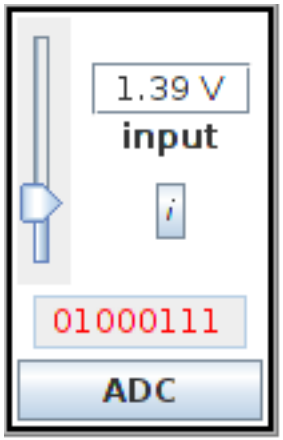
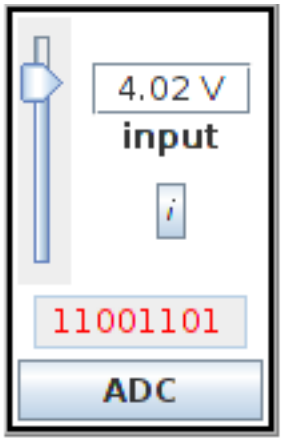
ADC Bar Graph Indicator on LCD



Low voltage on ADC input results in few bars being displayed.



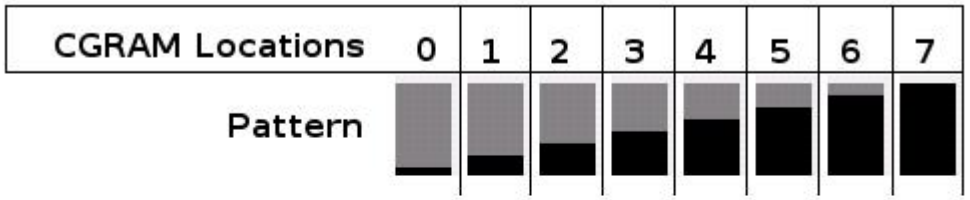
Higher voltage on ADC input results in more bars being displayed.

The program below displays a bar graph on the LCD. The graph is an indicator of the voltage applied to the ADC, as illustrated.

As can be seen, the first location on the top line of the display and the first location on the bottom line of the display together make up the bar graph indicator. Since each character has eight rows (5 * 8, when cursor position is used), the maximum number of bars is 16.

The ADC has eight output lines, resulting in 256 digital outputs representing the 0 to 5 V input voltage range. However, since the indicator only has 16 bars, the program below uses the top four bits from the ADC output.

Eight different bar patterns are stored in the LCD module's CGRAM, and these are used to make up the bar graph indicator. The patterns are shown opposite.



Note: for a given pattern, the number of rows with all dots displayed (ie: row containing 1111B) is one greater than the CGRAM location - location 0 has 1 bar, location 1 has 2 bars ... location 7 has 8 bars.

Of the four outputs from the ADC that are used, the bottom three (ranging from 000B to 111B - ie: 0 to 7) are used to select the appropriate bar pattern in CGRAM.

The MSB is used to determine on which display line the bar pattern should be placed. If the MSB is 0, put the pattern on the bottom line and put a space on the top line (ie: blank the top line).

If the MSB is 1, put the pattern on the top line and put pattern 7 from CGRAM (ie: all eight bars displayed) on the bottom line.

Once the program completes initialisation, it enters an endless loop. A timer interrupts every 200 us and the timer ISR initiates an ADC conversion. When the conversion is complete, the ADC causes an external interrupt. The external ISR then updates the LCD accordingly.

The simulator is of course not real time, therefore when running this program it is best to set the update frequency to 1000 (ie: 1000 instructions executed between each update of the simulator GUI).

```
org 0                           ; reset vector
        JMP main                ; jump to the main program

org 3                           ; external 0 interrupt vector
        JMP ext0ISR             ; jump to the external 0 ISR

org 0BH                         ; timer 0 interrupt vector
        JMP timer0ISR           ; jump to timer 0 ISR


org 30H
main:
        SETB IT0                ; set external 0 interrupt as edge-activated
        SETB EX0                ; enable external 0 interrupt
        MOV TMOD, #2            ; set timer 0 as 8-bit auto-reload interval timer
        MOV TH0, #-200          ; | put -200 into timer 0 high-byte - this reload value,
                                ; | with system clock of 12 MHz, will result in a timer 0 overflow every 200 us
        MOV TL0, #-200          ; | put the same value in the low byte to ensure the timer starts counting from
                                ; | 56 (256 - 200) rather than 0
        SETB TR0                ; start timer 0
        SETB ET0                ; enable timer 0 interrupt



; initialise the display
; see instruction set for details


        CLR P1.3                ; clear RS - indicates that instructions are being sent to the module

; function set
        CLR P1.7                ; |
        CLR P1.6                ; |
        SETB P1.5               ; |
        CLR P1.4                ; | high nibble set

        SETB P1.2               ; |
        CLR P1.2                ; | negative edge on E
```

```
        CALL delay              ; wait for BF to clear
                                ; function set sent for first time - tells module to go into 4-bit mode
; Why is function set high nibble sent twice? See 4-bit operation on pages 39 and 42 of HD44780.pdf.

        SETB P1.2               ; |
        CLR P1.2                ; | negative edge on E
                                ; same function set high nibble sent a second time

        SETB P1.7               ; low nibble set (only P1.7 needed to be changed)

        SETB P1.2               ; |
        CLR P1.2                ; | negative edge on E
                                ; function set low nibble sent

        CALL delay              ; wait for BF to clear


; entry mode set
; set to increment with no shift
        CLR P1.7                ; |
        CLR P1.6                ; |
        CLR P1.5                ; |
        CLR P1.4                ; | high nibble set

        SETB P1.2               ; |
        CLR P1.2                ; | negative edge on E

        SETB P1.6               ; |
        SETB P1.5               ; |low nibble set

        SETB P1.2               ; |
        CLR P1.2                ; | negative edge on E

        CALL delay              ; wait for BF to clear


; display on/off control
; the display is turned on, the cursor is turned off and blinking is turned off
        CLR P1.7                ; |
        CLR P1.6                ; |
        CLR P1.5                ; |
        CLR P1.4                ; | high nibble set

        SETB P1.2               ; |
        CLR P1.2                ; | negative edge on E

        SETB P1.7               ; |
```

```
        SETB P1.6               ; | low nibble set

        SETB P1.2               ; |
        CLR P1.2                ; | negative edge on E

        CALL delay              ; wait for BF to clear


; set CGRAM address
; set to character 0, row 0 (address 000 000B) - all 8 CGRAM locations will be used to store the bar graph patterns
        CLR P1.7                ; |
        SETB P1.6               ; |
        CLR P1.5                ; |
        CLR P1.4                ; | high nibble set

        SETB P1.2               ; |
        CLR P1.2                ; | negative edge on E

        CLR P1.6                ; |low nibble set

        SETB P1.2               ; |
        CLR P1.2                ; | negative edge on E

        CALL delay              ; wait for BF to clear


; put first bar symbol in RAM
        MOV 30H, #0     ; | Locations 30H to 37H in RAM will be used to store the bar graph patterns before sending them
to CGRAM.
        MOV 31H, #0     ; | Start by putting 0 into all except the last one (37H).
        MOV 32H, #0     ; | R0 will be used to point to the last position. Then 11111B will be put into this location
(using indirect
        MOV 33H, #0     ; | addressing via R0). This means the first pattern is all rows 0, except the last row all 1.
        MOV 34H, #0     ; | R0 will then be incremented, which means the next row (location 36H in RAM) gets 11111B.
This results in
        MOV 35H, #0     ; | the first 6 rows all 0 and the last two all 1. Repeating 8 times results in the last pattern
containing
        MOV 36H, #0     ; | 11111B in all rows.
        MOV 38H, #0FFH  ; end of data - sendBar subroutine checks for FFH to see if all 8 rows have been sent.

        MOV R0, #37H    ; point to highest bar (ie: 11111B) position


; send bar symbols to CGRAM
        SETB P1.3               ; set RS - indicates that data is being sent to module
        MOV R2, #8              ; 8 patterns need to be sent
again:
```

```
        MOV @R0, #11111B        ; R0 points to the next row (row location in RAM) where a bar (11111B) needs to be
stored
        DEC R0                          ; decrement R0 to point to next row for next iteration
        CALL sendPattern        ; send the current pattern to CGRAM
        DJNZ R2, again          ; repeat for 8 patterns


        SETB EA                 ; set the global interrupt enable bit
        JMP $                   ; do nothing



;-------------------------------------------------------------------------------



; updateBarGraph subroutine
updateBarGraph:
        CLR A                   ; | The bar graph has a max 16 rows (2 characters, 8 rows in each).
        MOV C, B.4              ; | Therefore, only the top 4 bits in B (outputs of ADC were transferred to B by ADC
ISR) are used.
        MOV ACC.0, C            ; | Of these 4 bits, the bottom 3 are placed in A, while the MSB is left in C.
        MOV C, B.5              ; | The bottom 3 bits (0 to 7) decide how many bars are displayed (ie; which of the
patterns
        MOV ACC.1, C            ; | in CGRAM are displayed), while the MSB decides which position to display it - MSB =
1, display
        MOV C, B.6              ; | on top line, MSB = 0, display on bottom line.
        MOV ACC.2, C            ; |
        MOV C, B.7              ; |

        JC topLine              ; if C is set, the voltage to the ADC is greater than 2.5V, therefore display on top
line

; do the following if voltage to ADC is less than 2.5V
        CALL toBottom           ; set DDRAM address to bottom line
        CALL sendCharacter      ; the value in A (ranging from 0 to 7) points to the appropriate bar symbol in CGRAM
        CALL toTop              ; set DDRAM address to top line
        MOV A, #' '             ; since the voltage to the ADC is less than 2.5V, the top line must be cleared (space
character is used)
        CALL sendCharacter      ; send space character to top line
        RET


; do the following if voltage to ADC is greater than 2.5V
topLine:
        CALL toTop              ; set DDRAM address to top line
        CALL sendCharacter      ; the value in A (ranging from 0 to 7) points to the appropriate bar symbol in CGRAM
```

```
        CALL toBottom           ; set DDRAM address to bottom line
        MOV A, #7               ; | since the voltage to the ADC is greater than 2.5V, the bottom line must get a full
                                ; | bar graph symbol (location 7 in CGRAM)
        CALL sendCharacter      ; send full bar graph to bottom line
        RET



; toTop subroutine - point to top row
; set DDRAM address to 0
toTop:
        CLR P1.3                ; clear RS - indicates that instructions are being sent to the module

        SETB P1.7               ; |
        CLR P1.6                ; |
        CLR P1.5                ; |
        CLR P1.4                ; | high nibble set

        SETB P1.2               ; |
        CLR P1.2                ; | negative edge on E

        CLR P1.7                ; low nibble set

        SETB P1.2               ; |
        CLR P1.2                ; | negative edge on E

        CALL delay              ; wait for BF to clear
        RET



; toBottom subroutine - point to bottom row
; set DDRAM address to 40H
toBottom:
        CLR P1.3                ; clear RS - indicates that instructions are being sent to the module

        SETB P1.7               ; |
        SETB P1.6               ; |
        CLR P1.5                ; |
        CLR P1.4                ; | high nibble set

        SETB P1.2               ; |
        CLR P1.2                ; | negative edge on E

        CLR P1.7                ; |
        CLR P1.6                ; | low nibble set
```

```
        SETB P1.2               ; |
        CLR P1.2                ; | negative edge on E

        CALL delay              ; wait for BF to clear
        RET



; sendPattern subroutine - send bar pattern symbol to CGRAM
sendPattern:
        MOV R1, #30H            ; pattern is stored in RAM starting at location 30H
loop:
        MOV A, @R1              ; move one row of pattern to A
        CJNE A, #0FFH, skip     ; | check to see if A contains FFH - if yes
        RET                     ; | then all rows of pattern have been sent, therefore return
skip:
        CALL sendCharacter      ; send the pattern to the lcd module
        INC R1                  ; increment R1 to point at next row in pattern
        JMP loop                ; repeat
        RET


; sendCharacter subroutine

sendCharacter:
        SETB P1.3               ; set RS - indicates that data is being sent to module

        MOV C, ACC.7            ; |
        MOV P1.7, C             ; |
        MOV C, ACC.6            ; |
        MOV P1.6, C             ; |
        MOV C, ACC.5            ; |
        MOV P1.5, C             ; |
        MOV C, ACC.4            ; |
        MOV P1.4, C             ; | high nibble set

        SETB P1.2               ; |
        CLR P1.2                ; | negative edge on E

        MOV C, ACC.3            ; |
        MOV P1.7, C             ; |
        MOV C, ACC.2            ; |
        MOV P1.6, C             ; |
        MOV C, ACC.1            ; |
        MOV P1.5, C             ; |
        MOV C, ACC.0            ; |
        MOV P1.4, C             ; | low nibble set
```

```
        SETB P1.2               ; |
        CLR P1.2                ; | negative edge on E

        CALL delay              ; wait for BF to clear


; delay subroutine - used to allow LCD module carry out internal operation.
delay:
        MOV R3, #50
        DJNZ R3, $
        RET



; timer 0 ISR - simply starts an ADC conversion
timer0ISR:
        CLR P3.6                ; clear ADC WR line
        SETB P3.6               ; then set it - this results in the required positive edge to start a conversion
        RETI                    ; return from interrupt


; external 0 ISR - responds to the ADC conversion complete interrupt
ext0ISR:
        CLR P3.7                ; clear the ADC RD line - this enables the data lines
        MOV B, P2               ; move ADC outputs to B
        SETB P3.7               ; disable the ADC data lines by setting RD
        CALL updateBarGraph     ; update the bar graph using the new reading from the ADC
        RETI                    ; return from interrupt
```
