AVR Freaks

DESIGN NOTE #037

AUTHOR: ALONZO TRUELAND

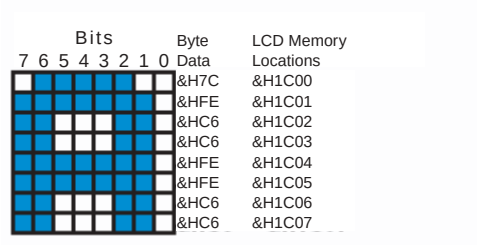
KEYWORDS: TC6963, BASCOM, LCD

Custom Graphic LCD Characters

Graphic displays are great for projects that require both text and pictures to be displayed, but the text can be difficult to read on displays that have a small pixel size. An easy solution is to utilize the custom character generator that is built into the Toshiba tc6963 controller to generate a font that is bolder than the characters generated by the internal CGROM.

The software example was created using the Bascom-Avr compiler. Bascom-Avr has a set of built-in graphic LCD tools that make interfacing painless. The two commands locate(x,y) and LCD are used to display text that is generated by the internal CGROM. The fist step in using custom characters is to create them. Each 8-byte line in the FONTC Data Table represents one custom character. Each Character's Data file is created using the format shown in Figure 1.

Figure 1. Character's Data File



It is easy to understand how the characters are created and stored by looking at Figure 1 and comparing it with the FONTC table. After the data table is built, it must be stored in the LCD Ram memory. The three steps needed to store the characters are set offset, set address pointer, and store characters. These routines access the library functions for the LCD that were included when the CONFIG GRAPHLCD command was invoked. Assembly language is used to interface with the library functions that are also written in assembly. The complier recognizes assembly lingo which makes it easy to include.

Description

- SET OFFSET

  This step is need to set the address location where the characters are stored. The command is &H22 and the data is &H0003. This will set beginning of the custom character storage at memory location &H1C00.

- SET ADDRESS POINTER

  This step sets the pointer to &H1C00. This is where the FONTC Table will be stored.

- STORE CHARACTERS

  This step is responsible for reading the FONTC Table and writing it to the LCD ram starting at the location of the address pointer.

To print the custom characters to the LCD, you first set the cursor position with the locate(x,y). The text to be displayed is stored in the variable Btxt. After setting the cursor and storing the text in Btxt, call the sub routine TEXT with Btxt as the argument. The text to be displayed must be in CAPS. In this example letters A-Z and numbers 1-9 get the bold treatment.

Figure 2. LCD Display, Display From Program

Final Thoughts

This application adds some additional functionally for text display that is easy to implement. The large font is easy to read and is invoked by a simple sub-call. The ability to print both regular and bold text will allow for a more creative interface.

Code

```
;// program that demonstrates the use of user-defined characters //
;// display used is a Toshiba 128*128 //
;!! text passed to sub must be in all caps to work !!
$crystal = 4000000

Dim Gl_byte As Byte
Dim V As Byte
Dim K As Byte
Dim P As Byte
Dim W As Word
Dim Q As String * 1
Dim Btxt As String * 16              ; string to be printed

Declare Sub Text(sf As String )      ; sub that prints the big text

Config Graphlcd = 240 * 128 , Dataport = Portb , Controlport = Portc , Ce = 2
, Cd = 3 , Wr = 0 , Rd = 1 , Reset = 4 , Fs = 5 , Mode = 8
Cls
Cursor Off
Wait 1

; set offset
ldi r24,&h03                         ; load r24 with offset location low byte
rcall _GWrite_Data
ldi r24,&h00                         ; load r24 with offset location high
byte
Rcall _GWrite_Data
ldi r24,&h22                         ; set offset command
rcall _GWrite_Cmd                    ; write command to LCD

; set address pointer
ldi r20,&h00                         ; address that pointer will point to
ldi r21,&h1c                         ; set address pointer command
rcall _set_address                   ; set address routine

; store characters
Restore Fontc                        ; select fontc table
For W = 1 To 288                     ; read each fontc table byte into
Read Gl_byte                         ; read a byte from the table
Loadadr Gl_byte , X                  ; load register x with gl_byte address
ld r24, x                            ; load r24 with gl_byte
rcall _GWrite_Data
ldi r24, &hc0                        ; command to write and increment pointer
rcall _GWrite_Cmd
Next W

; display Character
Locate 2 , 1                         ; set cursor location
```

```
Btxt = "ABCDEFGHIJKLMNOP"            ; set Btext with text to be printed
Call Text(btxt)                      ; call text sub
Locate 3 , 1
Btxt = "QRSTUVWXYZ"
Call Text(btxt)
Locate 4 , 1
Btxt = "0123456789"
Call Text(btxt)

End



;sub used to display characters
 Sub Text(sf As String )             ; !!sf text must be in caps to work!!!!
V = Len(sf)                          ; get length of string
For K = 1 To V                       ; repeat until end of string
Q = Mid(sf , K , 1)                  ; read each character
P = Asc(q)                           ; convert to ascII
If P < 65 Then                       ; adjust ascII code
   P = P + 106
   Else
   P = P + 63
End If
Loadadr P , X                        ; load x with address of p
ld r24,x                             ; load r24 with p
rcall _GWrite_Data                   ; write and increment
ldi r24,&hc0
rcall _GWrite_Cmd
Next K
End Sub

;//////////////////character data ////////////////////////////
Fontc:
Data &H7C , &HFE , &HC6 , &HC6 , &HFE , &HFE , &HC6 , &HC6       ; A
Data &HFC , &HFE , &HC6 , &HFE , &HFE , &HC6 , &HFE , &HFC       ; B
Data &H7E , &HFE , &HC0 , &HC0 , &HC0 , &HC0 , &HFE , &H7E       ; C
Data &HF8 , &HFC , &HC6 , &HC6 , &HC6 , &HC6 , &HFC , &HF8       ; D
Data &HFE , &HFE , &HC0 , &HFE , &HFE , &HC0 , &HFE , &HFE       ; E
Data &HFE , &HFE , &HC0 , &HFC , &HFC , &HC0 , &HC0 , &HC0       ; F
Data &H7E , &HFE , &HC0 , &HDE , &HDE , &HC6 , &HFE , &H7C       ; G
Data &HC6 , &HC6 , &HC6 , &HFE , &HFE , &HC6 , &HC6 , &HC6       ; H
Data &HFE , &HFE , &H38 , &H38 , &H38 , &H38 , &HFE , &HFE       ; I
Data &H7E , &H7E , &H18 , &H18 , &HD8 , &HD8 , &HF8 , &HF8       ; J
Data &HC6 , &HCC , &HD8 , &HF0 , &HF0 , &HD8 , &HCC , &HC6       ; K
Data &HC0 , &HC0 , &HC0 , &HC0 , &HC0 , &HC0 , &HFE , &HFE       ; L
Data &H82 , &HC6 , &HEE , &HEE , &HD6 , &HD6 , &HC6 , &HC6       ; M
Data &HE6 , &HE6 , &HF6 , &HD6 , &HD6 , &HDE , &HCE , &HCE       ; N
```

```
Data &H7C , &HFE , &HC6 , &HC6 , &HC6 , &HC6 , &HFE , &H7C       ; O
Data &HFE , &HFE , &HC6 , &HFE , &HFE , &HC0 , &HC0 , &HC0       ; P
Data &H7C , &HFE , &HC6 , &HC6 , &HC6 , &HCE , &HFE , &H7E       ; Q
Data &HFE , &HFE , &HC6 , &HFE , &HFE , &HF0 , &HD8 , &HCC       ; R
Data &HFE , &HFE , &HC0 , &HFE , &HFE , &H06 , &HFE , &HFE       ; S
Data &HFE , &HFE , &H38 , &H38 , &H38 , &H38 , &H38 , &H38       ; T
Data &HC6 , &HC6 , &HC6 , &HC6 , &HC6 , &HC6 , &HFE , &HFE       ; U
Data &HEE , &H6C , &H6C , &H6C , &H28 , &H38 , &H38 , &H10       ; V
Data &HEE , &HC6 , &HC6 , &HD6 , &HD6 , &H7C , &H6C , &H6C       ; W
Data &HC6 , &H6C , &H28 , &H38 , &H38 , &H28 , &H6C , &HC6       ; X
Data &HC6 , &HC6 , &H6C , &H28 , &H38 , &H38 , &H38 , &H38       ; Y
Data &HFE , &HFE , &H0E , &H1C , &H78 , &HE0 , &HFE , &HFE       ; Z
Data &H3C , &H7E , &H66 , &H76 , &H6E , &H66 , &H7E , &H3C       ; 0
Data &H38 , &H78 , &H98 , &H18 , &H18 , &H18 , &H18 , &H7E       ; 1
Data &H38 , &H6C , &H6C , &H0C , &H18 , &H30 , &H60 , &H7C       ; 2
Data &HFE , &HFE , &H06 , &H7E , &H7E , &H06 , &HFE , &HFE       ; 3
Data &H18 , &H38 , &H78 , &HD8 , &HFE , &HFE , &H18 , &H18       ; 4
Data &H7C , &H60 , &H78 , &H4C , &H0C , &H0C , &H4C , &H38       ; 5
Data &H1C , &H30 , &H60 , &H78 , &H6C , &H6C , &H6C , &H38       ; 6
Data &H7C , &H4C , &H0C , &H0C , &H18 , &H18 , &H18 , &H18       ; 7
Data &H38 , &H6C , &H6C , &H38 , &H6C , &H6C , &H6C , &H38       ; 8
Data &H38 , &H6C , &H6C , &H6C , &H3C , &H0C , &H18 , &H70       ; 9
```