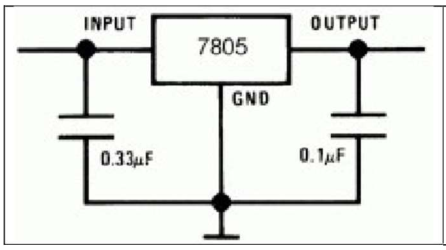
Knowledge of baseline

Vdd

The purpose of these pages is to provide some information about the supply voltage of Baseline PICs that is usually overlooked:

The Vdd

This term indicates the supply voltage of the chip, which for Baseline PICs can range from 2V to 5.5V.
These are operational limits (datasheet, param. D001): the maximum value must not be exceeded, otherwise the component will be irreversibly damaged, and the minimum value must not be lowered, otherwise it will operate irregularly.
In fact, the Vdd can go from 0 to 6.5V (**absolute values**!), but above 6Vsi you are assured of the rapid destruction of the chip.
On the other hand, below 2V there are no security problems, but the storage of data in RAM and the operation of the integrated peripherals are put in crisis.
It is therefore necessary that the 2-5.5V limits are not exceeded.



The solution generally used is that of a linear regulator with three terminals, a field in which the famous (but old) 7805 is still widely present, which outputs 5V with an input between 7 and 35V.
Its strength is the simplicity of the application circuit.

The values given for capacitors are the minimum values recommended by the data sheet.

Now let's consider a detail that is not so obvious: we see thousands of examples of microcontroller-based circuits with Vdd = 5V. And you can get the idea that this is the value to use.

In fact, the voltage of 5V was standardized some time ago to power TTL circuits and, with them, it has become so widespread that it is the typical microprocessor and microcontroller, which, even if made in MOS technologies, adapt to the logic levels typical of TTL, to maintain compatibility with the thousands and thousands of other integrated circuits.
If microprocessors have very strict power supply limits, given that they are typically "CPU-only" components, in which performance counts and that are part of complex systems, given that memories and peripherals are external to the chip, microcontrollers, in which everything is integrated into the component and which are designed for different areas of use, they have much more elastic limits; As we said at the beginning, any voltage between 2 and 5V is fine for PICs. In this range the microcontroller will perform as expected.

We can then also power it with one of the other "standard" voltages, i.e. 3.3V or 3V, voltages for which there are linear regulators of application similar to the 7805.
In this sense, there is nothing to prevent you from powering the circuit with two or three 1.5V batteries in series or with a 3-4.2V lithium cell.

In practice, a battery power supply has the only disadvantage that the voltage drops with use; It is necessary to keep this voltage under control, especially if rechargeable batteries are used, in order to avoid too deep a discharge.

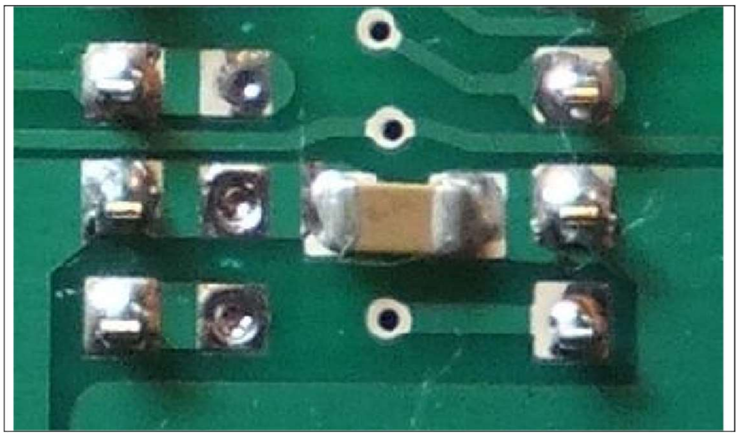
So, let's get **the image of the ICP+7805 out of our minds: it is not an obligation**.

If as far as the operation of the program is concerned, in the sense of its execution by the microcontroller, the supply voltage is indifferent except for the maximum possible clock frequency, it is certainly not so with regard to the connected peripherals.
In particular, the classic LED that we used in the various exercises, requires a resistance in series to limit the current and this must have a different value (see Appendix I). Exercise 1) depending on the voltage. Since the brightness depends on the current, calculating a resistance for a Vdd=5V and then powering the circuit to 3V we will have a reduction in the brightness of the LED. If the circuit is permanently powered at 3V, the resistance will need to be recalculated. And there can also be cases where connected peripherals, such as LCD displays, have a 5V operating limit and, powered by 3V, refuse to work properly. This does not exclude the possibility of having parts powered at different voltages in the same circuit, using interface solutions between the various sections.

In general, **more than the absolute value, what matters is the quality of this power supply**: apart from the problems of unwanted couplings, if, in practice, the microcontroller accepts supply voltages that are certainly not perfect, it should be borne in mind that, if the supply voltage is used as a reference for AD conversion and comparators, It is not so much its nominal value (for which it is sufficient to adjust the calculations in the algorithms) or its precision (which can also be corrected by program) that becomes extremely important, but **its stability** and the absence of ripple and disturbing pulses. In particular, in Baselines it is mandatory to use the Vdd as the reference voltage of the ADC module and, even if it is a conversion to only 8 bits, any problem on the power supply is reflected in the result and the attempt to correct the situation using averaging algorithms on several measures is not always decisive.

This is not the place for an in-depth discussion of the problem of nutrition: it is easy to find extensive documentation, articles, diagrams and advice to which we refer.

Here we only add that an indispensable element that must never be missing is **a multilayer ceramic capacitor, with low inductance, connected between the power pins as close as possible to the chip**, better than directly under the socket. Let's see here the solution adopted on the **LPCuB**, using an SMD component:



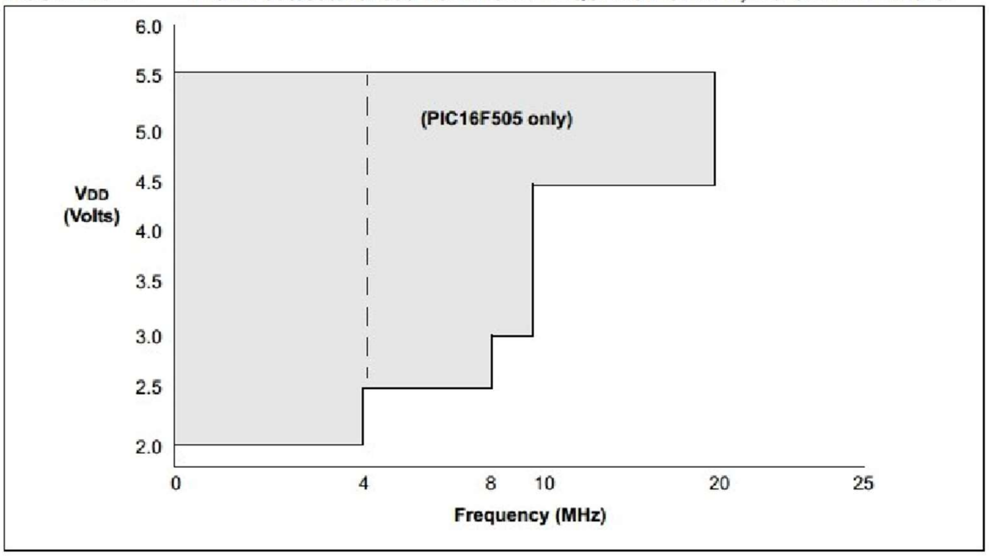
Obviously, decoupling capacitors will have to be inserted into the circuit where necessary, but these are topics that concern hardware design, rather than software design.
However, it should be kept in mind that if the hardware is not in place, the software does not work (and vice versa...).

So, for PCIs:

1. **The value of the Vdd is not decisive**, which can go between 2 and 5V or a little more.
2. It can be, however, for peripherals connected to the microcontroller.
3. If the application does not use the Vdd as an analog reference, it does not matter how it varies, always bearing in mind the limitations of points 1 and 2. On the other hand, the quality (absence of disturbances) is always good to be taken care of to the maximum.
4. But if we use analog functions (ADCs, comparators) **the stability and quality of the** VDD is essential, more than its absolute value, which will be correctable with appropriate algorithms inserted in the program.

It should also be considered, as we have seen when talking about the clock, that the maximum working frequency is directly related to the supply voltage:

FIGURE 10-1: PIC12F508/509/16F505 VOLTAGE-FREQUENCY GRAPH, -40°C ≤ TA ≤ +125°C



The lower the Vdd, the lower the maximum frequency that can be reached. For example, we can see from the graph that clock frequencies above 10MHz are only possible with confidence if the Vdd is greater than 4.5V.

The working frequency is also linked to the consumption factor: the lower the frequency, the lower the current absorbed.

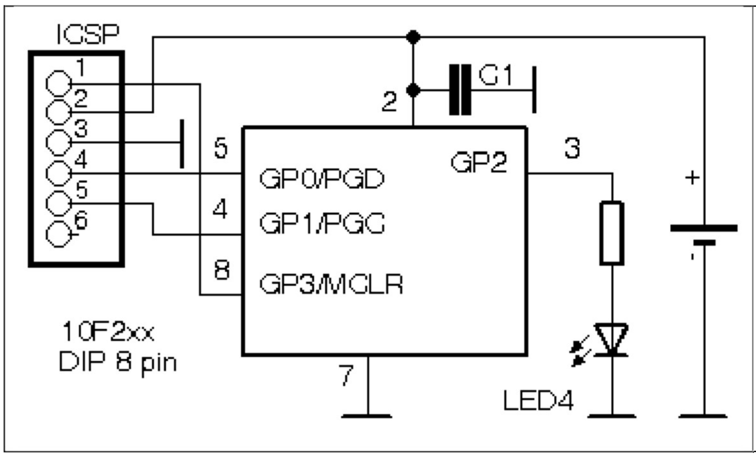
Therefore, by lowering the voltage and frequency, we lower the power consumption. It follows that a low power solution can be achieved by implementing:

- Lowest possible clock
- Power supply at the lowest possible voltage
- Extensive use of sleep mode

Let's look at an application.

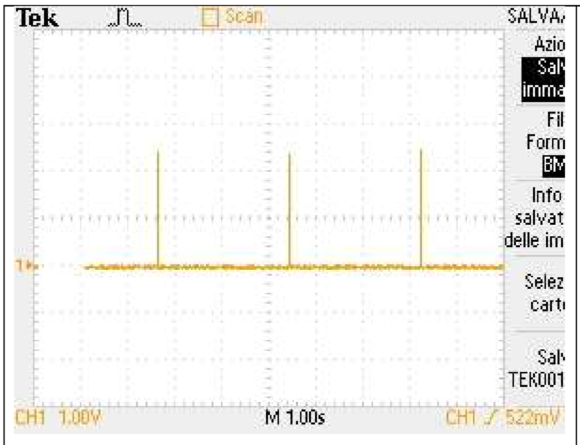
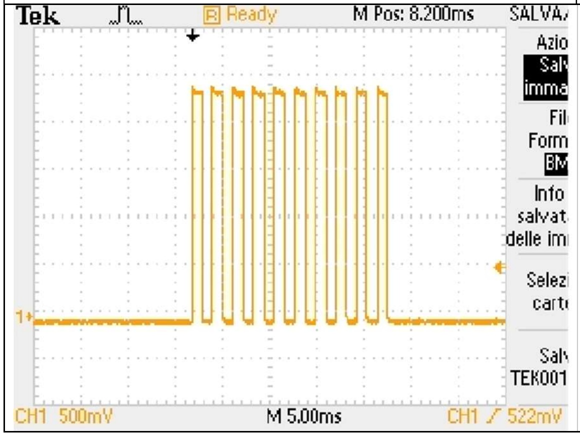
An application

We can make a circuit based on these principles:

 ICSP 1 2 3 4 5 6 2 C1 5 4 8 GP0/PGD GP1/PGC GP3/MCLR GP2 3 + 10F2xx DIP 8 pin 7 LED4	

A PIC10F2xx is powered at Vdd=3V with a CR2032 battery; it is the classic disk element used to support CMOS memory in personal computers and therefore very common and easy to recover, but any other battery between 2 and 4.5V is fine.

As far as switching times are concerned, remember that it is not possible to act on the clock, since these chips can only operate with the internal 4MHz one; So it is not possible to have a lower frequency clock. If we were to use the usual time procedures, the chip would always be running at 4MHz and this would produce excessive power consumption.
This is remedied by using sleep mode, using the WDT as a timer.

	The LED is lit for about 20ms once every two seconds. During pause periods, the chip is sent to sleep to reduce power consumption. The result is an oscillogram like the one on the side (timebase: 1s/div). It can be seen that the WDT interval is, in this case, more than 2.4ms approximately.
	To further reduce the average current, the switch-on period consists of a series of 1ms pulses. In the oscillogram on the side, the timebase has changed to 5ms/div. Due to the speed, there is no flickering and the eye recognizes a single flash, but the oscilloscope captures the real situation.

In the program, the duration of the Delay1ms delay routine determines the on/off cycle, while the step parameter determines how many times it is repeated; by varying them, you can experiment with different combinations.
With the current values, you have 10 pulses of 1ms each.

As mentioned, the time of about two seconds is produced not by a counting routine, but by the intervention of the WDT on sleep. Overall, for an LED current of more than 20mA at the switch-on time, the average current is about 2.2uA! A 200mAh battery should be able to last thousands of hours.
With a high-brightness LED, the flashing is clearly visible even in the sun, while at night it can be perceived well from a hundred meters away.
The LED series resistor is used to limit the current. The calculation formula is commonly found:

$$R_{led} = (V_{dd} - V_{led}) / I_{led}$$

Where ***Vled*** is the conduction voltage of the LED. If we have ***Vdd=3V***, ***Vled=2V*** and ***Iled=10mA***:

$$R_{led} = (3 - 2) / 0.010 = 100\text{ohm}$$

However, with low Vdd voltage values and high LED conduction voltage values, the Voh voltage drop on the I/O MOSFET, which is not 0V, must also be taken into account.

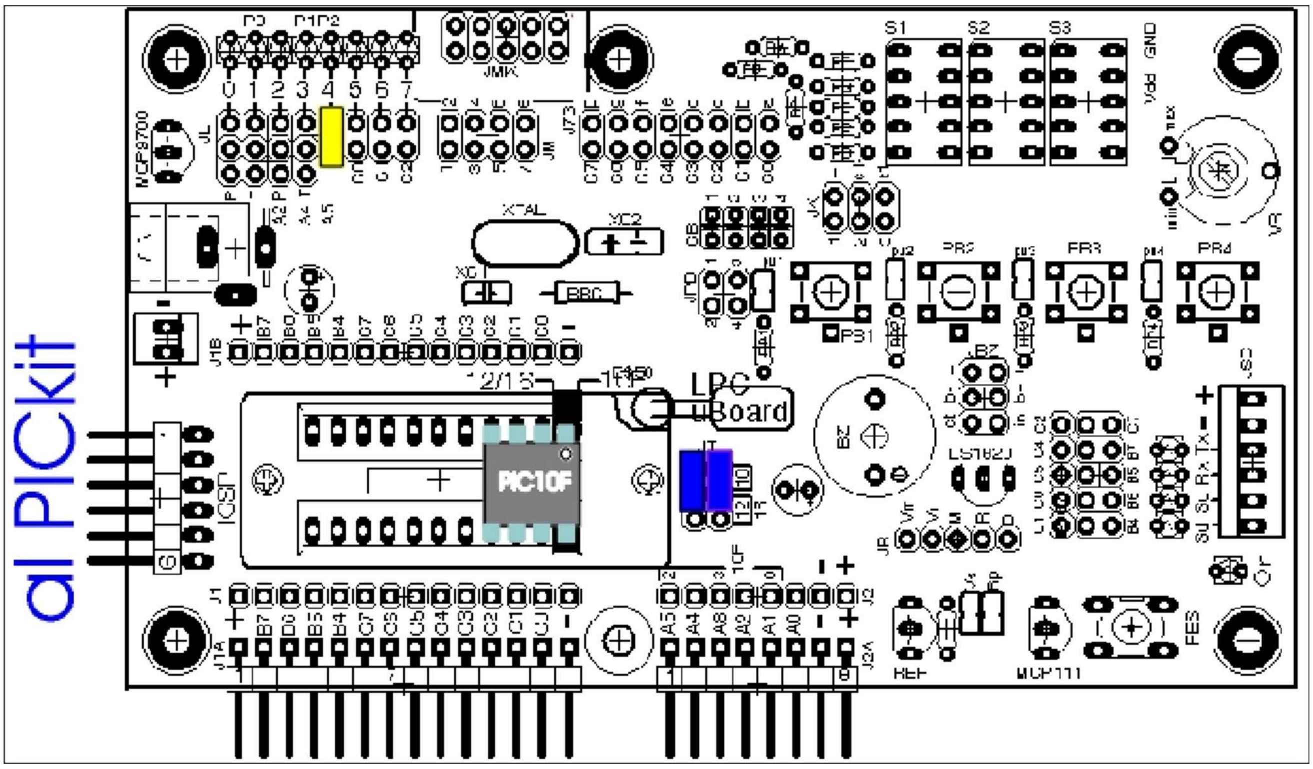
$$R_{led} = (V_{dd} - V_{led} - V_{oh}) / I_{led}$$

So, if ***Voh=0.4V*** we have:

$$R_{led} = (3 - 2 - 0.4) / 0.01 = 60\text{ohm}$$

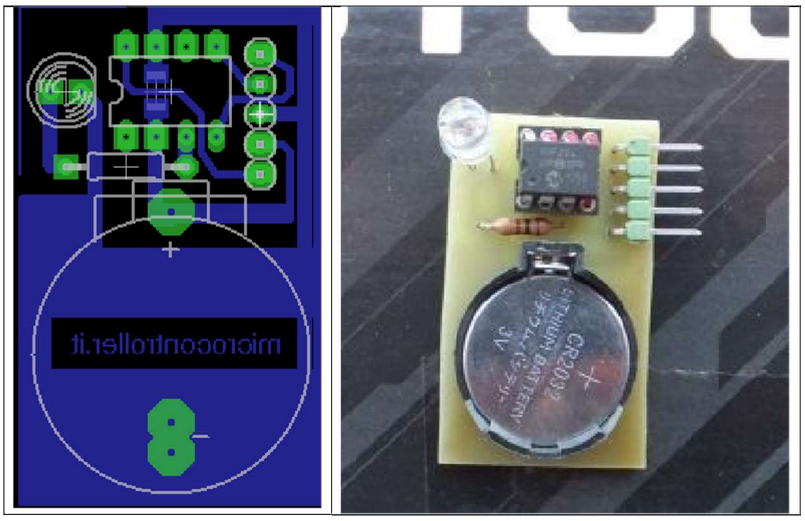
Which can be normalized to the commonly available value of 56ohm.
It should also be considered that the battery voltage tends to reduce with use and thus reduce the current in the LED; This will reduce the brightness slightly over time.
Since, however, the ports can nominally carry 25mA and that the current to the LED is delivered in pulses, with a low duty cycle, we can use a lower resistance than the one calculated, for example 33ohm, which allows the passage of about 18mA. With low current LEDs (2-5mA) it can be around 100ohm; For a precise calculation of the value, you can consult these pages.
However, the resistance depends on the LED, the brightness we want to obtain and the current consumption we can accept, obviously without exceeding the 25mA that can be managed by the digital output.
Of course, it is not possible to turn on LEDs whose voltage is greater than that of the battery; therefore, with the chosen battery we cannot think of using blue or white LEDs that have a conduction voltage higher than 3V with this circuit.

The circuit can be experienced on the **LPCuB**:



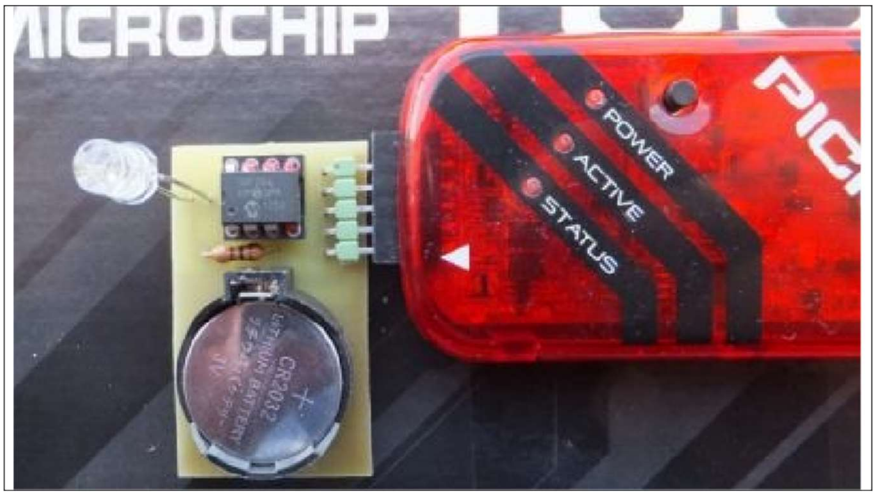
Here we can vary the period of the ignition pulses and their number, depending on what we want to achieve: the less time outside the sleep, the greater the consumption, but the greater the brightness obtained. The shorter the switch-on time or duty cycle, the lower the perceived brightness of the LED, but the lower the power consumption and the longer the battery life.
It is also possible to vary the intervention time of the WDT bringing it to 1s (prescaler 1:64), or even less, obtaining an acceleration of the flashes (higher average current absorbed).

The development board contains many current-drawing components, such as the green LED of ON and the VR potentiometer. Once the operation has been verified, you can also bring everything to a small print to have a finished object:

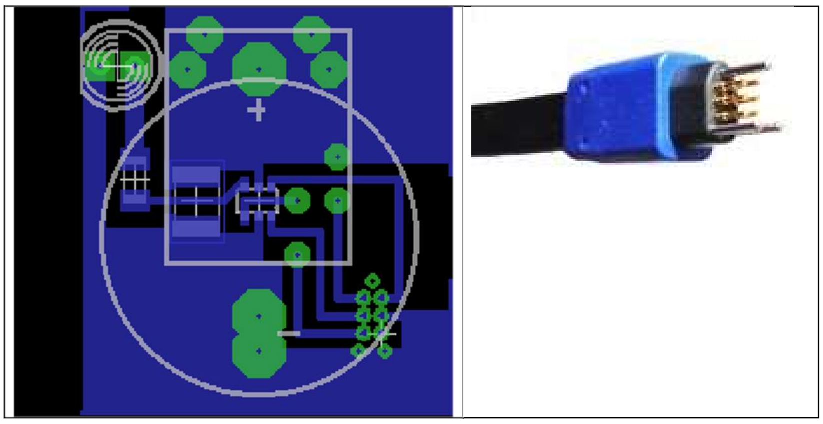


Used common components for hole mounting, apart from the capacitor parallel to the chip's power pins (0.1-4.7uF).

The circuit includes an ICSP socket to be able to connect the Pickit here as well and make changes to the firmware without removing the chip. This socket has only 5 pins for space reasons; Pin 6 is excluded, which is not used for programming these chips:



If you want to miniaturize, you can use the SOT-23 package, mounted on the opposite side of the battery. Here, too, we can provide an ICSP socket using the [Tag-connect](#):



The circuit can be used as a warning light for various devices, part of toys or to identify objects or obstacles in the dark or as a simple gadget; or, by controlling several LEDs, it can be the basis of a recognition light for cyclists or pedestrians at night. Multiple LEDs can be driven using multiple pins.

The same application, with the usual small modifications, will be able to run on any Baseline, and, in general, on any PIC.

Measurement of Vdd through the ADC module

We can use the ADC module to evaluate the supply voltage of the microcontroller. In many applications, it is necessary to evaluate this voltage: one of the most common cases is that of battery-powered systems, where it is necessary to keep the voltage under control to request replacement when it discharges or, in the case of accumulators, to avoid excessive discharge.

This is easily implemented in all chips that have an ADC module, using one channel and one input. The solution is obvious, but it occupies one pin and usually requires an additional one for an external reference voltage, since you can't use the Vdd as a reference because it is variable with the state of discharge of the batteries.
Unfortunately, the Baselines have their own reference for analog-to-digital conversion and only the Vdd. Therefore it is not possible to connect an analog input to this voltage, since the reference would also vary with it and therefore no evaluation of variations would be possible (since the result would always be identical).
You can use **the comparator**, but it's a pin-using solution, which is not welcome in low-I/O chips.

We can, however, use the internal reference at 0.6V, which is constant for any variation of the Vdd in the allowed operating range, making an "inverse" measurement. When we perform an AD conversion, the result is:

conversion result = value to be converted / (Vref / number of steps)

For example, if we have an 8-bit conversion available, 256 steps are possible. With a reference voltage of 5V, this means that each step is 5/256 = 19,531mV per step. Obviously, for a very precise measurement, we will try to have a very stable reference voltage available, perhaps a multiple of the number of steps. For example, with a 2048mV Vref each step is 2048/256 = 8mV. If the conversion yields 64h (100 decimal), the read value is 100 * 8 = 800mV.

All this should be clear: we have a stable reference voltage and a variable voltage to measure.
But let's look at it from the opposite point of view, which is that the value to be converted is stable and the reference voltage is variable.
If we have as a reference the Vdd at 5V (5000mV) and measure the reference voltage at 0.6V, we have:

Vref/step = 5000/256 = 19,531mV/step

and the conversion yields:

$$ADC = Vin / (V_{ref} / step) = 0.6 / 19.531 = 30.7 \text{ decimal} \rightarrow 1Eh.$$

Now let's say that Vdd, used with Vref+, drops to 4V; As a result, the resolution will be:

Vref/step = 4000/256 = 15,625 mV/step.

Therefore, the reading of the internal reference, **which remains constant**, will yield:

$$ADC = Vin / (V_{ref} / step) = 0.6 / 15.625 = 38.4dec \rightarrow 26h$$

If the Vdd voltage drops to 3V:

$$Vref/step = 3000/256 = 11.71875\ mV/step.$$

So the 0.6V reading will yield:

$$ADC = Vin / (V_{ref}/step) = 0.6 / 11.71875 = 51.2\ dec \rightarrow 33h$$

Decimal values are rounded down to the whole number, since the Assembler uses this principle. So the accuracy is limited, also because the voltage of the internal reference is low and the converter yields only 8 bits.

In general, in the case of the 8-bit AD converter (256 steps) and the measurement voltage at 0.6V (600mV):

$$ADC = 600 / ((V_{dd}*1000) / 256))$$

From which :

$$Vdd= 0.6 * (256 / ADC) = 153.6 / ADC$$

In practice, we can say that, with the variation of the voltage used as a reference for the measurement, the result of the conversion of a constant value, less than the reference voltage, gives **a value inversely proportional to the reference voltage** itself.

We can draw up a table:

Vdd	ADC		Vdd	ADC		Vdd	ADC		Vdd	ADC
5.5	1B		4.1	25		3.1	31		2.1	48
5.4	1C		4.0	26		3.0	33		2.0	4C
5.2	1D		3.9	27		2.9	34			
5.1	1E		3.8	28		2,8	36			
4.9	1F		3.7	29		2.7	38			
4.7	20		3.6	2Nd		2.6	3Rd			
4.6	21		3.5	2B		2.5	3-D			
4.5	22		3.4	2D		2.4	3F			
4.3	23		3.3	2E		2.3	42			
4.2	24		3.2	2F		2.2	45			

Using a spreadsheet it is easy to draw the table that relates the Vdd with the value obtained from the conversion to the Vref measure. Using this table you can derive a lookup table for the measurement of the Vdd or the limit value below which you must not fall. Alternatively, a calculation algorithm can be implemented, but usually the table is the simplest way, also considering that an indirect measurement of this kind typically has the purpose of monitoring the state of the voltage of the batteries that power the circuit and reporting the state of discharge.

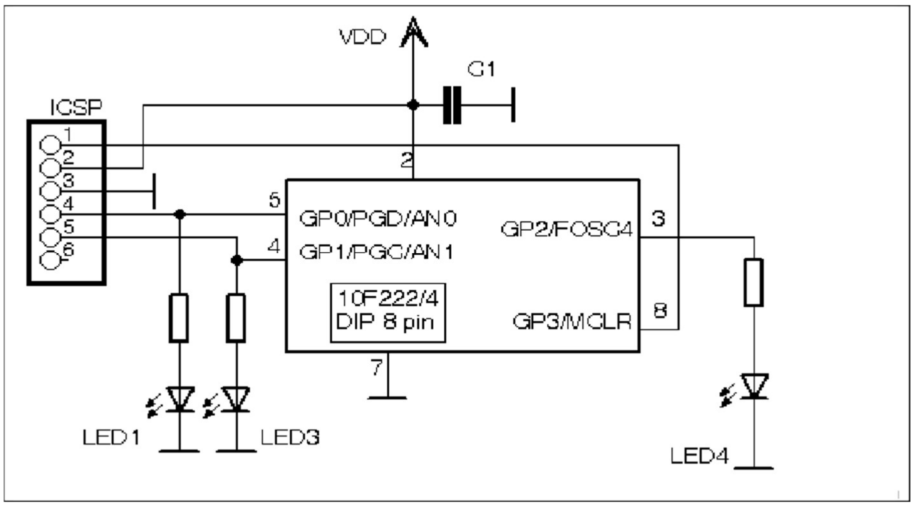
This "inverse" measurement has a limited accuracy compared to that obtainable with a "direct" method, but for the intended use, it is more than adequate and allows to evaluate differences of about ten millivolts or less on the Vdd. In addition, compared to a "direct" measurement with the ADC converter, it has significant advantages:

- No external pin is used, which is essential for chips with low pin counts
- It does not require an external divider which, in low power applications, is an unwelcome addition
- It does not require an external reference voltage for measurement accuracy, as it is "intrinsic" to the measurement method itself

As far as this last point is concerned, the precision of the voltage provided by the internal reference, as mentioned, is less than that which could be given by an external voltage reference, such as LM4040, REF-01, etc.: its value depends on the construction and chip characteristics, with a possible variation between 0.5V and 0.7V. Therefore, if you want to make precise measurements, a calibration is required.

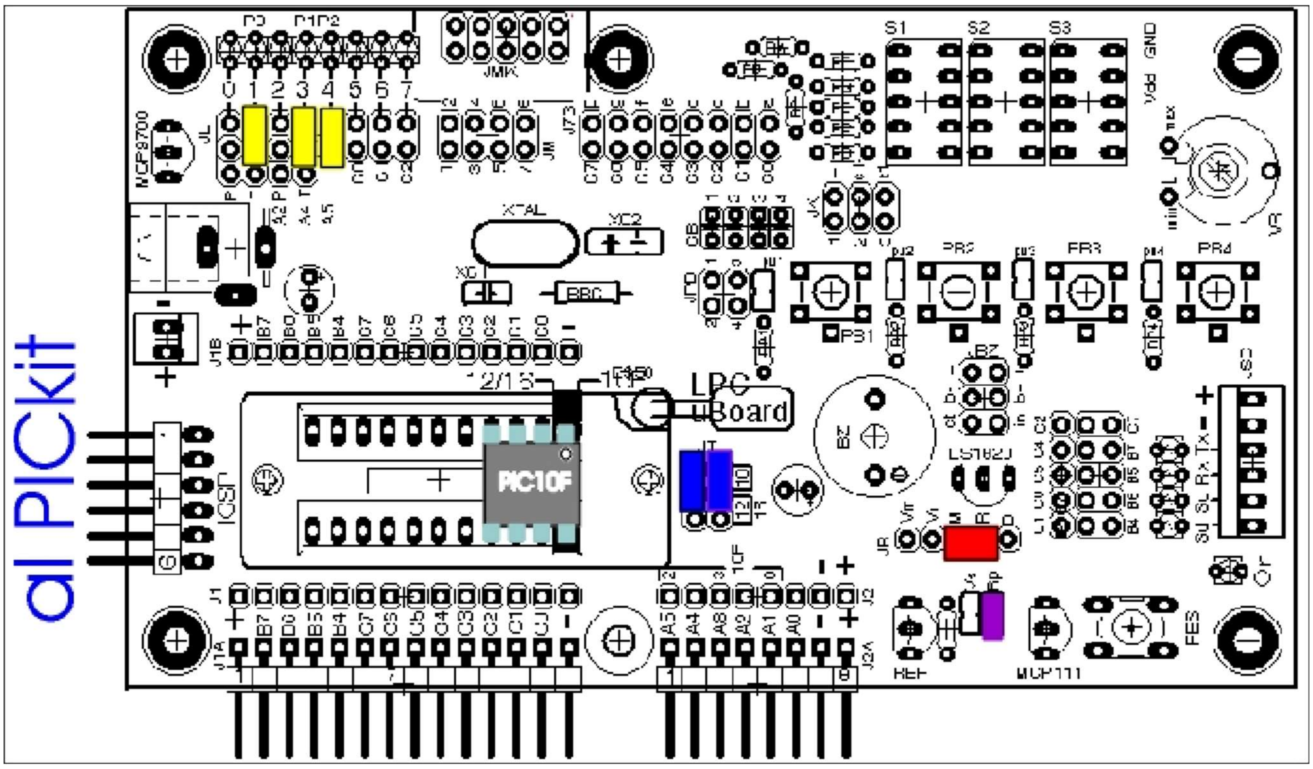
A test application

Let's first turn to the smallest PIC with ADC module, the 10F220 or 10F222, and use three LEDs to indicate the situation of the Vdd.



Note that no pins are used for analog measurement, as the ADC module is connected internally to the chip at the 0.6V reference source. The pins are used to control 3 LEDs that indicate the measured value of the supply voltage.

And on the LPCuB:



All you need is the "yellow" jumpers to connect the I/O to the LEDs and, of course, the "blue" ones for the processor. The RES button is not used and the "red" and "purple" jumpers are not essential.

The program.

We want to turn on the three LEDs in order to indicate three limits in the supply voltage, like this:

Vdd	ADC	LED		
		1	3	4
Vdd< 3.3	adc>2Eh	x	-	-
3.3<Vdd<4.5	2Eh>adc>22h	-	x	-
Vdd>4.5	ADC<22H	-	-	x

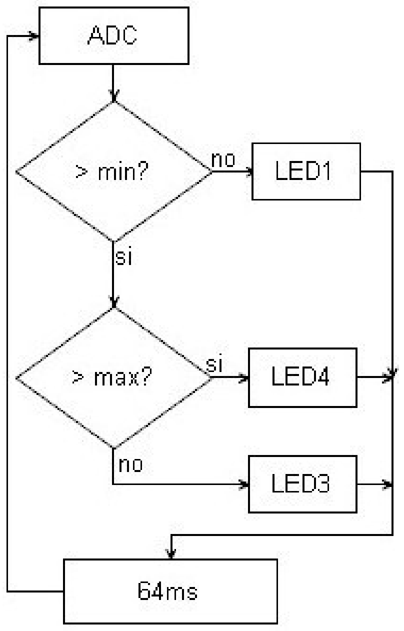
If we vary the Vdd, we will observe that the LEDs switch when the imposed thresholds are reached.

To obtain the indirect measurement, we connect the input of the ADC module with the internal reference.

No analog input needed:

```
; ADC for internal reference
    measurement movlw b'00111101'
           ; 00------ No analog inputs
           ; --11------------NTOSC/4
           ; ----11-- Internal Reference
           ; -------1 enable ADC movwf
    ADCON
```

After that, let's perform the conversion as we saw in the related tutorial and compare the result with the expected limits, according to this flowchart:



```
 ; Limits
 CONSTANT min = 0x2E      ; 3.0V
 CONSTANT max = 0x22      ; 4.5V

; ADRES > min? (Vdd>3.3V ?)
     CFLJA ADRES, min, next1
; ADRES < min
     movlw B'001'         ; LED 1 on
     movwf GPIO Goto
            Mainloop

; ADRES > max? (Vdd>4.5V ?)
next1    CFLJA ADRES, max, next2
; min<ADRES<max
     movlw B'010'         ; LED 3 on
     movWf GPIO
     Goto    Mainloop

; ADRES>max
next2
     movlw  B'100'        ; LED 4 on
     movwf  GPIO
     Goto   Mainloop
```

A macro **CFLJA** It makes it somewhat more convenient than the series of implied opcodes:

```
;*******************************************************************
; CFLJA Compare File to Literal and Jump if Above
; It compares the contents of a log with a number and goes to address
; If > literal file
CFLJA MACRO file,lit,addr
       movlw    (255-lit)    ; W = (~lit)
       addwf    file,W       ; W = file + W = file + (~lit)
       skpnc                 ; C=0 if file <= lit, exit
        goto    Addr         ; C=1 if file > lit, go to address
       ENDM
```

We use the `addwf sum` instead of the subwf subtraction, with the complement to 1 of the fixed number with which the comparison is made.
This macro, along with others, is offered in a ***compmacro.asm file*** that is included in the source.
We would like to remind you once again that the macros declared with the inclusion become part of the

Only those that are then used are compiled, while the others remain "virtual" without occupying program memory.
Subsequent AD conversions are interspersed with a 64ms delay routine obtained with the help of Timer0. In a real application, here it will be possible to insert other operations that the micro has to perform.

Obviously, it will be possible to verify the operation of the program only if you have a variable Vdd.

It should be noted that the AD conversion reports the measured value based on the Vref and the input voltage.
If the Vdd, which, here, is the reference of the conversion, is plagued by ripple or disturbances and these overlap with the nominal value during the sampling carried out by the ADC, the result obtained will be incorrect and plagued by seemingly random variations.

The way to correct the problem is to have a proper VDD. If this is not possible, you will need to implement an algorithm that averages multiple samples (although this method is not always decisive).

A more complex application.

Based on what we have seen so far, we can create a circuit that allows us to read the value of the measured voltage Vdd on a display.
For this, we take advantage of what we learned in Tutorial 11A, where we saw how you can transfer a value from a two-digit register to 7 segments.

We therefore have three "concurrent" processes to manage, which, as we said in exercise 10A, we actually run sequentially:

1. AD Conversion
2. Transpose the result to the display
3. Managing the Display Multiplexing

From the AD conversion we get a hexadecimal value, but we can transform it into an indication in volts through a lookup table, made according to the table seen above.
In the table, we relate the result of the conversion to the corresponding value of the voltage:

```
  ..
   addwf    PCL,f                 ; PC tip
;                #      ADC
   retlw    0x57   ;0    1B
   retlw    0x55  ; 1    1C
   retlw    0x53  ; 2    1D
   retlw    0x51  ; 3    1E
   retlw    0x50  ; 4    1F
   retlw    0x48  ; 5    20
   retlw    0x47  ; 6    21
   retlw    0x45  ; 7    22
   retlw    0x44  ; 8    23
  ..
```

We have to consider the values between 1Bh (derived from the measurement with a Vdd=5.7V) to 4Ch (derived from a Vdd=2V). That's 50 values, but that's not the problem, since the table can span up to 256 items (the search index in the table is W, which, being 8-bit, goes from 00 to FFh). The real problem is twofold: we have to start the table not from 1Bh, but from 0 and we have to avoid that, for whatever reason, the index goes beyond the end of the table.
If the index pointed past the end of the table, the codes contained in those locations would be executed, with unpredictable results.
For this reason, we insert a filter at the entrance of the table that prevents the entry of values that are too low or too high, sending them to an error message:

```
; lookup table to convert ADC from hex to volt
volttbl_r:
   CFLJB    Vtemp, min, aderL   ; if ADC<minimum, LL error
   CFLJA    Vtemp, max, aderH   ; if ADC>maximum, HH error    retlw
  ..
```

If the input value is less than the minimum of the table, you exit the table with an error. So too if it is higher than the maximum.
To start from 0 with the index, just subtract the input value of the table from the minimum value:

```
    movlw    min                        ; Parts from 0 in the
    subwf    ADRES,w
```

So, for example, if the conversion yields 1Bh, subtracting the minimum, which is 1Bh, we will have an index of 0 in W, which will point to the first in the table. If the value is 1Fh, subtracting the minimum will give you 4, which corresponds to the fifth row of the table. And so on.

To each value coming out of the ADC we match in the table the value in volts expressed with two digits, but in hexadecimal basis. These figures will be used to introduce them, one at a time, into a second lookup table that transforms them into the masks for lighting the segments, as we saw in Tutorial 5A:

```
; segment data table - common cathode display
segtbl_r
        andlw 0x0F      ; low nibble only
        addwf PCL,f      ; PC tip
        retlw  b'00111111'  ;  "0"  -|-|F|E|D|C|B|A
        retlw  b'00000110'  ;  "1"  -|-|-|-|-|C|B|-
        retlw  b'01011011'  ;  "2"  -|G|-|E|D|-|B|A
        retlw  b'01001111'  ;  "3"  -|G|-|-|D|C|B|A
        retlw  b'01100110'  ;  "4"  -|G|F|-|-|C|B|-
        retlw  b'01101101'  ;  "5"  -|G|F|-|D|C|-|A
        retlw  b'01111101'  ;  "6"  -|G|F|E|D|C|-|A
        retlw  b'00000111'  ;  "7"  -|-|-|-|-|C|B|A
        retlw  b'01111111'  ;  "8"  -|G|F|E|D|C|B|A
        retlw b'01101111'  ; "9" -|G|F|-|D|C|B|At
        retlw b'00111000'  ; "L" -|-|F|E|D|-|-|- exit for Ah
        retlw b'01110110'  ; "H" -|G|F|E|-|C|B|- exit to Bh retlw
  ......
```

Let's try to understand the mechanism.

1. For example, AD conversion yields a value **of 1Fh**.
2. We reduce it to **1Fh-1Bh** = **4**
3. **W** = **4** Tip the line `retlw      0x50 ; 4    1F`
4. So let's exit the table with **W=50h**
5. Let's enter the low nibble in the segment table, which is 0
6. We get the first line `retlw b'00111111'`
7. By returning this to the output port we will turn on the segments corresponding to the digit 0.
8. We pass the high nibble, i.e. 5, and we get the `retlw b'01101101' line`
9. By returning this to the output port, we will turn on the segments corresponding to digit 5.

In practice, the object of the `retlw` of the first table is the decimal value that the display will display through the second table: we carried out the conversion without using mathematical formulas, which would have taken up a much larger number of instructions.

It remains to be seen how the error messages work: the initial comparison macros refer to two outputs in the table:

```
aderL  retlw   0xAA    ; LL display
aderH  retlw   0xBB    ; HH display
```

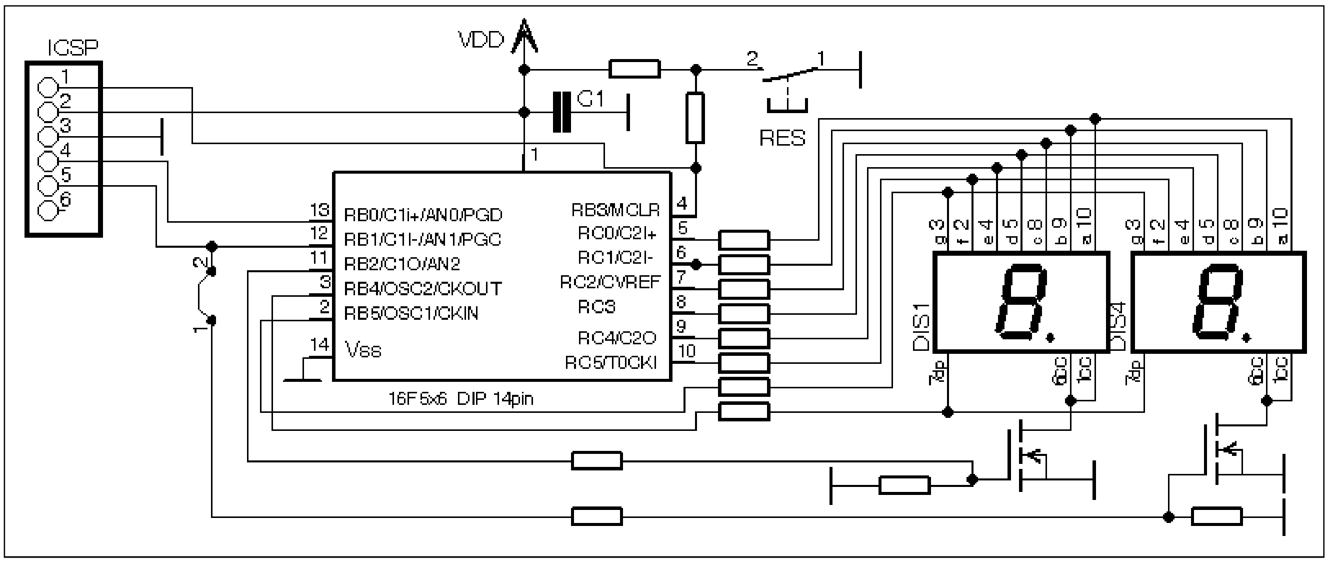
If the converted value is too low, you come out with **W=AAh**. Ah is the hexadecimal number after 9 and so the segment table points to the `retlw b'00111000'` row which will light up the

segments corresponding to the letter **L**.
Equally, a value above the maximum will result in **W = BBh**, and in the segments table this corresponds to the mask to turn on the letter **H**.

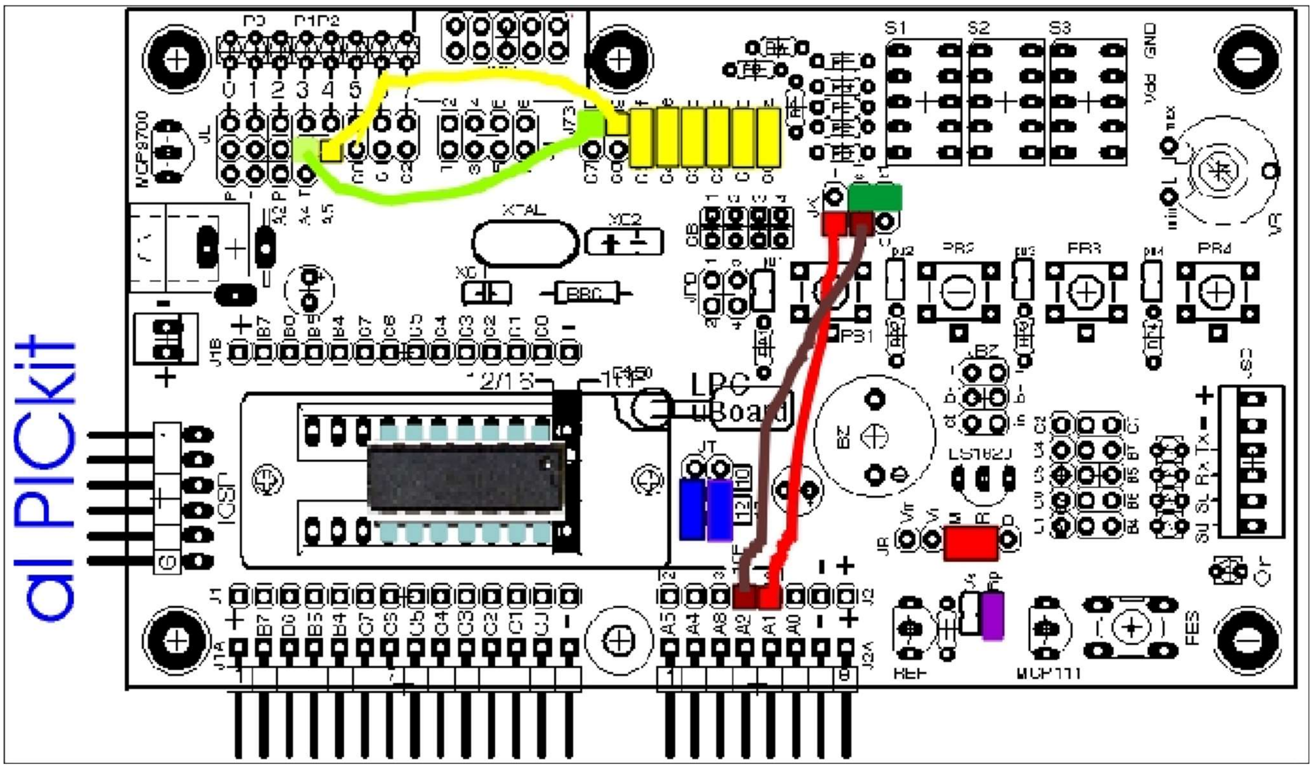
From a circuit point of view, we need a PIC that has:

- Enough pins to control the segments
- and the ADC module

so the choice falls on 16F506/526, for which we use this scheme:



And on the **LPCuB**:



"Yellow" jumpers connect RC5:0 to f:a segments
The "yellow" flying jumper connects RB5 to the g-segment

The "green" flying jumper connects RB4 to the decimal point
The "brown" flying jumper connects RB2 to the MOSFET gate of the units
(S2) The "red" flying jumper connects RB1 to the gate of the tens MOSFET
(S1) The "green" jumper connects to the gate of the MOSFET that drives the digit S1
The RES button is used and therefore the "red" and "purple" jumpers are required.

During the programming of the chip, it is necessary to detach the "red" flying jumper, i.e. disconnect the gate of the MOSFET of S1 from the ICSP socket, because the load on RB1/PGC would prevent the operation.

The program.

We have already outlined the essential points.
If we look at the source (***18A_526.asm***) we see that it follows exactly the structure of what we saw in exercise 11A: the timer 0 is the basis of the times that conditions the operation of the multiplex.

Here we use a cycle of about 4ms; the use of symbolic references also for constants allows you to quickly and easily modify this time (we could also use 2ms, as in the version of the 11A tutorial, since there would still be more than enough time for AD conversion and table management).

As for the ADC, we do the conversion every second or so, using a counter to total 256 4ms periods:

```
; AD Conversion
        decfsz cntr,f       ; Capture only every time * BitTMR
         Goto    dslp_3     ; 256 * 4ms = approx. 1s
        movlw    time       ; Charging Meter
        movwf    cntr
```

The choice is dictated by these considerations:

1. The variation of the supply voltage due to the decay of the battery is a fairly slow phenomenon. You don't need to monitor the Vdd at the ms level, but at the seconds level.
2. the result of the AD conversion is subject to almost certain fluctuations on the last digit. If we had a very rapid repetition of conversions, such as to provide the display with the last variable digit, the effect would be that we would not be able to read it.

In fact, the AD conversion itself is subject to an error of 1/2LSB and in addition to this, we are operating on values that make half of the 8 bits available, since the voltage measured (0.6V) is much lower than the reference (2-5.5V). We can see in the table how the result becomes more precise as the difference between the reference voltage and the voltage to be measured decreases: in the first part of the table the difference of 1 in the result of the conversion, due to the inverse measurement, derives from the variation of tens of millivolts in the Vdd. If we find ourselves with a Vdd affected by a minimum ripple and/or at an intermediate value between two in the table, the result will find itself "jumping" between two neighboring values. This can only be minimized by applying some averaging algorithm, but it can also be reduced by slowing down conversions so that any variation of the last digit is reported only at intervals of the order of seconds.

So, if we find ourselves with the diplay that goes from 4.8 to 5.0 it is probably because the Vdd has an intermediate value. Higher definition would be possible by using an ADC with a higher number of bits and a higher internal reference, such as those available in the PIC18F.
In the prototype used for the tutorial, a correction of the value output from the conversion has been introduced: the contents of the ADRES register are copied to a temporary location Vtemp, which will be used for subsequent actions, and here it is incremented by 1:

```
        movf    ADRES,w
        movwf   Vtemp
        incf    Vtemp,f     ; Fix +1
```

Microchip's AN1072, listed in the Bibliography, provides a lot of detail on the subject, even if it refers to Midrange/Enhanced Midrange and PIC18 which have much more complex and performing ADC modules than those of the Baselines.

For a better presentation, let's also turn on the decimal point of the tens digit:

```
        Bsf     segmdp      ; Add Decimal Point
        DECon               ; Turn on dozens
```

which will be turned off together with the digit itself (if we don't, the digit of the units would also have the dp on):

```
        DECoff              ; Turn off dozens
        Bcf     SEGMDP      ; Turn off decimal
```

The tables are placed in the first 256 bytes of page 1:

```
TABLES CODE 0x200
```

Let' **s remember the limits imposed by Baselines for the management of tables and subroutine calls**, which we have already discussed.
Tables and subroutines can be accessed indirectly by jumping from page 0, while `pageels` ensures proper indexing.
The position of the tables on page 1 allows you to have the main instructions on page 0, which is practically all available.

The source is written in a relocatable form. The intensive use of symbols and macro libraries allows the drafting of a fairly linear and simple source.

Also in this case a variable Vdd is needed to power the circuit.

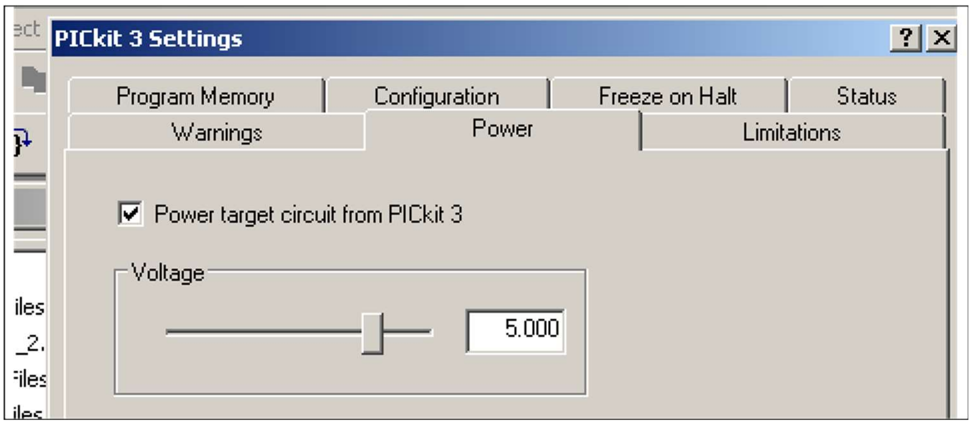
Vary the Vdd.

To test the above examples, you need a variable supply voltage for the microcontroller.

A first way is to have a power supply adjustable at least between 2 and 5V, to supply the Vdd to the circuit. The **UBS-12** module, designed to work with the **LPCuB, is excellent.**

If you don't have one or the other available, there is an alternative way: **use the voltage provided by the Pickit**.

We saw at the beginning of the course that the circuits in the examples can be fed directly from the Pickit. And the voltage this provides is adjustable, either as a programmer or as a debugger. From the ***Settings*** item of the relevant submenu, we must enable the Power function, where it is possible to select the value of the voltage generated with a slider:



☑ Power target circuit from PICkit 3

Voltage: 5.000

The voltage is theoretically variable between 2.75V and 5.5V. In practice, the maximum voltage will not exceed the value provided by the USB port that powers the Pickit and which is typically 5V (but can also be a 5% less).

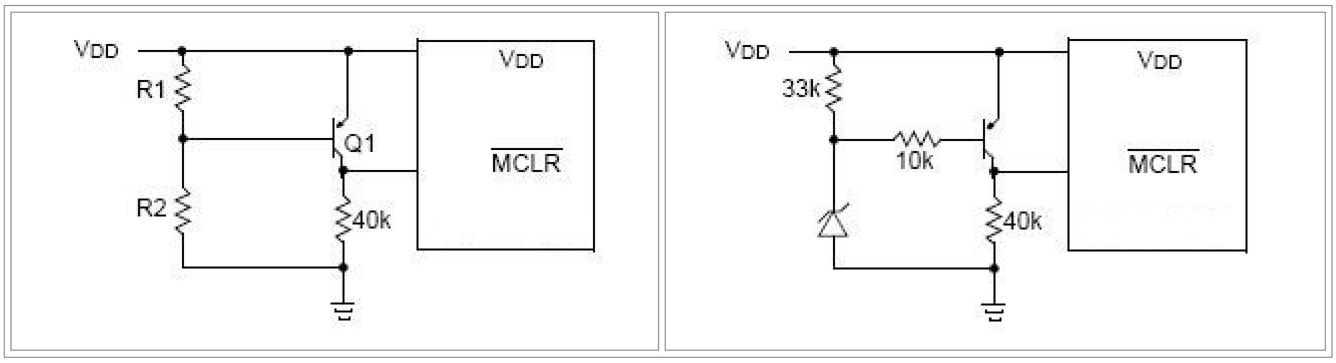
Brown Out Protection

Remaining in the field of supply voltage, it is necessary to mention the protection against brown-out. This function has nothing to do with the program, but it is an indispensable hardware support to avoid possible problems on the power supply.

Brown Out is a condition of the supply voltage that falls below the minimum safe value required for the operation of the chip or circuit.
In this case, the voltage does not reach zero or even the value that allows the operation of the reset circuit (<0.8V) which, therefore, does not intervene.
Baselines can operate between 2 and 5.5V on the Vdd and, as we have seen in relation to the clock, the lower the voltage, the lower the possible operating frequency. In addition, below 1.5V on the Vdd it is not guaranteed that data will be kept in RAM. As a result, if the Vdd drops even briefly below 2V, without getting close to 0, it is likely that the chip will stop working properly and in an unpredictable way.
These conditions occur in the case of short mains failures that cannot be compensated by the electrolytics of the power supply or with the use of batteries that at the end of charge or defects can fall below 2V.

For this reason, from the Midrange upwards, the PICs incorporate a circuit called ***Brown Out Protection*** which is intended to reset the chip if the power supply is faulty in the direction indicated.
However, **the Baselines do not have this module** and, in case you are dealing with a power supply that can present problems, you need to add an external circuit, connected to the reset pin (**`MCLR`**).

Microchip recommends a few simple circuits:

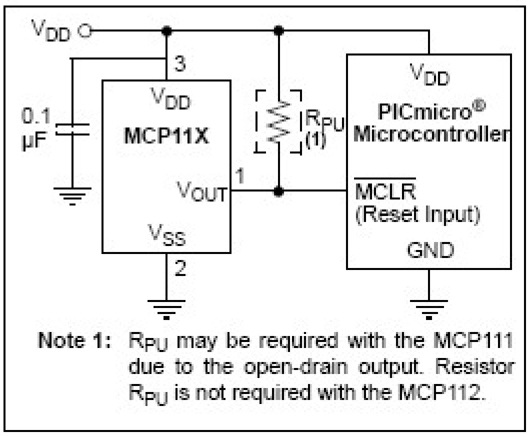


These are two triggers that connect the MCLR pin to the Vdd until it is greater than a certain threshold.
In the first circuit, ***Q1*** is turned off when the Vdd drops below the conduction voltage of the transistors:

$$Vdd * R1/(R1+R2)$$

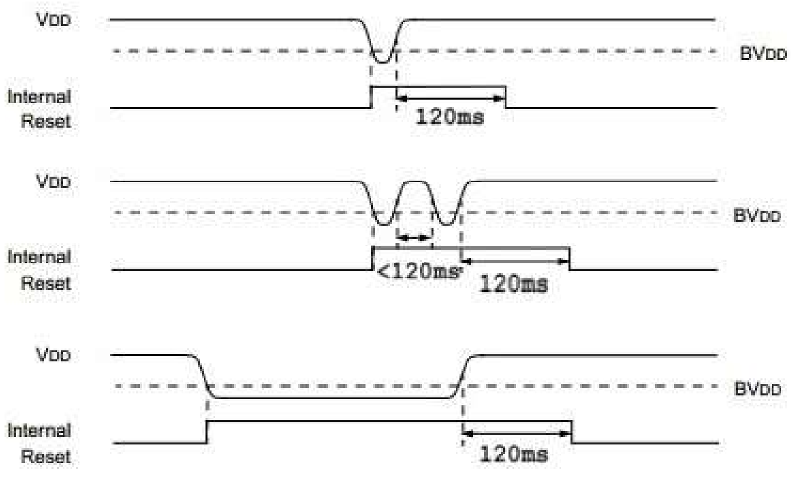
This value is around 0.7V, but it depends a lot on the transistor (which will be any PNP, type BC807, BC307, etc.) and must be searched experimentally. For greater accuracy, the second circuit uses a zener diode. In this case you will have the transistor lock for ***Vdd < Vz + 0.7V***.
In both cases, these are circuits that essentially aim for low cost, while the tripping threshold must be verified according to the characteristics of the semiconductors used.

From the point of view of professional achievement, it makes much more sense to use a ***Voltage Supervisor***, i.e. an integrated circuit that allows a high precision of intervention, two elements that compensate for the higher cost. In particular, those of the Microchip MCP11x series are adequate. The **LPCuB** board is designed for the insertion of one of these components on a special socket.



VDD
3
0.1 µF
VDD
MCP11X
VOUT
1
VSS
2
RPU (1)
VDD
PICmicro® Microcontroller
MCLR (Reset Input)
GND
Note 1: RPU may be required with the MCP111 due to the open-drain output. Resistor RPU is not required with the MCP112.

Using an **MCP111** you will need to enable the pull-up with ul jumper ***Rp***, while using an **MCP112** this will not be necessary. These ICs are built for different voltages (1.90V, 2.32V, 2.63V, 2.90V, 2.93V, 3.08V,
4.38V and 4.63V) and therefore adapt to the most varied conditions.
The choice of tripping value will depend on the minimum expected working voltage; for example, if the circuit has a well-stabilized Vdd of 5V, a trip at 4.63V, below 4.75V (5V-5%) is adequate. If you power the circuit with two 1.5V batteries or one 3V lithium battery, you can choose an intervention between 2.32 and 2.63V depending on the minimum voltage the circuit components can sustain and that of the accumulator, to avoid over-discharge.
In addition, what is connected to the microcontroller must also be considered; for example, if an external peripheral can work up to a Vdd of 2.5V, even if the micro reaches 2V (and even less..), it will be necessary to provide with a threshold of at least 2.63V to ensure safe operation. MCP11x do not introduce a delay in the output pin release when the voltage returns above the tripping value; The delay between the release of the reset and the start of the micro is that established by the characteristics of the chip.



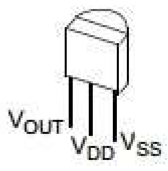
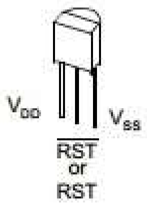
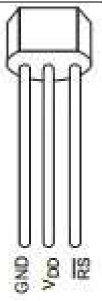
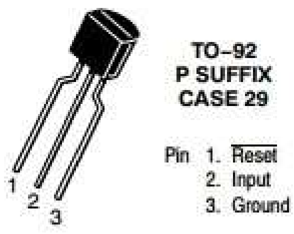
Instead, the MCP10x, 12x, and 13x models add 120ms. The behavior is as indicated on the side.

The insertion of a delay could be positive if the possibility of a rapid recurrence of the undervoltage is foreseen, as in the second image.

Or, even, when an extension of the reset time is necessary for other components connected to the microcontroller.

It should be noted that MCP10x, 12x and 13x have a different pinout from MCP11x and therefore the insertion on the plinth is not direct, but it is necessary to refer to the data sheet of the component used.

Just to get an idea of the possible variations:

			 TO-92 P SUFFIX CASE 29 Pin 1. Reset 2. Input 3. Ground 1 2 3
MCP111/2, MCP10xD, TC54	**MCP10xH**	**TC32M, TL7757**	**MC33064, DS1812, AMS26, TS831**

If you are using one of the many other similar components (TS831, DS1233, TC32M, TC54, EC95809, DS1812, DS1818, MC33064/33164, TL7757, AMS26, MN1380, etc.) you will need to Refer to the relevant data sheet for the correct insertion on the card and for the presence or absence of the pull-up.
Models with a push-pull output can be chosen, but in this case, the reset button cannot be used (which would short-circuit the high MOSFET). Open drain ones are therefore preferable, in which closing the reset button does not create problems.

The insertion of the Voltage Supervisor does not involve any change in the software, except that an analysis of the causes of Reset will reveal conditions equal to those due to the intervention of a low level on the MCLR pin.

To verify this, you will need to install a voltage detector, for example MCP111, in the version for 3.08V (but any other will do). It is then necessary to connect the LPCuB to a power supply variable between 2 and 5V. Once any of the above programs have been loaded, if the supply voltage is higher than the voltage detector tripping voltage, the circuit will operate normally; by lowering the voltage, the program will continue to run until it falls below the threshold. At this point, the program will stop and remain in reset conditions until a higher voltage returns.

BIBLIOGRAPHY

Some links to application notes of interest

Using voltage regulator to convert 5-12V range to 3.3V
Improving the Transient Immunity - Freescale
Hardware design guideline power supply and voltage measurement – ST
External Brown-out Protection – Atmel
TRIAC+Microcontroller Safety Precautions – ST
Protecting Microcontroller Systems against Power-Supply Imperfections – Philips
Transformerless Power Supply – Microchip
PRINTED CIRCUIT BOARD DESIGN NOTES – Silabs
AN1072 Measuring VDD Using the 0.6V Reference - Microchip

17A_10F.asm

```
;*********************************************************************
; 17A_10F.asm
;---------------------------------------------------------------------
;
;    Title          : Assembly&C Course - Tutorial 17A
;                      An LED connected to GP2 is flashing.
;                      ton=20ms toff=2,3s in sleep for low power consumption.

;    PIC            : 10F200/2/4/6
;    Support        : MPASM
;    Version        : 1.0
;    Date           : 01-05-2013
;    Hardware ref.  :
;    Author         :Afg
;
;
; Pin use :
;  ----------------
;     10F200/2/4/6 @ 8 pin DIP       10F200/2/4/6 @ 6 pin SOT-23
;
;                |‾‾\/‾‾|                    *‾‾‾‾‾‾|
;           NC -|1    8|- GP3         GP0 -|1    6|- GP3
;          Vdd -|2    7|- Vss         Vss -|2    5|- Vdd
;          GP2 -|3    6|- NC          GP1 -|3    4|- GP2
;          GP1 -|4    5|- GP0              |______|
;                |______|
;
;                                DIP  SOT
;    NC                           1:
;    Vdd                          2:   5: +
;    GP2/T0CKI/FOSC4/[COUT]       3:   4: Out LED at Vss
;    GP1/[CIN-]/ICSPCLK           4:   3:
;    GP0/[CIN+]/ICSPDAT           5:   1:
;    NC                           6:
;    Vss                          7:   2: -
;    GP3/MCLR/VPP                 8:   6:
;
;    [] only 204/6
;
;*********************************************************************
;             DEFINITION OF PORT USE
; GPIO map
; | 3 | 2 | 1 | 0 |
; |-----|-----|-----|--------|
; | in | LED |      |      |
;
;#define     GPIO,GP0    ;
;#define     GPIO,GP1    ;
#define     LED GPIO,GP2    ; LED between pin and Vss
;#define     GPIO,GP3    ;
;
; ######################################################################
; Processor Definition
 #ifdef___10F200
```

```
        LIST      p=10F200
        #include <p10F200.inc>
 #endif

 #ifdef __10F202
        LIST      p=10F202
        #include <p10F202.inc>
 #endif
 #ifdef __10F204
        LIST      p=10F204
        #include <p10F204.inc>
#define proccomp
 #endif
 #ifdef __10F206
        LIST      p=10F206
        #include <p10F206.inc>
#define proccomp
 #endif
        Radix     DEC          ; Decimal Numbers Defaults

; ####################################################################
;                          CONFIGURATION
;
; Yes WDT, no CP, pin4=GP3;
  __config  _CP_OFF & _MCLRE_OFF  & _WDT_ON


; ####################################################################
;                          RAM
    CBLOCK 0x10
    d1,d2,d3               ; Counter for ENDC timing

step EQU .10               ; Flashing repetitions

; ####################################################################
;                           RESET ENTRY
;
; Reset Vector
RESET_VECTOR   ORG     0x00

        clrwdt
; Internal Oscillator Calibration
        ANDLW   0xFE
        movwf   OSCCAL
; disable T0CKI to have free GP2
; prescaler 1:128 to WDT
; OPTION def '11111111'
;             1------- GPWU disabled
;             -1------ GPPU disabled
;             --0----- internal clock (no T0CKI)
;             ---1---- done
;             ----1--- prescaler at WDT
;             -----111 1:128 approx. 2.3s
     movlw b'11011111'
     OPTION

 #ifdef proccomp    ; only 10F204/6
; Disable Comparator
```

```
     movlw b'11110111'
     movwf CMCON0
 #endif

; GP2 out
     movlw   B'11111011'
     Tris    GPIO

     movlw   Step
     movwf   d3
ll   Bsf     LED
     Call    Delay1ms
     Bcf     LED
     Call    Delay1ms
     decfsz d3
      Goto   ll

     sleep


; Delay1ms
subroutines
        movlw   0xC6
        movwf   D1
        movlw   0x01
        movwf   d2
Delay1ms_0
        decfsz  D1, F
         Goto   $+2
        decfsz  D2, F
         Goto   Delay1ms_0
        Goto    $+1
        Nop
        retlw   0

;*******************************************************************
;                            THE END
    END
```

17A_10FVdd.asm

```
;**********************************************************************
; 17A_10FVdd.asm
;----------------------------------------------------------------------
;
;    Title          :  Assembly & C Course - Exercise 17A
;                      Evaluation of the Vdd through the indirect
;                      measurement of the reference voltage of the
;                      ADC module.

;    PIC            :  Code 10F220/222
;    Support        :  MPASM
;    Version        :  1.0
;    Date           :  01-05-2013
;    Hardware Ref.  :
;    Author         :  Afg
;
;
;  Pin use :
;  ----------------
;     Code          8 DIP pin       10F220/222 @ 6-pin SOT-23
      10F220/222 @
;
;                |¯¯\/¯¯|                    *¯¯¯¯¯¯|
;           NC -|1    8|- GP3        GP0 -|1    6|- GP3
;          Vdd -|2    7|- Vss        Vss -|2    5|- Vdd
;          GP2 -|3    6|- NC         GP1 -|3    4|- GP2
;          GP1 -|4    5|- GP0             |______|
;                |______|
;
;                           DIP  SOT
;    NC                      1:     Nc
;    Vdd                     2:  5: ++
;    GP2/T0CKI/FOSC4         3:  4: Out LED
;    GP1/ICSPCLK/AN1         4:  3: Out LED
;    GP0/ICSPDAT/AN0         5:  1: Out LED
;    NC                      6:     Nc
;    Vss                     7:  2: --
;    GP3/MCLR/VPP            8:  6:
;
;**********************************************************************
;            DEFINITION OF PORT USE
; GPIO map
; | 3 | 2 | 1 | 0 |
; |-----|-----|-----|--------|
; |     |LED4 |LED3 |LED1 |
;
#define    LED1 GPIO,GP0    ; LED between pins
and Vss #define             LED3 GPIO,GP1 ;
#define    LED4 GPIO,GP2    ;
;#define    GPIO,GP3   ;
;
; #####################################################################
 #ifdef ___10F220
        LIST      p=10F220     ; Processor Definition
        #include <p10F220.inc>
 #endif
 #ifdef __10F222
        LIST      p=10F222      ; Processor Definition
        #include <p10F222.inc>
 #endif

        Radix     DEC           ; Decimal Numbers Defaults

; ####################################################################
;                          CONFIGURATION
;
; 4MHz, no WDT, no CP, pin4=GP3;
  __config  _IOFSCS_4MHZ  & _CP_OFF & _MCLRE_OFF  & _WDT_OFF


; ####################################################################
;                          RAM

; ####################################################################
;                          CONSTANT
                           S
; Limits
 CONSTANT  min = 0x33   ; 3.0V
 CONSTANT  max = 0x22   ; 4.5V

; ####################################################################
;                         MACRO
; Macro Comparison Set
 #include C:\PIC\Library\Baseline\compmacro.asm

; ####################################################################
;                           RESET ENTRY
;
; Reset Vector
RESET_VECTOR   ORG     0x00


; ####################################################################
;                         MAIN PROGRAM
;
Main:
; Internal Oscillator Calibration
        ANDLW   0xFE
        movwf   OSCCAL

; disable T0CKI to have free GP2
; prescaler 1:256 to Timer0
; OPTION def '11111111'
;            1------- GPWU disabled
;            -1------ GPPU disabled
;            --0----- internal clock (no T0CKI)
;            ---1---- done
;            ----0--- prescaler to Timer0
;            -----111 1:256
     movlw b'11011111'
     OPTION

; ADC for Internal Reference Measurement
; 00------ No analog inputs
```

```
              ; --11------------NTOSC/4
              ; ----11-- Internal Reference
              ; ------0- GO/Done
              ; -------1 enable ADC movlw
     b'00111101'
     movwf ADCON0

; GP out
     movlw b'11111000'
     tris   GPIO

Mainloop:
; Delay 64ms
     movlw .6
     movwf TMR0
LL1 MOVF    TMR0,w
     bnz    LL1

; Start BSF
     Conversion
            ADCON0,GO
adlp btfsc ADCON0,NOT_DONE
      goto adlp

; ADRES > min? (Vdd>3.3V ?)
     CFLJA ADRES, min, next1
; ADRES < min
```

```
     movlw b'001'        ; LED 1 on
     movwf GPIO
     Goto   Mainloop

; ADRES > max? (Vdd>4.5V ?)
next1
     CFLJA ADRES, max, next2
; min<ADRES<max
     movlw b'010'        ; LED 3 on
     movWf GPIO
     Goto   Mainloop

; ADRES>max
next2
     movlw b'100'        ; LED 4 on
     movwf GPIO
     Goto   Mainloop


;**********************************************************************
;                                THE END
```

```
        END
```

17A_526.asm

```
;*******************************************************************
; 17A_526.asm
;-------------------------------------------------------------------
;
;    Title          : Assembly&C Course - Tutorial 17A
;                     Indirect measurement of Vdd with ADC result
;                     in volts in two digits
;
;    PIC            : 16F506/526
;    Support        : MPASM
;    Version        : V.519-1.0
;    Date           : 01-05-2013
;    Hardware ref. :
;    Author         :Afg
;
;-------------------------------------------------------------------
;
; Pin use :
;  ----------------
;       16F506/526 @ 14 pin
;
;                   |¯¯\/¯¯|
;             Vdd -|1   14|- Vss
;             RB5 -|2   13|- RB0
;             RB4 -|3   12|- RB1
;        RB3/MCLR -|4   11|- RB22
;             RC5 -|5   10|- RC0
;             RC4 -|6    9|- RC1
;             RC3 -|7    8|- RC2
;                   |______|
;
;    Vdd                       1: ++
;    RB5/OSC1/CLKIN            2: Out   Segm G
;    RB4/OSC2/CLKOUT           3:
;    RB3/! MCLR/VPP            4: MCLR
;    RC5/T0CKI                 5: Out   Segm F
;    RC4/C2OUT                 6: Out   et seq.
;    RC3                       7: Out   Segm D
;    RC2/CVref                 8: Out   Segm C
;    RC1/C2IN-                 9: Out   Segm B
;    RC0/C2IN+                10: Out   Segm A
;    RB2/C1OUT/AN2            11: Out   Gate Units
;    RB1/C1IN-/AN1/ICSPC      12: Out   Gate Tens
;    RB0/C1IN+/AN0/ICSPD      13:
;    Vss                      14: --
;
; #####################################################################
; Choice of #ifdef
  processor 16F526
         LIST      p=16F526
         #include <p16F526.inc>
  #endif
  #ifdef____ 16F506
        LIST      p=16F506
         #include <p16F506.inc>
  #endif

        radix dec

; ####################################################################
;                          CONFIGURATION
  #ifdef____ 16F526
; Internal Oscillator, 4MHz, No WDT, No CP, MCLR
 __config  _IntRC_OSC_RB4 & _IOSCFS_4MHz & _WDTE_OFF & _CP_OFF &

_CPDF_OFF & _MCLRE_ON
  #endif
  #ifdef____ 16F506
; Internal Oscillator, 4MHz, No WDT, No CP, MCLR
  __config  _IntRC_OSC_RB4EN & _IOSCFS_OFF & _WDT_OFF & _CP_OFF &
_MCLRE_ON
  #endif


; ####################################################################
;                           RAM
; general purpose RAM
```

```
    UDATA                        ; RAM area
Temp     RES 1                   ; temporary
Vtemp    RES 1                   ; Temporary ADC
cntr     RES 1                   ; Counter


 ;******************************************************************
;==================================================================
;            DEFINITION OF PORT USE
;
;P ORTC map
;| 5 | 4 | 3 | 2 | 1 | 0 |
;|-----|-----|-----|-----|-----|---------|
;|segmf|segme|segmd|segmc|segmb|segma|
;
#define      Segma  PORTC,0     ; F:A Segments
#define      segmb  PORTC,1     ;
#define      SEGMC  PORTC,2     ;
#define      SEGMD  PORTC,3     ;
#define      Segme  PORTC,4     ;
#define      SEGMF  PORTC,5     ;
```

```
;P ORTB map
;| 5 | 4 | 3 | 2 | 1 | 0 |
;|-----|-----|-----|-----|-----|---------|
;|segmg|     |     | AN2 |GATE1|GATE2|
;
;#define          PORTB,0    ;
#define     GATE1 PORTB,1    ; MOSFET gate digits tens
#define     GATE2 PORTB,2    ; Gate MOSFET Digit Unit
;#define          PORTB,3    ; MCLR
#define     segmdp PORTB,4   ; DP Segment
#define     segmg PORTB,5    ; Segment G
;
; ####################################################################
;                          CONSTANTS
;
#define  bit05k  TMR0,1 ; TMR0 bit for 512us
#define  bit1k   TMR0,2 ;            1024us
#define  bit2k   TMR0,3 ;            2048us
#define  bit4k   TMR0.4 ;            4096us
#define  bit8k   TMR0.5 ;            8192us
```

```
#define bittmr   bit4k  ; 4096us
time    EQU 0           ; Number of CCLIs between
                        conversions
                        ; 256 x 4096us = 1s

; ADC result limits
 constant min = 0x1B
 constant max = 0x4C

; ######################################################################
;                            LOCAL MACROS
; UNITon MOSFET gate
command  MACRO
        bsf GATE2
          ENDM
UNIToff MACRO
        bcf GATE2
          ENDM
DECon   MACRO
        bsf GATE1
          ENDM
```

```
DECoff MACRO
        bcf GATE1
          ENDM

; General Macros
 #include C:\PIC\Library\Baseline\compmacro.asm

; ####################################################################
;                         RESET ENTRY
;
; Reset Vector
        TAILS    0x00

; MOWF Internal Oscillator
        Calibration OSCCAL

; Skip to make room for indirect vectors for pageel start subroutines
        Goto    Start

;-----------------------------------------------------------------
```

```
; Subroutines Vectors
ADConv:                  ; AD pageel
        conversion ADConv_r
        Goto    ADConv_r
VoltTBL:;  Lookup Table for Pageel volttbl_r Voltages
        Goto    volttbl_r
segtbl:                  ; Lookup Table Pageel Segment Masks
        segtbl_r
        Goto    segtbl_r

; ####################################################################
;                       MAIN PROGRAM
Main     TAILS

Start:
; Disable comparators to free the BCF digital function
                CM1CON0, C1ON
        Bcf     CM2CON0, C2ON

; disable T0CKI from RB5, prescaler 1:256 to Timer0
```

```
         ;      b'11010111'
         ;        1-------     GPWU Enabled
         ;        -1------     GPPU Enabled
         ;        --0-----     Internal Clock
         ;        ---1----     Falling
         ;        ----0---     prescaler to Timer0
         ;        -----111     1:256
        movlw   b'11010111'
        OPTION

        CLRF    PORTB     ; Clear Port Latch
        CLRF    PORTC

; All useful ports come out
        movlw   0
        Tris    PORTB
        Tris    PORTC

; ADC for Internal Reference Measurement
                ; 00------ No analog inputs
                ; --11 ----------- NTOSC/4
                ;---------11--  internal reference
                ;------------ 0-  GO/Done
                ;-------------- 1  ADC Skills
        movlw   b'00111101'
        movwf   ADCON0

        movlw   time      ; Conversion Cadence Counter
        movwf   cntr

        CLRF    TMR0      ; Initialize Timer

;------------------------------------------------------------------
; Display Cycle
; Digit Unit
displp btfss    bittmr     ; 2.048ms ?
         Goto   displp     ; No -
         Waiting
        movf    Vtemp,w    ; Yes - Load ANDLW Conversion
        Result  0x0F       ; Low nibble only
        pagesel segtbl
        Call    segtbl     ; lookup table
        pageel $
        movwf   Temp         ; Save to Temporary
        Movwf   PORTC        ; Switch to Port SEGM
        F:A BTFSC            temp,6   ; Command G-
        Segment
         Bsf    segmg
        BTFSS   temp,6
         Bcf    segmg
        UNITon               ; Unit digit lit
dslp_1 BTFSC    bittmr       ; 2.048ms ?
         Goto   dslp_1       ; no - waiting
dslp_2 UNIToff               ; Yes - Turn off
units
; digit tens
; swap Vtemp in W, low nibble<->high nibble
        swapf   Vtemp,w
        andlw   0x0F         ; just nibble bass
        pagesel segtbl
        Call    segtbl       ; lookup table
        pageel $
        movwf   Temp
        movwf   PORTC
        BTFSC   temp,6
         Bsf     segmg
        BTFSS   temp,6
         BCF     segmg
        Bsf     segmdp       ; Add Decimal Point Turn
        DECon                ; On Tens

; AD Conversion
        decfsz  cntr,f       ; Capture only every time * BitTMR
         Goto   dslp_3       ; 256 * 4ms = approx. 1s
        movlw   time         ; Charging Meter
        movwf   cntr
        pagesel ADConv
        call    ADConv       ; AD Conversion +1
        movf    ADRES,w
        movwf   Vtemp
        incf    Vtemp,f      ; Correction
        pagesel volttbl
        call    volttbl      ; Table by value in volts
        pagesel $
        movwf   Vtemp        ; Save for Later Processing

; end of time ?
```

```
dslp_3  BTFSS   bittmr      ; 2.048ms ?
         Goto   dslp_3      ; No - Waiting
        DECoff              ; Yes - Turn off tens
        Bcf     SEGMDP      ; Turn off decimal
                            point
        Goto    displp      ; Loop


TABLES CODE 0x200
; ####################################################################
;             TABLES AND SUBROUTINES AREA in PAGINA1
;--------------------------------------------------------------------
; lookup table to convert ADC from hex to volts
volttbl_r:
   CFLJB   Vtemp, min, aderL ; if ADC<minimum, LL CFLJA
   error   Vtemp, max, aderH ; if ADC>maximum, HH error
   movlw   Min                 ; Start from 0 in the SUBWF
   table   Vtemp,w
   addwf   PCL,f               ; PC tip
;                #     ADC
   retlw   0x57 ;0     1B
   retlw   0x55 ; 1    1C
   retlw    0x53   ; 2   1D
   retlw    0x51   ; 3   1E
   retlw    0x50   ; 4   1F
   retlw    0x48   ; 5   20
   retlw    0x47   ; 6   21
   retlw    0x45   ; 7   22
   retlw    0x44   ; 8   23
   retlw    0x43   ; 9   24
   retlw    0x42   ; 10  25
   retlw    0x40   ; 11  26
   retlw    0x39   ; 12  27
   retlw    0x38   ; 13  28
   retlw    0x37   ; 14  29
   retlw    0x37   ; 15 2Nd
   retlw    0x36   ; 16  2B
   retlw    0x35   ; 17  2C
   retlw    0x34   ; 18  2D
   retlw    0x33   ; 19  2E
   retlw    0x33   ; 20  2F
   retlw    0x32   ; 21  30
   retlw    0x31   ; 22  31
   retlw   0x31  ; 23  32
   retlw   0x30  ; 24  33
   retlw   0x30  ; 25  34
   retlw   0x29  ; 26  35
   retlw   0x28  ; 27  36
   retlw   0x28  ; 28  37
   retlw   0x27  ; 29  38
   retlw   0x27  ; 30  39
   retlw   0x26  ; 31 3Rd
   retlw   0x26  ; 32  3B
   retlw   0x26  ; 33  3C
   retlw   0x25  ; 34 3-D
   retlw   0x25  ; 35  3E
   retlw   0x24  ; 36  3F
   retlw   0x24  ; 37  40
   retlw   0x24  ; 38  41
   retlw   0x23  ; 39  42
   retlw   0x23  ; 40  43
   retlw   0x23  ; 41  44
   retlw   0x22  ; 42  45
   retlw   0x22  ; 43  46
```

```
   retlw   0x22   ;   47
                 44
   retlw   0x21   ;   48
                 45
   retlw   0x21   ;   49
                 46
   retlw   0x21   ;   4Th
                 47
   retlw   0x20   ;   4B
                 48
   retlw   0x20   ;   4C
                 49

aderL retlw  0xAA   ; LL display
aderH retlw  0xBB   ; HH display

;-------------------------------------------------------------------------
; Segment Data Table - Common Cathode Display
segtbl_r
        andlw 0x0F           ; Low nibble only
        addwf PCL,f          ; PC tip
        retlw  b'00111111'  ;  "0"  -|-|F|E|D|C|B|A
        retlw  b'00000110'  ;  "1"  -|-|-|-|-|C|B|-
        retlw  b'01011011'  ;  "2"  -|G|-|E|D|-|B|A
        retlw  b'01001111'  ;  "3"  -|G|-|-|D|C|B|A
        retlw  b'01100110'  ;  "4"  -|G|F|-|-|C|B|-
        retlw  b'01101101'  ;  "5"  -|G|F|-|D|C|-|A
        retlw  b'01111101'  ;  "6"  -|G|F|E|D|C|-|A
        retlw  b'00000111'  ;  "7"  -|-|-|-|-|C|B|A
        retlw  b'01111111'  ;  "8"  -|G|F|E|D|C|B|A
        retlw b'01101111' ; "9" -|G|F|-|D|C|B|At
        retlw  b'00111000'  ;  "L"  -|-|F|E|D|-|-|-  exit  for  Ah
        retlw b'01110110' ; "H" -|G|F|E|-|C|B|- exit to Bh

;-------------------------------------------------------------------------
; Voltage measurement
ADConv_r:
; Waiting for noise reduction
        DLY10US
; Start Conversion
        Bsf     ADCON0,GO
ADLP    BTFSC   ADCON0,NOT_DONE
         goto   ADLP
        retlw   0

;**********************************************************************
;                            THE END
        END
```