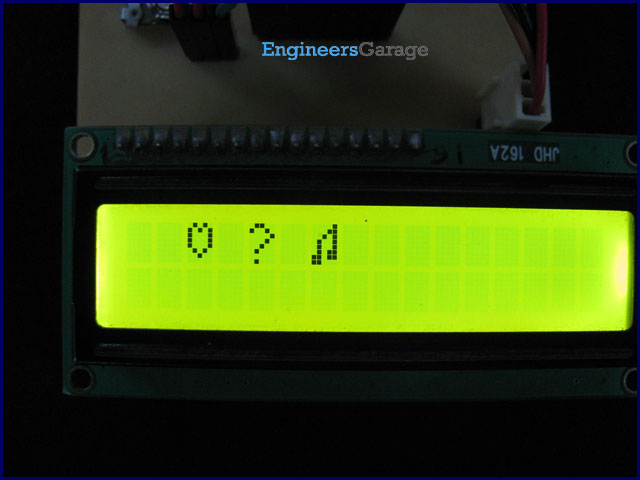
**[How to create custom characters on 16x2 LCD using PIC18F4550](http://www.engineersgarage.com/pic-microcontroller-projects/interfacing-lcd-custom-character-circuit" \o "How to create custom characters on 16x2 LCD using PIC18F4550)**

**Summary**

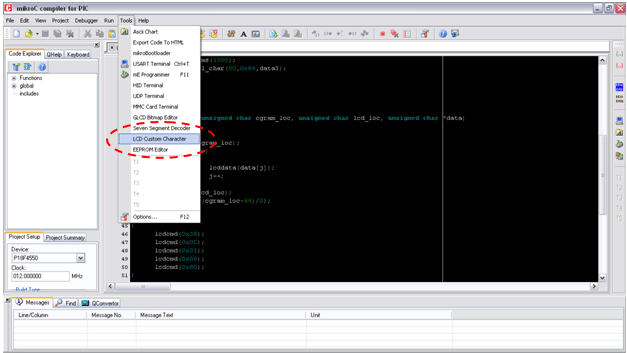


The 16x2 character [LCD](http://www.engineersgarage.com/electronic-components/16x2-lcd-module-datasheet) can also be used to display custom characters other than numerals, alphabets & special characters. Refer [LCD interfacing with PIC](http://www.engineersgarage.com/embedded/pic-microcontroller-projects/interface-lcd-circuit). Some special shapes like hearts, arrows, smileys etc. can easily be displayed on the 5x8 pixel pattern of character LCD. These shapes are first stored at a special location in LCD’s controller and then displayed on the LCD module. This procedure has been explained here by using [PIC18F4550](http://www.engineersgarage.com/electronic-components/pic18f4550-microcontroller).

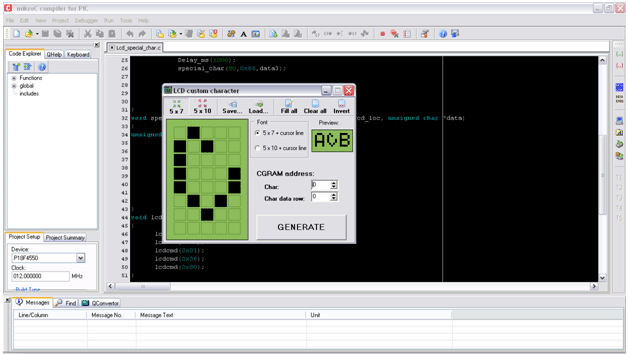
The special characters are generated by bit-mapping of LCD’s 5x8 bit pixel matrix. Refer [Creating custom characters on LCD using 8051](http://www.engineersgarage.com/microcontroller/8051projects/create-custom-characters-LCD-AT89C51-circuit) for more details on bitmap generation and storing custom values in custom generator (CG) RAM of LCD’s controller.

The mikroC IDE provides LCD Custom Character tool to create the bitmap of user defined custom character. (Also see Working with mikroC) To create the bitmaps using this tool, following steps are to be followed:

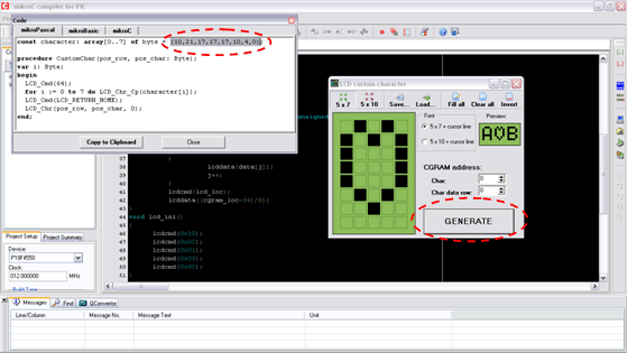
1. Go to Tools -> LCD Custom Character

****

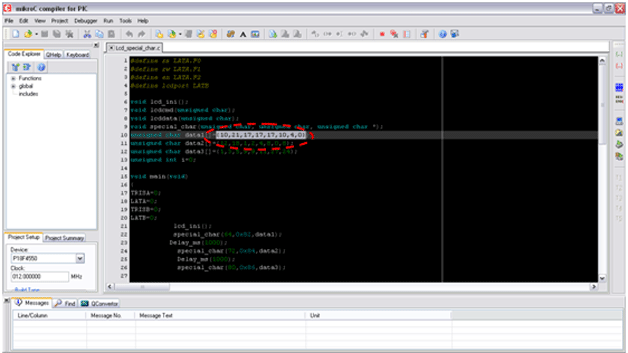
2. Select 5x7 + cursor line font and start filling the pixels in the matrix by clicking on them to create a custom character. The following figure depicts the generation of heart shape’s bitmap.



3. After creating the character and click on GENERATE button. A window will appear containing the bitmap values of designed custom character as highlighted in the following figure.



4. These bitmap values can now be used in the code.



The bitmap values of a custom character are stored in CGRAM of LCD controller module. The CGRAM can store up to 8 custom characters’ bitmaps. For more details, refer Custom character display using 8051. The addresses of CGRAM where bitmaps are stored are shown in the following table.

|  |  |
| --- | --- |
| **ASCII Code** | **Base Address** |
| 0 | 64 |
| 1 | 72 |
| 2 | 80 |
| 3 | 88 |
| 4 | 96 |
| 5 | 104 |
| 6 | 112 |
| 7 | 120 |

The following programming steps are used to store the bitmap values into CGRAM and display the corresponding custom character on LCD. An example is also given at the end to understand the code and concept.

**Programming Steps:**

·            Select the base address of CGRAM where the bitmap values are to be stored. This address is sent as command instruction (RS=0).

·            After the address, the bitmap values are sent one by one as data (RS=1).

·            Next the LCD location is sent where the character is to be displayed.

·            The corresponding ASCII value of the base address of the CGRAM is sent to print the stored character. This is sent as data value (RS=1).

The above steps are integrated into a single function special\_char() which makes it easier to display the custom characters on LCD.

*void special\_char(unsigned char cgram\_loc, unsigned char lcd\_loc, unsigned char \*data)*

*{*

*unsigned int j=0;*

*lcdcmd(cgram\_loc);         // sends location of CGRAM*

*while(j<8)*

*{*

*lcddata(data[j]);      // sends bitmap values of the character*

*j++;*

*}*

*lcdcmd(lcd\_loc);           // sends LCD location where the character is to displayed*

*lcddata((cgram\_loc-64)/8); //ASCII value of corresponding base address.*

*}*

*If a* data array value[] (containing bitmap values) is to be stored at CGRAM location 64 (base address), and is to be displayed at 0x82 location on LCD (i.e., first line, third column); then the above function is called as follows.

*special\_char(64,0x82,value);*

**// Program to display custom characters on 16x2 LCD using PIC18F4550 Microcontroller**  
// Configuration bits  
/\* \_CPUDIV\_OSC1\_PLL2\_1L, // Divide clock by 2  
 \_FOSC\_HS\_1H, // Select High Speed (HS) oscillator  
 \_WDT\_OFF\_2H, // Watchdog Timer off  
 MCLRE\_ON\_3H // Master Clear on  
\*/  
  
//LCD Control pins  
#define rs LATA.F0  
#define rw LATA.F1  
#define en LATA.F2  
  
//LCD Data pins  
#define lcdport LATB  
  
void lcd\_ini();  
void lcdcmd(unsigned char);  
void lcddata(unsigned char);  
void special\_char(unsigned char, unsigned char, unsigned char \*);  
unsigned char data1[]={10,21,17,17,17,10,4,0}; // Bitmap values of "heart" shape  
unsigned char data2[]={12,18,1,2,4,8,0,8};  
unsigned char data3[]={1,3,5,9,9,11,27,24};  
unsigned int i=0;  
  
void main(void)  
{  
 TRISA=0; // Configure Port A as output port  
 LATA=0;  
 TRISB=0; // Configure Port B as output port  
 LATB=0;  
 lcd\_ini(); // LCD initialization  
 special\_char(64,0x82,data1); // Function call to store "Heart" shape's bitmap at 64th base address   
 // and print it at 0x82 location on LCD  
 Delay\_ms(1000);  
 special\_char(72,0x84,data2);  
 Delay\_ms(1000);  
 special\_char(80,0x86,data3);  
}  
  
void special\_char(unsigned char cgram\_loc, unsigned char lcd\_loc, unsigned char \*data)  
{  
 unsigned int j=0;  
 lcdcmd(cgram\_loc); // Send location of CGRAM  
 while(j<8)  
 {  
 lcddata(data[j]); // Send bitmap values of the character  
 j++;  
 }  
 lcdcmd(lcd\_loc); // Send LCD location where the character is to displayed  
 lcddata((cgram\_loc-64)/8); // ASCII value of corresponding base address  
}  
  
void lcd\_ini()  
{  
 lcdcmd(0x38); // Configure the LCD in 8-bit mode, 2 line and 5x7 font  
 lcdcmd(0x0C); // Display On and Cursor Off  
 lcdcmd(0x01); // Clear display screen  
 lcdcmd(0x06); // Increment cursor  
 lcdcmd(0x80); // Set cursor position to 1st line, 1st column  
}  
  
void lcdcmd(unsigned char cmdout)  
{  
 lcdport=cmdout; //Send command to lcdport=PORTB  
 rs=0;   
 rw=0;  
 en=1;  
 Delay\_ms(10);  
 en=0;  
}  
  
void lcddata(unsigned char dataout)  
{  
 lcdport=dataout; //Send data to lcdport=PORTB  
 rs=1;  
 rw=0;  
 en=1;  
 Delay\_ms(10);  
 en=0;  
}

