Introduction to PIC Programming

Baseline Architecture and Assembly Language

by David Meiklejohn, Gooligum Electronics

Lesson 9: Analog Comparators

We've seen how to respond to simple on/off digital signals, but we live in an "analog" world; many of the sensors you will want your PIC-based projects to respond to, such as temperature or light, have smoothly varying outputs whose magnitude represents the value being measured; they are *analog* outputs.

In many cases, you will want to treat the output of an analog sensor as though it was digital (i.e. on or off, high or low), without worrying about the exact value. It may be that a pulse, that does not meet the requirements for a "digital" input, has to be detected: for example, the output of a Hall-effect sensor attached to a rotating part. Or we may simply wish to respond to a threshold (perhaps temperature) being crossed.

In such cases, where we only need to respond to an input voltage being higher or lower than a threshold, it is usually appropriate to use an analog *comparator*. In fact, analog comparators are so useful that most PICs include them, as built-in peripherals.

This lesson explains how to use comparators, on baseline PIC devices[1], to respond to analog inputs.

In summary, this lesson covers:

- Using comparators to compare voltage levels
- Adding comparator hysteresis
- Using a comparator to drive Timer0
- Using absolute and programmable voltage references

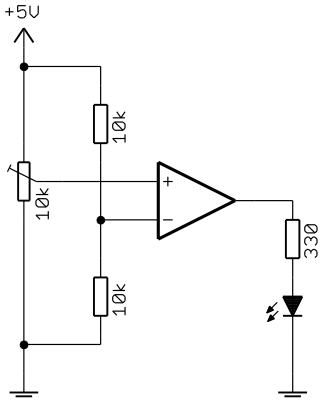
Comparators

A *comparator* (technically, an analog comparator, since comparators with digital inputs also exist) is a device which compares the voltages present on its positive and negative inputs. If the voltage on the positive input is greater than that on the negative input, the output is set "high"; otherwise the output is "low".

An example is shown in the diagram on the right.

The two 10 kΩ resistors act as a voltage divider, presenting 2.5 V at the comparator's negative input. This sets the on/off threshold voltage.

The potentiometer can be adjusted to set the positive input to any voltage between 0 V and 5 V.



[1] The comparators on most baseline PICs (those with comparators), including the 10F2xx series, are very similar to those in the PIC16F506, used in this lesson.

When the potentiometer is set to a voltage under 2.5 V, the comparator's output will be low and the LED will not be lit. But when the potentiometer is turned up past halfway, the comparator's output will go high, lighting the LED.

Comparators are typically used when a circuit needs to react to a sensor's analog output being above or below some threshold, triggering some event (e.g. time to fill the tank, turn off a heater, or start an alarm).

They are also useful for level conversion. Suppose a sensor is outputting pulses which are logically "on/off" (i.e. the output is essentially digital), but do not match the voltage levels needed by the digital devices they are driving. For example, the digital inputs of a PIC, with V_{DD} = 5 V (i.e. TTL-compatible), require at least 2.0 V to register as "high". That's a problem if a sensor is delivering a stream of 0 - 1 V pulses. By passing this signal through a comparator with an input threshold of 0.5 V, the 1 V pulses would be recognised as being "high".

Similarly, comparators can be used to *shape* or *condition* poorly defined or slowly-changing signals. The logic level of a signal between 0.8 V and 2.0 V is not defined for digital inputs on a PIC with V_{DD} = 5 V. And excessive current can flow when a digital input is at an intermediate value. Digital input signals which spend any significant amount of time in this intermediate range should be avoided. Such input signals can be cleaned up by passing them through a comparator; the output will have sharply-defined transitions between valid digital voltage levels.

The PIC16F506 includes two comparators, having different capabilities, as discussed in the following sections.

Comparator 1

Comparator 1 is the simpler of the two comparators.

It is controlled by the CM1CON0 register:

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CM1CON0	C1OUT	$\overline{\text{C1OUTEN}}$	C1POL	$\overline{\text{C1T0CS}}$	C1ON	C1NREF	C1PREF	$\overline{\text{C1WU}}$

The comparator's inputs are determined by C1NREF and C1PREF.

C1PREF selects which pin will be used as the *positive reference* (or input):

C1PREF = 1 selects C1IN+

C1PREF = 0 selects C1IN-

[Note that, despite its name, the C1IN- pin can be used as the comparator's positive reference.]

C1NREF selects the *negative reference*:

C1NREF = 1 selects the C1IN- pin

C1NREF = 0 selects a 0.6 V internal reference voltage

By default (after a power-on reset), every bit of CM1CON0 is set to '1'.

This selects the C1IN+ pin as the positive reference, and C1IN- as the negative reference – the "normal" way to use the comparator.

Alternatively, the internal 0.6 V *absolute* reference can be selected as the negative reference, freeing up one I/O pin, with either C1IN+ or C1IN- providing the positive reference. Selecting C1IN+ as the positive reference is clearer, but occasionally it might make more sense to use C1IN-, perhaps because it simplifies your PCB layout, or you may be using comparator 1 for multiple measurements, alternately comparing C1IN- with 0.6 V, and then C1IN+ with C1IN-. For example, the 0.6 V reference could be used to solve the problem of detecting 0 - 1 V pulses, mentioned earlier.

The comparator's output appears as the C1OUT bit: it is normally set to '1' if and only if the positive reference voltage is higher than the negative reference voltage.

That's the normal situation, but the operation of the comparator can be inverted by clearing the C1POL output *polarity* bit:

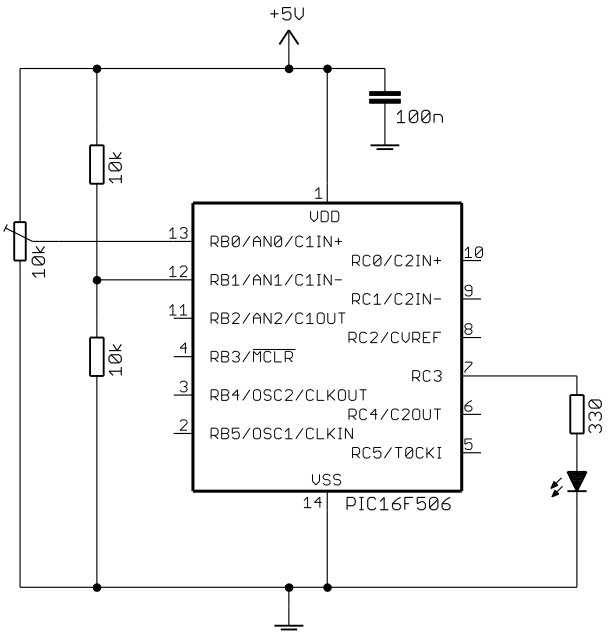
C1POL = 1 selects normal operation

C1POL = 0 selects inverted operation, where C1OUT = 1 only if positive ref < negative ref.

Finally, the C1ON bit turns the comparator on or off: '1' to turn it on, '0' to turn it off.

Those are the only bits needed for basic operation of comparator 1. We'll examine the other bits in the CM1CON1 register, later.

We will illustrate the comparator's basic operation using the circuit shown below.



If you have the Gooligum baseline training board, you can implement this circuit by placing a shunt across pins 1 and 2 ('POT') of JP24, connecting the 10 kΩ pot (RP2) to C1IN+, and in JP19 to enable the LED on RC3.

The connection to C1IN- (labelled 'GP/RA/RB1' on the board) is available as pin 9 on the 16-pin header. +V and GND are brought out on pins 15 and 16, respectively, making it easy to add the 10 kΩ resistors (supplied with the board), forming a voltage divider, by using the solderless breadboard.

The circuit can alternatively be built using the Microchip Low Pin Count Demo Board.

The 10 kΩ potentiometer on that board is already connected to C1IN+ (labelled 'RA0' on the board), via a 1 kΩ resistor (not shown in this diagram). But you must ensure that jumper JP5 is closed; it will be, if you haven't modified your demo board. The LED labelled 'DS4' on the demo board is connected to RC3 (via a 470 Ω resistor, instead of 330 Ω as shown here, but that makes no difference) via jumper JP4.

You can connect 10 kΩ resistors (which you will need to provide, along with a breadboard or other prototyping board) via the 14-pin header on the demo board. C1IN- (labelled 'RA1') is available on pin 8, and +V and GND are pins 13 and 14 respectively.

It is straightforward to configure the PIC as a simple comparator, turning on the LED if the potentiometer is turned more than halfway toward the +5V supply (right) side.

First configure RC3 as an output:

```
; configure ports
movlw   ~(1<<nLED)      ; configure LED pin (only) as an output
tris    PORTC
```

There is no need to configure RB0 or RB1 as inputs, as the comparator settings override TRISB.

Next, we can configure comparator 1:

```
        ; configure comparator 1
        movlw   1<<C1PREF|1<<C1NREF|1<<C1POL|1<<C1ON
                                ; +ref is C1IN+ (C1PREF = 1)
                                ; -ref is C1IN- (C1NREF = 1)
                                ; normal polarity (C1POL = 1)
                                ; comparator on (C1ON = 1)
        movwf   CM1CON0         ; -> C1OUT = 1 if C1IN+ > C1IN-
```

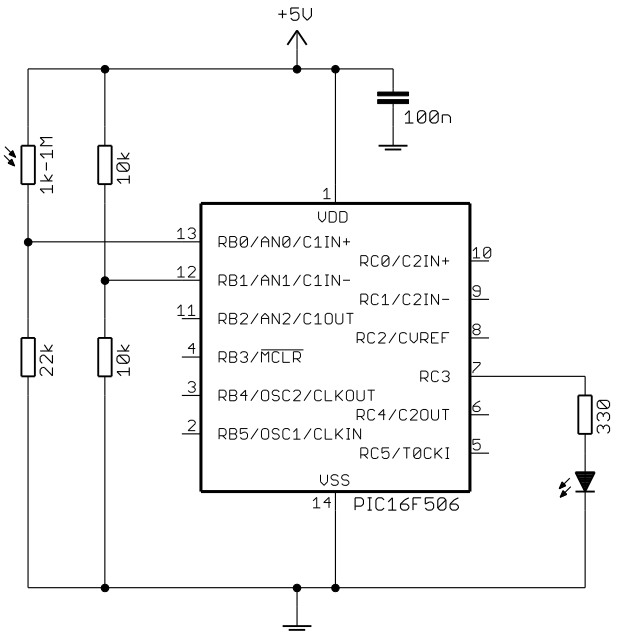
This turns comparator 1 on, and configures it to test for C1IN+ > C1IN-. This is the default setting, so in theory these initialisation instructions are unnecessary. But relying implicitly on default settings is obscure and error-prone; it is much clearer to explicitly initialise the registers for the functions you are using.

To turn on the LED when the comparator output is high, repeatedly test C1OUT:

```
loop    btfsc   CM1CON0,C1OUT   ; if comparator output high
        bsf     LED             ;   turn on LED
        btfss   CM1CON0,C1OUT   ; if comparator output low
        bcf     LED             ;   turn off LED

        goto    loop            ; repeat forever
```

You should find that, as you turn the potentiometer past halfway, the LED turns on and off

To make the circuit a little more interesting, we'll add a light-dependent resistor (LDR, or CdS photocell), as shown on the right.



As the light level increases, the resistance of the LDR falls, and the voltage at the potential divider (formed by the LDR and the 22 kΩ resistor) connected to C1IN+ rises.

The exact resistance range of the photocell is not important; the ones supplied with the Gooligum baseline training board have a resistance of around 20 kΩ or so for normal indoor lighting conditions, which is why a 22 kΩ resistor is used for the lower arm of the potential divider here – the voltage at C1IN+ will vary around 2.5 V or so when it's not too bright, not too dark.

If you are using the Gooligum baseline training board, you only need to move the shunt in JP24 across to pins 2 and 3 ('LDR1'), connecting the photocell (PH1) and 22 kΩ resistor in the lower left of the board to C1IN+.

If you are using the Microchip Low Pin Count Demo Board, you can't take the 10 kΩ potentiometer out of the circuit – it, and the 1 kΩ resistor in series between it and C1IN+, must be used as the "fixed" resistance, forming the lower arm of the potential divider. You must also remove jumper JP5 (you may need to cut the PCB trace – ideally you'd install a jumper, so that you can reconnect it again later), to disconnect the pot from the +5 V supply.

If you turn the pot all the way to the right, you'll have a total resistance of 11 kΩ between C1IN+ and ground. That means that ideally you'd use a photocell with a resistance of around 10 kΩ with normal indoor lighting. The photocell can then be connected between pin 7 on the 14-pin header and +5 V.

If you don't have a photocell, there is no problem with continuing to use the potentiometer-only circuit for these lessons; but it's certainly more fun to build a circuit that responds to light!

When you build this circuit and use the above code, you will find that the LED turns on when the LDR is illuminated.

If you would prefer it to work the other way, so that the LED is lit when the light level falls (simulating e.g. the way that streetlamps turn on automatically when it gets dark), there's no need to change the circuit connections or program logic. Simply change the comparator initialisation instructions, to invert the output, by clearing the C1POL bit:

```
; configure comparator 1
movlw   1<<C1PREF|1<<C1NREF|0<<C1POL|1<<C1ON
                        ; +ref is C1IN+ (C1PREF = 1)
                        ; -ref is C1IN- (C1NREF = 1)
                        ; inverted polarity (C1POL = 0)
                        ; turn comparator on (C1ON = 1)
movwf   CM1CON0         ; -> C1OUT = 1 if C1IN+ < C1IN-
```

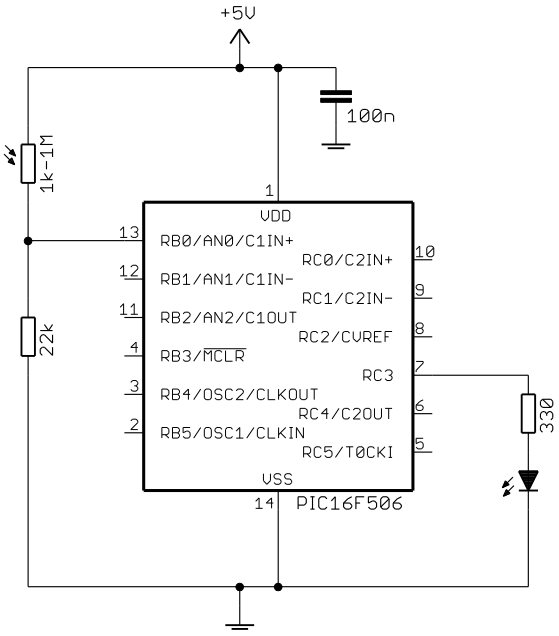
We can save an I/O pin if we don't use an external voltage divider as the negative reference; we could instead use the 0.6 V internal reference, which is enabled by clearing the C1NREF bit:

```
; configure comparator 1
movlw   1<<C1PREF|0<<C1NREF|0<<C1POL|1<<C1ON
                        ; +ref is C1IN+ (C1PREF = 1)
                        ; -ref is 0.6 V (C1NREF = 0)
                        ; inverted polarity (C1POL = 0)
                        ; turn comparator on (C1ON = 1)
movwf   CM1CON0         ; -> C1OUT = 1 if C1IN+ < 0.6 V
```

Since the negative reference is now 0.6 V instead of 2.5 V, you will find that you will need to make it darker than before, to make the LED come on.

To convince yourself that it really is the internal reference voltage being used, you could remove the 10 kΩ voltage divider resistors, as shown on the right. You should find that removing the resistors makes no difference to the circuit's operation.

The important difference is that RB1 is now available for use.



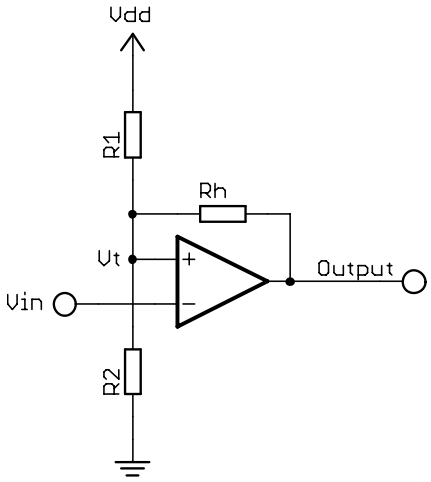
Adding hysteresis

You will notice, as the light level changes slowly past the threshold where the LED turns on and off, that the LED appears to fade in and out in brightness. This is caused by noise in the circuit and fluctuations in the light source (particularly at 50 or 60 Hz, from mains-powered incandescent or fluorescent lamps): when the input is very close to the threshold voltage, small input voltage variations due to noise and/or fluctuating

light levels cause the comparator to rapidly switch between high and low. This rapid switching, similar to switch bounce, can be a problem if the microcontroller is supposed to count input transitions, or to perform an action on each change.

To avoid this phenomenon, *hysteresis* can be added to a comparator, by adding positive feedback – feeding some of the comparator's output back to its positive input.

Consider the comparator circuit shown on the right.



The threshold voltage, Vt, is set by a voltage divider, formed by resistors R1 and R2.

This would normally set the threshold at $Vt = \frac{R2}{R1+R2}Vdd$.

However, resistor Rh feeds some of the comparator's output back, increasing the threshold to some higher level, Vth, when the output is high, and decreasing it to a lower level, Vtl, when the output is low.

Now consider what happens when Vin is low (less than Vtl) and begins to increase. Initially, since Vin < Vt, the comparator output is high, and the threshold is high: Vt = Vth.

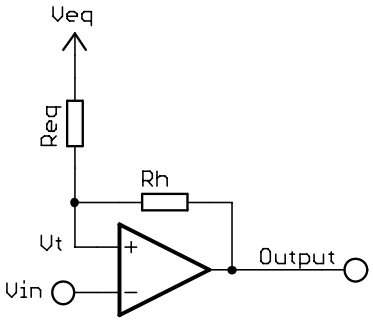
Eventually Vin rises above Vth, and the comparator output goes low, lowering the threshold: Vt = Vtl.

Now suppose the input voltage begins to fall. As it falls past the high threshold, Vth, nothing happens. It has to keep falling, all the way down to the low threshold, Vtl, before the comparator output changes again.

There are now two voltage thresholds: one (the higher) applying when the input signal is rising; the other (lower) applying when the input is falling. The comparator's output depends not only on its inputs, but on their history – a key characteristic of hysteresis.

The voltage difference between the two thresholds is known as the hysteresis band: Vhb = Vth – Vtl.

This is the amount the input signal has to fall, after rising through the high threshold, or rise, after falling through the low threshold, before the comparator output switches again. It should be higher than the expected noise level in the circuit, making the comparator immune to most noise.



To calculate the high and low thresholds, recall Thévenin's theorem, which states that any two-terminal network of resistors and voltage sources can be replaced by a single voltage source, Veq, in series with a single resistor, Req.

Thus, the circuit above is equivalent to that on the left, where

$$Veq = \frac{R2}{R1+R2}Vdd$$

and Req is the parallel combination of R1 and R2: $Req = \frac{R1R2}{R1+R2}$

Therefore, when the comparator's output is low (= 0 V):

$$Vtl = \frac{Rh}{Rh+Req}Veq \qquad \text{(thus Vtl < Veq)}$$

and, when the comparator's output is high (= Vdd):

$$Vth = Veq + \frac{Req}{Rh + Req}(Vdd - Veq) \quad \text{(thus Vth > Veq)}$$

This little bit of mathematics proves that Vth > Vtl, that is, the high input threshold is higher than the low input threshold.

The output of comparator 1 can be made available on the C1OUT pin (shared with RB2); we can use this to add hysteresis to the circuit.

The comparator output is enabled by clearing the $\overline{\text{C1OUTEN}}$ bit in the CM1CON0 register:

$\overline{\text{C1OUTEN}}$ = 0 places the output of comparator 1 on the C1OUT pin

$\overline{\text{C1OUTEN}}$ = 1 disables comparator output on C1OUT (i.e. normal operation).

> *Note: The comparator output overrides digital I/O. To use a pin for digital I/O, any comparator output assigned to that pin must be disabled. Comparator outputs are disabled on start-up.*

In most examples of comparator hysteresis, the comparator's positive input is used as the threshold, and feedback is used to alter that threshold, as shown above.

However, in the example above, where the internal 0.6 V reference is used as the negative reference, it is not possible to use feedback to adjust the threshold; being an internal reference, there is no way to affect it.

But that's not a problem – it is also possible to introduce hysteresis by feeding the comparator output into the input signal (i.e. the signal being measured), assuming the input is connected to the positive input. It may not seem as intuitive, but the principle is essentially the same.

C1 on: C1IN+ < 0.6VWhen the input is higher than the threshold, the comparator output goes high, pulling the input even higher, via the feedback resistor. The circuit driving the input then has to effectively work harder, against the comparator's output, to bring the input back below the threshold. Similarly, when the input is low, the comparator's output goes low, dragging the input even lower, meaning that the input drive has to increase further before the comparator will change state again.

Suppose there is no feedback resistor and that the voltage on C1IN+ is equal to the threshold voltage of 0.6 V. This would happen if the light level is such that the LDR has a resistance of 161.3 kΩ:

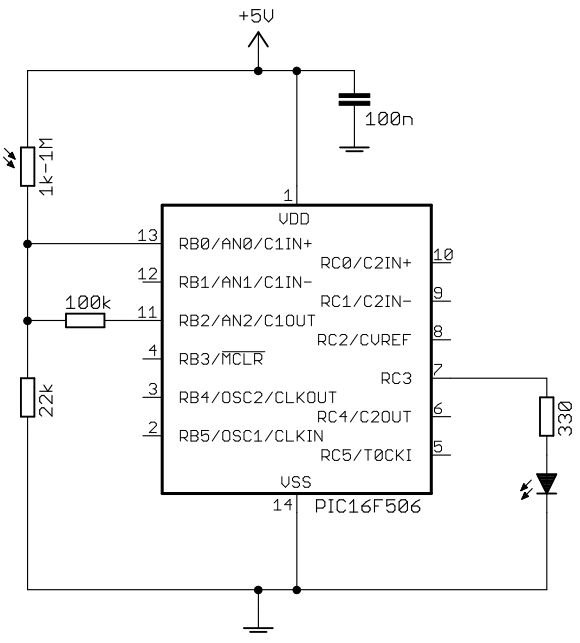
Vin = 22 / (22 + 161.3) × 5 V = 0.6 V

With the input so close to the threshold, we would expect the output to jitter.

Now suppose that we add a 100 kΩ feedback resistor, as shown on the right.

You can do this with the Gooligum baseline training board by placing the supplied 100 kΩ resistor between pins 8 ('GP/RA/RB0') and 13 ('GP/RA/RB2') on the 16-pin header.

Or, if you are using the Microchip Low Pin Count Demo Board, you would place the feedback resistor between pins 7 and 9 on the 14-pin header.



The effect is similar to that for threshold voltage feedback. At that level of illumination, the potential divider formed by the LDR and the 22 kΩ resistor is equivalent to a source (input) voltage of 0.6 V, connected to C1IN+ via a resistance of:

Req = (161.3 × 22) / (161.3 + 22) kΩ = 19.4 kΩ

When the comparator output is low, the feedback resistor pulls the input voltage down to:

Vinl = 100 / (100 + 19.4) × 0.6 V = 0.50 V

When the comparator is high, the input voltage is pulled up to:

Vinh = 0.6V + 19.4 / (100 + 19.4) × (5.0 V – 0.6 V) = 1.3 V

Note that these voltages are not input thresholds. The comparator threshold is still the internal voltage reference of 0.6 V. These calculations only serve to demonstrate that the addition of positive feedback pulls the input lower when the comparator output is low and higher when the output is high, making the input less sensitive to small changes.

The code is essentially the same as before; since $\overline{\text{C1OUTEN}}$ must be cleared to enable the comparator output pin, that bit should not be set when CM1CON0 is loaded. Since the symbol for it (NOT_C1OUTEN) hasn't been included in the examples so far, it has been cleared (through omission), so C1OUT has in fact always been enabled so far.

However, it is very important to set the C1POL bit. If it is not set, the comparator output, appearing on C1OUT, will be inverted, and the feedback will be negative, instead of the positive feedback needed to create hysteresis.

Instead, we can have the LED continue to indicate low light by inverting the display logic: light the LED only when C1OUT = 0.

But before the comparator output can actually appear on the C1OUT pin, that pin has to be deselected as an analog input.

> *Note: To enable comparator output on a pin, any analog input assigned to that pin must first be disabled.*

Deselecting analog inputs is explained in the next lesson on analog-to-digital conversion, but for now all we need to remember is that all the analog inputs can be disabled by clearing the ADCON0 register.

The code for turning on an LED on RC3, when the photocell is not illuminated, using the internal 0.6 V reference, with hysteresis, is:

```
; configure ports
movlw   ~(1<<nLED)      ; configure LED pin (only) as an output
tris    PORTC
clrf    ADCON0          ; disable analog inputs -> C1OUT usable

; configure comparator 1
movlw   1<<C1PREF|0<<C1NREF|1<<C1POL|0<<NOT_C1OUTEN|1<<C1ON
                        ; +ref is C1IN+ (C1PREF = 1)
                        ; -ref is 0.6 V (C1NREF = 0)
                        ; normal polarity (C1POL = 1)
                        ; enable C1OUT pin (/C1OUTEN = 0)
                        ; turn comparator on (C1ON = 1)
movwf   CM1CON0         ; -> C1OUT = 1 if C1IN+ > 0.6V,
                        ;    C1OUT pin enabled
```

```
;***** Main loop
main_loop
        ; display comparator output (inverted)
        btfsc   CM1CON0,C1OUT   ; if comparator output high
        bcf     LED             ;   turn off LED
        btfss   CM1CON0,C1OUT   ; if comparator output low
        bsf     LED             ;   turn on LED

        ; repeat forever
        goto    main_loop
```

Wake-up on comparator change

As we saw in lesson 7, most PICs can be put into standby, or sleep mode, to conserve power until they are woken by an external event. We've seen that that event can be a change on a digital input; it can also be a change on comparator output.

That's useful if your application is battery-powered and has to spend a long time waiting to respond to an input level change from a sensor.

Wake-up on change for comparator 1 is controlled by the $\overline{\text{C1WU}}$ bit in the CM1CON0 register.

By default (after a power-on reset), $\overline{\text{C1WU}}$ = 1 and wake-up on comparator change is disabled.

To enable wake-up on comparator change, clear $\overline{\text{C1WU}}$.

> *Note: You should read the output of the comparator configured for wake-up on change just prior to entering sleep mode. Otherwise, if the comparator output had changed since the last time it was read, a "wake up on comparator change" reset will occur immediately upon entering sleep mode.*

Since the symbol for setting $\overline{\text{C1WU}}$ (NOT_C1WU) has not been included in the expression loaded into CM1CON0 in any of the examples so far, $\overline{\text{C1WU}}$ has always been cleared (by omission), and wake-up on comparator change has been implicitly enabled every time.

But to explicitly enable wake-up on comparator change, you should use an expression such as:

```
        movlw   1<<C1PREF|1<<C1NREF|0<<C1POL|0<<NOT_C1WU|1<<C1ON
        movwf   CM1CON0
```

To determine whether a reset was due to a comparator change, test the CWUF flag in the STATUS register:

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STATUS	RBWUF	CWUF	PA0	$\overline{\text{TO}}$	$\overline{\text{PD}}$	Z	DC	C

CWUF = 1 indicates that a wake-up from sleep on comparator change reset occurred

CWUF = 0 after all other resets.

Although there are two comparators in the 16F506, this flag doesn't tell you which comparator's output changed; by default (after a power-on reset) wake-up on change for each comparator is disabled, but if you have enabled wake-up on change for both, you need to have stored the previous value of each comparator's output so that you can test to see which one changed. It's the same situation as for wake-up on change for the digital inputs; there is only one flag (GPWUF/RBWUF), but potentially a number of inputs that may have changed to trigger the wake-up. The only way to know what has changed is to compare each comparator output against the previously-recorded value.

The following example program configures Comparator 1 for wake-up on change and then goes to sleep. When the comparator output changes, the PIC resets, the wake-up on change condition is detected, and a LED is turned on for one second, to indicate that the comparator output has changed. The PIC then goes back to sleep, to wait until the next comparator output change.

We'll keep the same circuit as before, with the feedback resistor providing hysteresis in place, to make the comparator less sensitive to small variations – we only want the PIC to wake on a significant change, as you move the photocell between light to dark and back again.

Note that, when the comparator is initialised, there is a delay of 10 ms to allow it to settle before entering standby. This is necessary because signal levels can take a while to settle after power-on, and if the comparator output was changing while going into sleep mode, the PIC would immediately wake and the LED would flash – a false trigger. Note also that the comparator output is read (any instruction which reads CM1CON0 will do) immediately before the `sleep` instruction is executed, as explained above.

The delays are generated by the `DelayMS` macro, introduced in lesson 6.

```
start
        ; configure ports
        clrf    PORTC           ; start with LED off
        movlw   ~(1<<nLED)      ; configure LED pin (only) as an output
        tris    PORTC
        clrf    ADCON0          ; disable analog inputs -> C1OUT usable

        ; check for wake-up on comparator change
        btfsc   STATUS,CWUF     ; if wake-up on comparator change occurred,
        goto    flash           ;   flash LED then sleep

        ; else power-on reset
        movlw   b'00111010'     ; configure comparator 1:
                ; -0------          enable C1OUT pin (/C1OUTEN = 0)
                ; --1-----          normal polarity (C1POL = 1)
                ; ----1---          turn comparator on (C1ON = 1)
                ; -----0--          -ref is 0.6 V (C1NREF = 0)
                ; ------1-          +ref is C1IN+ (C1PREF = 1)
                ; -------0          enable wake on comparator change (/C1WU = 0)
        movwf   CM1CON0         ;   -> C1OUT = 1 if C1IN+ > 0.6V,
                                ;      C1OUT pin enabled,
                                ;      wake on comparator change enabled

        DelayMS 10              ; delay 10 ms to allow comparator to settle

        goto    standby         ; sleep until comparator change

;***** Main code
        ; flash LED
flash   bsf     LED             ; turn on LED
        DelayMS 1000            ; delay 1 sec

        ; sleep until comparator change
standby bcf     LED             ; turn off LED
        movf    CM1CON0,w       ; read comparator to clear mismatch condition
        sleep                   ; enter sleep mode
```

Note that, in this example, the binary value '`b'01111010`' was used to initialise the comparator control register, instead of the equivalent (but much longer!) expression:
'`1<<C1PREF|0<<C1NREF|1<<C1POL|1<<NOT_C1OUTEN|0<<NOT_C1WU|1<<C1ON`'.

Either style is ok, as long as it is clearly commented.

Finally, you should be aware that, if a comparator is turned on when the PIC enters sleep mode, it will continue to draw current. If the comparator is to wake the PIC up when an input level changes, then of course it has to remain active, using power, while the PIC is in standby. But if you're not using wait on comparator change, you should turn off all comparators (clear C1ON to turn off comparator 1) before entering sleep mode, to save power.

> *Note: To minimise power consumption, turn comparators off before entering sleep mode.*

Incrementing Timer0

We saw in lesson 5 that Timer0 can be used as a counter, clocked by an external signal on T0CKI.

That's useful, but what if you want to count pulses that are not clean digital signals? The obvious answer is to pass the pulses through a comparator (with hysteresis, if the signal is noisy), and then feed the output of the comparator into T0CKI. Indeed, on some PICs, you need to make an electrical connection, external to the PIC, from the comparator output (e.g. the C1OUT pin) to the counter input (e.g. T0CKI).

However, on the 16F506, the output of comparator 1 can drive Timer0 directly, through an internal connection.

To enable the connection from comparator 1 to Timer0, clear the $\overline{\text{C1T0CS}}$ bit in the CM1CON0 register:

$\overline{\text{C1T0CS}}$ = 1 selects T0CKI as the Timer0 counter clock source

$\overline{\text{C1T0CS}}$ = 0 selects the output of comparator 1 as the Timer0 counter clock source

Note that this setting only matters if Timer0 is in counter mode, i.e. OPTION:T0CS = 1.

If Timer0 is in timer mode (T0CS = 0), the timer will be incremented by the instruction clock (FOSC/4), regardless of the setting of $\overline{\text{C1T0CS}}$.

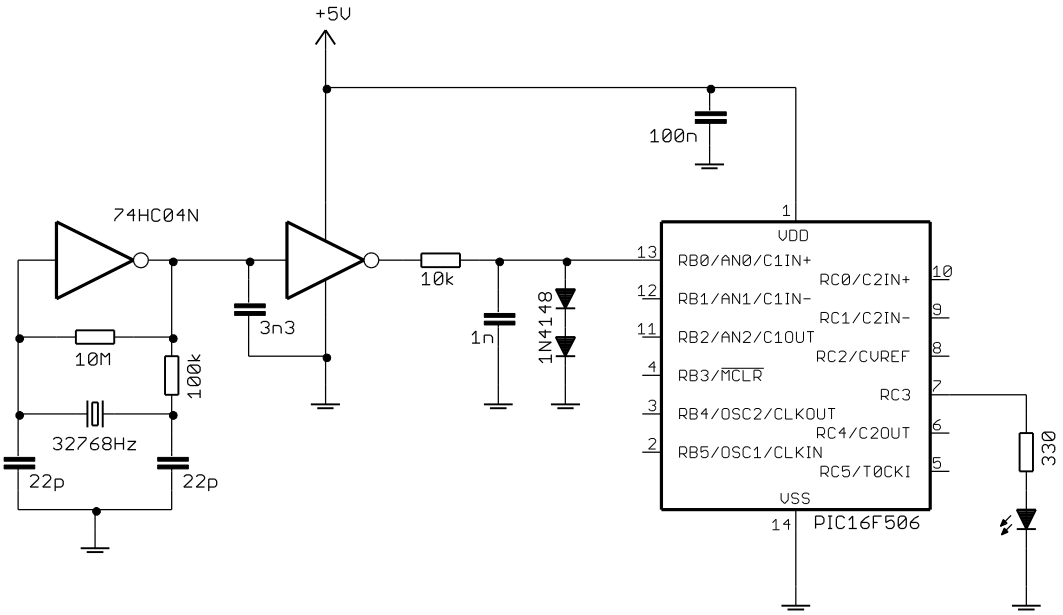
So, to use comparator 1 as the counter clock source, you must set T0CS = 1 and $\overline{\text{C1T0CS}}$ = 0.

In this mode, Timer0 otherwise operates as usual, with the T0SE bit determining whether the counter increments on the rising or falling edge of the comparator output, and the PSA and PS<2:0> bits selecting the prescaler assignment.

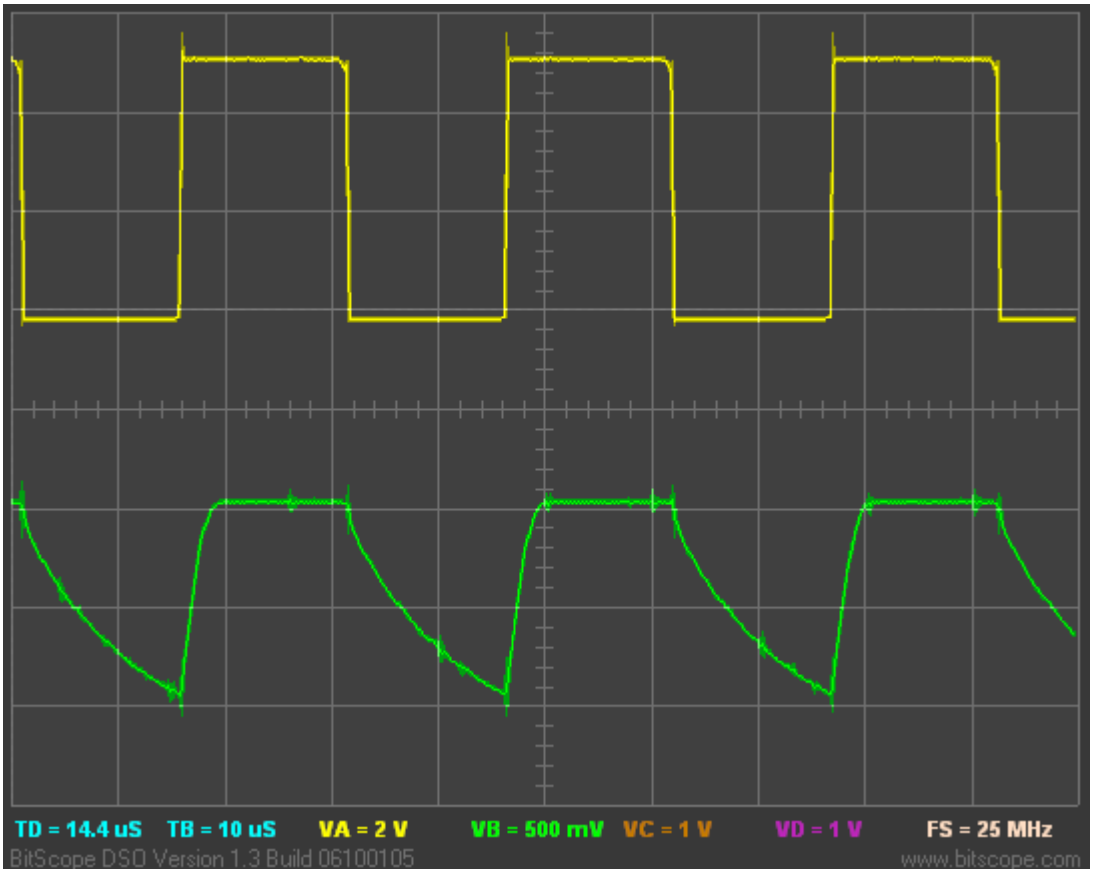
There is one quirk to be aware of: when T0CS is set to '1', the TRIS setting for RC5 is overridden, and RC5 cannot be used as a digital output – regardless of whether T0CKI is actually used as the counter clock source.

> *Note: When using comparator 1 to drive Timer0, RC5 cannot be used as an output – even though the T0CKI function on that pin is not being used.*

To show how comparator 1 can be used with Timer0, we can use the external clock circuit from lesson 5, but with the clock signal degraded by passing it through an RC filter and clamping with two diodes, as shown:

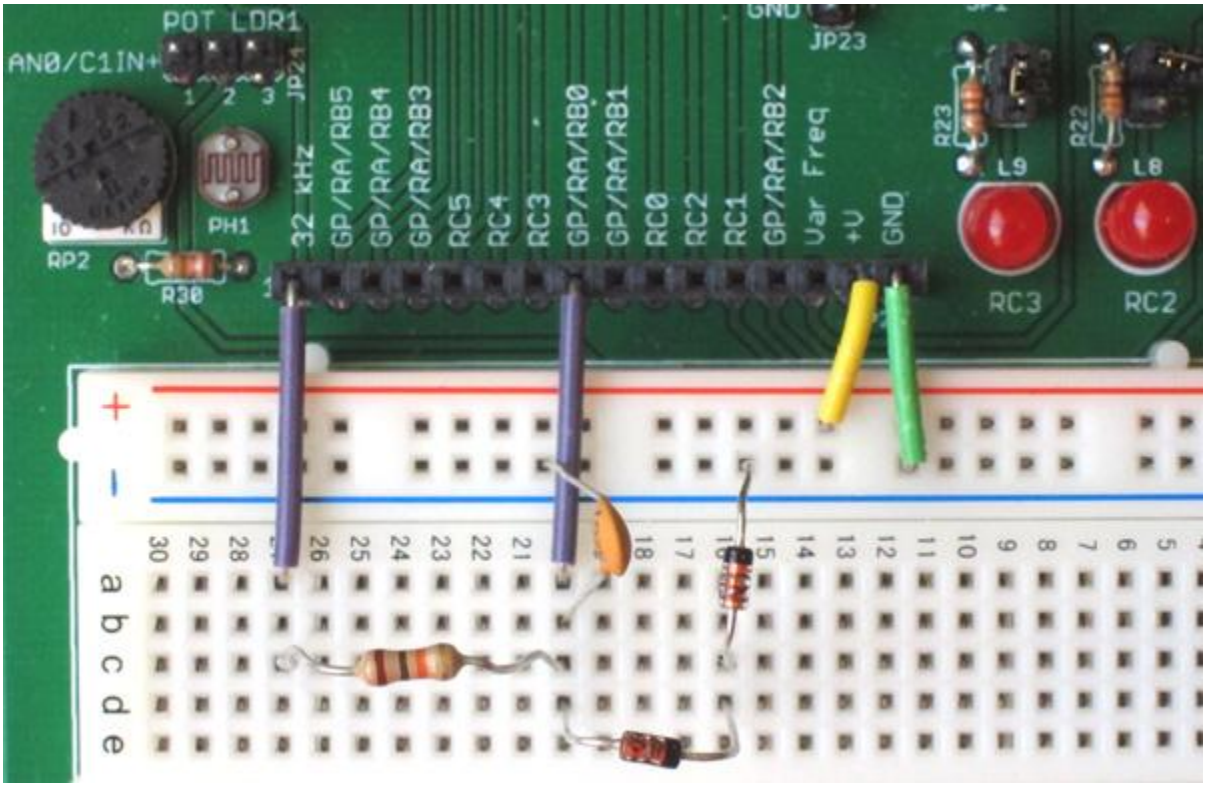


The effect of this can be seen in the oscilloscope trace, below:



The top trace is the "clean" digital output of the clock circuit, and the degraded signal is below. It peaks at approximately 1 V.

You can build this circuit with the Gooligum baseline training board, using the supplied 10 kΩ resistor, 1 nF capacitor and two 1N4148 diodes – connecting them to signals on the 16-pin header: 32.768 kHz oscillator output on pin 1 ('32 kHz'), C1IN+ input on pin 8 ('GP/RA/RB0') and ground on pin16 ('GND') – using the solderless breadboard, as illustrated below:



You should also remove the shunt from JP24 (disconnecting the pot or photocell from C1IN+), but leave JP19 in place (enabling the LED on RC3).

If you are using Microchip's Low Pin Count Demo Board, you can build the circuit in a similar way, by making connections to the 14-pin header on that board, although you will of course have to supply your own 32.768 kHz oscillator. *Note that, if you are using Microchip's LPC Demo Board, the components connected to the C1IN+ input, including the potentiometer, will affect the signal on C1IN+. You may need to adjust the pot to make it work.*

Whichever board you use, you must only connect these additional components to C1IN+ **after** programming the PIC, to avoid interference with the programming process. You need to program the PIC before making the connection to C1IN+. You can then apply power (whether from a PICkit 2, PICkit3, or external power supply) and release reset – and the LED on RC3 should start flashing.

> *Note: Components such as diodes and capacitors connected to the ICSP programming pins (RB0/AN0/C1IN+ and RB1/AN1/C1IN- on a PIC16F506) may interfere with the ICSP programming signals, and must be disconnected before the PIC can be successfully programmed.*

The degraded signal is not suitable for driving T0CKI directly; it doesn't go high enough to register as a digital "high". But it is quite suitable for passing through a comparator referenced to 0.6 V, so the internal reference voltage is a good choice here. Therefore the only comparator input needed is C1IN+, as shown in the circuit on the previous page.

Lesson 5 included a program which flashed an LED at 1 Hz, with Timer0 driven by an external 32.768 kHz clock. It needs very little modification to work with comparator 1 instead of T0CKI. Other than changing references from GPIO to PORTC (since we are now using a 16F506 instead of a 12F509), all we need do is to configure the comparator, with comparator output selected as the Timer0 clock source:

```
        ; configure comparator 1
        movlw   1<<C1PREF|0<<C1NREF|0<<C1POL|0<<NOT_C1T0CS|1<<C1ON
                                    ; +ref is C1IN+ (C1PREF = 1)
                                    ; -ref is 0.6 V (C1NREF = 0)
                                    ; normal polarity (C1POL = 1)
                                    ; select C1 as TMR0 clock (/C1T0CS = 0)
                                    ; turn comparator on (C1ON = 1)
        movwf   CM1CON0             ; -> C1OUT = 1 if C1IN+ > 0.6V,
                                    ;    TMR0 clock from C1
```

Once again, although it is not strictly necessary to include '`0<<NOT_C1T0CS`' in the expression being loaded into CM1CON0 (since ORing a zero value into an expression has no effect), doing so makes it explicit that we are enabling comparator 1 as a clock source, by clearing $\overline{\text{C1T0CS}}$.

Complete program

Here is how the example from lesson 5 (actually the lesson 6 version) has been modified:

```
;*****************************************************************************
;                                                                            *
;   Description:    Lesson 9, example 3                                      *
;                   Crystal-based (degraded signal) LED flasher              *
;                                                                            *
;   Demonstrates comparator 1 clocking TMR0                                  *
;                                                                            *
;   LED flashes at 1 Hz (50% duty cycle),                                    *
;   with timing derived from 32.768 kHz input on C1IN+                       *
;                                                                            *
;*****************************************************************************
;                                                                            *
;   Pin assignments:                                                         *
;       C1IN+ = 32.768 kHz signal                                            *
;       RC3   = flashing LED                                                 *
;                                                                            *
;*****************************************************************************

    list        p=16F506
    #include    <p16F506.inc>

    radix       dec


;***** CONFIGURATION
                ; ext reset, no code protect, no watchdog, 4 MHz int clock
    __CONFIG    _MCLRE_ON & _CP_OFF & _WDT_OFF & _IOSCFS_OFF & _IntRC_OSC_RB4EN

; pin assignments
    constant nFLASH=3               ; flashing LED on RC3


;***** VARIABLE DEFINITIONS
        UDATA_SHR
sPORTC  res 1                       ; shadow copy of PORTC
```

```
;***** RC CALIBRATION
RCCAL   CODE    0x3FF           ; processor reset vector
        res 1                   ; holds internal RC cal value, as a movlw k

;***** RESET VECTOR **************************************************
RESET   CODE    0x000           ; effective reset vector
        movwf   OSCCAL          ; apply internal RC factory calibration


;***** MAIN PROGRAM **************************************************

;***** Initialisation
start
        ; configure ports
        movlw   ~(1<<nFLASH)        ; configure LED pin (only) as output
        tris    PORTC

        ; configure Timer0
        movlw   1<<T0CS|0<<PSA|b'110'
                                    ; counter mode (T0CS = 1)
                                    ; prescaler assigned to Timer0 (PSA = 0)
                                    ; prescale = 128 (PS = 110)
        option                      ; -> incr at 256 Hz with 32.768 kHz input

        ; configure comparator 1
        movlw   1<<C1PREF|0<<C1NREF|0<<C1POL|0<<NOT_C1T0CS|1<<C1ON
                                    ; +ref is C1IN+ (C1PREF = 1)
                                    ; -ref is 0.6 V (C1NREF = 0)
                                    ; normal polarity (C1POL = 1)
                                    ; select C1 as TMR0 clock (/C1T0CS = 0)
                                    ; turn comparator on (C1ON = 1)
        movwf   CM1CON0             ; -> C1OUT = 1 if C1IN+ > 0.6V,
                                    ;    TMR0 clock from C1

;***** Main loop
main_loop
        ; TMR0<7> cycles at 1 Hz, so continually copy to LED (GP1)
        clrf    sPORTC              ; assume TMR0<7>=0 -> LED off
        btfsc   TMR0,7              ; if TMR0<7>=1
        bsf     sPORTC,nFLASH       ;   turn on LED

        movf    sPORTC,w            ; copy shadow to port
        movwf   PORTC

        ; repeat forever
        goto    main_loop


        END
```

Comparator 2

Comparator 2 is quite similar to comparator 1, but a wider range of inputs can be selected and it can be used with the programmable voltage reference.

It cannot, however, be selected as a clock source for Timer0.

It is controlled by the CM2CON0 register:

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CM2CON0	C2OUT	$\overline{\text{C2OUTEN}}$	C2POL	C2PREF2	C2ON	C2NREF	C2PREF1	$\overline{\text{C2WU}}$

Most of these bits are directly equivalent to those in CM1CON0:

C2OUT	is the comparator output bit (the output appears here)
$\overline{\text{C2OUTEN}}$	determines whether the output is placed on the C2OUT pin or not
C2POL	selects the output polarity
C2ON	turns the comparator on or off
$\overline{\text{C2WU}}$	enables/disables wake-up on comparator change

They have the same options and work in the same way as the corresponding bits in CM1CON0.

The C2PREF bits select the positive reference:

C2PREF1	C2PREF2	Positive Reference
0	0	C2IN-
0	1	C1IN+
1	–	C2IN+

Note that C1IN+ (a "comparator 1" input) can be selected as in input to comparator 2.

Furthermore, C1IN+ can be used as an input to both comparators at the same time – something that can be useful if you want to compare two signals (one for each comparator) against a common external (perhaps varying) reference on C1IN+. Or, the two comparators could be used to define upper and lower limits for the signal on C1IN+.

C2NREF selects the negative reference:

C2NREF = 1 selects the C2IN- pin

C2NREF = 0 selects the programmable internal reference voltage (see below)

By default (after a power-on reset), every bit of CM2CON0 is set to '1'.

Programmable voltage reference

We've seen that comparator 1 can be used with a fixed 0.6 V internal voltage reference, avoiding the need to provide an external reference voltage and saving a pin. However, 0.6 V is not always suitable, so an external reference may need to be used.

Comparator 2 is more flexible, in that it can be used with a programmable internal voltage reference (CV_{REF}), selectable from a range of 32 voltages, from 0V to 0.72 × V_{DD}.

The voltage reference is controlled by the VRCON register:

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
VRCON	VREN	VROE	VRR	-	VR3	VR2	VR1	VR0

The reference voltage is set by the VR<3:0> bits and VRR, which selects a high or low voltage range:

VRR = 1 selects the low range, where CV_{REF} = VR<3:0>/24 × V_{DD}.

VRR = 0 selects the high range, where CV_{REF} = V_{DD}/4 + VR<3:0>/32 × V_{DD}.

The available reference voltages are summarised in the following table, as a fraction of VDD and as an absolute voltage for the case where VDD = 5 V:

VRR = 1 (low range)		
VR<3:0>	fraction VDD	CVREF (VDD = 5V)
0	0.000	0.00V
1	0.042	0.21V
2	0.083	0.42V
3	0.125	0.62V
4	0.167	0.83V
5	0.208	1.04V
6	0.250	1.25V
7	0.292	1.46V
8	0.333	1.67V
9	0.375	1.87V
10	0.417	2.08V
11	0.458	2.29V
12	0.500	2.50V
13	0.542	2.71V
14	0.583	2.92V
15	0.625	3.12V

VRR = 0 (high range)		
VR<3:0>	fraction VDD	CVREF (VDD = 5V)
0	0.250	1.25V
1	0.281	1.41V
2	0.313	1.56V
3	0.344	1.72V
4	0.375	1.88V
5	0.406	2.03V
6	0.438	2.19V
7	0.469	2.34V
8	0.500	2.50V
9	0.531	2.66V
10	0.563	2.81V
11	0.594	2.97V
12	0.625	3.13V
13	0.656	3.28V
14	0.688	3.44V
15	0.719	3.59V

Note that the low and high ranges overlap, with 0.250×VDD, 0.500×VDD and 0.625×VDD selectable in both. Thus, of the 32 selectable voltages, only 29 are unique.

The VREN bit enables (turns on) the voltage reference.

To use the voltage reference, set VREN = 1.

Since the voltage reference module draws current, you should turn it off by clearing VREN to minimise power consumption in sleep mode – unless of course you are using wake on comparator change with CVREF as the negative reference, in which case the voltage reference needs to remain on.

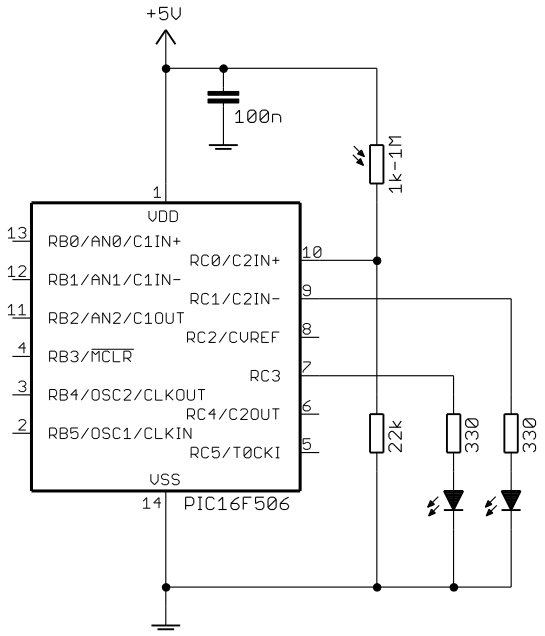
The only exception to this is you wish to set CVREF = 0V. In that case, with VRR = 1 and VR<3:0> = 0, the module can be turned off (VREN = 0) to conserve power and the reference voltage will be very close to 0V. That's useful if you wish to test for the input signal going negative (*zero-crossing*); the inputs can accept small negative voltages, down to -0.3V.

The VROE bit enables the voltage reference output on the CVREF pin. When VROE = 1, the CVREF output is enabled and, since it shares a pin with RC2, RC2 cannot then be used for digital I/O.

To demonstrate how the programmable voltage reference can be used with comparator 2, we can use the circuit shown on the right, with a photocell (and 22 kΩ resistor) connected to C2IN+. The LED on RC3 will indicate a low level of illumination, and the LED on RC1 will indicate bright light. When neither LED is lit, the light level will be in the middle; not too dim or too bright.

To implement this circuit using the Gooligum baseline training board, you should remove the shunt from of JP24 and instead place a shunt in position 2 ('C2IN+') of JP25, connecting photocell PH2 to C2IN+. You should also place shunts in JP17 and JP19, enabling the LEDs on RC1 and RC3.

If you are using Microchip's Low Pin Count Demo Board, you can connect a photocell and resistor (or pot) to C2IN+ via pin 10 on the 14-pin header. The demo board already has LEDs on RC1 and RC3.



To test whether the input is within limits, we will first configure the programmable voltage reference to generate the "low" threshold voltage, compare the input with this low level, and then reconfigure the voltage reference to generate the "high" threshold and compare the input with this higher level.

This process could be extended to multiple input thresholds, by configuring the voltage reference to generate each threshold in turn. However, if you wish to test against more than a few threshold levels, you would probably be better off using an analog-to-digital converter (described in the next lesson).

This example uses 2.0 V as the "low" threshold and 3.0 V as the "high" threshold, but, since the reference is programmable, you can always choose your own levels!

Comparator 2 is configured to use C2IN+ as the positive reference, and CVREF as the negative reference:

```
; configure comparator 2
movlw   1<<C2PREF1|0<<C2NREF|1<<C2POL|1<<C2ON
                            ; +ref is C2IN+ (C2PREF1 = 1)
                            ; -ref is CVref (C2NREF = 0)
                            ; normal polarity (C2POL = 1)
                            ; turn comparator on (C2ON = 1)
movwf   CM2CON0             ; -> C2OUT = 1 if C2IN+ > CVref
```

The voltage reference can be configured to generate approximately 2.0 V, by:

```
movlw   1<<VREN|0<<VRR|.5   ; configure voltage reference:
                            ;   enable voltage reference (VREN = 1)
                            ;   CVref = 0.406*Vdd (VRR = 0, VR = 5)
movwf   VRCON               ;   -> CVref = 2.03 V
```

The closest match to 3.0 V is obtained by:

```
movlw   1<<VREN|0<<VRR|.11  ; configure voltage reference:
                            ;   enable voltage reference (VREN = 1)
                            ;   CVref = 0.594*Vdd (VRR = 0, VR = 11)
movwf   VRCON               ;   -> CVref = 2.97 V
```

After changing the voltage reference, it can take a little while for it to settle and stably generate the newly-selected voltage. According to the PIC12F510/16F506 data sheet, this settling time can be up to 10 µs.

Therefore, we should insert a 10 µs delay after configuring the voltage reference, before reading the comparator output.

As we saw in lesson 2, a useful instruction for generating a two-instruction-cycle delay is '`goto $+1`'. Since each instruction cycle is 1 µs (with a 4 MHz processor clock), each '`goto $+1`' creates a 2 µs delay, and five of these instructions will give us the 10 µs delay we are after.

Since we need to insert this delay twice (once for each time we re-configure the voltage reference), it makes sense to define it as a macro:

```
; 10 us delay
;   (assuming 4 MHz processor clock)
Delay10us   MACRO
            goto $+1        ; 2 us delay * 5 = 10 us
            goto $+1
            goto $+1
            goto $+1
            goto $+1
            ENDM
```

This '`Delay10us`' macro can then be used to add the necessary 10 µs delay after each voltage reference re-configuration.

Complete program

Here is how these code fragments fit together.

Note that a shadow register is used for PORTC – in this case not so much to avoid read-write-modify problems, but simply because it makes the main loop logic more straightforward. If the loop started with PORTC being cleared, and then one of the LEDs turned on, the LED would end up being turned off then on, rapidly. The flickering would be too fast to be visible, but the LED's apparent brightness would be lower. Using a shadow register avoids that, without having to add more complex logic.

```
;*****************************************************************
;   Description:    Lesson 9, example 4                          *
;                                                                *
;   Demonstrates use of Comparator 2 and programmable voltage reference *
;                                                                *
;   Turns on Low LED  when C2IN+ < 2.0 V (low light level)       *
;         or High LED when C2IN+ > 3.0 V (high light level)      *
;                                                                *
;*****************************************************************
;                                                                *
;   Pin assignments:                                             *
;       C2IN+ = voltage to be measured (LDR/resistor divider)    *
;       RC3   = "Low" LED                                        *
;       RC1   = "High" LED                                       *
;                                                                *
;*****************************************************************

    list        p=16F506
    #include    <p16F506.inc>

    radix       dec


;***** CONFIGURATION
                ; ext reset, no code protect, no watchdog, 4 Mhz int clock
    __CONFIG    _MCLRE_ON & _CP_OFF & _WDT_OFF & _IOSCFS_OFF & _IntRC_OSC_RB4EN
```

```
; pin assignments
    constant    nLO=RC3         ; "Low" LED
    constant    nHI=RC1         ; "High" LED


;***** MACROS
; 10 us delay
; Assumes: 4 MHz processor clock
;
Delay10us   MACRO
            goto $+1        ; 2 us delay * 5 = 10 us
            goto $+1
            goto $+1
            goto $+1
            goto $+1
            ENDM


;***** VARIABLE DEFINITIONS
        UDATA_SHR
sPORTC  res 1                   ; shadow copy of PORTC


;***** RC CALIBRATION
RCCAL   CODE    0x3FF           ; processor reset vector
        res 1                   ; holds internal RC cal value, as a movlw k

;***** RESET VECTOR ****************************************************
RESET   CODE    0x000           ; effective reset vector
        movwf   OSCCAL          ; apply internal RC factory calibration


;***** MAIN PROGRAM ****************************************************

;***** Initialisation
start
        ; configure ports
        movlw   ~(1<<nLO|1<<nHI)    ; configure PORTC LED pins as outputs
        tris    PORTC

        ; configure comparator 2
        movlw   1<<C2PREF1|0<<C2NREF|1<<C2POL|1<<C2ON
                                    ; +ref is C2IN+ (C2PREF1 = 1)
                                    ; -ref is CVref (C2NREF = 0)
                                    ; normal polarity (C2POL = 1)
                                    ; turn comparator on (C2ON = 1)
        movwf   CM2CON0             ; -> C2OUT = 1 if C2IN+ > CVref


;***** Main loop
main_loop
        ; start with shadow PORTC clear
        clrf    sPORTC

;*** Test for low illumination
        ; set low input threshold
        movlw   1<<VREN|0<<VRR|.5   ; configure voltage reference:
                                    ;   enable voltage reference (VREN = 1)
                                    ;   CVref = 0.406*Vdd (VRR = 0, VR = 5)
        movwf   VRCON               ;   -> CVref = 2.03 V
        Delay10us                   ; wait 10 us to settle
```

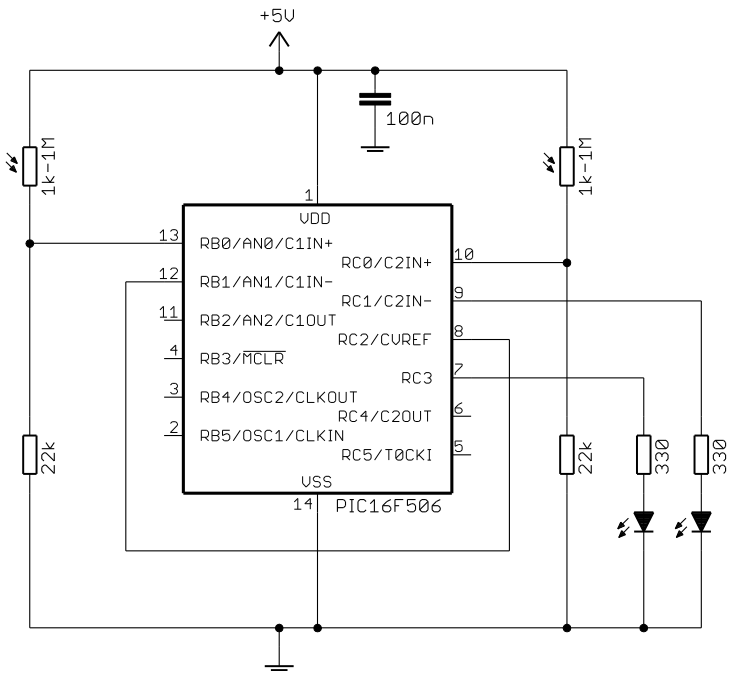
```
        ; compare with input
        btfss   CM2CON0,C2OUT       ; if C2IN+ < CVref
        bsf     sPORTC,nLO          ;   turn on Low LED

;*** Test for high illumination
        ; set high input threshold
        movlw   1<<VREN|0<<VRR|.11  ; configure voltage reference:
                                    ;   enable voltage reference (VREN = 1)
                                    ;   CVref = 0.594*Vdd (VRR = 0, VR = 11)
        movwf   VRCON               ;   -> CVref = 2.97 V
        Delay10us                   ; wait 10 us to settle

        ; compare with input
        btfsc   CM2CON0,C2OUT       ; if C2IN+ > CVref
        bsf     sPORTC,nHI          ;   turn on High LED

;*** Display test results
        movf    sPORTC,w            ; copy shadow to PORTC
        movwf   PORTC

        ; repeat forever
        goto    main_loop


        END
```

Using both comparators with the programmable voltage reference

For a final example, suppose that we want to test two input signals (say, light level in two locations) by comparing them against a common reference. We would need to use two comparators, with an input signal connected to each, and a single threshold voltage level connected to both.

What if we want to use the programmable voltage reference to generate the common threshold?

We've see that CVREF cannot be selected as an input to comparator 1, so it would seem that it's not possible to use the programmable voltage reference with comparator 1.



But although no internal connection is available, that doesn't rule out an external connection – and as we saw above, the programmable reference can be made available on the CVREF pin.

So, to use the programmable voltage reference with comparator 1, we need to set the VROE bit in the VRCON register, to enable the CVREF output, and connect the CVREF pin to a comparator 1 input – as shown in the circuit diagram on the left, where CVREF is connected to C1IN-.

If you are using the Gooligum baseline training board, you can keep the board set up as before, with shunts in JP17, JP19, and position 2 ('C2IN+') of JP25, and add a shunt across pins 2 and 3 ('LDR1') of JP24, to also connect photocell PH1 to C1IN+. You also need

to connect CVREF to C1IN-, which you can do by linking pins 9 ('GP/RA/RB1') and 11 ('RC2') on the 16-pin header.

If you are using Microchip's Low Pin Count Demo Board, the connection from CVREF to C1IN- can be made by linking pins 8 and 12 on the 14-pin header, and the photocells and associated resistors can be connected via the 14-pin header as before.

Note: Whichever board you are using, you should disconnect your PICkit 2 or PICkit 3 from the board when you run the program (applying external power instead), because the programmer loads RB1/AN1/C1IN-, pulling down the reference voltage delivered by the CVREF pin. The circuit will still operate with a PICkit 2 or PICkit 3 connected, but the reference voltage will be much lower than it should be.

Most of the initialisation and main loop code is very similar to that used in earlier examples, but when configuring the voltage reference, we must ensure that the VROE bit is set:

```
        movlw   1<<VREN|1<<VROE|1<<VRR|.12
                                ; CVref = 0.500*Vdd (VRR = 1, VR = 12)
                                ; enable CVref output pin (VROE = 1)
                                ; enable voltage reference (VREN = 1)
        movwf   VRCON           ; -> CVref = 2.50 V,
                                ;    CVref output pin enabled
```

Complete program

Here is the full listing for the "two inputs with a common programmed voltage reference" program:

```
;*****************************************************************
;                                                                *
;   Description:    Lesson 9, example 5                          *
;                                                                *
;   Demonstrates use of comparators 1 and 2                      *
;   with the programmable voltage reference                      *
;                                                                *
;   Turns on: LED 1 when C1IN+ > 2.5 V                           *
;         and LED 2 when C2IN+ > 2.5 V                           *
;                                                                *
;*****************************************************************
;   Pin assignments:                                             *
;       C1IN+ = input 1 (LDR/resistor divider)                   *
;       C1IN- = connected to CVref                               *
;       C2IN+ = input 2 (LDR/resistor divider)                   *
;       CVref = connected to C1IN-                               *
;       RC1   = indicator LED 2                                  *
;       RC3   = indicator LED 1                                  *
;                                                                *
;*****************************************************************

    list        p=16F506
    #include    <p16F506.inc>

    radix       dec


;***** CONFIGURATION
                ; ext reset, no code protect, no watchdog, 4 Mhz int clock
    __CONFIG    _MCLRE_ON & _CP_OFF & _WDT_OFF & _IOSCFS_OFF & _IntRC_OSC_RB4EN
```

```
; pin assignments
    constant    nLED1=RC3       ; indicator LED 1
    constant    nLED2=RC1       ; indicator LED 2


;***** VARIABLE DEFINITIONS
        UDATA_SHR
sPORTC  res 1                   ; shadow copy of PORTC


;***** RC CALIBRATION
RCCAL   CODE    0x3FF           ; processor reset vector
        res 1                   ; holds internal RC cal value, as a movlw k

;***** RESET VECTOR ****************************************************
RESET   CODE    0x000           ; effective reset vector
        movwf   OSCCAL          ; apply internal RC factory calibration


;***** MAIN PROGRAM ****************************************************

;***** Initialisation
start
        ; configure ports
        movlw   ~(1<<nLED1|1<<nLED2)    ; configure PORTC LED pins as outputs
        tris    PORTC

        ; configure comparator 1
        movlw   1<<C1PREF|1<<C1NREF|1<<C1POL|1<<C1ON
                                ; +ref is C1IN+ (C1PREF = 1)
                                ; -ref is C1IN- (C1NREF = 1)
                                ; normal polarity (C1POL = 1)
                                ; comparator on (C1ON = 1)
        movwf   CM1CON0         ; -> C1OUT = 1 if C1IN+ > C1IN- (= CVref)

        ; configure comparator 2
        movlw   1<<C2PREF1|0<<C2NREF|1<<C2POL|1<<C2ON
                                ; +ref is C2IN+ (C2PREF1 = 1)
                                ; -ref is CVref (C2NREF = 0)
                                ; normal polarity (C2POL = 1)
                                ; comparator on (C2ON = 1)
        movwf   CM2CON0         ; -> C2OUT = 1 if C2IN+ > CVref

        ; configure voltage reference
        movlw   1<<VREN|1<<VROE|1<<VRR|.12
                                ; CVref = 0.500*Vdd (VRR = 1, VR = 12)
                                ; enable CVref output pin (VROE = 1)
                                ; enable voltage reference (VREN = 1)
        movwf   VRCON           ; -> CVref = 2.50 V,
                                ;    CVref output pin enabled


;***** Main loop
main_loop
        ; start with shadow PORTC clear
        clrf    sPORTC

        ; test input 1
        btfsc   CM1CON0,C1OUT   ; if C1IN+ > CVref
        bsf     sPORTC,nLED1    ;   turn on LED 1
```

```
; test input 2
btfsc   CM2CON0,C2OUT       ; if C2IN+ > CVref
bsf     sPORTC,nLED2        ;   turn on LED 2

; display test results
movf    sPORTC,w            ; copy shadow to PORTC
movwf   PORTC

; repeat forever
goto    main_loop

END
```

Conclusion

This has been a long lesson, for such an apparently simple peripheral.

As we've seen, the comparators on baseline PICs can be configured with flexible combinations of inputs, including an absolute voltage reference and a programmable voltage reference (which can be made available externally). We've also seen how to use the external comparator outputs to generate hysteresis, and how comparator 1 can be used to clock the timer – which, as we demonstrated, can be used to count pulses that would be otherwise unsuited to digital inputs.

The next lesson continues the topic of analog inputs on baseline PICs, with an overview of analog-to-digital conversion.