PIC12Fxxx
Programming PIC in C

Vojtěch Krmíček
vojtec@ics.muni.cz

C Introduction

- We can use high level language C to avoid writing a code in assembly code
- C compiler converts C code to the assembler code
- We can use debugger tools in MPLAB for debugging – stepping, watch, memory listing, disassemlby listing, …

Logic Operations in C

- Operations with bits:
  - & - AND
  - | - OR
  - ^ - XOR
  - << - shift to left
  - >> - shift to right
  - ~ - negation
- Operations with bytes:
  - && - AND
  - || - OR

Expressing Numbers

- Number Definitions

```
unsigned char x;      // 8 bits on PIC

x=255;                // give decimal number
x=0xFF;               // give hex number, x=255
x=0b11111111;         // give binary number, x=255
```

- Whats wrong with the following code?

```
unsigned char i;
for (i=7;i>=0;i--) { };     //will loop forever
```

Available Datatypes on a PIC

- unsigned char a; //8 bits, 0 to 255
- signed char b; //8 bits, -128 to 127
- unsigned int c; //16 bits, 0 to 65535
- signed int d; //16 bits, -32768 to 32767
- long e; //32 bits, -2147483648 to 2147483647
- float f; //24 or 32 bits, depending on options under ‘edit project’
- bit g; // 1bit, 0 or 1

Switching Bit in Variable

```
unsigned char  x = 0b10010101    // var. definition
```

- Setting bit number *bitno*:

```
x = x | (1 << bitno);
```

- Clearing bit number *bitno*:

```
x = x & ~(1 << bitno);
```

- We can use macros:

```
#define bit_set(var,bitno) ((var) |= 1 << (bitno))
#define bit_clr(var,bitno) ((var) &= ~(1 << (bitno)))
```

Handling Interrputs

- compiler automatically manages saving of context (STATUS and W registers)

```
void interrupt tc_int(void)
{
  if (T0IE && T0IF) {
     T0IF=0;
     ++tick_count;
  }
}
```