4. PIC example
   4.1. Basic control software
       4.1.1. Requirements / features
       4.1.2. Global declarations
           4.1.2.1. Register declarations
           4.1.2.2. Literal declarations
           4.1.2.3. Procedure declarations / library interface
       4.1.3. Code
           4.1.3.1. LCD initialisation
           4.1.3.2. Busy flag
           4.1.3.3. Clear display
           4.1.3.4. Cursor home
           4.1.3.5. Entry mode
           4.1.3.6. Display mode
           4.1.3.7. Set character generator RAM address
           4.1.3.8. Set display data RAM address
           4.1.3.9. Get address counter contents
           4.1.3.10. Write character
           4.1.3.11. Write command
           4.1.3.12. Delay loops
       4.1.4. Availability
   4.2. Advanced control software
       4.2.1. User defined characters
...
   4.3. Used hardware
       4.3.1 Controller
       4.3.2 LCD hardware interface
   4.4. Development environment
       4.4.1. Software
       4.4.2. Hardware
4.1.1. Requirements / features

- HD44780-based (industry-standard) character-LCD, all software in this chapter is based on it's instruction-set.
- PIC16C84 running on a 4MHz crystal, some code is based on this frequency.
- 8-bit interface between microcontroller and LCD-module.

4.1.2. Global declarations

To get things working.

4.1.2.1. Register declarations

*Purpose:*
- Tells MPASM which ports and registers (files) to use.

*Code:*

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCD_DATA EQU</td>
<td>PORTB</td>
<td>LCD data lines interface</td>
</tr>
<tr>
<td>LCD_DATA_TRIS</td>
<td>EQU TRISB</td>
<td>LCD data lines interface</td>
</tr>
<tr>
<td>LCD_CTRL EQU</td>
<td>PORTA</td>
<td>LCD control lines interface</td>
</tr>
<tr>
<td>LCD_TEMP EQU</td>
<td>0x020</td>
<td>LCD subroutines internal use</td>
</tr>
<tr>
<td>DELAY EQU</td>
<td>0x023</td>
<td>Used in DELAYxxx routines</td>
</tr>
<tr>
<td>X_DELAY EQU</td>
<td>0x024</td>
<td>Used in X_DELAYxxx routines</td>
</tr>
</tbody>
</table>

4.1.2.2. Literal declarations

*Purpose:*
- Literal declarations (Equates) used in the code.

*Code:*

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCD_E</td>
<td>2</td>
<td>LCD Enable control line</td>
</tr>
<tr>
<td>LCD_RW</td>
<td>1</td>
<td>LCD Read/Write control line</td>
</tr>
<tr>
<td>LCD_RS</td>
<td>0</td>
<td>LCD Register-Select control line</td>
</tr>
</tbody>
</table>

4.1.2.3. Procedure declarations / library interface

Since MPLIB and MPLINK are not yet available, no declarations are needed.

4.1.3. Code
4.1.3.1. LCD initialisation

**Purpose:**
- LCD initialisation code to be executed after power-up (i.e.: before any other subroutine)
- Should be modified to your needs (i.e. display type, cursor on/off, etc.)

**Code:**

```assembly
; LCD initialisation code to be executed after power-up (i.e.: before any other subroutine).
; Should be modified to your needs (i.e. display type, cursor on/off, etc.)

; Busy flag is not yet valid
CLR LCD_CTRL ; ALL PORT output should output Low.
; power-up delay
MOVLW 0x01E
CALL X_DELAY500 ; 30 * 0.5mS = 15mS
; Busy Flag should be valid from here
MOVLW 0x038 ; 8-bit-interface, 2-lines
CALL LCDPUTCMD
MOVLW 0x000 ; disp.off, curs.off, no-blank
CALL LCDMODE
CALL LCDCLEAR
MOVLW 0x004 ; disp.on, curs.off
CALL LCDMODE
MOVLW 0x002 ; auto-inc (shift-cursor)
CALL LCDMODE
RETURN
```

4.1.3.2. Busy flag

**Purpose:**
- Tests if the LCD is busy. Returns when LCD busy-flag is inactive.

**Code:**

```assembly
; Tests if the LCD is busy. Returns when LCD busy-flag is inactive.

; Select Register page 1
BSF STATUS,RP0
; Set PORTB for input
MOVLW 0xOFF
MOVF LCD_DATA, W
; Read busy flag + DDram address
MOVF LCD_DATA, W
; Read busy flag + DDram address
ANDLW 0x80
; Check Busy flag, High = Busy
BTFSS STATUS, Z
GOTO LCDBUSY

; Select Register page 0
BCF STATUS, RP0
; Set LCD for command mode
BCF LCD_CTRL, LCD_RS
; Setup to read busy flag
BCF LCD_CTRL, LCD_E
; LCD E-line High
MOVF LCD_DATA, W
; Read busy flag + DDram address
BCF LCD_CTRL, LCD_E
; LCD E-line Low
ANDLW 0x80
; Check Busy flag, High = Busy
BTFSS STATUS, Z
GOTO LCDBUSY

; Select Register page 1
BCF LCD_CTRL, LCD_RW
; Select Register page 0
BCF STATUS, RP0
; Set PORTB for output
MOVLW 0x000
MOVF LCD_DATA, W
; Read busy flag + DDram address
MOVF LCD_DATA, W
; Read busy flag + DDram address
ANDLW 0x80
; Check Busy flag, High = Busy
BTFSS STATUS, Z
GOTO LCDBUSY

; Select Register page 0
BCF STATUS, RP0
; Select Register page 0
RETURN
```

4.1.3.3. Clear display

**Purpose:**
- Clears display and returns cursor to home position (upper-left corner).

**Code:**
4.1.3.4. Cursor home

**Purpose:**
- Returns cursor to home position.
- Returns display to original position (when shifted).

**Code:**
```
LCDCLEAR
    MOVLW 0x001
    CALL LCDPUTCMD
    RETURN
```

```
4.1.3.5. Entry mode

**Purpose:**
- Sets entry mode of the LCD
- Required entry mode must be set in $W$
  - $b0$: 0 = no display shift, 1 = display shift
  - $b1$: 0 = auto-decrement, 1 = auto-increment
  - $b2$-$b7$: don't care

**Code:**
```
LCDEMODE
    ANDLW 0x003    ; Strip upper bits
    IORLW 0x004    ; Function set
    CALL LCDPUTCMD
    RETURN
```

4.1.3.6. Display mode

**Purpose:**
- Sets display control
- Required entry mode must be set in $W$
  - $b0$: 0 = cursor blink off, 1 = cursor blink on (if $b1 = 1$)
  - $b1$: 0 = cursor off, 1 = cursor on
  - $b2$: 0 = display off, 1 = display on (display data remains in DD-RAM)
  - $b3$-$b7$: don't care

**Code:**
```
LCDDMODE
    ANDLW 0x007    ; Strip upper bits
    IORLW 0x008    ; Function set
    CALL LCDPUTCMD
    RETURN
```

4.1.3.7. Set character generator RAM address

**Purpose:**
4.1.3.8. Set display data RAM address

Purpose:
- Sets the Display-Data-RAM address. DDRAM data is read/written after this setting.
- Required entry mode must be set in W
  b0-6 : required DDRAM address
  b7 : don't care

Code:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LCDSDDA</td>
<td>IORLW 0x080; Function set</td>
</tr>
<tr>
<td></td>
<td>CALL LCDPUTCMD</td>
</tr>
<tr>
<td></td>
<td>RETURN</td>
</tr>
</tbody>
</table>

4.1.3.9. Get address counter contents

Purpose:
- Returns address counter contents, used for both DDRAM and CGRAM.
- RAM address is returned in W

Code:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LCDGADDR</td>
<td>BSF STATUS,RP0; Select Register page 1</td>
</tr>
<tr>
<td></td>
<td>MOVLW 0x0FF; Set PORTB for input</td>
</tr>
<tr>
<td></td>
<td>MOVF LCD_DATA_TRIS</td>
</tr>
<tr>
<td></td>
<td>BCF STATUS,RP0; Select Register page 0</td>
</tr>
<tr>
<td></td>
<td>BCF LCD_CTRL, LCD_RS; Set LCD for command mode</td>
</tr>
<tr>
<td></td>
<td>BSF LCD_CTRL, LCD_RW; Setup to read busy flag</td>
</tr>
<tr>
<td></td>
<td>BSF LCD_CTRL, LCD_E; LCD E-line High</td>
</tr>
<tr>
<td></td>
<td>MOVF LCD_DATA, W; Read busy flag + RAM address</td>
</tr>
<tr>
<td></td>
<td>BCF LCD_CTRL, LCD_E; LCD E-line Low</td>
</tr>
<tr>
<td></td>
<td>ANDLW 0x07F; Strip upper bit</td>
</tr>
<tr>
<td></td>
<td>BSF LCD_CTRL, LCD_RW</td>
</tr>
<tr>
<td></td>
<td>BSF STATUS,RP0; Select Register page 1</td>
</tr>
<tr>
<td></td>
<td>MOVF 0x000</td>
</tr>
<tr>
<td></td>
<td>MOVF LCD_DATA_TRIS; Set PORTB for output</td>
</tr>
<tr>
<td></td>
<td>BCF STATUS,RP0; Select Register page 0</td>
</tr>
<tr>
<td></td>
<td>RETURN</td>
</tr>
</tbody>
</table>

4.1.3.10. Write character

Purpose:
- Sends character to LCD
- Required character must be in W

Code:
### 4.1.3.11. Write command

**Purpose:**
- Sends command to LCD
- Required command must be in W

**Code:**

```assembly
LCDPUTCHAR
    MOVWF  LCD_TEMP ; Character to send is in W
    CALL   LCDBUSY  ; Wait for LCD to be ready
    BCF    LCD_CTRL, LCD_RW; Set LCD in read mode
    BSF    LCD_CTRL, LCD_RS; Set LCD in data mode
    BSF    LCD_CTRL, LCD_E ; LCD E-line High
    MOVF   LCD_TEMP, W
    MOVF   LCD_DATA; Send data to LCD
    BCF    LCD_CTRL, LCD_E ; LCD E-line Low
    RETURN
```

### 4.1.3.12. Delay loops

**Purpose:**
- Used in LCDINIT subroutine
- Required delay factor must be in W
  (Could be coded more efficient, but this approach gives more flexibility)

**Code:**

```assembly
;****************************************************** a 500uS delay @ 4MHz X-tal
DELAY500 MOVLW D'165'       ; +1 1 cycle
             MOVF  DELAY ; +2 1 cycle
             DECFSZ DELAY, F; step 1 1 cycle
             GOTO  DELAY500_LOOP ; step 2 2 cycles
             RETURN ; +3 2 cycles
;****************************************************** a delay of 'W' * 500mS
X_DELAY500  MOVWF  X_DELAY ; +1 1 cycle
X_DELAY500_LOOP CALL   DELAY500 ; step1 wait 500uSec
             DECFSZ X_DELAY, F ; step2 1 cycle
             GOTO  X_DELAY500_LOOP ; step3 2 cycles
X_DELAY500_END RETURN ; +2 2 cycles
```

### 4.1.4. Availability

**LCD-PIC.ZIP:** an example using some of the above subroutines (all subroutines are included).
Source is coded for a 4*20 LCD, adjust it to your needs!

Shows the following screen on a 4*20 LCD:
Shows the following screen on a 2*40 LCD:

<table>
<thead>
<tr>
<th>This is on line : 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is on line : 1</td>
</tr>
<tr>
<td>This is on line : 2</td>
</tr>
<tr>
<td>This is on line : 3</td>
</tr>
</tbody>
</table>

Shows the following screen on a 2*20 LCD:

<table>
<thead>
<tr>
<th>This is on line : 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is on line : 1</td>
</tr>
</tbody>
</table>

4.2. Advanced control software

4.2.1. User defined characters

**Purpose:**
After several requests a quick explanation on how to implement user-defined characters:

First you'll need to make a pixel definition for the characters you want to use. This is the pixel definition for an underlined '0' (char code 0x30) based on a 5x7 dots character definition:

<table>
<thead>
<tr>
<th>row</th>
<th>bits</th>
<th>byte value</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>xxx</td>
<td>0x0E</td>
</tr>
<tr>
<td>001</td>
<td>x x</td>
<td>0x11</td>
</tr>
<tr>
<td>010</td>
<td>x xx</td>
<td>0x13</td>
</tr>
<tr>
<td>011</td>
<td>x x x</td>
<td>0x15</td>
</tr>
<tr>
<td>100</td>
<td>xxx</td>
<td>0x19</td>
</tr>
<tr>
<td>101</td>
<td>x x</td>
<td>0x11</td>
</tr>
<tr>
<td>110</td>
<td>xxx</td>
<td>0x0E</td>
</tr>
<tr>
<td>111</td>
<td>xxxxx</td>
<td>0x1F</td>
</tr>
</tbody>
</table>

The byte values need to be loaded into CGRAM address 00cccrrr (binary), where:
- ccc = user-defined character number (0...7)
- rrr = row number of the user defined character (0...7)

Once that's done you can write character codes 0...7 to the desired LCD character position, just like you do with 'normal' characters.

User-defined character definitions may be changed 'on-the-fly'.
4.3. Used hardware

4.3.1 Controller

- A PIC16C84 is used to control the LCD.
- 8-bit data interface between controller and LCD.

4.3.2 LCD hardware interface

(More detailed code may be published some day)
4.4.1. Software

- Assembler: MPASM V1.30
- Programmer software: PICSTART 16B1 V5.00.00

4.4.2. Hardware

- Programmer PICSTART 16B1 (firmware V2.00)