PIC-EL III for Elmer-160 Students

Craig B. Johnson, AA0ZZ
3/13/2009

Elmer 160 Lesson 10 – Installing FPP
Unfortunately, FPP cannot talk to the PIC-EL III on a USB port so this lesson is not applicable for PIC-EL III users. Instead, install the Microchip PICKit2 Application or the MPLAB IDE as described in the PIC-EL III user manual.

Elmer 160 Lesson 11 – Programming a PIC
The first four topics of this lesson (Setting up FPP, Reading a PIC, Programming a PIC, Troubleshooting) are not applicable for PIC-EL III users since they are about using FPP. The last section of the lesson, Code Protected PICs, is still applicable.

Elmer 160 Appendix A – PIC Microcontroller Varieties
This Appendix contains a section called "16F84 to 16F628 Conversion". It contains information similar to what is contained here in the following "cookbook".

"Cookbook" for converting from 16F84A to 16F628A

1) Change the processor type "#include file" and the processor specification. This will bring in new CONFIG bit definitions as well as the new register definitions. Many definitions are changed from one bank to another and their locations within the banks may also be different.

Example:

```
processor  PIC16F628A
#include   'p16f628a.inc'
```

2) Change the CONFIG line to handle the new configuration bits:

```
LVP_ON     or   LVP_OFF       (LVP_OFF recommended.  See below.)
BODEN_ON   or   BODEN_ OFF    (BODEN_OFF recommended)
MCLRE_ON   or   MCLRE_OFF     (MCLRE_ON recommended)
```

Example:

```
__config   _CP_OFF & _LVP_OFF & _BODEN_OFF & _MCLRE_ON & _PWRTE_ON &
           _WDT_OFF & _XT_OSC
```

3) Change the "CBLOCK" directive to specify 0x20 instead of 0x0C. RAM in the 16F628A starts at address 0x20 while in the 16F84A it starts at 0x0C.

Example:

```
CBLOCK  0x20
   freq_0              ; local variable
   BCD_0               ; local variable
ENDC
```

4) Turn off the analog comparators (default in 16F628A) to make them digital, just like they are in the 16F84A. Analog comparators are new to the 16F628 and can be enabled later. Put this analog comparator turn-off code near the top of your first initialization routine.

Example:

```
Start
     movlw   0x07       ; Load 3-bit mask (CM2, CM1, CM0)
     movwf   CMCON      ; Write to CMCON register to turn off comparators
```

5) Be very careful of the 16F628A's low voltage programming mode, even if you are using the PIC-EL III and its high voltage programming method. Simply turn the LVP off in the CONFIG statement. (i.e. specify _LVP_OFF as in Item 2 above). This is important because the 16F628A is more sensitive than the 16F84A to current flow into PIC Pin 10 (RB4) when programming. Any circuitry attached to this pin that could source power which may cause the 16F628A to fail when programming.

6) The EEPROM usage was changed. EEDATA and EEADR are in Bank 1 for the 16F628A while they were in Bank 0 in the 16F84A. EECON1 is new. You need to check all statements in the code in which these registers are accessed and make sure you switch to Bank 1 before trying to use them.

Example:

```
    bsf     STATUS,RP0  ; Switch to Bank 1
    clrf    EEADR       ; Reset the EEPROM read address to beginning
    bsf     EECON1,RD   ; Read byte - set up EEDATA
    movf    EEDATA,w    ; Move the byte to W-register
    incf    EEADR,f     ; Increment the read address for next read
    bcf     STATUS,RP0  ; Back to Bank 0 for store into local variable
    movwf   osc_0       ; Save byte in a local variable
```

7) In the 16F628A, the EEIF bit is in the PIR1 register with the other interrupt flags. (In the 16F84, the EEIF bit is located in EECON1.) The EEIF interrupt bit is set by the PIC hardware when an EEPROM write sequence is complete. As an alternative, the WR bit in EECON1 can be polled for EEPROM write completion. (See the example code for Item 8 below.)

8) A special sequence is required for writing to the EEPROM. The sequence is different for a 16F628A than for the 16F84A. The code segment has the special sequence that is required to write one byte of data to EEPROM. (It uses the polling method (mentioned in Item 7) for determining EEPROM write completion.)

Example:

```
write_EEPROM
       bsf     EECON1,WREN       ; Set the EEPROM write enable bit
       ; Start required sequence
       bcf     INTCON,GIE        ; Disable interrupts
       movlw   0x55              ; Write 0x55 and 0xAA to EECON2
       movwf   EECON2            ;   control register, as required
       movlw   0xAA              ;
       movwf   EECON2            ;
       bsf     EECON1,WR         ; Set WR bit to begin write
       ; End required sequence
bit_check
       btfsc   EECON1,WR         ; Has the write completed?
       goto    bit_check         ; No, keep checking
       bsf     EECON1,GIE        ; Yes, enable  interrupts
       bcf     EECON1,WREN       ; Clear the EEPROM write enable bit
       incf    EEADR,f           ; Increment the EE write address
       return                    ; Return to the caller
```