AN1734

Pulse Width Modulation Using the 16-Bit Timer

**By Brad Bierschenk and Allan Jones
Applications Engineering
Austin, Texas**

Introduction

This application note presents a method of implementing pulse width modulation (PWM) in Motorola microcontrollers through the output compare function of the 16-bit free-running timer counter. Using this method, a PWM signal can be produced without dedicated on-chip PWM circuitry.

PWM is a technique used to control devices or to provide a variable DC voltage. Common applications include motor, lighting, and climate controls. If the production of a PWM signal is not a vital part of an application, the added cost and complexity of dedicated PWM hardware in a microcontroller might not be justified. In this case it would be desirable to implement PWM output through software control of a common MCU module.

By using the 16-bit timer counter and its output compare function, a PWM of desired duty cycle and frequency can be quickly and easily implemented. This method uses modules common in most Motorola microcontrollers and requires only a small amount of processing overhead.

Freescale Semiconductor, Inc.

PWM

A PWM signal is a signal with a fixed frequency and variable on and off times. In other words, the period of the signal remains constant, but the amount of time that the signal remains high and low can vary within a period. The duty cycle is the ratio of on time (t_{On}) to the total period ($t = t_{On} + t_{Off}$).

Figure 1 illustrates a square wave. When viewed as a PWM signal, its duty cycle is 50%. In other words, it is on half of the time.
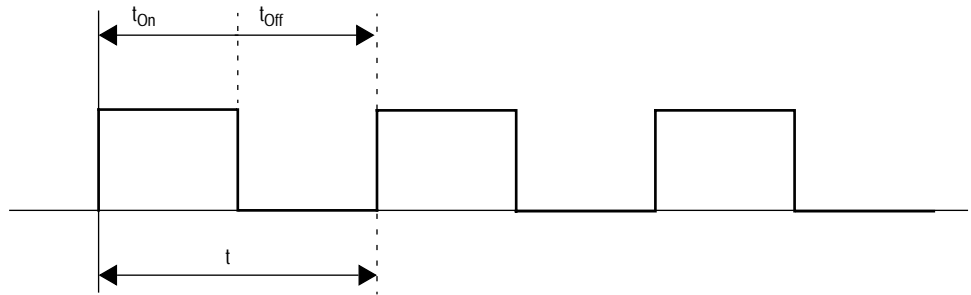

Figure 1. PWM Signal with 50% Duty Cycle

Figure 2 illustrates another PWM signal. Its duty cycle is 10%. By varying the duty cycle, the average DC voltage output can be controlled. For example, a PWM signal that has a 10-V amplitude and a 50% duty cycle can provide an average 5-V output. When increasing or decreasing a PWM duty cycle, the average output increases or decreases respectively.
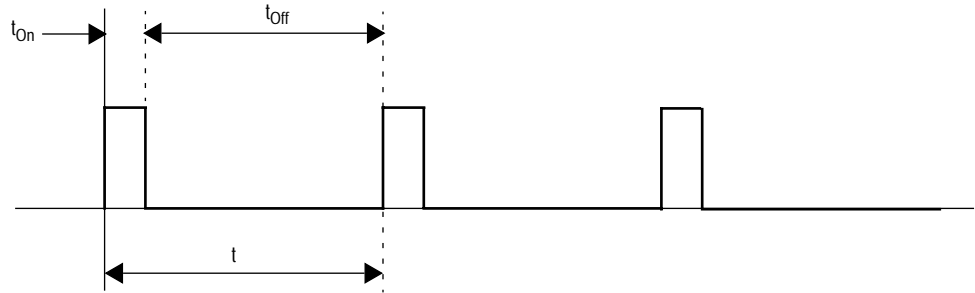
AN1734

2

Figure 2. PWM Signal with 10% Duty Cycle

This controllable output is useful in many applications. For example, this output could be used to control the speed of a motor. By increasing the duty cycle, the average voltage across a motor can be controlled. The output could also be fed through an RC network, forming a simple digital-to-analog converter.

Output Compare Method

One effective method of producing a PWM signal in a Motorola microcontroller involves the use of the 16-bit free-running timer. This method does not require dedicated PWM circuitry and can be implemented across a wide variety of Motorola MCUs.

The 16-bit timer and its output compare function can be used to produce a specific output level at a particular time. When the output of the timer compare (TCMP) pin is alternated between low and high logic levels, the result is a periodic waveform. Values in the output compare register are used to cause a delay between output events. Specifically, t_{On} and t_{Off} of a desired duty cycle can be calculated and programmed into the output compare function.

The basic steps in this procedure are:

1. Compute a timer delay value, add it to the current counter value, and place the result in the output compare register.

2. Set or clear the output level bit (OLVL) in the timer control register (TCR). This will determine the TCMP pin's output level for the next valid compare.

3. Set the output compare interrupt enable bit (OCIE) in the TCR to enable output compare interrupts.

4. Clear the interrupt mask.

5. On timer interrupt:

    a. Complement the OLVL bit for next compare.

    b. Compute and store new value for output compare register.

The first step in implementing this procedure involves computing the values needed in the output compare register. Given the desired frequency (f) and duty cycle (DC) of the PWM signal to be generated, delay values needed to create the waveform can be determined.

PWM Definitions

Period: $\quad t = 1/f = t_{On} + t_{Off}$

Duty cycle: $\quad DC = t_{On}/(t_{On}+t_{Off}) = t_{On}/t$

Given f and DC of a PWM signal, find:

Period: $\quad t = 1/f$

On time: $\quad t_{On} = DC * t$

Off time: $\quad t_{Off} = t - t_{On}$

For example, a PWM is to be generated with a frequency of 2 kHz and a duty cycle of 50%. In this case, the period t is 500 μs, and t_{On} and t_{Off} are both 250 μs.

Next, the frequency of the timer subsystem should be calculated. For the HC05 Family of MCUs to find the MCU internal operating frequency, f_{op}, divide the oscillator frequency, f_{osc}, by 2. The timer prescaler divides this frequency further by 4. Therefore, the timer subsystem frequency, f_{tim}, is $f_{osc}/8$.

For example, a 4-MHz oscillator would yield a timer frequency of 500kHz. The timer period would then be 2 μs. In other words, one timer

AN1734

increment takes 2 μs to complete. We will define this time as t_{tim}, the period of the timer system.

Once the timer period, t_{tim}, is found, compute the number of timer cycles required to cause a specific delay. This delay can be used to time an output. In the case of generating a PWM signal, the high and low levels of the output need to be timed. Specifically, the on and off delay values should be calculated. These delays reflect the number of timer cycles necessary to produce a pulse of a specific duration.

The delay values are proportional to the on and off times of the desired PWM signal, and are defined as:

On delay: $D_{On} = t_{On}/t_{tim}$

Off delay: $D_{Off} = t_{Off}/t_{tim}$

Using the previous example of a 4-MHz oscillator, 2-kHz PWM, and 50% duty cycle, D_{On} and D_{Off} would each require a decimal value of 125.

The delay values D_{On} and D_{Off} can be added to the timer counter, and the result will be stored in the output compare register. On a valid compare (when the contents of the counter match the output compare register), an output compare interrupt can be generated. This interrupt can then be used to prepare the MCU for the next pulse segment delay and its associated output level.

In MCUs equipped with output compare (TCMP) pins, the logic level to be output on a valid compare can be preset. In this way, an output compare timer interrupt will set the output level of the TCMP pin immediately. A new value for the output compare register to time the current output level can then be loaded, and the next triggered output level set. In this manner, a highly accurate PWM output can be achieved, as the output level is set automatically on a valid compare.

If a microcontroller does not have a TCMP pin, the same procedure for setting up output compares can be followed. The timer interrupt service routine can be programmed to manually set the PWM output level at the pin of an I/O port. In this case, the software would set the current pulse segment's output level and delay, rather than the next segment's output level and the current segment's delay.

Freescale Semiconductor, Inc.

> **NOTE:** *When using this method, one must consider the latency of entering the interrupt routine when calculating the necessary delays. The time between a valid compare and the manual setting of the output level in this case is not always negligible.*

By controlling D_{On} and D_{Off}, different combinations of PWM duty cycles and frequencies can be achieved. Some applications, such as motor control or lighting, involve control of the duty cycle of a PWM output. Other applications, such as tone generation, require a fixed duty cycle and different frequencies.

This technique of generating a PWM signal is limited by the duration of the timer interrupt routine. Because the timer interrupt computes and loads the compare values needed to produce the desired output, the minimum delay that can be tolerated is the worst-case duration of the software. A pulse cannot be produced if its segment delay time is shorter than the time needed to process the output compare interrupt.

If other tasks are to be performed between timer interrupts, the worst case time requirements of the other tasks should be considered as well. The time to process other software functions may need to be included in the minimum delay.

AN1734

Timing limits

f_{osc} (MHz)	f_{op} (MHz)	t_{op} (µs)	f_{tim} (kHz)	t_{tim} (µs)	Minimum delay (µs) = (interrupt cycles)*t_{op}	Minimum timer change = (interrupt cycles)/4
8	4	0.25	1000	1	13	13
4	2	0.5	500	2	26	13
2	1	1	250	4	52	13
1	0.5	2	125	8	104	13

Duty cycle limits

	Desired PWM frequency					
	1 kHz		**2 kHz**		**5 kHz**	
f_{osc} (MHz)	**Min DC**	**Max DC**	**Min DC**	**Max DC**	**Min DC**	**Max DC**
8	1.3%	98.7%	2.6%	97.4%	6.5%	93.5%
4	2.6%	97.4%	5.2%	94.8%	13.0%	87.0%
2	5.2%	94.8%	10.4%	89.6%	26.0%	74.0%
1	10.4%	89.6%	20.8%	79.2%	52.0%	48.0%

Figure 3. Example Limits for MC68HC705C8A (1 to 4 MHz) and MC68HSC705C8A (8 MHz)

Figure 3 illustrates the limits encountered when using this method of PWM generation. The bus frequency and the interrupt routine latency determine the minimum delay that can be tolerated.

For example, in our sample application the interrupt routine demands 52 bus cycles in the worst case. This makes the minimum tolerable delay time 52 * t_{op}. The minimum delay time applies to both on and off times, therefore affecting the minimum and maximum achievable duty cycles. **Figure 3** shows that higher bus frequencies and lower PWM frequencies provide greater duty cycle control. The minimum duty cycle is the minimum time delay divided by the desired PWM period. The maximum duty cycle results from the off delay being equal to the minimum tolerable delay.

AN1734

Sample Application — Motor Controller

In this example application, we want to produce a PWM signal of a fixed frequency. We want to control the duty cycle with as wide a range as possible. The PWM output will drive a fan motor. **Figure 4** shows the example application circuit using the Motorola MC68HC705P9.

We have attached a potentiometer to the analog-to-digital (A/D) subsystem of the microcontroller. By monitoring the A/D data register, a number between $00 and $FF is obtained. This number is used as the on delay, D_{On}. The main software loop simply polls the A/D subsystem, waiting for new data. To further expand this application, the potentiometer could be replaced with a temperature sensor, creating a temperature-controlled fan.

When a new A/D value is ready, the software determines the difference between the new D_{On} (the ADDR value) and the current D_{On}. If the new A/D value does differ from the D_{On}, it can be determined whether the duty cycle should increase or decrease.

AN1734
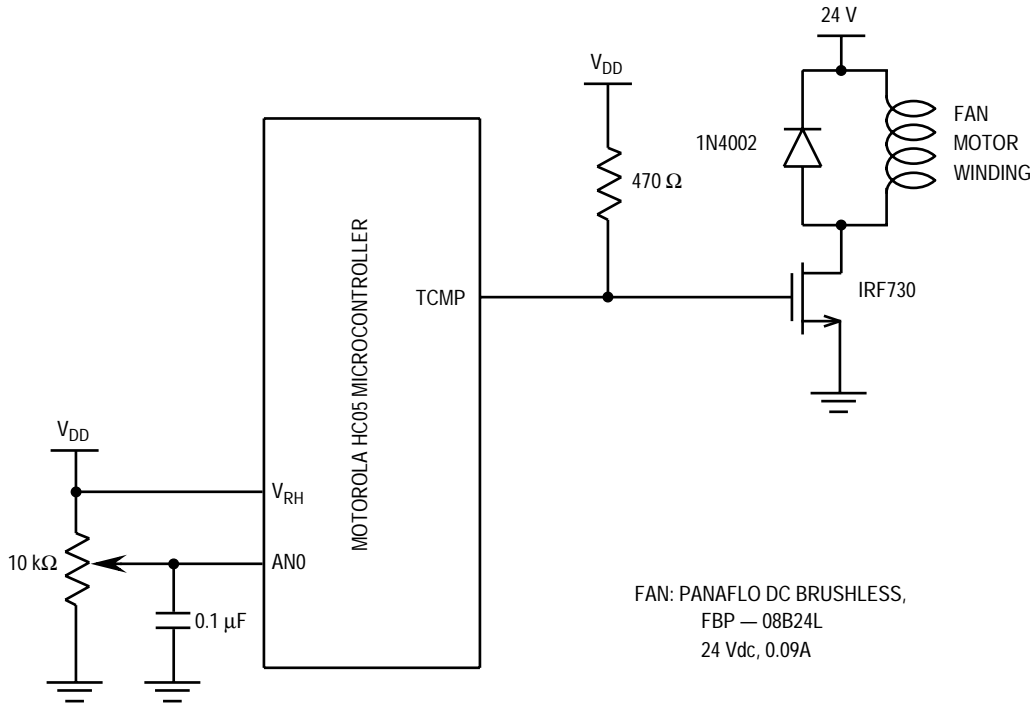
Figure 4. Fan Control Circuit

When the duty cycle is altered through the potentiometer, the new D_{On} and D_{Off} is calculated. However, before these new values are recorded, they must be checked against the upper and lower delay limits. If they exceed these limits, no change is made in the delays.

Because an 8-bit delay value is used, the upper limit on delays is $FF. There is also a lower limit on delays. The time it takes to process the timer output compare interrupt routine determines this limit. Output compare interrupts should not happen faster than they can be serviced.

In this example, the worst case interrupt service time is 52 cycles. With an internal operating frequency of 2 MHz, the time required is 26 μs. Because the 16-bit timer runs at 500 kHz, its period is 2 μs. Therefore, the minimum delay this interrupt routine can tolerate is 13 "ticks" of the free-running timer (13 * 2 μs = 26 μs). Keeping track of this lower limit ensures an accurate and clean output waveform.

Conclusion

There are many other methods of producing PWM outputs using Motorola microcontrollers. In choosing one method over the others, trade-offs must be considered regarding convenience, cost, and precision.

The method discussed in this application note provides a good balance between cost, convenience, and duty cycle control. This method can be implemented across a wide variety of Motorola microcontrollers.

PWM generation can also be implemented entirely in software. This method can be implemented in any microcontroller, but requires a great deal of software overhead. The simultaneous use of the timer overflow interrupt and real-time interrupt as timing references for a PWM signal can also be implemented. This method can produce only a few different duty cycles for any given frequency and cannot be implemented in all Motorola microcontrollers.

For applications which require precise control of multiple PWM signals, Motorola provides parts with dedicated PWM circuitry. Examples include the MC68HC708MP16, MC68HC705MC4, and MC68HC05D9/D24/D32 microcontrollers. For more information on such products, consult the 68HC05/68HC08 selector guide, Motorola document order number SG165/D.

AN1734

Freescale Semiconductor, Inc.

```
           ┌─────────────────────┐
           │   INITIALIZE ADC    │
           │  GET INITIAL VALUE  │
           │ SET UP INITIAL DUTY │
           │        CYCLE        │
           └─────────────────────┘
                     │
           ┌─────────────────────┐
           │        LOOP         │
           └─────────────────────┘
                     │
     NO              ▼
           ◇ NEW ADC DATA READY? ◇
                     │
                    YES                    YES
                     │
           ┌─────────────────────┐
           │   GET NEW VALUE     │
           │   OF ON DELAY       │
           │   FROM AD DATA      │
           └─────────────────────┘
                     │
           ◇ IS NEW VALUE         ◇
             THE SAME AS OLD
             VALUE OF
             ON DELAY?
                     │
                    NO
           ┌─────────────────────┐
           │   SUBTRACT OLD      │
           │   ON DELAY FROM     │
           │   NEW VALUE         │
           └─────────────────────┘
                     │
     YES    ◇ IS RESULT          ◇    NO
             POSITIVE?
             (NEW > OLD)

   ┌──────────────┐                 ┌────────────────┐
   │ CALL DUTY UP │                 │ CALL DUTY DOWN │
   └──────────────┘                 └────────────────┘
```
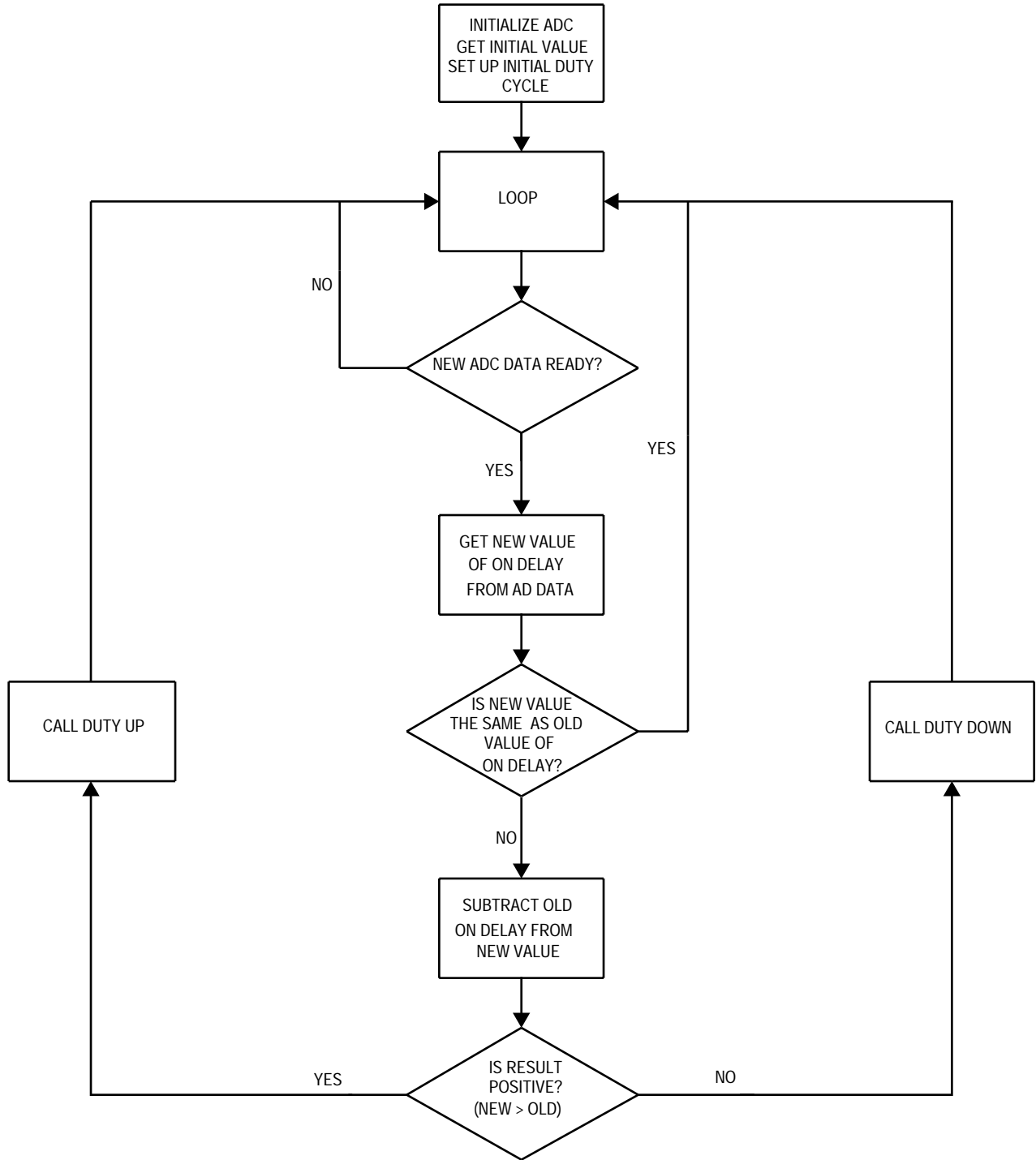
Figure 5. Main Program Flow
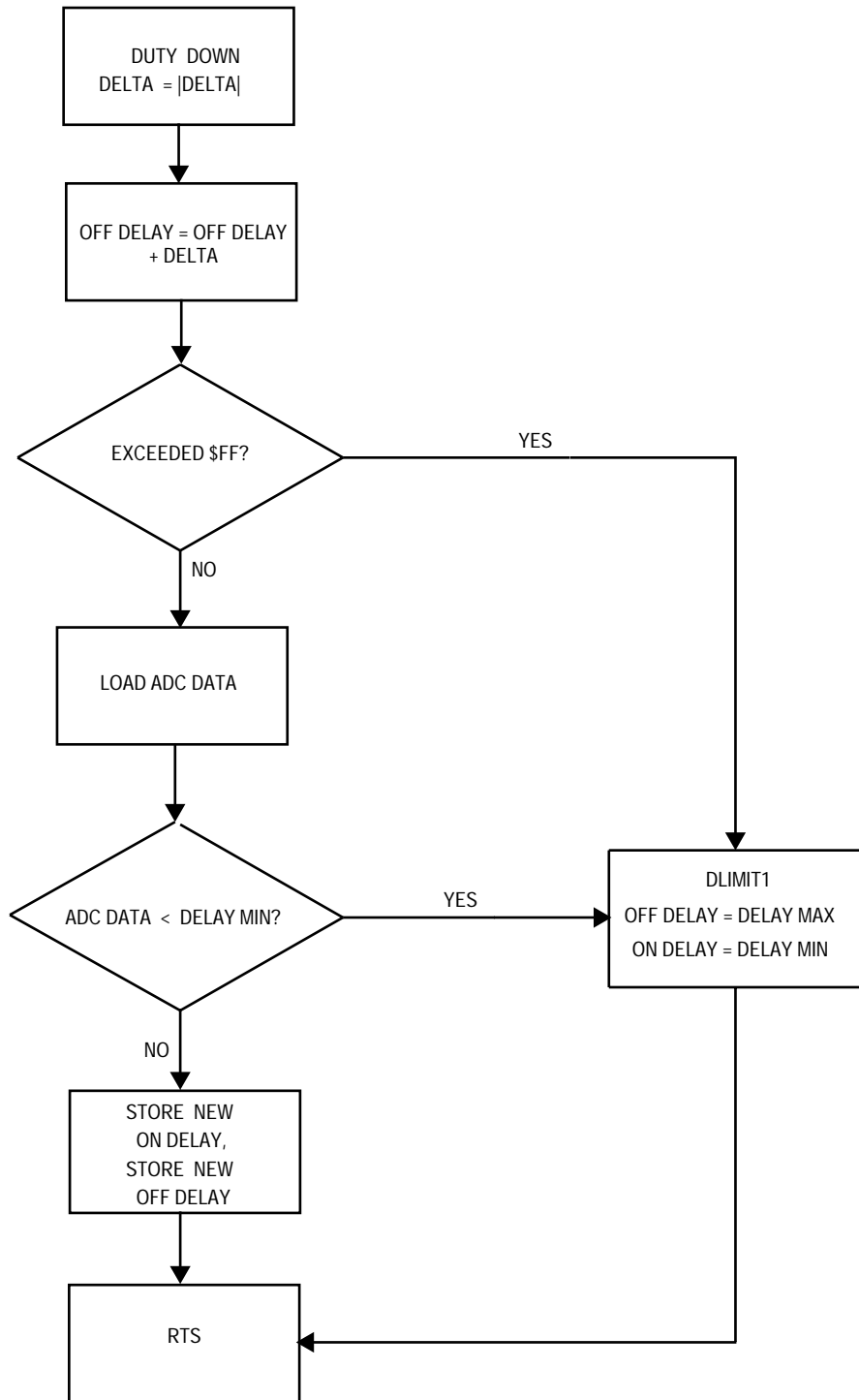
Figure 6. Duty Down Subroutine

Freescale Semiconductor, Inc.

```
                              ┌──────────────┐
                              │   DUTY UP    │
                              └──────┬───────┘
                                     │
                                     ▼
                              ◇ DELTA > OFF DELAY? ◇
            YES ◄─────────────────── │
                                     │ NO
                                     ▼
                              ┌──────────────┐
                              │  OFF DELAY = │
                              │ OFF DELAY – DELTA │
                              └──────┬───────┘
                                     │
                                     ▼
  ┌──────────────────┐        ◇ NEW OFF DELAY < ◇
  │     DLIMIT2      │◄── YES ──  DELAY MIN
  │ ON DELAY = DELAY MAX │
  │ OFF DELAY = DELAY MIN │
  └────────┬─────────┘              │ NO
           │                        ▼
           │                 ┌──────────────┐
           │                 │ RECORD NEW   │
           │                 │  OFF DELAY   │
           │                 └──────┬───────┘
           │                        │
           │                        ▼
           │                 ┌──────────────┐
           │                 │  ON DELAY =  │
           │                 │  ADC DATA    │
           │                 └──────┬───────┘
           │                        │
           │                        ▼
           │                 ┌──────────────┐
           └────────────────►│    RTS       │
                             └──────────────┘
```
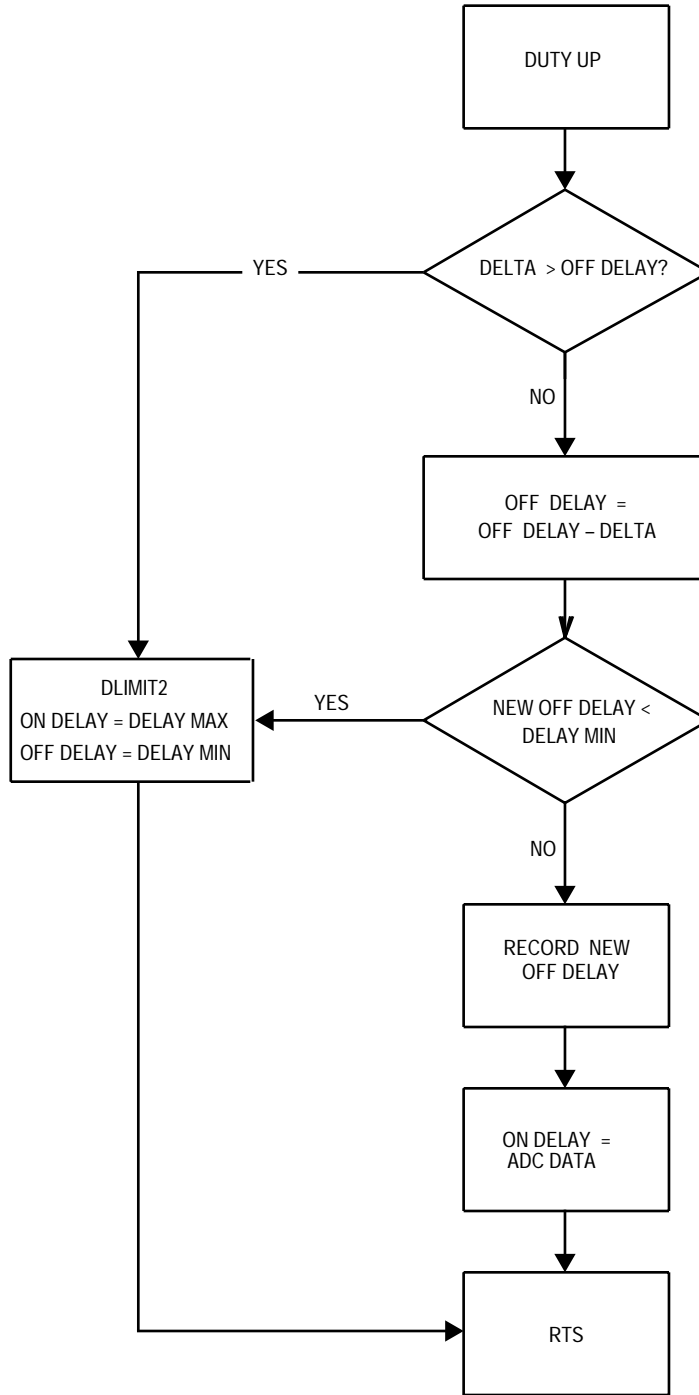
Figure 7. Duty Up Subroutine

Freescale Semiconductor, Inc.

```
          ┌─────────────────────┐
          │      TIMERINT       │
          │   TIMER INTERRUPT   │
          └──────────┬──────────┘
                     │
                     ▼
              ╱─────────────╲
      NO    ╱  IS CURRENT    ╲    YES
◄──────────  OUTPUT LEVEL    ──────────►
      │      ╲ (TCMP) HIGH? ╱      │
      │        ╲──────────╱        │
      ▼                            ▼
┌─────────────┐            ┌─────────────┐
│   GO HIGH   │            │   GO LOW    │
│ SET OLVL BIT│            │CLEAR OLVL BIT│
│FOR NEXT     │            │FOR NEXT     │
│OUTPUT LEVEL │            │OUTPUT LEVEL │
└──────┬──────┘            └──────┬──────┘
       ▼                          ▼
┌─────────────┐            ┌─────────────┐
│ADD OFF DELAY│            │ ADD ONDELAY │
│TO OUTPUT    │            │TO OUTPUT    │
│COMPARE LOW  │            │COMPARE LOW  │
│BYTE (OCL)   │            │BYTE (OCL)   │
└──────┬──────┘            └──────┬──────┘
       ▼                          ▼
┌─────────────┐            ┌─────────────┐
│COMPENSATE   │            │COMPENSATE   │
│FOR CARRY    │            │FOR CARRY    │
│IF NEEDED    │            │IF NEEDED    │
└──────┬──────┘            └──────┬──────┘
       │      ┌─────────────┐     │
       └─────►│  CLEAR OCF  │◄────┘
              └──────┬──────┘
                     ▼
              ┌─────────────┐
              │WRITE NEW    │
              │OUTPUT       │
              │OUTPUT COMPARE│
              │LOW BYTE(OCL)│
              └──────┬──────┘
                     ▼
              ┌─────────────┐
              │     RTI     │
              └─────────────┘
```

Figure 8. Timer Interrupt Routine

Code Listing

```
****************************************************************************
* DCVAR.ASM                                                                *
* This program illustrates the use of the 16-bit free-running timer to     *
* generate a Pulse Width Modulated (PWM) signal using a Motorola HC05       *
* MCU.                                                                      *
*                                                                          *
* This example was written for the M68HC05P9 microcontroller. As shown      *
* in the accompanying application note, this program is used in             *
* conjunction with a potentiometer connected to an ADC channel to control   *
* the duty cycle of a fixed-frequency PWM signal, which appears on the      *
* TCMP pin.                                                                 *
*                                                                          *
* The loop portion of the program reads in the value of the A/D data        *
* register. When the contents of the data register do not match the         *
* OnDelay of the program, an adjustment is made to OnDelay and OffDelay      *
* and the duty cycle of the PWM signal is changed.                          *
*                                                                          *
* OnDelay is determined and controlled by the value in the A/D data         *
* register (ADDR).                                                          *
*                                                                          *
* The PWM signal is achieved by keeping track of two variables, OnDelay     *
* and OffDelay. These two values determine the amount of time the output    *
* signal is kept high or low. The timer routine reads the free-running      *
* timer, adds the delay amount to it, and stores the result in the output   *
* compare register. When the free-running timer matches the value in the    *
* output compare register, the TCMP pin is toggled, and new delay values    *
* are computed.                                                             *
*                                                                          *
* The maximum delay value is $FF, due to the use of an 8-bit value.         *
* The minimum delay value is $0D, determined by the time it takes to        *
* execute the timer interrupt routine. In this case it is around 52 bus     *
* cycles, which at an internal operating frequency of 2 MHz is ~26 µs.      *
* Because the timer frequency is the operating frequency divided by 4,      *
* the minimum delay value for the timer becomes the interrupt latency       *
* divided by 4.                                                             *
****************************************************************************
* Equate statements for assembler
****************************************************************************
* Registers
TCR             EQU             $12             ;Timer control register
TSR             EQU             $13             ;Timer status register
OCH             EQU             $16             ;Output compare registers
OCL             EQU             $17             ;(High/Low Bytes)
ACRH            EQU             $1A             ;Alternate counter registers
ACRL            EQU             $1B             ;(High/Low Bytes)
ADDR            EQU             $1D             ;A/D data register
ADSCR           EQU             $1E             ;A/D status and control register
```

AN1734

```
* Bit positions
OCIE        EQU         $06         ;OCIE bit position in TCR
OLVL        EQU         $00         ;OLVL bit position in TCR
CCF         EQU         $07         ;CCF bit position in ADSCR


* Vector and memory assignments
TIMERVEC    EQU         $1FF8       ;Timer interrupt vector
RESETVEC    EQU         $1FFE       ;Reset vector
RAMSPACE    EQU         $80         ;User RAM
ROMSPACE    EQU         $0100       ;User ROM


* Initial OnDelay and OffDelay
* 500 kHz timer frequency => 2 µs timer period
HALFPERIOD  EQU         $7D         ;2 kHz frequency @ 50% duty cycle


* PWM definitions
* Minimum achievable delay is determined by the timer interrupt latency
* Maximum achievable delay is determined from period - (minimum delay)
DELAYMIN    EQU         $0D         ;Minimum achievable delay
                                    ;determined by interrupt latency
DELAYMAX    EQU         $ED         ;Maximum delay
                                    ;(HALFPERIOD*2) - (DELAYMIN)


*******************************************************************************
* RAM Variables
*******************************************************************************
            ORG         RAMSPACE    ;Start of user RAM
OnDelay     RMB         1           ;PWM variables
OffDelay    RMB         1
ADCData     RMB         1
Delta       RMB         1
TempA       RMB         1           ;Temporary storage variable


*******************************************************************************
* Main Program
* Setup timer interrupts and analog-to-digital converter
*******************************************************************************
            ORG         ROMSPACE    ;Start of user ROM
Setup       LDA         #$20
            STA         ADSCR       ;Turn on A/D and select channel AN0
            LDA         #HALFPERIOD
            STA         OnDelay     ;Record delay variables
            STA         OffDelay
Init        BRCLR       CCF,ADSCR,Init
            LDA         ADDR        ;Get initial potentiometer reading
            STA         ADCData     ;Record value
            BSR         UpdateDC    ;Update the PWM duty cycle
            BSET        OLVL,TCR    ;Set to output high on first compare
            LDX         ACRH        ;Get the current timer value,
            LDA         ACRL        ;store in X and A
```

AN1734

16

```
               ADD           OffDelay      ;Compute the next compare time
               BCC           Setup2        ;Compensate for 8-bit carry,
               INCX                         ;increment high byte if needed
Setup2         STX           OCH           ;Store OC high byte and inhibit OC
               LDA           TSR           ;Clear OCF if set
               STA           OCL           ;Prepare for next compare
               BSET          OCIE,TCR      ;Enable OC interrupt
               CLI                          ;Clear interrupt mask
***************************************************************************
* Loop and wait for new A/D data
***************************************************************************
Loop           BRCLR         CCF,ADSCR,Loop
               LDA           ADDR          ;Get new potentiometer reading
               STA           ADCData       ;Record new value
               BSR           UpdateDC      ;Update the PWM duty cycle
               BRA           Loop          ;Repeat
UpdateDC       LDA           ADCData       ;Load new potentiometer reading
               SUB           OnDelay       ;Difference of old and new OnDelay
               BHI           DutyUp        ;If positive difference, increase DC
               BLO           DutyDown      ;If negative difference, decrease DC
               RTS                          ;Return if no difference
***************************************************************************
* Decrease the duty cycle; decrease OnDelay, increase OffDelay
***************************************************************************
DutyDown       NEGA                         ;If negative difference, decrease DC
               ADD           OffDelay      ;OffDelay += |difference|
               BCS           Dlimit1       ;Exceeded $FF; don't change
               STA           TempA         ;Temporary storage
               LDA           ADCData       ;Get ADC data
               CMP           #DELAYMIN     ;Compare with minimum limit
               BLO           Dlimit1       ;If lower than minimum, don't change
               STA           OnDelay       ;Record new OnDelay
               LDA           TempA
               STA           OffDelay      ;Record new OffDelay
               BRA           Ddone1
Dlimit1        LDA           #DELAYMAX     ;Maximum, compute limits
               STA           OffDelay
               LDA           #DELAYMIN
               STA           OnDelay
Ddone1         RTS


***************************************************************************
* Increase the duty cycle; increase OnDelay, decrease OffDelay
***************************************************************************
DutyUp         STA           Delta         ;Record difference
               LDA           OffDelay
               CMP           Delta         ;Compare OffDelay and difference
               BLO           Dlimit2       ;If OffDelay < difference, at minimum
               SUB           Delta         ;If OffDelay >= difference, go ahead
               CMP           #DELAYMIN     ;See if minimum delay was violated
               BLO           Dlimit2       ;If so, record the limit
```

AN1734

17

```
              STA        OffDelay     ;Record new OffDelay
              LDA        ADCData
              STA        OnDelay      ;Record new OnDelay
              BRA        Ddone2
Dlimit2       LDA        #DELAYMAX    ;Maximum, compute limits
              STA        OnDelay
              LDA        #DELAYMIN
              STA        OffDelay
Ddone2        RTS


*****************************************************************************
* Timer Interrupt Routine
* This routine services the output compare interrupt.
* The latency of this routine determines the minimum delay possible.
*****************************************************************************
* NOTE:
* If other timer interrupts are active, need to arbitrate them
* before servicing the output compare interrupt
*****************************************************************************
TimerInt      BRSET      OLVL,TCR,GoLow ;Determine current output level
GoHigh        BSET       OLVL,TCR       ;Setup OLVL for high next time
              LDA        OCL            ;Setup next output compare time
              ADD        OffDelay       ;Add off delay offset
              BCC        NoCarry        ;Compensate for 8-bit carry
              BRA        Carry
GoLow         BCLR       OLVL,TCR       ;Setup OLVL for low next time
              LDA        OCL            ;Setup next output compare time
              ADD        OnDelay        ;Add on delay offset
              BCC        NoCarry        ;Compensate for 8-bit carry
Carry         INC        OCH
NoCarry       LDX        TSR            ;Clear OCF bit
              STA        OCL            ;Ready for next compare
              RTI                       ;Return from interrupt


*****************************************************************************
* Vector Definitions
*****************************************************************************
              ORG        TIMERVEC       ;Timer interrupt vector
              FDB        TimerInt
              ORG        RESETVEC       ;Reset vector
              FDB        Setup
```

Freescale Semiconductor, Inc.