HD44780 Character LCD Library for STM32 Microcontrollers

Mikrotronics Pakistan | mikro.pk | halroad.org

HD44780 character LCD is one of the most common LCD used by hobbyists and prototype engineering projects. One of reasons behind this is easy interfacing and widespread availability. Driving the LCD needs some LCD specific commands and procedures that are usually implemented as middleware in the form of a Library or a driver. Many popular compilers and integrated development environments usually bundle a library with their package as value added includes. However, many professional type IDEs like CubeIDE for STM32 does not provide such a library with its package. Most developers therefore have to get it from third parties (like this one) or write their own.

How to use the Library.

The library consists of two files:

1. Lcd.c
2. Lcd.h

You need to import or copy these files into your project folder. In STM32CubeIDE project folder the lcd.c file should be imported into src folder:

<your project>\Core\Src

And the lcd.h file into:

<tour project>\Core\Inc

Once these are copied the Library and its functions are accessed by including the lcd.h file in the source code.

LCD Connections

The library supports both 4 bit and 8 bit data modes. The read/Write line is supposed to be connected to GND or through a GPIO pin set to LOW. The library assumes write only functionality. The backlight and LED connections should be managed separately as they are not addressed by the library.

The lcd can be connected to any GPIO lines, and it can be in any order as per requirements of the project or routing. However there is only one constraint that all connections should be on the same GPIO Port.

Setting Up the Library

lcdSetup()

The library needs to call lcdSetup() function once to define the lcd connections. All lcd connections including data and control bits have to be on the same port.

```
void lcdSetup(GPIO_TypeDef * lcdPort, uint16_t RS,uint16_t E,enum
lcdType mode,uint16_t D0,uint16_t D1,uint16_t D2,uint16_t D3,uint16_t
D4,uint16_t D5,uint16_t D6,uint16_t D7);
```

The first parameter is the GPIO port as defined by the HAL libraries in STM32 CubeIDE, like GPIOA or GPIOB etc. RS and E are the pin numbers where RS and Enable pin are connected. Pin numbers are masks as defined by GPIO_HAL libraries (GPIO_PIN_8) describes pin 8 of any GPIO port. Internally its 16 bit mask (1 << 8).

lcdType is an Enumeration defined in lcd.h file Lcd4Bit (defined as 0) to indicate connections in 4 bit mode and Lcd8Bit (defined as 1) for 8 bit data connections. D0 to D7 are the data pins connected to LCD data pins. These can be in any order of convenience, however pins must be mentioned as masks as defined by HAL libraries like GPIO_PIN_0, GPIO_PIN_1 etc. In case of 4 bit mode D0 to D3 are set to 0 and D4 to D7 are set to GPIO pins.

lcdInit()

next function to call is lcdInit() it does not require any parameters and does not return any value. lcdInit() also needs to be called once, however it can be called repeatedly in case you want to reset the lcd any time during program execution.

lcdCommand(*Command*)

This function sends special commands to the lcd to control its behavior. Various commands have been defined in the lcd.h file. These include:

```
#define lcdClear           0x01
#define lcdCursorOFF       0x0C
#define lcdCursorON        0x0E
#define lcdCursorBlink     0x0D
#define lcdCursorHome      0x02
#define lcdShiftRight      0x1C
#define lcdShiftLeft       0x18
#define lcdOFF             0x08
```

The commands are self-explaining. Like lcdClear will clear the data on lcd and bring cursor to home position.

```
void lcdSetCursor(uint8_t row, uint8_t col);
```

lcdSetCursor() function sets the position of cursor, both rows and columns begin from 0.

```
void lcdString(char * string);
```

lcdWriteString() function accepts a string constant, or a null terminated character array. It will display the characters starting from the current cursor position.

```
void lcdWrite(uint8_t data);
```

this function writes a byte sized data to the lcd, in case you want to display the non standard characters defined in the lcd ROM. These special characters differ among manufacturers and have asci codes above 127.

```
void lcdWriteInt(char * format, uint32_t number );
```

most libraries allow only to print strings and rely on the programmer to format integers and numbers into a string to display. We have made a function that accepts an 32 bit integer number and a format as string to format the number.

%d	Signed integer
%02d	Two digits with leading zeros %03d will be three digits number with leading zeros. You can use any number like %03d, %04d, %05d etc
%u	Unsigned decimal
%o	Octal format
%x	Hexadecimal

```
void lcdWriteFloat(char * format, double number );
```

this function is similar to lcdWriteInt() except that it takes floating point umbers and display decimal number. The format string can :

%f	Floating point with standard decimal places
%6.2f	Total number length of 6 with two decimal places xxxxxx.xx %3.1f will show xxx.x
%e	Will display in scientific notation

Example Code:

```
lcdSetup(GPIOA, GPIO_PIN_8, GPIO_PIN_9,lcd4Bit, 0, 0, 0, 0,
GPIO_PIN_4, GPIO_PIN_5, GPIO_PIN_6, GPIO_PIN_7);
lcdInit();

  /* USER CODE END 2 */

  /* Infinite loop */
```

```
  /* USER CODE BEGIN WHILE */
  while (1)
  {
        lcdCommand(lcdClear);
        lcdString("**MIKROTRONICS**");
        lcdSetCursor(1, 3);
        lcdString("LCD Library");
        HAL_Delay(3000);
        lcdCommand(lcdClear);
        lcdString("Int Numbers:");
        for(int i=0; i< 20;i++)
        {
              lcdSetCursor(1, 0);lcdWriteInt("%d", i);
              lcdSetCursor(1, 3);lcdWriteInt("%02d", i);
              lcdSetCursor(1,6);lcdWriteInt("%03d",i-20);
              lcdSetCursor(1, 10);lcdWriteInt("0X%X", i);
              HAL_Delay(500);
        }
        lcdCommand(lcdClear);
        lcdString("Float Cursor OFF");
        lcdCommand(lcdCursorOFF);
        for(float i=0;i<3;i+=0.1)
        {
              lcdSetCursor(1, 0);lcdWriteFloat("%f", i);
              lcdSetCursor(1, 10);lcdWriteFloat("%1.2f",i);
              HAL_Delay(200);
        }
        lcdCommand(lcdClear);
        lcdString("Screen Shift");
        lcdSetCursor(1, 0);lcdString("A Quick Brown Fox Jumps over the
Lazy Dog");
        HAL_Delay(500);
        for(int i=0;i<32;i++)
        {
              lcdCommand(lcdShiftLeft);
              HAL_Delay(300);
        }
```