Displaying Characters on an LCD Character Module

I. Introduction:

This application note describes a simple technique to display characters from both the internal character generator and user designed characters on an LCD character module. The controlling microcontroller is a Phillips 87C751, a derivative of the popular Intel 8051. The LCD module is connected to the microcontroller through its I/O ports. It could also be connected directly to the data bus with the addition of address decoding logic.

The process of displaying character to this module is divided into three steps. First the module must be initialized. This sets up the built-in LCD controller chip. Second, some user designed characters are uploaded to the CGRAM. This allows the displaying of up to 8 custom characters in addition to the 192 character permanently stored in the module. Lastly, a message consisting of a mix of standard ASCII characters and custom designed characters is displayed on the module.

II. Circuit Schematic

[Diagram of the circuit schematic is included here.]
III. Software Flowchart:

Begin

P1 = 38h, Command_Byte

4.1mSec Delay

P1 = 38h, Command_Byte

100uSec Delay

P1 = 38h, Command_Byte
P1 = 0ch, Command_Byte
P1 = 01h, Command_Byte

CGRAM

First_Line

Second_Line:

IDLE

* CGRAM: Program in CGRAM DATA
First_Line: Display First Line
Second_Line: Display Second Line

DELY

Start Timer

TestFlag = 0

TestFlag = 1

TestFlag = 1?

No

Yes

Return

Write

Data=[DPTR]

Data ? 99h

Yes

No

Call Data_Byte

INC DPTR

Command_Byte

Data_byte

RS low

RS high

RW low - Enable write

E low - generate enable pulse

nop

E high - pull up enable signal

RW high - Read mode

Configure P1 to input port

DB7 = 1?

No

Yes

Return
Displaying Characters on an LCD Character Module

Description: Demo software to display "canned" message and custom characters.

Controller: Phillips 87C751

LCD controller: HD44780, KS0066, SED1278

**Constant Definition**

```
EnableT0 equ 082h ; enable timer 0
Disable equ 000h ; disable timer
D4100h equ 00ch ; timer reload high byte def.
D4100l equ 003h ; timer reload = 4.1mSec.
D100h equ 000h ; timer reload
D100l equ 04ch ; timer reload = 100uSec.
```

**Ram Definition**

```
Flags DATA 020h ; flag
TstFlag BIT Flags.0 ; interrupt flag bit
```

**Port Connections**

```
P1.0 -> D0
P1.1 -> D1
P1.2 -> D2
... 
P1.7 -> D7
P3.0 -> Enable
P3.1 -> RS
P3.2 -> RW
```

**Interrupt Vectors**

```
org 000h
jmp PowerUp ; Power up reset vector
org 003h
jmp ExInt0 ; External interrupt 0 vector
org 008h
jmp Timer0 ; Counter/Timer 0 int vector
org 013h
jmp ExInt1 ; External int 1 vector
org 018h
jmp Timer1 ; Timer 1 int vector
org 023h
jmp I2C ; I2C serial int vector
org 50h
```

**PowerUp:**

```
;***** Timer 0 Interrupt preparation
clr TR ; disable timer
clr TF ; clear overflow
```

**LCD Initialization Routine**

```
cinit:
clr P3.1 ; RS low
clr P3.2 ; RW low
setb P3.0 ; Enable
mov RTL,#D4100l ; set timer reload value
mov RTH,#D100h ; delay time = 4.1mSec
```

**Subroutine: WRITE**

```
write:
write_loop:
mov a,#0
mova,@a+dptr
cjne a,#99h,write_cont
ret
write_cont:
mov p1,a
acall data_byte
inc dptr
jmp write_loop
```

**Delay Routine:**

```
ddelay:  
setb TR ; start timer
mov IE,#EnableT0 ; enable timer
clr TstFlag ; reset flag
dloop: jnb TstFlag,dloop
ret
```

**Subroutine: command_byte**

```
mov p1,#38h
acall command_byte
acall ddelay ; initial delay 4.1mSec
```

**Subroutine: command_byte**

```
mov p1,#38h ; function set
acall command_byte
mov p1,#0ch ; display on
acall command_byte
mov p1,#01h ; clear display
acall command_byte
acall cgram ; define custom fonts
acall first_line ; display first line
acall second_line ; display second line
```

**SDone:**

```
sdone:
setb IDLE ; power down mode
jmp sdone
```

**Subroutine: command_byte**

```
mov p1,#38h ; function set
acall command_byte
mov p1,#01h ; clear display
acall command_byte
acall first_line ; display first line
acall second_line ; display second line
```

**Subroutine: command_byte**

```
mov p1,#38h ; function set
acall command_byte
mov p1,#01h ; clear display
acall command_byte
acall first_line ; display first line
acall second_line ; display second line
```

**Subroutine: command_byte**

```
mov p1,#38h ; function set
acall command_byte
mov p1,#01h ; clear display
acall command_byte
acall first_line ; display first line
acall second_line ; display second line
```
cgram:
    mov p1,#40h
    acall command_byte
    mov dptr,#cgram_data
    acall write
    ret

; Set DDRAM to the beginning of the first line - 00
;
first_line:
    mov p1,#080h
    acall command_byte
    mov dptr,#fline_data
    acall write
    ret

; Set DDRAM to the beginning of the second line - 40
;
second_line:
    mov p1,#0c0h
    acall command_byte
    mov dptr,#sline_data
    acall write
    ret

; Feed Command/Data to the LCD module

command_byte:
    clr p3.1
    jmp bdelay

data_byte:
    setb p3.1
    nop

bdelay:
    clr p3.2
    clr p3.0
    setb p3.0
    nop

; Check Busy Flag
    mov p1,#00fh
    setb p3.2
    cplt
    setb p3.1
    cplt
    setb p3.0
    cplt
    bloop: nop

; Check busy flag twice
    bwait:
    mov a,p1
    anl a,#80h
    cplt
    bloop: ret

*>Display on a 16x2 Character Module

| Display Character Position and DDRAM Address of a 16x2 Character Module. |
| First Line |  |  |  |  |  |  |  |  |  |  |  |  |  | Display Position |  |  |  |  |
| 00 | 01 | 02 | 03 |   |   |   |   | 0D | 0E | 0F |
| 40 | 41 | 42 | 43 |   |   |   |   | 4D | 4E | 4F |

* Display Character Position and DDRAM Address of a 16x2 Character Module.

* Custom characters memory map.