Knowledge of baseline

Flashing LED - Assembly

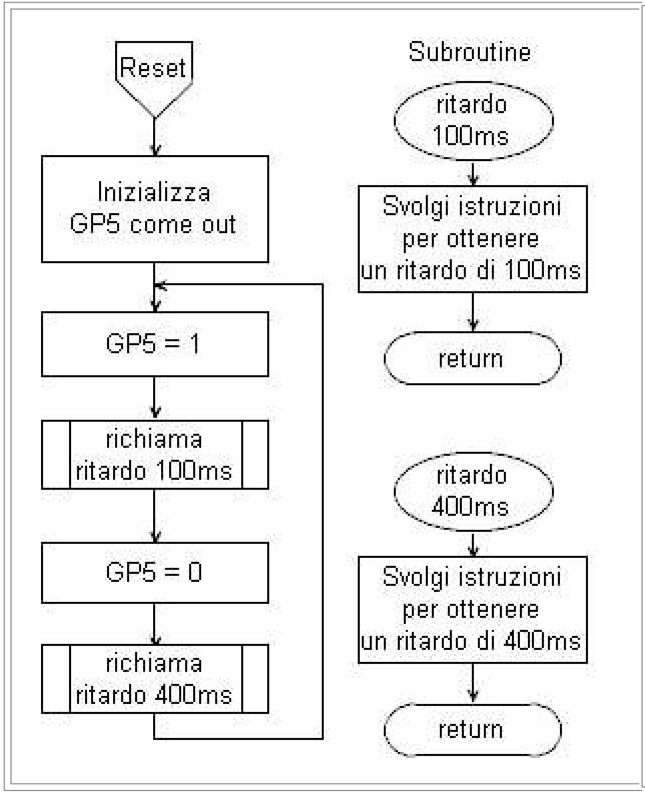
What we want to achieve

The purpose of the tutorial is the same as the previous one, which is to flash an LED through the microcontroller.

However, we want to introduce another element, which is to expand the possibilities of the tempo subroutine, modulating it through a parameter passed with the W register.

The demand for our flashing LED is to **reduce power consumption**.

This can be easily achieved by reducing the switch-on time compared to the switch-off time, i.e. by varying the duty cycle.
We could, for example, devote 25% of the time of a cycle to the "on" state and the remaining 75% to the "off" state: the flashing will still be clearly visible, while the average current will be reduced.
For a cycle time of 1/2 second, this means having a 100ms and a 400ms wait routine.
The flowchart clarifies the situation better.



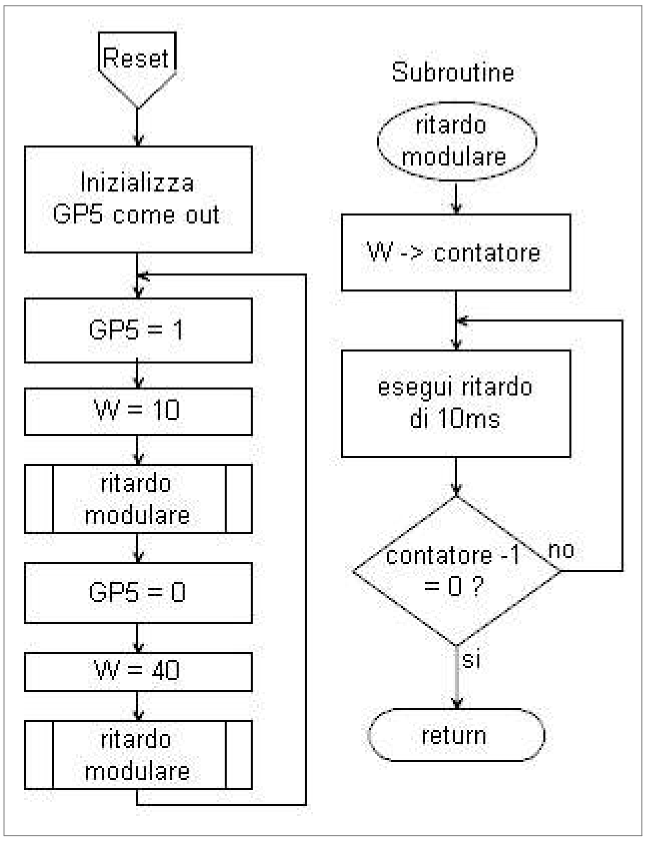
Let's simply introduce two time routines, calculated with the applet already seen.

If we write these two, we realize that they are practically identical in structure, differing essentially in the pre-charge value of the meters and their number.

We can observe that it is certainly possible to write waits for any time, but, if different times are needed, it would be necessary to rewrite the same structure many times with different values.

This is not very practical, it takes up space in program memory, and it is certainly not a generalizable or modular structure.

Is there a more efficient alternative than writing practically similar time routines? We can look at the problem from a slightly different point of view.



A 100ms wait is analogous to 10 10ms waits, and a 400ms wait is the same as 4 100ms waits.

We can then think of a different solution: **a "programmable" delay routine that performs a "basic" wait and is replicated as many times as necessary**.

This allows you to have only one time routine and have a constant amount of RAM used for counters.

We can take the "base" time of 10ms and make a loop that makes this delay; By running it as many times as necessary, we get the desired times.

An effective way to pass parameters to the routine is to use the W register: by indicating the number of repetitions in W, you get a variable delay between 10ms (W=1) and 2.55s (W=255).

Obviously, in other applications we can consider a different base time, for example 1ms or 100ms, in order to cover all the needs of the application.

It should be noted that we are not implementing a very precise structure here: even if the time routine for 10ms is precise in itself, the addition of the repetition instructions adds up to some loops that lengthen the execution. However, this is in the worst case 512 microseconds and therefore more than acceptable for this application.
By using other procedures for the generation of times, which we will see later, it will still be possible to obtain very precise "modular" timings.

On an occasion like this, we have an example of the difference between Assembly and C: in C there is a library of functions for waiting times, such as ***DelayMS*** or ***DelayUS*** or ***DelayCyc*** (or similar, depending on the version of C).
In the case of Assembly, it is also possible to have similar functions, but you have to create these libraries yourself, since the language does not contain them.

Once you've drawn your flowchart, it's simple to turn it into instructions. Also in this example the source is made available already fully written, in the 2Aa_519.asm file. As always, we comment on the various parts.

The basic routine for the 10ms @ 4MHz time is derived like the others from the **Delay Code Generator** already seen.

```
; Dele = 0.01 Seconds1
; Clock 4 MHz - 10ms = 10000 cycles
Delay10msW:                 ; 9993 cycles
  movlw    0xCE             ; Initialize Counters
  movwf    d1
  movlw    0x08
  movwf    d2
dlyw_0:                     ; Counting Loops-<-|
  decfsz   d1, f            ; d1=d1-1          |
   goto    $+2              ;->--|             |
  decfsz   d2, f            ;    | d2=d2-1     |
   goto    dlyw_0           ;-<>-|---->>---->> ---|
; End of Count 9993 Cycles - Now Sum the 7 Missing
  goto     $+1              ;->-|   3 cycles
  nop                       ;-<-|
  return                    ; 4 cycles with the call
```

The procedure requires two counters, **d1** and **d2**.

We still need to put it in the structure that uses the W register as an indicator of the number of times it needs to be repeated.
We need an additional RAM location for counting the number of repetitions (**d3**) in which the W content is initially copied and which is decremented to zero, repeating the 10 ms cycle each time.

```
; W-modulated delay
; Performs W cycles of 10 ms each
Delay10msW
        movwf    d3          ; copy W to counter
                             ; (+ 1 extra cycle)
; ciclo da 10ms
dlyw_1  movlw    0xCE        ; 9993 cycles
        movwf    d1
        movlw    0x08
        movwf    d2
dlyw_0  decfsz   d1, f
         goto    $+2
        decfsz   d2, f
         goto    dlyw_0
; end cycle 10 ms            ; 3 cycles
        goto     $+1
        nop
        decfsz   d3          ; Another loop?
         goto    dlyw_1      ; Yes (3 + 9996 = 9999)
        retlw    0           ; No (+5 cycles with
```

We observe that in fact the total time obtained is (N x 9.999ms) + 6us, so the precision in the repetitions is not absolute, but more than adequate to the application and that we can consider equal to 10ms per repetition. The +6us are due to the loops required by the **`call`** and **`retlw`** of the subroutine, plus the loop to copy the contents of **W** to **d3**.

The maximum achievable delay time will be (256 x 9,999ms) + 6us = 2559,750ms.

If desired, of course, it is possible to create more precise, but more complex loops; For now, no more is needed. Let's keep the macros already seen for the LED and modify the mainloop:

```
mainloop:                  ;<<-----<<<-------<<----|
; LED ON                   ;                       |
  LED_ON                   ;                       |
                           ;                       |
; Wait 100 ms                                      |
  movlw    10              ; 10 reps call          |
           Delay10msW      ;                       |
                           ;                       |
; turns off LEDs                        Main Loop
  LED_OFF                  ;                       |
                           ;                       |
; Wait 400 ms                                      |
  movlw    40              ; 40 Repetitions        |
  Call     Delay10msW      ;                       |
                           ;                       |
; Loop                                             |
  Goto     Mainloop      ; >>----->>>-------->>---|
```

We can, however, make some observations and variations.

But first we need to say something about the internal structure of PCIs.

Modular Codes

We can see that the need to have a subroutine that achieves a certain delay time is a fact that we find repeated in the previous two exercises. Each time we rewrote the sequence of instructions that allows the delay. Is this necessary?
Not really. You can use two different ways that allow you to write the routine once and for all and use it in any program without rewriting it:

- use the #include Directive
- Use modular programming

The first way is simple: we have seen how, through the #include directive, it is possible to "include" in the source a stretch of code that is physically outside of it. In the case of the line:

```
#include <p12F509.inc>
```

The directive causes the contents of the indicated file to be considered by the compiler as part of the source. The reason for this situation is very simple: for every source we write for that given processor, its ***.inc*** file will be indispensable. Obviously, one could rewrite each time the set of equivalences necessary for symbolic compilation, but it is

Obviously, since this can be treated as a separate element, it will be enough to have written it only once in order to be able to arrange for its inclusion when you wish.

A similar situation occurs in many other situations, for example with the procedures for generating delays that we have used: how many programs require the same functions?
Why, then, rewrite them each time and not adopt the same solution provided by the directive **#include** ? All you have to do is place the subroutine in a folder and call it up when you need it.

The directive places the text that the label object represents at the point in the source where the relevant line is placed. Then we can modify the source like this:

```
mainloop:                ;<<--<<<--|
; LED ON                 ;         |
  LED_ON                 ;         |
                         ;         |
; Wait 100ms                       |
  movlw   .10            ;         |
  call     Dly10ms       ;         |
                         ;         |
; LED OFF                         loop
  LED_OFF                ;         |
                         ;         |
; Wait 400ms                       |
  movlw   .40            ;         |
  call     DlyW10ms      ;         |
                         ;         |
; loop                   ;         |
  goto     mainloop      ;>>-->>---|

;####################################################################
;====================================================================
;                               SUBROUTINES
;
 #include C:\PIC\LIBRARY\Delay\Baseline\DlyW10ms.asm
```

where the subroutine is located in a folder `C:\PIC\LIBRARY\Delay\Baseline\` , which can collect algorithms to generate delays, constituting a small library of functions ready to be invoked when needed. The `DlyW10ms` is part of this "library".

A library is a collection of general-purpose strokes of code that can be retrieved by other programs without the need to rewrite them each time.

Of course we can put it in the folder we prefer, but in any case we have to specify the full path for the directive to fetch it, modifying the source text.

In the source code you will find the file ***Delay10msW.asm***.

You can put it in a folder `C:\PIC\LIBRARY\Delay\Baseline\` . In this case, there is no need to change the source.

However, if you place it in another folder, you will need to replace the full path of the `#include line`. For example, if the subroutine is located in **`C:\PIC\vaseline\`** the directive would become:

```
#include C:\PIC\Baseline\DlyW10ms.asm
```

and so for any other change in the location of the file to be included. This precaution is of course necessary to allow the compiler to fetch the file required by the inclusion.

You might notice that the inclusion of the **`processorname.inc`** file does not require a specific path; this is because the file is located in the Macro Assembler folder, whose path is already automatically considered by MPLAB.

In a later tutorial, we will see how to write **relocatable codes**, which is the heart of professional programming.

Other versions

We can have different PICs perform the same operation and notice that they are always the same actions.
Sources are available for PIC12F508/509, 10F200/202, 16F505/526.

Extensions

We can propose the same variations on the theme:

1. vary the LED flashing time
2. flash the LED on GP2
3. turn on an LED on GP0 and flash an LED on GP2

The solutions are those already presented in the 2A exercise, introducing the modular aspect we have just seen in the source.

Conclusions

We can conclude that modularity is a fundamental element not only of Assembly programming, but of any other language. Libraries are the key to the success or failure of C or BASIC compilers, since they provide the user with ready-to-use functions, without requiring rewriting each time.

Similarly for Assembly, the use of modules and libraries makes it possible to significantly lighten the work of the programmer.

Unfortunately, in the often bad examples on the WEB, it is not a common argument, also due to the simplicity of most of the sources.
However, the fact that it is not an indispensable feature for the implementation of small programs does not make it superfluous; On the contrary, for programmes of a certain size, if we did not have recourse to this and other more advanced working methods, we would not be able to carry them out. And even in the case of limited sources, the use of a minimum modularity is always convenient.

Delay10msW.asm

```
;*********************************************************************
;--------------------------------------------------------------------
; Delay10msW.asm
;
;    Title          : 10ms delay subroutine @ 4MHz clock.
;                     Performs 9.995ms N loops; N passed with W.
;                     Total time is (N x 9.995ms)+ 6us
;    PIC            :Baseline
;    Support        : MPASM
;    Version        : 1.0
;    Date           : 01-05-2013
;    Hardware ref.  :
;    Author         :Afg
;
;*********************************************************************
; ####################################################################
;                          RAM
; You need 3 counters
; D1
; D2
; D3

; ####################################################################

Delay10msW
       Banksel D3       ; 2 extra movwf
       cycles  D3
DLWLP0 movlw   0xCE     ; 9993 MovWF
       Cycles  D1
       movlw   0x08
       movwf   D2
DLWLP1 decfsz D1,F
        Goto   $+2
       decfsz d2,f
        Goto   dlwlp1
       decfsz d3,f      ; 2 goto
        cycles dlwlp0
       retlw   0        ; 4 extra cycles including call

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

12F519 - 2Aw_519.asm

```
;*********************************************************************
;---------------------------------------------------------------------
;
;    Title          : Assembly & C Course - Tutorial 2As_519
;                      An LED connected to GP5 with duty flashes
;                      25%
;
;    PIC            : 12F519
;    Support        : MPASM
;    Version        : 1.0
;    Date           : 01-05-2013
;    Hardware ref.  :
;    Author         :Afg
;
;---------------------------------------------------------------------
;
; Pin use :
;   ________________
;       12F519 @ 8 pin
;
;                   |¯¯\/¯¯|
;             Vdd  -|1    8|- Vss
;             GP5  -|2    7|- GP0
;             GP4  -|3    6|- GP1
;        GP3/MCLR  -|4    5|- GP2
;                   |______|
;
;    Vdd                 1: ++
;    GP5/OSC1/CLKIN      2: Out LED to Vss
;    GP4/OSC2            3:
;    GP3/! MCLR/VPP      4:
;    GP2/T0CKI           5:
;    GP1/ICSPCLK         6:
;    GP0/ICSPDAT         7:
;    Vss                 8: --
;
;*********************************************************************
;            DEFINITION OF PORT USE
;
; GPIO map
;| 5 | 4 | 3 | 2 | 1 | 0 |
;|-----|-----|-----|-----|-----|-----|
;| LED |     | in |      |     |     |
;
;#define    GPIO,GP0    ;
;#define    GPIO,GP1    ;
;#define    GPIO,GP2    ;
;#define    GPIO,GP3    ; Input only
;#define    GPIO,P4     ;
#define     GPIO LED,GP5    ; LED between pin
                            and Vss
;
; ####################################################################

        LIST      p=12F519         ; Processor Definition
        #include <p12F519.inc>

        radix     DEC

; ######################################################################
;                          CONFIGURATION
;
; Internal oscillator, no WDT, no CP, pin4=MCLR;
 __config _IntRC_OSC & _IOSCFS_4MHz & _WDTE_OFF & _CP_OFF & _CPDF_OFF &
_MCLRE_ON


; ######################################################################
;                        RAM
; general purpose RAM
        CBLOCK 0x07          ; start of RAM area
      d1,d2,d3               ; Delay counters
        ENDC

; ######################################################################
;                       LOCAL MACROS
;
; Controls for the
LED_ON LED    MACRO
        Bsf
                END
          M LEDs
LED_OFF   MACRO
        Bcf
                END
          M LEDs

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

; MOWF Internal Oscillator
        Calibration OSCCAL

; ######################################################################
;                         MAIN PROGRAM
;
Main
; Reset Initializations
     CLRF    GPIO        ; GPIO preset latch to 0

; TRISGPIO     --011111 GP5 out
     movlw   b'11011111'
     Tris    GPIO        ; To the Management Register

;===================================================================
Mainloop:                ; <<--<<<--|
; Lights up LEDs         ;          |
     LED_ON              ;          |
                         ;          |
; Wait 100ms                        |
movlw   .10          ; 10 x 10ms = 100ms
```

```
     Call    Delay10msW ;          |
                         ;         |
; turns off LEDs                 Loop
     LED_OFF             ;         |
                         ;         |
; Wait 400ms                       |
     movlw   .40         ; 40 x 10ms = 400ms
     call    Delay10msW ;          |
                         ;         |
; Loop                   ;         |
    Goto    Mainloop    ; >>-->>---|


;**********************************************************************
;                           SUBROUTINE
  #include C:\PIC\LIBRARY\Delay\Baseline\Delay10msW.asm

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

12F508/509 - 2Aw_5089.asm

```
;***********************************************************************
;-----------------------------------------------------------------------
;
;    Title          : Assembly & C Course - Tutorial 2As_5089
;                     An LED connected to GP5 flashes at the
;                     1 second cadence.
;                     Subroutine.
;    PIC            : 12F508/9
;    Support        : MPASM
;    Version        : V.519-1.0
;    Date           : 01-05-2013
;    Hardware ref.  :
;    Author         :Afg
;
;-----------------------------------------------------------------------
;
; Pin use :
;   _______________
;       12F508/9 @ 8 pin
;
;                      |¯¯\/¯¯|
;             Vdd -|1    8|- Vss
;             GP5 -|2    7|- GP0
;             GP4 -|3    6|- GP1
;        GP3/MCLR -|4    5|- GP2
;                      |______|
;
;    Vdd                 1: ++
;    GP5/OSC1/CLKIN      2: Out LED to Vss
;    GP4/OSC2            3:
;    GP3/! MCLR/VPP      4:
;    GP2/T0CKI           5:
;    GP1/ICSPCLK         6:
;    GP0/ICSPDAT         7:
;    Vss                 8: --
;
;***********************************************************************
;            DEFINITION OF PORT USE
;
; GPIO map
;| 5 | 4 | 3 | 2 | 1 | 0 |
;|-----|-----|-----|-----|-----|-----|
;| LED |     | in |     |     |     |
;
;#define    GPIO,GP0    ;
;#define    GPIO,GP1    ;
;#define    GPIO,GP2    ;
;#define    GPIO,GP3    ; Input only
;#define    GPIO,P4     ;
#define     GPIO LED,GP5    ; LED between pin
                              and Vss
;
; ######################################################################
; Choice of #ifdef
 processor_12F509
```

```
        LIST      p=12F509         ; Processor Definition
        #include <p12F509.inc>
 #endif

 #ifdef____12F508
       LIST      p=12F508         ; Processor definition
        #include <p12F508.inc>
 #endif

        Radix     DEC

; ######################################################################
;                        CONFIGURATION
;
 Internal oscillator, no WDT, no CP, pin4=GP3
 __config _IntRC_OSC & _WDT_OFF & _CP_OFF & _MCLRE_ON


; ######################################################################
;                        RAM
;
; general purpose RAM
        CBLOCK 0x07          ; start of RAM area
      d1,d2,d3               ; Delay counters
        ENDC


; ######################################################################
;                        LOCAL MACROS
; Controls for the
LED_ON LED     MACRO
        Bsf
                END
          M LEDs
LED_OFF   MACRO
        Bcf
                END
          M LEDs

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

 ; MOWF Internal Oscillator
        Calibration OSCCAL

; ######################################################################
;                        MAIN PROGRAM
;
MAIN:
; Reset Initializations
     CLRF    GPIO        ; GPIO preset latch to 0

; TRISGPIO     --011111   GP5 out
     movlw   b'11011111' ; PORT tris direction
```

```
mask    GPIO         ; To the Management
Register
```

```
;===================================================================
Mainloop:                 ; <<--<<<--|
; Lights up LEDs          ;          |
     LED_ON               ;          |
                          ;          |
; Wait 100ms                         |
     movlw   .10          ; 10 x 10ms = 100ms
     call    Delay10msW ;            |
                          ;          |
; turns off LEDs                   Loop
     LED_OFF              ;          |
                          ;          |
; Wait 400ms                         |
     movlw   .40          ; 40 x 10ms = 400ms
     call    Delay10msW ;            |
                          ;          |
; Loop                    ;          |
    Goto    Mainloop    ; >>-->>---|


;********************************************************************
;                          SUBROUTINE
  #include C:\PIC\LIBRARY\Delay\Baseline\Delay10msW.asm

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

10F200/202 - 2As_20x.asm

```
;*********************************************************************
;---------------------------------------------------------------------
;
;    Title          : Assembly & C Course - Tutorial 1As_20x
;                      An LED connected to GP0 flashes at the
;                      1 second cadence.

;    PIC            : 10F200/2
;    Support        : MPASM
;    Version        : 1.0
;    Date           : 01-05-2013
;    Hardware ref.  :
;    Author         :Afg
;
;
; Pin use :
;   _______________
;     10F200/202 @ 8 pin DIP       10F200/202 @ 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:     Nc
;    Vdd                 2:   5: ++
;    GP2/T0CKI/FOSC4     3:   4:
;    GP1/ICSPCLK         4:   3:
;    GP0/ICSPDAT         5:   1: Out LED at Vss
;    NC                  6:      Nc
;    Vss                 7:   2: --
;    GP3/MCLR/VPP        8:   6:
;
;*********************************************************************
;            DEFINITION OF PORT USE
; GPIO map
; | 3 | 2 | 1 | 0 |
; |-----|-----|-----|-----|
; | in |      |     | LED |
;
#define    GPIO LED,GP0    ; LED between pin and Vss
;#define    GPIO,GP1    ;
;#define    GPIO,GP2    ;
;#define    GPIO,GP3    ; Input only
;
; ####################################################################
 #ifdef___10F200
       LIST      p=10F200      ; Processor Definition
       #include <p10F200.inc>
 #endif
 #ifdef___10F202
```

```
        LIST      p=10F202      ; Processor Definition
        #include <p10F202.inc>
 #endif

        Radix     DEC           ; Decimal Numbers Defaults

; #####################################################################
;                            CONFIGURATION
;
; No WDT, no CP, pin4=GP3;
  __config _CP_OFF & _MCLRE_OFF & _WDT_OFF

; #####################################################################
;                            LOCAL MACROS
;
; Controls for the
LED_ON LED    MACRO
        Bsf
                END
          M LEDs
LED_OFF   MACRO
        Bcf
                END
          M LEDs

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

 ; MOWF Internal Oscillator
        Calibration OSCCAL

; #####################################################################
;                           MAIN PROGRAM
;
MAIN:
; Reset Initializations
        CLRF    GPIO            ; GPIO preset latch to 0

; TRISGPIO        --111110     GP0 out
        movlw   b'11111110'
        Tris    GPIO            ; To the Management
                                Register

;===================================================================
Mainloop:                 ; <<--<<<--|
; Lights up LEDs          ;          |
     LED_ON               ;          |
                          ;          |
; Wait 100ms                         |
     movlw   .10          ; 10 x 10ms = 100ms
     call    Delay10msW ;            |
                          ;          |
; turns off LEDs                Loop
     LED_OFF              ;          |
                                     ;           |
; Wait 400ms                                     |
     movlw   .40          ; 40 x 10ms = 400ms
     call    Delay10msW ;           |
                         ;          |
; Loop                   ;          |
    Goto    Mainloop    ; >>-->>---|


;*********************************************************************
;                          SUBROUTINE
  #include C:\PIC\LIBRARY\Delay\Baseline\Delay10msW.asm

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

16F526/505 - 2As_526.asm

```
;*******************************************************************
;-------------------------------------------------------------------
;
;    Title          : Assembly & C Course - Tutorial 2As_20x
;                      An LED connected to GP0 flashes at the
;                      1 second cadence.
;                      Subroutine.
;    PIC            : 16F526-16F505
;    Support        : MPASM
;    Version        : 1.0
;    Date           : 01-05-2013
;    Hardware ref.  :
;    Author         :Afg
;
;-------------------------------------------------------------------
;
;  Pin use :
;  ________________
;       16F505 - 16F526 @ 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 LED to Vss
;    RB4/OSC2/CLKOUT          3:
;    RB3/! MCLR/VPP           4:
;    RC5/T0CKI                5:
;    RC4/[C2OUT]              6:
;    RC3                      7:
;    RC2/[Cvref]              8:
;    RC1/[C2IN-]              9:
;    RC0/[C2IN+]             10:
;    RB2/[C1OUT/AN2]         11:
;    RB1/[C1IN-/AN1/]ICSPC   12:
;    RB0/[C1IN+/AN0/]ICSPD   13:
;    Vss                     14: --
;
;    [ ] only 16F526
;
;***************************************************************
;===============================================================
;           DEFINITION OF PORT USE
;
;P ORTC not used
;
```

```
; PRTB map
; | 5 | 4 | 3 | 2 | 1 | 0 |
; |-----|-----|-----|-----|-----|-----|
; | LED |     |     |     |     |     |
;
#define   LED  PORTB,RB5    ; LED between pin
               and Vss
;#define       PORTB,RB3
;#define       PORTB,RB2
;#define       PORTB,RB1
;#define       PORTB,RB0

; ###########################################################################
;                            PROCESSOR SELECTION
 #ifdef___16F526
      LIST p=16F526
      #include <p16F526.inc>
  #endif
  #ifdef___16F505
      LIST p=16F505
      #include <p16F505.inc>
 #endif

; ###########################################################################
;                             CONFIGURATION
;
 #ifdef___16F526
; Internal Oscillator, 4MHz, No WDT, No CP, No MCLR
 __config _IntRC_OSC & _IOSCFS_4MHz & _WDTE_OFF & _CP_OFF & _CPDF_OFF
& MCLRE_OFF
 #endif

 #ifdef___16F505
; Internal Oscillator, 4MHz, No WDT, No CP, No MCLR
 __config _IntRC_OSC_RB4 & _WDT_OFF & _CP_OFF & _MCLRE_OFF
 #endif

; ###########################################################################
;=========================================================================
; ###########################################################################
;                          RAM
;
; general purpose RAM
        CBLOCK 0x07          ; start of RAM area
      d1,d2,d3               ; ENDC Delay Counters

; ###########################################################################
;                        LOCAL MACROS
;
; Controls for the
LED_ON LED    MACRO
        Bsf
                END
          M LEDs
LED_OFF   MACRO
        Bcf
                END
          M LEDs
```

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

 ; MOWF Internal Oscillator
        Calibration OSCCAL

; ####################################################################
;                          MAIN PROGRAM
;
MAIN:
; Reset Initializations
     CLRF    PORTB       ; Preset Port Latch to 0

; TRISB        --011111 RB5 out
     movlw   b'11011111'
     Tris    PORTB       ; To the Management Register

;==================================================================
Mainloop:                ; <<--<<<--|
; Lights up LEDs         ;          |
     LED_ON              ;          |
                         ;          |
; Wait 100ms                        |
     movlw   .10         ; 10 x 10ms = 100ms
     call    Delay10msW ;           |
                         ;          |
; turns off LEDs                  Loop
     LED_OFF             ;          |
                         ;          |
; Wait 400ms                        |
     movlw   .40         ; 40 x 10ms = 400ms
     call    Delay10msW ;           |
                         ;          |
; Loop                   ;          |
    Goto    Mainloop    ; >>-->>---|


;*******************************************************************
;                            SUBROUTINE
  #include C:\PIC\LIBRARY\Delay\Baseline\Delay10msW.asm

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