Pulse Width Modulation Using HPC

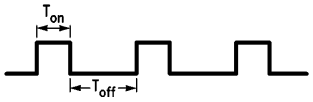
National Semiconductor
Application Note 586
Alvin Chan, Bill Miller,
Bob Moeckel, Bob Hanrahan
April 1989

As the use of MicroControllers in embedded control applications grows in popularity, we find more use of width modulated pulse trains. Typical applications that use Pulse Width Modulation are automotive engine control, motor speed control, display intensity control, and sound generation.

PWM DEFINITION

Pulse width modulation is simply a method of communicating information to a device. It can be viewed as an analog signal provided in digital form. *Figure 1* shows a typical timing diagram of a PWM signal. The duty cycle is expressed as the duration of T_{on} over the sum of T_{on} and T_{off}. A signal has a constant duty cycle if T_{on} and T_{off} are uniform. If T_{on} is equal to T_{off}, the signal has a 50% duty cycle.

$$\text{Duty Cycle} = \frac{T_{on}}{T_{on} + T_{off}}$$

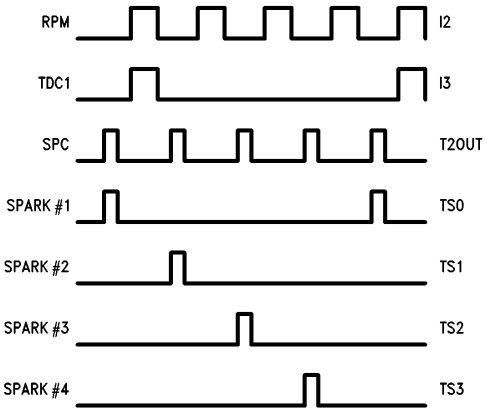


TL/DD/10347–1

FIGURE 1. A Typical PWM Signal

TYPICAL APPLICATIONS THAT REQUIRE PWM

One element of an automotive engine control system is the spark ignition. In a distributorless ignition system, spark control signals are required to appear in sequence, with a time delay between each of them. Typical signals for a four spark plug system are shown in *Figure 2*. The generation of these signals will be explained further in the timer synchronous output section.

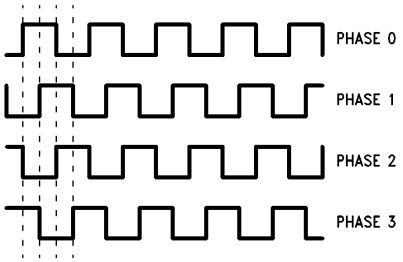


TL/DD/10347–2

FIGURE 2. HPC Based Spark Ignition Control

Another element of an automotive system is the carburetion and idle speed control. When no pressure is applied to the accelerator pedal, the throttle is completely shut off. The idle speed control utilizes a stepping motor to operate an auxilliary fuel valve. *Figure 3* shows the control signals that have to be generated for a four phase stepper motor. Each of the PWM signals should have a phase lag of one quarter of a cycle from the previous one.

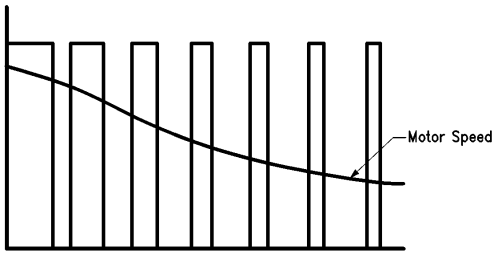
PWM is applied to motor speed control. The speed of a dc motor is directly proportional to the voltage applied.



TL/DD/10347–3

FIGURE 3. Stepper Motor Control Signals

PWM is used selectively to switch full supply power on and off to the motor at some frequency and duty cycle. The bigger the duty cycle, the more power is supplied to the motor. Hence the speed is higher. Motor speed can be controlled by adjusting the ON time of the signal. *Figure 4* depicts the relationship of motor speed and the applied signal.



TL/DD/10347–4

FIGURE 4. Using PWM to Control Motor Speed

The same manipulation also applies to controlling the intensity of light emitting diodes. The brightness of the LED can be varied by using different duty cycles.

Sound synthesis can be achieved by uniting the process of sinusoidal signal generation and envelope generation.

A sinusoidal signal can be generated by a variety of methods. A common technique is to use Walsh functions. Walsh functions are the digital equivalent of Fourier Series. They are essentially pulse signals with varying duty cycles. The individual Walsh components are generated by the microcontroller and combined with the proper weighting factors to form the sinusoids.

RRD-B30M75/Printed in U. S. A.

Envelope generation can be done by using PWM to build a D/A converter. The envelope will give the composite sinusoidal signal the characteristic sharp attack followed by slow decay. The amplitude of the envelope function is altered by changing the duty cycle of the PWM input to the D/A converter. This function is performed by another timer.

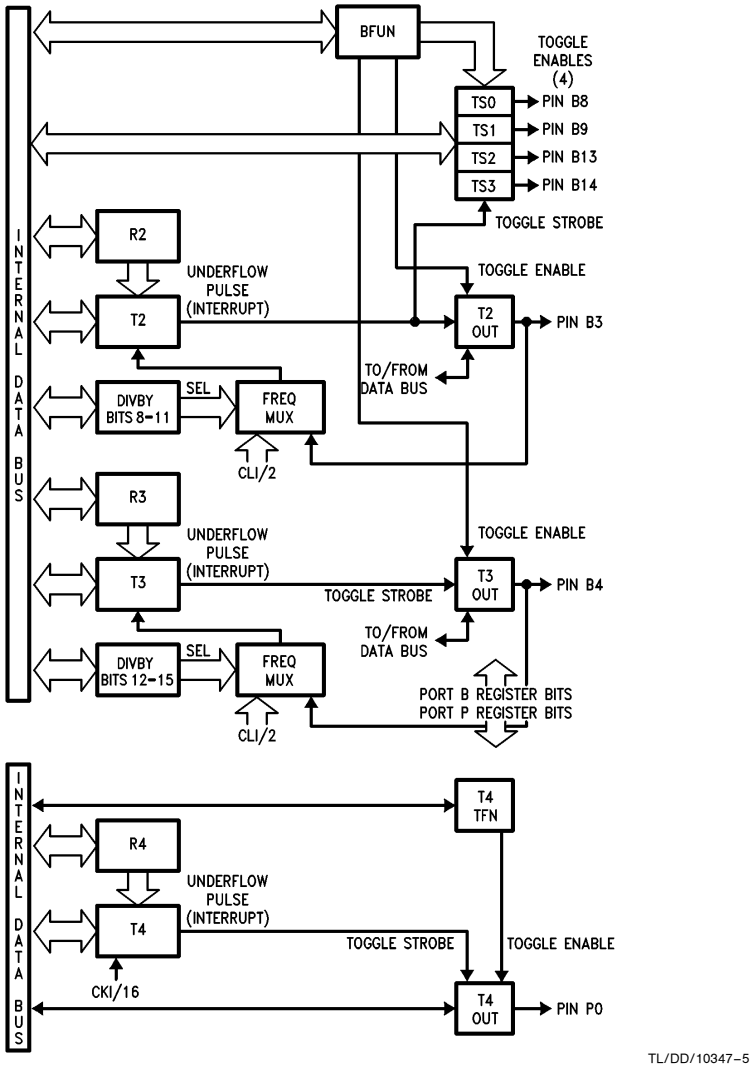
HPC Implementation

National Semiconductor's HPC, High Performance MicroController, provides a simple method for generating width modulated pulse trains, with little or no software overhead, by use of the device's 9 on-chip timers, T0 through T8.

SETTING UP HPC TO DO PWM

PWM Outputs in the HPC

Timers T1 through T7 are down-counters with associated input registers R1 through R7. The value in the registers is loaded automatically into the timers when the timers underflow. Timers T2 through T7 have individual output signals which toggle when the timers underflow. Interrupts are generated at the time of underflow *Figure 5* shows the structure of these timers.



TL/DD/10347–5

Note: Only Time 4 is Shown. T5, T6, and T7 are identical.

FIGURE 5. HPC Timers T2–T7

Timers T2 through T7 can be separated into 2 groups. Different procedures and registers are used to set up the two groups of timers. In one group is timers T4 through T7, which are dedicated to PWM applications. They count down at a constant rate of $1/16$ of the input clock (CKI/16) while enabled to do so. In the other group are the more versatile timers, T2 and T3. The clock input to timers T2 and T3 may be independently selected as coming from one of 14 available prescaled versions of the CKI clock, or from an external pin, as specified in the DIVBY register. Timer T2 can also be specified to be clocked on underflows from timer T3 by appropriate selection in the DIVBY register; the pair then form, in effect, a single 32-bit counter.

With timers T4 through T7, the maximum PWM frequency that can be achieved is half of CKI/16. The associated register provides a 16-bit resolution for the duration of the pulse width.

To use T2 and T3 as PWM timers, the clock must come from an internal source. By configuring the DIVBY register and selecting a value for the counter, the maximum frequency that can be achieved is half CKI/16 and the minimum frequency is half (CKI/131072)/65536.

50% Duty Cycle PWM

On underflow of the timers T2 through T7, the value in the corresponding input register is automatically reloaded into the counters. Therefore a 50% duty cycle PWM can be generated without software intervention once the timer is set up.

Listings 1 and 2 illustrate the use of T4 and T2 in generating PWM outputs. The PWM frequency to be generated is 20 kHz. By using a 16 MHz crystal and CKI/16 as the input clock, the counter value to be loaded into the registers is 24 so that an underflow occurs and the output toggles every 25 μs.

Generating Non 50% Duty Cycle PWM without Listing 1 Use of T4 to Generate 50% Duty Cycle

```
          .TITLE  'T4 PWM 50% DC'
          .SECT   CODE,ROM16,REL
TMMODE    =       0190:W
DIVBY     =       018E:W
T4        =       0140:W
R4        =       0142:W
T5        =       0144:W
R5        =       0146:W
PWMODE    =       0150:W
PORTP     =       0152:W
PWMSTR:   LD      SP,#STKS      ;initialize stack pointer
          LD      PWMODE,#0x4   ;stop timer T4
          NOP                   ;delay to provide 8
                                ;CK2 cycles
          NOP                   ;to make sure timer
                                ;is updated
          LD      PWMODE,#0xC     ;clear T4
                                ;interrupt pending bit
          LD      T4,#24        ;load T4 with counter
                  ;value to obtain a 20 kHz PWM
                  ;frequency the counter should
                  ;underflow on a 40 kHz frequency,
                  ;therefore by using a 16 MHz crystal
                  ;and CKI/16 input to the timer, the
                  ;counter value should be 24
                  ;16 MHz/16/25 = 40 kHz
          LD      R4,#24        ;load auto-reload
                                ;register
          SBIT    0,PORTP       ;set initial value of
                                ;output pin for T4 to 0
          SBIT    3,PORTP       ;enable toggling of
                                ;pin on underflow
          RBIT    2,PWMODE      ;start timer
STOP:
          JP      STOP
          .ENDSECT

          .SECT   STACK,BASE
STKS:     DSW     10
          .ENDSECT
          .END    PWMSTR
```

Non 50% Duty Cycle PWM (Software/Interrupts)

Timers T1 through T7 will generate an interrupt on underflow. For non-50% duty cycle PWM, software has to be involved in controlling the duty cycle. The same software for the 50% duty cycle is used to set up the timers for counting down. On interrupt from the timers, the interrupt service routine loads the other half of the cycle time into the timer register.

On each interrupt from the timer the user software alternately loads T_{on} and T_{off} into the register. The result is a constant duty cycle output. Examples of programming the interrupts are shown in listings 3 and 4.

TIMER SYNCHRONOUS OUTPUTS OF TIMER T2

Timer T2 has in addition to the normal output pin, four output pins which can be independently selected. These pins are referred to collectively as the ''Timer Synchronous'' outputs. *Figure 2* shows the synchronous output being applied to engine control in spark ignition. The signals TS0 to TS3 are synchronous outputs derived from timer T2. By enabling each pin in sequence, the spark control signals SP1 to SP4 can be generated.

SOFTWARE INTERVENTION

Another problem facing the designer of a MicroController based system is that software overhead must be kept to a minimum. Interrupt latency and changing input registers can use a significant portion of the time which would otherwise be available for processing of sensor data.

The conventional way of generating non-50% duty cycle was discussed earlier. That involves software changing the value of the auto-reload register every time the timer counts down and interrupts. Two timers can be used to generate two synchronized and offset 50% duty cycle pulses. By EXCLUSIVE-ORing them, a non-50% duty cycle PWM is generated.

Listing 2 Use of T2 to Generate 50% Duty Cycle

```
          .TITLE 'T2 PWM 50% DC'
          .SECT  CODE,ROM16,REL
TMMODE    =      0190:W
DIVBY     =      018E:W
BFUN      =      00F4:W
DIRB      =      00F2:W
PORTB     =      00E2:W
T2        =      0188:W
R2        =      0186:W
PWMSTR:   LD     SP,#STKS     ;initialize stack pointer
          LD     TMMODE,#0x400  ;stop timer T2
          NOP                 ;delay to provide 8
          NOP                 ;CK2 cycles to make
                              ;sure timer is updated
          LD     TMMODE,#0xC00  ;clear T2
                              ;interrupt pending bit
          SBIT   3,BFUN       ;set pin 3 of port B
                              ;as timer 2 output
          SBIT   3,DIRB       ;set output direction
                              ;on port B pin 3
          LD     DIVBY,#0x200;set clock as CKI/16
                              ;for T2
          LD     T2,#24       ;load T2 with counter
                 ;value to obtain a 20 kHz PWM
                 ;frequency the counter should
                 ;underflow on a 40 kHz frequency
                 ;therefore by using a 16 MHz crystal
                 ;and CKI/16 input to the timer, the
                 ;counter value should be 24
                 ;16 MHz/16/25 = 40 kHz
          LD     R2,#24       ;load auto-reload
                              ;register
          RBIT   3,PORTB      ;initialize output pin
                              ;value to 0
          RBIT   3,TMMODE     ;start timer
STOP:
          JP     STOP
          .ENDSECT

          .SECT  STACK,BASE
STKS:     DSW    10
          .ENDSECT
          .END   PWMSTR
```

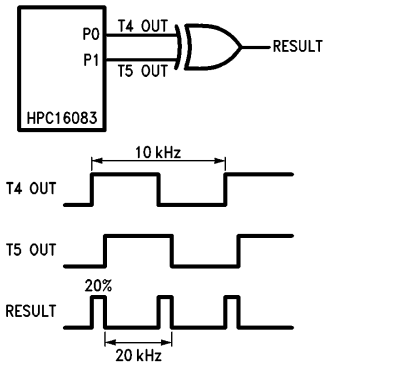
Listing 3 Use of T4 to Generate Non-50% Duty Cycle with Interrupts

```
          .TITLE 'T4 NON-50% DC'
          .SECT  CODE,ROM16,REL
TMMODE    =      0190:W
DIVBY     =      018E:W
T4        =      0140:W
R4        =      0142:W
T5        =      0144:W
R5        =      0146:W
PWMODE    =      0150:W
PORTP     =      0152:W
ENIR      =      00D0:B
IRPD      =      00D2:B
PWMSTR:   LD     SP,#STKS   ;initialize stack pointer
          LD     PWMODE,#0x4;stop timer T4
          NOP               ;delay to provide 8
          NOP               ;CK2 cycles to make
                            ;sure timer is updated
          LD     PWMODE,#0xC   ;clear T4
                            ;interrupt pending bit
          LD     ENIR,#00   ;disable interrupts
          LD     IRPD,#00   ;clear interrupt
                            ;pending bits
          LD     T4,#9      ;load T4 with counter
                ;value to obtain a 20 kHz PWM
                ;frequency with 20% duty cycle
                ;using a 16 MHz crystal and CKI/16
                ;input to the timer, the counter
                ;value should be 9
          LD    R4,#39           ;load auto-reload
                                 ;register with count
                                 ;of 80%
          LD    TCYCLE,#48       ;set total cycle time
                                 ;count -2
          SBIT  0,PORTP          ;set initial value of
                                 ;output pin for T4 to 0
          SBIT  3,PORTP          ;enable toggling of
                                 ;pin on underflow
          LD    ENIR,#0x20       ;enable timer interrupt
          RBIT  2,PWMODE         ;start timer
STOP:
          JP    STOP
          .ENDSECT

          .SECT STACK,BASE
STKS:     DSW   10
          .ENDSECT

          .IPT  5,INTRPT5

          .SECT DATA,BASE,REL
TCYCLE:   .DSW  1                ;total cycle time
          .ENDSECT

          .SECT SUBR,ROM 16,REL
INTRPT5:
          LD    A,TCYCLE         ;get total cycle time
          SC
          SUBC  A,R4             ;subtract current
                                 ;counter
          ST    A,R4             ;to get alternate cycle
                                 ;time and store to
                                 ;auto-reload reg
          RETI
          .ENDSECT
          .END  PWMSTR
```

Listing 4 Use of T2 to Generate Non-50% Duty Cycle with Interrupts

```
          .TITLE 'T2 NON-50% DC'
          .SECT  CODE,ROM16,REL
TMMODE    =      0190:W
DIVBY     =      018E:W
BFUN      =      00F4:W
DIRB      =      00F2:W
PORTB     =      00E2:W
T2        =      0188:W
R2        =      0186:W
ENIR      =      00D0:B
IRPD      =      00D2:B
PWMSTR:   LD     SP,#STKS     ;initialize stack pointer
          LD     TMMODE,#0x400  ;stop timer T2
          NOP    ;delay to provide 8 CK2 cycles
          NOP    ;to make sure timer is updated
          LD     TMMODE,#0xC00  ;clear T2
          LD     ENIR,#00     ;interrupt pending bit
                              ;disable interrupts
          LD     IRPD,#00     ;clear interrupt
                              ;pending bits
          SBIT   3,BFUN       ;set pin 3 of port B
                              ;as timer 2 output
          SBIT   3,DIRB       ;set output direction
                              ;on port B pin 3
          LD     DIVBY,#0x200;set clock as CKI/16
                              ;for T2
          LD     T2,#9        ;load T2 with counter
                 ;value to obtain a 20 kHz PWM
                 ;frequency using a 16 MHz crystal
                 ;and CKI/16 input to the timer, the
                 ;counter value should be 9
          LD     R2,#39       ;load auto-reload
                              ;register with count
                              ;of 80%
          LD     TCYCLE,#48   ;set total cycle time
                              ;count -2
          RBIT   3,PORTB      ;initialize output pin
                              ;value to 0
          LD     ENIR,#0x20   ;enable timer interrupt
          RBIT   3,TMMODE     ;start timer
STOP:
          JP     STOP
          .ENDSECT

          .SECT STACK,BASE
STKS:     .DSW   10
          .ENDSECT

          .IPT   5,INTRPT5

          .SECT  DATA,BASE,REL
TCYCLE:   .DSW   1            ;total cycle time
          .ENDSECT

          .SECT  SUBR,ROM16,REL
INTRPT5:
          LD     A,TCYCLE     ;get total cycle time
          SC
          SUBC   A,R2         ;subtract current
                              ;counter
          ST     A,R2         ;to get alternate cycle
                              ;time and store to
                              ;auto-reload reg
          RETI
          .ENDSECT
          .END   PWMSTR
```

Figure 6 shows the result of EXCLUSIVE-ORing the two timers. The duty cycle depends only on the phase shift between the timer outputs. In can be seen that the resulting frequency is actually twice the frequency of the original timers. Therefore, in order to generate a 20 kHz result, two 10 kHz timers must be used. The code is shown in listing 5. By varying the initial delay in the second timer, different duty cycles can be chosen. In the example given, a one digit difference in the counter value results in a 2% difference in the duty cycle.



TL/DD/10347–6

FIGURE 6. Using 2 Timers to Generate Non 50% Duty Cycle

Listing 5 Use of T4, T5 to Generate Non-50% Duty Cycle without Interrupts

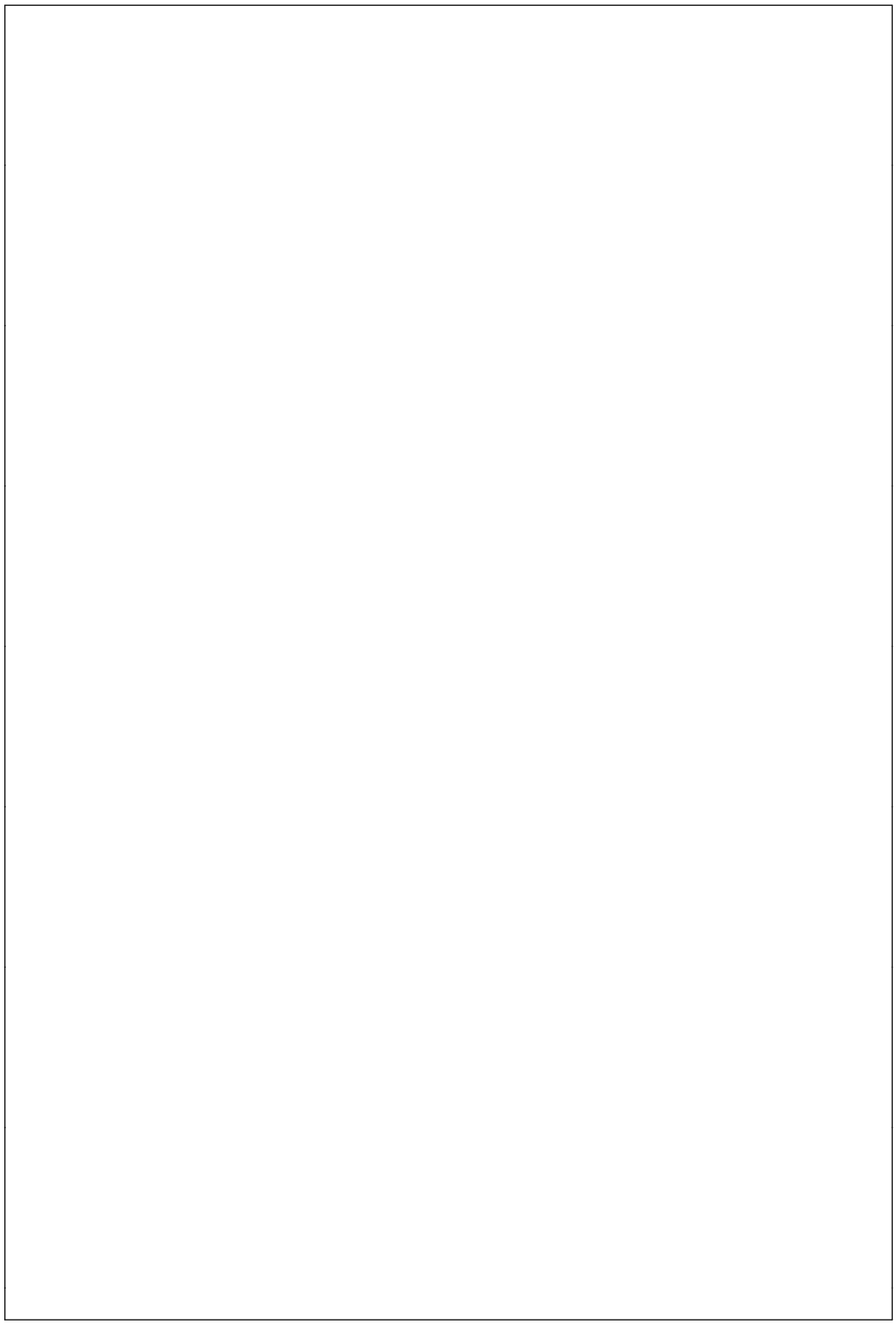
```
          .TITLE 'NON 50% PWM (T4,T5)'
          .SECT  CODE,ROM16,REL
TMMODE    =      0190:W
DIVBY     =      018E:W
T4        =      0140:W
R4        =      0142:W
T5        =      0144:W
R5        =      0146:W
PWMODE    =      0150:W
PORTP     =      0152:W
FREQ:     .DW    49            ;counter value for the timers
                                ;this generates 10 kHz PWM
DC:       .DW    20            ;duty cycle = 20%
PWMSTR:   LD     SP,#STKS
          LD     PWMODE,#0X44     ;stop T4 and
                                  ;T5
          NOP
          NOP
          LD     PWMODE,#0XCC     ;clear T4, T5
                                ;int pending bits
          LD     T4,FREQ       ;load T4, R4, R5
          LD     R4,FREQ       ;with counter value
          LD     R5,FREQ
          LD     A,DC          ;calculate delay for
          MULT   A,FREQ           ;T5
          DIV    A,#100
          ST     A,T5          ;store in T5
          LD     PORTP,#0X10   ;set output pins, T4
                                ;low T5 high
          SBIT   3,PORTP       ;enable toggling of
          SBIT   7,PORTP       ;pins on underflow
          AND    PWMODE,#0XFFBB   ;start T4 and
                                  ;T5
STOP:
          JP     STOP
          .ENDSECT

          .SECT  STACK,BASE
STKS:     .DSW   10
          .ENDSECT
          .END   PWMSTR
```

Listing 6 Use of T2, T3 to Generate Non-50% Duty Cycle without Interrupts

```
          .TITLE 'NON 50% PWM (T2,T3)'
          .SECT  CODE,ROM16,REL
BFUN      =      00F4:W
DIRB      =      00F2:W
PORTB     =      00E2:W
TMMODE    =      0190:W
DIVBY     =      018E:W
T4        =      0140:W
R4        =      0142:W
T5        =      0144:W
R5        =      0146:W
PWMODE    =      0150:W
PORTP     =      0152:W
T2        =      0188:W
R2        =      0186:W
R3        =      018A:W
FREQ:     .DW    49    ;counter value for the timers
                       ;this generates 10 kHz PWM
DC:       .DW    20    ;duty cycle = 20%
PWMSTR:   LD     SP,#STKS
          LD     TMMODE,#0x4400    ;stop T2,T3
          NOP
          NOP
          LD     TMMODE,#0xCCC8    ;clear T2, T3
                                   ;int pending bits
          LD     PORTB,#0x10       ;set output pins, T2
                                   ;low T3 high
          LD     DIRB,#0xFFFF      ;output on PORT B
          OR     BFUN,#0x0018      ;set T2,3 as timers
          OR     DIVBY,#0x2200     ;select CKI/16 clock
          LD     T2,FREQ           ;load T2 R2, R3 with
                                   ;counter value
          LD     R2,FREQ
          LD     R3,FREQ
          LD     A,DC              ;calculate delay for
          MULT   A,FREQ               ;T3
          DIV    A,#100
          ST     A,R3              ;store delay in T3
          AND    TMMODE,#0xBBFF       ;start T2,T3
          STOP:
          JP     STOP
          .ENDSECT

          .SECT  STACK,BASE
STKS:     .DSW   10
          .ENDSECT
          .END   PWMSTR
```

CONCLUSION

PWM is easily generated by the HPC 16083 with its abundant source of timers. With a 30 MHz crystal, the maximum PWM frequency that can be achieved is 937.5 kHz. The timers run by themselves once the proper setup is performed. A method of obtaining non-50% duty cycle PWM without software intervention was presented.



Lit. # 100586

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, CA 95052-8090
Tel: 1(800) 272-9959
TWX: (910) 339-9240

National Semiconductor GmbH
Livry-Gargan-Str. 10
D-82256 Fürstenfeldbruck
Germany
Tel: (81-41) 35-0
Telex: 527649
Fax: (81-41) 35-1

National Semiconductor Japan Ltd.
Sumitomo Chemical
Engineering Center
Bldg. 7F
1-7-1, Nakase, Mihama-Ku
Chiba-City,
Ciba Prefecture 261
Tel: (043) 299-2300
Fax: (043) 299-2500

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductores Do Brazil Ltda.
Rue Deputado Lacorda Franco
120-3A
Sao Paulo-SP
Brazil 05418-000
Tel: (55-11) 212-5066
Telex: 391-1131931 NSBR BR
Fax: (55-11) 212-1181

National Semiconductor (Australia) Pty, Ltd.
Building 16
Business Park Drive
Monash Business Park
Nottinghill, Melbourne
Victoria 3168 Australia
Tel: (3) 558-9999
Fax: (3) 558-9998