Microprocessor Interfacing & Programming

LECTURE 5&6

Using Instructions with the Default Access Bank

- PIC also allows direct access to other locations in the file register for ALU and other operations.
- The access bank of the file register is the default bank upon powering up the PIC18.
- **MOVWF Instruction:**

This instruction tells the Processor to move/copy the source register WREG to a destination in the file register.

- The location in the file register can be one of the SFRs or a location in the GPRs region.

- MOVWF PORTA ------- will move the contents of WREG into the SFR register called PORTA.

- **EXAMPLE:**

```
MOVLW 55H;        ------------  WREG = 55H
MOVWF PORTB;      ------------  copy WREG to Port B
MOVWF PORTC;      ------------  copy WREG to Port C
MOVWF PORTD;      ------------  copy WREG to Port D
```

- SFRs are mapped in the same RAM address space, but they are not actual RAM cells.
- Writing to an SFR writes to hardware.

Example with GPRs (RAM)

MOVLW 99H; ------------------ WREG = 99H

MOVWF 0H; ----------------- move (copy) WREG contents to location 0h

MOVWF 1H;

MOVWF 2H;

MOVWF 3H;

MOVWF 4H;

Address	Data
000	99
001	99
002	99
003	99
004	99

NOTE: **We cannot move literal (immediate) values directly into the general purpose RAM locations or in SFRs in the PIC18. They must be moved there via WREG.**

- So, to load a literal directly into an SFR or GPR, we typically do it in two instructions.

ADDWF Instruction

- Among the group of logic and arithmetic instructions that involve both the WREG and a location in the file register, ADDWF is one of them.
- This instruction adds together the contents of WREG and a file register location.
- This file register location can be one of the SFRs or GPRs.
- The destination for the result can be the WREG or the file register.

FORMAT: ADDWF fileReg, D

ADDWF fileReg, D

where, fileReg is the file register location and D indicates the destination bit.

- If D=0, it means that the destination is WREG. If D=1, then the result will be placed in the file register.

EXAMPLE 1:

```
MOVLW 22H;
MOVWF 5H;
MOVWF 6H;
MOVWF 7H;
ADDWF 5H, 0;
ADDWF 6H, 0;
ADDWF 7H, 0;
```

Address	Data
005	22
006	22
007	22

- ❖ WREG after execution “MOVWF 7H” has 22H
- ❖ WREG after execution “ADDWF 7H,0” has 88H

EXAMPLE 2:

```
MOVLW 22H;
MOVWF 5H;
MOVWF 6H;
MOVWF 7H;
ADDWF 5, 0;
ADDWF 6, 0;
ADDWF 7, 1;
```

Address	Data
005	22
006	22
007	22

- ❖ After execution of “MOVWF 7H” WREG has ???
- ❖ After execution of “ADDWF 7,1” WREG and Address 005, 006 and 007 has??????

Alternative use of D bit

- PIC assembler allows us to use the letters W or F instead of 0 or 1 to indicate the destination.

```
ADDWF  fileReg, w    or    ADDWF  fileReg, f
```

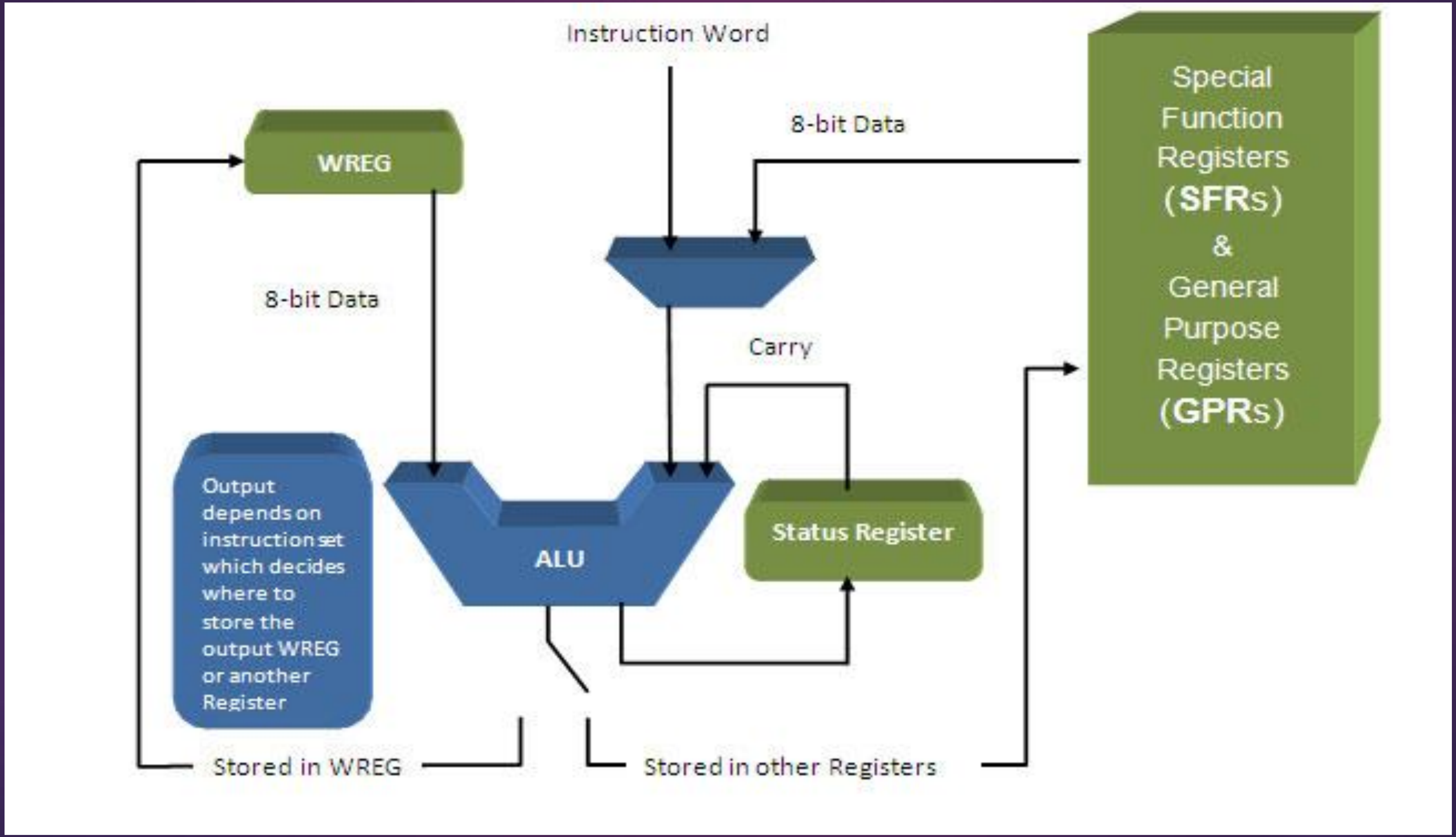
Example:

```
MOVLW 22H;
MOVWF 5H;
MOVWF 6H;
MOVWF 7H;
ADDWF 5, W;
ADDWF 6, W;
ADDWF 7, F;
```

Practice

- Lets see Example 2.2 and 2.3 of textbook.

WREG, fileReg, and ALU in PIC18


Instruction Word
8-bit Data
WREG
Special Function Registers (SFRs) & General Purpose Registers (GPRs)
8-bit Data
Carry
Output depends on instruction set which decides where to store the output WREG or another Register
ALU
Status Register
Stored in WREG
Stored in other Registers

COMF Instruction

- It complemenets the contents of fileReg and places the reult in WREG or fileReg.
- This is an example of Read-Modify-Write
- **FORMAT:** COMF fileReg, d
- EXAMPLE:

```
MOVLW 55H;
MOVWF  PORTB;
COMF PORTB, F;
```

Example:

- Write a simple program to toggle the SFR of PORT B continuously forever.

```
        MOVLW 55H;
        MOVWF  PORTB;
B1      COMF PORTB  F;
        GOTO B1;
```

DECF Instruction

- It decrements (subtarcts one from) the contents of fileReg and places the result in WREG or fileReg.
- **FORMAT:** DECF fileReg, d
- Example:

```
MOVLW 3;
MOVWF 20H;
DECF 0x20, F;
DECF 0x20, F;
DECF 0x20, F;
```

What if instead of F in DECF instructions, we write W?

MOVF Instruction

- The MOVF mnemonic is intended to perform MOVFW.
- **FORMAT:** MOVF fileReg, D
- We typically use this instruction to bring data into WREG from I/O pins and sometimes use it to copy fileReg to itself for the purpose of testing fileReg contents.

Examples:

- Write a program to get data from the SFRs of PORT B and send it to the SFRs of PORT C continuously.

```
AGAIN    MOVF PORTB,  W;
         MOVWF PORTC;
         GOTO AGAIN;
```

- Write a program to get data from the SFRs of PORT B. Add the value 5 to it and send it to the SFRs of PORT C.

MOVFF Instruction

- It copies data from one location in fileReg to another location in fileReg.
- It allows us to move data within the 4K space of the data RAM without going through the WREG register.

- **FORMAT:** MOVFF PORTB, PORTC

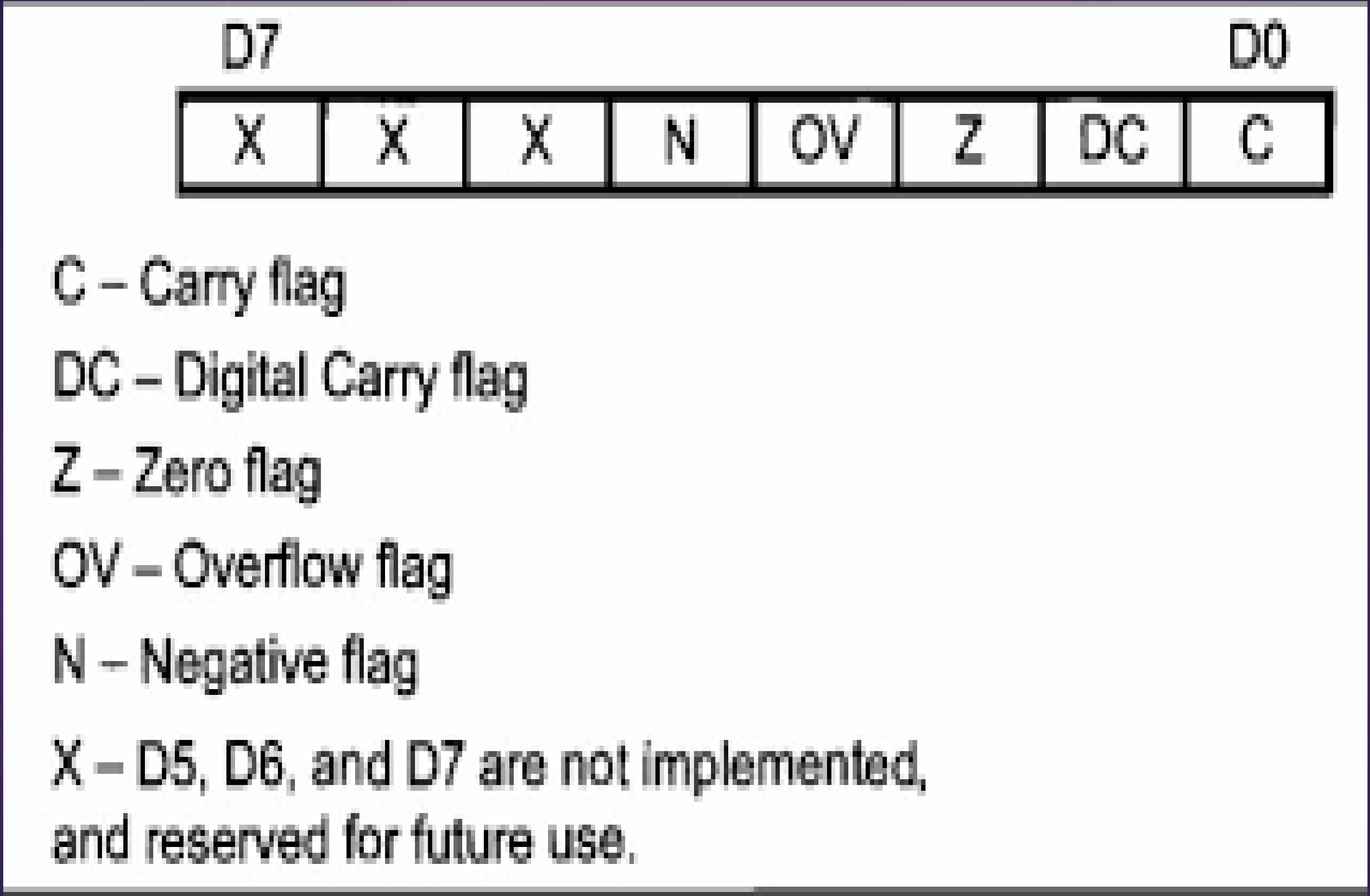
Example:

- Write a program to get data from the SFRs of PORT B and send it to the SFRs of PORT C continuously.
- Compare the above program with MOVF instruction program.
- **NOTICE:**

Using MOVFF, we simply copy data from one location to another. But when we use WREG, we can perform arithmetic and logic operationson data before it is moved.

PIC Status Register

- It is a flag register to indicate arithmetic conditions.
- It is an 8 bit register, also known as flag register.
- Although it is 8 bits wide, only 5 bits of it are used by the PIC18. The three unused bites are unimplemented and read as 0.
- The five flags are called conditional flags, meaning they indicate some conditions that result after an instruction is executed.



D7							D0
X	X	X	N	OV	Z	DC	C

C – Carry flag

DC – Digital Carry flag

Z – Zero flag

OV – Overflow flag

N – Negative flag

X – D5, D6, and D7 are not implemented, and reserved for future use.

Flag bits

- **C, the carry flag----** It is set whenever there is a carry out from D7 bit.

  Overflow beyond 8 bits

- **DC, the digital carry flag----** If there is carry from D3 to D4, this bit is set, otherwise, it is cleared. It is used in BCD arithmetic.

  Overflow from lower nibble (bit 3) to upper nibble (bit 4)

- **Z, the zero flag----** It reflects the result of an arithmetic or logic operation. If the result is 0, then Z=1 else, Z=0.
- **OV, the overflow flag----** This flag is set whenever the result of a signed number operation is too large, causing the high-order bit to overflow into the sign bit.

Flag bits

- In general, the carry flag is used to detect errors in unsigned arithmetic operations while the overflow flag is used to detect errors in signed arithmetic operations.
- **N, the negative flag----** Binary representation of signed numbers uses D7 as the sign bit. the negative flag reflects the result of an arithmetic operation. If D7 is 0, then N=0 and the result is positive. If D7 is 1 then N=1 and the result is negative.

Impact of ADDLW on the flag bits

- **Examples:** Show the status of C, DC and Z flags after the execution of 38H and 2FH in the given instructions:

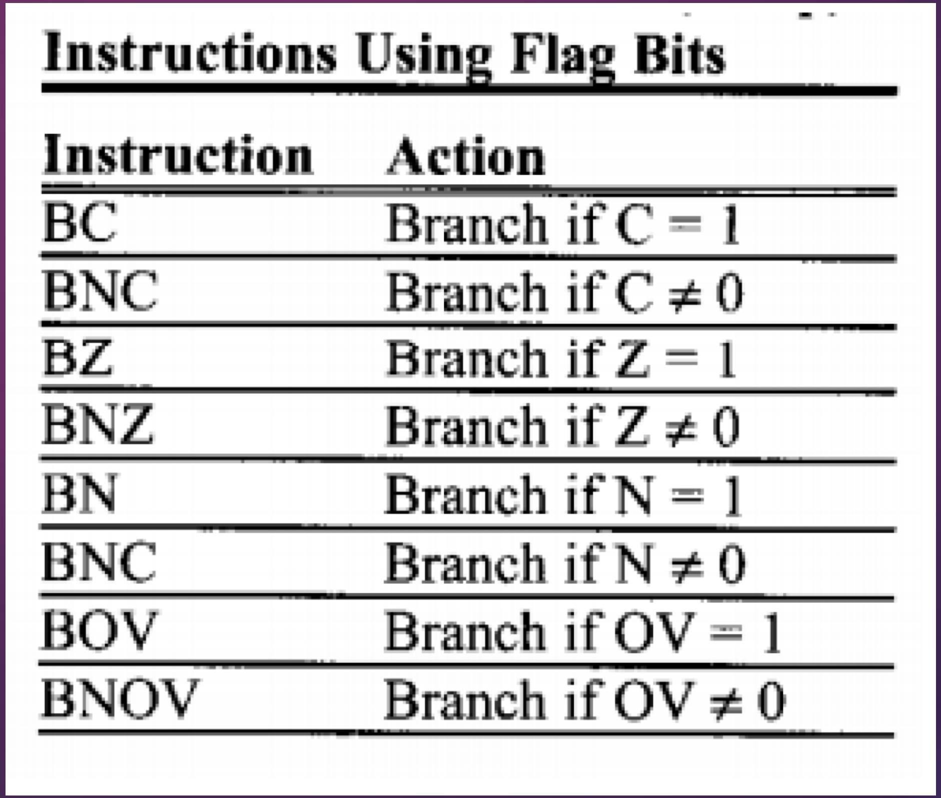
```
MOVLW 38H;
ADDLW 2FH;
```

- Write the instructions to add 9CH and 64H and then show the sttaus of C, DC and Z flags.

- Show the status of the above flags after the addiiton of 88H and 93H.

Flag bits and Decision making

- Because status flags are also called conditional flags, there are instructions that will make a conditional jump (branch) based on the status of the flag bits.



Instructions Using Flag Bits

Instruction	Action
BC	Branch if C = 1
BNC	Branch if C ≠ 0
BZ	Branch if Z = 1
BNZ	Branch if Z ≠ 0
BN	Branch if N = 1
BNC	Branch if N ≠ 0
BOV	Branch if OV = 1
BNOV	Branch if OV ≠ 0

Example:

```
    MOVLW 0xFF        ; Load W with 0xFF
    ADDWF COUNT, F    ; Add W to COUNT
    BNC    NoCarry    ; If Carry = 0, branch to NoCarry
    ; This code runs if Carry = 1
    INCF   OVERFLOW, F
NoCarry:
    NOP
```

Assembler Directives

- As instructions tell the CPU what to do, directives give instructions to the assembler.
- Some most widely used directives of the PIC are:
- **EQU (equate):** Used to define a constant value or a fixed address.

Example:

```
COUNT  EQU  0x25
        ..........
       MOVLW  COUNT;   WREG=25H
```

Advantage??

- **SET:** Used to define a constant value or a fixed address. The only difference between EQU and SET is the value assigned by the SET directive may be reassigned later.
- **ORG (origin):** Used to indicate the beginning address of a code. The number after ORG must be in hex.
- **END:** It indicates the end of the source (asm) file. It is the last line of the PIC program.
- **LIST:** It indicates to the assembler the specific PIC chip for which the program should be assembled.   LIST P=18F458
- **#include:** It tells the PIC assembler to use the libraries associated with the specific chip for which we are compiling the program.
- **_config:** Tells the assembler the configuration bits for the targeted PIC chip.
- **radix:** To indicate whether the numbering system is hexadecimal or decimal.   radix dec   (before ORG)

_config directive

- Configures hardware at programming time. (during chip programming)
- Examples of things it controls:

Oscillator selection

Watchdog Timer enable/disable

Power-up timer

```
LIST    P=16F877A
#include <p16f877a.inc>
         CONFIG  FOSC = HS       ; High-speed crystal oscillator
        CONFIG  WDTE = OFF      ; Disable Watchdog Timer
        CONFIG  PWRTE = ON      ; Enable Power-up Timer
```