

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 one or two of them, as built-in peripherals.

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

Of course, to demonstrate that, we need a device with comparators. A good choice is the 14-pin PIC16F506, since it adds analog input functionality to the PIC16F505 (described in [lesson 8](#)).

In summary, this lesson covers:

- The PIC16F506 MCU
- Using comparators to compare voltage levels
- Adding comparator hysteresis
- Using a comparator to drive Timer0
- Using absolute and programmable voltage references

Introducing the PIC16F506

The 16F506 is essentially a 16F505 with two comparators and a single 8-bit analog-to-digital converter, or *ADC* (described in the [next lesson](#)), with three input channels.

In the same way that the 12F508 and 12F09 are smaller variants of the 16F505, there is also a 12F510, which is an 8-pin variant of the 14-pin 16F506.

The table on the next page compares these features of these devices.

Device	Program Memory (words)	Data Memory (bytes)	Package	I/O pins	Comparators	Analog Inputs	Clock rate (maximum)
12F508	512	25	8-pin	6	-	-	4 MHz
12F509	1024	41	8-pin	6	-	-	4 MHz
12F510	1024	38	8-pin	6	1	3	8 MHz
16F505	1024	72	14-pin	12	-	-	20 MHz
16F506	1024	67	14-pin	12	2	3	20 MHz

In the 12F510/16F506 devices, the internal RC oscillator can optionally run at a nominal 8 MHz instead of 4 MHz. *Be careful, if you select 8 MHz, that any code (such as delays) written for a 4 MHz clock is correct.*

The speed of the internal RC oscillator is selected by the IOSCFS bit in the configuration word:

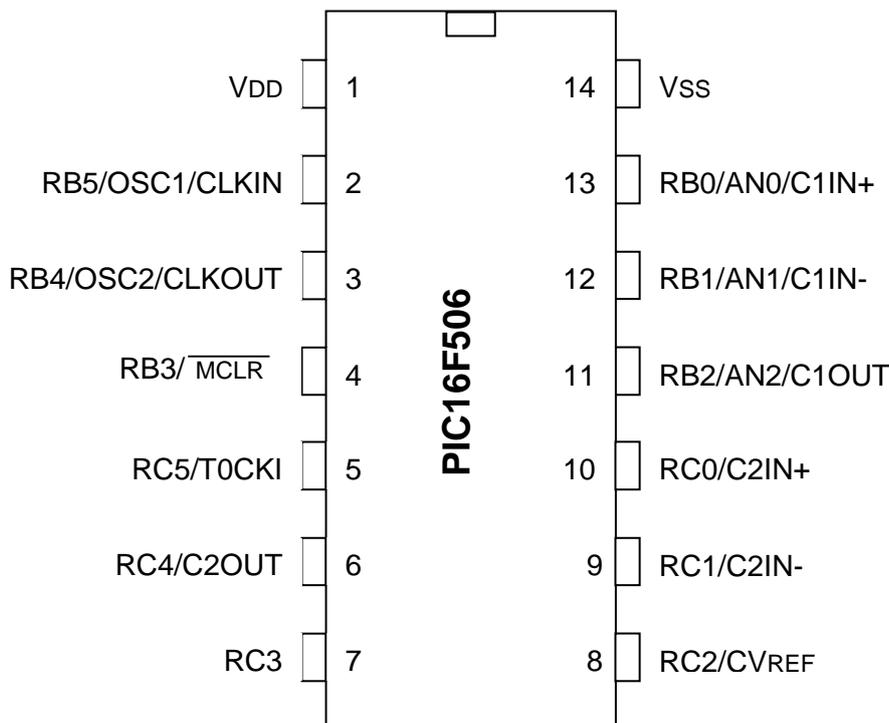
Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
-	-	-	-	-	IOSCFS	MCLRE	\overline{CP}	WDTE	FOSC2	FOSC1	FOSC0

Setting IOSCFS to '1' (by ANDing the symbol '`_IOSCFS_ON`' into the configuration word expression) selects 8 MHz operation; clearing it to '0' (with '`_IOSCFS_OFF`') selects 4 MHz.

The other oscillator modes are all the same as for the 16F505.

Pin diagram

The 16F506 comes in a 14-pin package, as shown below.



RC5 and T0CKI share the same pin, as was the case on the 16F505; to use RC5 as an output on the 16F506, you must first disable T0CKI by clearing the T0CS bit in the OPTION register.

The RB0-2 pins can also be used as analog inputs, and are configured as analog inputs by default. To use any of these pins for digital I/O, they must be deselected as analog inputs, as explained in the [next lesson](#) on analog-to-digital conversion.

The RB0, RB1, RC0 and RC1 pins are configured as comparator inputs by default. To use any of these pins for digital I/O, the appropriate comparator must be disabled, or its inputs reassigned, as explained on the section on comparators, later in this lesson.

Note: On PICs with comparators and/or analog inputs, the comparator and analog inputs are enabled on start-up. To use a pin for digital I/O, any comparator or analog input assigned to that pin must first be disabled.

This is a common trap for beginners, who wonder why their LED won't light, when they haven't deselected analog input on the pin they are using. That is why this tutorial series began with digital-only PICs.

Data memory

The data memory, or register file, on the 16F506 is arranged in four banks, as follows:

PIC16F506 Registers

Bank 0		Bank 1		Bank 2		Bank 3	
00h	INDF	20h	INDF	40h	INDF	60h	INDF
01h	TMR0	21h	TMR0	41h	TMR0	61h	TMR0
02h	PCL	22h	PCL	42h	PCL	62h	PCL
03h	STATUS	23h	STATUS	43h	STATUS	63h	STATUS
04h	FSR	24h	FSR	44h	FSR	64h	FSR
05h	OSCCAL	25h	OSCCAL	45h	OSCCAL	65h	OSCCAL
06h	PORTB	26h	PORTB	46h	PORTB	66h	PORTB
07h	PORTC	27h	PORTC	47h	PORTC	67h	PORTC
08h	CM1CON0	28h	CM1CON0	48h	CM1CON0	68h	CM1CON0
09h	ADCON0	29h	ADCON0	49h	ADCON0	69h	ADCON0
0Ah	ADRES	2Ah	ADRES	4Ah	ADRES	6Ah	ADRES
0Bh	CM2CON0	2Bh	CM2CON0	4Bh	CM2CON0	6Bh	CM2CON0
0Ch	VRCON	2Ch	VRCON	4Ch	VRCON	6Ch	VRCON
0Dh	Shared GP Registers	2Dh	Map to Bank 0 0Dh – 0Fh	4Dh	Map to Bank 0 0Dh – 0Fh	6Dh	Map to Bank 0 0Dh – 0Fh
0Fh		2Fh		4Fh		6Fh	
10h	General Purpose Registers	30h	General Purpose Registers	50h	General Purpose Registers	70h	General Purpose Registers
1Fh		3Fh		5Fh		7Fh	

The additional registers, used for the two comparators (CM1CON0, CM2CON0), programmable voltage reference (VRCON) and ADC (ADCON0, ADRES), require five addresses, leaving space for only 3 shared data registers (0Dh – 0Fh)¹, which are mapped into all four banks.

On the other hand, there is a full complement of $4 \times 16 = 64$ banked data registers, filling the top half of each bank.

Thus, the 16F506 has a total of $3 + 64 = 67$ general purpose data registers.

Besides having comparators and analog inputs, being able to run the internal RC oscillator at 8 MHz, and having fewer shared registers, there are no other significant differences (for most users) between the 12F508/509/16F505 family and the 12F510/16F506 devices.

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.

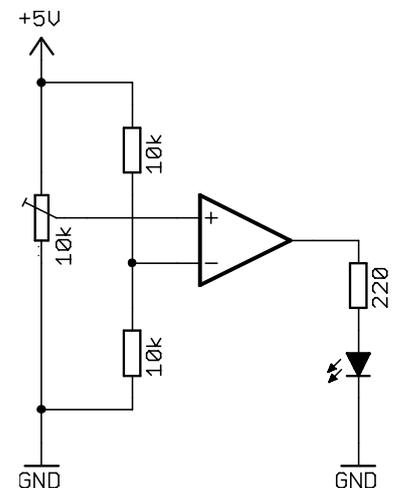
When the potentiometer is set to a voltage under 2.5 V, the comparator’s output will be low and the LED will not be lit. But when the potentiometer is turned up past halfway, the comparator’s output will go high, lighting the LED.

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. 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.



¹ Be careful when migrating existing code to a 16F506; if it requires more than three shared registers, you’ll have to make changes...

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 above.

The comparator's output appears as the C1OUT bit: it is 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

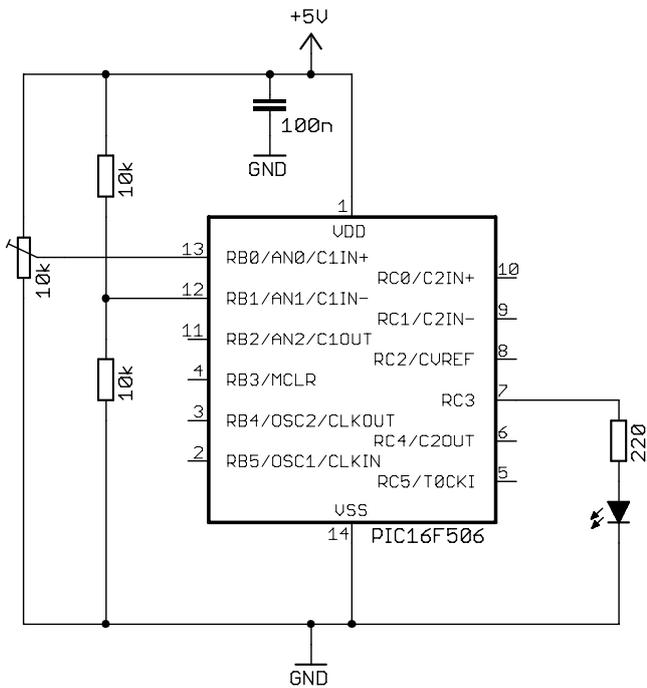
C1POL = 0 selects inverted operation, where C1OUT = 1 only if positive ref < negative ref.

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 use the circuit on the next page to illustrate the comparator's basic operation.

It can be readily built using the Microchip Low Pin Count Demo Board we've been using in this tutorial series.



The 10 k Ω potentiometer on the demo board is already connected to C1IN+ (labelled 'RA0') on the board, via a 1 k Ω resistor (not shown in this diagram). But you must ensure that jumper JP5 is in place; 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 220 Ω as shown here, but that makes no difference); all you need to do is ensure that jumper JP4 is in place.

The connection to C1IN- (labelled 'RA1' on the board) is available as pin 8 on the 14-pin header. +5V and GND are brought out to pins 13 and 14, respectively, making it easy to add the 10 k Ω resistors, forming a voltage divider, by using a solderless breadboard connected to the 14-pin header.

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 side.

First configure RC3 as an output:

```
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. It is only necessary to 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 ('DS4') turns on and off

To make the circuit a little more interesting, you could add a light-dependent resistor (LDR), as shown on the right.

To build this using the Low Pin Count Demo Board, you should remove jumper JP5 (you may need to cut the PCB trace), to disconnect the pot from the +5 V supply².

The LDR can then be connected between pin 7 on the 14-pin header and +5 V.

Note that, if you are not using the demo board, you do not need to include the 1 k Ω resistor; it is only shown in this circuit because the demo board includes that 1 k Ω resistor. Or, you could use a fixed 10 k Ω resistor instead of the pot.

The exact resistance range of the LDR is not important; any LDR that varies around 10 k Ω or so for normal lighting conditions would be ok.

If you don't have an LDR, 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!

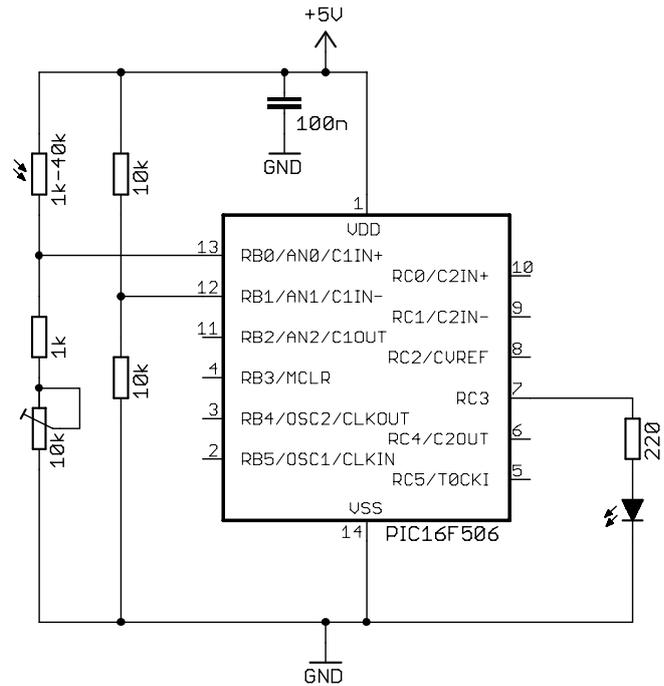
When you build this circuit and use the above code, you will find that the LED turns on when the LDR is illuminated (you may need to adjust the pot to set the threshold for the ambient light level). That's because, when an LDR is illuminated, its resistance falls, increasing the voltage at C1IN+.

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
```

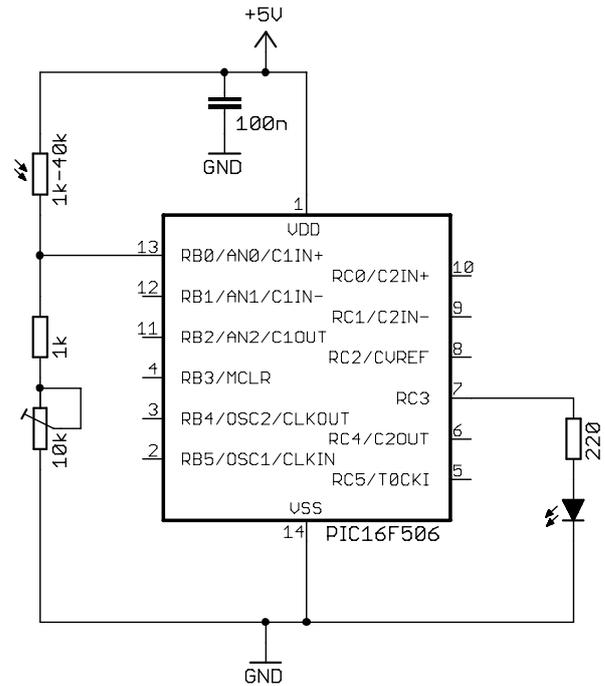


² The +5 V supply connected through JP5 is also used to hold $\overline{\text{MCLR}}$ high when the reset line is tri-stated – as it will be if you are using a PICkit 2 with MPLAB and have selected the ‘3-State on “Release from Reset”’ setting. For correct operation with JP5 removed, you must either disable this setting, or use internal $\overline{\text{MCLR}}$.

Since the negative reference is now 0.6 V instead of 2.5 V, you will need to adjust the potentiometer, but you should find a position where the LED is normally off, but comes on when the LDR is in shadow.

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 free to use for digital I/O (or AN1 as an analog input, as we'll see in the [next lesson](#)...).



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, V_t , is set by a voltage divider, formed by resistors R1 and R2.

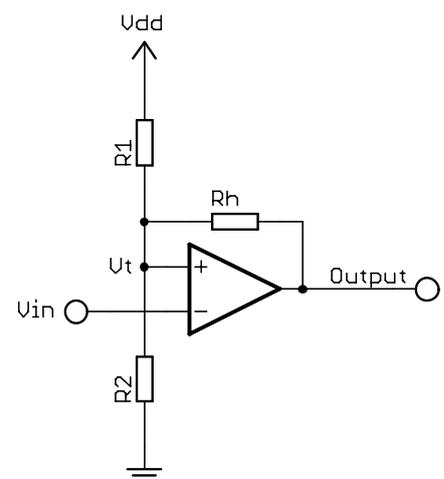
This would normally set the threshold at $V_t = \frac{R_2}{R_1 + R_2} V_{dd}$.

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

Now consider what happens when V_{in} is low (less than V_{tl}) and begins to increase. Initially, since $V_{in} < V_{tl}$, the comparator output is high, and the threshold is high: $V_t = V_{th}$.

Eventually V_{in} rises above V_{th} , and the comparator output goes low, lowering the threshold: $V_t = V_{tl}$.

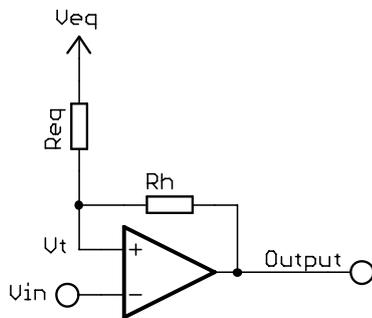
Now suppose the input voltage begins to fall. As it falls past the high threshold, V_{th} , nothing happens. It has to keep falling, all the way down to the low threshold, V_{tl} , before the comparator output changes again.



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: $V_{hb} = V_{th} - V_{tl}$.

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, V_{eq} , in series with a single resistor, R_{eq} .

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

$$V_{eq} = \frac{R_2}{R_1 + R_2} V_{dd}$$

and R_{eq} is the parallel combination of R_1 and R_2 : $R_{eq} = \frac{R_1 R_2}{R_1 + R_2}$

Therefore, when the comparator's output is low ($= 0\text{ V}$):

$$V_{tl} = \frac{R_h}{R_h + R_{eq}} V_{eq} \quad (\text{thus } V_{tl} < V_{eq})$$

and, when the comparator's output is high ($= V_{dd}$):

$$V_{th} = V_{eq} + \frac{R_{eq}}{R_h + R_{eq}} (V_{dd} - V_{eq}) \quad (\text{thus } V_{th} > V_{eq})$$

This little bit of mathematics proves that $V_{th} > V_{tl}$, 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.

When 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 pot is set to 1 k Ω , for a resistance between C1IN+ and ground of 2 k Ω (= 1 k Ω + 1 k Ω), with the light level such that the LDR has a resistance of 14.7 k Ω :

$$V_{in} = 2 / (2 + 14.7) \times 5 \text{ V} = 0.6 \text{ V}$$

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

Now suppose that we add a 22 k Ω feedback resistor, as shown.

The effect is similar to that for threshold voltage feedback. The circuit is equivalent to a source (input) voltage of 0.6 V, connected to C1IN+ via a resistance of:

$$R_{eq} = (14.7 \times 2) / (14.7 + 2) \text{ k}\Omega = 1.76 \text{ k}\Omega$$

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

$$V_{inl} = 22 / (22 + 1.76) \times 0.6 \text{ V} = 0.56 \text{ V}$$

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

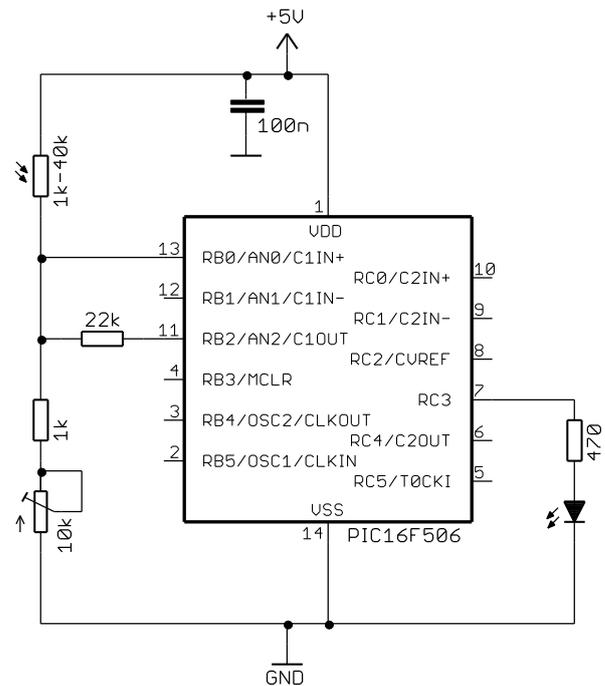
$$V_{inh} = 0.6 \text{ V} + 1.76 / (22 + 1.76) \times (5.0 \text{ V} - 0.6 \text{ V}) = 0.93 \text{ V}$$

Note that the voltages calculated above 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.

Note also that, if you build this circuit using Microchip's Low Pin Count demo board and use a PICkit 2 to program and power it, you will see slightly different voltage levels if you test this circuit while the PICkit 2 is connected. This is because the RB0 (C1IN+) pin is used for in-circuit programming and debugging, and is loaded by the PICkit 2, changing the effective resistances from those shown.

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.



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 know 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 LDR is 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
; 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
mainloop
; display comparator output
btfsc  CM1CON0,C1OUT   ; if comparator output high
bsf    LED             ; turn on LED
btfss  CM1CON0,C1OUT   ; if comparator output low
bcf    LED             ; turn off LED

; repeat forever
goto   mainloop

```

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 $\overline{C1WU}$ bit in the CM1CON0 register.

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

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

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 $\overline{C1WU}$ (NOT C1WU) has not been included in the expression loaded into CM1CON0 in any of the examples so far, $\overline{C1WU}$ 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{TO}	\overline{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.

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    ; initialise port
         bcf      LED                ; start with LED off
         movlw   ~(1<<nLED)          ; configure LED pin (only) as an output
         tris    PORTC

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

         ; else power-on reset, so configure comparator 1
         movlw   b'01111010
         ; -1-----          disable C1OUT pin (*C1OUTEN = 1)
         ; --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 disabled,
                        ;     wake on comparator change enabled

DelayMS 10                ; delay 10 ms to allow comparator to settle

goto     standby         ; sleep until comparator change

; flash LED
flash    bsf             LED          ; turn on LED
        DelayMS 1000        ; delay 1s

; 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 off comparators before entering sleep mode.

Incrementing Timer0

As we saw in [lesson 5](#), 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 $\overline{\text{C1T0CS}}$.

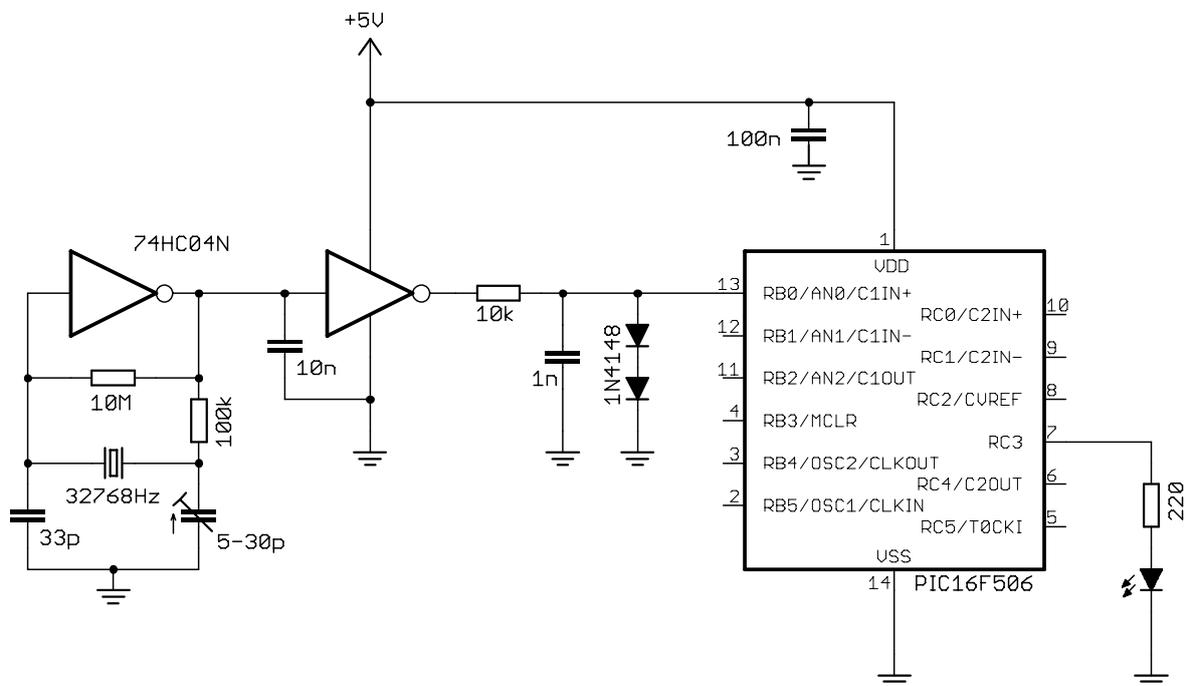
So, to use comparator 1 as the counter clock source, you must set T0CS = 1 and $\overline{\text{C1T0CS}} = 0$.

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.

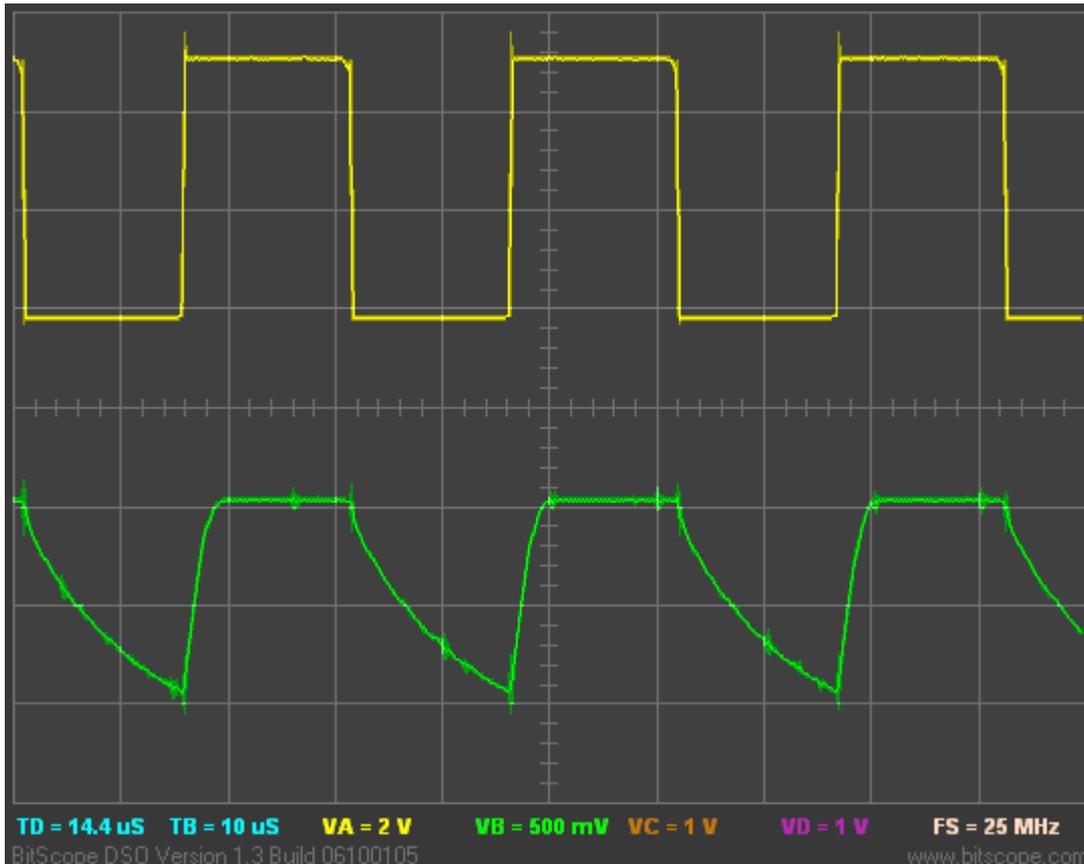
To illustrate how comparator 1 can be used with Timer0, we can use the external clock circuit from [lesson 5](#), but with the clock signal degraded by passing it through an RC filter and clamping with two diodes:



The effect of this can be seen in the oscilloscope trace, on the next page.

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

Note that, if you are using the LPC Demo Board, the components connected to the C1IN+ input, including the potentiometer, will affect this signal. You may need to adjust the pot to make it work. Note also that, because the diodes and capacitor interfere with the ICSP programming signal, you must first disconnect the input from C1IN+ before the PICkit 2 programmer can successfully program the PIC.



This degraded signal is not suitable for driving TOCK1 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 above.

[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 TOCK1. 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)
; -> C1OUT = 1 if C1IN+ > 0.6V,
;     TMR0 clock from C1

movwf    CM1CON0

```

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 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, 4MHz 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

;***** RESET VECTOR *****
RESET    CODE    0x000             ; effective reset vector
         movwf   OSCCAL            ; update OSCCAL with factory cal value

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

;***** Initialisation
; configure ports
movlw    ~(1<<nFLASH)              ; configure LED pin (only) as output
tris     PORTC
clrf     ADCON0                    ; disable analog inputs
; configure timer
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
loop    ; TMR0<7> cycles at 1 Hz
        ; so continually copy to LED
        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   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 an 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 $0.72 \times V_{DD}$.

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 $CV_{REF} = VR<3:0>/24 \times V_{DD}$.

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

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

VRR = 1 (low range)			VRR = 0 (high range)		
VR<3:0>	fraction V_{DD}	CV_{REF} ($V_{DD} = 5V$)	VR<3:0>	fraction V_{DD}	CV_{REF} ($V_{DD} = 5V$)
0	0.000	0.00V	0	0.250	1.25V
1	0.042	0.21V	1	0.281	1.41V
2	0.083	0.42V	2	0.313	1.56V
3	0.125	0.62V	3	0.344	1.72V
4	0.167	0.83V	4	0.375	1.88V
5	0.208	1.04V	5	0.406	2.03V
6	0.250	1.25V	6	0.438	2.19V
7	0.292	1.46V	7	0.469	2.34V
8	0.333	1.67V	8	0.500	2.50V
9	0.375	1.87V	9	0.531	2.66V
10	0.417	2.08V	10	0.563	2.81V
11	0.458	2.29V	11	0.594	2.97V
12	0.500	2.50V	12	0.625	3.13V
13	0.542	2.71V	13	0.656	3.28V
14	0.583	2.92V	14	0.688	3.44V
15	0.625	3.12V	15	0.719	3.59V

Note that the low and high ranges overlap, with $0.250 \times V_{DD}$, $0.500 \times V_{DD}$ and $0.625 \times V_{DD}$ 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.

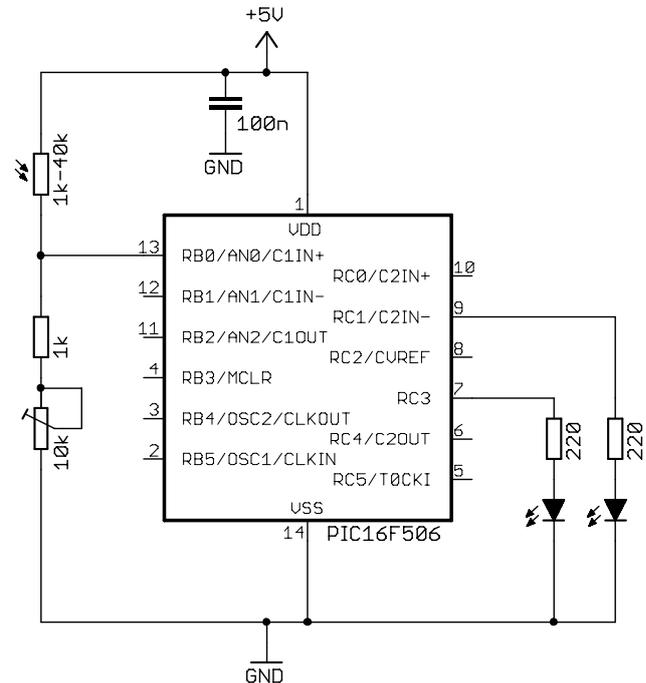
The VROE bit enables the voltage reference output on the CVREF pin. When VROE = 1, the CVREF output is enabled and, since it shares a pin with RC2, RC2 cannot then be used for digital I/O.

To demonstrate how the programmable voltage reference can be used with comparator 2, we can add another LED to the LDR circuit we used earlier, as shown on the right. 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 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 1.0 V as the “low” threshold and 2.0 V as the “high” threshold, but, since the reference is programmable, you can always choose your own levels!



Comparator 2 is configured to use C1IN+ as the positive reference³, and CVREF as the negative reference:

```

; configure comparator 2
movlw    0<<C2PREF1|1<<C2PREF2|1<<C2POL|1<<C2ON
                                ; +ref is C1IN+ (C2PREF1 = 0, C2PREF2 = 1)
                                ; -ref is CVref (C2NREF = 0)
                                ; normal polarity (C2POL = 1)
                                ; turn comparator on (C2ON = 1)
movwf    CM2CON0                ; -> C2OUT = 1 if C1IN+ > CVref

```

The voltage reference can be configured to generate approximately 1.0 V , by:

³ It might seem more logical to use C2IN+, but the pot on the LPC demo board is already connected to C1IN+.

```

movlw    1<<VREN|1<<VRR|.5    ; configure voltage reference:
                                ; enable voltage reference (VREN = 1)
                                ; CVref = 0.208*Vdd (VRR = 1, VR = 5)
movwf    VRCON                  ; -> CVref = 1.04 V

```

The closest match to 2.0 V is obtained 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

```

After changing the voltage reference, it can take a little while for it to settle and stably generate the newly-selected voltage. According to table 13-2 in the PIC12F510/16F506 data sheet, this settling time can be up to 10 μ s.

Therefore, we should insert a 10 μ s 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 μ s (with a 4 MHz processor clock), each 'goto \$+1' creates a 2 μ s delay, and five of these instructions will give us the 10 μ s 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 μ s 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 C1IN+ < 1.0 V (low light level)
; or High LED when C1IN+ > 2.0 V (high light level)
;
;*****

```

```

;                                                                    *
; Pin assignments:                                                                    *
;   C1IN+ - 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
; (assuming 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

;***** RESET VECTOR *****
RESET     CODE    0x000         ; effective reset vector
            movwf OSCCAL        ; update OSCCAL with factory cal value

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

;***** Initialisation
            ; configure ports
            movlw  ~(1<<nLO|1<<nHI)    ; configure PORTC LED pins as outputs
            tris   PORTC
            ; configure comparator 2
            movlw  0<<C2PREF1|1<<C2PREF2|1<<C2POL|1<<C2ON
                                ; +ref is C1IN+ (C2PREF1 = 0, C2PREF2 = 1)
                                ; -ref is CVref (C2NREF = 0)
                                ; normal polarity (C2POL = 1)
                                ; turn comparator on (C2ON = 1)
            movwf  CM2CON0        ; -> C2OUT = 1 if C1IN+ > CVref

```

```

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

    ;*** Test for low illumination
    ; set low input threshold
    movlw  1<<VREN|1<<VRR|.5    ; configure voltage reference:
                                ; enable voltage reference (VREN = 1)
                                ; CVref = 0.208*Vdd (VRR = 1, VR = 5)

    movwf  VRCON                  ; -> CVref = 1.04 V
    Delay10us                    ; wait 10us to settle
    ; compare with input
    btfss  CM2CON0,C2OUT         ; if C1IN+ < CVref
    bsf    sPORTC,nLO           ; turn on Low LED

    ;*** Test for high illumination
    ; set high 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 10us to settle
    ; compare with input
    btfsc  CM2CON0,C2OUT         ; if C1IN+ > CVref
    bsf    sPORTC,nHI           ; turn on High LED

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

    ; repeat forever
    goto  mainloop

END

```

Using both comparators with the programmable voltage reference

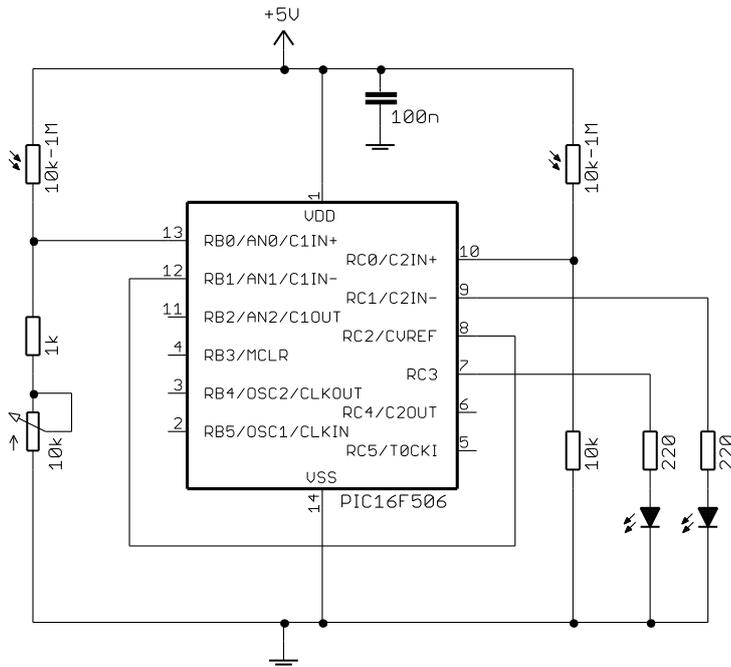
For a final example, suppose that we want to test two input signals (say, light level in two locations) by comparing them against a common reference. We would need to use two comparators, with an input signal connected to each, and a single threshold voltage level connected to both.

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

We've seen 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 next page, where CVREF is connected to C1IN-.



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.

Although the LPC Demo Board has LEDs (with series resistors) connected to RC0 and RC2, they will not appreciably affect the operation of this circuit, and hence do not have to be disconnected. This is because LEDs have a forward voltage of at least 1.8 V, meaning that they will conduct only a negligible current near the 1.0 V reference threshold used in this example.

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|.5
; CVref = 0.208*Vdd (VRR = 1, VR = 5)
; enable CVref output pin (VROE = 1)
; enable voltage reference (VREN = 1)
movwf   VRCON
; -> CVref = 1.04 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+ > 1.0 V
;           and LED 2 when C2IN+ > 1.0 V
;
;*****
;
; Pin assignments:
; C1IN+ - input 1 (LDR/resistor divider)
; C1IN- - connected to CVref
; C2IN+ - input 2 (LDR/resistor divider)
; RC1   - indicator LED 1
; RC3   - indicator LED 2
;
    
```

```

;*****
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 & _IOSCFSS_OFF & _IntRC_OSC_RB4EN

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

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

;***** RESET VECTOR *****
RESET    CODE    0x000          ; effective reset vector
          movwf   OSCCAL        ; update OSCCAL with factory cal value

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

;***** Initialisation
           ; 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    CM2CON0                ; -> 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|.5
           ; CVref = 0.208*Vdd (VRR = 1, VR = 5)
           ; enable CVref output pin (VROE = 1)
           ; enable voltage reference (VREN = 1)
movwf    VRCON                  ; -> CVref = 1.04 V,
           ; CVref output pin enabled

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

```

```
; 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 High LED

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

; repeat forever
goto   mainloop

END
```

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