**Lesson 63**

**Part 1**

**Module LCD 16 × 2. MENU**

In the  [**last lesson**](http://narodstream.ru/stm-urok-61-modul-lcd-16x2-rabotaem-s-knopkami/) on the LCD module with buttons, we did a great job learning how to track not only the status of pressing or pressing the buttons, but also the moment when this status changes and implemented it on our display with buttons.

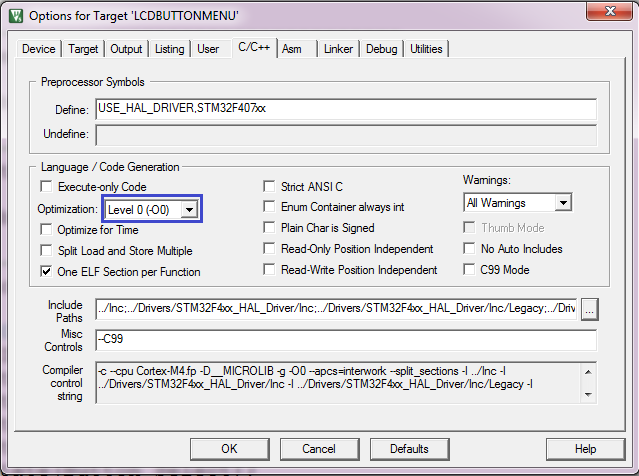
Today we will try, using the experience of past studies, somehow it all structured and, still, to compose some kind of navigation or menu. And there we'll see what happens.

The project was made from the previous project  **LCDBUTTONMODE** and named **LCDBUTTONMENU** .

We will launch a new project in **Cube MX** and, without changing anything and without touching it, we will generate a project for Keil and open it.

Set up the programmer for autocut, connect the files button.c and lcd.c to the project tree and collect the project.

We disable the optimization and try again, after assembling the project, to flash the controller. To make sure. that everything works



Add one more global variable to main.c

volatile uint16\_t adctim\_cnt = 0;   
**volatile uint16\_t menutim\_cnt = 0;**

We will need it to blink the menu item.

Add the code to the ADC handler

 if (adctim\_cnt> 1000)   
 {   
  adctim\_cnt = 0;   
**if (menutim\_cnt> 400)   
  {   
   menutim\_cnt = 0;   
   HAL\_GPIO\_TogglePin (GPIOD, GPIO\_PIN\_12);   
  }   
  menutim\_cnt ++;**  
  TIM\_Callback ();   
 }

While we blink green LED to estimate the speed of flashing. We take into account that the points will flash twice faster.

We will collect the project, we will impose the controller and we will look.

Now we will create and add the files **menu.c** and  **menu.h** to the project in the  most usual way. Connect the file  **menu.h** to **main.h**

#include "button.h"   
**#include "menu.h"**

Also connect the file **menu.h** to the file **menu.c** , and also add a pair of variables

#include "menu.h"   
// ----------------   
**extern uint16\_t menutim\_cnt; // counter for flashing   
static uint8\_t i = 0;**

Contents of the file menu.h

**#ifndef MENU\_H\_   
#define MENU\_H\_   
// ----------------   
#include "stm32f4xx\_hal.h"   
#include "stdint.h"   
// ---------- ------   
#endif / \* MENU\_H\_ \* /**

Connect it to the library display and buttons

#include "stdint.h"   
**#include "lcd.h"**

**#include "button.h"**

Add a function in the **menu.c** file

static uint8\_t i = 0;   
// ----------------   
**void MenuProcess (void)   
{   
  HAL\_Delay (50);   
}**

Add for this function a prototype in the header file

#include "stdint.h"   
// ----------------   
**void MenuProcess (void);**  
// ----------------

Remove the delay from the infinite loop of the **main ()** main function , while the rest of the code in the body of this loop is commented out, since. come in handy. We also call in the infinite loop the new function just written

  while (1)   
  {   
**MenuProcess ();**  
// adc\_value = ADC\_Data; // enter the result of the transformations in the variable

Remove the declaration of a variable in **main** ()

~~uint8\_t i = 0;~~

We move the array **str1** and correct it in the menu.c file a little, and delete it in the source

static uint8\_t i = 0;   
// ----------------   
**char str1 [6] [11] =   
{   
  "MM\_1 \ 0",   
  "MM\_2 \ 0",   
  "MM\_3 \ 0",   
  "MM\_4 \ 0 ",   
  " MM\_5 \ 0 ",   
  " EXIT \ 0 "   
};**  
// ----------------

In the menu.h file, add two structures to remember the state of our program (such as which menu we are in)

#include "stdint.h"   
// ----------------   
**typedef enum {   
  MENU\_STATE\_IDLE = 0,   
  MENU\_STATE\_WAIT,   
  MENU\_STATE\_MAIN,   
} MENU\_StateTypeDef;**

**enum {typedef   
  MAIN\_MENU\_STATE\_IDLE = 0,   
  MAIN\_MENU\_STATE\_WAIT,   
  MAIN\_MENU\_STATE\_MM1,   
  MAIN\_MENU\_STATE\_MM2,   
  MAIN\_MENU\_STATE\_MM3,   
  MAIN\_MENU\_STATE\_MM4,   
  MAIN\_MENU\_STATE\_MM5,   
  MAIN\_MENU\_STATE\_EXIT,   
  MAIN\_MENU\_STATE\_MM1\_WAIT,   
  MAIN\_MENU\_STATE\_MM2\_WAIT,   
  MAIN\_MENU\_STATE\_MM3\_WAIT,   
  MAIN\_MENU\_STATE\_MM4\_WAIT,   
  MAIN\_MENU\_STATE\_MM5\_WAIT,   
  MAIN\_MENU\_STATE\_EXIT\_WAIT   
} MAIN\_MENU\_StateTypeDef;**  
// ----------------

Add two more initialized variables to the **menu.c** file

"EXIT \ 0"   
};   
**MENU\_StateTypeDef menu\_state = MENU\_STATE\_IDLE;   
MAIN\_MENU\_StateTypeDef main\_menu\_state = MAIN\_MENU\_STATE\_IDLE;**

In the **menu.c** file, add the following code to the **MenuProcess** function , which tracks the current state of the program

void MenuProcess (void)   
{   
**switch (menu\_state)   
  {   
    case MENU\_STATE\_IDLE: // start the program   
    break;   
    case MENU\_STATE\_WAIT: // wait for the start of the main menu   
    break;   
    case MENU\_STATE\_MAIN: // launch main menu   
    break;   
  }**  
  HAL\_Delay (50);

Add the code to the first case

  case MENU\_STATE\_IDLE: // start the program   
**menu\_state = MENU\_STATE\_WAIT;   
   LCD\_Clear ();   
   LCD\_SetPos (0,0); // position the cursor   
   LCD\_String ("MENU");   
   LCD\_SetPos (0,1);   
   LCD\_String ("Press SELECT");**  
   break;

You can build a project, flash it in the MK and check the display of these lines.

In the [**next part of**](http://narodstream.ru/stm-urok-63-modul-lcd-16x2-menyu-chast-2/) our lesson we will continue to work with the following cases of this function, and also start writing another very interesting function that will manage all the menu items.

**Lesson 63**

**Part 2**

# Module LCD 16 × 2. MENU

In the [**previous part of**](http://narodstream.ru/stm-urok-63-modul-lcd-16x2-menyu-chast-1/) our lesson, we created the project, configured it, created the program management function and filled it with one case.

Well, we will continue.

We will connect one more global variable in menu.c

extern uint16\_t menutim\_cnt; // counter for flashing items   
**extern uint8\_t button\_state [5];**

Add the code to the 2nd case

  case MENU\_STATE\_WAIT: // wait for the main menu to start   
**if (button\_state [Button\_Select] & ST\_UNPRESSURE)   
   {   
    SetButtonState (Button\_Select, ST\_LOCKED);   
    ResetButtonState (Button\_Select, ST\_UNPRESSURE);   
    SetButtonState (Button\_Select, ST\_PRESSURE);   
    // processing the button press   
    menu\_state = MENU\_STATE\_MAIN;   
   }**  
   break;

Also remove all local variables from the main function main ()

int main (void)   
{   
  / \* USER CODE BEGIN 1 \* /   
  / \* USER CODE END 1 \* /

Once again, we will collect and sew the code.

Let's start now to implement the functionality of the buttons.

Create another function in the menu.c file

**// ----------------   
void MainMenuProcess (void)   
{   
  while (1)   
  {   
    switch (main\_menu\_state)   
    {   
    case MAIN\_MENU\_STATE\_IDLE: // start the main menu   
      break;   
    case MAIN\_MENU\_STATE\_WAIT: // We are waiting for the menu   
      break;   
    case MAIN\_MENU\_STATE\_MM1\_WAIT: // waiting for 1   
      break;   
    case MAIN\_MENU\_STATE\_MM2\_WAIT: // waiting for the choice of 2 points   
      break;   
    case MAIN\_MENU\_STATE\_MM3\_WAIT: // wait for the choice of 3 points   
      break;   
    case MAIN\_MENU\_STATE\_MM4\_WAIT: // waiting for the choice of 4 points   
      break;  
    case MAIN\_MENU\_STATE\_MM5\_WAIT: // wait for the choice of 5 points   
      break;   
    case MAIN\_MENU\_STATE\_EXIT\_WAIT: // waiting for the choice of the item EXIT   
      break;   
    case MAIN\_MENU\_STATE\_MM1: // select item 1   
      break;   
    case MAIN\_MENU\_STATE\_MM2: // select item 2   
      break;   
    case MAIN\_MENU\_STATE\_MM3: // select item 3   
      break;   
    case MAIN\_MENU\_STATE\_MM4: // select item 4   
      break;   
    case MAIN\_MENU\_STATE\_MM5: // select item 5   
      break;   
    case MAIN\_MENU\_STATE\_EXIT: // select the item EXIT   
      break;   
    }   
  }   
}**

Add a prototype for it in the same file from the top

static uint8\_t i = 0;   
// ----------------   
**void MainMenuProcess (void);**  
// ----------------

Call this function in the previous function:

  case MENU\_STATE\_MAIN: // start the main menu   
**main\_menu\_state = MAIN\_MENU\_STATE\_IDLE;   
   MainMenuProcess ();**  
   break;

Fill the first case in **MainMenuProcess**

case MAIN\_MENU\_STATE\_IDLE: // start the main menu   
**LCD\_SetPos (0,0);   
   for (i = 0; i <3; i ++)   
   {   
    LCD\_String (str2 [i]);   
   }   
   LCD\_SetPos (0,1);   
   for (i = 0; i <3; i ++)   
   {   
    LCD\_String (str2 [i + 3]);   
   }   
   HAL\_Delay (50);   
   ResetButtonState (Button\_Select, ST\_LOCKED);   
   main\_menu\_state = MAIN\_MENU\_STATE\_WAIT;**  
   break;

We will collect the code, we will sew the controller and check the menu entry by the button "SELECT"

Fill in the 2nd case

   case MAIN\_MENU\_STATE\_WAIT: // We are waiting for the menu selection   
**if (button\_state [Button\_Right] & ST\_UNPRESSURE)   
   {   
    SetButtonState (Button\_Right, ST\_LOCKED);   
    ResetButtonState (Button\_Right, ST\_UNPRESSURE);   
    SetButtonState (Button\_Right, ST\_PRESSURE);   
    // processing the   
    HAL\_Delay button (50);   
    ResetButtonState (Button\_Right, ST\_LOCKED);   
    main\_menu\_state = MAIN\_MENU\_STATE\_MM1\_WAIT;   
   }**  
   break;

Now write a function that will make the selected menu item flash

// ----------------   
**// blinking menu item   
void BlinkItem (uint8\_t ind)**

**{  
 if (ind <3) LCD\_SetPos (ind \* 5,0);   
 else LCD\_SetPos ((ind-3) \* 5.1);   
 if (menutim\_cnt> 200) LCD\_String (str2 [ind]);   
 else LCD\_String ("");   
}**

At the top in the same file, add a prototype for it

void MainMenuProcess (void);   
**void BlinkItem (uint8\_t ind);**

Write the code in the next case

   case MAIN\_MENU\_STATE\_MM1\_WAIT: // wait for 1 choice of   
**BlinkItem (0);   
   // right key   
   if (button\_state [Button\_Right] & ST\_UNPRESSURE)   
   {   
    SetButtonState (Button\_Right, ST\_LOCKED);   
    ResetButtonState (Button\_Right, ST\_UNPRESSURE);   
    SetButtonState (Button\_Right, ST\_PRESSURE);   
    // processing the   
    HAL\_Delay button (50);   
    ResetButtonState (Button\_Right, ST\_LOCKED);   
    main\_menu\_state = MAIN\_MENU\_STATE\_MM2\_WAIT;   
    LCD\_SetPos (0,0);   
    LCD\_String (str2 [0]); // light 1 point, and then suddenly it's in extinct state   
   }   
   // left key   
   if (button\_state [Button\_Left] & ST\_UNPRESSURE)   
   {  
    SetButtonState (Button\_Left, ST\_LOCKED);   
    ResetButtonState (Button\_Left, ST\_UNPRESSURE);   
    SetButtonState (Button\_Left, ST\_PRESSURE);   
    // processing the   
    HAL\_Delay button (50);   
    ResetButtonState (Button\_Left, ST\_LOCKED);   
    main\_menu\_state = MAIN\_MENU\_STATE\_EXIT\_WAIT;   
    LCD\_SetPos (0,0);   
    LCD\_String (str2 [0]); // light 1 point, and then suddenly it is in extinct state   
   }   
   // down key   
   if (button\_state [Button\_Down] & ST\_UNPRESSURE)   
   {   
    SetButtonState (Button\_Down, ST\_LOCKED);   
    ResetButtonState (Button\_Down, ST\_UNPRESSURE);   
    SetButtonState (Button\_Down, ST\_PRESSURE);  
    // processing the   
    HAL\_Delay button (50);   
    ResetButtonState (Button\_Down, ST\_LOCKED);   
    main\_menu\_state = MAIN\_MENU\_STATE\_MM4\_WAIT;   
    LCD\_SetPos (0,0);   
    LCD\_String (str2 [0]); // light 1 point, and then suddenly it is in an extinct state   
   }   
   // SELECT key   
   if (button\_state [Button\_Select] & ST\_UNPRESSURE)   
   {   
    SetButtonState (Button\_Select, ST\_LOCKED);   
    ResetButtonState (Button\_Select, ST\_UNPRESSURE);   
    SetButtonState (Button\_Select, ST\_PRESSURE);   
    // processing the   
    HAL\_Delay button (50);   
    ResetButtonState (Button\_Select, ST\_LOCKED);   
    main\_menu\_state = MAIN\_MENU\_STATE\_MM1;  
    LCD\_SetPos (0,0);   
    LCD\_String (str2 [0]); // light 1 point, or suddenly it's in an extinct state   
   }**  
   break;

We will collect the code, we will sew the controller and see the response to the right button after selecting the main menu.

We will work with other cases of this function in the [**next part of**](http://narodstream.ru/stm-urok-63-modul-lcd-16x2-menyu-chast-3/) our lesson.

**Lesson 63**

**Part 3**

**Module LCD 16 × 2. MENU**

In the [**previous part of the**](http://narodstream.ru/stm-urok-63-modul-lcd-16x2-menyu-chast-2/) lesson we already started working with the reaction to the items on our menu and wrote the code in one of the function cases of the main menu items

Add the code in the following four similar cases

**MAIN\_MENU\_STATE\_MM2\_WAIT**

**MAIN\_MENU\_STATE\_MM3\_WAIT**

**MAIN\_MENU\_STATE\_MM4\_WAIT**

**MAIN\_MENU\_STATE\_MM5\_WAIT**

We will collect the project, we will sew the controller and see the result.

All items normally switch and flash, well of course except for **EXIT**



We will write the code in the next case - to wait for pressing the exit button from the main menu

 case MAIN\_MENU\_STATE\_EXIT\_WAIT: // wait for the item to be selected EXIT   
**BlinkItem (5);   
// right key   
if (button\_state [Button\_Right] & ST\_UNPRESSURE)   
{   
SetButtonState (Button\_Right, ST\_LOCKED);   
ResetButtonState (Button\_Right, ST\_UNPRESSURE);   
SetButtonState (Button\_Right, ST\_PRESSURE);   
// processing the   
HAL\_Delay button (50);   
ResetButtonState (Button\_Right, ST\_LOCKED);   
main\_menu\_state = MAIN\_MENU\_STATE\_MM1\_WAIT;   
LCD\_SetPos (10.1);   
LCD\_String (str2 [5]); // light the EXIT clause, and then suddenly it is in extinct state   
}   
// left button   
if (button\_state [Button\_Left] & ST\_UNPRESSURE)   
{   
SetButtonState (Button\_Left, ST\_LOCKED);  
ResetButtonState (Button\_Left, ST\_UNPRESSURE);   
SetButtonState (Button\_Left, ST\_PRESSURE);   
// processing the   
HAL\_Delay button (50);   
ResetButtonState (Button\_Left, ST\_LOCKED);   
main\_menu\_state = MAIN\_MENU\_STATE\_MM5\_WAIT;   
LCD\_SetPos (10.1);   
LCD\_String (str2 [5]); // ignite the EXIT clause, and then suddenly it is in an extinct state   
}   
// key up   
if (button\_state [Button\_Up] & ST\_UNPRESSURE)   
{   
SetButtonState (Button\_Up, ST\_LOCKED);   
ResetButtonState (Button\_Up, ST\_UNPRESSURE);   
SetButtonState (Button\_Up, ST\_PRESSURE);   
// processing the   
HAL\_Delay button (50);   
ResetButtonState (Button\_Up, ST\_LOCKED);  
main\_menu\_state = MAIN\_MENU\_STATE\_MM3\_WAIT;   
LCD\_SetPos (10.1);   
LCD\_String (str2 [5]); // lit the EXIT clause, and then suddenly it is in an extinct state   
}   
// SELECT key   
if (button\_state [Button\_Select] & ST\_UNPRESSURE)   
{   
SetButtonState (Button\_Select, ST\_LOCKED);   
ResetButtonState (Button\_Select, ST\_UNPRESSURE);   
SetButtonState (Button\_Select, ST\_PRESSURE);   
// processing the   
HAL\_Delay button (50);   
ResetButtonState (Button\_Select, ST\_LOCKED);   
main\_menu\_state = MAIN\_MENU\_STATE\_EXIT;   
LCD\_SetPos (10.1);   
LCD\_String (str2 [5]); // light the item EXIT, and then suddenly it's in an extinct state   
}**  
break;

We will collect the code, we will sew the controller and check once again the ability to switch the selection of points in a circle in different directions and from top to bottom.

Add a global variable to the menu.c file

static uint8\_t i = 0;   
**char str1 [6];**

Let's write a function for processing the input to a certain menu item

**// ----------------   
// enter any point in the main menu   
void MainMenuItemProcess (uint8\_t ind)   
{   
LCD\_Clear ();   
LCD\_SetPos (0,0);   
LCD\_String ("ITEM");   
sprintf (str1, "% d", ind);   
LCD\_String (str1);   
LCD\_String ("Press");   
LCD\_SetPos (0,1);   
LCD\_String ("SELECT from exit");   
// here are the functions called by menu items   
switch (ind)   
{   
case 1:   
// function-handler 1 of the menu item   
break;   
case 2:   
// function-handler 2 of the menu item   
break;   
case 3:   
// function-handler 3 items of the menu   
break;   
case 4:   
// handler function 4 menu items  
break;   
case 5:   
// function-handler 5 of the menu item   
break;   
}   
while (1)   
{   
if (button\_state [Button\_Select] & ST\_UNPRESSURE)   
{   
LCD\_Clear ();   
SetButtonState (Button\_Select, ST\_LOCKED);   
ResetButtonState (Button\_Select, ST\_UNPRESSURE);   
SetButtonState (Button\_Select, ST\_PRESSURE);   
// processing the   
HAL\_Delay button (50);   
ResetButtonState (Button\_Select, ST\_LOCKED);   
return;   
}   
HAL\_Delay (50);   
}   
}**

Add to it a prototype at the top of the file,

void BlinkItem (uint8\_t ind);   
**void MainMenuItemProcess (unsigned char ind);**

Let's add one more line array

char str1 [6];   
**char str3 [5];**

And at the end of the session add a reaction to the selection of menu items in the function **MainMenuProcess**

    case MAIN\_MENU\_STATE\_MM1: // select item 1   
**main\_menu\_state = MAIN\_MENU\_STATE\_IDLE;   
MainMenuItemProcess (1);**  
**LCD\_Clear ();**  
break;   
case MAIN\_MENU\_STATE\_MM2: // select item 2   
**main\_menu\_state = MAIN\_MENU\_STATE\_IDLE;   
MainMenuItemProcess (2);**  
**LCD\_Clear ();**  
break;   
case MAIN\_MENU\_STATE\_MM3: // select item 3   
**main\_menu\_state = MAIN\_MENU\_STATE\_IDLE;   
MainMenuItemProcess (3);**  
**LCD\_Clear ();**  
break;   
case MAIN\_MENU\_STATE\_MM4: // select item 4   
**main\_menu\_state = MAIN\_MENU\_STATE\_IDLE;   
MainMenuItemProcess (4);**  
**LCD\_Clear ();**  
break;   
case MAIN\_MENU\_STATE\_MM5: // select item 5  
**main\_menu\_state = MAIN\_MENU\_STATE\_IDLE;   
MainMenuItemProcess (5);   
LCD\_Clear ();**  
break;   
case MAIN\_MENU\_STATE\_EXIT: // select the item EXIT   
**menu\_state = MENU\_STATE\_IDLE;   
return;**

  }

When the EXIT button is pressed, it is the **output from the entire function** , and not from the **switch** .

We collect the code, we flash the controller, and if everything works, then I consider the goal of the lesson to be achieved.

And it is already specifically the reaction in the handler of each item, each will be written for itself, depending on which it will solve the tasks, someone will manage something, and someone will write another menu and the entry will be already in it , thanks to the fact that in today's lesson he learned how to do it.