Thread: Writing PBP Commands/macros

- 1st May 2010 15:21 #5

Darrel Taylor
Admin / meLabs Support

Join Date: Jul 2003
Location: Colorado Springs
Posts: 4,102

Ditto what Bruce said.
But I would add ...

When working at the ASM level, **everything is just Numbers**.
The assembler doesn't know anything about Byte Word or Long variables.
And it has no way of knowing if you passed it a constant, variable or your birthday.

If you pass a constant, the number is the value of the constant.
Passing a variable, will give the starting address of that variable in RAM.
Passing a Label, will give the address to a piece of code in Flash.
They are all just numbers, and it's up to you to handle them properly.

If you look through PBP's .mac files, you'll see that there is a different macro for each possibility of inputs.

LCDOUT?**C** ; expects to be passed a constant.
LCDOUT?**B** ; will work with a byte variable
... the other vars each have their own LCDOUT macro ...

The ?CB suffix indicates the type of input, in PBP they are ...

A = W register (**A**ccumulator)
C = Constant
B = Byte variable
W = Word variable
N = Long variable
T = Bit variable

Placing the suffix there, doesn't do anything by itself, it's just an easy way to recognize which macro you need to use, depending on the type of inputs you have.

With your **PE_Write** macro, there should be at least 4 versions.
And keeping with the PBP syntax, they might be called ...

Code:

```
;             :   Data,     Register
;-------------------------------------
PE_Write?CC   ; Constant,   Constant
PE_Write?CB   ; Constant,   Byte
PE_Write?BC   ; Byte,       Constant
PE_Write?BB   ; Byte,       Byte
```

Fortunately, PBP already has all the macros you need to move the different values around, so you don't really need to do MOVF, MOVWF, etc ... and those macro's will handle all the Banking issues that are difficult to manage with MOVF type instructions.
Those Macro's have the Name of **MOVE?**, and use the same suffixes shown above.

So the first macro using two constants would look like ...

Code:

```
ASM
PE_Write?CC   macro Data, Register
    MOVE?CB  Data, _DATA_BYTE
    MOVE?CB  Data, PORTB
    MOVE?CB  Register, _REGISTER_BYTE
    L?CALL   _SEND_PE_BYTE
  endm
```

The second macro with a constant Data, and Byte Register

Code:

```
PE_Write?CB   macro Data, Register
    MOVE?CB  Data, _DATA_BYTE
    MOVE?CB  Data, PORTB
    MOVE?BB  Register, _REGISTER_BYTE
    L?CALL   _SEND_PE_BYTE
  endm
```

Note that there is only one character different between the last two code sections (in blue).

The third macro ... Byte Data, Constant register ...

Code:

```
PE_Write?BC   macro Data, Register
    MOVE?BB  Data, _DATA_BYTE
    MOVE?BB  Data, PORTB
    MOVE?CB  Register, _REGISTER_BYTE
    L?CALL   _SEND_PE_BYTE
  endm
```

And the forth ... Byte Data, Byte register ...

Code:

```
PE_Write?BB   macro Data, Register
    MOVE?BB  Data, _DATA_BYTE
    MOVE?BB  Data, PORTB
    MOVE?BB  Register, _REGISTER_BYTE
    L?CALL   _SEND_PE_BYTE
  endm
ENDASM
```

You can probably see that if you consider ALL the possibilities, there's a lot of macros to write for a single function.

To use them, simply choose the macro that fits your inputs...

Code:

```
@  PE_Write?CC   55h, _GPIOA

Temp = $55
@  PE_Write?BC   _Temp, _GPIOA

Reg = GPIOA
@  PE_Write?BB   _Temp, _Reg
```

HTH,