AN1473

Various Solutions for Calculating a Pulse and Duty Cycle

Author: *Justin Bauer*
Microchip Technology Inc.

INTRODUCTION

Many times it is desirable to quantify the pulse width of a periodic signal, such as that of a servo motor or duty cycle of a pulse-width modulated signal. There are other instances where a pulse that is non-periodic needs to be measured, such as those commonly found in a Capacitive Discharge ignition circuit. This application note describes six different strategies to measure a pulse of both periodic and non-periodic waveforms as well as six methods for calculating a duty cycle of a periodic waveform using an 8-bit PIC device.

Depending on the microcontroller chosen and what peripherals are being used, the design may require a more obscure solution then anticipated, which is why this document includes numerous approaches. Some solutions require the Configurable Logic Cell (CLC), and Numerically Controlled Oscillator (NCO), which were introduced in the year 2011. These solutions provide a hardware solution with minimal software overhead, whereas the simple Interrupt-On-Change (IOC) peripheral requires a larger software portion devoted to its calculation.

All of the solutions in this application note include associated software routines. The end result of each may differ from the results given in the document for numerous reasons, such as the clock speed of the PIC MCU, software optimization, and general environment setup.

EXECUTIVE SUMMARY

The ideal implementation would be performed entirely in hardware with the external waveform synchronized to the PIC MCU system clock. Fortunately, Timer1 Gate and the CLC, along with the NCO, provide a pure hardware solution. Since in most cases, the pulse or duty cycle to be measured is an external waveform that is produced from another source, the resolution of the measurement will always be at least one clock period.

Other solutions require software intervention to either compensate for timer rollover or register setup between edges, which cause the accuracy to suffer, as well as the min/max time constraints on the measured waveform. The software routine in these cases should be written in assembly for best accuracy. A software approach may suffice depending on the accuracy needed in the application.

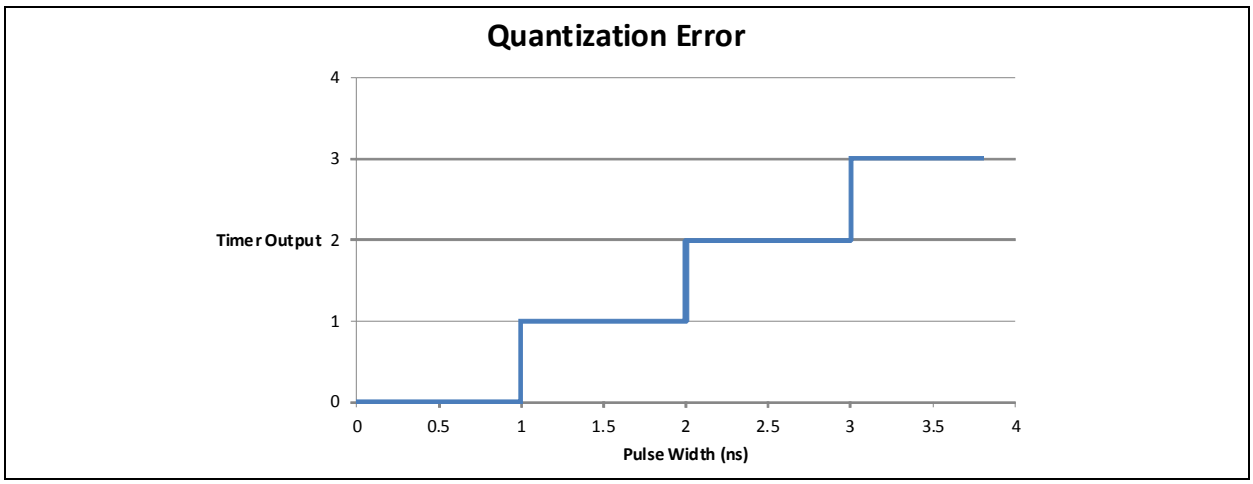
Since the duty cycle calculation is the ratio of the pulse and its period, most of the duty cycle chapter references the pulse measurement chapter. Some solutions rearrange their routine to trigger on the falling to rising edges and vice versa, whilst the CLC/NCO and Timer1 Gate solutions are setup completely different from its pulse measurement counterpart.

TERMINOLOGY

This document uses the term "Accuracy" as determined by the accuracy of the clock frequency and granularity of measurement. Higher granularity equates to a smaller resolution. To get a more accurate measurement, choose a clock with high accuracy and high frequency for smaller granularity and higher resolution. Another term that is used is measurement uncertainty. It is uncertain when the timer will stop relative to the pulse edge: it may stop exactly when the pulse edge occurs, or it may be as much as one clock later. The uncertainty is one full clock.

As seen below (Figure 1), the resolution of the timer greatly affects the resultant measurement of the pulse.

FIGURE 1: QUANTIZATION ERROR WHEN DETERMINING THE ASSOCIATED TIMER VALUE TO A CERTAIN PULSE WIDTH. THE TIMER INCREMENTS AT A RATE OF 1 ns



POSSIBLE SOLUTIONS

The most accurate measurement will consist of the fastest clock source with a timer on the lowest prescale. The lowest timer prescale yields the highest resolution. Higher resolution typically increases the maximum count, which may necessitate compensation for rollover of the timer.

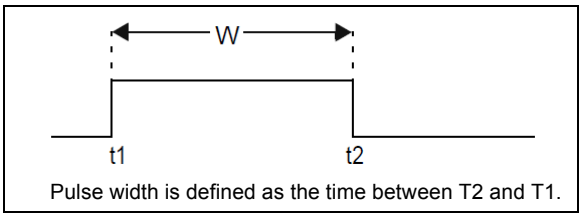
An interrupt routine can be incorporated into the solutions if non-blocking code is to be used. The accuracy of such an implantation may suffer accuracy as a result. For example, the IOC pin can cause an interrupt when a rising/falling edge is detected. The pulse measurement can now be completed inside of the ISR without the need to constantly poll the pin. While this may sound ideal, the user must now accommodate for the 3-5 instruction cycle delay that is caused by servicing an asynchronous interrupt.

When absolute accuracy is important, then an external crystal should be used since the internal oscillator block can have a drift up to 5% of its nominal frequency.

All of the measurements in this document including the associated code use an internal 16 MHz system clock with all timers on a 1:1 prescale of the system clock (FOSC), unless otherwise noted. The Low and High waveform length constraints, as well as the accuracy of the measured result are calculated using Table 1 and Table 2. If rollover is accounted for in software-based solutions, then the accuracy of the measurement will decrease in proportion to the software routine overhead.

These solutions assume a pulse to be active-high and a period to be the time between two rising edges. Please see Figure 2 and Figure 3 for clarification.

FIGURE 2: PULSE WIDTH DEFINITION



Pulse width is defined as the time between T2 and T1.

FIGURE 3: DUTY CYCLE IS THE RATIO OF PULSE WIDTH AND PERIOD

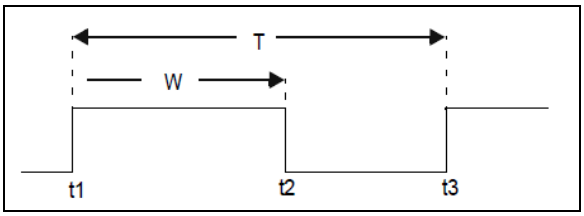


TABLE 1: ASSOCIATED CODE NUMBERS

Pulse Measurements – Code Numbers						
Modules	**PIC® MCU**	**Interrupt**	**Language**	**Size (Program / Data)**	***Limit High**	**Resolution**
Timer1 Gate	PIC16LF1509	Yes	C	539 / 12	4.1 ms	62.5 ns
Timer1						
CLC*2	PIC16LF1509	Yes	C	400 / 5	65.54 ms	62.5 ns
NCO1						
CCP1	PIC16LF1847	No	ASM	126 / 6	16.38 ms	3 us
Timer1/3						
IOC/INT	PIC16LF1509	Yes	C	50 / 2	1 ms	4 us (16x prescaler)
Timer0						
IOC	PIC16LF1509	Yes	C	50 / 3	1.073s	6 us
Polled Input	PIC16LF1509	No	C	50 / 3	1.073s	8 us
Duty Cycle Measurements – Code Numbers						
Timer1 Gate	PIC16LF1509	Yes	C	500 / 2	4.1 ms	125 ns
Timer1						
NCO1	PIC16LF1509	Yes	C	400 / 5	65.54 ms	62.5 ns
CLC						
CCP1	PIC16LF1847	No	ASM	126 / 6	16.38 ms	3 ns
Timer1/3						
IOC/INT	PIC16LF1509	Yes	C	50 / 2	4.1 ms	8 ns
TimerX						
IOC	PIC16LF1509	Yes	C	50 / 3	1.073s	12 us
Polled Input	PIC16LF1509	No	C	50 / 3	1.073s	20 us
* The upper limit on these calculations can be extended to any set amount. The limit that is presented in this table reflects a single timer rollover.						

TABLE 2: EQUATIONS USED FOR THE RESULTS SHOWN IN Table 1

Pulse Measurements – Equations		
Modules	**Limit High**	**Resolution**
Timer1 Gate Timer1	$\left(\frac{1}{f}\right) * 2^n$	$\frac{1}{f}$
CLC*2 NCO1 TimerX	$\left(\frac{1}{f}\right) * 2^n$ n=NCO ACCUM bit width	$\frac{1}{f}$
CCP1 Timer1/3	$\left(\frac{1}{f}\right) * 2^n$	Software dependent
IOC/INT Timer0	$\left(\frac{1}{f}\right) * 2^n * 4$	Software dependent (16x prescaler)
IOC	$\left(\frac{4}{f}\right) * 2^n$	Software dependent
None	$\left(\frac{4}{f}\right) * 2^n$	Software dependent
Duty Cycle Measurements – Equations		
Timer1 Gate Timer1	$\left(\frac{1}{f}\right) * 2^n$	$\frac{1}{f}$
CLC*2 NCO1 TimerX	$\left(\frac{1}{f}\right) * 2^n$ n=NCO ACCUM bit width	$\frac{1}{f}$
CCP1 Timer1/3	$\left(\frac{4}{f}\right) * 2^n$	Software dependent
IOC/INT Timer0	$\left(\frac{1}{f}\right) * 2^n * 4$	Software dependent (16x prescaler)
IOC	$\left(\frac{4}{f}\right) * 2^n$	Software dependent
None	$\left(\frac{4}{f}\right) * 2^n$	Software dependent
n = TimerX bit width f = Clock frequency		

PULSE

This section will describe how to measure a single pulse of both periodic and non-periodic waveforms. Solutions that require a periodic source involve a pulse measurement that is subtracted from its period until the pulse value is smaller than the period. This strategy can be employed in situations when it is impossible to trigger on a rising edge and then setup for a falling event due to software speed limitations.

Timer1 Gate

TABLE 3: TIMER1 GATE CODE CALCULATIONS

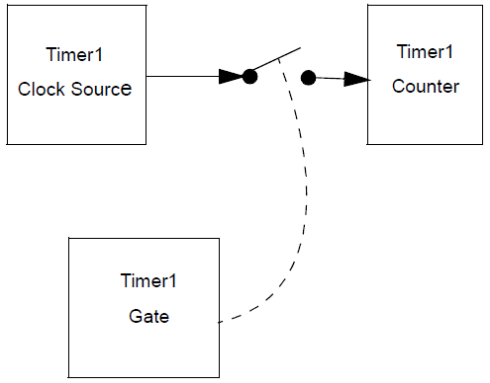
Modules	Limit High	Resolution
Timer1 Gate Timer1	4.1 ms	62.5 ns

Timer1 Gate is the classical approach to measuring the pulse width of a periodic and non-periodic signal. It is recommended that this method be given preference, since it is very accurate and simple to configure. The entire capture is performed in hardware. It is also widely available on most PIC devices.

OVERVIEW

Timer1 Gate controls when Timer1 increments based on external triggers. The trigger can either be a rising or falling edge on the Timer1 gate input.

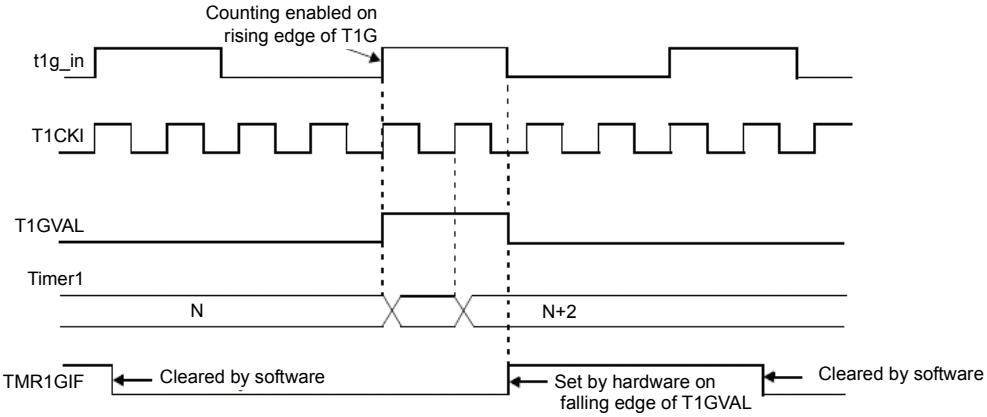
FIGURE 4: TIMER1 OPERATION



Timer1 Gate connects the clock source to the timer which will subsequently start counting. The strategy only concerns itself with closing and opening the gate.

Assuming that the pulse being measured is active-high, and a rising edge has just occurred, the gate will connect the clock source to its counter, and Timer1 will now increment as long as the pulse is kept High. When the waveform goes Low, the gate will disconnect, and an interrupt flag will be set. The pulse width can now be determined by reading out the 16-bit value in the Timer1 count registers.

FIGURE 5: TIMER1 GATE WILL CONNECT WHEN THE SIGNAL ON 'T1G_IN' GOES HIGH

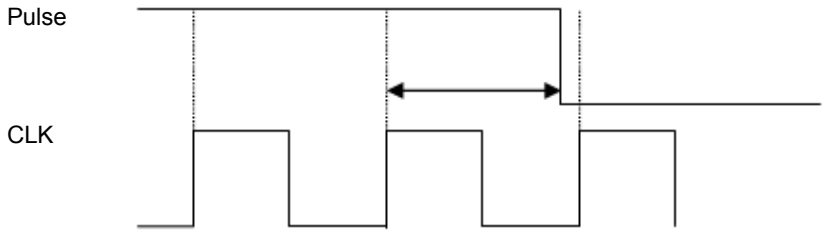


SETUP

1. Setup Timer1 Gate for Single-Pulse mode on rising/falling edge
2. Choose an appropriate Timer1 clock source and prescale for the pulse
3. Clear `TMR1H` and `TMR1L`
4. Enable the module and set `T1GG0` bit
5. (pulse occurs)
6. `TMR1GIF` is set
7. Read pulse width out of TMR1H:L
8. Clear `TMR1GIF`
9. Repeat steps 2->8

FOSC should be selected as the Timer1 clock source with a 1:1 prescale value for the best resolution.

FIGURE 6: UNCERTAINTY DEPENDS ON CLOCK ACCURACY



For a 16 MHz clock source, the uncertainty can be as much as +/-62.5 ns.

LIMITATIONS

There is uncertainty of plus or minus one clock period. The worst case will be when the pulse goes Low just before the rising edge of the clock, or when the pulse goes High just after the clock. Timer1 has a maximum count of 65535. On the 65536th period the timer count overflows to 0. You can accommodate pulses longer than 65535 Timer1 periods by counting the number of Timer1 overflows. The TMR1IF bit is set at each overflow event. Count the number of overflow events and add 65536 times that number to the Timer1 count.

CLC AND NCO

TABLE 4: CLC/NCO CODE CALCULATIONS

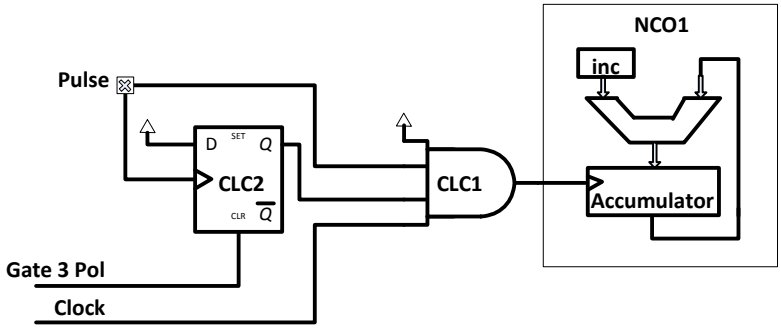
Modules	Limit High	Resolution
CLCX2	65.54 ms	62.5 ns
NCO1		

This strategy uses the NCO and CLCs together.

OVERVIEW

Two CLCs can be setup as a pulse detector. Upon software Reset, CLC2 will output a Low, which will enable the CLC1 AND gate allowing the clock signals through when the pulse is high. When the pulse goes High, HFINTOSC will clock the accumulator for as long as this pulse is High. When the pulse goes Low, CLC1 outputs a Low and stops clocking the NCO. The pulse width can now be read from the 20-bit wide NCO accumulator registers.

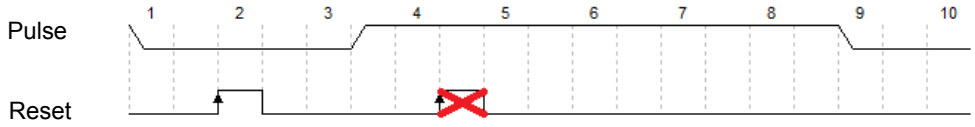
FIGURE 7: NCO AND CLC CONNECTIONS FOR MEASURING A SINGLE PULSE



Top-level view of the interconnections within the CLC and NCO. All of the connections are internal to the processor except the pulse.

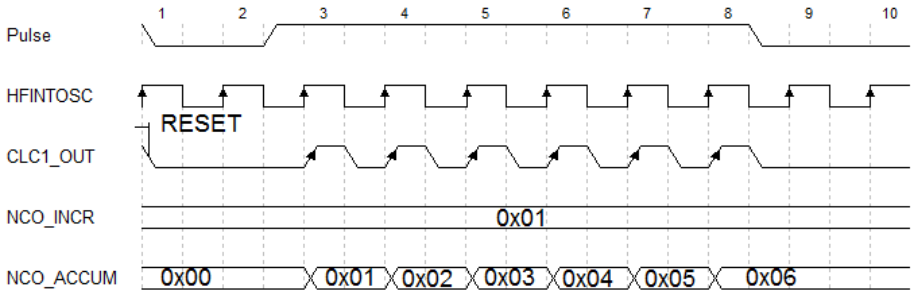
An important aspect about this setup is that inaccurate measurements can be made if CLC2 is not reset before the rising edge of the pulse, as seen in Figure 8.

FIGURE 8: UNEXPECTED RESET FROM SOFTWARE SHOULD BE AVOIDED WHEN PULSE IS HIGH



It is crucial that CLC1 is reset before the pulse occurs, otherwise an inaccurate measurement will be made.

FIGURE 9: TIMING DIAGRAM OF Figure 7



When the pulse goes High, the NCO starts accumulating by a value of 1 every HFIINTOSC clock edge until the pulse goes Low.

SETUP

1. Setup CLC1 as a 4-input AND gate
   a) Input 1 to VDD (invert Gate 1 output). Input 2 is the pulse to measure
   b) Input 3 is HFINTOSC
   c) Input 4 is the inverse of the output of CLC2
2. Setup CLC2 as a D-Flop.
   a) Connect 'D' to VDD (invert Gate 2 output)
   b) Connect the clock to the inverse of the pulse pin
   c) Reset connected to gate 3
3. Setup NCO
   a) Configure the NCO for Fixed Duty Cycle (FDC) mode
   b) Choose an increment value appropriate for the pulse to be measured
4. To start, toggle Reset gate in CLC2
5. (pulse High)
6. (counting)
7. (pulse Low)
8. Read pulse width from the accumulator
9. Repeat steps 3->8

The NCO accumulator value is in terms of the clock source and must be scaled appropriately to find the pulse width. A common solution to this is to create a look-up table with the NCO accumulator value being the offset into it.

Problem 1:

Q: Servo pulse to be measured is approximately 1 ms → 2 ms wide. Using NCO accumulator as the counter with a clock input of 16 MHz, what is the corresponding look-up table?

A: Configure the NCO to increment by 1 at FDC mode with CLC1 as its clock source. When the pulse is High, the NCO will increment by 1 every 62.5 ns. After 2 ms, the NCO accumulator will have a value of 32000. Subtract 1 ms from the accumulator value (16000), and then left shift 7 times – essentially divide by 128 in order to create a feasible look-up table of 125 values. Each value now corresponds to 8 us.

TABLE 5: LOOK-UP TABLE FOR PROBLEM 1

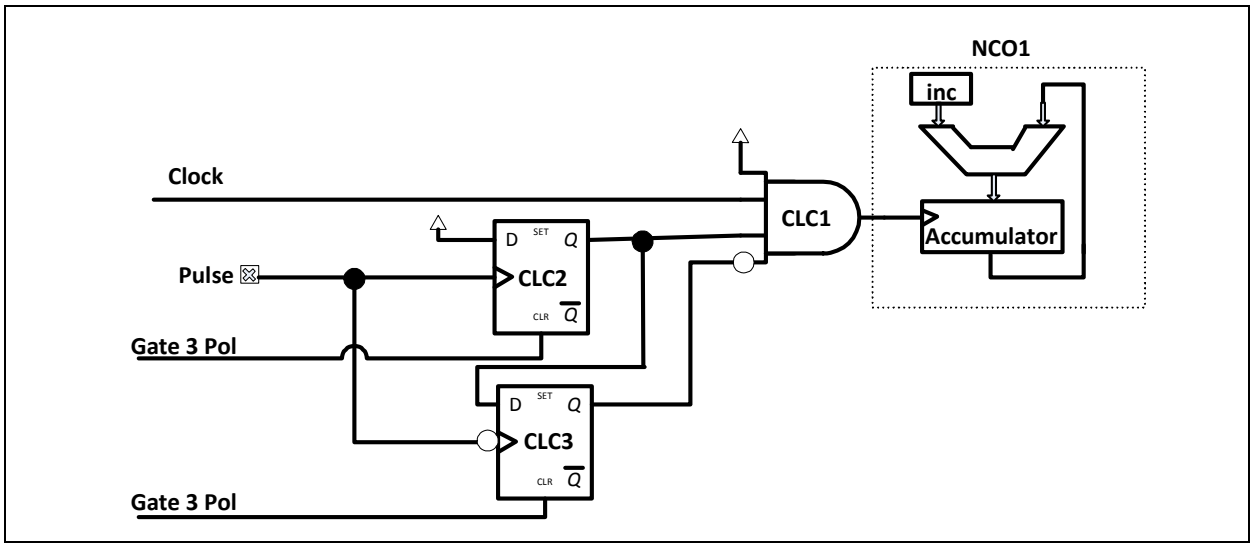
Offset	Scaled Result
0	<=1 ms
1	1.8 us
2	1.16 us
…	….
127	2 ms

A drawback of this method is that it requires a new configuration of the NCO and look-up table every time the expected pulse width is changed.

Note:	Maximum accumulator value is 2^20. This corresponds to 65.536 ms, if a clock of 16 MHz is used.

A better solution will capture only a single pulse which will force the NCO to retain its accumulator value until a Reset is issued. This is useful if the circuit cannot be reset before another pulse arrives, as seen in Figure 8 above.

FIGURE 10: CLC AND NCO CONNECTIONS FOR MEASURING A SINGLE PULSE THAT CANNOT BE RESET DURING PULSE



On Power-on Reset (POR), CLC2 and CLC3 outputs are both Low. CLC1 is inhibited by CLC3 Q Low. Upon a Pulse, the following events occur:

1. Rising edge sets CLC2 Q, enabling CLC1 gate to pass the clock to NCO
2. Falling edge sets CLC3 Q, disabling CLC1 gate and stopping the clock to NCO. CLC3 Q high is count ready signal.
3. The system stays in this state until reset.

Gate 3 polarity is controlled through a single bit in the CLC Configuration registers for each CLC block. Use this to reset the circuit in this order:

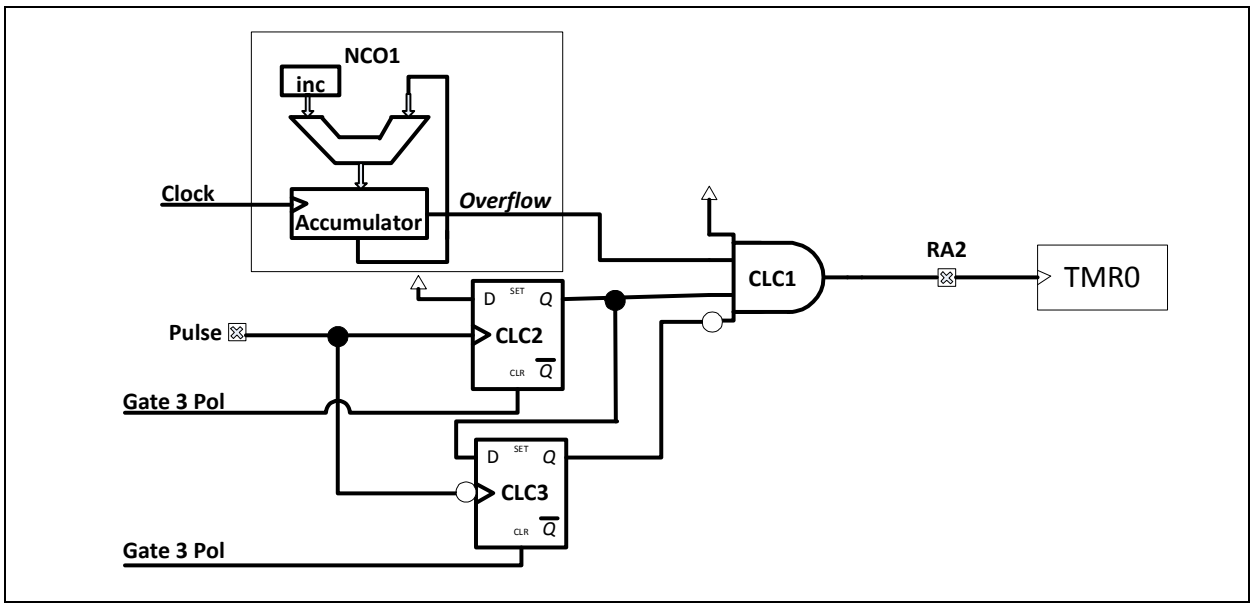
1. CLC2 Reset high – pulse cannot set this while Reset is high, Q is low inhibiting CLC1
2. CLC3 Reset high
3. CLC3 Reset low – pulse transition from high-to-low cannot set CLC3 because the D input is held low by CLC2
4. CLC2 Reset low – Reset is complete and ready for next pulse

SETUP

1. Setup CLC1 as a 4-input AND gate
   a) Input 1 to VDD (invert Gate 1 output)
   b) Input 2 is the clock source (HFINTOSC)
   c) Input 3 is the CLC2 output
   d) Input 4 is the inverse of CLC3 output
2. Setup CLC2 as a D-FLOP
   a) Data to VDD(invert Gate 2 output)
   b) Clock source is the pulse
   c) Invert Gate 3 to hold in Reset. Release this after releasing CLC3 Reset when starting pulse measurement.
3. Setup CLC3 as a D-FLOP
   a) Data is CLC2 output
   b) Clock source is the inverse of the pulse input
   c) Invert Gate 3 to hold in Reset. Release this after releasing CLC3 Reset when starting pulse measurement.
4. Setup NCO
   a) Configure the NCO for FDC mode
   b) Choose an increment value appropriate for the pulse to be measured

Division and subtraction is a time consuming process for 8-bit microcontrollers. Because of this, a third solution using the CLC and NCO blocks is presented.

FIGURE 11: CLC AND NCO CONNECTIONS TO PROVIDE A NORMALIZED CLOCK FOR TIMER0



This solution uses Timer0 multiplexed on the same pin as CLC1 out. The NCO is configured for FDC mode as previously, although now the output is used as a clock for Timer0. Since the NCO can produce up to 20 bits of resolution as a linear frequency generator, the look-up table values can be derived directly from the NCO period.

Problem 2:

Q: A Servo pulse to be measured is approximately 0 ms → 1 ms wide. Using Timer0 as the counter, what is the maximum NCO frequency without TMR0 experiencing an undetectable overflow/underflow condition?

A: Calculate the NCO frequency that will scale Timer0 so that a value of 255 corresponds to 1 ms. Configure the NCO for Pulse Frequency mode. Divide the expected pulse width by the width of Timer0 (1ms/255 bits). Take the inverse of that to arrive at a frequency of 255 kHz for the NC0.

*Foverflow = (NCO Clock Frequency * Increment Value) / 2^n*

*255kHz = 16MHz * Increment Value / 2 ^ 20*

Solve the above equation for the increment value and the answer is approximately 16711. This will configure the NCO to output a single pulse to clock TMR0 at a rate of 255 kHz. If the timer overflows due to error in the clock frequency, check the Timer0 interrupt flag and account for the overflow in software.

TABLE 6: LOOK-UP TABLE FOR PROBLEM 2

Offset	Scaled Result
0	0 ms
1	3.92 us
2	7.84 us
…	….
255	1 ms

SETUP

1. Configure CLC1 out pin as a digital output
2. Setup CLC1 as 4-input AND gate
   a) Input 2 is VDD (invert Gate 1 output)
   b) Input 1 is the NCO output
   c) Input 3 is the output from CLC2
   d) Input 4 is the inverse of CLC3 output
3. Setup CLC2 as D-FLOP
   a) Connect D to VDD (invert Gate 3 output)
   b) Clock source is the pulse
   c) Invert Gate 3 to hold in Reset. Release this after releasing CLC3 Reset when starting pulse measurement.
4. Setup CLC3 as a D-FLOP
   a) Connect D to CLC2 output
   b) Clock source is the inverse of the pulse
   c) Invert Gate 3 to hold in Reset. Release this before releasing CLC2 Reset when starting pulse measurement.

LIMITATIONS

Since the measurement is done in hardware, there is no software overhead with the exception of resetting CLC2 and CLC3 if another pulse measurement is to be made. Uncertainty is minus 0, plus one clock period.

CCP

TABLE 7: CCP CODE CALCULATIONS

Modules	Limit High	Resolution
CCP1	16.38 ms	3 us
Timer3		

The capture and compare module provides an accurate hardware and software method to measure the width of a pulse. The presented solution requires the waveform to be periodic since a period measurement is taken.

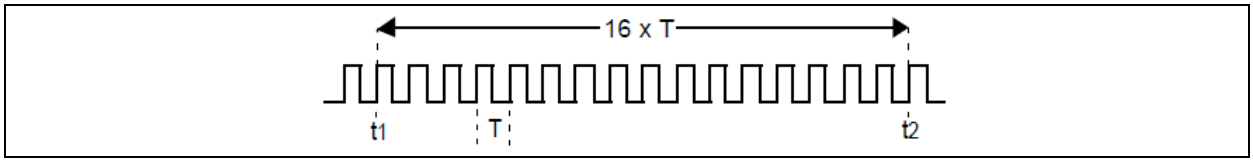
OVERVIEW

The CCP can be setup to capture both a rising and falling edges of a waveform. A timer count value is captured when the CCP receives an event condition. An event is defined as one of the following:

1. Every falling edge
2. Every rising edge
3. Every 4th rising edge
4. Every 16th rising edge

An event on every 4th or 16th rising edge is typically used to calculate the period of a signal, as seen in Figure 12.

FIGURE 12: CALCULATING THE PERIOD OVER 16 ITERATIONS OF THE WAVEFORM



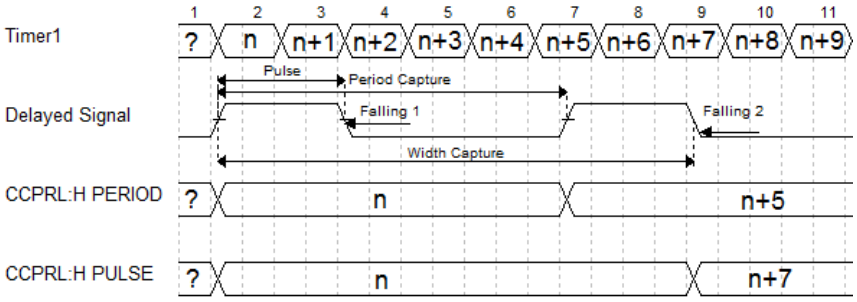
Timer1 is used on the PIC16/12/10 devices with Timer3 being another option on the PIC18 devices. When a capture is made, the timer value will be latched to the CCPRxL:H registers. As seen in Figure 1, a pulse width is defined as the time between a rising and falling edge. Since a pulse measurement requires both a rising and falling edge, the CCP module needs software intervention to capture both events.

Note:	Clocking Timer1 from the system clock (FOSC) should not be used in Capture mode. In order for Capture mode to recognize the trigger event on the CCPx pin, Timer1 must be clocked from the instruction clock (FOSC/4).

Ideally, the pulse being measured is many times wider than the setup time required by the CCP module to switch between capturing the rising and falling edges. Often times, the pulse is much less than the required instruction clocks it takes to make the switch. Due to this limitation, a wiser approach is to also take a period measurement. If the pulse measurement is greater than the period measurement then subtract the period measurement from the pulse measurement until the result is less than the period.

The pulse measurement requires a period measurement as well, as seen in Figure 13.

FIGURE 13: EXTRACTING THE PULSE FROM PERIOD PLUS PULSE CAPTURE



The Period is approximately five Timer1 increments wide and the pulse being two. Since the CCP is unable to setup for falling edge before the first falling edge occurs (Falling 1), the second one is captured (Falling 2). The pulse width can be measured by subtracting the previously measured period (5) from the pulse capture (7): 7-5 = 2. The signal is delayed due to the internal synchronizing circuitry. These measurements are not done simultaneously.

The period can be measured by setting the CCP to capture on every 16th rising edge provided that 16 periods are less than 65536 Timer1 clock periods. If that is the case then select 4 or one events as the Capture mode. Make two captures and calculate the difference by subtracting the first from the second and ignore borrows. Divide the result by 16 (four right shifts without extending the sign) to determine the period. It does not matter if the second capture value is less than the first when calculating the difference.

EXAMPLE 1: CODE SNIPPET FOR CALCULATING A SINGLE PULSE WIDTH IN Figure 13

```
// The pulse width may have been measured over the length
// of more than one period, so subtract the period
// out until the pulse width is less than the period.
while(MeasuredPulse > Period) MeasuredPulse -= Period;
```

The code in Example 2 and Example 3 show how the falling edge detection can be missed soon after detecting the rising edge due to software setup time for the CCP module.

Because of software intervention, pulses that are less than the time required to get out of the `while(!CCP2IF)` loop and setup for the falling edge will not be detected. The lines in bold in the ‘C’ code below highlight the routines that must be executed before the falling edge is detected.

EXAMPLE 2: 'C' CODE THAT IS USED TO CAPTURE A SINGLE PULSE

```
//*** First capture a rising edge
CCP2IF = 0;
while(!CCP2IF && !TMR1IF);  // Wait for edge or Timer overflow
CCP2CON = 0x00;             // stop capture
itemp = CCPR2L;             // Now store the capture

if(TMR1IF)
{
// if Timer1 overflowed then the signal period is too long
// and the MeasuredPulse will be forced to the Period or 0
// depending on the pin voltage
    if(PWM_INPUT_PIN)
    {
        MeasuredPulse = Period;
    }
    else
    {
        MeasuredPulse = 0;
    }
}
else
{
    //*** Next capture a falling edge
    ……..
}
```

Since this is a top-level language, the actual instructions that the PIC MCU uses need to be analyzed to get an accurate expected delay.

EXAMPLE 3: DISASSEMBLY OF THE SOFTWARE ROUTINE OF CHANGING EDGES FOR THE CCP TO CAPTURE

```
//*** First capture a rising edge
98:                         CCP2IF = 0;
  070C    1012     BCF 0x12, 0
99:                         while(!CCP2IF && !TMR1IF);  // Wait
  070D    1812     BTFSC 0x12, 0
  070E    2F10     GOTO 0x710
  070F    2F11     GOTO 0x711
  0710    2F15     GOTO 0x715
  0711    1C11     BTFSS 0x11, 0
  0712    2F14     GOTO 0x714
  0713    2F15     GOTO 0x715
  0714    2F0D     GOTO 0x70d
100:                        CCP2CON = 0x00;  // stop capture
  0715    0025     MOVLB 0x5
  0716    019A     CLRF 0x1a
101:                        itemp = CCPR2L; // store the capture
  0717    0818     MOVF 0x18, W
  0718    0020     MOVLB 0
  0719    00A8     MOVWF 0x28
  071A    01A9     CLRF 0x29
102:                         if(TMR1IF)
……..
```

The above disassembly was compiled with the free version of XC8 v1.01. Your routine may differ depending on the optimization used. Remember that each `GOTO` or any instruction that modifies the PC takes two instruction cycles.

If the period is considerably smaller than the CCP setup time, the CCP module should be configured to capture every 16th rising edge.

SETUP

Steps to measure the period:

1. Configure CCPXCON<3:0> to capture every 16th rising edge
2. Configure TimerX prescaler for the lowest setting without overflowing on the 16th rising edge
3. Enable the CCP module and timer
4. (16th rising edge)
5. Disable CCP module and save CCPRX in a temporary variable (temp)
6. Enable the CCP module
7. (16 edges later)
8. Disable CCP module and timer
9. The period is now

:

```
Period_16 =  CCPR2 – temp
Period = Period_16 >> 4
```

Steps to measure the pulse:

1. Configure CCPXCON<3:0> to capture every rising edge
2. Configure TimerX prescaler for the lowest setting without overflowing before period
3. Enable the CCP module and timer
4. (rising edge)
5. Disable CCP module and save CCPRX in a temporary variable (temp)
6. Configure CCPXCON<3:0> to capture every falling edge
7. Enable CCP module
8. (falling edge)
9. Disable CCP module and timer
10. The pulse width is now:

```
MeasuredPulse = CCPRX - itemp;
while(MeasuredPulse > Period) MeasuredPulse -= Period;
```

LIMITATIONS

The resolution of this measurement is a function of the PIC MCU instruction clock. The period of the waveform of interest must be at least two times the width of the instruction clock period. This is because the CCP module detection is synchronized to the clock and must register each rising and falling edge.

IOC (Interrupt-On-Change) With Timer

TABLE 8: IOC WITH TIMER CODE CALCULATIONS

Modules	Limit High	Resolution
IOC	1 ms	4 us (16x prescaler)
TimerX		

If no hardware modules are available to measure a pulse, a single interrupt-on-change pin can be configured to cause an interrupt on both rising and falling edges. The minimum pulse width must be at least 25 ns wide for the PIC16F1847 according to the electrical specifications. Other factors will limit the performance of this method, such as interrupt latency.

OVERVIEW

This method is similar to what the CCP module does, except it cannot stop the timer immediately when an event is detected (rising/fall edge). This raises the uncertainty in the calculation, since software must now stop the timer. Some PIC devices can specify rising and falling edge triggers, while others cannot differentiate between the two. The INT pins can be used as well as any of the PORTB pins if the device supports IOC.

First, the period of the periodic waveform will be measured. Then a pulse measurement will be performed.

SETUP

Steps to measure the period:

1. Configure one of the IOC pins to interrupt on rising edge (IOCB**P**)
2. Set the pin as a digital input
3. Setup any timer so that it will count T (Figure 3) without overflowing
4. Enable the timer
5. When the IOC flag is set, clear it and save the timer into a temporary variable (temp)
6. When the interrupt flag is set again, stop the timer and save the timer into a temporary variable (temp2)
7. Period is now:

```
Period = temp2 - temp
```

Steps to measure the pulse:

1. Configure one of the IOC pins to interrupt on rising edge (IOCB**P**)
2. Set the pin as a digital input
3. Setup any timer so that it will count `W` (Figure 3) without overflowing
4. Enable the timer
5. When the IOC flag is set, clear it and save the timer into a temporary variable (`temp`)
6. Configure the pin to interrupt on falling edge (IOCB**N**)
7. When the interrupt flag is set again, clear it and stop the timer and save the timer into a temporary variable (`temp2`)
8. Pulse width is now:

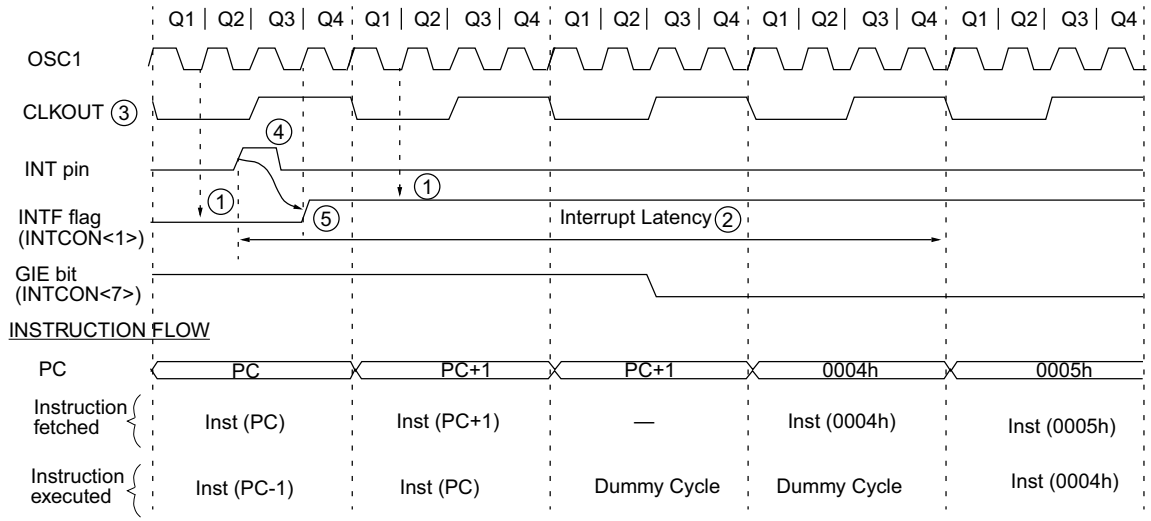
```
MeasuredPulse = temp2 - temp;
while(MeasuredPulse > Period) MeasuredPulse -= Period;
```

LIMITATIONS

This method has a restriction that the period must be greater than the amount of time required to capture two rising edges. If the software setup time between rising edges is greater than the period, then the period measurement will be incorrect.

If the IOC flags cause a jump to the ISR when set, the designer must be aware of the latency incurred by the interrupt vectoring. Since this interrupt is asynchronous to the clock, the expected delay is 3-5 instruction cycle times, as seen below (Figure 14).

FIGURE 14: INTERRUPT LATENCY



Note 1: INTF flag is sampled here (every Q1).
2: Interrupt latency = 3-4 TCY where TCY = instruction cycle time.
Latency is the same whether Instruction (PC) is a single cycle or a 2-cycle instruction.
3: CLKOUT is available only in RC oscillator mode.
4: For minimum width of INT pulse, refer to AC specs.
5: INTF is enabled to be set anytime during the Q4-Q1 cycles.

IOC (Interrupt-On-Change) Without a Timer

TABLE 9: IOC WITHOUT TIMER CODE CALCULATIONS

Modules	Limit High	Resolution
IOC	1.073s	6 us

This method is similar to the interrupt-on-change with a timer, with the exception that it uses the system clock to count, and not a timer

OVERVIEW

This provides a “cheap” method in determining the width of a pulse. The user must simply wait for a rising edge to be detected and then increment a register until a second rising interrupt is detected. This measures the period. A rising edge, and then subsequent falling edge measurement, will be saved and then subtracted from the period to get the pulse width.

SETUP

Steps to measure the period:

1. Configure one of the IOC pins to interrupt on rising edge (IOCBP)
2. Set the pin as a digital input
3. Setup any variable so that it will count 'T' (Figure 3) without overflowing
4. Start incrementing the temporary variable (count)
5. When the IOC flag is set, clear it and save the count variable into another temporary variable (temp)
6. Continue incrementing count variable
7. When the interrupt flag is set again, clear it and save the count variable into another temporary variable (temp2)
8. Period is now:

```
Period = temp2 - temp
```

Steps to measure the pulse:

1. Configure one of the IOC pins to interrupt on rising edge (IOCBP)
2. Set the pin as a digital input
3. Setup any variable so that it will count 'W' (Figure 3) without overflowing
4. Start incrementing the temporary variable (count)
5. When the IOC flag is set, clear it and save the timer into a temporary variable (temp)
6. Configure the pin to interrupt on falling edge (IOCBN)
7. When the interrupt flag is set again, clear it and save the count variable into another temporary variable (temp2)
8. Pulse width is now:

```
MeasuredPulse = temp2 - temp;
while(MeasuredPulse>Period) MeasuredPulse-= Period;
```

LIMITATIONS

Since this implementation is in 'C', there is a greater number of instructions needed compared to an assembly routine. The snippet in Example 4 is what is required to simply check if a pin is high and increment a variable if it is. Notice the amount of instructions required for just a few simple statements.

EXAMPLE 4: DISASSEMBLY OF TWO LINES OF C CODE

```
308:                    while(!IOCBFbits.IOCBF4) //wait for falling edge
  0014    281F      GOTO 0x1f
  001F    0027      MOVLB 0x7
  0020    1E16      BTFSS 0x16, 0x4
  0021    2823      GOTO 0x23
  0022    2824      GOTO 0x24
  0023    2815      GOTO 0x15
  0024    2825      GOTO 0x25
309:                    count++; //increment the variable
  0015    3001      MOVLW 0x1
  0016    0020      MOVLB 0
  0017    07A8      ADDWF 0x28, F
  0018    3000      MOVLW 0
  0019    3DA9      ADDWFC 0x29, F
  001A    3000      MOVLW 0
  001B    3DAA      ADDWFC 0x2a, F
  001C    3000      MOVLW 0
  001D    3DAB      ADDWFC 0x2b, F
  001E    281F      GOTO 0x1f
```

As one can infer, the number of instructions to complete just two lines in 'C' can be cumbersome. The approximate delay to increment 'count' on each instruction cycle is 20 instructions clocks, or rather 5 microseconds, if FOSC == 16 MHz. If the pulse occurred and the 'count' register contains a value of 12, then the pulse width is then:

```
Pulse Width =
count*delay*InstructionTimer

Pulse Width = 12*20* 1/(16MHz/4) = 60us
```

Polled Input

This method can be used by any PIC device and is the most basic in terms of modules required (PORT).

OVERVIEW

TABLE 10: POLLED INPUT CALCULATIONS

Modules	Limit High	Resolution
NONE	1.073s	8 us

The code to measure a single pulse width is similar to the IOC strategy, with the exception that it will not have interrupt-on-change capabilities. Software must constantly loop while polling a single pin.

SETUP

Steps to measure the period:

1. Configure one of the pins to be a digital input
2. Setup any variable so that it will count 'T' (Figure 3) without overflowing
3. Increment a temporary variable (`count`) when the pin transitions from its low-to-high state
4. When the pin transitions again from low-to-high, stop incrementing the count variable
5. Period is now:

```
Period = count
```

Steps to measure the pulse:

1. Configure one of the pins to be a digital input
2. Setup any variable so that it will count 'W' (Figure 3) without overflowing
3. Increment a temporary variable (`count`) when the pin transitions from its low-to-high state
4. When the pin transitions from high-to-low, stop incrementing the count variable
5. Pulse width is now:

```
MeasuredPulse = count
```

LIMITATIONS

The number of instructions that it takes to increment the register must be taken into account of the final pulse width. The most accurate way to determine this is to look at the disassembly or write the routine in assembly. The first falling edge may be impossible to capture if the pulse is small enough.

DUTY CYCLE MEASUREMENTS

This section will describe how to measure a waveform's duty cycle. The majority of the modules listed in the "Pulse Measurement" section have already measured the period and pulse in the previous section. Only a simple ratio between the two is needed in order to calculate the duty cycle.

Duty cycle measurements are inherently accurate because it is the ratio of two numbers — the pulse and period. Absolute errors in the pulse measurements are canceled by performing the division.

Timer1 Gate

TABLE 11: TIMER1 GATE CODE CALCULATIONS

Modules	Limit High	Resolution
Timer1 Gate	4.1 ms	1.2 us
Timer1		

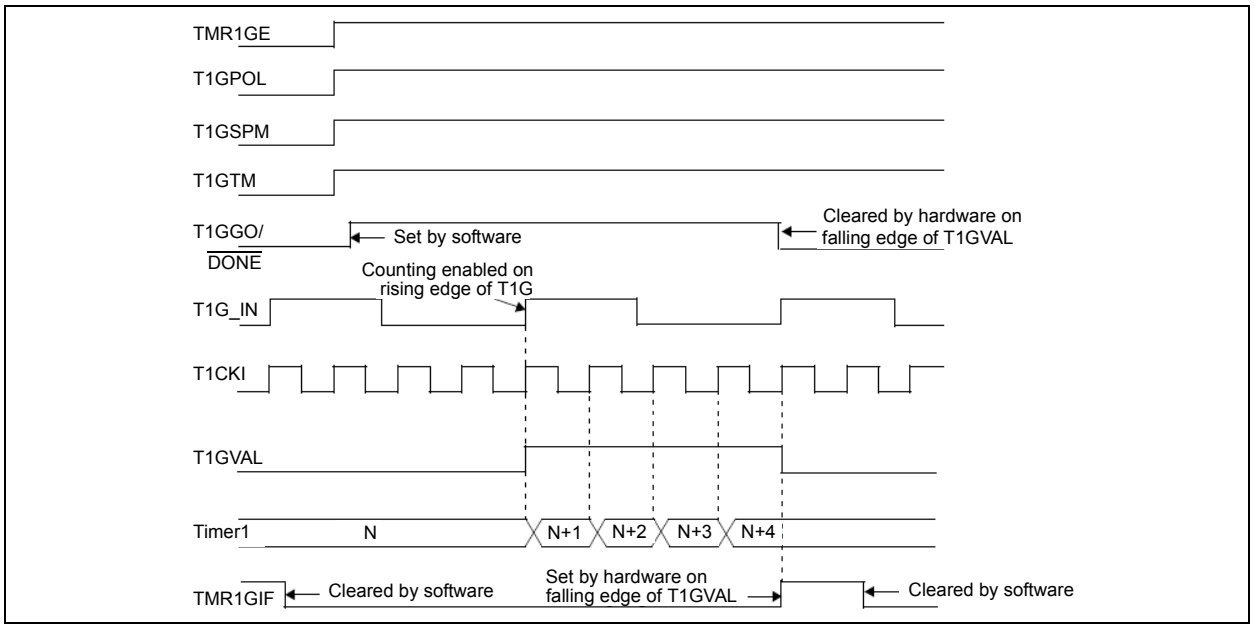
Microcontrollers that have the gate feature of Timer1 can also measure the duty cycle of a waveform.

OVERVIEW

An important prerequisite is that the period of the waveform that is to be measured must occur at least twice. This is because the result requires two measurements to be made, as well as post processing. This forces the need of software intervention between the two measurements to setup the next correct configuration.

Timer1 Gate single-pulse and toggle can be combined to measure the period of the waveform, as seen in Figure 15:

FIGURE 15: TIMER1 GATE SINGLE-PULSE AND TOGGLE COMBINED MODE



If interrupts are enabled, the end of the period will set the interrupt flag. After this measurement is saved, the procedure to measure a single pulse as seen earlier in this document (see **Section "Timer1 Gate"** under Pulse Measurement) can be repeated. The duty cycle is then simply a ratio of the pulse width over period length.

SETUP

1. Setup Timer1 Gate for Single-Pulse and toggle mode on rising/falling edge
2. Choose an appropriate Timer1 clock source and prescale for the period
3. Clear `TMR1H` and `TMR1L`
4. Enable the module and set `T1GG0` bit
5. (two consecutive rising/falling edges occur)
6. `TMR1GIF` is set
7. Save period width from of `TMR1H:L`
8. Clear `TMR1GIF`
9. Setup Timer1 Gate for Single-Pulse (see **Section "Timer1 Gate"** under Pulse Measurement)
10. Save pulse width measurement
11. Divide the pulse width by the period to determine the duty cycle ratio

LIMITATIONS

This solution requires two measurements to be made: the waveform's pulse and period. Both of these are done entirely inside of the module after software initiation. The uncertainty for each measurement is plus or minus one clock period.

NCO + CLC + TimerX

TABLE 12: CODE CALCULATIONS

Modules	Limit High	Resolution
CLC	65.54 ms	62.5ns
NCO		
TimerX		

This method uses the NCO and CLC to clock the NCO accumulator when the pulse is High. Any timer can be used to increment when the other modules are running. For best accuracy, this should be used to measure duty cycles of a waveform over an extended period of time of at least 100 periods.

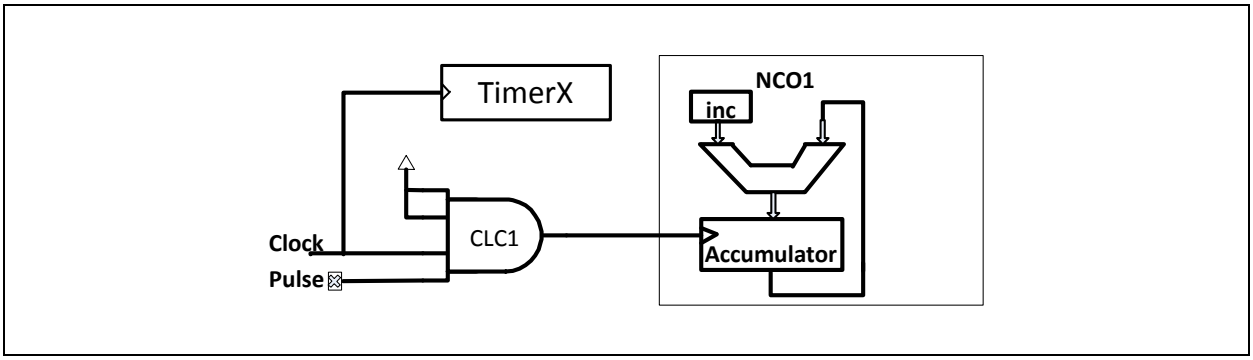
OVERVIEW

When the timer is enabled, immediately enable the NCO to start accumulating with an increment value of one. When the waveform of interest is High, the NCO will increment with every clock edge until the waveform goes Low.

The NCO is disabled to conclude the measurement. This can be initiated in multiple ways such as an interrupt on the timer or NCO accumulator overflow. If an overflow has occurred, the maximum value of that register should be used and not its current value since the interrupt would not stop it from incrementing from zero.

It is important that the same clock source (Fosc) is used between the CLC and timer to avoid post processing. For example, Timer2 can only utilize Fosc/4 as its clock input. Two left shifts on the resultant timer value can compensate for this by multiplying its value by 4.

FIGURE 16: HIGH LEVEL VIEW OF THE INTERACTION BETWEEN CLC1 AND NCO. ALL OF THE CONNECTIONS ARE INTERNAL TO THE PROCESSOR EXCEPT THE PULSE



The duty cycle is subsequently the 20-bit NCO value divided by the timer value (assuming both the timer and NCO increment by 1 on the same clock source with a 1:1 ratio).

EQUATION 1: Duty Cycle Calculation for NCO

$$Duty\ cycle = \frac{AccumulatorVal}{timerVal}$$

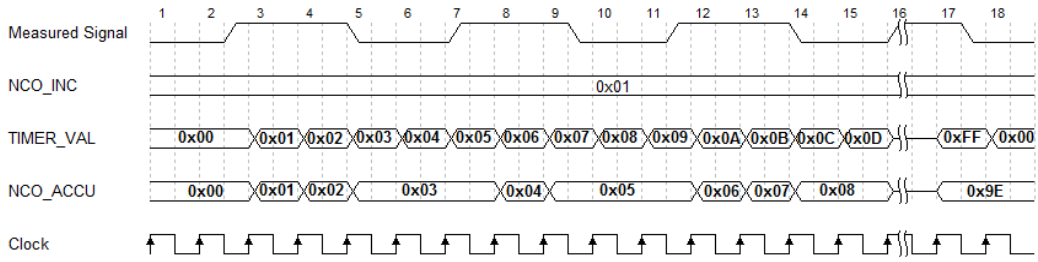
One important consideration is the timer rollover length. The timer must not expire before the waveform makes a complete period.

SETUP

1. Setup the NCO in FDC or PFC mode with LC1OUT as the clock source
2. Setup CLC as a 4-input AND gate
   a) Inputs 1 and 2 set to V_{DD} (invert Gate 1 and 2 output)
   b) Input 3 is the clock source (HFINTOSC)
   c) Input 4 is the Pulse

Figure 17 shows a timing diagram as an implementation of Figure 16.

FIGURE 17: MEASUREMENT OF DUTY CYCLE IS THE RATIO OF THE NCO ACCUMULATOR VALUE AND TIMER VALUE



When the signal is High and upon a rising edge of the clock, the NCO increments its accumulator value by 1. The timer is continuously incrementing until it is disabled (not shown). The waveform has an approximate duty cycle of 60% (3 High counts / 5 total counts). A more realistic waveform may have its period many times greater than this 5 clock counts.

The measured duty cycle is then:

EQUATION 2: Duty Cycle Calculation for NCO

$$Duty\ cycle = \frac{AccumulatorVal}{TimerVal} = \frac{158}{255}(100) = 62\ \%$$

There is uncertainty associated with this measurement due to when the timer starts and stops incrementing relative to when the waveform period starts and ends. This uncertainty fluctuates with the placement of the timer overflow. If the timer is disabled just before the start of the second pulse (not pictured), then the duty cycle would appear to be much less than it is.

For this reason, this strategy should be employed to be counting during much of the waveform as possible. As an example, a 50 kHz square wave that goes through 100 periods, will have an accuracy of one percent, or rather one period length. This would require that the timer be counting for at least two milliseconds.

EQUATION 3: Accuracy of this Method is Dependent on Number of Periods Measured

$$Accuracy = \frac{\#\ of\ periods\ -1}{\#\ of\ periods}(100)$$

$$Timer\ Length = \frac{1}{Period} * \#\ of\ periods$$

CCP

TABLE 13: CCP CODE CALCULATIONS

Modules	Limit High	Resolution
CCP1	16.38 ms	3 us
Timer1/3		

The CCP module can also measure the duty cycle of a waveform. This has a similar approach to **Section "CCP"** under pulse measurement.

OVERVIEW

Another method to measure the duty cycle is setting up the CCP as in **Section "CCP"** under pulse measurement, and then have the module to interrupt on 4 or 16 edges, calculate the period, and divide the two results to get the ratio.

SETUP

See **Section "CCP"** Setup to get both the period and pulse width.

LIMITATIONS

Any absolute errors incurred from the pulse measurement are canceled. For best results a large number of counts should be captured during the duration of the capture. For example, the CCP must have at least 10 counts for 10% accuracy.

IOC (Interrupt-On-Change) With Timer

TABLE 14: IOC WITH TIMER CODE CALCULATIONS

Modules	Limit High	Resolution
IOC/INT	4.1 ms	8 us
TimerX		

This method is cheap in terms of hardware modules used, but at a cost of accuracy. This strategy incorporates the pulse measurement using interrupt-on-change in section (see **Section "IOC (Interrupt-On-Change) With Timer"** in Pulse Management).

OVERVIEW

This method is similar to what the CCP module does, except it cannot latch the timer value when an event is detected (rising/fall edge). This introduces further deviation in the calculation, since software must now save the timer. Some PIC microcontrollers can specify rising and falling edge triggers, while others cannot differentiate between the two. The INT pins can be used, as well as any of the PORTB pins, if the device supports IOC. Refer to Figure 3 for a picture of what is being measured.

SETUP

After executing the setup in **Section "IOC (Interrupt-On-Change) With Timer"**, the duty cycle percentage is then:

EQUATION 4: DUTY CYCLE PERCENTAGE EQUATION

$$Duty\ cycle\ (\%) = \frac{W}{T} * (100)$$

LIMITATIONS

This is the same resolution as the pulse measurements.

IOC (Interrupt-On-Change) Without Timer

TABLE 15: IOC WITHOUT TIMER CODE CALCULATIONS

Modules	Limit High	Resolution
IOC/INT	1.073s	12 us

This strategy uses the IOC feature to simply increment a single register in lieu of a timer.

SUMMARY

Without a timer, the microcontroller must spend its time polling and incrementing registers. This forces the CPU to be stuck inside of the software routine until the duty cycle is finished being measured. Refer to Figure 3 for a picture of what is being measured.

SETUP

After executing the setup in **Section "IOC (Interrupt-On-Change) Without Timer"**, the duty cycle percentage can be then found from Equation 4.

LIMITATIONS

This is the same resolution as the pulse measurements. The uncertainty is plus or minus two clock periods.

Polled Input

TABLE 16: NONE CODE CALCULATIONS

Modules	Limit High	Resolution
NONE	1.073s	12 us

This is the most basic routine. It involves no peripherals, yet it is the most time consuming.

SUMMARY

This method simply uses software to poll and increment variables to measure the duty cycle.

SETUP

After performing the steps seen in **Section "Polled Input"**, the duty cycle percentage can then be found from Equation 4.

LIMITATIONS

The resolution is similar to the previous software routines — proportional to the number of instructions to setup each rising and falling edge detection, as well as increment routine.

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Trademarks

Printed on recycled paper.

ISBN: 9781620767252

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

Toronto
Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Hangzhou
Tel: 86-571-2819-3187
Fax: 86-571-2819-3189

China - Hong Kong SAR
Tel: 852-2943-5100
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

Japan - Tokyo
Tel: 81-3-6880- 3770
Fax: 81-3-6880-3771

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-5778-366
Fax: 886-3-5770-955

Taiwan - Kaohsiung
Tel: 886-7-213-7828
Fax: 886-7-330-9305

Taiwan - Taipei
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

11/27/12