5. Alphanumeric LCD

The use of an alphanumeric LCD display with 16 characters, two lines is demonstrated.

5.1. Hardware – a commonly used 2X16 LCD

It is often necessary to present results of calculations to a user, and a liquid crystal display (LCD) may come handy. There are many different LCDs available; a simple LCD capable of presenting two lines with 16 characters in each line is used here. The display implements a standard bus and related datasheet and examples of driving such display can easily be found on the web. The LCD is connected to the microcontroller using a four-bit data bus and two-bit control bus. Additionally, power supply and contrast setting lines increase the number of wires at the connector to 10, all is available at connector K350, see Fig. 2.2.3.

Lower four pins of port D (0 to 3) are used to drive the four-bit data bus for the display, and pins 6 and 7 of port D are used to drive the control lines named Enable (E) and Register Select (RS). Since it is not planned to read from the LCD the control line Read/Write (RW), used to determine the direction of data transfer, is permanently connected to GND allowing only writing into the LCD.

The contrast of the characters displayed at the LCD depends of the position of trimmer potentiometer P350.

5.2. Software – the include file “LCD2x16.c”

The use of the LCD in a user program starts by its initialization, which must be performed at the beginning of the execution. Later the LCD can be used to display any alphanumerical symbols, strings of symbols or integer numbers of different lengths. A set of functions to utilize the display has been prepared, and is available in an include file named “LCD2x1616.c”. The functions include:

- LCD_init: This function is called at the beginning of the program to configure port D, lower four plus additional two pins, and the LCD itself using a four-bit interface. The function requires no parameters and returns no values.
- LCD_string: This function displays a string of characters (maximum 16 characters in length) at the LCD at the desired position, and accepts two parameters. The first is the string to be written (for instance “ABCD”), the second is the position at the LCD where the string should start. The first position in the upper line is indexed as 0x00, the last position in the first line is indexed as 0x0f. The first position in the second line is indexed as 0x40, the last position in the second line is indexed as 0x4f due to the hardware of the LCD used.
- LCD_uInt16: The function is used to display a 16-bit unsigned integer number at the LCD. The function requires three parameters. The first parameter is the unsigned integer to be displayed, the second parameter fixes the position on the LCD (as for the LCD_string function), and the third parameter is either 0 or 1. If 0 the number is shown including leading zeroes, if 1 the leading zeroes are removed. In both cases the number is right aligned.

- LCD_sInt16: The same as LCD_uInt16, but capable of displaying a 16-bit signed integer (15 bits and sign).
- LCD_uInt32: The same as LCD_uINT16 but capable of displaying a 32-bit unsigned integer number. Also requires three parameters (maybe not complete 32 bits, check the software).
- LCD_sInt32: The same as LCD_uInt32, but capable of displaying a 32-bit signed integer (31 bits & sign, maybe not complete 31 bits, check the software).
- LCD_sInt3DG: The same as LCD_uInt16, but capable of displaying only 3 digits.
- LCD_sInt4DG: The same as above, but capable of displaying 4 digits.

The file “LCD2x16.c” must be included at the beginning of the user program before the first call to any function from it. This is done by a “#include “LCD2x16.c” ” statement. In all integer displaying functions a conversion from binary to binary-coded-decimal in performed following the well-known algorithm from the web (search keyword: “shift and add 3 algorithm”).

The demo program to write a simple message (imitation of the measurement result) is given below.

```
#include "stm32f4xx.h"
#include "stm32f4xx_rcc.c"
#include "stm32f4xx_gpio.c"
#include "dd.h"
#include "LCD2x16.c"

void main (void){
int counter = 0;

  LCD_init();                                  // init LCD
  LCD_string("LCD test", 0x04);                // display title string

  while (1) {
    LCD_uInt16(counter++,0x40,0x01);    // write to LCD, unsigned, 16 bits
    //LCD_sInt16(counter++,0x40,0x01);    // write to LCD, signed, 16 bits
    //LCD_uInt32(counter++,0x40,0x01);    // write to LCD, unsigned, 32 bits
    //LCD_sInt32(counter++,0x40,0x01);    // write to LCD, signed, 32 bits
    //LCD_sInt3DG(counter++,0x40,0x01);   // write to LCD, unsigned, 3 digits
    for (int i = 0; i<1000000; i++) {}; // waste some time
  };
}
```

The program starts with the familiar set of four include statements, and the fifth include statements is added to allow the use of LCD functions.

The “main” function is executed, and a variable ‘counter’ is declared and initialized, then port D and the alphanumeric LCD module are configured and initialized by calling the function “LCD_init()”. Following the initialization a simple string is displayed using function “LCD_string()”, then the execution continues with the endless loop, where the content of the variable ‘counter’ is repeatedly displayed in the second line of the display. An empty for loop is added to slow down the execution of the loop.

A note: an LCD module can interpret messages sent from the microcontroller only if a well-defined protocol and timing is followed by the microcontroller. The protocol is fixed in functions enclosed in file “LCD2x16.c”. The timing associated with sending messages to the LCD module is implemented by executing empty for loops, where the number of iterations defines the delay. These delays were thoroughly tested for different LCD modules and optimized for speed, but might not be adequate for all LCDs. The resulting screen is either empty or garbled. If timing is too fast then it can be slowed down by increasing the numbers given in the define statements at the beginning of file “LCD2x16.c”, lines 5 to 7.