INFO-0064 Embedded systems

Exercise session 7
PIC programming

Kevin Hogan
kevin.hogan@ulg.ac.be
Planning for today

• How to program your PIC?
• Simple C program
How to program a PIC?

Figure from http://www.mikroe.com
How to program a PIC?

**Programmer** ➔ ICD 3 (& PICkit 3)

Connection between the ICD3, your PC and your PIC?

➔ **Datasheet**!

The PIC has to be correctly powered up!
How to program a PIC?

Compiler:

→ MPASM (assembly)
→ XC8 (C programming)

Is that enough?

Yes but it is simpler to use with: MPLAB X IDE
(On Windows, Mac & Linux)
How to program a PIC?

What to do step-by-step:

1. Download and install MPLAB X (MPASM comes with it)
2. Download and install XC8
3. Launch MPLAB and create a new project (specify your PIC, your compiler and your programmer)
4. Right click on Source Files folder >> New >> C Main file
5. Write your program
6. Connect the ICD3 and your PIC >> power up your PIC >> plug the ICD3 to your PC (you can also use the ICD3 to power up your PIC)
7. Click on “Make and program device”
Simple C program

Program structure?

⇒ XC8 Getting started & XC8 User Guide (found in xc8\v1.35\docs)

Foundation code

```c
#include <xc.h>

void main(void){
    // do some stuff here
    while(1){
        // or here
    }
}
```
Simple C program

```c
//header files
#include <xc.h>

//configuration bits settings
...

//interrupt vector
...

void main (void){
    //initialization code
    
    while(1){
        ...
    }
}

Declares all device specific types and variables, so accessing some SFRs consists in read/write some variables

Examples:

TRISB = 0;
LATBbits.LATB3 = 1;

You can check the names of those variables in the file “pic18f4620.h” or in MPLAB X: Window >> PIC Memory Views >> SFRs
Simple C program

Configuration bits?

→ PIC18F4620 datasheet or in the file “xc8/v1.35/docs/pic18_chipinfo.html”

→ MPLAB X: Window >> PIC memory Views >> Configurations bits

Example:

```c
#pragma config OSC = HS            // external clock
#pragma config MCLRE = ON          // on active mclear
#pragma config PBADEN = OFF        // Portb analog input OFF
#pragma config LVP = OFF           // Low voltage icps off
#pragma config WDT = OFF,DEBUG=OFF // watchdog off, debug off
```
Simple C program

Interrupt vector?
➔ XC8 user’s guide

```c
void interrupt MyIntVec(void) {
    // check flags to find the source of the interrupt
}
```

With priority

```c
void interrupt low_priority MyLowIntVec(void) {
    // check flags to find the source of the interrupt
}
void interrupt high_priority MyHighIntVec(void) {
    // check flags to find the source of the interrupt
}
```
Simple C program

How to use the peripheral hardwares (timers, ADC, ...) ?

➡️ Write to the corresponding SFRs
➡️ Use peripheral libraries

Example: timers

```c
#include <timers.h>

OpenTimer0(TIMER_INT_ON & T0_16BIT & T0_SOURCE_INT & T0_PS_1_64);
```

- Interrupt enabled
- 16 bits mode
- Use internal clock
- Prescaler set to 1:64
Simple C program: Blinky

Let’s put the pieces together!

```c
//header files
#include <xc.h>
#include <timers.h>

//configuration bits settings
#pragma config OSC = HS   // external clock
#pragma config MCLRE = ON // on active mclear
#pragma config PBADEN = OFF // Portb analog input OFF
#pragma config LVP = OFF   // Low voltage icps off
#pragma config WDT = OFF,DEBUG=OFF // watchdog off, debug off
```
void main(void){
    TRISB = 0; // PORTB = output
    LATB = 0; // PORTB = 0V

    OpenTimer0(TIMER_INT_ON & T0_16BIT & T0_SOURCE_INT & T0_PS_1_64);

    GIEH = 1; // Enable global Interrupt

    while(1){
        // wait for interrupts
    }
}
Simple C program

```c
//interrupt vector
void interrupt MyIntVec(void) {
    if (TMR0IE && TMR0IF) {
        LATB3 = !LATB3;
        return;
    }
}
```

There is a mistake! Can you spot it?
Simple C program

```c
//interrupt vector
void interrupt MyIntVec(void) {
    if (TMR0IE && TMR0IF) {
        TMR0IF = 0;
        LATB3 = !LATB3;
        return;
    }
}
```
Documentation

• PIC18F4620 Datasheet
• ICD 3 User’s guide

• xc8/v1.35/docs