INTRODUCTION

This application note shows how to implement a table look-up for the following devices:

- PIC12CXXX
- PIC12CEXXX
- PIC16CXXX
- PIC16CEXXX
- PIC16FXXX

The examples shown are for the PIC16CXXX family. An explanation of differences for the PIC16CSX family is at the end of this application note.

To access data in program memory, a table read operation must be performed. The table consists of a series of `retlw K` instructions where, the 8-bit table constants are assigned to the literal K. The first instruction in the table computes the offset to the table by using `addwf PCL,F` and consequently, the program branches to the appropriate `retlw K` instruction (Example 1).

EXAMPLE 1:

```
movlw offset ;load offset in w reg
call Table
```

Table:
```
addwf PCL,F ;add offset to pc to
 ;generate a computed goto
retlw 'A' ;return the ASCII char A
retlw 'B' ;return the ASCII char B
retlw 'C' ;return the ASCII char C
```

The method is straightforward, however, certain precautions have to be exercised when doing a table read in the PIC16CXXX.

IMPLEMENTATION

Program Counter Loading

The Program Counter (PC) in the PIC16CXXX is 13-bits wide. The low 8-bits (PCL) are mapped in RAM at location 02h and are directly readable and writable. The high 5-bits are not accessible directly and can only be written through the PCLATH register (Figure 1). The PCLATH register is a R/W register with only five of its bits implemented <4:0>, all other bits read as '0'.

FIGURE 1: LOADING OF PC IN DIFFERENT SITUATIONS

```
<table>
<thead>
<tr>
<th>PCH</th>
<th>PCL</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
```

Instruction with PCL as Destination

 Opcode <10:0>
SECTION 1

CALL and GOTO Instructions

When executing a CALL or GOTO, the low 11-bits are loaded directly from the instruction opcode. The high 2-bits are loaded from bits 3 and 4 of the PCLATH register. It is a good practice to pre-load PCLATH with the high byte of the routine’s address before executing the routine. This can be done as follows:

EXAMPLE 2:

```
movlw HIGH Table ;load high 8-bit address of Table
movwf PCLATH ;into PCLATH
call Routine ;execute Call
```

EXAMPLE 3:

```
org 0x80 ;code location in page 0
movlw offset ;load offset in w reg
call Table
```

```
org 0x320 ;Table located in page 3
Table:
  addwf PCL,F ;address of Table
  retlw 'A' ;return the ASCII char A
  retlw 'B' ;return the ASCII char B
  retlw 'C' ;return the ASCII char C
```

EXAMPLE 3A:

```
org 0x80
movlw HIGH Table
movwf PCLATH
movlw offset
call Table
```

```
org 0x320
Table:
  addwf PCL,F
  retlw 'A'
  retlw 'B'
```

Note: If the program memory size is less than 2K-words, then the above precaution is not necessary.

Computed GOTO Instruction

Any instruction with PCL as the destination, will load the PCH with the 5 low bits from the PCLATH (Figure 1). In Example 3, if the address where the CALL was made was on Page 0 and the address of the actual table was on Page 3, then when executing the computed GOTO, the program will go to a location in Page 0 instead of a location on Page 3. To prevent the program from branching to an unintended location when doing a table read, the PCLATH register should be pre-loaded with the high byte of the “Table” address. Example 3A shows how this can be done.
When doing a computed GOTO for a table read, care should be taken about page boundaries. The ADDWF PCL instruction will not compute a value greater than 8-bits. In Example 4, the result of the computed GOTO will result in a branch to an unintended portion of the code for a value in offset greater than zero. The user either has to be cautious as to where in a page the Table resides or has to monitor page roll-over and add it to the PCLATH ahead of the computed GOTO.

EXAMPLE 4:

```
org 0x80 ;code location in page 0
movlw HIGH Table ;load PCLATH with hi address
movwf PCLATH ;
movlw offset,F ;load offset in w reg
call Table
.
.
org 0x02ff ;Table located end of page 2
Table:
addwf PCL,F ;value in pc will not roll over to page 3
retlw 'A' ;return the ASCII char A
retlw 'B' ;return the ASCII char B
retlw 'C' ;return the ASCII char C
.
.
```

To take care of both table location and page boundary crossing, it is necessary to do a 13-bit computed GOTO operation as shown in Example 5.

The code in Example 5 will allow the user to place and access a table anywhere in program memory.

EXAMPLE 5:

```
org 0x80
movlw LOW Table ;get low 8 bits of address
addwf offset,F ;do an 8-bit add operation
movlw HIGH Table ;get high 5 bits of address
btfsc status,c ;page crossed?
addlw 1 ;yes then increment high address
movwf PCLATH ;load high address in latch
movf offset,w ;load computed offset in w reg
call Table
.
.
org 0x9FD
Table:
movwf PCL,F ;load computed offset in PCL
retlw ‘A’ ;return the ASCII char A
retlw ‘B’ ;return the ASCII char B
retlw ‘C’ ;return the ASCII char C
.
.
```

SECTION 2

Implementation for the PIC16C5X Family

The PIC16C5X has no PCH or PCLATH register, so the user has to take into consideration all the precautions mentioned in Section 1. In the PIC16C5X, the location of the Table has to be in the top half of a 512 word page. This restriction is not valid for the PIC16CXXX family. To convert a table read operation from PIC16C5X code to the PIC16CXXX code, the following should be done:

- Remove any program memory page select instructions (PIC16C56/57), if present.
- Do a 13-bit computed GOTO operation (as shown in Example 5), when doing a table read operation.
Note the following details of the code protection feature on PICmicro® MCUs.

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable".
- Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our product.

If you have any further questions about this matter, please contact the local sales office nearest to you.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

Note the following details of the code protection feature on PICmicro® MCUs.

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable".
- Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our product.

If you have any further questions about this matter, please contact the local sales office nearest to you.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip’s products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

Trademark

The Microchip name and logo, the Microchip logo, FilterLab, KEELOQ, microID, MPLAB, PIC, PICmicro, PICMASTER, PICSTART, PRO MATE, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

dsPIC, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, microPort, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, MXDEV, PICC, PICDEM, PICDEM.net, rIPIC, Select Mode and Total Endurance are trademarks of Microchip Technology Incorporated in the U.S.A.

Serialized Quick Turn Programming (SQTP) is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2002, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

Printed on recycled paper.