Application Note

AN211

Creating Custom LCD Characters

Introduction

This application note describes the method of creating custom characters on LCD and VFD displays that use the Hitachi HD44780 or a compatible controller. The code described here builds on display functions from **ICOM.LIB**, the Intellicom driver library.

Custom LCD Characters

In addition to the actual display RAM, the widely-used Hitachi HD44780 LCD controller contains enough “character generation” RAM (CGRAM) for eight 5x8 characters. When ASCII values 0-7 are sent to the LCD, the actual character image is taken from the CGRAM instead of the fixed character ROM.

Character Definition

The characters are defined as eight bytes, one for each horizontal row in the 5x8 LCD character. The two lowest five bits in each byte correspond to the five vertical columns in the 5x8 character. The bottommost row will also contain the cursor (if enabled), which should be kept in mind as new characters are defined. Here is an example of a custom character definition:

```
const char plug[8] = {
    0x0A,       // . # . # .
    0x0A,       // . # . # .
    0x1F,       // # # # # #
    0x11,       // # . . . #
    0x11,       // # . . . #
    0x0E,       // . # # # .
    0x04,       // . . # . .
    0x04        // . . # . .  cursor row
};
```

Library Functions

The following function uses the low-level display interface functions in the library **ICOM.LIB** to download a custom LCD character into the display's CGRAM. It takes as input a character number to define (0-7) and an 8-byte array of values.

```
/************************************************************
dispDefineChar

SYNTAX:        int dispDefineChar(int charnum, char values[]);

PARAMETER1:    Character number to define, 0-7.
PARAMETER2:    Custom character bitmap (8 bytes).

DESCRIPTION:   This function loads the character generation RAM of the dis-
play with a custom-defined character.  Each byte in the bitmap corresponds
to one row in the 5x8 character, and only the lowest five bytes are loaded.
NOTE: this function requires functions from ICOM.LIB, the Intellicom driver
library.

RETURN VALUE:  0 if successful, -1 if charnum is invalid.
************************************************************/

int dispDefineChar(int charnum, char values[])
{
    int i;

    if ((charnum < 0) || (charnum > 7))  return(-1);

    // set LCD address to CG RAM
    dispCmd(0x40 + charnum*8);

    // enter character bytes into CG RAM
    for (i=0; i<8; i++) {
        dispData(values[i]);
    }

    // set LCD address back to display RAM
    dispGoto(_dispCol, _dispRow);

    return(0);
}
```

Example Program

The following code defininеs eight custom LCD characters and then displays them.

```
/********************************************************

    lcd_char.c
    2001, Z-World

    Example of using user-defined LCD characters (written
    for the OP6600/6700 Intellicom LCD).

    The standard controller on all LCDs used by Z-World
    allows the user to create up to eight custom
    characters in the character-generation RAM on
    the LCD controller.  This program provides a
    function to define the characters and an example
    of their use.

    After they are defined, the characters can be
    accessed as ASCII values 0 through 7.

 ********************************************************/

#use CUSTOMLCD.LIB  // library containing dispDefineChar

/************************************
    Character definitions
 ************************************/
/*
    The characters are defined in 8-byte arrays, one
    byte per line.  The character size is actually
    only 5x8, so the three highest bits in each byte
    should be set to zero.  Also note that the bottom
    line is normally used by the cursor and should
    probably not be used unless the cursor is turned off.
 */
const char box[8] = {
    0x00,        // . . . . .
    0x1F,        // # # # # #
    0x11,        // # . . . #
    0x11,        // # . . . #
    0x11,        // # . . . #
    0x1F,        // # # # # #
    0x00,        // . . . . .
    0x00         // . . . . . cursor row
};

const char xbox[8] = {
    0x00,        // . . . . .
    0x1F,        // # # # # #
    0x1B,        // # # . # #
    0x15,        // # . # . #
    0x1B,        // # # . # #
    0x1F,        // # # # # #
    0x00,        // . . . . .
    0x00         // . . . . . cursor row
};
```

```
const char plug[8] = {
    0x0A,         // . # . # .
    0x0A,         // . # . # .
    0x1F,         // # # # # #
    0x11,         // # . . . #
    0x11,         // # . . . #
    0x0E,         // . # # # .
    0x04,         // . . # . .
    0x04          // . . # . . cursor row
};

const char battery[8] = {
    0x0E,         // . # # # .
    0x1F,         // # # # # #
    0x11,         // # . . . #
    0x11,         // # . . . #
    0x1F,         // # # # # #
    0x1F,         // # # # # #
    0x1F,         // # # # # #
    0x1F          // # # # # # cursor row
};

const char white_circle[8] = {
    0x00,         // . . . . .
    0x0E,         // . # # # .
    0x11,         // # . . . #
    0x11,         // # . . . #
    0x11,         // # . . . #
    0x0E,         // . # # # .
    0x00,         // . . . . .
    0x00          // . . . . . cursor row
};

const char black_circle[8] = {
    0x00,         // . . . . .
    0x0E,         // . # # # .
    0x1F,         // # # # # #
    0x1F,         // # # # # #
    0x1F,         // # # # # #
    0x0E,         // . # # # .
    0x00,         // . . . . .
    0x00          // . . . . . cursor row
};

const char smiley[8] = {
    0x00,         // . . . . .
    0x0A,         // . # . # .
    0x00,         // . . . . .
    0x04,         // . . # . .
    0x11,         // # . . . #
    0x0E,         // . # # # .
    0x00,         // . . . . .
    0x00          // . . . . . cursor row
};
```

```
const char frowny[8] = {
    0x00,         // . . . . .
    0x0A,         // . # . # .
    0x00,         // . . . . .
    0x04,         // . . # . .
    0x00,         // . . . . .
    0x0E,         // . # # # .
    0x11,         // # . . . #
    0x00          // . . . . . cursor row
};

/************************************
    Main program -- displays chars
 ************************************/


main()
{
    int i;

    brdInit();

    dispDefineChar(0, box);
    dispDefineChar(1, xbox);
    dispDefineChar(2, plug);
    dispDefineChar(3, battery);
    dispDefineChar(4, white_circle);
    dispDefineChar(5, black_circle);
    dispDefineChar(6, smiley);
    dispDefineChar(7, frowny);

    dispPrintf("User-defined chars:\n\n");

    dispPrintf("  %c %c %c %c %c %c %c %c\n",
                    0, 1, 2, 3, 4, 5, 6, 7);
    dispPrintf("  0 1 2 3 4 5 6 7\n");
}
```