CH-1092 Belmont, Switzerland
info@didel.com, www.didel.com

Compatibility 16F84 - 12C50x/16C5x

The 16F84 and 12C50x have minor differences on their instruction set, due to the 12-bit instruction length of the 12C50x and 16C5x. They also have I/O differences in addition to the unavailable I/O pins. If you plan to develop your application on a reprogrammable 16F84 before the final application on a 12C50x/16C5x, you have better to take care of the following points.

1) On the 12C50x/16C5x registers are available from address 16´7 to 16´1F (25 bytes). The 16C84/16F84 space for variables is from 16´C to 16´2F (36 bytes).
2) The 16C84/16F84 has a 64 byte EEPROM memory for storing parameters, not available on 12C50x/16C5x.
3) On the 16F84, use PortB <5..0>, address 6, and TrisB. TrisB is not an addressable register on the 12C, but the CALM assembler generates the correct code for Move W,TrisB. Bit 3 is an input only on 12C508 when initialized as a Port and not as the Clr input; on the 16F84, connect only an input line on RB3.
4) Pin 4 (GP3) of the 12C50x can be programmed as a reset pin, required if you need external hardware to reset the processor properly when the voltage is getting low. Pin 2 (GP5) can be programmed as RC oscillator. It is not clear if frequency lower than 4MHz are supported. Pins 2 and 3 (GP4 GP5) can be progammed to receive a quartz or resonator, required if the clock precision must be better than a few percent.
5) Pin 4 (GP3) of the 12C50x is an input only. Pins 7, 6,4 (GP0, GP1, GP3) can be configured with a weak pull-up and wake-up on change. Weak pull-ups ara automatic on the 16F84. On the 16F84, interrupt on pin change is available on RB<4..7> only.
6) Initialize differently the Option register; its bit 5 (TOCS) is one by default and forces GP2 (portB:#2) as an input, even if the Tris corresponding bit is zero. By default, initialize option with:
   Move #2´10000xxx,W Usual initialization xxx predivider value :2 4 8 .. 256
7) If you use the timer TMR0 to count some external event, you have to use pin 5 (GP2) on the 12C50x, and RA4, pin3, on the 16F84.
8) Do not use the Add #Constant,W and Sub W,#Constant,W instructions, not supported on the 12C50x. It is easy usually to change your "natural" way of programming:

```
            Move    Variable,W
            Add     #Constant,W     (result in W)
    by
            Move    #Constant,W
            Add     Variable,W
```

9) By the way, do not use the Sub instruction on the 16F84; the reversed operands and carry inversion due to the internal "add complement" is confusing. You have better to do your own "Add complement". Instead of

```
            Move    #Constant,W
            Sub   W,Variable  (result in  Variable)
    write
            Move    #256-Constant,W
            Add   W,Variable
```

What you may naturally want to do on the PIC

```
            Move    #256-Constant,W
            Sub     Variable,W
  is not available, but can be implemented as
            Move    #256-Constant,W
            Add     Variable,W
```

10) Saturation for a table is quite powerful; it is not required to make the table longer when the valiues do not change any more. On the 16F84 your are tempted to write

```
        ...
        Move    Var,W
        Call    TaDur
        ...
TaDur:
        Sub     #Max,W
        Skip,CC
        Clr     W
        Add     #Max,W
```

```
        Add     W,PCL
A:      RetMove #1,W    ; #1 is given back if Var = 0
        RetMove #5,W
        RetMove #9,W
B:      RetMove #11,W
 Max    = B-A   ; from 3 to 255, #11 is given back
```

On the 12C50x, the Add and Sub immediate do not exist. It is as fast to write

```
        ...     ; Var prêt
        Call    TaDur
        ...
TaDur:
        Move    #MaxD,W
        Sub     W,Var
        Skip,CC
        Clr     Var
        Add     Var,W
        Move    W,Var   ; if Var must be restored
        Add     W,PCL
A:      RetMove #1,W    ; #1 is given back if Var = 0
        RetMove #5,W
        RetMove #9,W
B:      RetMove #11,W
 Max    = B-A   ; from 3 to 255, #11 is given back
```

11) On the 16F84, do not use the IntCon register for synchronous programming, since this register is not available on the 12C50x. That is, if you want an operation to be done every 100 µs, do not wait for the flag T0IF (bit 2) of IntCon. Check the TMR0 value directly

```
    A:  Move TMR0,W
        Skip,EQ
        Jump    A
        Move    #256-(100/4)+2,W    ; 100 µs, predivider by 4
        Move    W,TMR0              ; Reload for the next loop
```

Do not use the instruction Test TMR0 (equivalent to Move TMR0,TMR0) which loaf the flag correctly, but interact with the counter, at least with the prescaler in divide by 4 mode; you will never get out of the loop.

Let the assembler handle the apparently complicated expression above:

a) it is an up-counter to be loaded by the complement of the number of cycles to count;

b) the prescaler has to be initialized as a divide by 4;

c) the +2 is to take care of the time lost before reloading the counter. Notice also that the A-loop is 4 cycles; do not pre-divide by 2, you may miss value zero!

12) The Ret instruction is not supported by the 12C50x. You have to use RetMove #0,W which modify W. Define the macro Ret when you switch toward the 12C50x/16C5x and never pass a variable parameter back from a routine through register W. Note that there are only two levels of routines on the 12C508. If you access a table, it is one level of routine!

13) Remove all access to PortA if you have been using some for helping the test on a 16F84 before programming the 12C508. PortA is address 5 and will map on OscCal register.

14) On the 508/509, The processor prepares ar reset a parameter in W which should be copied to OscCal register. This will guarantee a frequency close to the nominal value, slightly higher at 5V. Osccal can be changed any time, but ther is no clear correspondance between the parameter and the frequency.

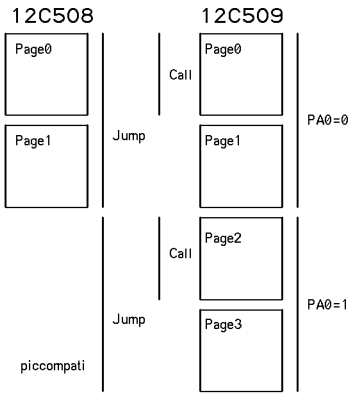
15) The 12C50x/16C5x has no interrupt. Intcon register does not exist (see above).

16) If the program use more than 256 bytes of memory, the Pic12C508 do not accept calls to address greater than 16´100. Calm assembler will signal an "illegal operand" error. On the PIC12C509, Jumps toward page 16´200 to 16´3FF must be prepared by setting bit F:#PA0. Call are possible if the same bit is prepared between 16´200 and 16´2FF only. Jumps toward or within page 2,3 must be masked to be accepted by assembler: Jump Adr.and.16´1FF. A macro Jump2 is preferably defined: when the assembler desagree, reflection is made about PA0 and Jump2 is used. Careful check with the listing must be frequently made.

Jump and call policy with 12C508/509

Careful page planning is required when programming for the 12C508/509. Let us name page 0,1,2,3 address range 00-16´FF, 16´100-16´1FF, etc. Pic12page 0 refers to our page 0 and page 1 (page01), Pic12page 1 is our page23. Be carefull with Microchip page definition, since it changes between 12- and 14-bit instruction processors.

Memory map for 12C509/509 is given next. Jumps are 9 bits. Calls are 8 bits, with address bit A8 forced to 0. No call can go toward page 1 and 3. PA0 bit in F selects page 23. When a jump or call will imply a page change, it must be preceded by a Set F:#PA0 or Clr F:#PA0 (macros SelPage23, SelPage01 predefined in 12C509.ref)



Usually page 0 must include the tables and calls. They can also fit on page 3. The recommended programming practice is to use different program counters, and update a memory map when the program develops.

.apc Page0 for tables and routines
.apc Page1 for main program
.apc Page2 for additional routines and program
.apc Page3 for more program

Compatibility with 12C67x processors

A/D registers are ... Interrupts are again available, as well as Option registers, and 3 instructions:

```
Add     #Val,W
Retl
Ret
```

The number of registers is increased. First bank provides xx registers. Second bank ..

Compatibility toward 16C7x processors

More complex processor of the PIC family do not have a direct access to TrisA and TrisB registers. These registers are addressable on a second register bank and a bit of the flag/mode register F do the bank selection. It the same on the 16C84/16F84, but the 12C50x/16C5x have only the Tris instructions. Microchip recommend not to use the Tris instruction, but the concern here is not to move toward more complex PIC processors, but be as much as possible compatible with low cost PICs. To make the change toward 17Cxx processors more easy, macros should be defined in the beginning of the program for initializing the port directions.

For instance, for the instructions that precharge an input to avoid wiring pull-up resistor, one can write (program not verified):

```
.Macro    PrechargeSwitch          ; On 16F84/
Set       PortB:#bSwitch
Move      #ModePrechargeSwitch,W   ; Output on b
Move      W,TrisB
Move      #ModeInputSwitch,W
Move      W,TrisB
.EndMacro
```

```
bRP0  = 16´5      ; Bank bit
      .Macro      PrechargeSwitch ; On 16C7x/17Cxx
      Set         F:#bRP0         ; sélectionne la banque
      Set         PortB:#bSwitch
      Clr         TrisB:#bSwitch
      Set         TrisB:#bSwitch
      Clr         F:#bRP0         ; remet la banque 0, en
      .EndMacro
(-- not verified --)
```

Let me know the other differences you may have noticed.
J.D. Nicoud nicoud@didel.com Jan 1999 - April 2001