%.

;         Also, update the ODO registers from EEPROM (Not shown).

          MOV   #10,HALFSEC         ;Start with the value 10.
          MOV   #5,MS50             ;Start with the value 5.

;         CLEAR THE REMAINING REGISTERS.

          MOV   #39,B              ;This routine will clear the 38 registers
          CLR   A                   ;  starting at FLAGS and ending at DATA+20.
CLRREGS   MOV   A,FLAGS-1(B)
          DJNZ  B,CLRREGS

;         Begin the Module initialization routines.

T1INIT    MOV   #0C3h,T1CM          ;Set up the T1 Compare register to contain
          MOV   #04Fh,T1CL          ;  (C34Fh).  PWM frequency = 100 Hz.  (The
                                    ;  actual frequency is not very important
                                    ;  for this application.)
                                    ;Must load MSB first then LSB

          MOV   HIDUTY,T1CCM        ;Load value for the Duty Cycle.
          MOV   LODUTY,T1CCL        ;  must load MSB first then LSB

          MOV   #0,T1PC1            ;T1EVT pin is set as a Gen. purpose input.
          MOV   #00100000b,T1PC2    ;Enable T1PWM (Initial output value (0)
                                    ;  selected by bit 6), T1IC/CR is Gen.
                                    ;  purpose input.
          MOV   #0,T1PRI            ;Set the T1 interrupt priority to level 1.
          MOV   #01110000b,T1CTL4   ;Select Dual Comp. mode, enable toggle
                                    ;  function of Compare regs. 1 and 2, and
                                    ;  reset Timer 1 on Compare 1 equal.
          MOV   #00000000b,T1CTL1   ;Select Sys. clock as timer clock source.
          MOV   #00000001b,T1CTL3   ;Clear all and enable T1C1 interrupt.
          MOV   #00000001b,T1CTL2   ;Reset the counter, (could enable WD here)
```

```
T2INIT  MOV   #0,T2PC1            ;T2EVT pin is set as a Gen. purpose input.
        MOV   #00100010b,T1PC2    ;Enable T2IC1 and T2IC2 pin to function
                                  ;   as Input Capture triggers.
        MOV   #0,T2PRI            ;Set the T2 interrupt priority to level 1.
        MOV   #10000011b,T2CTL3   ;Select Dual Capture mode, enable high to
                                  ;   low pulse to cause a capture for both
                                  ;   the SPEED and TACH signals.
        MOV   #00000110b,T1CTL2   ;Clear and enable both Input Capture Ints.
        MOV   #011h.T2CTL1        ;Enable and clear the T2 Overflow flag,
                                  ;   Select the sys clock as clock source,
                                  ;   and reset T2.


SPIINIT MOV   #2,SPIPC1           ;Enable the SPICLK pin.
        MOV   #20h,SPIPC2         ;Enable the SPISIMO pin, make SPISOMI
                                  ;   a general purpose input pin.
        MOV   #11000110b,SPICCR   ;Reset SPI, 7-bit data out on falling
                                  ;   SPICLK. Baud rate = clkin/8.
        MOV   #00000110b,SPICTL   ;Master mode, enable TALK.


ADINIT  MOV   #001h,ADSTAT        ;Enable interrupt clear flags.
        MOV   #0,ADPRI            ;Select interrupt level 1 for the A/D.
        MOV   #040h,ADCTL         ;Start Sampling. Vss3 selected as Vref,
                                  ;   AN0 selected as input channel.
        MOV   #0C0h,ADCTL         ;Start conversion

        EINT                      ;Enable interrupts.

;       The Initialization Block is completed.

;       Begin MAIN Program here.
;       CHECK TO SEE IF THE IGNITION SWITCH IS TURNED ON.

MAIN    MOV   T1PC1,TEST1         ;See if the Ingition switch is on or off.
        BTJZ  #08h,TEST1,CLRIGN   ;If low (Ign OFF) jump to CLRING.
SETIGN  SBIT1 IGNITION            ;   If hi (Ign ON), set the INGITION bit.
        JMP   PAST1
CLRIGN  SBIT0 IGNITION            ;If Ign. is off, clear the IGNITION bit.
PAST1   ...                       ;Continue on with the MAIN routine.
        ...
        ...
CHKIGN  JBIT0 IGNITION,MAIN       ;If the IGNITION flag is cleared (IGN off)
                                  ;   then jump back to MAIN.
        SBIT0 IGNITION            ;If IGNITION flag is set (IGN on), clear
                                  ;   the flag then update the display.
```

```
;       UPDATE THE DISPLAY.

;       When the Ingition switch is on, the display needs to be updated.
;       How often the display needs updated is dependent on your system
;       requirements.  Also, all information may not need updated each time.
;       (i.e., the Odometer does not need updated as often as the Tachometer
;       does.)  Also, the number of data bytes sent via the SPI depends on
;       the type of display being used.  Typically 1 bit of data will be sent
;       per segment displayed.

;       The display routine assumes that partial information needs updated
;       every 1/20th second, and all display information needs updated every
;       1/2 second.  It is up to you to decide what values are required and
;       how often they need updated.  A block of 20 bytes starting at 'DATA'
;       is set aside to store the information required.

UPDATE  JBIT0 DELAY1,UPDATE       ;Wait for the 1/20th second delay from the
                                  ;  Timer 1 interrupt routine.
        SBIT0 DELAY1              ;Clear the DELAY1 flag after being set.
        DJNZ  HALFSEC,LOADPART    ;Check to see if the complete display
                                  ;  needs updated yet.
        MOV   #10,HALFSEC         ;Yes, reload SECOND and set the 'B'
LOADALL MOV   #??,B               ;  register to your desired value.
        JMP   CHKSPI
LOADPART MOV  #??,B               ;Load 'B' register with your desired value.

CHKSPI  BTJZ  #040h,SPICTL,CHKSPI ;Check to see if you can send a byte of
                                  ;  data yet. If so continue.
        MOV   DATA-1(B),A         ;Load the data to be sent out into
        MOV   A,SPIDAT            ;  the SPIDAT register.
        DJNZ  B,CHKSPI            ;Is the data string through yet?
        MOV   #025h,SPIPC2        ;Toggle SPISOMI to latch data.
        MOV   #021h,SPIPC2        ;Pull SPISOMI low again.

;       CHECK TO SEE IF A NEW A/D READING HAS BEEN TAKEN.  If so, check to
;       see if this reading is different from the last reading.

CHKAD   JBIT0 ADFLAG,RETURN       ;Has a new value been read by the A/D
        SBIT0 ADFLAG              ;  Interrupt Service routine?  No, jump
                                  ;  to RETURN.
CHKSAME CMP   ADREAD,ADLAST       ;  Yes,Are values same?
        JEQ   RETURN              ;Yes, jump to RETURN.
        MOV   ADREAD,ADLAST       ;No, load new A/D data into the ADLAST reg.


CALCDUTY ...                      ;Calculate the new daty cycle values.
        ...
;       In this section of code, you will need to decide what algorithm and
;       variables your application requires for the dimming function.
;       The register pair HI/LODUTY will need to be loaded with the values
;       that will then be loaded into the T1 Capture/Compare register by
;       the Timer 1 Interrupt serivce routine to determine a new PWM duty
;       cycle.  A possible solution could be a table look-up algorithm that
;       would load a 16-bit value into the HI/LODUTY registers with a maximum
;       value of less than C34Fh. (Value of the T1 PWM signal period.)
        ...
        ...
        MOV   #??,HIDUTY          ;Load the new Duty cycle value into the
        MOV   #??,LODUTY          ;  HI/LODUTY register pair.

        SBIT1 NEWVALUE            ;  Set the NEWVALUE flag, which is used
                                  ;  in the Timer 1 Service routine.

;       CHECK FOR A NEW SPEEDOMETER VALUE.
```

```
CHKSPEED JBIT0 SPDREAD,CHKTACH      ;Has a new SPEED value been seen by the
                                    ;  Timer 2 Interrupt routine?  No, jump
                                    ;  to CHKTACH.
         SBIT0 SPDREAD              ;Yes, reset the flag and calculate the
                                    ;  SPEED variable.

CALCSPD  ...                        ;Calculate the new SPEED and Odometer
                                    ;  values.

LDSPEED  MOV   #3,B                 ;Move the calculated speed readings to the
         MOV   HISPEED-1(B),A       ;  3 registers in the DATA buffer set up
         MOV   A,DATA+3(B)          ;  for the SPEED info. (Used by the SPI.)
         DJNZ  B,LDSPEED

LOADODO  MOV   #7,B                 ;Move the calculated Odometer values to the
         MOV   ODO100K-1(B),A       ;  7 registers in the DATA buffer set up
         MOV   A,DATA+6(B)          ;  for the ODO info. (Used by the SPI.)
         DJNZ  B,LOADODO

;        CHECK FOR A NEW TACHOMETER VALUE.

CHKTACH  JBIT0 TACHREAD,RETURN      ;Has a new TACH value been seen by the
                                    ;  Timer 2 Interrupt routine?  No, jump
                                    ;  to RETURN.
         SBIT0 TACHREAD             ;Yes, reset the flag and calculate the
                                    ;  TACH variable.

CALCTACH ...                        ;Your TACH calculation routine goes here...


LOADTACH MOV   #4,B                 ;Move the calculated TACH readings to the
         MOV   TACH1-1(B),A         ;  4 registers in the DATA buffer set up
         MOV   A,DATA-1(B)          ;  for the TACH information. (Used by
         DJNZ  B,LOADTACH           ;  the SPI.)

RETURN   BR    MAIN                 ;Return to beginning.


;     INTERRUPT ROUTINES TO FOLLOW:

;        The Timer 1 Interrupt Service Routine follows.  This routine will
;        will be entered every 10 mS.  The duty cycle will be altered only
;        if new data has been loaded into the HIDUTY/LODUTY Register Pair.

T1INT    DJNZ  MS50,CLEAR           ;Every 5th time through this routine,
                                    ;  the DELAY1 flag needs to be set.
         MOV   #5,MS50              ;Reset the MS50 register.
         SBIT1 DELAY1               ;Set the DELAY1 flag.

CLEAR    MOV   #00000001b,T1CTL3    ;Clear the T1 Compare 1 interrupt flag and
                                    ;  reenable the T1 Compare 1 flag again.
         JBIT0 NEWVALUE,T1RET       ;If an update to the PWM duty cycle is
         SBIT0 NEWVALUE             ;  required, continue with the rest of
                                    ;  of the routine.  If not, jump to RTI.

         MOV   #00000011b,T1CTL1    ;Stop T1 since an update has been read.
         MOV   HIDUTY,T1CCM         ;Load new value for the PWM duty cycle.
         MOV   LODUTY,T1CCL         ;  Must load MSB first then LSB.
         MOV   #00000001b,T1CTL2    ;Reset the counter.
         MOV   #01010000b,T1PC2     ;Reset the T1PWM pin to gen. purpose output
                                    ;  with the present value of the PWM pin.
         MOV   #01010000b,T1PC2     ;T1PWM pin will output a '1'.
```

```
        MOV   #01100000b,T1PC2    ;Reenable the T1PWM function with an
                                  ;  initial value of '1'.
        MOV   #01110000b,T1CTL4   ;Reenable the PWM toggling (T1C and T1CC)
        MOV   #00h,T1CTL1         ;Reselect the System clock as the T1 clock
                                  ;  source.
                                  ;The PWM signal will now run with the new
                                  ;  duty cycle until the next change.
T1RET   RTI                       ;Return to the MAIN routine.


;       The Timer 2 Interrupt Service Routine follows.  This routine will
;       provide the frequency data from the SPEED and TACH inputs.

T2INT   BTJO  #08h,T2CTL1,OVRFLW  ;Was the Interrupt caused by the Timer 2
                                  ;  Overflow bit?  If so, goto OVRFLW.

        BTJO  #040h,T2CTL2,CAPT2  ;Was Interrupt caused by TACH signal?
                                  ;  if so, go to CAPTURE2.  If not
                                  ;  Int must have caused by SPEEd signal.

;       Read the Capture/Compare register for the SPEED value.

CAPT1   MOV   #01100110b,T2CTL2   ;Clear the flag and reenable the interrupt.
        MOV   T2CCL,SPEEDLSB      ;Read the Capture/comp reg and store values
        MOV   T2CCM,SPEEDMSB      ;  into SPEEDMSB/LSB reg pair.  Must read
                                  ;  LSB first.
        MOV   OVERCNT,OVERSPD      ;Save the contents of the OVERCNT reg
                                  ;  in OVERSPD.  Used in CALC routine.
        SBIT1 SPDREAD             ;Set the SPDREAD flag.
SPDRET  RTI


;       Read the Capture register for the TACH value.

CAPT2   MOV   #10100110b,T2CTL2   ;Clear the flag and reenable the interrupt.
        MOV   T2CL,TACHLSB        ;Read the Capture register and store values
        MOV   T2CM,TACHMSB        ;  into the TACHMSB/LSB reg pair.  Must
                                  ;  read LSB first.
        MOV   OVERCNT,OVERTACH     ;Save the contents of the OVERCNT reg
                                  ;  in OVERTACH.  Used in CALC routine.
        SBIT1 TACHREAD            ;Set the TACHREAD flag.
TACHRET RTI

;       Increment the OVERCNT register.

OVRFLW  INC   OVERCNT             ;Increment the Overflow counter register
                                  ;  if an overflow has occured.
        RTI

;       The A/D Interrupt Service Routine is to follow.  This routine will
;       read ADDATA and store the value into the ADREAD register.

ADINT   MOV   ADDATA,ADREAD       ;Read the A/D data.
        MOV   #040h,ADCTL         ;Start new sample.
        MOV   #080h,ADCTL         ;Start new conversion
        SBIT1 ADFLAG              ;Set the ADFLAG bit to signal an A/D
                                  ;  reading has recently been completed.
GOBACK  RTI                       ;Return to the MAIN routine.
```

```
.sect "VECTORS",7FFCh      ;Interrupt Vectors:
.word ADINT                ;  A/D      vector
.word T2INT                ;  Timer 2  vector
.word GOBACK               ;  SCI TX   vector  (Not used)
.word GOBACK               ;  SCI RX   vector  (Not used)
.word T1INT                ;  Timer 1  vector
.word GOBACK               ;  SPI      vector  (Not used)
.word GOBACK               ;  INT 3    vector  (Not used)
.word GOBACK               ;  INT 2    vector  (Not used)
.word GOBACK               ;  INT 1    vector  (Not used)
.word START                ;  RESET    vector
.end
```

Chapter 6

Conclusion

The Timer modules of the TMS370 8-bit Microcontroller family are designed to provide the flexibility to meet a broad range of timer/counter applications. The software and interface examples illustrate how the basic functions of the Timer modules along with other modules of the TMS370 family can be used to provide a cost-effective system solution. This application report has been designed to be used in conjunction with the *TMS370 Family Data Manual*. The data manual is a valuable reference source and provides many answers to questions not addressed in this report.

All software examples illustrated in this report, as well as other helpful codes and information may be downloaded from our TMS370 bulletin board. The parameters are: 8 data, no parity, and 1 stop bit. If you have questions concerning the TMS370 family, please feel free to contact us as follows:

- Technical Hotline (713) 274-2370
- Bulletin Board (713) 274-3700

Appendix A

Timer 1 Control Registers

Timer 1 is controlled and accessed through registers in the Peripheral File. These registers are listed in Figure A–1 and described in the *TMS370 Family Data Manual* (Section 7). The bits shown in shaded boxes in Figure A–1 are Privilege Mode bits, that is, they can only be written to in the Privilege Mode. The T1 Operational Mode Block diagrams are shown in Figures A–2 and A–3.

Figure A–1. Timer 1 Control Registers

Peripheral File Frame 4: Timer 1 Control Registers

PF	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Name
P040	Bit 15	Counter MSB						Bit 8	T1CNTR
P041	Bit 7	Counter LSB						Bit 0	
P042	Bit 15	Compare Register MSB						Bit 8	T1C
P043	Bit 7	Compare Register LSB						Bit 0	
P044	Bit 15	Capture/Compare Register MSB						Bit 8	T1CC
P045	Bit 7	Capture/Compare Register LSB						Bit 0	
P046	Bit 15	Watchdog Counter MSB						Bit 8	WDCNTR
P047	Bit 7	Watchdog Counter LSB						Bit 0	
P048	Watchdog Reset Key								WDRST
P049	WD OVRFL TAP SEL	WD INPUT SELECT 2	WD INPUT SELECT 1	WD INPUT SELECT 0	—	T1 INPUT SELECT 2	T1 INPUT SELECT 1	T1 INPUT SELECT 0	T1CTL1
P04A	WD OVRFL RST ENA	WD OVRFL INT ENA	WD OVRFL INT FLAG	T1 OVRFL INT ENA	T1 OVRFL INT FLAG	—	—	T1 SW RESET	T1CTL2

Mode: Dual Compare Registers

P04B	T1EDGE INT FLAG	T1C2 INT FLAG	T1C1 INT FLAG	—	—	T1EDGE INT ENA	T1C2 INT ENA	T1C2 INT ENA	T1CTL3
P04C	T1MODE = 0	T1C1 OUT ENA	T1C2 OUT ENA	T1C1 RST ENA	T1CR OUT ENA	T1EDGE POLAR-ITY	T1CR RST ENA	T1EDGE DET ENA	T1CTL4

Mode: Capture/Compare Registers

P04B	T1EDGE INT FLAG	—	T1CC1 INT FLAG	—	—	T1EDGE INT ENA	—	T1C1 INT ENA	T1CTL3
P04C	T1MODE = 1	T1C1 OUT ENA	—	T1C1 RST ENA	—	T1EDGE POLAR-ITY	—	T1EDGE DET ENA	T1CTL4
P04D	—	—	—	—	T1EVT Data In	T1EVT Data Out	T1EVT Function	T1EVT Data Dir	T1PC1
P04E	T1PWM Data In	T1PWM Data Out	T1PWM Function	T1PWM Data Dir	T1IC/CR Data In	T1IC/CR Data Out	T1IC/CR Function	T1IC/CR Data Dir	T1PC2
P04F	T1 STEST	T1 Priority	—	—	—	—	—	—	T1PRI

Figure A–2. Timer 1 – Dual Compare Mode

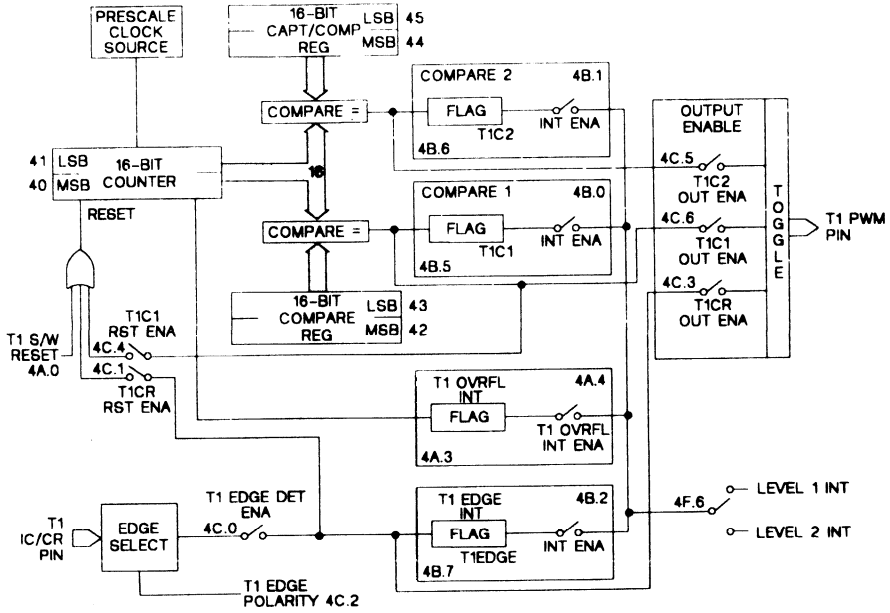
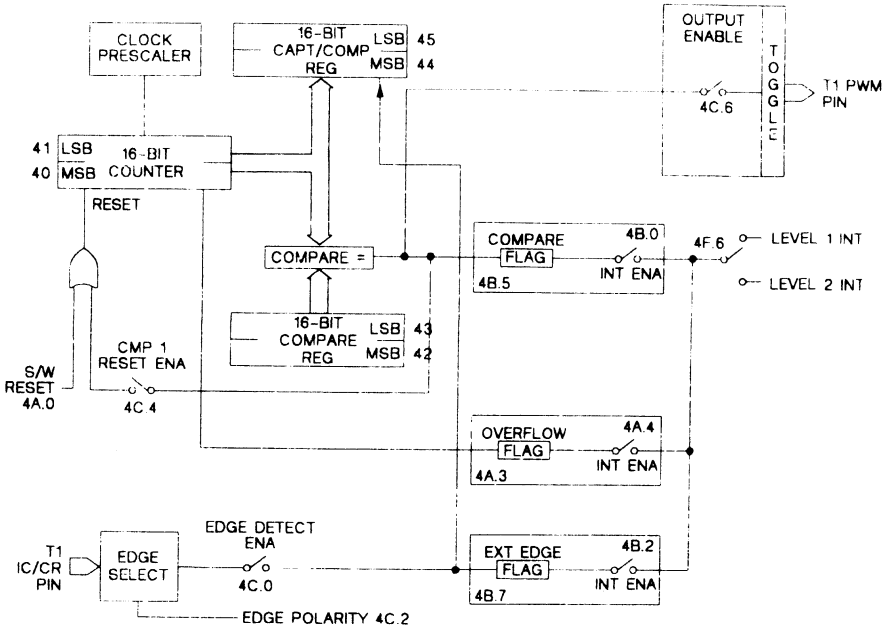


Figure A–3. Timer 1 – Capture/Compare Mode



Appendix B

Timer 2 Control Registers

Timer 2 is controlled and accessed through registers in the Peripheral File. These registers are listed in Figure B–1 and described in the *TMS370 Family Data Manual* (Section 8). The bits shown in shaded boxes in Figure B–1 are Privilege Mode bits, that is, they can only be written to in the Privilege Mode. Figures B–2 and B–3 illustrate the T2 Operational Mode block diagrams.

Figure B–1. Timer 2 Control Registers

Peripheral File Frame 6: Timer 2 Module Control Registers

PF	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Name
P060	Bit 15	T2 Counter MSB						Bit 8	T2CNTR
P061	Bit 7	T2 Counter LSB						Bit 0	
P062	Bit 15	T2 Compare 1 Register MSB						Bit 8	T2C
P063	Bit 7	T2 Compare 1 Register LSB						Bit 0	
P064	Bit 15	Capture 1/Compare 2 Register MSB						Bit 8	T2CC
P065	Bit 7	Capture 1/Compare 2 Register LSB						Bit 0	
P066	Bit 15	Capture Register 2 MSB						Bit 8	T2IC
P067	Bit 7	Capture Register 2 LSB						Bit 0	
P068 P069		Reserved							
P06A	—	—	—	T2 OVRFL INT ENA	T2 OVRFL INT FLAG	T2 Input Select 1	T2 Input Select 0	T2 SW Reset	T2CTL1

Mode: Dual Compare Registers

P06B	T2EDGE1 INT FLAG	T2C2 INT FLAG	T2C1 INT FLAG	—	—	T2EDGE1 INT ENA	T2C2 INT ENA	T2C1 INT ENA	T2CTL2
P06C	T2MODE = 0	T2C1 OUT ENA	T2C2 OUT ENA	T2C1 RST ENA	T2EDGE1 OUT ENA	T2EDGE1 POLAR-ITY	T2EDGE1 RST ENA	T2EDGE1 DET ENA	T2CTL3

Mode: Dual Compare Registers

P06B	T2EDGE1 INT FLAG	T2EDGE2 INT FLAG	T2C1 INT FLAG	—	—	T2EDGE1 INT ENA	T2EDGE2 INT ENA	T2C1 INT ENA	T2CTL2
P06C	T2MODE = 1	—	—	T2C1 RST ENA	T2EDGE2 POLAR-ITY	T2EDGE1 POLAR-ITY	T2EDGE2 DET ENA	T2EDGE1 DET ENA	T2CTL3
P06D	—	—	—	—	T2EVT Data In	T2EVT Data Out	T2EVT Function	T2EVT Data Dir	T2PC1
P06E	T2IC2/ PWM Data In	T2IC2/ PWM Data Out	T2IC2/ PWM Function	T2IC2/ PWM Data Dir	T2IC1/ CR Data In	T2IC1/ CR Data Out	T2IC1/ CR Function	T2IC1/ CR Data Dir	T2PC2
P06F	T2 STEST	T2 Priority	—	—	—	—	—	—	T2PRI

Figure B–2. Timer 2 – Dual Compare Mode

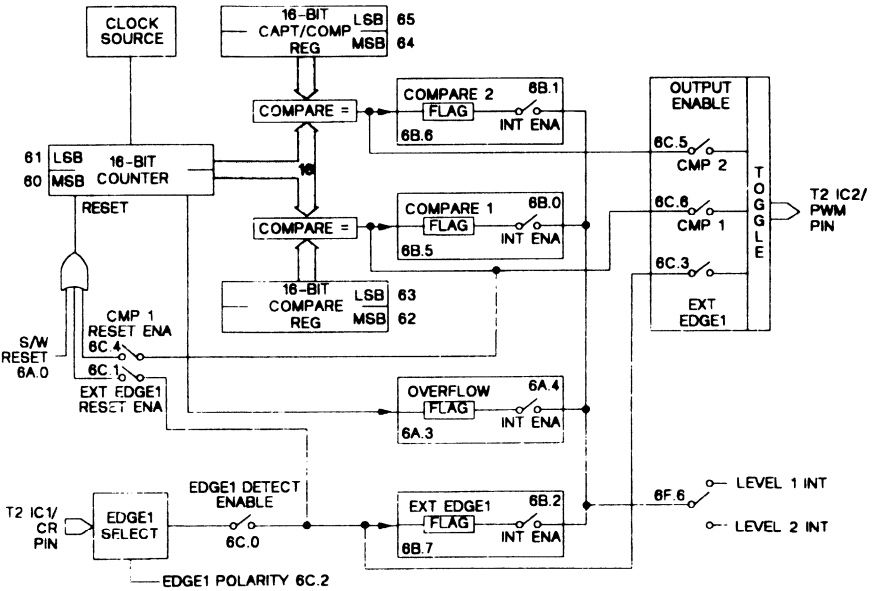
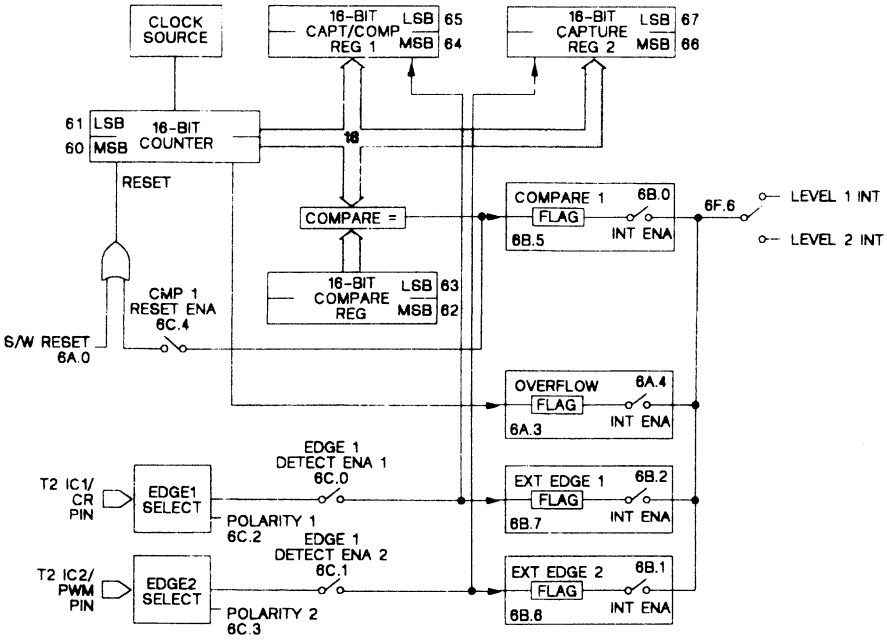


Figure B–3. Timer 2 – Dual Capture Mode



Appendix C

References

Linear and Interface Circuits Applications, literature number SLYA003, Texas Instruments Incorporated, 1987.

TMS370 Family Data Manual, literature number SPNS014, Texas Instruments Incorporated, 1988.

Using the TMS370 SPI and SCI Modules Application Report, literature number SPNA006, Texas Instruments Incorporated, 1988.

Appendix D

Glossary

C

Capture register: A Timer 1 or Timer 2 register which is loaded with the 16-bit counter value on the occurrence of an external input transition. Either edge of the external input can be configured to trigger the capture.

CLKIN: The External Oscillator Frequency (20 MHz maximum)

Compare register: The compare register, in the Timer 1 or Timer 2 module, contains a value which is compared to the counter value. The compare function triggers when the counter matches the contents of the compare register.

E

edge detection: Edge detection circuitry senses an active pulse transition on a given timer input and provides appropriate output transitions to the rest of the module. The active transition can be configured to be low-to-high or high-to-low.

event count: A Timer 1 or Timer 2 clock source option where the timer is clocked from the rising edge of a signal on an external pin (T1EVT or T2EVT).

EEPROM: Electrically Erasable Programmable Read Only Memory; has the capability to be programmed and erased under direct program control.

I

Interrupts: A signal input to the CPU to stop the flow of a program and force the CPU to execute instructions at an address corresponding to the source of the interrupt. When the interrupt is finished, the CPU resumes execution at the point where it was interrupted.

P

PPM: Pulse Position Modulation; a serial signal in which the information is contained in the frequency of a signal with a constant pulse width. A

TMS370 device can output a PPM signal with a constant duty cycle without any program intervention using the Timer 1 or Timer 2 compare registers.

prescale: Circuitry in the Timer 1 module which effectively divides the SYSCLK by a set value. i.e., /64 prescale divides the SYSCLK signal by 64 (SYSCLK = 4/CLKIN).

pulse accumulation: A Timer 1 mode which keeps a cumulative count of SYSCLK pulses as long as the T1EVT pin is high.

PWM: Pulse Width Modulation; a serial signal in which the information is contained in the width of a pulse of a constant frequency signal. A TMS370 device can output a PWM signal with a constant duty cycle without any program intervention using the Timer 1 or Timer 2 compare features.

S

SPI Module: Serial Peripheral Interface module; used to send serial data in a simple bit–format to devices such as shift registers.

SYSCLK: The Internal System Clock Period. SYSCLK = 4/CLKIN.

W

Watchdog Timer: A free–running counter in the Timer 1 module which must be cleared by the program at a set interval. If the program gets lost or is not working properly the counter will overflow causing a system reset.