5. Alphanumeric LCD

It is often necessary to present results of calculations to user. A liquid crystal display (LCD) may come handy to do it. There are many different LCDs available; a simple LCD capable of presenting two lines with 16 characters in each line is planned to be used here. It implements a standard bus and related datasheet and examples of driving software 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 K350, see Fig. 1.

Lower four bits of port C (PC00 to PC03) are used to drive the four-bit data bus, and next two bits of port C (PC04, PC05) are used to drive the control lines named Register Select (RS) and Enable (E). Since it is not planned to read from the LCD the control line Read/Write (RW), used to distinguish the direction of data transfer, is permanently connected to GND allowing only writing into the LCD.

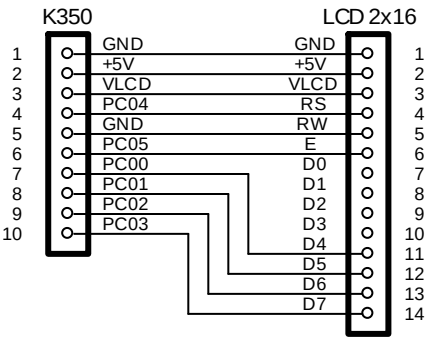

Figure 1: The connection of the liquid crystal display using a four but data-bus and two-bit control bus

The contrast of the characters displayed at the LCD depends of the position of trimmer potentiometer P350. The use of the LCD in a program starts by the initialization, which must be performed at the beginning of the program. Later the LCD can be used to display any alphanumerical symbols, strings of symbols or integer numbers of different lengths. A set of functions is available, it consists of:

- LCD_init: This function is called at the beginning of the program to initialize the port C, lower six bits and the LCD 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. The function 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.
- 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 itself, the second parameter is the position on the LCD (as for the LCD_string), the third parameter is

either 0 or 1. If 0 the number is shown including leading zeros, if 1 the leading zeros are erased.

- LCD_sInt16: The same as LCD_uInt16, but capable of displaying a 16-bit signed integer (15 bits & 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).

Please note that these functions are combined into a file that must be included into the user source code. This is done by a "#include "LCD2x16.c" " statement at the beginning of every program utilizing the LCD. 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 in Fig. 2, and the corresponding photo of the LCD is given in Fig. 3.

```
#include "stm32f4xx.h"
#include "LCD2x16.c"                    // include file with LCD functions

int main (){
unsigned int i = 0;                     // declare & init a variable i

  LCD_init();                           // initialize the LCD

  LCD_string("Demo", 0x03);             // write a text into top line
  LCD_string("x=", 0x41);               // write a text into bottom line
  LCD_string("mV", 0x48);               // write a text into bottom line

  while (1) {
    LCD_uInt16(i++,0x43,1);             // write a number into bottom line
    i &= 0x7fff;                        // limit the value of the number
  };
}
```

Figure 4: A listing of the program to write a result in the second line on the LCD

Figure 5: The photo of the LCD as a result of running the program in Fig. 4