```
//------------------------------------------------------------------------------
// TP0 28/11/2019
// dsPIC 33FJ128MC802
// L. BAGHLI
//------------------------------------------------------------------------------
// FBS
#pragma config BWRP = WRPROTECT_OFF     // Boot Segment Write Protect (Boot
Segment may be written)
#pragma config BSS = NO_FLASH           // Boot Segment Program Flash Code
Protection (No Boot program Flash segment)
#pragma config RBS = NO_RAM             // Boot Segment RAM Protection (No Boot
RAM)

// FSS
#pragma config SWRP = WRPROTECT_OFF     // Secure Segment Program Write Protect
(Secure segment may be written)
#pragma config SSS = NO_FLASH           // Secure Segment Program Flash Code
Protection (No Secure Segment)
#pragma config RSS = NO_RAM             // Secure Segment Data RAM Protection (No
Secure RAM)

// FGS
#pragma config GWRP = OFF               // General Code Segment Write Protect
(User program memory is not write-protected)
#pragma config GSS = OFF                // General Segment Code Protection (User
program memory is not code-protected)

// FOSCSEL
#pragma config FNOSC = PRIPLL           // Oscillator Mode (Primary Oscillator
(XT, HS, EC) w/ PLL)
#pragma config IESO = OFF               // Internal External Switch Over Mode
(Start-up device with user-selected oscillator source)

// FOSC
#pragma config POSCMD = XT              // Primary Oscillator Source (XT
Oscillator Mode)
#pragma config OSCIOFNC = OFF           // OSC2 Pin Function (OSC2 pin has clock
out function)
#pragma config IOL1WAY = OFF            // Peripheral Pin Select Configuration
(Allow Multiple Re-configurations)
#pragma config FCKSM = CSDCMD           // Clock Switching and Monitor (Both Clock
Switching and Fail-Safe Clock Monitor are disabled)

// FWDT
#pragma config WDTPOST = PS32768        // Watchdog Timer Postscaler (1:32,768)
#pragma config WDTPRE = PR128           // WDT Prescaler (1:128)
#pragma config WINDIS = OFF             // Watchdog Timer Window (Watchdog Timer
in Non-Window mode)
#pragma config FWDTEN = OFF             // Watchdog Timer Enable (Watchdog timer
enabled/disabled by user software)

// FPOR
#pragma config FPWRT = PWR128           // POR Timer Value (128ms)
#pragma config ALTI2C = OFF             // Alternate I2C  pins (I2C mapped to
SDA1/SCL1 pins)
#pragma config LPOL = ON                // Motor Control PWM Low Side Polarity bit
(PWM module low side output pins have active-high output polarity)
#pragma config HPOL = OFF               // Motor Control PWM High Side Polarity
bit (PWM module high side output pins have active-low output polarity)
```

```
#pragma config PWMPIN = ON             // Motor Control PWM Module Pin Mode bit
(PWM module pins controlled by PORT register at device Reset)

// FICD
#pragma config ICS = PGD1              // Comm Channel Select (Communicate on
PGC1/EMUC1 and PGD1/EMUD1)
#pragma config JTAGEN = OFF            // JTAG Port Enable (JTAG is Disabled)

// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.

#include <xc.h>
// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.

//Program Specific Constants
#define FCY 20000000                   //Instruction cycle rate (Osc x PLL / 4) =
20 MIPS
#define FPWM 16000                 //PWM freq
#define HalfDUTY   624    // set the pwm half duty = PTPER = 624UL=FCY/FPWM/2-1;
#define FullDUTY HalfDUTY*2          //  car PDCx a une double resolution % PTPER
#define MILLISEC FCY/20000       // 1 mSec delay constant

#define RunningLED  _LATB15          // Output
#define InfoLED     _LATB13          // Output
#define KeyEnter    _RB2            // Input         // BP pour Start
#define IRsignal    _RA4            // Input         // IR TS1838 Sense sur CN0
(change notification)
#define EnableL6234          _LATB11
// AN0 RA0 analog Potentiometer
// AN1 RA1 analog Potentiometer
// AN5 RB3 UART1_TX    Extgernal AD input
// RP7 QEPI
// RP6 QEPA
// RP5 QEPB
// RP3 UART1_TX
// RP4 UART1_RX
// RP8/SCL1 SCL
// RP9/SDA1 SDA

// prototypes des fonctions
void setup_ports();
void DelayNmSec(unsigned int N);
//-----------------------------------------------------------------------------
// Setup ports
//-----------------------------------------------------------------------------
void setup_ports()
{
   RCONbits.SWDTEN = 0; // Disable Watch Dog Timer
    // setup clock FOSC
    _PLLPRE = 0;
    _PLLPOST = 1;
    _PLLDIV = 30;
    // When clock switch occurs switch to Primary Osc (HS, XT, EC)
    __builtin_write_OSCCONH(0x03); // Initiate Clock Switch to Primary Oscillator
with PLL (NOSC=0b011)
    __builtin_write_OSCCONL(0x01); // Start clock switching
#ifndef __SIMUL_ISIS
    while (OSCCONbits.COSC != 0b011); // Wait for Clock switch to occur
```

```
    while (OSCCONbits.LOCK != 1); // Wait for PLL to lock, only if PLL is needed
#endif
   AD1PCFGL = 0xFFFF;          // all PORT A/B = Digital(1) NoADC= analog(0)
   PORTA=0;   //Initialize LED pin data to off state
   PORTB=0;   //Initialize LED pin data to off state
   // Now set pin direction registers
   TRISA = 0xFFFF;       // NC
   TRISB = 0x2FF7;       // RB in NC, RB15,13,12 Out LEDs, UART RX et TX config
par RP4/RP3
                        // mais j'ai besoin dun RB3 Out pour UART BREAK
0010|1111|1111|0111
  __builtin_write_OSCCONL(OSCCON & ~0x40); //clear the bit 6 of OSCCONL (IOLOCK)
to unlock pin re-map

  __builtin_write_OSCCONL(OSCCON | 0x40); //set the bit 6 of OSCCONL (IOLOCK) to
lock pin re-map
// debug PWM
}
//------------------------------------------------------------------------------
//Main routine
int main()
{
   setup_ports();
   RunningLED=1;                // LED run = On
   RunningLED=0;                // LED run = Off
   while(1)
      {
        RunningLED=1;                   // LED run = On
        RunningLED=0;                   // LED run = Off
      }  // end of while (1)
}  // end of main
//------------------------------------------------------------------------------
//-----------------------------------------------------------------------
// This is a generic 1ms delay routine to give a 1mS to 65.5 Seconds delay
// For N = 1 the delay is 1 mS, for N = 65535 the delay is 65,535 mS.
// Note that FCY is used in the computation.  Please make the necessary
// Changes(PLLx4 or PLLx8 etc) to compute the right FCY as in the define
// statement above.
//-----------------------------------------------------------------------
void DelayNmSec(unsigned int N)
{
unsigned int j;
while(N--)
   for(j=0;j < MILLISEC;j++);
}
//-----------------------------------------------------------------------
```