4. The Liquid Crystal Display module of the Dragon12-Plus2 board

4.1 The purpose of the work

On the Dragon12-Plus2 board are available two display circuits, namely 7-segment displays and the LCD module. This laboratory session aims to develop fundamental concepts associated with connecting and programming the Liquid Crystal Display (LCD) module.

4.2 Bitwise operators. Creating and working with bit patterns (masks)

C programing language provides six operators for bit manipulations. These operators may be applied only to integral operands, which are: *char, short, int,* and *long*, whether they are *signed* or *unsigned*. The bitwise operators are presented in Table 4.1.

&	AND
|	OR
^	XOR
~	NOT
>>	right shift
<<	left shift

Table 4.1 Bitwise operators

- The *&* operator is often used to *clear* one or more bits to 0.
    - Example 1: PORTB = PORTB & 0xAA; /*PORTB is 8 bits*/
        - Will clear the even bits from PORTB to 0;
        - 0xAA is called a *mask;*
    - Example 2: PTT = PTT & 0xFD; /*0xFD is 11111101 in binary*/
        - Will clear bit 1 of PTT;
        - 0xFD in this case is called a *mask;*
        - It is equivalent to: PTT = PTT & ~0x02;
    - General rule:
        - & with 1 will keep the value, & with 0 will always put that particular bit to 0 (will clear it).
- The | operator is often used to set one or more bits to 1.
    - Example 1: PORTB = PORTB | 0xAA; /* PORTB is 8 bits */

        - Sets the odd bits of PORTB to 1;

    - Example 2: PTT = PTT | 0x02;
        - Set bits 6 and 1 of PTT to 1;

    - General rule:
        - | with 0 keeps the value, | with a 1 is always 1.

- The *XOR* (^) operator can be used to toggle one or multiple bits

    - Example 1: var1 = var1 ^ 0xF0;
        - Toggles the upper 4 bits of the variable *var1*.

    - General rule:
        - ^ with a 0, will preserve that value, ^ with a 1 will flip the value.

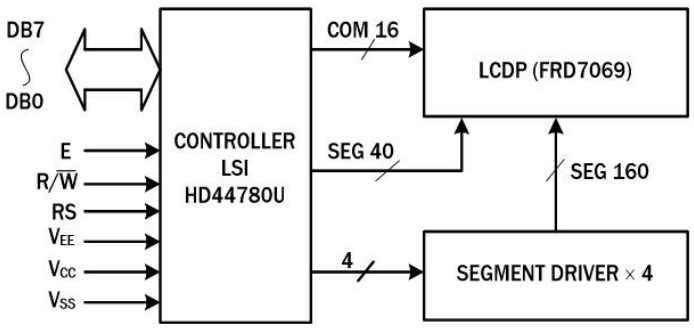
- The << left shift operator, will implement a left shift with number of bits in second operand, while the vacated bits will be filled with zeros; leftmost bits will be lost.

    - Example: PTT = PTT << 4;
        - Shift lower 4 bits of PTT to upper 4 places

- The >> right shift operator, will right shift with number of bits in second operand, while vacated bits will be filled with zero if the number is unsigned or nonnegative; otherwise it will usually repeat the sign bit.
    - Example: var1 = var2 >> 3;
        - Will shift the variable *var2* to the right three places and assigns it to the variable *var1*.

4.3 LCD module presentation

The Dragon12 development board has a 16 character x 2 lines Liquid Crystal Display (LCD) module. Each character is displayed using a 5 x 7 dot matrix. The most common type of



LCD allows the light to pass through when activated. An LCD segment is activated when a low frequency bipolar signal in the range of 30 Hz to 1000 Hz is applied to it, being able to display characters and graphics. The Hitachi HD44780 [8] is one of the most popular character-based LCD controller being used (fig. 4.1). This controller is used to convert the ASCII character to 5 x 7 bit image. The pins DB7~DB0 are used to exchange data with the CPU, E is the enable signal and should be connected to one of the address decoder output or I/O pin, the RS signal selects instruction register (if is equal to 0) or data register (in this case it should be 1), while the V_{EE} signal allows the user to adjust the LCD contrast (Table 4.2).

Fig. 4.1 Block diagram of a HD44780U-based LCD module

Pin Number	Symbol	I/O	Function
1	V_{SS}	-	Power supply (GND)
2	V_{DD}	-	Power supply (+5V)
3	V_{EE}	-	Control contrast
4	RS	I	0 = instruction input, 1 = data input
5	R/W	I	0 = write to LCD, 1 = read from LCD
6	E	I/O	Enable signal
7	DB0	I/O	Data bus line 0
8	DB1	I/O	Data bus line 1
9	DB2	I/O	Data bus line 2
10	DB3	I/O	Data bus line 3
11	DB4	I/O	Data bus line 4
12	DB5	I/O	Data bus line 5
13	DB6	I/O	Data bus line 6
14	DB7	I/O	Data bus line 7

Table 4.2 Pin assignement for the display

The HD44780 has a display data RAM (DDRAM) to store data to be displayed on the LCD. The HD44780 has a character generator ROM that can generate character patterns from an 8-bit code. The user can rewrite character patterns into the character generator RAM (CGRAM). Up to eight 5 x 8 patterns or four 5 x 10 patterns can be programmed [8].

The HD44780 has two 8-bit user accessible registers: instruction register (IR) and data register (DR). To write data into display data RAM or character generator RAM, the microcontroller writes into the DR register. The address of the data RAM should be set up with a previous instruction. The DR register is also used for data storage when reading data from DDRAM or CGRAM. The HD44780 has a busy flag that is output from the DB7 pin. The HD44780 uses a 7-bit address counter to keep track of the address of the next DDRAM or CGRAM location to be accessed.

The general-purpose I/O Port K of the MC9S12 microcontroller is also used for the control of the LCD module. The interface can be 4 bits or 8 bits. An example of 4-bit interface is shown in fig 4.2. To read or write the LCD successfully, one must satisfy the timing requirements of the LCD as presented in the datasheet [9].

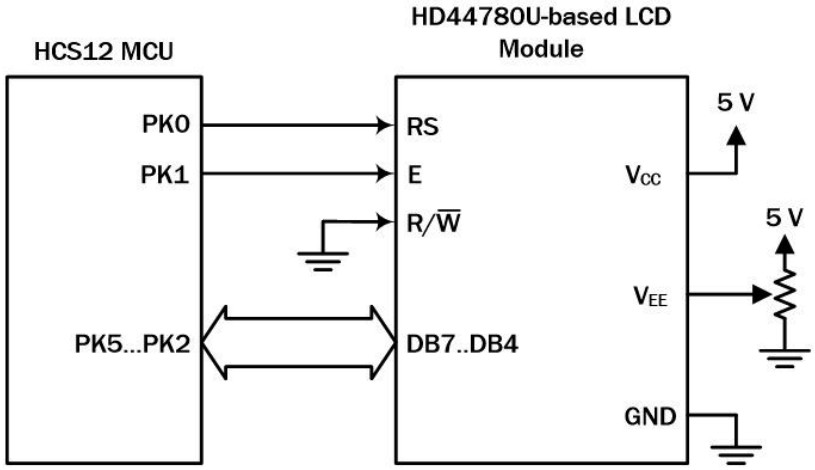


Fig 4.2 LCD interface example (4-bit bus, used in Dragon12-Plus2 board)

According to [3], the pinouts of J11 (the portion from Dragon12 board where the LCD is placed) and J12 are as follows:

Pin 1 GND
Pin 2 VCC (5V)
Pin 3 Via a 220 Ohm resistor to GND
Pin 4 PK0 RS pin for LCD module
Pin 5 PK7 R/W pin for LCD module
Pin 6 PK1 EN pin for LCD module
Pin 7 Not used
Pin 8 Not used
Pin 9 Not used
Pin 10 Not used
Pin 11 PK2 DB4 pin for LCD module
Pin 12 PK3 DB5 pin for LCD module
Pin 13 PK4 DB6 pin for LCD module
Pin 14 PK5 DB7 pin for LCD module
Pin 15 Via a 22 Ohm resistor to VCC LED backlight for LCD module
Pin 16 GND

From the manual [3], we notice that pins PK2-PK5 (not PK4-PK7) are used to drive half of the LCD data bus: DB4 – DB7.

You can send commands or data to the controller chip to control the LCD display [9]. The commands are:

- Clear display and return cursor to home (upper left)
- Cursor home (don't clear display)
- Entry mode (move cursor left or write)
- Display on/off (turns display on or off, cursor on or off, cursor blink)
- Cursor/display shift (move cursor or shift display, which direction)
- Function set - bus data width (8 or 4), number of display lines (1 or 2), font size

(6x8 or 5x7)

- Set CG RAM address (set address of CG (Character Generation) RAM to generate your own characters)
- Set DD RAM Address (set address of DD (Data Display) RAM to display characters)
- Read busy flag and address (we can't use)
- Write data to DD RAM or CG RAM
- Read data from DD RAM or CG RAM (we can't use)

The LCD command codes to be used and sent to LCD are presented in Table 4.3.

Code (hex)	Command to LCD Instruction Register
1	Clear display and return cursor to home (upper left)
2	Cursor home (don't clear display)
4	Shift cursor to left (decrement cursor)
5	Shift display right
6	Shift cursor to right (increment cursor)
7	Shift display left
8	Display off, Cursor off
A	Display off, Cursor on
C	Display on, cursor off
E	Display on, cursor blinking
F	Display on, cursor blinking
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift the entire display to the left
1C	Shift the entire display to the right
80	Force cursor to beginning of 1st line
C0	Force cursor to beginning of 2nd line
38	2 lines, 5x7 matrix (D0-D7, 8-bit)
28	2 lines, 5x7 matrix (D4-D7, 4-bit)

Figure 4.3 LCD Command Codes

The LCD display can use either 8-bit or 4-bit data bus. The Dragon12 board uses a 4-bit bus, so it takes two transfers to send one command. The Dragon12 board is set so that you cannot read from the display; you can only write to it. When you write a command, you need to wait until the command has be executed by the LCD controller [9]. The Busy Flag (from Read Busy Flag command) tells when the command is done. We cannot read Busy Flag, so we have to wait specified time before proceeding.

To write to the controller, the following steps must be done:

1. Set RS (PK0) to 0 for command, 1 for data
2. Set R/W (On Dragon12, R/W tied low for write only)
3. Put 4 MSB on Port K bits 5-2
4. Bring E (PK1) high for at least 230 ns
5. Bring E (PK1) low
6. Put 4 LSB on Port K bits 5-2
7. Bring E (PK1) high for at least 230 ns
8. Bring E (PK1) low

9. Wait specified amount of time for execution to complete

The function for sending commands to the LCD, *COMWRT4*, is depicted below. The input parameter is represented by an unsigned command byte. In order to send it to the LCD, the high nibble is aligned to port PK5-PK2 and the command control signals are activated, followed by the same operations applied to the low nibble.

```
void COMWRT4(unsigned char command)
 {
     unsigned char x;

     x = (command & 0xF0) >> 2;              //shift high nibble to center of byte for PK5-PK2
     LCD_DATA =LCD_DATA & ~0x3C; //clear bits Pk5-Pk2
     LCD_DATA = LCD_DATA | x;          //sends high nibble to PORTK
     MSDelay(1);
     LCD_CTRL = LCD_CTRL & ~RS;      //set RS to command (RS=0)
     MSDelay(1);
     LCD_CTRL = LCD_CTRL | EN;         //raise enable
     MSDelay(5);
     LCD_CTRL = LCD_CTRL & ~EN;      //Drop enable to capture command
     MSDelay(15);                              //wait
     x = (command & 0x0F)<< 2;              // shift low nibble to center of byte for PK5-PK2
     LCD_DATA =LCD_DATA & ~0x3C; //clear bits PK5-PK2
     LCD_DATA =LCD_DATA | x;            //send low nibble to PORTK
     LCD_CTRL = LCD_CTRL | EN;         //raise enable
     MSDelay(5);
     LCD_CTRL = LCD_CTRL & ~EN;      //drop enable to capture command
     MSDelay(15);
 }

      void DATWRT4(unsigned char data)
 {
 unsigned char x;

       x = (data & 0xF0) >> 2;
     LCD_DATA =LCD_DATA & ~0x3C;
     LCD_DATA = LCD_DATA | x;
     MSDelay(1);
     LCD_CTRL = LCD_CTRL | RS;
     MSDelay(1);
     LCD_CTRL = LCD_CTRL | EN;
     MSDelay(1);
     LCD_CTRL = LCD_CTRL & ~EN;
     MSDelay(5);
```

```
    x = (data & 0x0F)<< 2;
    LCD_DATA =LCD_DATA & ~0x3C;
    LCD_DATA = LCD_DATA | x;
    LCD_CTRL = LCD_CTRL | EN;
    MSDelay(1);
    LCD_CTRL = LCD_CTRL & ~EN;
    MSDelay(15);
 }
```

Task 4.1: Analyze the function *void DATWRT4(unsigned char data)* by commenting each line of the function and taking into consideration all the parameters. Please take a look at the function: *void COMWRT4(unsigned char command).*

Task 4.2: Using the functions provided above, write the C code of the main function, in order to display the message "Hello world!" on the LCD display. The reset sequence of the LCD module is included in the *main* function, as depicted below.

- Explain which is the meaning of the hex values sent as parameters to the COMWRT4 function.
- This exercise will have to call each time the function DATWRT4

```
void main(void)
{
    DDRK = 0xFF;
    COMWRT4(0x33);   //reset sequence provided by data sheet
    MSDelay(1);
    COMWRT4(0x32);   //reset sequence provided by data sheet
    MSDelay(1);
    COMWRT4(....);   //Function set to four bit data length
                          //2 line, 5 x 7 dot format
    MSDelay(1);
    COMWRT4(....);  //entry mode set, increment, no shift
    MSDelay(1);
    COMWRT4(....);  //Display set, display on, cursor on, blink off
    MSDelay(1);
    COMWRT4(....);  //Clear display
    MSDelay(1);
    COMWRT4(0x80);  //set start position, home position
    MSDelay(1);
    …
}
```

Task 4.3: Implement a function *void putsLCD(unsigned char *ptr)* that will display a *string* on to the LCD.

- Prove the correct functioning of this function by displaying the text: "ES lab"

Task 4.4: Write a program that displays all 26 uppercase letters of the alphabet (ABC...Z) on the LCD panel, starting in the first position of the first line. Note:

- Please do not hard-code the letters only by calling *void DATWRT4(unsigned char data)* function! Use the function *void putsLCD(unsigned char *ptr)* from Task 2.3.3 in order to display the letters.
- You'll probably find that the letters **A** through **P** appear on the first line, but nothing appears on the second line. That's because the LCD does not automatically wrap to the second line when the first line is full. Rather, it keeps trying to display letters to the right of the panel's right edge. To see the letters **Q** through **Z** on the second line, you'll need to reposition the cursor at the right time.

Task 4.5: Modify your program so that the letters are displayed in uppercase if the leftmost DIP switch on the Dragon12 is set HIGH, but the letters are displayed in lowercase if the switch is set LOW. The display should change if the user changes the switch setting while the program is running.