JIEE Časopis priemyselnej elektrotechniky
Journal of Industrial Electrical Engineering
ISSN 2454-0900
www.jiee.eu

Character LCD display driver design to the STM32 microcontrollers

[1]*Ľubomír ŠOGANIČ*, [2]*Patrik JACKO*

[1,2]Department of Theoretical and Industrial Electrical Engineering,

Technical University of Košice, Slovakia

[1]lubomir.soganic@student.tuke.sk, [2]patrik.jacko.2@tuke.sk

***Abstract* — At present, programming is at a very high level, resulting in a large number of support functions. Therefore, it is not possible for the programmer to do all the functions in the source code himself. Libraries thus help the programmer to simplify and speed up programming, which leads to comfortable programming. Libraries are used in almost all programming languages. This article is focused on the creation of a driver and a library for character LCD displays controlled by STM32 microcontrollers.**

***Keywords* — driver, library, character LCD displays, LCD display, STM32F446RE**

I. Introduction

For testing of functionality of a library, that were made for microcontroller STM32F446RE to control a character LCD displays, we need make a test device. This test device consists of main panel. The main panel consist of microcontroller STM32F446RE, two kinds of character LCD displays which can communicate with 4-bit or 8-bit communication, Bluetooth module HC06, CANON9 connector, DC connector, switch and additional module for setting a contrast of displays and for choosing between displays. First LCD display has 2 lines and 16 columns and second display has 4 lines and 20 columns. Bluetooth module is used for communication between computer and microcontroller. Switch is used to choose a USB supply or external supply voltage in 6V to 12V range, which we can connect with DC connector [1]. For connecting LCD display with microcontroller are used these pins: PB1, PB2, PB3, PB5, PB8, PB10, PB13, PB14, PB15 and PA9. Pin PC13 is used to blink an internal LED when ENABLE pin is activated during debugging. The program for using a LCD display is composed of the libraries: LCD_display.h, Clock_config.h and Peripheral_config.h and files. They have same names as libraries.

II. Schematic diagram and description of additional modules

Additional module (Fig. 1) which is in the main panel is used for switching between LCD displays that communicate through 4-bit or 8-bit communication and for setting a contrast of these displays. If it appears log.1 value on a pin PA0, the transistor T1 will open and switch on a relay KM1, that connect the 4x20 characters display on a supply voltage. If it appears log.1 value on a pin PA1, the transistor T2 will open and switch on a relay KM2, that connect the 2x16 characters display on a supply voltage. We can set up a contrast of the LCD displays with potentiometers R1 and R2. Transistors T3 and T4 are not used in this module. These transistors are planned to use to adjust or switch the brightness of the LCD displays through microcontroller. The transistors T1 and T2 are protected by diodes D1 and D2 before reverse voltage.

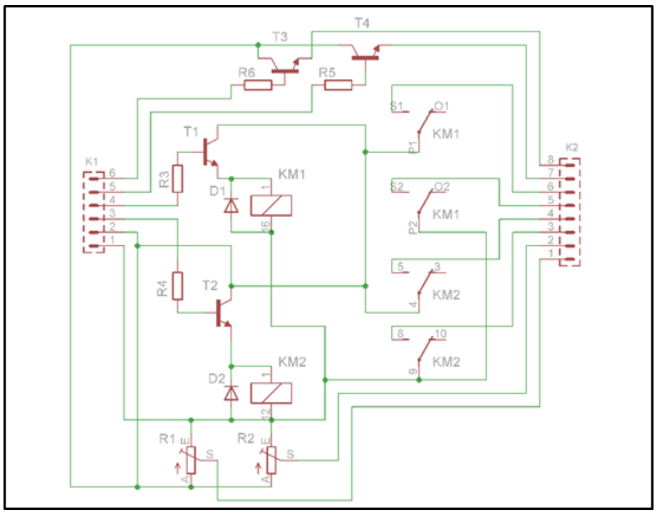


Fig. 1 Additional module in main panelA

III. SOFRWARE

A. Libraries

The LCD_display.h library, have 10 main functions to driving a LCD display. The Clock_config.h library have only one function for setting a clock for peripherals. It names a clock_config(). We created a Peripheral_config.h library foe setting used peripherals for communication between computer and microcontroller via USB or via Bluetooth and between I2C LCD display and microcontroller via I2C communication.

B. Timers setting

First step for using full parameters of STM32 microcontrollers is right setting of timers. We create a Clock_config.c file for these settings. This file is composed of setting up the external clock generator and right setting of dividers for the peripherals that we use [2].

C. Peripherals setting

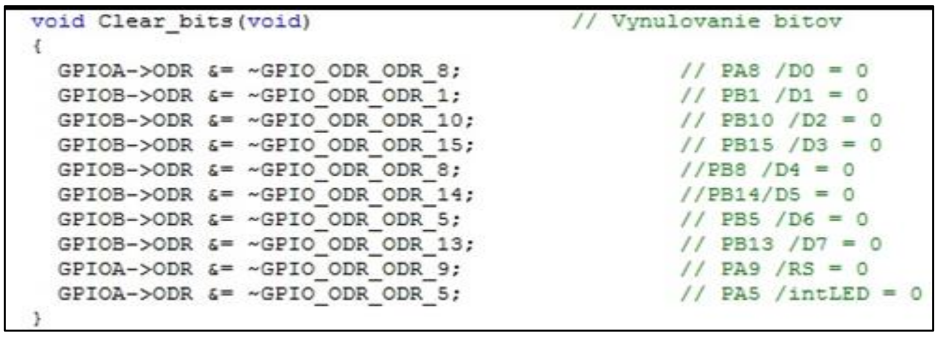
Peripheral_config.c file consist of three functions. They are USART2_config(), USART6_config() and I2C1_config(). The function USART2_config() is used for setting a communication between computer and microcontroller via USB COM port [2].

The function USART6_config() is used for setting a communication between computer and microcontroller via Bluetooth technology.

The function I2C1_config() is used for setting a communication between I2C display and microcontroller via I2C technology.

IV. SUBFUNCTIONS OF THE LCD DISPLAY

The LCD_display.c file have a 24 subfunctions. There is GPIOA_conf(), GPIOB_conf(), GPIOC_conf() which configures the necessary parameters and allows the timers of the ports whose pins we use. Next functions are Clear_bits(), delay(int time), EN(), Set_8_bit(), Entry_mode_8(), Disp_cur_bli_on_8(), Disp_cur_on_8(), Disp_on_8(), Display_clear_8(), Shift_line_8(), Shift_char_8(), Cursor_home_8(). The function Clear_bits() (Fig. 2) is a simple function, which clear all the pins after we call this function. We are using this function after we send “Enable” bit or before restarting the displays so that random information does not affect their correct start [4].



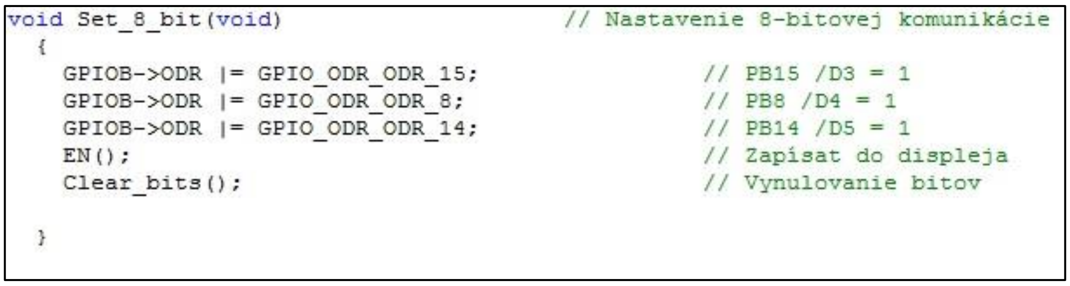
```
void Clear_bits(void)                      // Vynulovanie bitov
 {
   GPIOA->ODR &= ~GPIO_ODR_ODR_8;                 // PA8 /D0 = 0
   GPIOB->ODR &= ~GPIO_ODR_ODR_1;                 // PB1 /D1 = 0
   GPIOB->ODR &= ~GPIO_ODR_ODR_10;                // PB10 /D2 = 0
   GPIOB->ODR &= ~GPIO_ODR_ODR_15;                // PB15 /D3 = 0
   GPIOB->ODR &= ~GPIO_ODR_ODR_8;                 //PB8 /D4 = 0
   GPIOB->ODR &= ~GPIO_ODR_ODR_14;                //PB14/D5 = 0
   GPIOB->ODR &= ~GPIO_ODR_ODR_5;                 // PB5 /D6 = 0
   GPIOB->ODR &= ~GPIO_ODR_ODR_13;                // PB13 /D7 = 0
   GPIOA->ODR &= ~GPIO_ODR_ODR_9;                 // PA9 /RS = 0
   GPIOA->ODR &= ~GPIO_ODR_ODR_5;                 // PA5 /intLED = 0
 }
```

Fig. 2 Function Clear_bits()

The confirmation of the information by the "Enable" pin is provided by the EN() function, which starts with delay of 4ms duration and then sets the "Enable" pin to the log.0 value, turn on the built-in LED then resets the acknowledgment pin to log.1 and turn off the built-in LED [3]. During debugging, the delay() function was also called between the transition of the acknowledgment pin from log.0 to log1 to make the LED flashing visible. After debug the program, we canceled this delay by entering delay function to the comment.

All subfunctions except GPIOA_conf(), GPIOB_conf(), GPIOC_conf(), Clear_bits(), delay(int time), and EN() have 8-bit version and 4-bit version. For example function Set_8_bit() have a Set_4_bit() equivalent. The reason of this solution is communication with the character LCD display with 8-bit communication and 4-bit communication [3].

To communicate with a character LCD, first you need to set the type of communication. The first type described communication is an 8-bit communication that can be set using the Set_8_bit() function (Fig. 4). The pins D0 to D7 set the required information, so the value of log.1 is set on pins D3 to D5, followed by the function EN(), which saves the entered information into the display. At the end of this function is the Clear_bits() subfunction, which sets all pins (except the EN pin) to log.0 [4].



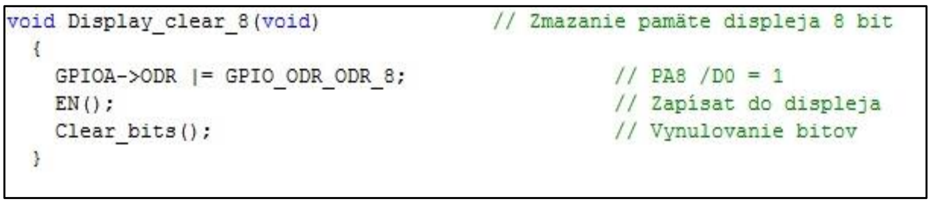
```
void Set_8_bit(void)                       // Nastavenie 8-bitovej komunikácie
  {
    GPIOB->ODR |= GPIO_ODR_ODR_15;                  // PB15 /D3 = 1
    GPIOB->ODR |= GPIO_ODR_ODR_8;                   // PB8 /D4 = 1
    GPIOB->ODR |= GPIO_ODR_ODR_14;                  // PB14 /D5 = 1
    EN();                                           // Zapísat do displeja
    Clear_bits();                                   // Vynulovanie bitov

  }
```

Fig. 3 Function Set_8_bit()

The Set_4_bit() function works in the same as Set_8_bit() function. But it sets 4-bit communication and the information is first divided into two parts. The next step is to send the first part of the information to the pins for which the acknowledgment bit is sent and reset the pins, after the confirmation. Subsequently, the second part of the information is set, the acknowledgment bit is sent, and all pins are reset. All other 4-bit functions work similarly.

Functions Disp_cur_bli_on_8(), Disp_cur_ on_8() and Disp_on_8() setting a mode of showing a cursor on the display. The function Disp_cur_bli_on8() set up a cursor with blinking. The function Disp_cur_on_8() set up only cursor without blinking. The last function which sets a cursor is a function Disp_on_8(). This function not show the cursor. This functions have a difference only in one bit [4].

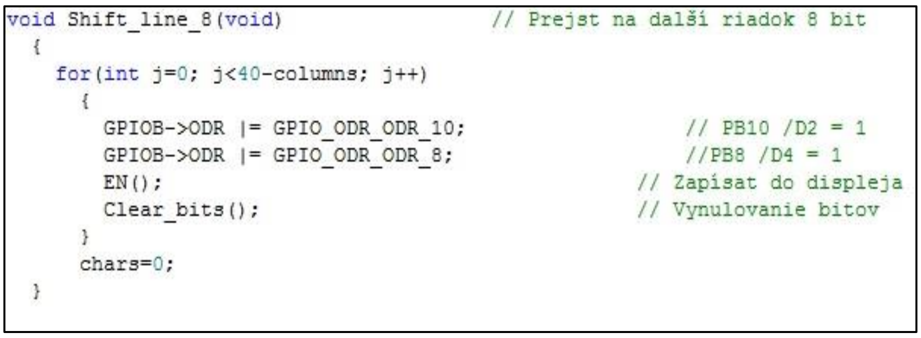
The Display_clear_8() (Fig. 4) function is used to a clear all characters from display and from memory. This function automatically set the cursor to first position of first line.



```
void Display_clear_8(void)            // Zmazanie pamäte displeja 8 bit
  {
    GPIOA->ODR |= GPIO_ODR_ODR_8;               // PA8 /D0 = 1
    EN();                                       // Zapísat do displeja
    Clear_bits();                               // Vynulovanie bitov
  }
```

Fig. 4 Function Display_clear_8()

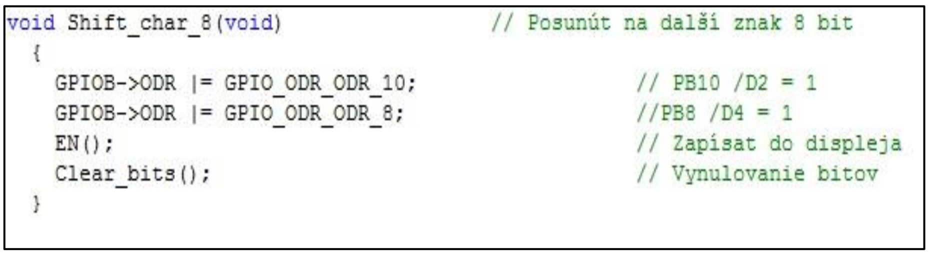
If we write the last character to line we need shift cursor to the next line because the next characters is saved to memory, but not showed on the display when we are using a 2x16 character LCD display. When we using a 4x20 character LCD display and we write the last character to first line, display automaticcally shifts the cursor to third line and we need set the cursor to second line [5]. For this setting we created a Shift_line_8() function. This function is showed in the Fig. 7.



```
void Shift_line_8(void)               // Prejst na další riadok 8 bit
  {
    for(int j=0; j<40-columns; j++)
      {
        GPIOB->ODR |= GPIO_ODR_ODR_10;             // PB10 /D2 = 1
        GPIOB->ODR |= GPIO_ODR_ODR_8;              //PB8 /D4 = 1
        EN();                                    // Zapísat do displeja
        Clear_bits();                            // Vynulovanie bitov
      }
      chars=0;
  }
```

Fig. 5 Function Shift_line_8()

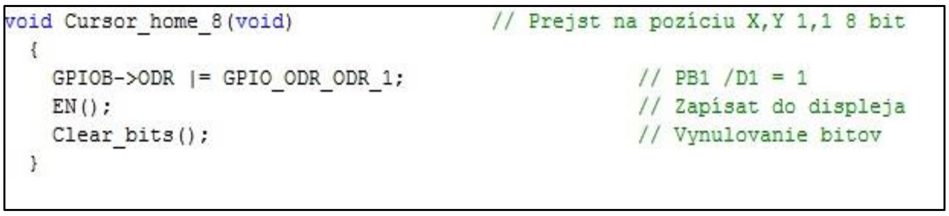
If we want to set the cursor to another position of cursor then first position we call the Shift_char_8() (Fig. 6) function. This function is called in main function GotoXY(int x, int y).



```
void Shift_char_8(void)               // Posunút na další znak 8 bit
  {
    GPIOB->ODR |= GPIO_ODR_ODR_10;               // PB10 /D2 = 1
    GPIOB->ODR |= GPIO_ODR_ODR_8;                //PB8 /D4 = 1
    EN();                                        // Zapísat do displeja
    Clear_bits();                                // Vynulovanie bitov
  }
```

Fig. 6 Code of Shift_char_8() function

For set the cursor to first position of first line we use a function Cursor_home_8(). This function sets the cursor to the home position but don't clear the characters from display and from memory [4]. Code of Cursor_home_8() function is showed on Fig. 7.



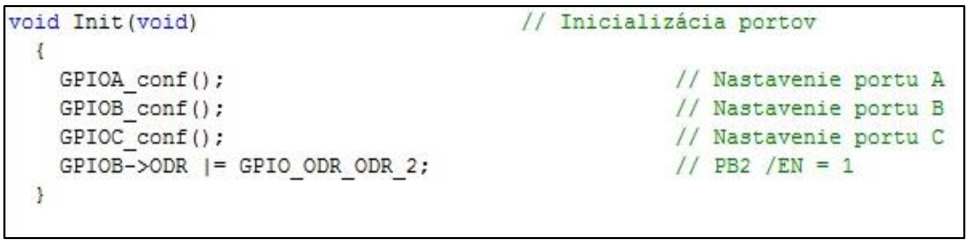
```
void Cursor_home_8(void)              // Prejst na pozíciu X,Y 1,1 8 bit
  {
    GPIOB->ODR |= GPIO_ODR_ODR_1;                // PB1 /D1 = 1
    EN();                                        // Zapísat do displeja
    Clear_bits();                                // Vynulovanie bitov
  }
```

Fig. 7 Code of Cursor_home_8() function

V. MAIN FUNCTIONS OF LCD_DISPLAY

We created a 9 main functions to driving a display. The names of this functions is: Init(), Comunication_type(int com), Dusplay_type(int rowsa, int columnsb), LCD_on(int number), LCD_start(int cursor, int blink), Clean(), LCD_reset(), GotoXY(int x, int y) and LCD_print(char Text[64]).

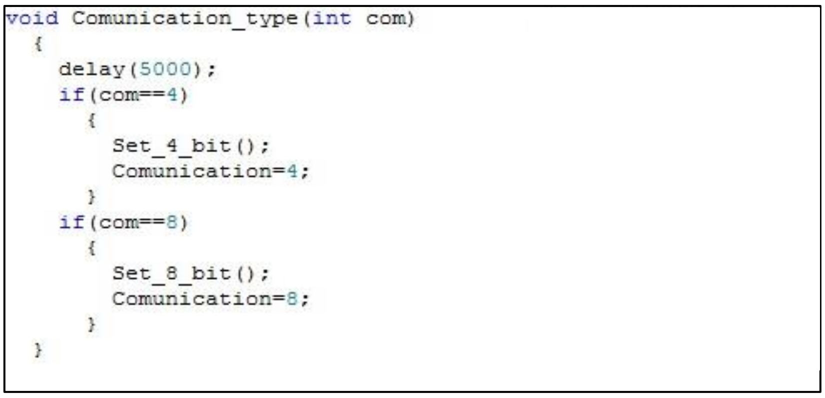
Function Init() (Fig. 8) calling the subfunctions GPIOA_conf(), GPIOB_conf(), GPIOC_conf() and set enable pin to log.1. This function set all used pins to driving a LCD display expect I2C communication.



```
void Init(void)                           // Inicializácia portov
  {
    GPIOA_conf();                                 // Nastavenie portu A
    GPIOB_conf();                                 // Nastavenie portu B
    GPIOC_conf();                                 // Nastavenie portu C
    GPIOB->ODR |= GPIO_ODR_ODR_2;                 // PB2 /EN = 1
  }
```

Fig. 8 Function Init()

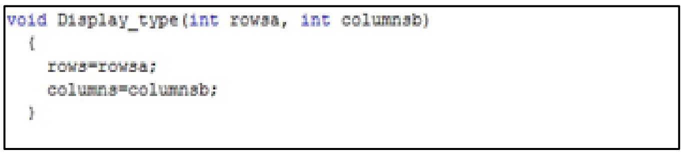
Communication_type(int com) (Fig. 9) using an Set_4_bit() or Set_8_bit() subfunctions for set up the 4 - bit communication or 8 - bit communication. When we want to set up 4 - bit communication we must write number 4 to variable com. When we want to set up 8 - bit communication we must write number 8 to variable com. If we set up variable com to 4 the function automatically set up variable Comunication to 4 and when we set up variable com to 8 the function set up variable Comunication to 8. Variable Comunication is used in the next functions to recognise a communication type.



```
void Comunication_type(int com)
  {
    delay(5000);
    if(com==4)
      {
        Set_4_bit();
        Comunication=4;
      }
    if(com==8)
      {
        Set_8_bit();
        Comunication=8;
      }
  }
```

Fig. 9 Communication_type(int com)

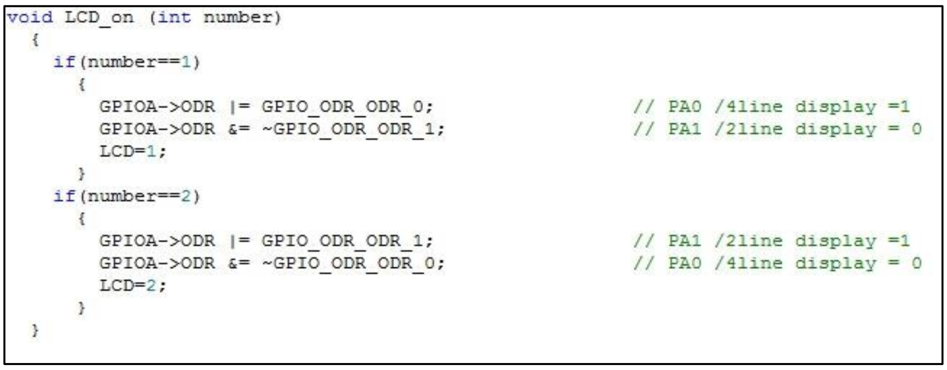
Function Display_type(*int rowsa, int columnsb*) load value from the variables rowsa and columnsb and write these values to variables rows and columns. Variables rows and columns is used in next functions as a variable Comunication. Code of this function is showed in Fig. 10.



```
void Display_type(int rowsa, int columnsb)
  {
    rows=rowsa;
    columns=columnsb;
  }
```

Fig. 10 Code of Display_type(int rowsa, int columnsb)

Function LCD_on(int number) (Fig. 11) switching the LCD displays with variable number. Variable number gains value 1 or 2. When the value of the variable number is 1, function activate the 4x20 character display (2x16 is turned off). If the value of the variable number is 2, function activate the 2x16 character display (4x20 is turned off). For switching the displays, we are using relays that are switched by NPN transistors connected to separate pins PA0 and PA1.

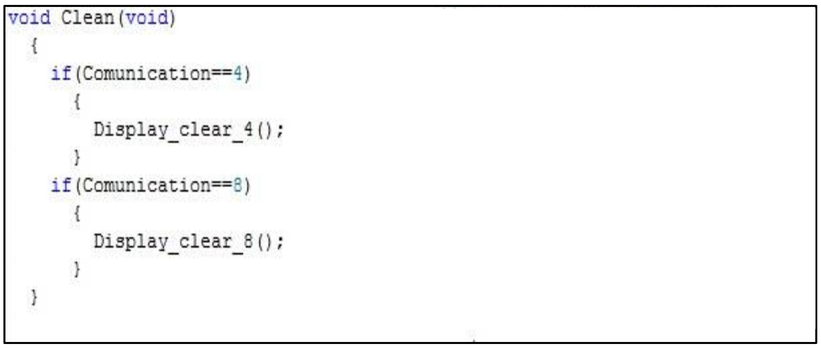


```
void LCD_on (int number)
  {
    if(number==1)
      {
        GPIOA->ODR |= GPIO_ODR_ODR_0;              // PA0 /4line display =1
        GPIOA->ODR &= ~GPIO_ODR_ODR_1;             // PA1 /2line display = 0
        LCD=1;
      }
    if(number==2)
      {
        GPIOA->ODR |= GPIO_ODR_ODR_1;              // PA1 /2line display =1
        GPIOA->ODR &= ~GPIO_ODR_ODR_0;             // PA0 /4line display = 0
        LCD=2;
      }
  }
```

Fig. 11 Function LCD_on(int number)

The function LCD_start(int cursor, int blink) uses the variable Comunication that defines whether the communication is 4-bit or 8-bit. This function using an Entry_mode_4() when the value of Comunication is 4 or Entry_mode_8() when the value of Comunication is 8. If the variable cursor value is 0, the function calling a Disp_on_4() or Disp_on_8() subfunctions. If the variable cursor value is 1, the function calling a Disp_cur_on_4() or Disp_cur_on_8() subfunctions and when the cursor value is 1 and the blink value is 1 function calling a Disp_cur_bli_on_4 or Disp_cur_bli_on_8(), which turn on cursor blinking [4].

To delete all the characters shown on the display, we can choose the simple function Clean() (Fig. 12) which calls the Display_clear_4 () or Display_ clear_8 () subfunctions according to the value of the variable Comunication. The function also set the cursor to the first column of the first column. Communication type memory and cursor settings remain unchanged [4].



```
void Clean(void)
  {
    if(Comunication==4)
      {
        Display_clear_4();
      }
    if(Comunication==8)
      {
        Display_clear_8();
      }
  }
```

Fig. 12 Code of Clean() function

The function Reset() is used to the restart display before setting up new communication type. If the value of the LCD = 1, function turn off the 4x20 character display and set all the data and control pins to log.0 value include the enable pin. The delay(int time) function is called to 0,5 seconds. Next step of Reset() function is set the EN pin to log.1 value and restore the power for LCD display. The same procedure is for the value of the LCD = 2, but with the difference that the 4x20 character display is restarted. The data and control pins are set to log.0, because if they set to the random value, the display will not lose memory and the restart will not have effect.

The function GotoXY(int x, int y) is used to set the cursor to own position. After setting the cursor to the first position of the first line, a cycle moves the cursor by one character up to the value of the variable y set by the user. This cycle counts up the value of a variable chars that moves the cursor to the next line if it reaches the number of columns of the used display. In this
cycle, it first verifies whether it is a 4-bit or 8-bit function so that the correct subfunctions can be called.

At the beginning of the function LCD_print (char Text [64]), is verified the value of the variable Comunication which gets value 4 or 8. If the value is 4, is used the 4-bit communication function. The next step of this function is created a cycle that loads the length of the Text variable, and this cycle is repeated as many times as the length of the Text variable. At the beginning of the cycle, variable b is set to 8. This variable is also used to count the bits of that letter. The next step is load the letter from the

Text variable and create its numeric value in the ASCII table, so that its type changes from character to integer. However, this value is stored in a float bin variable. Next, the character number is stored in the bin_i variable, which is already an integer type. These variables were created because if we want to change the decimal value to binary, we need to divide it by two, and if we divide the integer value with the integer value we would not get the decimal number. The previous sentence suggests that the next step is the splitting cycle, which is repeated eight times, because 8 bits is required to set the 8-bit character. At the beginning of this cycle, the value of variable bin is divided by two and the result is stored back into the variable bin which is char type. After this division, the new value is stored in the bin_i (integer) variable, and the value bin_i is then taken from the bin value and the result is stored in the variable which is also of the char type. In the previous step, the integer is based on the decimal number, so if you divide the decimal point number, it is now stored in the remainder of the variable, which is then tested for its zero or nonzero, based on the determination of the bit logical value. 0 or log.1. After setting the bit, the variable b is subtracted at -1 and the cycle is repeated on the lower bit. When all 8 bits have been set, the bit setting cycle is terminated and the character is printed on the display, followed by reading the next character from the Text[64] variable. It works in the same way for 8-bit communication but uses the 8-bit functions.

VI. CONCLUSION

The article is focused to the character LCD display driver used by the STM32 microcontroller. The driver has default pin configuration where user can connect LCD display to the pins and use that. There was created number of functions, where user can used the 4-bit or 8-bit parallel communication. The extended version of the driver can by additional I2C peripheral communication performed by I2C to parallel communication module determined to the character LCD displays.

REFERENCES

[1]. "UM1724 User manual". [online]. [cit. 2018-05-08]. Available on internet: <http://www.st.com/content/ccc/resource/technical/document/user_manual/98/2e/fa/4b/e0/82/43/b7/DM00105823.pdf/files/DM00105823.pdf/jcr:content/translations/en.DM00105823.pdf>.

[2]. "RM0390 Reference manual". [online]. [cit. 2018-05-08]. Available on internet: <http://www.st.com/content/ccc/resource/technical/document/reference_manual/4d/ed/bc/89/b5/70/40/dc/DM00135183.pdf/files/DM00135183.pdf/jcr:content/translations/en.DM00135183.pdf>.

[3]. "Znakové LCD displeje - časť 1.". [online]. [cit. 2018-05-08]. Available on internet: <http://www.mikrozone.sk/pluginy/content/content.php?content.73>.

[4]. "HD44780U (LCD-II)". [online]. [cit. 2018-05-08]. Available on internet: <https://www.sparkfun.com/datasheets/LCD/HD44780.pdf>.

[5]. "20 x 4 Character LCD". [online]. [cit. 2018-05-08]. Available on internet: <https://www.vishay.com/docs/37314/lcd020n004l.pdf>.